

# VISIT - TECHNISCHE DOKUMENTATION

Tobias Baumgärtner<sup>1</sup> & Emanuel Berndl<sup>2</sup> &  
Robert Kathrein<sup>3</sup> & Kris Raich<sup>3</sup> &  
Florian Schlenker<sup>4</sup>

10. September 2019



## INHALTSVERZEICHNIS

1	Einleitung	7
2	Installationsprozess	8
2.1	Infrastruktur . . . . .	8
2.2	Projekt auf dem LAS installieren . . . . .	8
3	TYPO3	11
3.1	Allgemein . . . . .	11
3.2	Login . . . . .	12
3.3	Aufbau von TYPO3 . . . . .	12
3.4	Konfiguration des Backends mit TYPO3 . . . . .	13
3.5	Anpassung . . . . .	14
3.6	Hinzufügen einer Applikation aus dem App-Bundle . . . . .	17
3.7	Erzeugung des Layouts . . . . .	17
3.8	Das Karten-Plug-In . . . . .	18
3.9	Erstellung eines Templates . . . . .	19
4	Karten-Applikation	20
4.1	Einpflegen der Daten in die Karten-Applikation . . . . .	20
4.2	Erstellung der Startseite für die Karten-Applikation . . . . .	20
4.3	Neues Kartenelement hinzufügen . . . . .	22
4.4	Bearbeitung und Löschen von angelegten Kartenelementen . . . . .	24
5	Glossar-Applikation	25
5.1	Einpflegen der Daten in die Glossar-Applikation . . . . .	25
5.2	Erstellung der Startseite für die Glossar-Applikation . . . . .	25
5.3	Neue Zelle hinzufügen . . . . .	26
5.4	Neuen Event hinzufügen . . . . .	26
5.5	Neuen Insassen hinzufügen . . . . .	27
5.6	Bearbeitung und Löschung von Insassen, Events und Zellen . . . . .	28
6	Galerie-Applikation	28
6.1	Einpflegen der Daten in die Galerie-Applikation . . . . .	29
6.2	Auswahl des gewünschten Layouts für die Startseite . . . . .	30
6.3	Erstellung der Startseite für die Galerie-Applikation . . . . .	30
6.4	Neues Inhaltselement anlegen . . . . .	31
6.5	Sortierung der Inhaltselemente . . . . .	32
6.6	Anlegen eines Sub-Inhaltselements . . . . .	32
6.7	Erstellung der Teaser . . . . .	33
6.8	Sortierung der Teaserelemente . . . . .	33
6.9	Bearbeitung und Löschung Inhaltselementen, Sub-Inhaltselementen sowie Teasern . . . . .	35
7	Fernrohr	36
7.1	Einpflegen der Daten in die Fernrohr-Applikation . . . . .	36
7.2	Erstellung der Startseite für die Fernrohr-Applikation . . . . .	36
7.3	. . . . .	36
8	Dateiverwaltung	36
8.1	Zugangsdaten zum Dateimanagement . . . . .	36
8.2	ViSIT-Partner-Liste . . . . .	38
8.3	Upload von 3D-Objekten und Bildern, Videos und anderen Dateien . . . . .	39
8.4	Hochladen von Dateien . . . . .	40
8.5	Veröffentlichung einer Datei im ViSIT-Netzwerk . . . . .	42
9	Update-Prozess	42
9.1	Update von Programmdateien . . . . .	42

9.2	Datenbank Update . . . . .	42
9.3	Backup Plan . . . . .	43
10	Projektverlauf	44
11	Kompression	45
11.1	Motivation . . . . .	45
11.2	Grundlagen . . . . .	45
11.3	Voraussetzungen . . . . .	49
11.4	Funktionsweise . . . . .	51
11.5	Installation und Steuerung . . . . .	54
11.6	Konfiguration . . . . .	56
11.7	Zugriff über die Web-Oberfläche . . . . .	62
11.8	Zugriff über die API . . . . .	65
12	ViSIT Metadaten und die Semantische Datenbank	71
12.1	Theoretische Grundlagen für die Semantische Datenbank . . . . .	71
12.2	Technische Details zur Semantischen Datenbank . . . . .	76
12.3	WissKI - Wissenschaftliche KommunikationsInfrastruktur . . . . .	79
12.4	Technischer Zugang zu den Metadaten - die ViSIT REST API	85
12.5	Wichtige Technische Charakteristika der Entwicklung und den Betrieb der Semantischen Datenbank . . . . .	87
12.6	Semantische Datenbank - FAQ und häufig auftretende Probleme	88
13	Appendix	93

## ABBILDUNGSVERZEICHNIS

Abbildung 1	Das Login-Fenster für TYPO3 im Browser . . . . .	11
Abbildung 2	Das Login-Fenster für TYPO3 . . . . .	12
Abbildung 3	Aufbau des TYPO3-Backends . . . . .	13
Abbildung 4	Installation der Extensions . . . . .	14
Abbildung 5	Änderung der Sprache . . . . .	15
Abbildung 6	Konfiguration der Benutzereinstellungen . . . . .	15
Abbildung 7	Änderung der Benutzereinstellungen und der Sprache	16
Abbildung 8	Änderung des Passworts . . . . .	16
Abbildung 9	Hinzufügen einer neuen Seite . . . . .	17
Abbildung 10	Erzeugung des Layouts der neu erstellten Seite . . . . .	18
Abbildung 11	Auswahl der Plug-Ins . . . . .	18
Abbildung 12	Einbindung eines Plug-Ins . . . . .	19
Abbildung 13	Leere Kartenübersicht . . . . .	20
Abbildung 14	Erstellung der Startseite für die Karten-Applikation .	20
Abbildung 15	Text für die Startseite der Applikationen, zu finden auf <a href="https://github.com/VisIT-Dev/appbundle">https://github.com/VisIT-Dev/appbundle</a> . . . . .	21
Abbildung 16	Erstellung der Startseite für die Karten-Applikation .	22
Abbildung 17	Ein neues Kartenelement hinzufügen . . . . .	22
Abbildung 18	Listenansicht über alle angelegten Kartenelemente . .	23
Abbildung 19	Startseite der Karten-Applikation . . . . .	23
Abbildung 20	Ansicht der Karte im Browser . . . . .	24
Abbildung 21	Kartenelement - Point of Interest - mit Detailinformation . . . . .	24
Abbildung 22	Ansicht der Glossar-Applikation auf einem Tablet . .	25
Abbildung 23	Übersicht über alle angelegten Insassen . . . . .	26
Abbildung 24	Konfiguration der Glossar-Applikation . . . . .	26
Abbildung 25	Befüllung der Startseite der Glossar-Applikation . .	27
Abbildung 26	Startseite der Glossar-Applikation . . . . .	27

Abbildung 27	Hinzufügen einer neuen Zelle . . . . .	28
Abbildung 28	Erstellen einer neuen Zelle . . . . .	28
Abbildung 29	Angelegte Zellen in der Listenübersicht . . . . .	29
Abbildung 30	Hinzufügen eines neuen Events . . . . .	29
Abbildung 31	Angelegte Events in der Listenansicht . . . . .	30
Abbildung 32	Anlegen eines neuen Insassens . . . . .	30
Abbildung 33	Anlegen eines Insassens . . . . .	31
Abbildung 34	Angelegte Insassen in der Listenansicht . . . . .	31
Abbildung 35	Ansicht der Insassen im Browser . . . . .	32
Abbildung 36	Bearbeitung bzw. Löschung von Insassen/Zellen/Events . . . . .	32
Abbildung 37	Übersicht über alle erstellten Inhalte . . . . .	33
Abbildung 38	Auswahl des gewünschten Layouts in den Einstellungen . . . . .	33
Abbildung 39	3er-Layout (3 Spalten, 1 Zeile) . . . . .	34
Abbildung 40	6er-Layout (3 Spalten, 2 Zeilen) . . . . .	34
Abbildung 41	Startseite der Galerie-Applikation . . . . .	35
Abbildung 42	Neues Inhaltselement anlegen . . . . .	35
Abbildung 43	Ansicht eines Inhaltselements aus der Galerie auf einem Tablet . . . . .	36
Abbildung 44	Anlegen neuer Sub-Inhaltselemente zu einem bestehenden Inhaltselement . . . . .	37
Abbildung 45	Erstellung eines Teasers für die Startseite der Galerie-Applikation . . . . .	37
Abbildung 46	Manuelle Sortierung der Teaserelemente auf der Startseite . . . . .	38
Abbildung 47	Einstellung der ViSIT App Extension . . . . .	38
Abbildung 48	ViSIT App Extensions . . . . .	39
Abbildung 49	Ansicht der ViSIT-Partner mit Zugang zur Medien-datenbank im Peer-to-Peer-Netzwerk . . . . .	39
Abbildung 50	Ansicht der bereits verfügbaren Dateien in der Datei-liste . . . . .	40
Abbildung 51	Ansicht des ausgefüllten Formulars für den Datei-Upload . . . . .	41
Abbildung 52	Bestätigungs-nachrichten nach einem erfolgreichen Upload . . . . .	41
Abbildung 53	Hochgeladene Datei in der Listenübersicht . . . . .	41
Abbildung 54	Scheduler und geplante Tasks . . . . .	43
Abbildung 55	Commits im GitHub Repository . . . . .	43
Abbildung 56	Cache leeren . . . . .	44
Abbildung 57	Visit App-Extension deaktivieren und wieder aktivie-ren . . . . .	44
Abbildung 58	Grundlegende Elemente der Geometrie eines 3D-Modells . . . . .	46
Abbildung 59	Beispiel einer Edge-Collapse-Operation. Die Zielpo-sition der beiden an das Edge angrenzenden Vertices ist <i>rot</i> markiert. . . . .	48
Abbildung 60	Beziehung von mediaTripleID und mediaTripleURL anhand eines Beispiels . . . . .	52
Abbildung 61	Startseite, unter anderem mit einem Überblick über die laufenden und anstehenden Kompressions-Aufträge . . . . .	63
Abbildung 62	Ansicht zum Absetzen eines neuen Kompressions-Auftrags . . . . .	64
Abbildung 63	Archiv-Ansicht mit einem Überblick über die ausge-führten Kompressions-Aufträge . . . . .	65

Abbildung 64	Einstellungsmöglichkeiten über die Web-Oberfläche . . . . .	66
Abbildung 65	Informationen aus obigen Aussagen, kombiniert als Graph. . . . .	72
Abbildung 66	Grundlegende eigene Wissensbasis (oben), erweitert um zwei externe Wissensbasen (unten). . . . .	73
Abbildung 67	Arbeitsprozess hinter der Entwicklung des ViSIT Modells. . . . .	76
Abbildung 68	Technische Infrastruktur der Semantischen Datenbank des ViSIT Projekts. . . . .	77
Abbildung 69	Zwei Beispiel-Pfade aus der WissKI Konfiguration des ViSIT Projekts. . . . .	82
Abbildung 70	Erste Maske zum Editieren eines WissKI Pfads. . . . .	83
Abbildung 71	Zweite Maske zum Editieren eines WissKI Pfads. . . . .	84
Abbildung 72	Beispiel-Pfad aus der WissKI Konfiguration des ViSIT Projekts für eine Relation zwischen zwei Entitäten der Datenbank. . . . .	84
Abbildung 73	Modell der digitalen Repräsentationen. . . . .	85
Abbildung 74	WissKI Pathbuilder Ansicht, die den Download der Pfad-Datei anbietet. . . . .	91
Abbildung 75	pom.xml Konfiguration zum produktiven Deployment der REST API. . . . .	91
Abbildung 76	Einstellungen zum Verbinden des Triplestores. Hier gezeigt: lokale Konfiguration, während die Konfiguration für Produktiv- und Testsystem oben vorhanden aber auskommentiert ist. . . . .	92
Abbildung 77	Übersicht der Konfiguration eines WissKI Salz Adapters, Teil 1. . . . .	105
Abbildung 78	Übersicht der Konfiguration eines WissKI Salz Adapters, Teil 2. . . . .	106
Abbildung 79	Detailansicht einer definierten Ontology im WissKI System am Beispiel VisMo für das ViSIT Projekt. . . . .	107
Abbildung 80	Übersicht der ersten Pfade des für das ViSIT Projekt definierten Pathbuilders. . . . .	108
Abbildung 81	Ausgangs-Interface zum Erzeugen einer Hauptentität im WissKI System. . . . .	109
Abbildung 82	Eingabe-Interface für ein Ausstellungsobjekt im ViSIT WissKI System. . . . .	110
Abbildung 83	Beispiel Ausgabe-Interface einer Partisane des ViSIT WissKI Systems. . . . .	111

## TABELLENVERZEICHNIS

Tabelle 1	Für ein 3D-Modell notwendige und optionale Dateien	51
Tabelle 2	Überblick über alle Konfigurationsmöglichkeiten der Kompressionskomponente . . . . .	57
Tabelle 3	Beispiele für die sich bei unterschiedlichen Vertexanzahlen ergebenden Texturauflösungen bei der Standardkonfiguration. . . . .	60
Tabelle 4	Pro Kompressionsstufe notwendige Parameter bei der Bildkompression . . . . .	61
Tabelle 5	API Funktionen für die Digital Representations. . . . .	86

Tabelle 6	API Funktionen zum Auslesen von Objekten. . . . .	86
-----------	---------------------------------------------------	----

---

<sup>1</sup> TODO, Universität Passau

<sup>2</sup> Lehrstuhl für verteilte Informationssysteme und Data Science, Universität Passau

<sup>3</sup> TODO, FH Kufstein

<sup>4</sup> Forwiss, Universität Passau

## 1 EINLEITUNG

blub hi!

Things to maybe mention here:

- ViSIT Projekt Bitbucket/github, welche wir zur Entwicklung nutzen

## 2 INSTALLATIONSPROZESS

### 2.1 Infrastruktur

Die ViSIT-Applikationen basieren auf der Server-Client-Architektur. Damit diese Applikationen installiert werden können, wird ein hausinternes Netzwerk (Intranet) und ein damit verbundener Server - lokaler Applikations-Server (kurz LAS) - benötigt. Die ViSIT-Applikationen sind in diesem Zusammenhang die Clients, welche über das Netzwerk mit dem lokalen Applikations-Server verbunden sind. Auf dem LAS ist das ViSIT-System installiert, welches über das Internet Zugang zum globalen ViSIT-Netzwerk hat. Das ViSIT-System ist eine Ansammlung von mehreren kleinen Applikationen, welche parallel auf dem LAS laufen können. Jeder Client, auf welchem eine der ViSIT-Applikationen läuft, hat eigene Server-Software, welche auf dem LAS installiert ist und für die serverseitigen Berechnungen zuständig ist.

Die Applikationen wurden mit der IT-Technologie "Docker" erstellt. Mit Docker hat man die Möglichkeit, Anwendungen in sogenannten Containern auszuführen und diese Container können aufeinander aufbauen und miteinander kommunizieren. Im Gegensatz zu einer virtuellen Maschine, ist eine Docker-basierte Anwendung nur ein Prozess, der auf dem System ausgeführt wird. Es ist somit kein Gastbetriebssystem erforderlich, wie dies bei Virtuellen Maschinen der Fall ist. Container sind einfach konfigurierbare, abgeschlossene Einheiten, in welchen die Anwendung ausgeführt werden. Mit Docker können Linux-Container erstellt und verwendet werden können. Die erstellten Container sind eine Virtualisierung auf der Ebene des Betriebssystems. Durch das Erstellen von Containern, werden isolierte Linux-Systeme auf dem gleichen Host erzeugt. Diese Container können flexibel erstellt, bereitgestellt, kopiert und zwischen Umgebungen verschoben werden. Zweck dieser Container ist die Unabhängigkeit und die Fähigkeit, mehrere Prozesse und Applikationen getrennt voneinander betreiben zu können. Die Vorteile von Docker-Containern sind unter anderem Modularität und Versionsverwaltung. Modularität ermöglicht es, bei zum Beispiel einer Reparatur oder Aktualisierung einer Applikation, nur einen Teil dieser Applikation außer Betrieb zu nehmen, ohne die gesamte Applikation außer Betrieb nehmen zu müssen. Docker bietet eine eingebaute Versionsverwaltung, welche es erlaubt, den aktuellen Stand eines Containers in ein sogenanntes Image zu sichern. Somit ist es möglich, die unterschiedlichen Zustände eines Images in einer Historie nachzuverfolgen. Ein Image ist ein Speicherabbild eines Containers und es besteht aus mehreren Layern, welche schreibgeschützt sind und somit nicht verändert werden können. Ein Layer ist wiederum ein Teil eines Images und enthält einen Befehl oder eine Datei, welche dem Image hinzugefügt wurde. Aufgrund dieser Layer kann die ganze Historie eines Images nachvollzogen werden.

### 2.2 Projekt auf dem LAS installieren

Als erster Schritt muss die Datenbank für die Applikation angelegt werden. Wie oben erklärt, wurde für das ViSIT-Projekt Docker verwendet. Damit gespeicherte Daten auch außerhalb eines Containers abgelegt oder in einem anderen Container eingebunden werden können, werden sogenannte Volumes erstellt. Volumes haben viele Vorteile, vor allem aber sind sie einfacher zu sichern oder zu migrieren. Volumes funktionieren sowohl auf Linux- als

auch auf Windows-Containern. Im ersten Schritt wird ein Volume mit der Datenbank auf dem lokalen Rechner im Terminal mit dem Kommando

```
1 | docker volume create visit-database
```

**Listing 1:** Docker Volume erstellen

erstellt. Einen eigenen Volume benötigt man deshalb, weil die dort abgelegten Daten permanent gespeichert werden müssen - würde z.B.: der Container gelöscht oder beendet werden - dann wären die nur im Docker Container gespeicherten Daten ebenfalls gelöscht werden. Damit dies nicht passieren kann, werden die Daten parallel lokal auf dem Rechner gespeichert. Damit Dateien zwischen Geräten in einem lokalen Netzwerk oder zwischen entfernten Geräten über das Internet synchronisiert werden können, wird eine Datensynchronisation mit Peer-to-Peer-Übertragung benötigt. Dies wird im ViSIT-Projekt mit Syncthing realisiert und auch dafür muss ein eigener Volume lokal auf dem Rechner erstellt werden. Dies geschieht mit

```
1 | docker volume create visit-syncthing
```

**Listing 2:** Docker Syncthing Volume erstellen

-Befehl, welcher ebenfalls im Terminal ausgeführt wird. Als nächster Schritt wird das gesamte ViSIT-Projekt von GitHub mittels

```
1 | docker run -d --name visit -p 80:80 -p 22000:22000 -p 21027:21027
2 |   -v visit-syncthing:/var/syncthing
3 |   -v s:/p2p/visit:/var/p2p
4 |   -v visit-database:/var/lib/mysql
5 |   --restart unless-stopped visitapp/maincontainer
```

**Listing 3:** Klonen des gesamten ViSIT-Projekts von GitHub

geklont. Beim erstmaligen Starten benötigt der Vorgang länger, da das Projekt aus dem Git Repository sowie das Appbundle (<https://github.com/ViSIT-Dev/appbundle>) heruntergeladen werden.

Erklärung der einzelnen Befehle:

```
1 | docker run -d --name visit -p 80:80 -p 22000:22000 -p 21027:21027
```

**Listing 4:** Docker run-Befehl

```
1 | docker run
```

**Listing 5:** docker run

startet den Container und mit den mit den Parametern

```
1 | -d
```

**Listing 6:** -d

gibt man an, dass der Container im Hintergrund dauerhaft laufen soll (Daemonmode). Weiters wird mit

```
1 | --name visit
```

**Listing 7:** --name visit

der Name des Containers festgelegt, in diesem Fall heißt der Container visit. Der Container kann im weiteren Verlauf auch über diesen Namen angesprochen werden. Mit dem Parameter

```
1 | -p 80:80
```

**Listing 8:** -p

werden die Ports vom Host an den Container gebunden. Hier wird der lokale Hostport 80 auf den Containerport 80 gemappt. Die weiteren Ports

```
1 | -p 22000:22000 -p 21027:21027
```

**Listing 9:** -p

werden für das Syncthing und für das Peer to Peer-Netzwerk benötigt. Als nächstes folgt der Befehl

```
1 | -v visit-syncthing:/var/syncthing
```

**Listing 10:** -v visit-syncthing

Mit dem Parameter

```
1 | -v
```

**Listing 11:** -v

wird ein Verzeichnis (Volume) auf dem Hostrechner zu einem Verzeichnis innerhalb des Containers verbunden, auf diese Weise werden die Daten persistent gespeichert, das heißt, dass ein Ordner auf dem Hostsystem auf einen Ordner im Container gemappt wird. Das bedeutet, dass die Daten in beiden Ordnern immer inhaltsgleich sind. Ohne dem Mapping zu einen Ordner auf dem Hostsystem, wären alle Daten aus dem Docker Container, wenn dieser Container gelöscht wird, ebenfalls gelöscht. Um die Daten persistent, also dauerhaft zu speichern, wird immer ein Ordner im Hostsystem mit dem entsprechenden Ordner im Docker Container gemappt. Zuerst wird das Verzeichnis auf dem Hostrechner angegeben, hier

```
1 | visit-syncthing
```

**Listing 12:** visit-syncthing

und nach dem Doppelpunkt steht das Verzeichnis innerhalb des Containers, hier

```
1 | /var/syncthing
```

**Listing 13:** Angabe des Verzeichnisses innerhalb des Containers

Im nächsten Teil des Befehls

```
1 | -v s:/p2p/visit:/var/p2p
```

**Listing 14:** Angabe des Verzeichnisses auf dem Hostrechner

wird ebenfalls zuerst das Verzeichnis auf dem Hostrechner angegeben,

```
1 | s:/p2p/visit
```

**Listing 15:** Angabe des Verzeichnisses innerhalb des Hostrechners

und dann das Verzeichnis innerhalb des Containers

```
1 | /var/p2p
```

**Listing 16:** Angabe des Verzeichnisses innerhalb des Containers

Im nächsten Befehl

```
1 | -v visit-database:/var/lib/mysql
```

**Listing 17:** Verbindung zur Datenbank

geht es um die Verbindung zur Datenbank. Hier wird ebenfalls zuerst das Verzeichnis auf dem Hostrechner angegeben

```
1 | visit-database
```

**Listing 18:** Angabe des Verzeichnisses auf dem Hostrechner

und nach dem Doppelpunkt steht das Verzeichnis innerhalb des Containers

```
1 | /var/lib/mysql
```

**Listing 19:** Angabe des Verzeichnisses innerhalb des Containers

Zuletzt wird mittels

```
1 | --restart unless-stopped visitapp/maincontainer
```

**Listing 20:** Automatisches Starten des Docker Containers

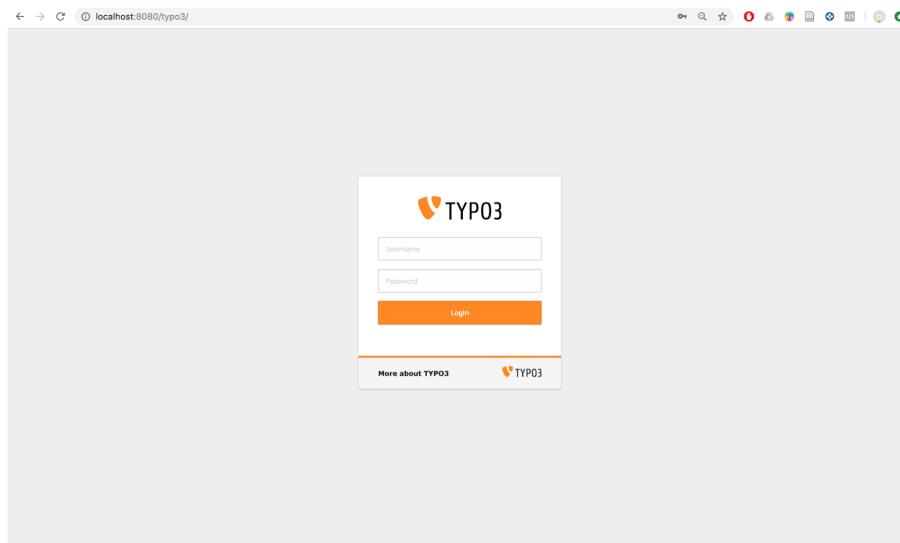
dem System mitgeteilt, dass der Docker Container

```
1 | visitapp/maincontainer
```

**Listing 21:** Angabe des Docker Containers

automatisch gestartet werden soll außer, wenn er manuell oder anderweitig gestoppt wird.

Wenn der Vorgang abgeschlossen ist, kann über Lokalhost im Browser unter `localhost:80/typo3/` das Backend aufgerufen werden (siehe Abbildung 1). Das erstmalige einloggen in das Backend (TYPO3) erfolgt mit dem **Benutzername: admin** und **Passwort: YoGrZOy1og**.



**Abbildung 1:** Das Login-Fenster für TYPO3 im Browser

## 3 TYPO3

### 3.1 Allgemein

TYPO3 ist ein freies Content-Management-System für Webseiten, es wird in Frontend und Backend getrennt. Als Frontend wird die Präsentationsebene bezeichnet, das ist der Teil einer Applikation, den der Betrachter sehen

kann. Als Backend hingegen, bezeichnet man die Datenzugriffsebene, das ist der Teil einer Applikation, welcher nicht für den Besucher sichtbar ist. Das Backend ist der Verwaltungsbereich einer Webseite. TYPO3 wird auf einem Webserver installiert und über den Webbrower benutzt.

Das Backend ist die Datenzugriffsebene, dieser Teil ist für den Endbenutzer nicht sichtbar. Es beinhaltet die Programmierung einer Applikation und den Administrationsbereich. Im Gegensatz dazu das Frontend, das ist die tatsächliche Webseite, die der Endbenutzer im Browser sieht, also die Benutzeroberfläche.

### 3.2 Login

Damit niemand unbefugter im Frontend sowie Backend etwas verändern kann, muss man sich zuerst ins Backend einloggen. Dies geschieht über den Aufruf der Domain **localhost:80/typo3/** im Webbrower (siehe Abbildung 1).

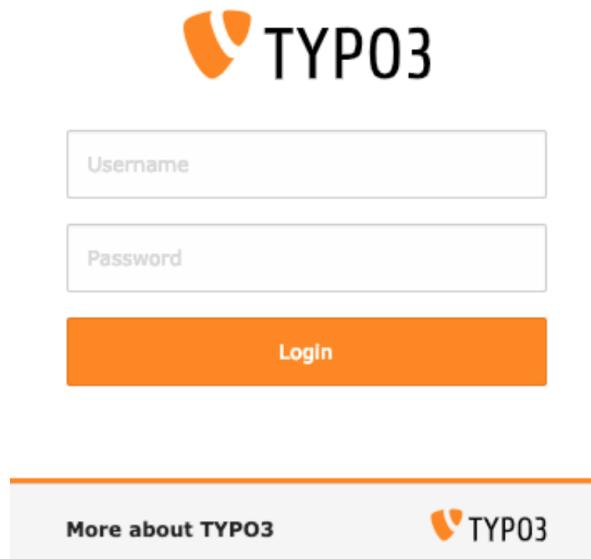


Abbildung 2: Das Login-Fenster für TYPO3

Im Login-Fenster kann der Benutzername sowie das Passwort eingetragen werden (siehe Abbildung 2). Beim ersten Login ist der **Benutzername: admin** und das **Passwort: visit-admin**, dieser muss in weiterer Folge verändert werden. Mehr dazu siehe Anpassung. Nach einem erfolgreichen Login wird das Backend mit den dazugehörigen Modulen im Browser geladen.

### 3.3 Aufbau von TYPO3

Das TYPO3-Backend besteht aus einem Kopfbereich (grün eingerahmt) und einem Hauptbereich (rot eingerahmt), welcher aus drei Spalten besteht (siehe Abbildung 3). Im Kopfbereich kann der Administrator seine TYPO3-Benutzerstellenungen konfigurieren. Im Hauptbereich werden Webdokumente bearbeitet. Das TYPO3-Backend wird von links nach rechts abgearbeitet.

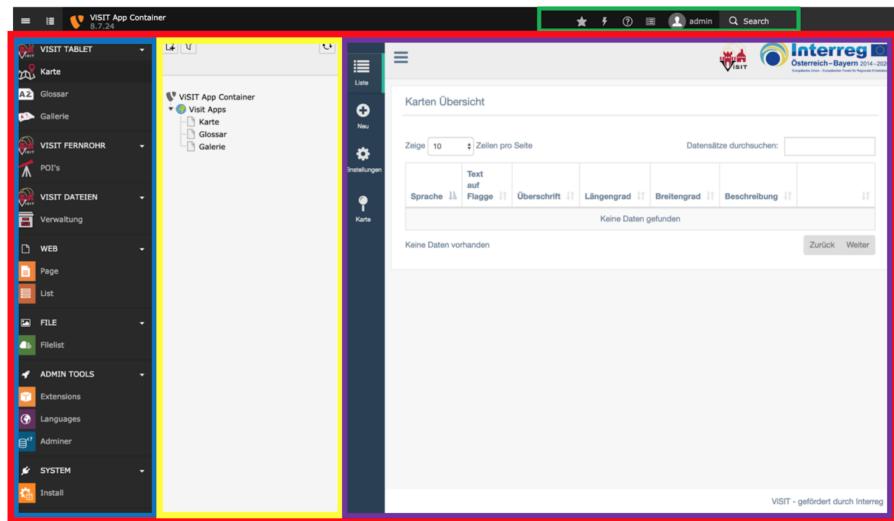


Abbildung 3: Aufbau des TYPO3-Backends

### 3.3.1 Kopfleiste

Die Kopfleiste bietet die Möglichkeit, die im TYPO3 Backend gespeicherten Lesezeichen aufzurufen (Stern-Symbol), den TYPO3 Cache der gesamten Webseite zu leeren (Blitz-Symbol) sowie Hilfe und Dokumentationen (Fragezeichen) zu TYPO3 aufzurufen. Das vierte Symbol zeigt die wichtigsten Systeminformationen. Mit einem Klick auf den Benutzernamen, in der Grafik "admin", öffnet sich ein Kontext-Menü mit der Möglichkeit Einstellungen an seinem Benutzer vorzunehmen oder sich aus dem TYPO3 Backend auszuloggen. Rechts neben dem Benutzer befindet sich das Suchfeld, mit dem sich das gesamte TYPO3 Backend durchsuchen lässt.

### 3.3.2 Die Spalten des Hauptbereichs

**Linke Spalte:** *Modulleiste* (blau eingerahmt), hier kann das Modul ausgewählt werden, welches bearbeitet werden soll (siehe Abbildung 3).

**Mittlere Spalte:** *Seitenbaum* (gelb eingerahmt), hier wird die zu bearbeitende TYPO3-Seite ausgewählt. Der Seitenbaum ist das zentrale Element, wenn es darum geht sich durch die Webseite zu navigieren. Hier wird der Aufbau und die Seitenhierarchie der Webseite in einer Struktur abgebildet, die der Ordnerstruktur ähnlich ist. Einzelne Seiten können Unterseiten enthalten, die im Seitenbaum eingerückt dargestellt werden (siehe Abbildung 3).

**Rechte Spalte:** *Arbeitsbereich* (violett eingerahmt), hier wird am ausgewählten TYPO3-Objekt gearbeitet (siehe Abbildung 3).

## 3.4 Konfiguration des Backends mit TYPO3

Die für die Applikationen benötigten TYPO3 Extensions werden automatisch installiert, sollte eine weitere Extension benötigt werden, befindet sich eine Anleitung für die Installation in diesem Abschnitt. Extensions sind optionale Software-Komponenten, also Zusatzmodule, die eine bestehende Software erweitern.

Installation von TYPO3 Extensions: Dazu wird in der linken Spalte zuerst das Modul “Extensions” ausgewählt. Dann erscheinen im Hauptfenster verschiedene Extensions, welche alphabetisch gelistet sind. Bei der Erstinstallation werden folgende Extensions (unten ist der Key angegeben, welcher sich in der mittleren Spalte befindet) automatisch installiert (siehe Abbildung 48):

- visit\_tablets
- scheduler
- tstemplate
- fluid\_styled\_content
- setup

Upd.	A/D	Extension	Key	Version	State	Type	Actions
		Help>About	about	8.7.24	stable	System	
		TYPO3 Backend	backend	8.7.24	stable	System	
		Tools>Log	belog	8.7.24	stable	System	
		Backend User Administration	beuser	8.7.24	stable	System	
		Context Sensitive Help	context_help	8.7.24	stable	System	
		TYPO3 Core	core	8.7.24	stable	System	
		Help>TYPO3 Manual	cshmanual	8.7.24	stable	System	
		CSS styled content	css_styled_content	8.7.24	deprecated	System	
		Documentation	documentation	8.7.24	stable	System	
		Extbase Framework for Extensions	extbase	8.7.24	stable	System	
		Extension Manager	extensionmanager	8.7.24	stable	System	
		Frontend Editing	feedit	8.7.24	stable	System	
		Frontend Login for Website Users	felogin	8.7.24	stable	System	
		File>List	filelist	8.7.24	stable	System	
		Advanced file metadata	filenmetadata	8.7.24	stable	System	

Abbildung 4: Installation der Extensions

Mittels einem Klick auf das Würfelsymbol mit einem Plus werden die oben angegebenen Extensions der Reihe nach aktiviert (siehe Abbildung 48). Die aktivierten Extensions erscheinen dann als auswählbare Module in der linken Spalte. Optional kann im nächsten Schritt die Sprache Deutsch installiert werden, sonst ist die Hauptsprache Englisch. Um die Sprache zu installieren, wird in der linken Spalte unter den ADMIN TOOLS “Languages” ausgewählt (siehe Abbildung 5).

Im Hauptfenster erscheinen nach dem Klick die unterstützten Sprachen, hier “German” suchen und zuerst mittels einem Klick auf das Plus-Symbol links von der Sprache die Sprache aktivieren, dabei erscheint oben rechts eine grüne Meldung mit “Success, language was successfully activated.”. Als nächstes muss die aktivierte Sprache mittels Klick auf das Download-Symbol rechts von der Sprache heruntergeladen werden. War der Download erfolgreich, so erscheint oben rechts eine grüne Meldung mit “Success. The translation update has been successfully completed.”.

### 3.5 Anpassung

Im Kopfbereich können die TYPO3-Benutzereinstellungen konfiguriert werden. Dazu wird im Kopfbereich oben rechts zuerst der Benutzer ausgewählt.

A/D	Language	Locale	Last update	Actions
	Afrikaans	af		
	Albanian	sq		
	Arabic	ar		
	Bahasa Malaysia	ms		
	Basque	eu		
	Bosnian	bs		
	Brazilian Portuguese	pt_BR		
	Bulgarian	bg		
	Catalan	ca		
	Chinese (Simp.)	ch		
	Chinese (Trad.)	zh		
	Croatian	hr		
	Czech	cs		
	Danish	da		
	Dutch	nl		
	English (US)	en		
	French	fr		
	German	de		
	Greek	el		
	Hungarian	hu		
	Icelandic	is		
	Indonesian	id		
	Irish	ga		
	Italian	it		
	Korean	ko		
	Lithuanian	lt		
	Macedonian	mk		
	Norwegian	no		
	Polish	pl		
	Romanian	ro		
	Serbian	sr		
	Slovenian	sl		
	Slovak	sk		
	Slovene	sv		
	Turkish	tr		
	Vietnamese	vi		
	Welsh	cy		
	Zulu	zu		

Abbildung 5: Änderung der Sprache

Bei der Erstinstallation ist es der "admin", dabei wird ein Menü aufgeklappt, aus welchem die "User settings" ausgewählt werden (siehe Abbildung 6).

Abbildung 6: Konfiguration der Benutzereinstellungen

Jetzt erscheinen im Hauptbereich die User Settings, welche in dieser Maske konfiguriert werden können. Jetzt kann zuerst die Sprache umgestellt werden. Dies kann gleich im ersten Raster "Personal data", im unteren Bereich unter Languages geändert werden (siehe Abbildung 7). Hier kann die heruntergeladene Sprache mittels Dropdown ausgewählt werden. Damit die Auswahl auch gespeichert und angewendet wird, muss auf das Speicher-Symbol (Diskette) ganz oben links im Hauptfenster geklickt werden. Nur durch diesen Klick werden die User Settings upgedated und die Sprache auch angewendet. Jetzt erscheinen oben im Hauptfenster drei Meldungen. Die grüne Meldung besagt, dass die Settings upgedated wurden. Die blaue Meldung sagt, dass die Seite (localhost:80/typo3/) neu geladen werden muss, um die Veränderungen zu aktivieren. Die rote Meldung sagt, dass

das neue Passwort nicht upgedated wurde, da es nicht zweimal eingegeben wurde.

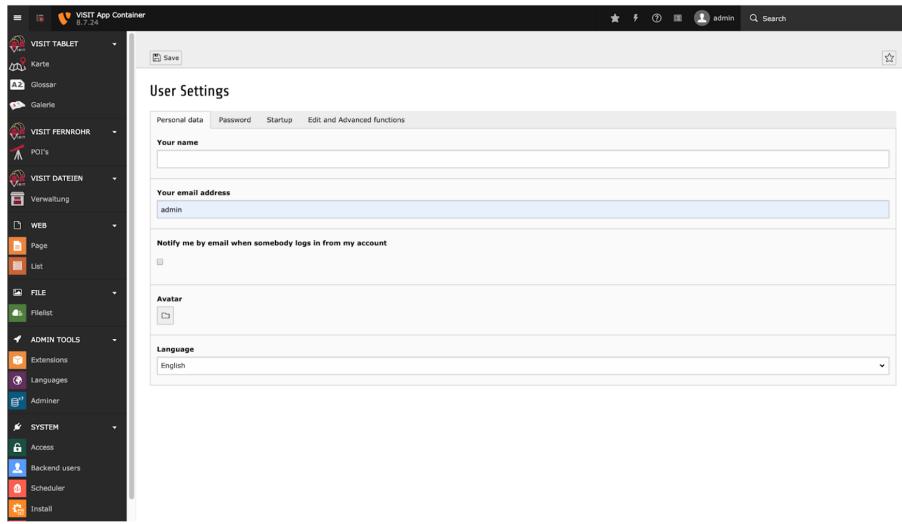


Abbildung 7: Änderung der Benutzereinstellungen und der Sprache

Als nächstes wird das Passwort verändert. Dazu wird die Registerkarte "Password" ausgewählt (siehe Abbildung 8). Jetzt erscheint das zuvor eingegebene Passwort "visit-admin" als eine Punkte-Kette in der ersten Zeile, hier kann das Passwort mit einem neuen Passwort überschrieben werden. Gleicher Passwort wird in der darunter liegenden Zeile nochmals eingegeben. Damit die Änderungen gespeichert werden, wird wieder oben links das Speichern-Symbol geklickt. Ab jetzt werden auch die Änderungen der Sprache angewendet und alles wird auf Deutsch angezeigt. Mit diesem Schritt ist das Backend fertig vorbereitet.

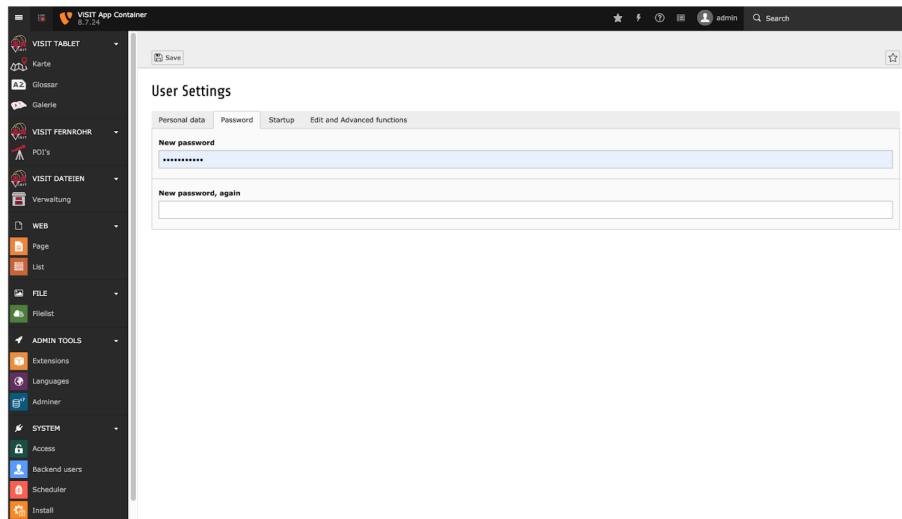


Abbildung 8: Änderung des Passworts

### 3.6 Hinzufügen einer Applikation aus dem App-Bundle

Dazu wird in der linken Spalte "Seite" ausgewählt. Jetzt kann dem ViSIT App Container eine Seite hinzugefügt werden. Zuerst muss auf das oben ganz links befindlichen Seiten-Symbol geklickt werden, dann erscheint eine Auswahl an möglichen Aktionen. Hier das erste leere Seite-Symbol anklicken und auf den darunter befindlichen ViSIT App Container ziehen und darüber loslassen, anschließend kann der Seite ein Name gegeben werden (siehe Abbildung 9).

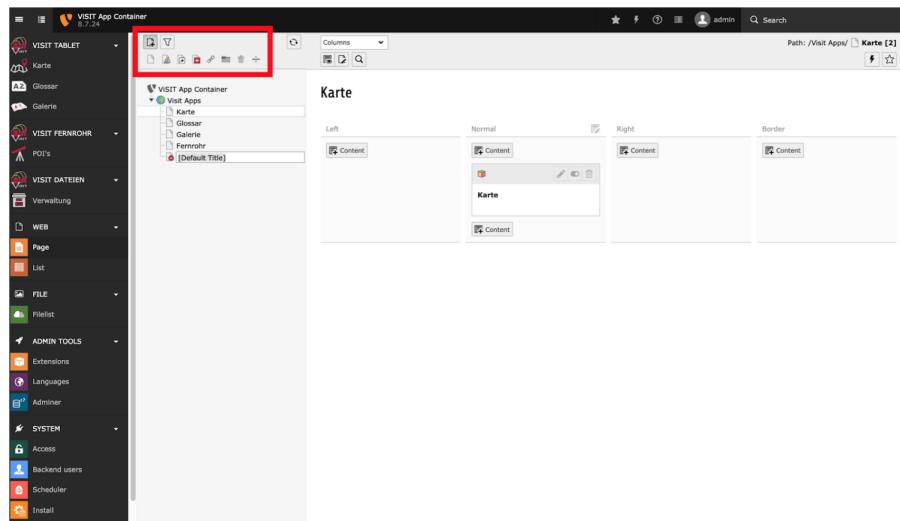


Abbildung 9: Hinzufügen einer neuen Seite

Mittels Rechtsklick auf die soeben erstellte Seite erscheint unter der Seite ein weiteres Menü, aus diesem dann "Bearbeiten" auswählen. Danach kann rechts die Seite konfiguriert werden.

Im nächsten Schritt muss das Verhalten der Seite konfiguriert werden. Dazu den Raster "Verhalten" anklicken und unter "Sonstige" "Als Anfang der Website benutzen" aktivieren. Dann den Raster "Zugriff" auswählen und unter "Sichtbarkeit" "Seite" deaktivieren. Nachdem die Änderungen durchgeführt wurden, müssen diese gespeichert werden. Dazu muss auf das Speicher-Symbol oben auf der Hauptseite geklickt werden. Danach erscheint ein Weltkugel-Symbol neben der soeben erzeugten Seite im linken Teil des Hauptfensters.

### 3.7 Erzeugung des Layouts

Um das Layout der Seite zu definieren, muss auf die soeben erzeugte Seite geklickt werden (siehe Abbildung 10).

Im rechten Teil des Hauptfensters erscheinen vier Möglichkeiten der Inhaltspositionierung. Für die ViSIT-Applikationen wird die normale Inhaltspositionierung benötigt. Um weitere Konfiguration durchzuführen, unter "Normal" auf das das Inhalts-Symbol klicken und im Raster "Plug-Ins" auswählen, hier können die Plugins für die jeweilige ViSIT-Applikation ausgewählt werden (siehe Abbildung 11).

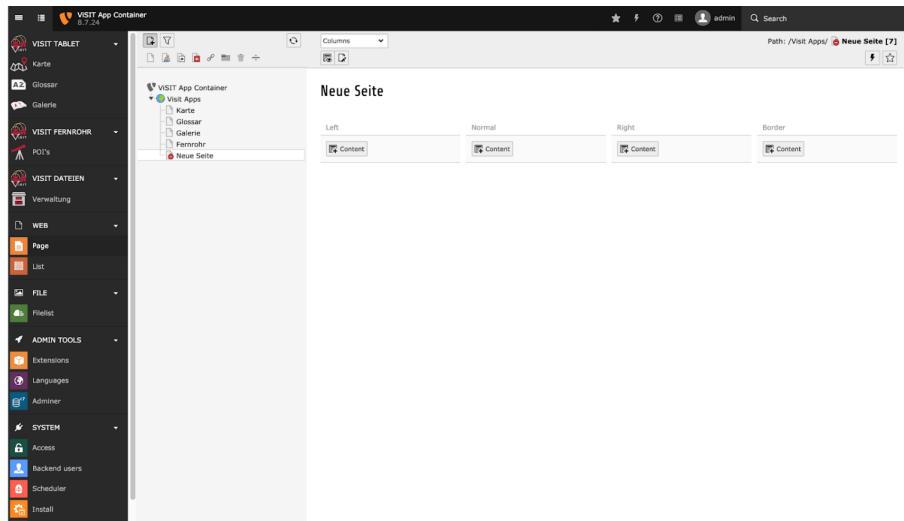


Abbildung 10: Erzeugung des Layouts der neu erstellten Seite

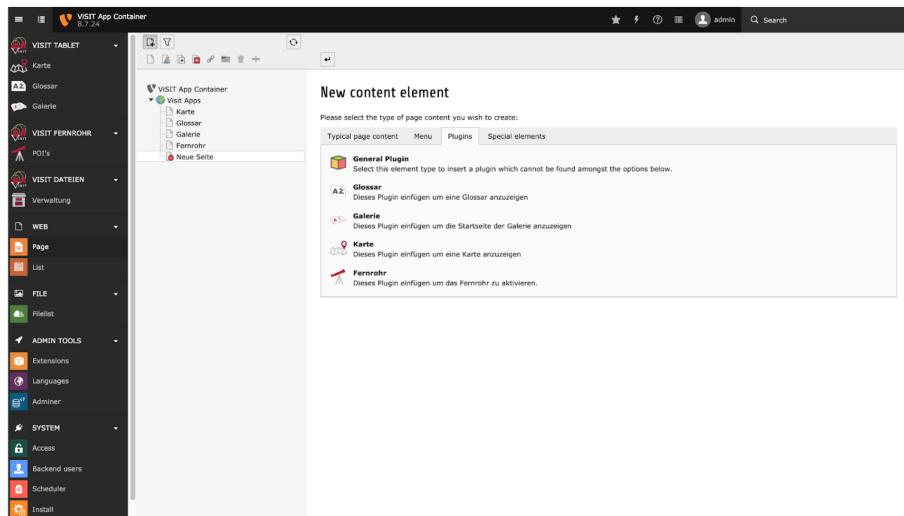


Abbildung 11: Auswahl der Plug-Ins

### 3.8 Das Karten-Plug-In

Im Raster “Plug-Ins” die “Karte - Dieses Plugin einfügen um eine Karte anzuzeigen” auswählen und oben auf das Speicher-Symbol klicken, damit die Änderungen gespeichert werden. Nach dem Speichern kann die Seite mit dem X-Symbol über der Überschrift geschlossen werden. Danach erscheint die Übersicht über die erzeugte Seite, hier sieht man, dass das Karten-Plugin eingebunden wurde (siehe Abbildung 12).

Wenn jetzt die soeben erstellte Seite in der linken Spalte des Hauptfensters, also da wo die Weltkugel ist, mit Rechtsklick ausgewählt, kommt ein Dropdown-Menü. Jetzt den ersten Eintrag “Ansehen” aus der Liste auswählen und die Seite kann im Browser angesehen werden.

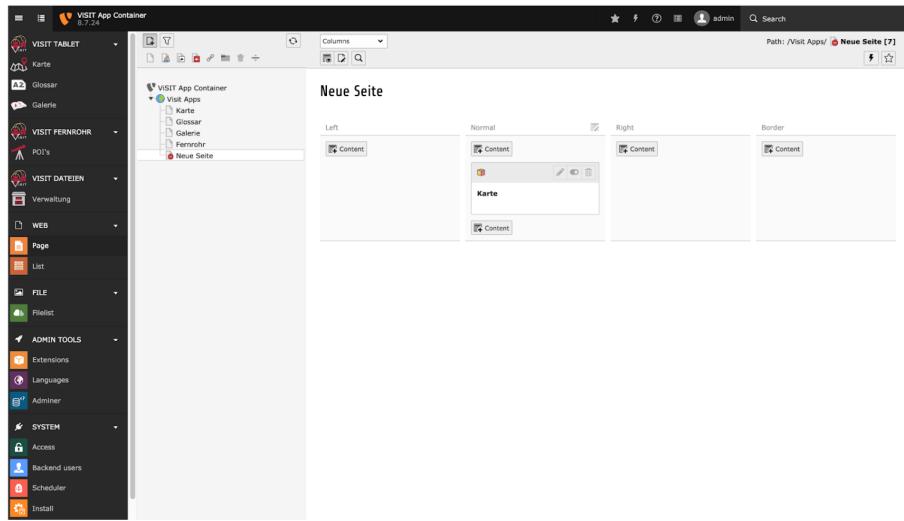


Abbildung 12: Einbindung eines Plug-Ins

### 3.9 Erstellung eines Templates

Wenn ein neuer Raum hinzugefügt wird, wird ein neuer Webroot benötigt, dieser wird mittels Template erzeugt und stellt den Seitenanfang der Webseite dar. Sollen mehrere gleiche Applikationen laufen, dann wird für jede einzelne Applikation ein eigenes Template benötigt. Für die Darstellung der Inhalte auf der Webseite werden Templates verwendet. Ein Template ist eine Design- und Formatierungsvorlage für ein Dokument, es ist das Grundgerüst, welches mit Inhalten gefüllt werden muss //. Um ein Template in TYPO3 zu erstellen, muss im ersten Schritt unter WEB das "Template" aus der Modul-Liste auf der linken Seite ausgewählt werden. Danach erscheinen die Template-Werkzeuge in der rechten Hälfte des Hauptfensters, hier kann "Template für neue Website erstellen" ausgewählt werden. Jetzt kann in der Werkzeugsleiste des Hauptbereichs das Dropdown-Feld aufgemacht und "Info/Bearbeiten" ausgewählt werden. In der Übersicht im Hauptbereich erscheinen die wichtigsten Template-Informationen. Danach "Vollständigen Template-Datensatz bearbeiten" auswählen. Hier kann im Raster "Allgemeines" der Titel des Templates hinzugefügt werden, des weiteren muss der Inhalt aus "Setup" gelöscht werden.

Danach ins Raster "Enthält" wechseln, hier können verschiedene Objekte aus der rechten Spalte "Verfügbare Objekte" in die linke Spalte "Ausgewählte Objekte" verschoben werden, hier muss jedoch auf die Reihenfolge dieser Objekte geachtet werden. Hier zuerst auf "Fluid Content Elements (fluid\_styled\_content)" klicken, dann wandert dieses Objekt in die linke Spalte. Das gleiche mit dem Objekt "tablets (visit\_tablets)". Jetzt befinden sich beide Objekte in der linken Spalte unter "Ausgewählte Objekte". Damit diese Änderungen gespeichert werden, muss wieder auf das Speichern-Symbol über der Überschrift im Hauptbereich geklickt werden. Wenn die Webseite auf dem localhost:80/ aufgerufen wird, erscheint die Karte.

## 4 KARTEN-APPLIKATION

### 4.1 Einpflegen der Daten in die Karten-Applikation

Dazu aus der Modulleiste links unter der Obergruppe VISIT TABLET die Karte auswählen. Im linken Teil des Hauptfensters ist der Seitenbaum zu sehen und rechts befindet sich die Kartenübersicht. Oben links im rechten Teil des Hauptfensters befindet sich ein Menü-Button, wird dieser angeklickt, wird eine weitere dunkelblaue Spalte zwischen dem Seitenbaum und dem Arbeitsbereich im Hauptfenster sichtbar (siehe Abbildung 13).

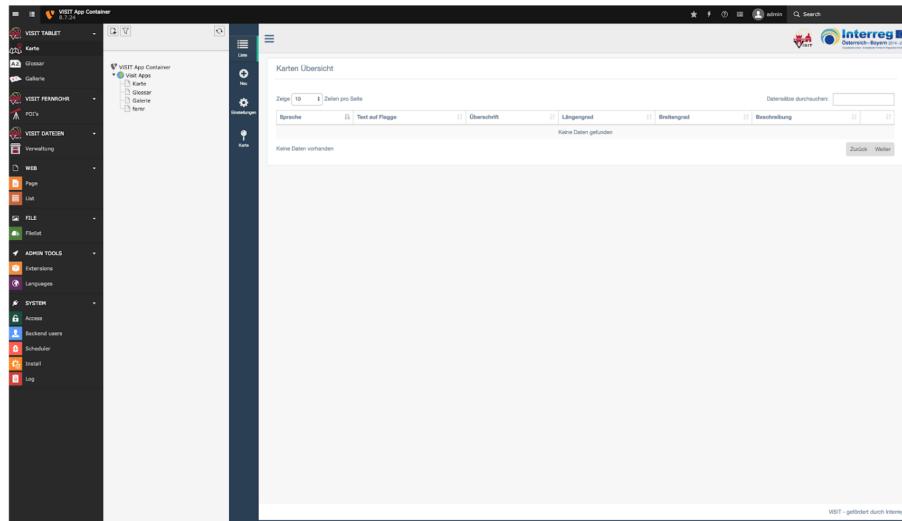


Abbildung 13: Leere Kartenübersicht

### 4.2 Erstellung der Startseite für die Karten-Applikation

Dazu in der dunkelblauen Leiste "Einstellungen" auswählen (siehe Abbildung 14).

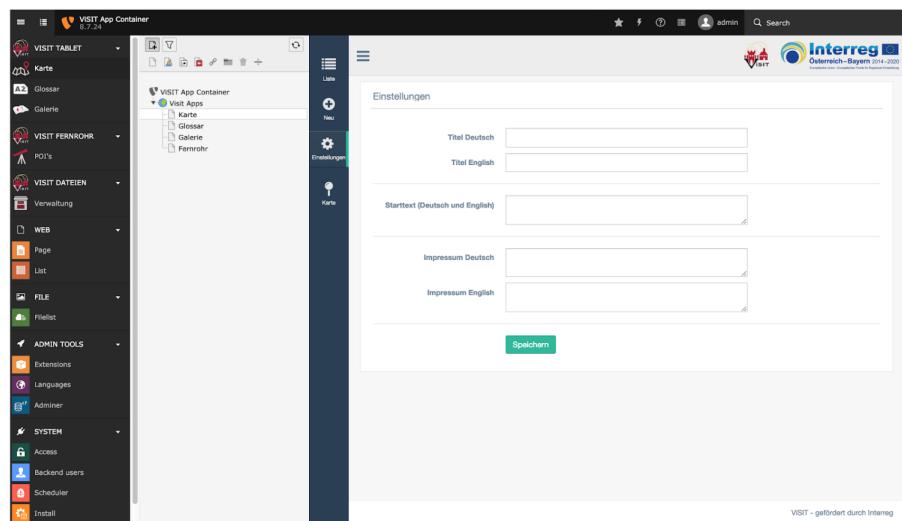


Abbildung 14: Erstellung der Startseite für die Karten-Applikation

Die Startseite wird dem Besucher als erstes angezeigt, auf dieser kann der Besucher die gewünschte Sprache auswählen. Damit das Design des Textes immer gleich aussieht, gibt es unter <https://github.com/ViSIT-Dev/appbundle> in der README.md ein Beispiel für die Startseite der Tablets (siehe Abbildung 15).

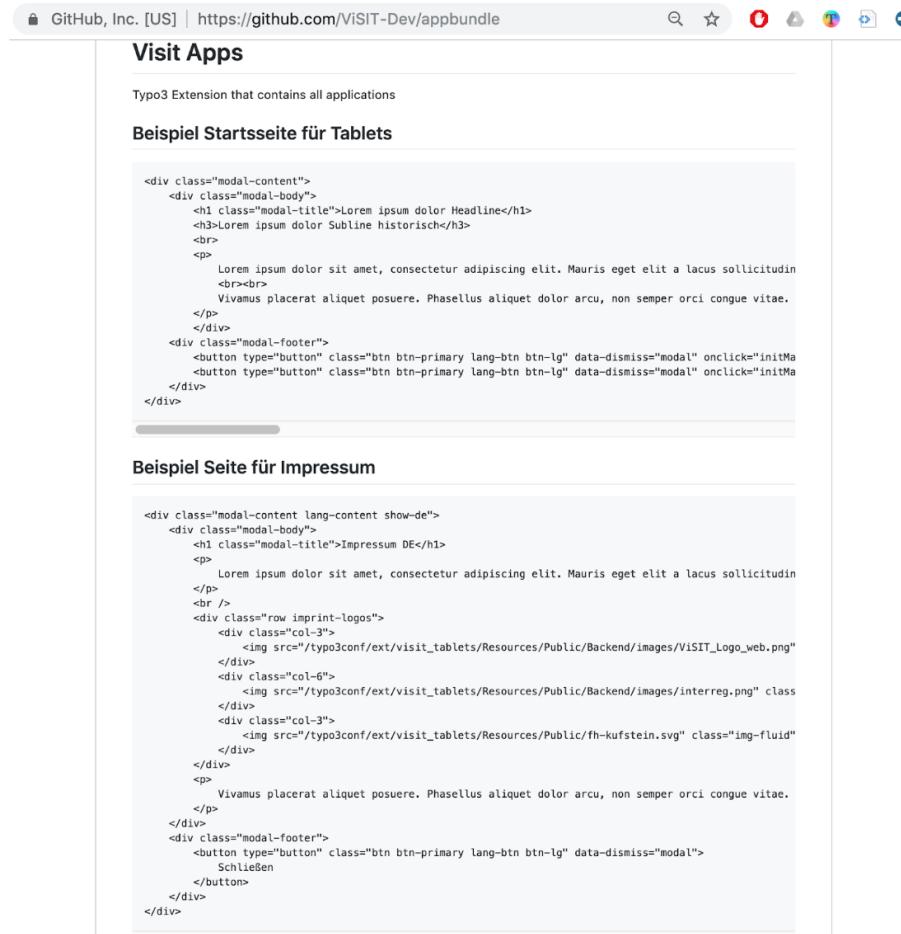


Abbildung 15: Text für die Startseite der Applikationen, zu finden auf <https://github.com/ViSIT-Dev/appbundle>

Für jede Sprache wird ein Titel sowie der Impressumstext benötigt. Jetzt werden die beiden Texte aus der zuvor genannten Github-Seite benötigt (siehe Abbildung 15). Der erste Text ist der Starttext, dieser beinhaltet die HTML-Elemente Überschrift, Paragraph und Buttons über welche die gewünschte Sprache gewählt werden kann. Die einzelnen Texte in den Tags können mit dem gewünschten Text überschrieben werden (siehe Abbildung 16).

Wenn weitere Sprachen außer Deutsch und Englisch verfügbar sind, können weitere Sprachauswahl-Buttons durch das Markieren des gesamten <button>-Tags ausgewählt werden, dann kopieren und darunter einfügen, erstellt werden. Zwei Sachen müssen beachtet werden: einerseits muss in der `on-click='initMap(...)'-Methode` die der Sprache entsprechende ID eingegeben werden und für den Button der Pfad für die Flagge im Image-Tag angegeben werden. Dazu muss zuvor im das benötigte Flaggen-Icon vorzugsweise im PNG-Format im entsprechenden Ordner gespeichert werden und der

Pfad angepasst werden  
 src="/typo3/sysext/core/Resources/Public/Icons/Flags/PNG/DE.png».

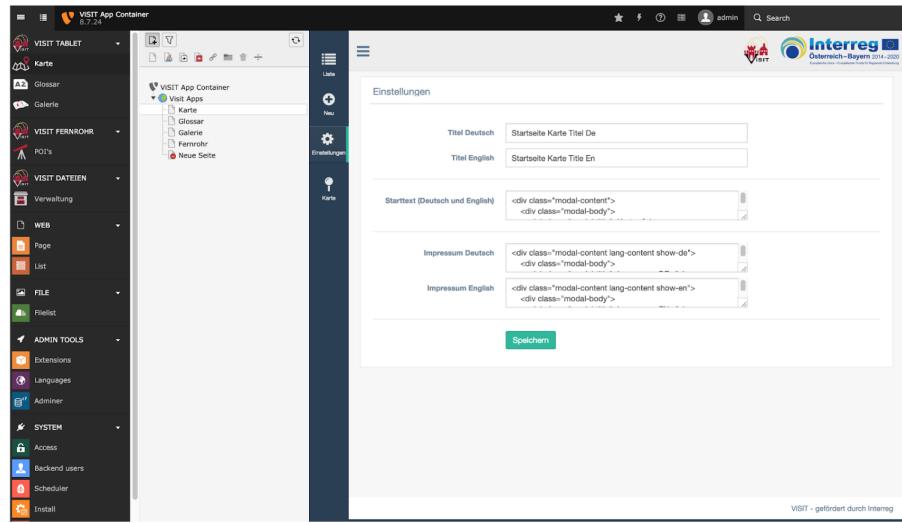


Abbildung 16: Erstellung der Startseite für die Karten-Applikation

#### 4.3 Neues Kartenelement hinzufügen

Mit einem Klick auf "Neu" kann ein neues Kartenelement - Point of Interest - hinzugefügt werden (siehe Abbildung 17).

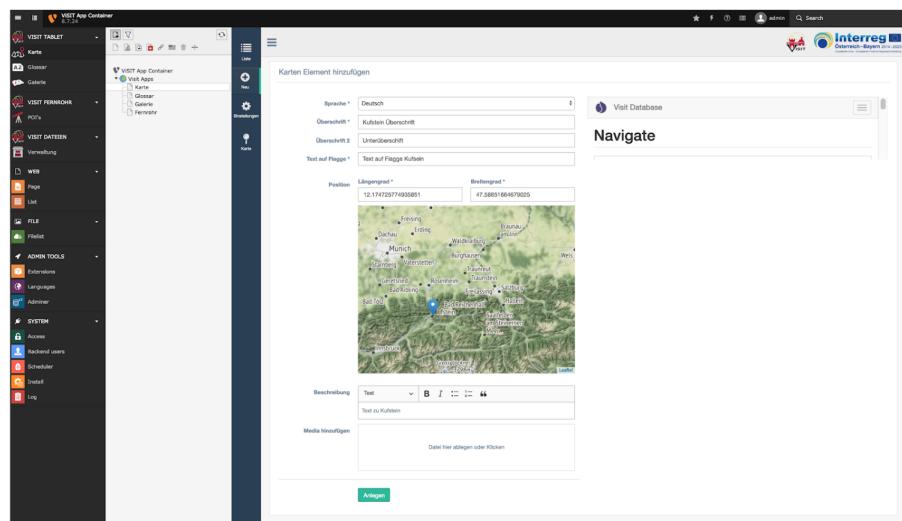


Abbildung 17: Ein neues Kartenelement hinzufügen

Ein neues Kartenelement - Point of Interest - benötigt eine Überschrift, eine Unterüberschrift ist optional, einen Text auf der Flagge und eine Beschreibung. Optional können auch weitere Medien hinzugefügt werden. Die geografische Position kann entweder über den Längen- und Breitengrad manuell eingetippt werden oder mittels setzen der Stecknadel auf die Karte, dann werden die Längen- und Breitengrade dieser Stecknadel übernommen. Ist alles vollständig ausgefüllt, kann die Eingabe mit "Anlegen" am Seitenende gespeichert werden. Nach dem Klick gelangt man zu der Kartenübersicht,

wo alle eingefügten Elemente angeführt sind, jedes dieser Elemente kann sowohl nochmals bearbeitet oder auch wieder gelöscht werden.  
 Klickt man im Seitenbaum mit der rechten Maustaste auf Karte, dann kann man die angelegten Kartenelemente im Browser anzeigen lassen.  
 Jedes Kartenelement muss sowohl auf Deutsch als auch auf Englisch angelegt werden (siehe Abbildung 18).

Sprache	Text auf Flagge	Überschrift	Längengrad	Breitengrad	Beschreibung
Deutsch	Text auf Flagge Kufstein	Kufstein Überschrift	12.174725774936	47.58851664679	Text zu Kufstein
Deutsch	Text auf Flagge Rosenheim	Rosenheim	12.119800071427	47.861958171191	Text zu Rosenheim
English	Text auf Flagge Saalfelden	Saalfelden EN	12.844873537072	47.41409818047	Text zu Saalfelden
English	Text auf Flagge Kufstein	Kufstein EN	12.169236878178	47.588516642268	Text zu Kufstein
English	Text auf Flagge Rosenheim	Rosenheim	12.119800071427	47.858273217776	Text zu Rosenheim

Abbildung 18: Listenansicht über alle angelegten Kartenelemente

Dabei wird zuerst die Startseite angezeigt, auf welcher der Besucher seine Sprache wählen kann (siehe Abbildung 19).

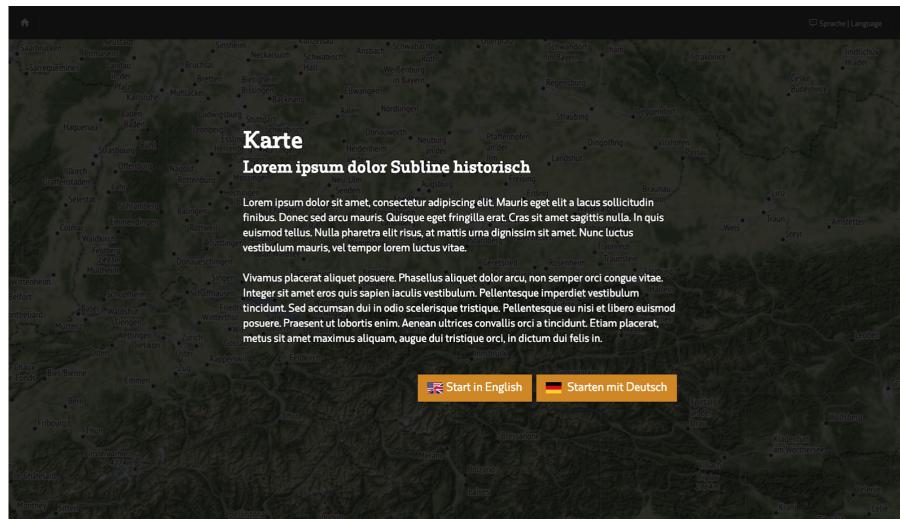


Abbildung 19: Startseite der Karten-Applikation

Nach der Auswahl der Sprache wird dem Besucher die Karte mit den einzelnen Kartenelementen - Point of Interest - auf dem Tablet angezeigt (siehe Abbildung 20).

Jetzt kann der Besucher eine Flagge auswählen, zu welcher er mehr Informationen haben möchte und via Klick öffnet sich der seitliche Infobereich auf der rechten Seite (siehe Abbildung 21).

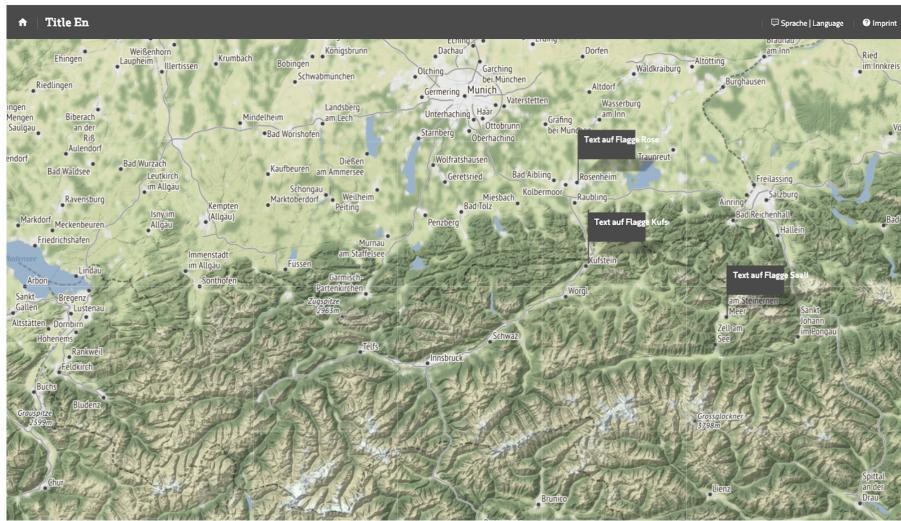


Abbildung 20: Ansicht der Karte im Browser

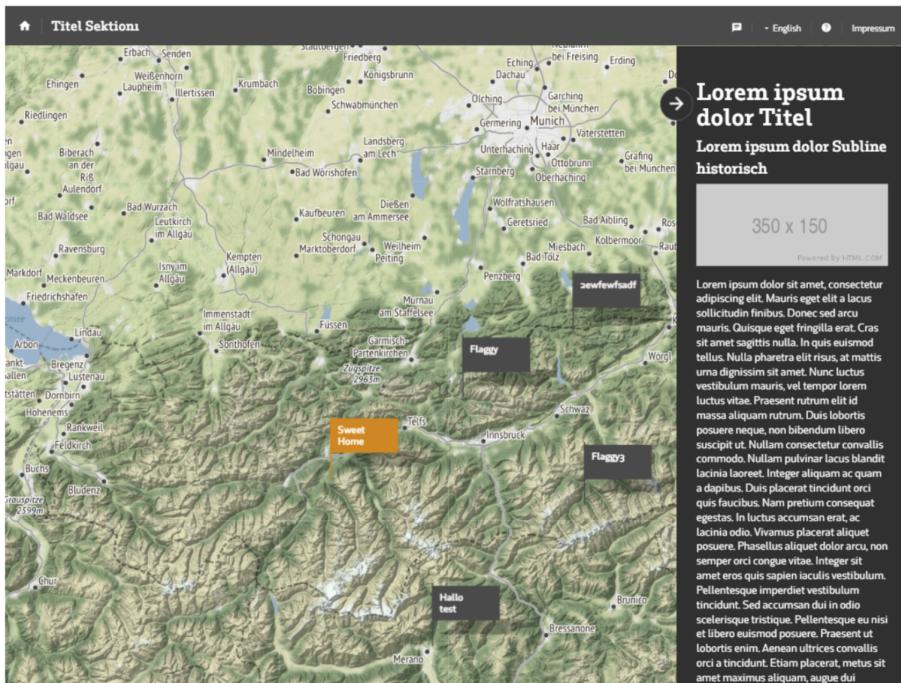


Abbildung 21: Kartenelement - Point of Interest - mit Detailinformation

#### 4.4 Bearbeitung und Löschen von angelegten Kartenelementen

Die Kartenelemente können jederzeit bearbeitet oder gelöscht werden. Dies geht indem zuerst die Listenansicht in der dunkelblauen Leiste ausgewählt wird. In weiterer Folge kann jedes einzelne Element (Zeile) einzeln bearbeitet oder gelöscht werden. Zum Bearbeiten auf das blaue Stiftsymbol auf der rechten Seite klicken, zum Löschen des Objekts, den orangen Müllkübel.

## 5 GLOSSAR-APPLIKATION

In der Glossar-Applikation werden Insassen des Gefängnisses aufgelistet. Die Applikation ist in zwei Bereiche aufgeteilt, rechts werden die Insassen aufgelistet, wählt der Benutzer einen Namen aus dieser Liste aus, so werden die Details zu dieser Person in der rechten Spalte angezeigt. Die Auflistung der Insassen ist alphabetisch nach Vornamen beziehungsweise, wenn vorhanden, nach dem Nachnamen [22](#). Die Auflistung kann auch nach Ereignis, Zelle oder VIP erfolgen, dies kann der Besucher in der oberen Menüzeile auswählen.

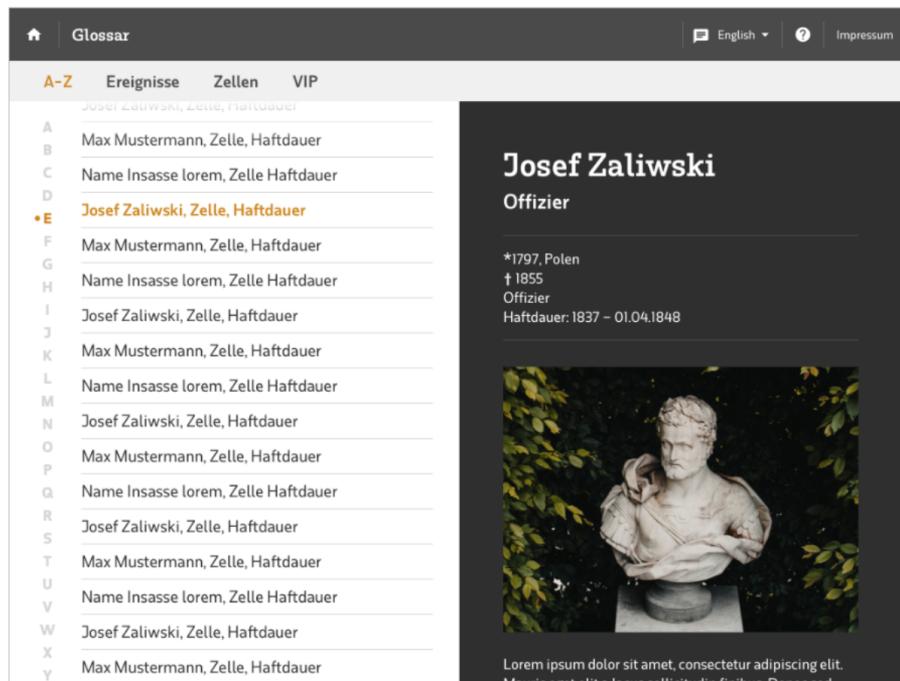


Abbildung 22: Ansicht der Glossar-Applikation auf einem Tablet

### 5.1 Einpflegen der Daten in die Glossar-Applikation

Dazu aus der Modulleiste links unter der Obergruppe VISIT TABLET das Glossar auswählen. Im linken Teil des Hauptfensters ist der Seitenbaum zu sehen und rechts befindet sich die Insassen-Übersicht [23](#).

### 5.2 Erstellung der Startseite für die Glossar-Applikation

Das Glossar kann mit einem Klick auf Glossar in der Modulleiste konfiguriert werden [24](#).

Die Startseite der Applikation kann unter Einstellungen in der dunkelblauen Leiste konfiguriert werden. Dafür benötigt man den Text für die Startseite <https://github.com/VisIT-Dev/appbundle>. Dieser Text muss in die entsprechenden Inputfelder kopiert werden [25](#).

Nachdem die Startseite befüllt wurde, wird sie dem Besucher als erstes angezeigt, auf dieser kann der Besucher dann die gewünschte Sprache auswählen [26](#).

Sprache	Name	Geburtsdatum	Todesdatum	Inhaftiert am	Freigelassen am	Event	Zelle
Deutsch	Aa Insasse 1	01.01.1780	01.01.1810	01.01.1905	01.01.1907	Event 1 DE	Zelle 1
Deutsch	Bb Insasse 2	02.02.1600	02.02.1628	02.02.1602	02.05.1620	Event 2 DE	Zelle 2
Deutsch	Cc Insasse 3	03.03.1700		03.03.1735	03.03.1740	Event 3 DE	Zelle 1
English	Aa Prisoner 1	01.01.1780	01.01.1810	01.01.1905	01.01.1907	Event 1 DE	Zelle 1
English	Bb Prisoner 2	02.02.1600	02.02.1628	02.02.1602	02.05.1620	Event 2 DE	Zelle 2
English	Cc Prisoner 3	03.03.1700		03.03.1735	03.03.1740	Event 3 DE	Zelle 1

Abbildung 23: Übersicht über alle angelegten Insassen

Abbildung 24: Konfiguration der Glossar-Applikation

### 5.3 Neue Zelle hinzufügen

Über einen Klick auf Neue Zelle in der dunkelblauen Leiste kann eine neue Zelle angelegt werden [27](#).

Die Zelle benötigt sowohl einen deutschen als auch englischen Zellennamen. Mit einem Klick auf Anlegen, werden die eingegebenen Daten dauerhaft gespeichert.

Alle angelegten Zellen können über Liste Zellen in der dunkelblauen Leiste angesehen werden [29](#).

### 5.4 Neuen Event hinzufügen

Ein Event benötigt ebenfalls einen deutschen und einen englischen Namen, mit einem Klick auf Anlegen werden die eingegebenen Daten dauerhaft gespeichert [30](#). In Liste Events in der dunkelblauen Leiste können alle Events angezeigt werden.

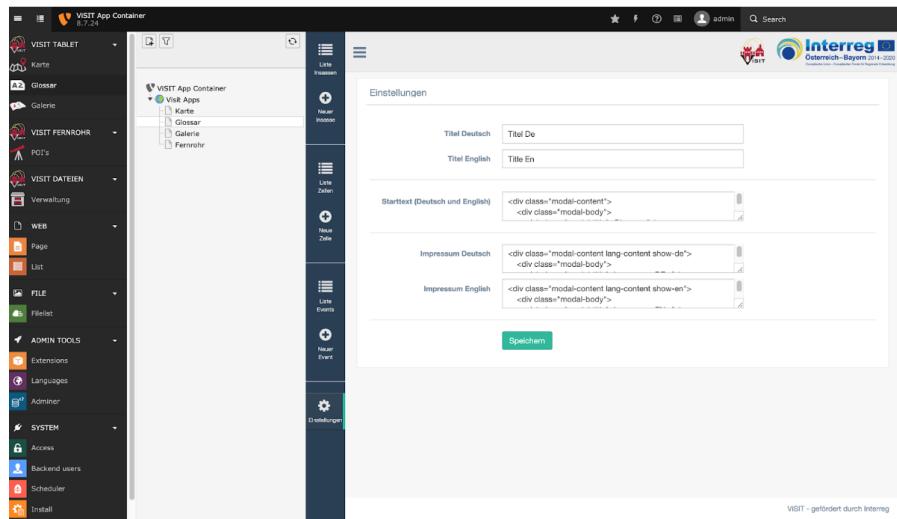


Abbildung 25: Befüllung der Startseite der Glossar-Applikation

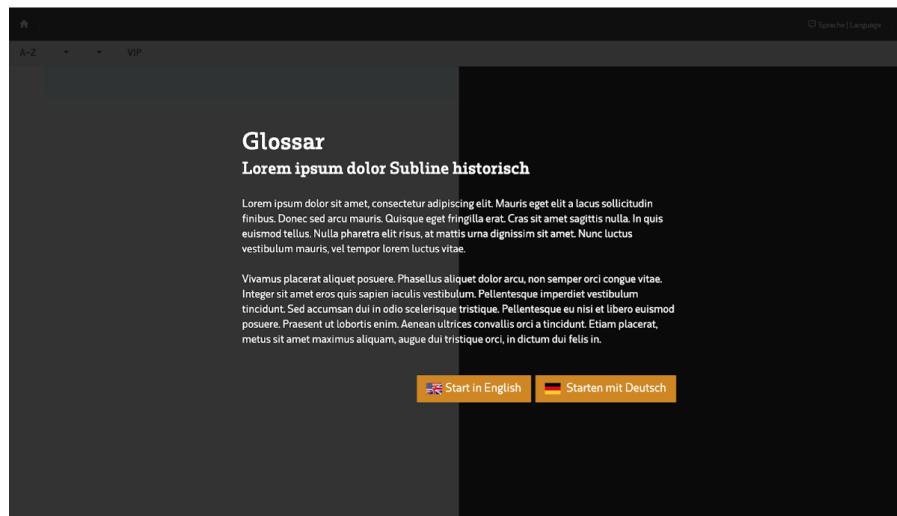


Abbildung 26: Startseite der Glossar-Applikation

## 5.5 Neuen Insassen hinzufügen

Über die Auswahl **Neuer Insasse** in der dunkelblauen Leiste kann ein neuer Insasse angelegt werden. Die Felder Geburtsdatum, Todestag, Inhaftiert sowie Freigelassen sind Datumsfelder. Ist eines der Daten nicht bekannt, kann das Feld freigelassen werden [32](#). Legt man einen Insassen an, muss bei Zelle sowie Event ein bereits angelegte Zelle beziehungsweise ein angelegtes Event angegeben werden. Klickt man auf die Pfeile im Inputfeld, so öffnet sich ein Dropdown Menü und hier kann die entsprechende Zelle beziehungsweise das entsprechende Event ausgewählt werden [33](#). Aus diesem Grund müssen die dazugehörigen Zellen und Events vor dem Anlegen eines Insassens angelegt werden. Die angelegten Insassen können über Liste Insassen in der dunkelblauen Leiste angesehen werden [34](#). Klickt man mittels rechtem Mausklick auf Glossar im Seitenbaum und wählt Show aus, kann die Seite im Browser angesehen werden. Die Insassen werden im Browser, nach dem Auswählen der bevorzugten Sprache, angezeigt.

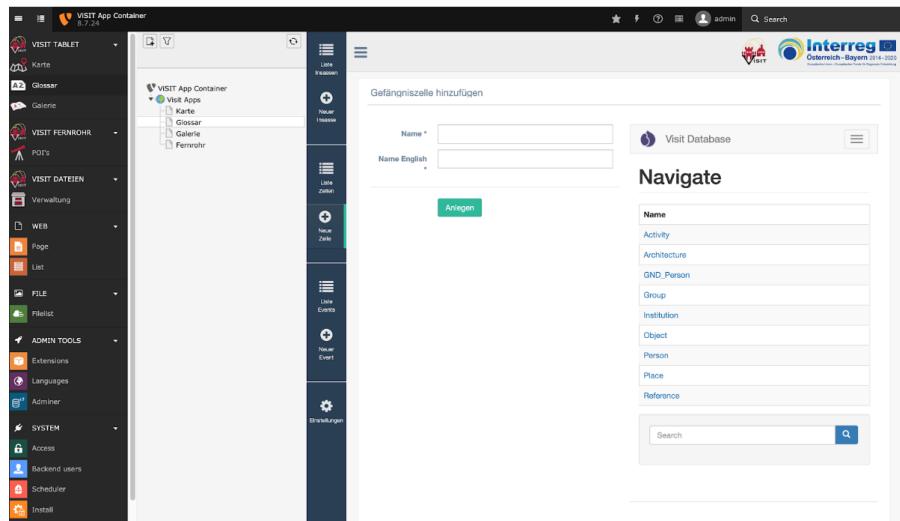


Abbildung 27: Hinzufügen einer neuen Zelle

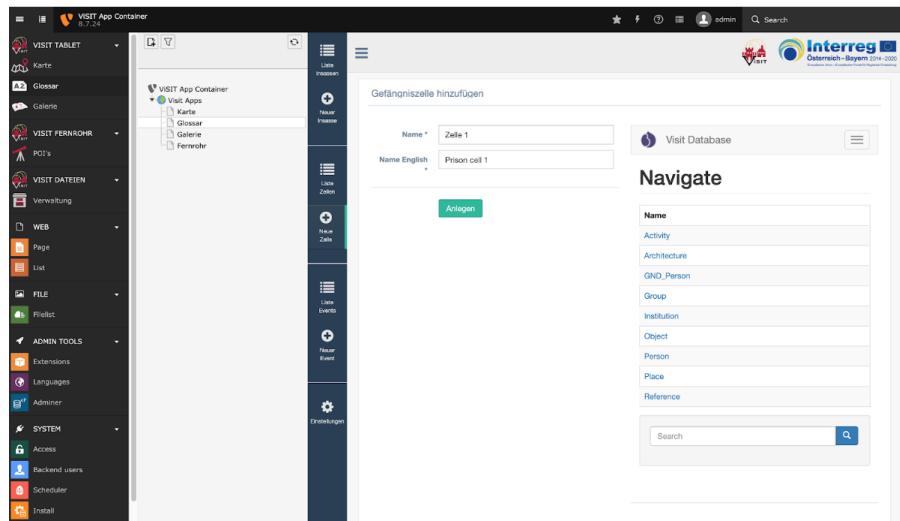


Abbildung 28: Erstellen einer neuen Zelle

## 5.6 Bearbeitung und Löschung von Insassen, Events und Zellen

Insassen, Zellen und Events können jederzeit einzeln bearbeitet oder gelöscht werden. Dies geht, indem zuerst die jeweilige Listenansicht (Liste Insassen/Zellen/Events) in der dunkelblauen Leiste ausgewählt wird. In weiterer Folge kann jedes einzelne Element (Zeile) aus der Liste einzeln bearbeitet oder gelöscht werden. Zum Bearbeiten auf das blaue Stiftsymbol auf der rechten Seite der gewünschten Zeile klicken, zum Löschen des Objekts, den orangen Müllkübel [36](#).

## 6 GALERIE-APPLIKATION

In der Galerie-Applikation kann der Benutzer aus verschiedenen Teasern auswählen, wird ein Teaser angeklickt gelangt man zu den Inhaltselementen und kann diese lesen. Die Inhaltselemente wiederum können aus mehreren

Name	Name En		
Zelle 1	Prison cell 1		
Zelle 2	Prison cell 2		
Zelle 3	Prison cell 3		
Zelle 4	Prison cell 4		
Zelle 5	Prison cell 5		

Abbildung 29: Angelegte Zellen in der Listenübersicht

Abbildung 30: Hinzufügen eines neuen Events

Subelementen bestehen. Der Benutzer muss hinunter scrollen beziehungsweise swipen. Wird nach rechts oder links geswiped, dann wird entweder das vorhergehende oder das nachkommende Inhaltselement (mit den dazugehörigen Subelementen) angezeigt. Nach einer bestimmten Zeit ohne eine Interaktion kehrt die Applikation zurück zum Menü und zeigt den Startbildschirm an.

### 6.1 Einpflegen der Daten in die Galerie-Applikation

Dazu aus der Modulleiste links unter der Obergruppe VISIT TABLET die Karte auswählen. Im linken Teil des Hauptfensters ist der Seitenbaum zu sehen und rechts befindet sich die Kartenübersicht. Oben links im rechten Teil des Hauptfensters befindet sich ein Menü-Button, wird dieser angeklickt, wird eine weitere dunkelblaue Spalte zwischen dem Seitenbaum und dem Arbeitsbereich im Hauptfenster sichtbar (siehe Abbildung 37).

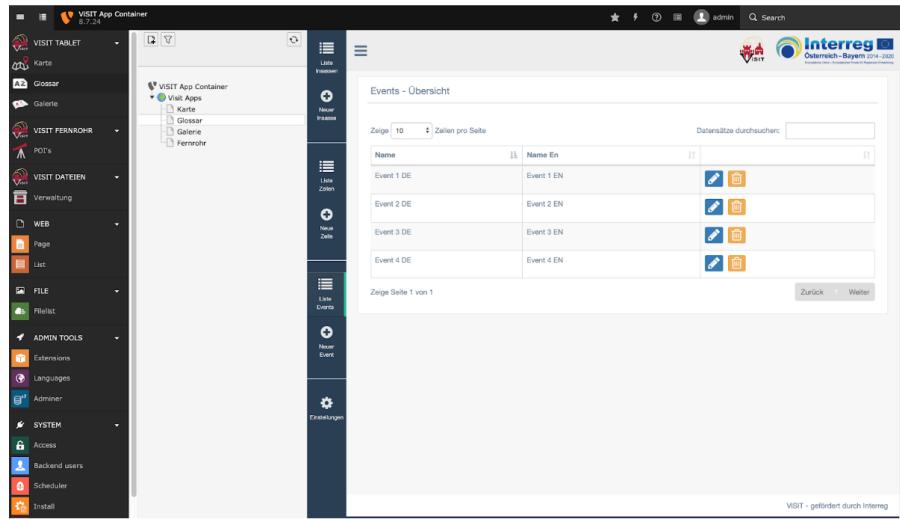


Abbildung 31: Angelegte Events in der Listenansicht

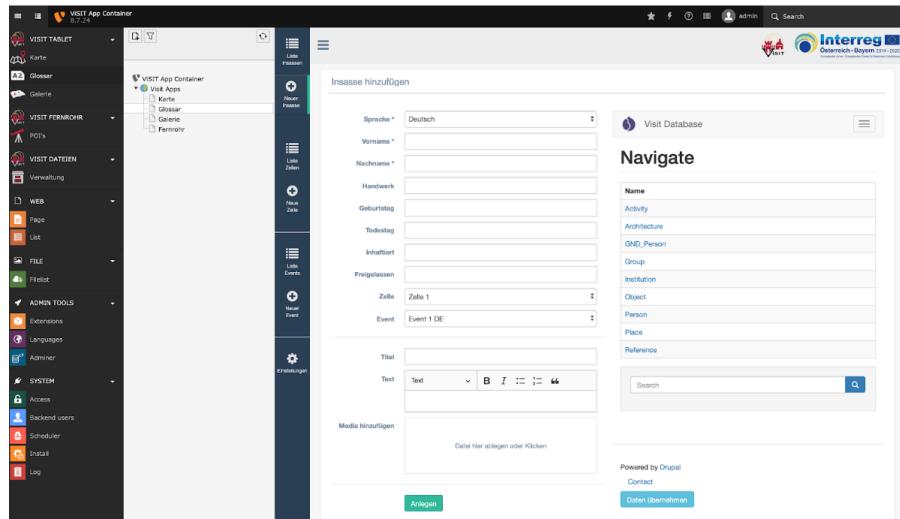


Abbildung 32: Anlegen eines neuen Insassens

## 6.2 Auswahl des gewünschten Layouts für die Startseite

Dazu in der dunkelblauen Leiste "Einstellungen" auswählen (siehe Abbildung 38). Zur Auswahl gibt es hier entweder das 3er-Layout mit 3 Spalten und 1 Zeile (siehe Abbildung 39) oder das 6er-Layout mit 3 Spalten und 2 Zeilen (siehe Abbildung 40) für die Teaser auf der Startseite.

## 6.3 Erstellung der Startseite für die Galerie-Applikation

Die Galerie kann mit einem Klick auf Galerie in der Modulleiste konfiguriert werden (siehe Abbildung 38).

Die Startseite der Applikation kann unter Einstellungen in der dunkelblauen Leiste konfiguriert werden. Dafür benötigt man den Text für die Startseite <https://github.com/VISIT-Dev/appbundle>. Dieser Text muss in die entsprechenden Inputfelder kopiert werden (siehe Abbildung 38).

Abbildung 33: Anlegen eines Insassens

Abbildung 34: Angelegte Insassen in der Listenansicht

Nachdem die Startseite befüllt wurde, wird sie dem Besucher als erstes angezeigt, auf dieser kann der Besucher dann die gewünschte Sprache auswählen (siehe Abbildung 41).

#### 6.4 Neues Inhaltselement anlegen

Als erstes muss ein neues Inhaltselement angelegt werden. Dies geschieht über einen Klick auf Neuer Inhalt in der dunkelblauen Leiste (siehe Abbildung 42).

Im nächsten Schritt muss zuerst die Sprache ausgewählt. Dann wird das Textfeld befüllt und ein Medienobjekt kann zugefügt werden. Bei diesem Text und Medienobjekt handelt es sich um den obersten Text (siehe Abbildung 43 blau eingerahmter Bereich).

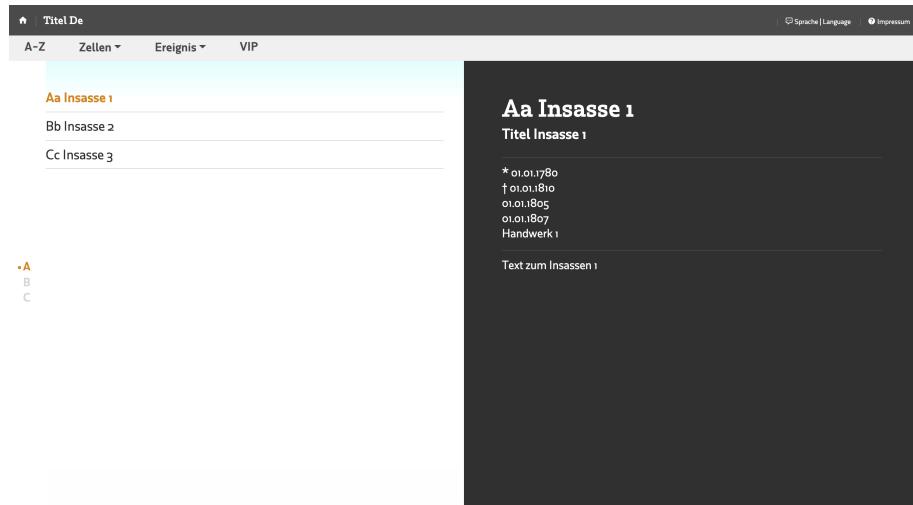


Abbildung 35: Ansicht der Insassen im Browser

Sprache	Name	Geburtsdatum	Todesdatum	Inhaftiert am	Freiglassen am	Event	Zelle
Deutsch	Aa Insasse 1	01.01.1600	09.09.1648	10.10.1623	Event 1 DE	Zelle 1	

Abbildung 36: Bearbeitung bzw. Löschung von Insassen/Zellen/Events

## 6.5 Sortierung der Inhaltselemente

Die Sortierung der Inhaltselemente gibt an, in welcher Reihenfolge die einzelnen Inhaltselemente durchgeswiped (sowohl nach rechts als auch nach links) werden können.

## 6.6 Anlegen eines Sub-Inhaltselements

Nachdem das Inhaltselement angelegt und gespeichert wurde, können - müssen aber nicht - weitere Sub-Inhaltselemente hinzugefügt werden (siehe Abbildung 44 sowie Abbildung 43 grün eingeklammerte Sub-Inhaltselemente).

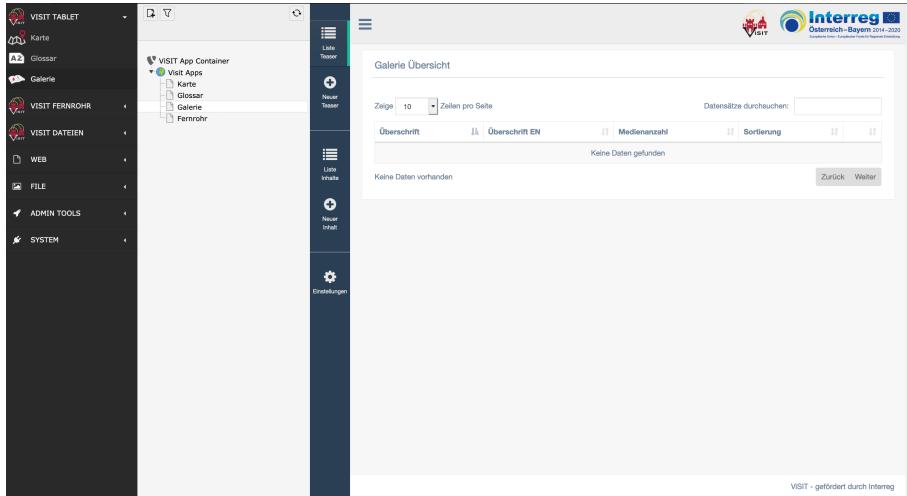


Abbildung 37: Übersicht über alle erstellten Inhalte

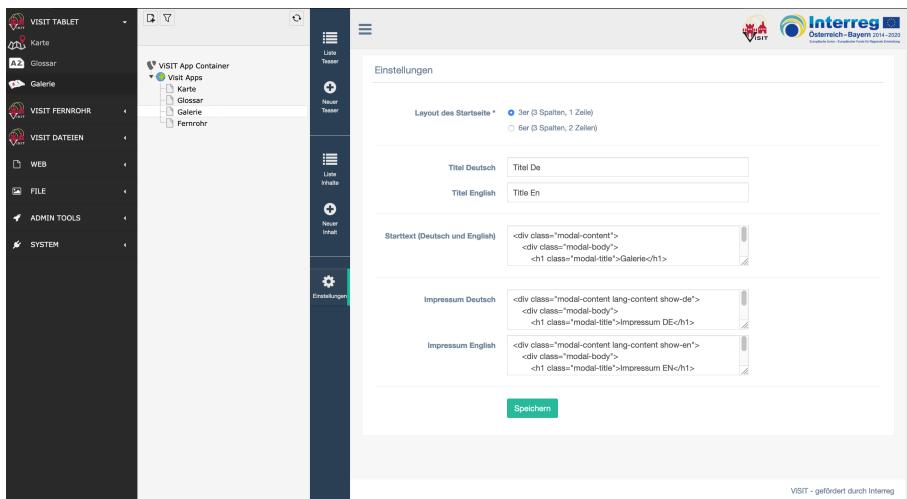


Abbildung 38: Auswahl des gewünschten Layouts in den Einstellungen

## 6.7 Erstellung der Teaser

Für die Startseite müssen als nächstes die Teaser erstellt werden (siehe Abbildung 39 und 40). Je nachdem welches Layout gewählt wurde, müssen entsprechend viele Teaser erstellt werden, also entweder 3 oder 6.

Dazu zuerst aus den dunkel blauen Leiste Neuer Teaser auswählen und den Titel sowohl in Deutsch als auch in Englisch eintragen. Des Weiteren muss der Link zum entsprechenden Inhaltselement angegeben werden (siehe Abbildung 45).

## 6.8 Sortierung der Teaserelemente

Die Sortierung der Teaser auf der Startseite kann manuell eingestellt werden. Die Reihung kann bei der Erstellung des Teasers angegeben werden (siehe Abbildung 45 bei Feld Sortierung). Hier kann über die Pfeiltasten eine Zahl ausgewählt werden, an der der Teaser auf der Startseite angezeigt wird. Im Beispiel 46 wird zuerst die Burg Kreuzenstein, dann die Festung Hohensalzburg und als dritter Teaser die Festung Kufstein angezeigt.

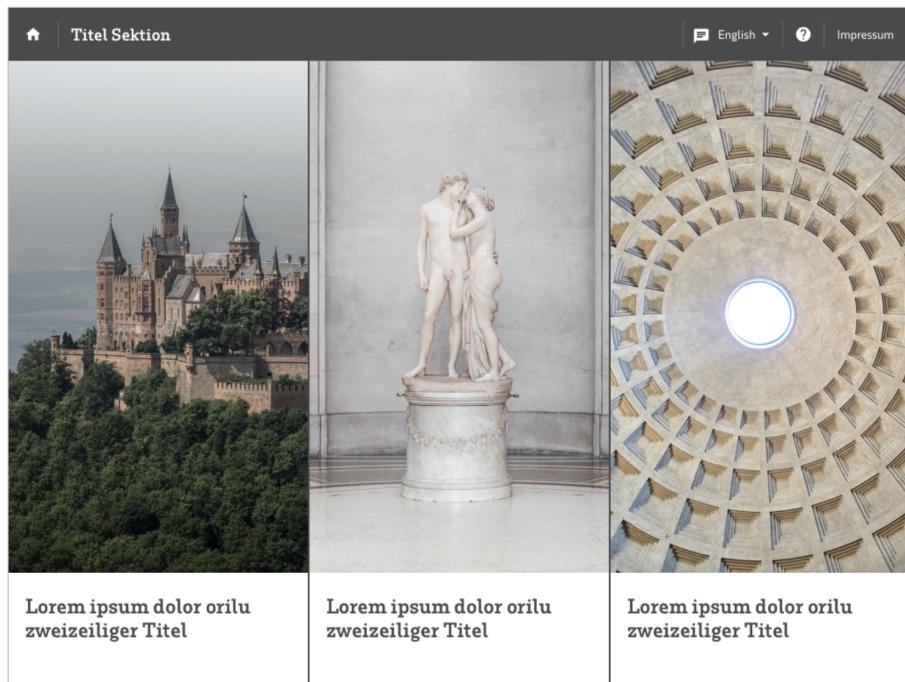


Abbildung 39: 3er-Layout (3 Spalten, 1 Zeile)

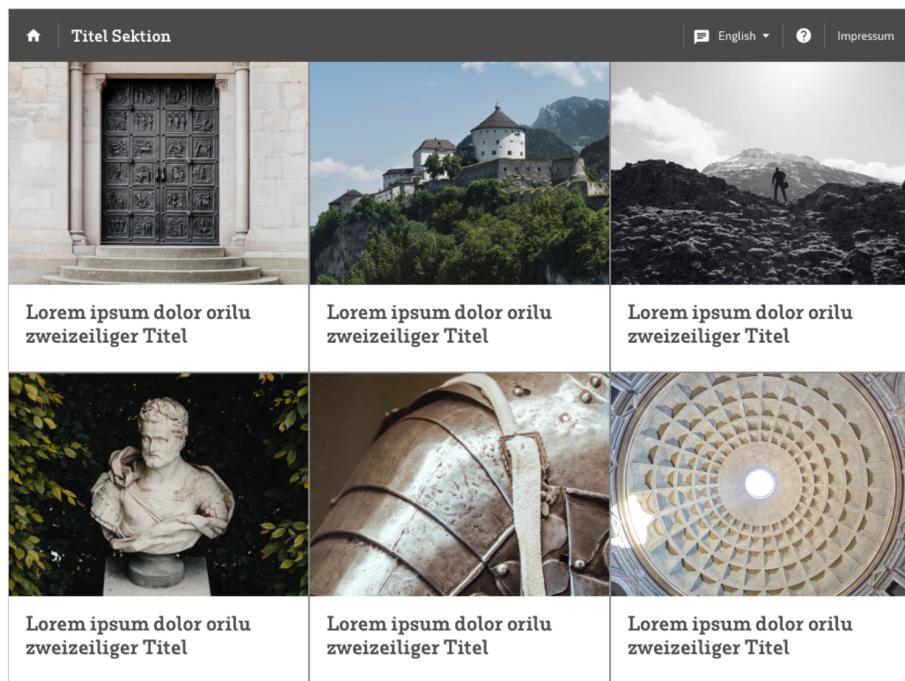


Abbildung 40: 6er-Layout (3 Spalten, 2 Zeilen)

Wird bei allen Null angegeben, dann erscheinen die Teaser in der erstellten Reihenfolge.

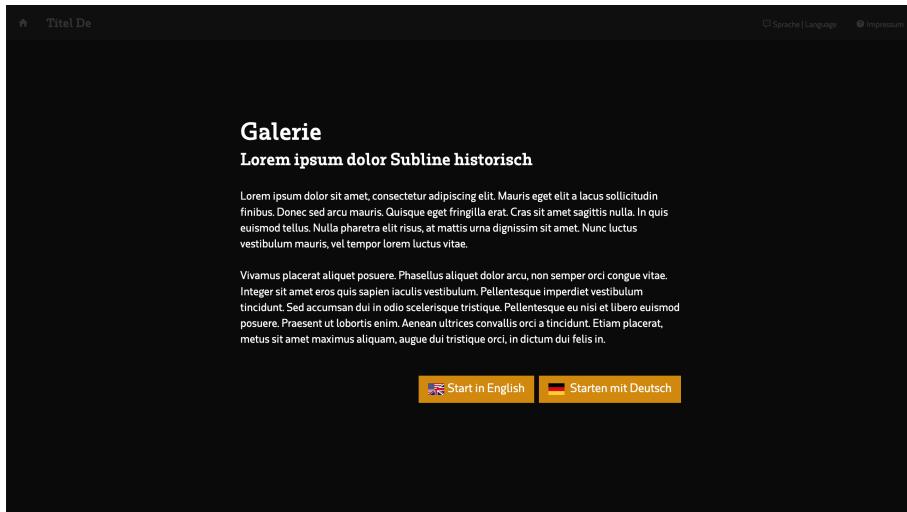


Abbildung 41: Startseite der Galerie-Applikation

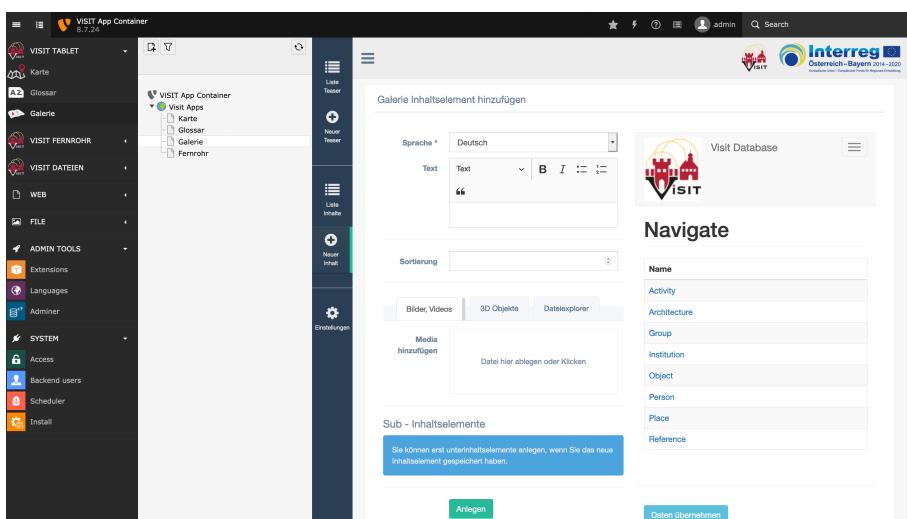


Abbildung 42: Neues Inhaltselement anlegen

## 6.9 Bearbeitung und Löschung Inhaltselementen, Sub-Inhaltselementen sowie Teasern

Die Inhaltselemente, Sub-Inhaltselemente sowie Teaser können jederzeit bearbeitet oder gelöscht werden. Dies geht indem zuerst die jeweilige Listenansicht in der dunkelblauen Leiste ausgewählt wird (für Inhaltselemente beziehungsweise Sub-Inhaltselemente die Liste Inhalte und für Teaser die Liste Teaser). In weiterer Folge kann jedes einzelne Element (Zeile) einzeln bearbeitet oder gelöscht werden. Zum Bearbeiten auf das blaue Stiftsymbol auf der rechten Seite klicken, zum Löschen des gewünschten Objekts, den orangen Müllkübel.

**Galerie, Zusatztitel Galerie**

**Wirkung Französische Revolution auf Österreich 1789–1803**

**Freiheit, Gleichheit, Brüderlichkeit**

Die Französische Revolution von 1789 bis 1799 mit ihrem im Volksmund verkürzten Motto *Liberté, égalité, fraternité* (Freiheit, Gleichheit, Brüderlichkeit) gehört zu den folgenreichsten Ereignissen der neuzeitlichen europäischen Geschichte. Die Abschaffung des feudal-absolutistischen Standestaats sowie die Propagierung und Umsetzung grundlegender Werte und Ideen der Aufklärung als Ziele der Französischen Revolution – das betrifft insbesondere die Menschenrechte – waren mittlerweile für tiefgreifende politische und gesellschaftspolitische Veränderungen in ganz Europa und haben das moderne Demokratieverständnis entscheidend beeinflusst.

Die Französische Revolution wird einerseits gefeiert als große Errungenschaft für ein modernes Rechts- und Gesellschaftssystem, andererseits verflucht aufgrund der massiven Gewalt und des Terrors, welche das Land jahrelang ...

[Mehr lesen](#)

**Subelement 1**

**Kapitel 2 lorem ipsum dolor**

„Der Kaiser ließ sich in seiner Außenpolitik ausschließlich von den Interessen des Ershauses leiten, so daß er zwar dafür sorgte, daß das revolutionäre Geschehen nicht auf Österreich übergriff, im übrigen blieb er aber Zurückhaltung gegenüber den revolutionären Vorgängen in Frankreich. [...] Anfang 1792 mußten Leopold II. jedoch erkennen, daß die Revolution sich langsam zu verschärfen begann, [...].“ „Der Kaiser, der die Gerüchte über französische Agenten ernst nahm, erteilte dem Polizeiminister Pergen den Auftrag, alle Ausländer, darunter insbesondere Franzosen und Italiener, überwachen zu lassen, sofern sie sich nicht aus bekannten Gründen in der Monarchie aufhielten. [...]“

Die Haltung der Habsburger zur Revolution war zunächst zwiespältig: Leopold II. zeigte Sympathien, er erkannte die Schwäche des Absolutismus in Frankreich. In die dortigen inneren Angelegenheiten konnte er sich ...

[Mehr lesen](#)

**Die Gefangenen**

Französische Staatsgefangene galten als besonders gefährlich. Sie trugen sozusagen das Gift der Revolution mit sich herum. Wien, vom Ausbruch der Französischen Revolution zu Beginn jedenfalls mehr oder weniger überrascht, war über ihre Ursachen und ihre Tendenz lückenhaft und in der Regel durch Emigranten falsch unterrichtet. Daher dienten die Gefangenen auch dazu, diese Nachrichtenlücken zu schließen.

Mit der Verhaftung Théroigne de Méricourt (1791) glaubte die Österreichische Regierung eine Rädelsführerin der Französischen Revolution in ihren Händen zu haben. Sie war nicht zufällig irgendwo aufgegriffen worden, sondern regelrecht entführt worden. Angeblich wurde sie in die Österreichischen Niederlande ausgeschickt um einen Aufruhr zu schüren und stand in Verdacht, Mordpläne gegen die Königin Marie-Antoinette gehegt zu haben. Hugues Bernhard Maret, Herzog von Bassano und Graf Charles Huet Semonville (1792), beides französische Staatsbedienstete, wurden auf ihrem Weg nach Italien festgehalten. Angeblich wurden sie als Repressalie Österreichs gegen die Behandlung Marie-Antoinette, Königin von Frankreich und Schwester des Habsburger Herrschers Leopold II. dienen.

**Subelement 3**

Abbildung 43: Ansicht eines Inhaltselements aus der Galerie auf einem Tablet

## 7 FERNROHR

- 7.1 Einpflegen der Daten in die Fernrohr-Applikation
- 7.2 Erstellung der Startseite für die Fernrohr-Applikation
- 7.3

## 8 DATEIVERWALTUNG

- 8.1 Zugangsdaten zum Dateimanagement

Das Dateimanagement ist eine Applikation, mit der die Daten in der Vi-

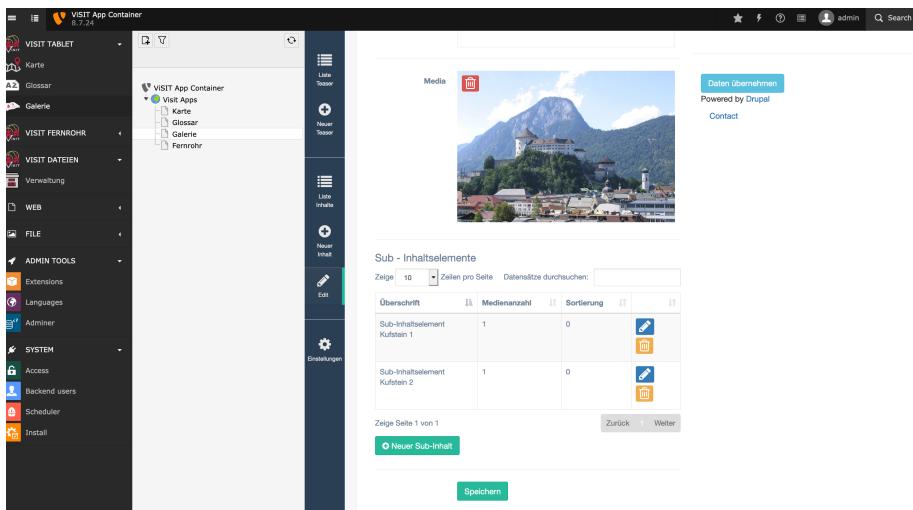


Abbildung 44: Anlegen neuer Sub-Inhaltselemente zu einem bestehenden Inhaltselement

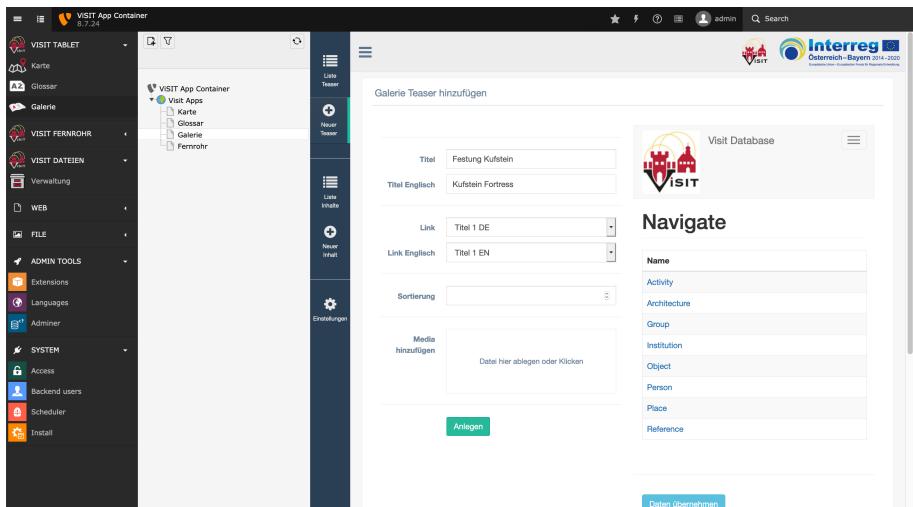


Abbildung 45: Erstellung eines Teasers für die Startseite der Galerie-Applikation

sich im TYPO3 Backend. Dafür müssen zuerst aus der Modulleiste die Extensions ausgewählt werden. In weiterer Folge kann im Hauptfenster die Visit App gefunden werden. Durch einen Klick darauf kommt man zu den Einstellungen (siehe Abbildung 47). Diese Einstellungen sind im ganzen System gleich. Diese Login Daten sind sensibel, da sie Zugang zum Peer-to-Peer-Netzwerk geben, deshalb sind sie nicht im Docker angeführt. Sie bekommen diese Zugangsdaten (API User, Password for API User sowie die Syncthing Master ID) entweder von der betreuenden Firma oder von einem anderen ViSIT-Partner (siehe Abbildung 48).

Ist aber ein ViSIT-Partner nicht am Hochladen von Dateien interessiert, dann werden die Zugangsdaten zum Dateisystem nicht benötigt.

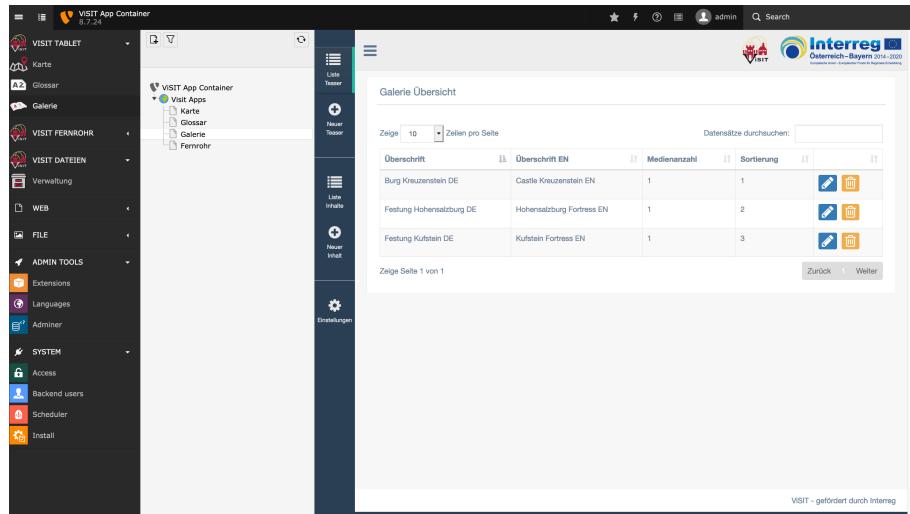


Abbildung 46: Manuelle Sortierung der Teaserelemente auf der Startseite

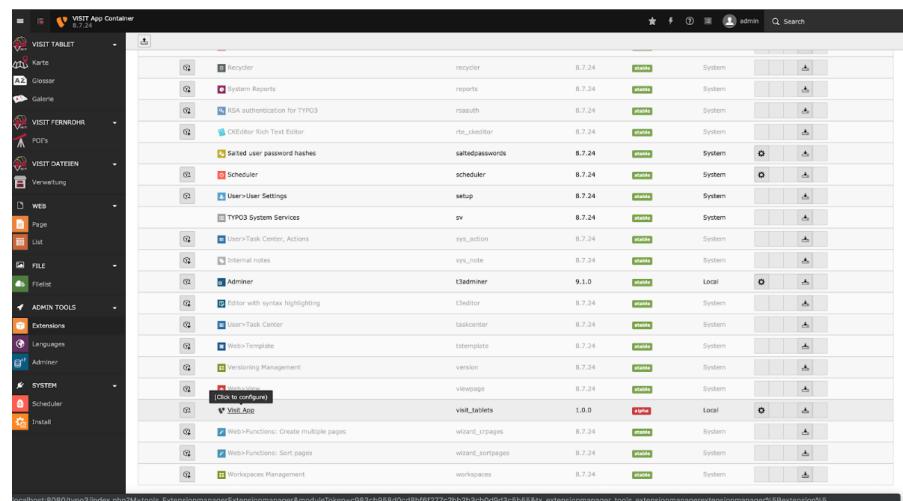


Abbildung 47: Einstellung der ViSIT App Extension

## 8.2 ViSIT-Partner-Liste

Um zu sehen, welche Partner Zugriff zum Peer-to-Peer-Netzwerk haben, muss zuerst aus der Modulleiste unter VISIT DATEIEN die Verwaltung ausgewählt werden. In weiterer Folge aus der dunkelblauen Leiste die Partner Liste auswählen. Hier sind alle ViSiT-Partner, die Zugang zum Dateiverwaltungssystem haben, aufgelistet.

Diese Partner haben Zugriff auf die bereits abgelegten Dateien in der Mediendatenbank im Peer-to-Peer-Netzwerk. Diese Dateien können in weiterer Folge beim Anlegen von Objekten ausgewählt werden. Ist eine benötigte Datei noch nicht vorhanden, so muss sie in die Datenbank eingepflegt werden. Dies geschieht über Verwaltung und dann Datei hochladen.

In der Abbildung 49 ist ersichtlich, dass es drei Benutzer gibt, die Zugriff auf die Mediendatenbank haben (K\*\*\*, M\*\*\* und P\*\*\*), im Feld ID ist ihre ViSIT-Zugangs-ID ebenfalls ersichtlich. Die einzelnen Partner sind im Grunde nur Verzeichnisse mit Namen und der entsprechenden Partner-ID. Der

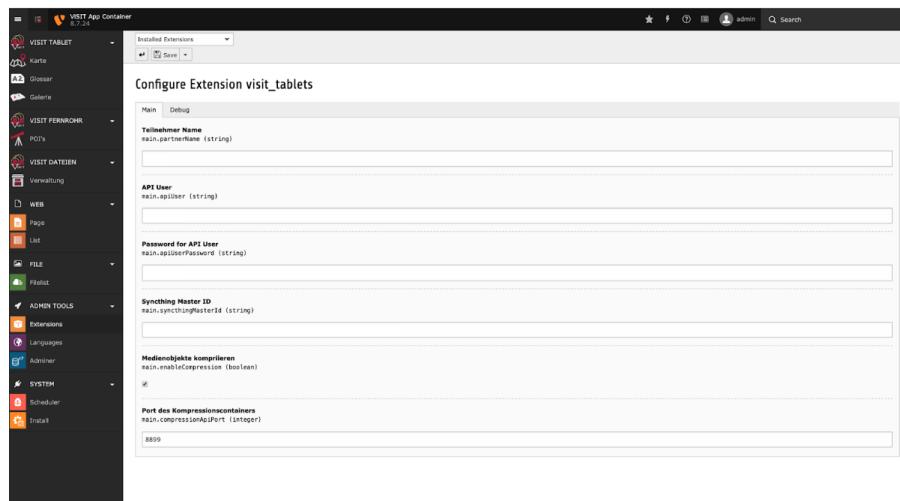


Abbildung 48: ViSIT App Extensions

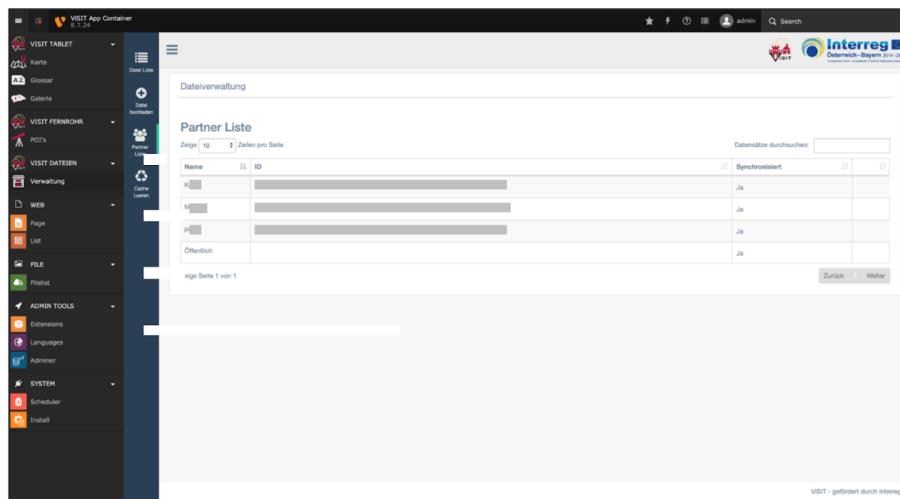


Abbildung 49: Ansicht der ViSIT-Partner mit Zugang zur Mediendatenbank im Peer-to-Peer-Netzwerk

Benutzer Öffentlich ist ein öffentliches Verzeichnis, er besitzt keine ID, er entspricht sozusagen einem virtuellen Benutzer, er ist nur aus organisatorischen Gründen angeführt.

### 8.3 Upload von 3D-Objekten und Bildern, Videos und anderen Dateien

Damit eine Datei in die Datenbank geladen werden kann, muss zuerst in der Modulleiste unter der Obergruppe VISIT DATEIEN die Verwaltung ausgewählt werden. Danach erscheint im Hauptfenster die Datei Liste über alle bereits verfügbaren Dateien (siehe Abbildung 50). Der Zugriffsmodifikator der bereits hochgeladenen Dateien kann entweder *public*, *visit* oder *private* sein.

Zugriff *public* bedeutet, dass jeder diese Datei sehen und verwenden kann. Bei Zugriff *visit* hat jeder ViSIT-Partner Zugang zu der Datei, wohingegen Zugriff *private* nur für die eine Installation zugänglich ist.

Die Dateien mit dem Zugriffsmodifikator *public* und *visit* können mit einem

Klick auf das blaue Downloadsymbol in der entsprechenden Zeile heruntergeladen werden.

Abbildung 50: Ansicht der bereits verfügbaren Dateien in der Dateiliste

#### 8.4 Hochladen von Dateien

Für das Hochladen eines Medienobjekts ist das Ausfüllen des Titels, des Urhebers, die ViSIT Partner ID sowie der ViSIT Partner Name zwingend erforderlich. Die „Beschreibung“ sowie „Hochgeladen“ von sind optional. Im nächsten Schritt muss der Dateipfad für die Datei (entweder 3D Objekt oder Bild, Video und weitere Dateien) angegeben werden.

Als nächstes muss die Datei einem bereits angelegten Objekt (Entität) zugeordnet werden. Dies wird in der rechten Spalte des Hauptfensters gemacht. Hier sieht man die Visit Database, hier muss das entsprechende Objekt ausgewählt werden. Klickt man beispielsweise auf „Place“, dann öffnen sich alle bereits angelegten Orte, aus diesen kann dann ein Ort ausgewählt werden. Wird ein Ort ausgewählt, dann erscheinen die Detailinformationen zu diesem in der gleichen Spalte.

Hat man zufällig ein falsches Objekt ausgewählt, kann mittels Klick entweder auf das Burger-Menü rechts neben Visit Database geklickt werden und dann Navigate ausgewählt werden oder unter Visit Database im Breadcrumb-Menü auf Navigate klicken, dann kommt man ebenfalls zur Übersicht.

Wurde das korrekte Objekt gefunden, dann können die Daten übernommen werden. Dies geschieht mittels Klick auf „Daten übernehmen“ (siehe Abbildung 51). Nach dem Klick wird das Feld oben rechts bei „Gewählte Entität\*:“ automatisch ausgefüllt. Danach kann das Objekt zur ViSIT Datenbank hinzugefügt werden. Dies geschieht mittels Klick auf „Medienobjekt zur ViSIT Datenbank hinzufügen“. Danach kommen zwei Meldungen, einerseits eine Meldung, dass die Kompression des Medienobjektes erfolgreich gestartet wurde und dass die Datei erfolgreich hinzugefügt wurde (siehe Abbildung 52).

Nachdem die Datei in der ViSIT Datenbank gespeichert wurde, kann sie in der Dateiliste angesehen werden (siehe Abbildung 53). Der Zugriff auf die hochgeladene Datei ist per default immer *private*. Will man die Datei

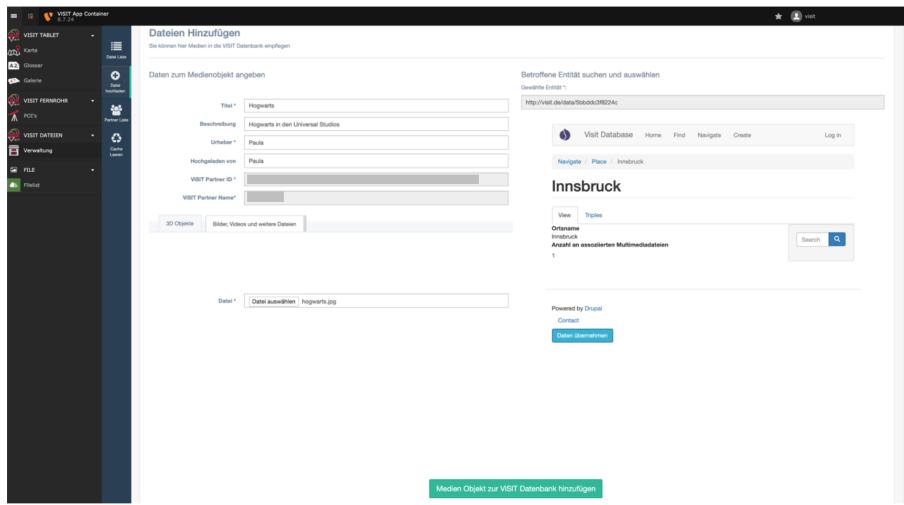


Abbildung 51: Ansicht des ausgefüllten Formulars für den Datei-Upload

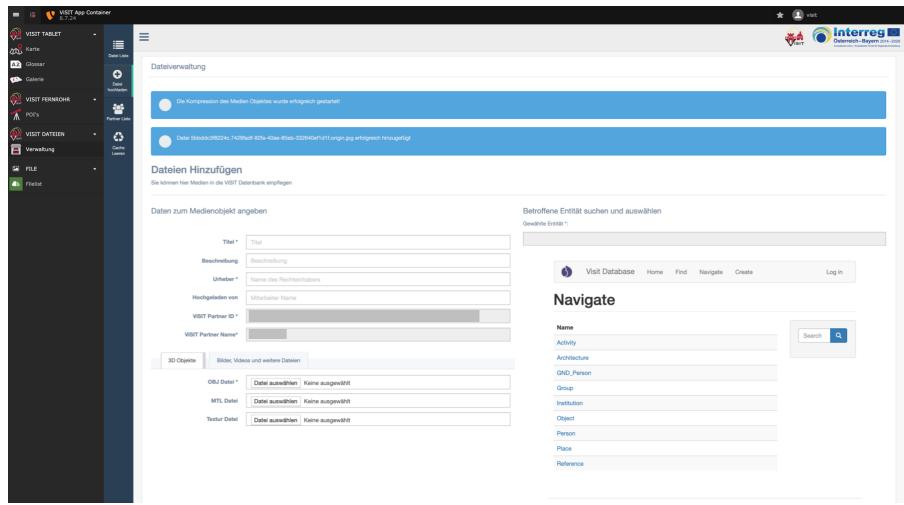


Abbildung 52: Bestätigungennachrichten nach einem erfolgreichen Upload

Dateiverwaltung					
Zeile	10	Zellen pro Seite	Datensätze durchsuchen:		
Titel	Beschreibung	Urheber	Datenbank Objekt	Komprimierungen	
Hogwarts	Hogwarts in den Universal Studios	Paula		Bez. Zugriff Erstellt Original private 04.05.2019 13:34	
Teller	Holzteller	FHK	Passauer Stadtbrand	Bez. Zugriff Erstellt Original public 09.05.2019 08:50	
Teller 123	Beschreibung 123	Kris		Bez. Zugriff Erstellt 6000 visit 22.05.2019 13:46 20000 private 22.05.2019 13:46 500 private 22.05.2019 13:46 Original private 22.05.2019 13:46 1000 public 22.05.2019 13:46	
Teller2		FHK	Legende des heiligen Georg	Bez. Zugriff Erstellt Original visit 09.05.2019 11:44	
Title test		FHK	Passauer Stadtbrand		

Abbildung 53: Hochgeladene Datei in der Listenübersicht

verwenden, muss sie zuerst heruntergeladen werden. Dies geschieht mit einem Klick auf das blaue Download-Symbol, jetzt ist die Datei lokal gespeichert. Falls ein Partner diese Datei löschen sollte, bleibt die lokale Kopie gespeichert.

## 8.5 Veröffentlichung einer Datei im ViSIT-Netzwerk

Wenn eine Datei hochgeladen wurde, welche für alle ViSIT-Partner zur Verfügung stehen soll, dann muss diese Datei explizit freigegeben beziehungsweise veröffentlicht werden. Dies kann in der Dateiverwaltung (Datei Liste) gemacht werden. Für Zugriff *public*, das heißt, dass jeder Zugriff auf die Datei hat, muss bei der entsprechenden Datei das orange Weltkugel-Symbol angeklickt werden (siehe Abbildung 53 rot eingerahmt). Soll die Datei hingegen nur für die ViSIT-Partner sichtbar sein - Zugriff *visit* - dann muss auf das grüne Symbol geklickt werden (siehe Abbildung 53 blau eingerahmt).

# 9 UPDATE-PROZESS

Ob ein Update verfügbar ist, kann im GitHub-Repository des Projekts unter Commits <https://github.com/ViSIT-Dev/appbundle/commits/master> eingesehen werden. Sollte ein Update fällig sein, da ein Commit kürzlich stattfand, dann gibt es zwei Möglichkeiten, dieses durchzuführen.

Vor jedem Update soll immer ein Backup vom kompletten Webspace, dies beinhaltet die Datenbank sowie alle Dateien, gemacht werden.

## 9.1 Update von Programmdateien

Programmdateien können leicht manuell über „Run Task“ im Scheduler des Backend aktualisiert werden. Dazu in der Modulleiste unter SYSTEM den Scheduler auswählen (siehe Abbildung 54). Im Arbeitsbereich sieht man die Scheduled Tasks und in der ersten Zeile befindet sich die App, die upgedated werden kann. Hier werden alle Änderungen vom oben genannten GitHub Repository heruntergeladen. Unter „Last Execution“ (siehe Abbildung 54 blau eingerahmt) wird das Datum des letzten Updates angezeigt. Ist dieses Datum älter als das Datum des letzten Commits, dann kann ein Update vorgenommen werden (siehe Abbildung 55 rot eingerahmtes Datum).

Dazu muss das ganz rechts in der entsprechenden Zeile befindliche Play-Symbol „Run task“ angeklickt werden (siehe Abbildung 54 rot eingerahmt). Danach sollte der Cache der gesamten Webseite (Front- und Backend) gelöscht werden. Dazu das Blitz-Symbol (siehe Abbildung 56 gelb eingerahmt) in der Kopfzeile auswählen und den roten Blitz „Flush all caches“ (siehe Abbildung 56 rot eingerahmt) anklicken.

Jetzt sollte die Applikation wieder auf dem neuesten Stand sein.

## 9.2 Datenbank Update

Sollte nach dem oben genannten Update über „Run task“ und Cache leeren die Applikation nicht funktionieren oder es handelte sich bei dem Update um ein Datenbank-Update, dann muss zuerst die Visit App-Extension deaktiviert und wieder aktiviert werden.

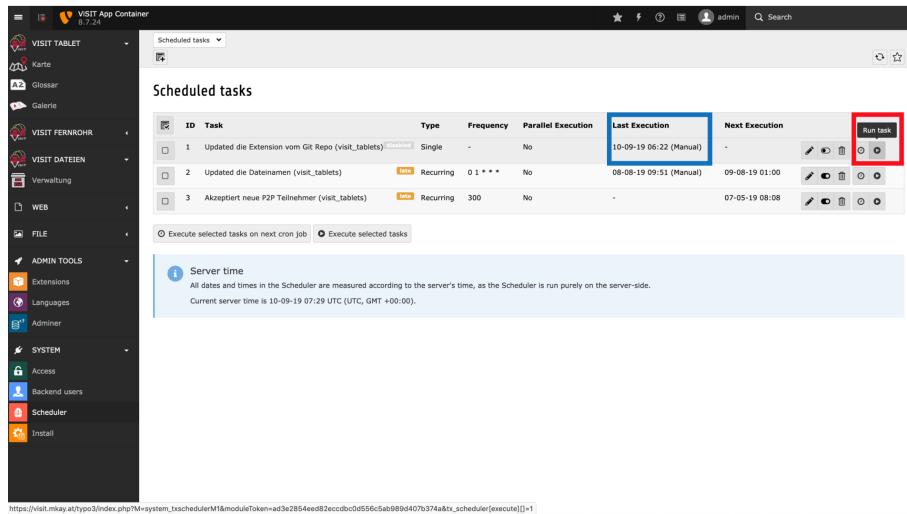


Abbildung 54: Scheduler und geplante Tasks

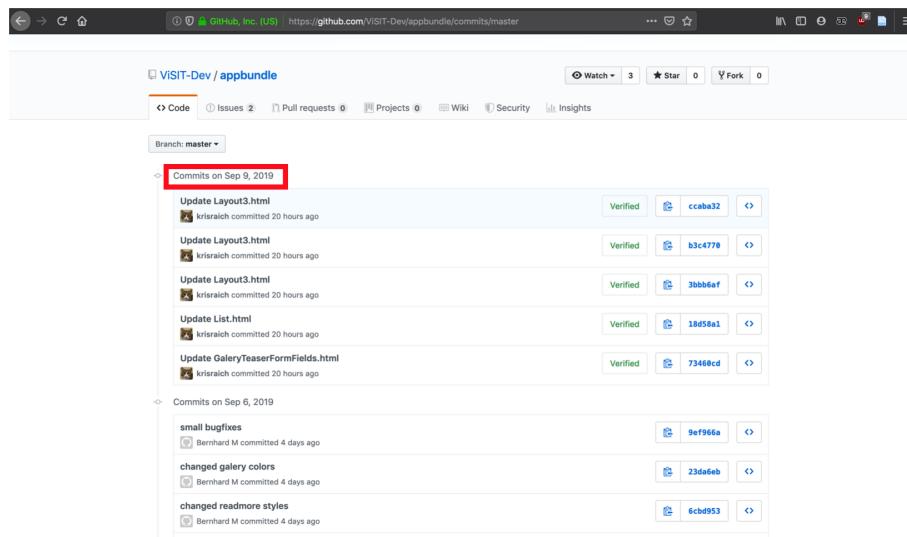


Abbildung 55: Commits im GitHub Repository

Die Extensions befinden sich in der Modulleiste unter ADMIN TOOLS. Danach im Arbeitsbereich hinunter scrollen bis zur Visit App (siehe Abbildung 57 blau eingerahmte Zeile). Im nächsten Schritt muss diese Extension kurz deaktiviert werden, dies passiert mittels Klick auf das Würfelsymbol ganz links in der Zeile (siehe Abbildung 57 rot eingehaumt). Nach kurzer Zeit kann die Extension auch wieder mit einem Klick auf das Würfelsymbol aktiviert werden.

Danach sollten die Applikationen wieder funktionieren und auf dem neuesten Stand sein.

### 9.3 Backup Plan

Sollten die beiden oben beschriebenen Update-Vorgänge scheitern, dann muss das Backup wiederhergestellt werden damit die Applikationen funktionsfähig sind und danach den Support anrufen und abklären, warum das Update fehlgeschlagen ist.

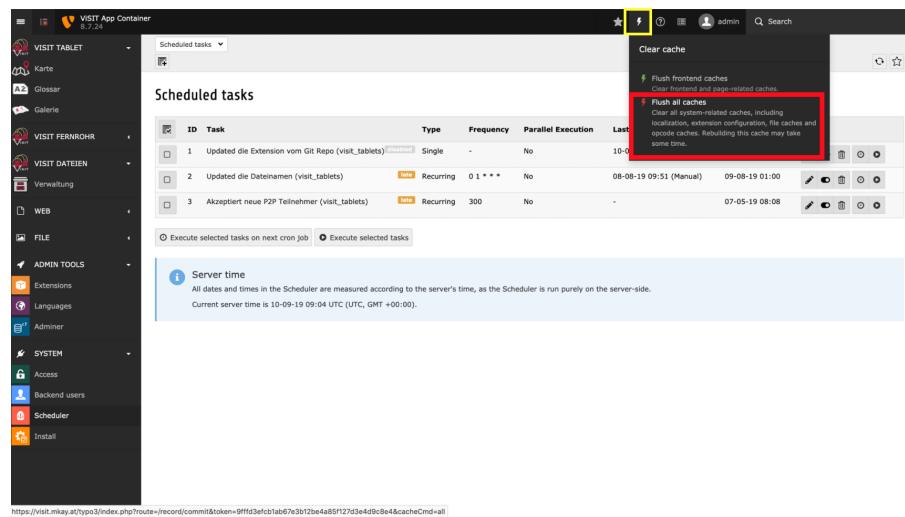


Abbildung 56: Cache leeren

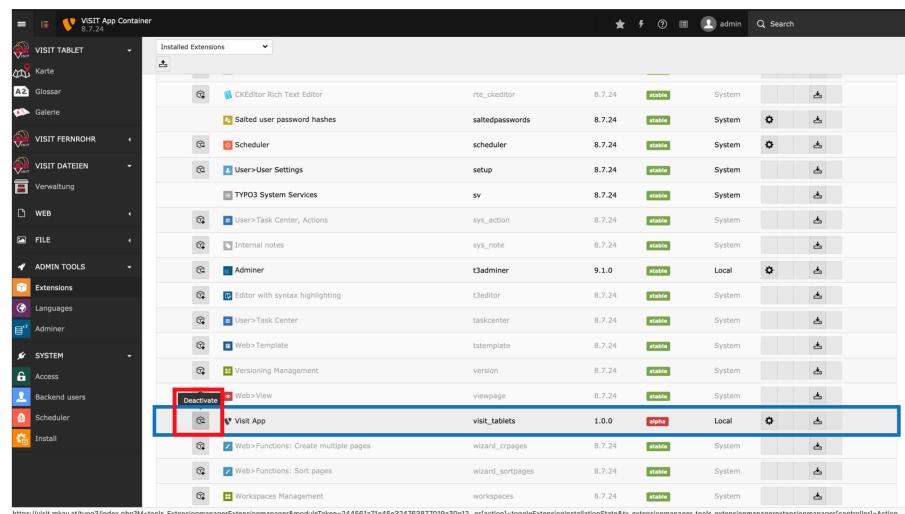


Abbildung 57: Visit App-Extension deaktivieren und wieder aktivieren

## 10 PROJEKTVERLAUF

## 11 KOMPRESSION

### 11.1 Motivation

Unabhängig davon, ob man eine Ausstellung in einem Museum gestaltet oder eine Webseite erstellt, lässt sich fast jede für Besucher oder Benutzer konzipierte Darstellung durch den Einsatz von Bildern aufwerten. Dieses lässt sich durch den Einsatz von 3D-Modellen, mit welchen der Benutzer durch Drehen, Zoomen, etc. interagieren kann, noch deutlich steigern. Abgesehen davon, dass dies in einer Ausstellung nur durch den Einsatz von Rechnern, wie beispielsweise von Tablets, möglich ist, sind damit jedoch einige technische Herausforderungen verbunden.

Einerseits erwartet der Benutzer ein möglichst realistisches Erlebnis, was nur durch hochauflöste 3D-Modelle und den damit einhergehenden großen Datenmengen möglich ist. Dennoch soll das Modell möglichst ohne Latenz angezeigt werden und flüssige Interaktion erlauben. Soll die Darstellung darüber hinaus auf einem mobilen Endgerät des Benutzers, wie beispielsweise einem Smartphone, stattfinden, das einerseits nur eingeschränkte Rechenkapazität bietet und auf das andererseits die gesamten Daten für jeden Benutzer separat übertragen werden müssen, entsteht hier ein Gegensatz, für den ein geeigneter Kompromis zu finden ist.

Dieser Kompromis besteht darin, die 3D-Modelle nicht in der höchsten verfügbaren Auflösung dem Benutzer darzustellen, sondern in einer dem Anwendungsfall und dem Endgerät angemessenen Größe. Beispielsweise ist für eine ansprechende Anzeige auf einem Smartphone-Bildschirm eine geringere Auflösung notwendig als auf der Workstation eines Museumsmitarbeiters mit entsprechend großem Bildschirm, die auch dementsprechend leistungsfähig ist, und über welche dieser Mitarbeiter das angezeigte Objekt erforschen möchte. Aus diesem Grund ist es sinnvoll, in der Medien-Datenbank die gespeicherten 3D-Modelle in verschiedenen Auflösungen bereit zu halten, weshalb diese in der Regel beim Hochladen in die Datenbank automatisch komprimiert werden.

Die eben beschriebene Problematik tritt nicht nur bei 3D-Modellen auf, sondern auch bei herkömmlichen zweidimensionalen Bildern. Da auch solche Medien in der Medien-Datenbank gespeichert werden sollen, sind auch Bilddateien zu komprimieren, was sich sehr einfach durch das Verringern der Auflösung bewerkstelligen lässt. Von jedem hochgeladenen Bild werden also komprimierte Versionen in verschiedenen Auflösungsstufen erstellt, auf welche anschließend zugegriffen werden kann.

In diesem Kapitel wird nach einer kurzen Erläuterung der theoretischen Grundlagen auf die an die ViSIT-Medien-Datenbank angebundene Kompressions-Komponente und deren Voraussetzungen, ihre Funktionsweise, die Installation und die Bedienung, die über eine Web-Oberfläche erfolgt, eingegangen. Außerdem werden die zur Verfügung stehenden Konfigurationsoptionen eingehend erläutert. Abschließend wird die Schnittstelle (API), über welche die Kompressions-Komponente unabhängig von der dafür verfügbaren Web-Oberfläche oder der ViSIT-Medien-Datenbank angesprochen werden kann, spezifiziert.

### 11.2 Grundlagen

Bei 3D-Modellen gilt es grundsätzlich zwischen der Geometrie, durch welche die Form der Oberfläche eines Objekts beschrieben wird, und der Textur,

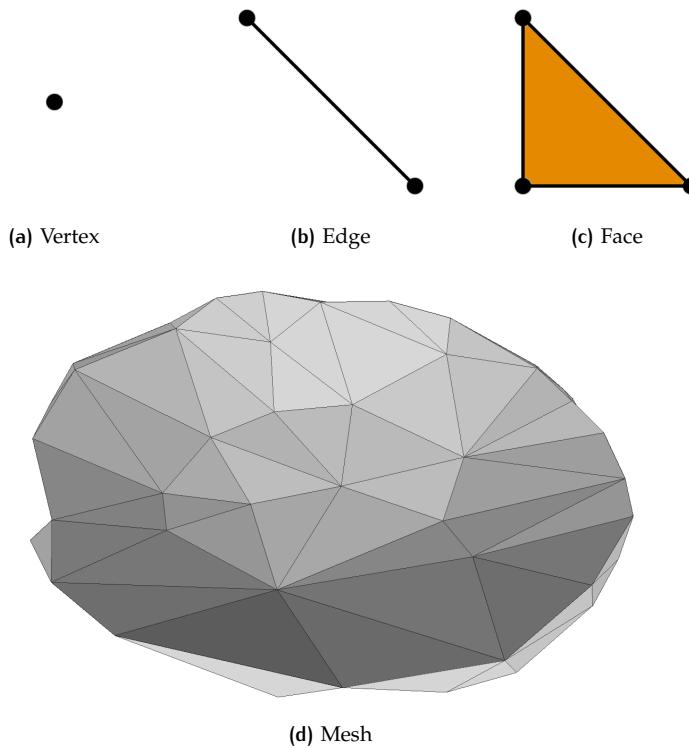


Abbildung 58: Grundlegende Elemente der Geometrie eines 3D-Modells

durch welche die Färbung der Oberfläche ausgedrückt wird, zu unterscheiden. Während die Geometrie obligatorisch für ein 3D-Modell ist, muss nicht zwangsläufig eine Textur existieren. Nicht jedes Digitalisierungsverfahren ist in der Lage, Texturdaten zu erfassen, wie beispielsweise ein Laserscanner. Im Folgenden werden Aufbau und Kompression dieser beiden Bestandteile eines 3D-Modells behandelt.

#### 11.2.1 Geometrie von 3D-Modellen

Herkömmliche 3D-Modelle, die ausschließlich die Oberfläche eines Objekts und nicht dessen Inneres beschreiben, bestehen aus mehreren Punkten im dreidimensionalen Raum, die *Vertices* genannt werden. Diese Punkte werden zu Flächen, den sogenannten *Faces* verbunden, wobei es sich hier in vielen Fällen um Dreiecke handelt. Die Anzahl der Ecken eines Faces wird als dessen *Ordnung* bezeichnet. Die Kanten dieser Flächen, die Verbindungen zwischen zwei Vertices darstellen, werden *Edges* genannt. Die Gesamtheit aller Vertices und Faces eines 3D-Modells wird auch als *Mesh* bezeichnet. Die soeben genannten Begriffe werden in Abbildung 58 grafisch dargestellt.

Während bei einfachen Dateiformaten, wie beispielsweise dem STL-Format<sup>1</sup>, für jedes Face die Koordinaten eines jeden Eckpunkts separat gespeichert werden, wird die dadurch verursachte Redundanz zum Beispiel bei OBJ-Dateien<sup>2</sup> vermieden, indem zunächst alle Vertices in Form der Koordinaten einmalig angegeben werden und anschließend bei der Definition der Faces auf diese Vertexdefinition indexbasiert zugegriffen wird.

<sup>1</sup> [http://www.fabbers.com/tech/STL\\_Format](http://www.fabbers.com/tech/STL_Format)

<sup>2</sup> <http://www.martinreddy.net/gfx/3d/OBJ.spec>

Bei Meshes, die ein reales Objekt beschreiben, sollte dieser möglichst einer orientierbaren stetigen zweidimensionalen Mannigfaltigkeit, die in den dreidimensionalen Raum eingebettet ist, entsprechen [Bot+10, S. 3]. Dies hat zur Folge, dass für jeden Punkt im Raum eindeutig entschieden werden kann, ob er im Inneren oder im Äußeren des Objekts liegt. Dies impliziert beispielsweise, dass kein Edge Teil von mehr als zwei Faces sein kann. Jedoch sind auch an die Vertices bestimmte Bedingungen zu stellen, wobei für Details auf [Bot+10, S. 11f] verwiesen wird.

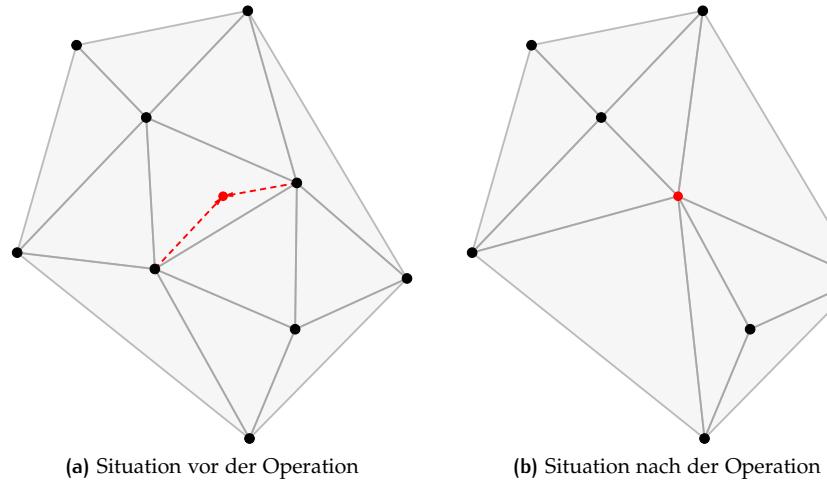
#### 11.2.2 Textur von 3D-Modellen

Die Textur eines 3D-Modells beschreibt dessen Färbung der Oberfläche, wodurch ihr eine äußerst wichtige Bedeutung für das visuelle Erlebnis beim Betrachten des Modells zukommt. Sie wird in der Regel getrennt von den eigentlichen Geometriedaten in einer separaten Bild-Datei gespeichert. Für jedes Face wird dann durch die sogenannten *Texturkoordinaten* festgelegt, welcher Ausschnitt des Textur-Bildes auf diesem Dreieck angezeigt wird. Pro Face werden also für jeden Eckpunkt zwei Werte gespeichert, durch welche eine eindeutige Position im Bild definiert wird. Da in der Regel bei den meisten Vertices alle angrenzenden Faces die gleichen Texturkoordinaten verwenden, wird dieses Paar von Werten beispielsweise bei OBJ-Dateien nur einmal gespeichert, worauf anschließend bei der Beschreibung der Faces indexbasiert zugegriffen wird.

Manchem Leser stellt sich hier die Frage, ob generell die Verwendung von einem Paar Texturkoordinaten pro Vertex, welches dann für alle angrenzenden Faces verwendet wird, nicht ausreichend wäre. Hier spielt jedoch die Geometrie des 3D-Modells, genauer deren *Topologie*, eine wichtige Rolle. Entspricht diese einer (verzerrten) Ebene, so wäre diese Vereinfachung in der Tat ausreichend. Betrachtet man aber beispielsweise eine Kugel, so lässt sich das zweidimensionale Bild nicht über die Kugel legen, ohne dass eine Kante entsteht, an welcher mindestens zwei Ränder des Bildes aneinandergrenzen. Genau entlang dieser Linie, dem sogenannten *Texture Seam*, befinden sich dann die Vertices, für welche je nach angrenzendem Face verschiedene Texturkoordinaten verwendet werden müssen. Unabhängig von der soeben dargelegten Notwendigkeit dieser Texture Seams lässt sich durch eine sinnvolle Unterteilung der Textur auch die Qualität erhöhen, indem für alle Bereiche des 3D-Modells eine ähnliche Auflösung verwendet wird und durch starke Streckungen oder Stauchungen verursachte Verzerrungen der Textur auf dem Modell minimiert werden.

#### 11.2.3 Kompression der Geometrie

Eine der wichtigsten Herangehensweisen zur Kompression von 3D-Modellen ist die Reduktion von Vertices und damit einhergehend die Verringerung der Anzahl an Faces. Ausgehend von einem hoch aufgelösten Modell mit einer hohen Anzahl an Vertices ist die auf den ersten Blick einfachste Vorgehensweise das Entfernen eines möglichst unwichtigen Vertices. Das dadurch entstehende Loch im Mesh muss anschließend geschlossen werden. Dadurch entsteht jedoch im Allgemeinen ein Face mit einer höheren Ordnung, was oft unerwünscht ist. In diesem Fall muss das entstehende Loch mit Dreiecken gefüllt werden, wofür es keine eindeutige und daher auch unterschiedlich gute Lösungen gibt. Stattdessen verwenden viele Kompressionsverfahren, wie auch der in dieser Kompressions-Komponente verwendete Algorithmus, sogenannte *Edge-Collapse-/Half-Edge-Collapse*-Operationen.



**Abbildung 59:** Beispiel einer Edge-Collapse-Operation. Die Zielposition der beiden an das Edge angrenzenden Vertices ist rot markiert.

Bei einer solchen Edge-Collapse-Operation, wie sie in Abbildung 59 dargestellt ist, werden zwei durch ein Edge verbundene Vertices zu einem neuen Vertex verschmolzen. Dabei degenerieren die an dieses Edge angrenzende Faces und werden entfernt. Zusätzlich lassen sich neben dem kontrahierten Edge noch weitere Edges entfernen. Entspricht der Mesh lokal einer Mannigfaltigkeit, werden durch eine Edge-Collapse-Operation also zwei Faces, drei Edges und ein Vertex entfernt. Vor dem Durchführen einer derartigen Operation muss jedoch überprüft werden, ob die in [Bot+10, S. 118f] erläuterte *Link Condition* erfüllt ist, da andernfalls die Topologie des Mesches durch die Operation verändert werden kann. Entspricht der neue Vertex einem der beiden ursprünglichen Vertices, so spricht man von einem Half-Edge-Collapse.

Es verbleibt jedoch die Frage, welche Paare von Vertices verschmolzen werden sollen. Hierfür wird in [GH97] ein Verfahren vorstellt, das mithilfe von Quadriken jeder möglichen Edge-Collapse-Operation einen Wert für die damit verbundenen Kosten zuweist. Iterativ werden nun die Vertices aus Operationen mit den geringsten Kosten verschmolzen, woraufhin die Kosten der Operationen, die sich auf die umliegenden Vertices beziehen, aktualisiert werden müssen.

Der Algorithmus aus [GH97] ist jedoch nur auf nicht-texturierte Meshes anwendbar. Möchte man dieses Verfahren verwenden, um 3D-Modelle mit Textur zu komprimieren, so müssen einerseits die Verschmelzungsoperationen auch auf die Texturkoordinaten angewandt werden, was insbesondere entlang von Texture Seams eine sehr sorgfältige Implementierung erfordert, andererseits müssen aber auch diese Texturkoordinaten in die Berechnung der Kosten für die jeweilige Operation miteinbezogen werden. Abweichungen in der resultierenden Oberfläche des 3D-Modells haben hier Verzerrungen in der Textur zur Folge. Dies gilt auch für Texture Seams, bei welchen es darüber hinaus nicht vermeiden lässt, dass auch Bereiche der als Textur verwendeten Bilddatei für die Einfärbung der Faces verwendet werden, die im ursprünglichen Modell überhaupt nicht parametrisiert waren und somit beliebigen Inhalt aufweisen können. Es empfiehlt sich daher, die Ränder der Bereiche in der Bilddatei mit einem ähnlichen Farbton wie die eigentliche Textur zu färben, was bei den meisten Textur erzeugenden Program-

men automatisch geschieht. Die daher notwendige Erweiterung des bestehenden Kompressions-Algorithmus auf texturierte Meshes wird in [GH98] vorgenommen, wobei Texture Seams dort kaum behandelt werden. Die für die Kompressions-Komponente erstellte Implementierung behandelt jedoch auch diese Bereiche auf eine sinnvolle Art und Weise.

#### 11.2.4 *Kompression der Textur*

Während die Kompression der Modellgeometrie insbesondere bei texturierten Meshes nichttrivial ist, gestaltet sich die Kompression der Texturdaten relativ einfach. Die Textur wird in einer herkömmlichen Bilddatei gespeichert, deren Dateigröße direkt von der Auflösung des darin gespeicherten Bildes abhängt. Wird die Auflösung des Bildes verringert, was der Funktionsumfang eines jeden brauchbaren Bildbearbeitungsprogramms zulässt, schlägt sich dies auch in der Größe der Texturdatei nieder. Darüber hinaus sind weitere aus der Bildverarbeitung bekannte Kompressionsverfahren einsetzbar, wie zum Beispiel die JPEG-Komprimierung<sup>3</sup>.

Allerdings muss darauf geachtet werden, dass die zusammen mit der Modellgeometrie gespeicherten Texturkoordinaten auch nach der Kompression der Textur wohldefiniert sind. Besonders angenehm ist hier jedoch die Tatsache, dass diese Texturkoordinaten sich nicht auf die Pixel beziehen, sondern stets im Intervall von null bis eins liegen, wobei sich null auf den oberen bzw. linken Rand und eins auf den unteren bzgl. rechten Rand der Texturdatei bezieht. Aus diesem Grund sind die Texturkoordinaten gänzlich unabhängig von der Auflösung der die Textur beinhaltenden Bilddatei und diese Datei kann ohne Modifizierung der Geometriedatei komprimiert werden. Diese Aussage gilt nicht nur für die Anpassung der Auflösung, sondern auch für andere Kompressionsverfahren aus der Bildverarbeitung.

### 11.3 Voraussetzungen

Die Kompressions-Komponente dient zur Kompression von Mediendaten, die im Dateimanagement auf dem Lokalen Anwendungsserver registriert sind. Die Kompressions-Komponente läuft innerhalb eines separaten Docker-Containers auf dem Server, wofür die Software Docker<sup>4</sup> auf dem als Host-System fungierenden Server installiert sein muss. Eine direkte Interaktion des Benutzers mit der Kompressions-Komponente findet in der Regel nicht statt. Es wird jedoch eine Web-Oberfläche zur Administration und Konfiguration des Systems bereitgestellt.

Da die zu komprimierenden Dateien auf dem Host-System gespeichert sind, worauf sowohl von der Medien-Datenbank als auch von der Kompressions-Komponente zugegriffen wird, ist es nicht möglich, das Kompressions-System unabhängig von der Medien-Datenbank auszuführen. Andernfalls würden in der Medien-Datenbank hochgeladene Dateien der Kompressions-Komponente nicht zur Verfügung stehen.

#### 11.3.1 *Hardwarevoraussetzungen*

Die Anforderungen an die Ressourcen, die sich für den LAS bzw. den Docker-Container der Kompressions-Komponente ergeben, hängen in erster Linie von der Größe der zu komprimierenden Mediendateien ab. Während die

---

<sup>3</sup> <https://www.ece.ucdavis.edu/cerl/reliablejpeg/compression/>

<sup>4</sup> <https://www.docker.com/>

Prozessorleistung primär die Dauer der Kompressions-Prozesse bedingt, beschränkt der zur Verfügung stehende Arbeitsspeicher die maximale Größe der Eingabedaten nach oben. Da der Algorithmus zur Kompression von 3D-Modellen kaum Parallelisierungs-Möglichkeiten bietet, werden die Berechnungen derzeit nicht auf mehrere Prozessorkerne oder auf die Grafikkarte verlagert. Experimente während der Implementierung haben gezeigt, dass Modelle mit ca. 8 Mio. Dreiecken auf einem aktuellen gut ausgestatteten Büreurechner (Intel-Core-i7-Prozessor, 32 GB RAM) komprimiert werden können. Da insbesondere der Arbeitsspeicher der limitierende Faktor ist, spielt jedoch der Speicherbedarf der anderen auf dem System laufenden Anwendungen eine wichtige Rolle.

#### *11.3.2 Softwarevoraussetzungen*

Die Kompressions-Komponente wurde in der Programmiersprache Java implementiert, weshalb zur Ausführung ein Java Runtime Environment (JRE) im Docker-Container installiert sein muss. Für die Kompression von Bild- und Texturdateien wird im Hintergrund die Software ImageMagick verwendet. Es ergeben sich also die folgenden beiden Software-Abhängigkeiten:

- Java Runtime Environment<sup>5</sup> (getestet mit Version 1.8.0)
- ImageMagick<sup>6</sup> (getestet mit Version 7.0.8)

Da diese Software-Produkte jedoch bei der Installation des entsprechenden Docker-Containers automatisch installiert werden, muss dies nicht manuell durch den Benutzer vollzogen werden.

#### *11.3.3 Abgrenzungskriterien*

Das Absetzen eines Kompressions-Auftrags wird in der Regel von der Medien-Datenbank-Komponente angestoßen. Dennoch wird eine Benutzeroberfläche zum manuellen Absetzen von Kompressions-Aufträgen zur Verfügung gestellt, die jedoch aufgrund der Notwendigkeit der manuellen Angabe von Identifikatoren keinen hohen Bedienkomfort bietet.

An Mediendaten werden Bilder und 3D-Modelle unterstützt. Andere Mediendaten können mit dieser Komponente nicht komprimiert werden. An Bilddaten können alle Bildformate komprimiert werden, die von der Software ImageMagick unterstützt werden. An 3D-Modellen können texturierte und nicht-texturierte Modelle komprimiert werden, die im OBJ-Format<sup>7</sup> gespeichert werden. Andere Dateien müssen zunächst manuell durch Drittsoftware in das OBJ-Format übersetzt werden. Pro Modell ist aufgrund von Limitierungen in der Medien-Datenbank maximal eine Material- und eine Texturdatei zulässig. Genauere Informationen über die für ein 3D-Modell notwendigen und möglichen Dateien sind in folgender Tabelle 1 dargestellt. Der implementierte Algorithmus wäre jedoch prinzipiell in der Lage, auch Objekte mit mehreren Texturdateien zu komprimieren.

Die in Abschnitt 11.3.1 benannten Hardware-Voraussetzungen gilt es zu beachten.

Bei 3D-Modellen wird vorausgesetzt, dass die Eingabedaten den Voraussetzungen mannigfaltigen Meshes, wie sie in Abschnitt 11.2.1 beschrieben wurden, genügen. Für Modelle, welche diese Eigenschaft nicht aufweisen,

---

5 <https://java.com/de/download/>

6 <https://imagemagick.org/index.php>

7 <http://www.martinreddy.net/gfx/3d/OBJ.spec>

Dateiendung	Informationen	Optional	Notwendigkeit
.obj	Geometriedaten	Nein	Genau eine Datei erforderlich
.mtl	Materialdefinitionen	Ja	Maximal eine Datei möglich
.jpg/.jpeg/.png	Texturdaten	Ja	Maximal eine Datei, nur falls auch MTL-Datei spezifiziert wird

Tabelle 1: Für ein 3D-Modell notwendige und optionale Dateien

wird keine Garantie bezüglich des Ergebnisses des Kompressionsvorgangs gegeben. Mögliche Konsequenzen sind ein Fehlschlagen des Kompressionsvorgangs oder unbrauchbare Resultate, auch wenn ein Fehlschlagen in den Experimenten nicht beobachtet wurde. Das Eingabemodell sollte möglichst ausschließlich aus Dreiecken bestehen. Faces höherer Ordnung werden beim Einlesen naiv in Dreiecke umgewandelt, wobei auch hier keine Garantie bezüglich der Qualität des Ergebnisses gegeben werden kann.

Eine Darstellung der Eingangsdaten oder deren komprimierter Versionen findet innerhalb der Kompressions-Komponente nicht statt. Die Resultate des Kompressionsvorgangs werden lediglich als Datei im System abgelegt.

Die Kompressions-Komponente bietet keinerlei Benutzerauthentifizierung oder sonstige Zugriffsbeschränkungen. Die Komponente muss anderweitig vor dem Zugriff Dritter geschützt werden. Für den Zugriff auf die API und die Konfigurations- und Administrationsoberfläche kann lediglich eine Whitelist mit IP-Adressen hinterlegt werden, für welche der Zugriff exklusiv gewährt wird.

#### 11.4 Funktionsweise

Die Kompressions-Komponente dient zum Abarbeiten von Kompressions-Aufträgen, wobei letztere die Kompression *einer* medialen Repräsentation eines Objekts in *mehreren* Auflösungsstufen umfasst. Die Medien-Dateien stehen dem Kompressions-System über das Dateisystem zur Verfügung. In der Regel wird ein Kompressions-Auftrag direkt nach dem Hochladen einer Medien-Datei in die ViSIT-Medien-Datenbank abgesetzt.

Bei Abarbeitung eines Kompressions-Auftrags werden zunächst die technischen Metadaten aus der Meta-Datenbank abgerufen. Diese technischen Metadaten werden für jede mediale Repräsentation gespeichert und umfassen wichtige Informationen über die dazugehörigen Medien-Dateien und die bestehenden Kompressionsstufen. Sie können außerdem Kontext-Informationen über eine mediale Repräsentation beinhalten, so speichert beispielsweise die Kompressions-Komponente bei 3D-Modellen die Anzahl der Vertices und der Faces. Die technischen Metadaten liegen in der Meta-Datenbank im JSON-Format<sup>8</sup> vor, wobei hierfür die in Listing 22 dargestellte Spezifikation gilt. Im Anschluss wird unterschieden, ob es sich bei der zu komprimierenden Medien-Datei um ein Bild oder ein 3D-Modell handelt. Ausschlaggebend hierfür ist der im Kompressions-Auftrag angegebene MIME-Typ.

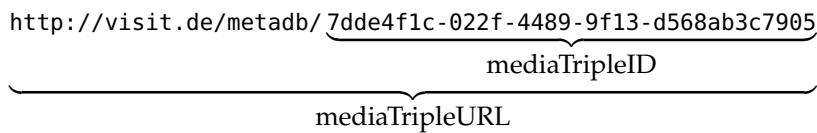
Die beiden in diesem Objekt verwendeten URLs bezeichnen die ID des Metadatums, welches durch die mediale Repräsentation beschrieben wird, in der Meta-Datenbank (`objectTripleURL`) bzw. die ID der digitalen Repräsentation selbst (`mediaTripleURL`), wie in Abbildung Abbildung 73 veranschaulicht wird. Diese beiden Werte haben die Form einer URL und können somit nicht als Dateinamen verwendet werden, wie es in der Medien-Da-

8 <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>

```

1  {
2      "title" : string,
3      "description" : string,
4      "objectTripleID" : string,
5      "objectTripleURL" : string,
6      "mediaTripleID" : string,
7      "mediaTripleURL" : string,
8      "MIMEtype" : string
9      "createDate" : timestamp,
10     "creatorID" : string,
11     "creatorName" : string,
12     "rightholder" : string,
13     "uploader" : string,
14     "files" : {
15         "origin" : {
16             "uploadDate" : timestamp,
17             "accessLevel" : string,
18             "license" : string,
19             "fileSize" : long,
20             "paths" : [string],
21             "fileTypeSpecificMeta" : object
22         },
23         ...
24     }
25 }
```

**Listing 22:** Spezifikation des die technischen Metadaten beinhaltenden JSON-Objekts



**Abbildung 60:** Beziehung von mediaTripleID und mediaTripleURL anhand eines Beispiels

tenbank geschieht. Jedoch verwenden alle in diesem Kontext auftretenden URLs das gleiche Präfix, wodurch die Eindeutigkeit auch bei einer Beschränkung auf das Suffix gewährleistet bleibt, welches sich als Dateiname eignet. Das Suffix der `objectTripleURL` wird als `objectTripleID`, das der `mediaTripleURL` als `mediaTripleID` bezeichnet. Ein Beispiel hierzu ist in Abbildung 60 zu sehen.

Die Dateien werden in der Medien-Datenbank im Format

```
1  objectTripleID.mediaTripleID.compressionLevelID.Dateiendung
```

**Listing 23:** Schema der Dateinamen in der Medien-Datenbank, das ebenfalls durch das Kompressions-System verwendet wird

abgespeichert, worauf ebenfalls durch das Kompressions-System zugegriffen wird. Hierbei bezeichnet `compressionLevelID` die Bezeichnung für die jeweilige Kompressions-Stufe. Für die Originaldatei wird hierbei `origin` verwendet, für 3D-Modelle die Vertexanzahl und für Bilder der in der Konfiguration festgelegte Titel für die jeweilige Auflösungsstufe.

Im folgenden wird auf die Semantik der einzelnen Werte dieses JSON-Objekts eingegangen:

- `title`: Ein vom Benutzer festgelegter Titel des Objekts
- `description`: Eine vom Benutzer festgelegte Beschreibung des Objekts
- `objectTripleID`: Wie oben beschrieben

- **objectTripleURL:** Wie oben beschrieben
- **mediaTripleID:** Wie oben beschrieben
- **mediaTripleURL:** Wie oben beschrieben
- **createDate:** Der UNIX-Timestamp, an dem die Datei in die Medien-Datenbank geladen wurde
- **creatorID:** Die Syncthing-ID des Netzwerkteilnehmers, von dem die Datei hochgeladen wurde
- **creatorName:** Der Name des Netzwerkteilnehmers, von dem die Datei hochgeladen wurde
- **rightholder:** Der Rechteinhaber an der Medien-Datei
- **uploader:** Name der Person, welche die Datei in die Medien-Datenbank geladen hat
- **MIMETYPE:** MIME-Typ der Medien-Datei (z. B. „image/png“ für ein PNG-Bild oder „text/plain“ für ein 3D-Modell)
- **files:** Ein Objekt, das für jede zur Verfügung stehende Kompressions-Stufe (inklusive der Originaldatei) weiterführende Informationen bereitstellt. Diese werden als Objekt in einer Eigenschaft gespeichert, deren Titel der Bezeichnung der jeweiligen Kompressions-Stufe entspricht, also beispielsweise `origin` für die Originaldatei. Dieses Objekt weist das folgende Schema auf
  - **uploadDate:** Der UNIX-Timestamp, an dem diese Kompressions-Stufe erstellt wurde
  - **accessLevel:** Zugriffsberechtigung für die Kompressions-Stufe in der Medien-Datenbank, kann entweder `public`, `private` oder `visit` sein
  - **license:** Lizenz, unter dem die Mediendatei in dieser Kompressions-Stufe verfügbar ist
  - **fileSize:** Größe der Mediendatei in Bytes (bei 3D-Modellen mit mehreren Dateien die Summe aller Dateien)
  - **paths:** Array mit den internen Dateinamen aller zu dieser medialen Repräsentation gehöriger Dateien
  - **fileTypeSpecificMeta:** Weitere unspezifizierte Informationen zu dieser Kompressions-Stufe

**BILDDATEI** Zur Kompression von Bilddateien wird im Hintergrund die Software ImageMagick verwendet. Für jede der in der Konfiguration festgelegten Auflösungsstufen, die kleiner als die Originalgröße des Bildes ist, wird damit eine verkleinerte Version des Bildes erstellt. Laut den technischen Metadaten bereits existierende Auflösungsstufen werden dabei übersprungen. Die neu erstellten Auflösungsstufen werden nun den technischen Metadaten hinzugefügt und in die Meta-Datenbank überführt.

**3D-MODELL** Ein texturiertes 3D-Modell besteht in der Regel aus drei Dateien. Die OBJ-Datei mit den Geometriedaten verweist auf eine MTL-Datei, welche die Materialeigenschaften der Oberfläche beschreibt und insbesondere vorhandene Textur-Dateien referenziert. Beim Hochladen eines 3D-Modells in die Medien-Datenbank werden die Dateien dort automatisch umbenannt. Die Referenzen, die zwischen den Dateien bestehen, werden dabei jedoch nicht angepasst und daher ungültig. Im ersten Schritt der Kompression werden diese Verweise in den Originaldateien repariert, indem sie an die neuen Dateinamen angepasst werden. Auch die zum unkomprimierten Modell gehörenden technischen Metadaten werden angepasst.

Nachdem das unkomprimierte Modell repariert wurde, kann die mit der eigentlichen Kompression begonnen werden. Zunächst werden die Geometriedaten (OBJ-Dateien) und ggf. auch die Material-Datei komprimiert und abgespeichert, wobei auch hier laut den technischen Metadaten bereits existierende Auflösungsstufen übersprungen werden. Nachdem alle Varianten der Geometriedaten erstellt wurden, erfolgt bei texturierten Modellen die Kompression der Textur separat für jede Auflösungsstufe, erneut unter der Zuhilfenahme der Software ImageMagick. Die zuvor erstellten Material-Dateien verweisen bereits auf die nun erstellten Bilddateien. Wie bereits bei den Bilddateien werden abschließend die um die neuen Auflösungsstufen erweiterten technischen Metadaten an die Meta-Datenbank übertragen.

## 11.5 Installation und Steuerung

### 11.5.1 Installation

Zur Installation des Kompressions-Systems muss zunächst über den Befehl

```
1 docker build -t "visitapp/compression" https://github.com/ViSIT-Dev/compressioncontainer.git
```

**Listing 24:** Befehl zum Erstellen des Docker-Images

ein Docker-Image erstellt werden. Nun muss ein Verzeichnis gewählt werden, über das von außen auf die das Kompressions-System betreffende Dateien, wie die Log-Dateien oder die Konfiguration zugegriffen wird. Hier werde beispielsweise das Verzeichnis /etc/visit-compression gewählt. Anschließend kann über den Befehl

```
1 docker run -d --name compression -p 1613:1613 -v visit-p2p-private:/var/www/Private -v /etc/visit-compression:/root/compression visitapp/compression
```

**Listing 25:** Befehl zum Erstellen des Docker-Containers

aus dem soeben erstellten Docker-Image ein ausführbarer Docker-Container generiert werden, sofern visit-p2p-private das Docker-Volume bezeichnet, in welchem die Medien-Dateien gespeichert sind. Die einzelnen Bestandteile dieses Befehls werden im folgenden kurz erläutert:

- `docker run`: Dies ist der Befehl zum Erstellen eines Docker-Images.
- `-d`: Der Container soll im Hintergrund ausgeführt werden.
- `-p 1613:1613`: Der Port 1613 innerhalb des Docker-Containers soll zum Host-System durchgeschleift werden. Sollen andere Ports verwendet werden, muss dieser Bestandteil entsprechend angepasst werden.
- `-v visit-p2p-private:/var/www/Private`: Das beim Einrichten der Medien-Datenbank erzeugte Docker-Volume visit-p2p-private, in dem die Medien-Dateien gespeichert sind, soll innerhalb des Docker-Containers unter dem Pfad /var/www/Private verfügbar sein.

- `-v /etc/visit-compression:/root/compression:` Auf dem Host-System soll unter dem Container-Verzeichnis `/etc/visit-compression` auf die Dateien zugegriffen werden können, welche sich im Docker-Container im Verzeichnis `/root/compression` befinden.
- `visitapp/compression`: Dies ist die gewählte Bezeichnung für das eben erstellte Docker-Image.

Nach dem Erstellen des Docker-Containers wird dieser automatisch gestartet und kann, wie im folgenden Abschnitt erläutert wird, beendet oder neu gestartet werden. Bei gestartetem Container ist die Web-Oberfläche in der Standardkonfiguration unter `http://localhost:1613` erreichbar, während die API unter `http://localhost:1613/api` zur Verfügung steht.

### 11.5.2 Steuerung des Docker-Containers

Der die Kompressions-Komponente beinhaltende Docker-Container kann wie jeder andere Container mit den Befehlen

```
1 docker stop compression
```

**Listing 26:** Kommando zum Beenden des Docker-Containers der Kompressions-Komponente

gestoppt und mit dem Befehl

```
1 docker start compression
```

**Listing 27:** Kommando zum Starten des Docker-Containers der Kompressions-Komponente

wieder gestartet werden. Alle Einstellungen bleiben dabei erhalten. Über das bei der Installation angegebene Verzeichnis (standardmäßig `/etc/visit-compression`) kann auch außerhalb des Containers auf das Kompressions-System betreffende Dateien zugegriffen werden. So kann durch Editieren der Datei `config.ini` die Konfiguration angepasst werden, wobei für ein Aktivieren der aktualisierten Einstellungen ein Neustart des Docker-Containers erforderlich ist. Außerdem kann auf die Log-Datei `compression.log` zugegriffen werden, die insbesondere bei fehlgeschlagenen Kompressions-Aufträgen oder bei unerwartetem Beenden des Kompressions-Systems wichtige Informationen über die zugrunde liegende Ursache liefern kann. Da die Ausgabe dieser Informationen zusätzlich über die Standardausgabe erfolgt, besteht eine weitere Möglichkeit des Zugriffs im Ausführen des Kommandos

```
1 docker logs compression
```

**Listing 28:** Kommando zum Anzeigen der Ausgaben des Kompressions-Systems

Für fortgeschrittene Benutzer, welche Anpassungen direkt im Docker-Container vornehmen möchten, kann über den Befehl

```
1 docker exec -it compression /bin/bash
```

**Listing 29:** Befehl zum Öffnen einer Kommandozeile im Kompressions-Container

eine Kommandozeile geöffnet werden.

### 11.5.3 Steuerung der Kompressions-Komponente

Sämtliche Kommunikation mit der Kompressions-Komponente selbst erfolgt über die bereitgestellte API. Die zur Verfügung stehende Web-Oberfläche greift ebenfalls über diese API auf das System zu. Für Details zur Web-Oberfläche und zur API wird auf die Abschnitte [11.7](#) und [11.8](#) verwiesen.

Die Kompressions-Komponente startet automatisch nach dem Start des entsprechenden Docker-Containers. Je nach Konfiguration wird daraufhin unmittelbar mit dem Abarbeiten von Kompressions-Aufträgen begonnen.

Die Kompressions-Komponente lässt sich über entsprechende API-Aufrufe in unterschiedlichen Modi herunterfahren. Diese Modi umfassen

- das Abarbeiten aller noch ausstehender Aufträge vor dem Beenden,
- das Abarbeiten nur des sich aktuell in Verarbeitung befindlichen Auftrags und
- das sofortige Beenden des Systems ohne Rücksichtnahme auf die ausstehenden Kompressions-Aufträge.

In jedem Fall werden keine weiteren Kompressions-Aufträge angenommen. Ausstehende Kompressionsaufträge bleiben beim erneuten Starten der Komponente erhalten. Jedoch wird ein etwaiger Auftrag, der sich während des Beendens in der Verarbeitung befand, als „Fehlerhaft abgeschlossen“ markiert.

Diese Methode des Herunterfahrens erlaubt im Gegensatz zum Stoppen des Docker-Containers die kontrollierte Handhabung noch ausstehender oder sich in Verarbeitung befindlicher Kompressions-Aufträge. Allerdings wird durch das Herunterfahren der Kompressions-Komponente der sie enthaltende Docker-Container nicht automatisch mit beendet. Dies muss separat durch das Kommando in [Listing 26](#) erfolgen. Wird der Container mit diesem Kommando ohne vorheriges Herunterfahren des Kompressions-Systems ausgeführt, entspricht dies dem sofortigen Beenden des Systems ohne Rücksichtnahme auf die ausstehenden Kompressions-Aufträge.

## 11.6 Konfiguration

Beim Starten des Kompressions-Systems wird automatisch eine Konfigurationsdatei mit den Standardwerten unter dem Pfad

`1 /root/compression/config.ini`

**Listing 30:** Example XML

angelegt, sofern eine solche nicht bereits an dieser Position existiert. Einige der Werte lassen sich nur manuell über diese Datei anpassen. Auf die wichtigsten Konfigurationsmöglichkeiten kann jedoch auch von außen über die API und die Web-Oberfläche sowohl lesend als auch schreibend zugegriffen werden. Beim manuellen Anpassen der Konfigurationsdatei muss darauf geachtet werden, dass innerhalb der Variablenwerte die Zeichen „:“, „=“ und „\“ mit einem vorgestellten Backslash versehen werden müssen, also beispielsweise durch „https\://“. Für Details diesbezüglich wird auf die entsprechende Java-Dokumentation<sup>9</sup> verwiesen. [Tabelle 2](#) gibt Aufschluss über alle Konfigurationsmöglichkeiten, welche in den folgenden Abschnitten detailliert erläutert werden.

### 11.6.1 Verwaltung von Kompressionsaufträgen

Zur Verwaltung der eingehenden Kompressionsaufträge und für die dazu notwendigen Einstellungen des Servers stehen die folgenden Konfigurationsmöglichkeiten zur Verfügung:

<sup>9</sup> [https://docs.oracle.com/javase/7/docs/api/java/util/Properties.html#load\(java.io.Reader\)](https://docs.oracle.com/javase/7/docs/api/java/util/Properties.html#load(java.io.Reader))

Variable	Standard	Web
<b>Verwaltung von Kompressionsaufträgen</b>		
accessWhiteListIps	[127.0.0.1,*]	Ja
apiPort	1613	Ja
archiveDisplayLength	250	Nein
autostart	true	Ja
mediaFileRootDirectory	/var/www/private	Nein
queueMaxLength	5000	Ja
<b>3D-Kompression</b>		
defaultLevels	siehe Beschreibung	Ja
targetSizeBoundaryPenalty	100.0	Nein
targetSizeNormalDifferencePenalization	1000.0	Nein
targetSizeNormalDifferenceThreshold	0.5	Nein
targetSizePartitionPenalization	10.0	Nein
targetSizeQualityThreshold	0.3	Nein
textureLimits	[5000, 50000]	Ja
textureSizes	[1024, 2048, 8192]	Ja
<b>Bildkompression</b>		
imageCompressionLevels	siehe Beschreibung	Ja
<b>Schnittstelle Meta-Datenbank</b>		
metadbApiAuthString	Basic XX...XX\=\=	Nein
metadbApiEndpointFetchUrl	https\://DOMAIN/metadb-rest-api/digrep/media	Nein
metadbApiEndpointSendUrl	https\://DOMAIN/metadb-rest-api/digrep/media	Nein
metadbApiMediaUidPrefix	http\://DOMAIN/metadb/	Nein

**Tabelle 2:** Überblick über alle Konfigurationsmöglichkeiten der Kompressionskomponente

**ZUGRIFFSBESCHRÄNKUNG** Der Wert von `accessWhiteListIps` beschreibt, über welche Rechner auf die API oder die Web-Oberfläche zugegriffen werden kann, indem die IP-Adressen dieser Rechner angegeben werden können. Generell sollte die Absicherung allerdings auf eine andere Art von außen vorgenommen werden. Die IP-Adressen werden in eckigen Klammern und durch Kommata getrennt notiert. Befindet sich ein Asterisk („\*“) in dieser Auflistung, wird der Zugriff für alle Rechner autorisiert. Der Standardwert für diese Eigenschaft ist `[127.0.0.1, *]`, wodurch keine Beschränkung des Zugriffs erfolgt.

**SERVERTPORT** Der Wert für die Variable `apiPort` muss einer Ganzzahl im Intervall von 1 bis 65535 entsprechen und legt den Port fest, über welchen sowohl auf die API als auch auf die Web-Oberfläche der Kompressions-Komponente zugegriffen werden kann. Der Standardwert für diese Variable ist 1613.

**LÄNGE DER ARCHIV-ANZEIGE** Über den Parameter `archiveDisplayLength` lässt sich festlegen, wie viele Einträge in dem über die Web-Oberfläche zugänglichen Archiv der letzten Kompressions-Aufträge angezeigt werden sollen. Auch dieser Wert muss einer positiven Ganzahl entsprechen und beträgt standardmäßig 250.

**AUTOMATISCHER START** Der Wert der Eigenschaft `autostart` kann entweder `true` oder `false` entsprechen und legt fest, ob direkt nach dem Starten der Kompressions-Komponente mit der Abarbeitung eingehender Kompressions-Aufträge begonnen werden soll, wobei dieser Fall dem Standard entspricht.

**HAUPTVERZEICHNIS FÜR MEDIEN-DATEIEN** Die Konfigurationsoption `mediaFileRootDirectory` legt fest, in welchem Verzeichnis innerhalb des Docker-Containers der Kompressions-Komponente die zu komprimierenden Mediendateien gespeichert sind. Etwaige Unterverzeichnisse können für jeden Kompressions-Auftrag separat angegeben werden. In ebendiesem Verzeichnis werden die komprimierten Versionen der Dateien nach der Aufführung des Auftrags auch abgelegt. Der Standardort für diese Dateien ist `/var/www/private`.

**MAXIMALE LÄNGE DER AUFRAGSLISTE** Mithilfe der Option `queueMaxLength` lässt sich eine Beschränkung für die Länge der Liste der unbearbeiteten Kompressions-Aufträge festlegen. Sollte dieser Wert erreicht sein, werden etwaige eingehenden Aufträge abgewiesen. Ist der Wert auf 0 festgelegt, so erfolgt keine Beschränkung der Liste. Der Standardwert beträgt 5000.

#### 11.6.2 3D-Kompression

Folgende Optionen stehen zur Konfiguration des Kompressions-Algorithmus für 3D-Modelle zur Verfügung:

**STANDARD-KOMPRESSIIONSSSTUFEN** Der Wert für die Option `defaultLevels` legt fest, welche Auflösungsstufen für 3D-Modelle standardmäßig erzeugt werden sollen. Jede Auflösungsstufe wird dabei durch eine Ganzzahl beschrieben, welche die gewünschte Anzahl an Vertices des komprimierten Modells festlegt. Diese Stufen werden durch Kommata voneinander getrennt und insgesamt von eckigen Klammern umrahmt, wie auch durch den in Listing 31 aufgeführten Standardwert

```
1 [500, 1000, 5000, 20000, 50000, 200000, 500000, 2000000, 5000000, 20000000, 50000000]
```

Listing 31: Standardwert für die Konfigurationsoption defaultLevels

deutlich wird. Hat das ursprüngliche Modell bereits weniger Vertices als die angestrebte Anzahl einer Auflösungsstufe, so wird diese Stufe übersprungen anstatt ein Modell mit einer größeren Anzahl an Vertices zu erzeugen.

**BESTRAFUNGEN VON ABWEICHUNGEN AM RAND** Der Gleitkommawert für die Eigenschaft `targetSizeBoundaryPenalty` ist nur für die Kompression von Meshes mit Rand, die also kein vollständiges Modell eines realen Objekts darstellen, relevant. Er legt fest, mit welchem Gewicht dieser Rand des ursprünglichen Modells beibehalten werden soll. Bei einem hohen Wert dieser Eigenschaft werden durch die Kompression kaum Änderungen an diesen Rändern vorgenommen, während bei einem niedrigen Wert viele Edge-Collapse-Operationen genau dort ausgeführt werden. Der Standardwert beträgt `100.0`.

**BESTRAFUNGEN BEI ABWEICHUNGEN DER OBERFLÄCHENNORMALE** Die Oberflächennormale ist einen Richtung, welche die Ausrichtung eines Face beschreibt und somit senkrecht zur Ebene, welche durch das Face erzeugt wird, verläuft. Über die beiden Eigenschaften `targetSizeNormalDifferencePenalization` und `targetSizeNormalDifferenceThreshold` lässt sich festlegen, in welchem Ausmaß starke Veränderungen dieser Normalen vermieden werden sollen. Der Wert von `targetSizeNormalDifferenceThreshold` beschreibt dabei, ab welcher Abweichung der durch eine Edge-Collapse-Operation verursachten Veränderung von Oberflächennormalen eine Bestrafung erfolgt, welche die Ausführung dieser Operation unwahrscheinlicher macht, indem die Kosten der Operation mit dem Faktor `targetSizeNormalDifferencePenalization` multipliziert werden. Perfekt übereinstimmende Normalen entsprechen dabei dem Wert `1.0`, während zueinander senkrechte Normalen durch den Wert `0.0` beschrieben werden. Entsteht durch eine Edge-Collapse-Operation ein Face, dessen Oberflächennormale eine Übereinstimmung mit der ursprünglichen Normale hat, die unter dem Wert von `targetSizeNormalDifferenceThreshold` liegt, so wird dieser Bestrafungsfaktor aktiviert.

**BESTRAFUNGEN ENTLANG VON TEXTURE SEAMS** Wie in Abschnitt [11.2.3](#) erläutert wurde, sorgen Edge-Collapse-Operationen entlang von Texture Seams nicht nur zu Verzerrungen in der Textur, sondern können auch die Darstellung von eigentlich nicht parametrisierten Bereichen in der Texturdatei zur Folge haben, was es möglichst zu vermeiden gilt. Aus diesem Grund werden Kompressionsoperationen entlang von Texture Seams mit einem Faktor bestraft, der sich im Wesentlichen aus dem Produkt des Quadrats der an die kollabierende Kante angrenzenden Texturpartitionen und des Werts der Eigenschaft `targetSizePartitionPenalization` ergibt, wobei letzterer standardmäßig auf `10.0` festgelegt ist.

**SCHWELLWERT FÜR DIE BEGÜNSTIGUNG WOHLGEFORMTER FACES** Bei der Erzeugung von Meshes werden in der Regel möglichst gleichmäßige Faces angestrebt, während hingegen sehr lange aber dünne Dreiecke unerwünscht sind. Als Maß für diese „Schönheit“ eines Faces wird der Quotient aus Fläche und maximaler Seitenlänge betrachtet. Die Kosten einer Edge-Collapse-Operation werden durch die minimale Qualität der durch diese Operation

Vertexanzahl	Texturauflösung
1000	1024 x 1024
5000	2048 x 2048
10000	2048 x 2048
75000	8192 x 8192

Tabelle 3: Beispiele für die sich bei unterschiedlichen Vertexanzahlen ergebenden Texturauflösungen bei der Standardkonfiguration.

entstehenden Faces dividiert, wodurch allerdings bei sehr wohlgeformten Faces und demzufolge hoher Qualität die Kosten der gesamten Operation unerwünscht stark sinken können. Aus diesem Grund lässt sich durch den Parameter `targetSizeQualityThreshold` der Divisor nach oben beschränken, wobei der Standardwert 0.3 beträgt und somit auch nicht perfekt geformte Dreiecke ohne Erhöhung der Kosten zulässt.

**TEXTURKOMPRESSION** Durch die Textur eines 3D-Modells kann ein relevanter Anteil des insgesamt notwendigen Speicherbedarfs verursacht werden, weshalb es auch diesen Bestandteil zu komprimieren gilt. Dies sollte je nach gewählter Vertexanzahl in einem ähnlichen Ausmaß geschehen. Allerdings können viele Programme nur Texturdateien mit einer Zweierpotenz als Seitenlänge verarbeiten oder erweitern die gegebene Textur durch Padding zu einer solchen Größe. Aus diesem Grund ist eine pixelgenaue Wahl der Texturauflösung nur bedingt sinnvoll. Stattdessen kann einem bestimmten Intervall an Vertexanzahlen eine bestimmte Texturauflösung zugewiesen werden. Dies lässt sich über die beiden Parameter `textureLimits` und `textureSizes` konfigurieren, wobei beide Parameter Listen mit ganzzahligen Einträgen als Werte akzeptieren. Die Einträge in diesen Listen werden wie bereits bei anderen Konfigurationsoptionen durch Kommata getrennt, während die ganze Liste durch eckige Klammern eingefasst wird. Zu beachten ist jedoch, dass die Anzahl an Einträgen in `textureSizes` stets um genau eins größer sein muss, als die Anzahl der Einträge in `textureLimits`.

Hat ein Modell nun weniger Vertices, als der erste Eintrag in der Schwellwert-Liste `textureLimits`, so wird eine Textur erzeugt, deren Auflösung dem ersten Eintrag in `textureSizes` entspricht. Hat es stattdessen mindestens so viele Vertices wie der erste Eintrag in `textureLimits`, jedoch weniger Vertices als der zweite Eintrag in `textureLimits` vorgibt, so wird eine Textur mit einer Seitenlänge entsprechend des zweiten Eintrags in `textureSizes` erzeugt. Diese Regel gilt entsprechend für jeden Eintrag in `textureLimits`. Standardmäßig sind die Schwellwerte `textureLimits` festgelegt durch [5000, 50000], während die dazugehörigen Auflösungen durch [1024, 2048, 8192] gegeben sind. Bei Bedarf lässt sich diese Konfiguration feiner gestalten und dabei insbesondere auch ein Intervall festlegen, das Texturdateien mit einer Seitenlänge von 4096 Pixeln erzeugt. Tabelle 3 veranschaulicht die resultierenden Texturauflösungen abhängig von unterschiedlichen Vertexanzahlen für die Standardkonfiguration.

Ist die Auflösung der ursprünglichen Texturdatei jedoch kleiner als die sich bei der Kompression ergebende Größe, so wird die Textur nicht vergrößert, sondern es wird die Datei in der ursprünglichen Auflösung unverändert übernommen.

Name	Wertebereich	Beispiel
maxWidth	Positive Ganzzahl	1920
maxHeight	Positive Ganzzahl	1080
title	A bis Z, a bis z, Binde-, Unterstriche	„FullHD“

Tabelle 4: Pro Kompressionsstufe notwendige Parameter bei der Bildkompression

### 11.6.3 Bildkompression

Ähnlich wie bei den 3D-Modellen sollen auch bei Bildern unterschiedliche Auflösungen vorgehalten werden, um für verschiedene Anwendungsfälle eine passende Größe zur Verfügung zu haben. Welche Auflösungen bei der Kompression einer Bilddatei erstellt werden sollen, wird durch die Konfigurationsoption `imageCompressionLevels` festgelegt. Aufgrund der Komplexität dieses Parameters muss dieser im JSON-Format<sup>10</sup> angegeben werden. Der Wert der Konfigurationsoption muss einem Array aus Objekten entsprechen, wobei jedes Objekt die in Tabelle 4 dargestellten Eigenschaften aufzuweisen hat. Jeder Eintrag des Arrays definiert auf diese Art eine Kompressions-Stufe für ein Bild. Ist das Bild in mindestens einer Dimension kleiner als eine bestimmte Kompressionsstufe, so wird die Erzeugung einer Version mit dieser Auflösung komplett übersprungen, es wird also im Gegensatz zur Texturkompression nicht die ursprüngliche Auflösung verwendet. Der Standardwert für diesen Parameter ist in Listing 32 dargestellt.

```
1  [{"maxWidth"\:3840,"maxHeight"\:2160,"title"\:"UHD"}, {"maxWidth"\:1920,"maxHeight"\:1080,"
  title"\:"FullHD"}, {"maxWidth"\:800,"maxHeight"\:600,"title"\:"Mittel"}, {"maxWidth"\:
  120,"maxHeight"\:120,"title"\:"Klein"}]
```

Listing 32: Standardwert für die Konfigurationsoption `imageCompressionLevels`

### 11.6.4 Schnittstelle Meta-Datenbank

Um die während des Kompressionsvorgangs erstellen Modelle im ViSIT-System zu registrieren, müssen die zu der Mediendatei gehörigen technischen Metadaten, die in Listing 22 spezifiziert wurden, in der Meta-Datenbank aktualisiert werden. Hierzu muss auf diese Datenbank zugegriffen werden können, wofür die nachfolgend erläuterten Parameter anzugeben sind. Alle in diesem Abschnitt angegebenen Konfigurations-Optionen müssen nach der Installation angepasst werden, um die Integration in das Gesamtsystem zu ermöglichen. In sämtlichen Standardwerten ist `DOMAIN` durch die jeweilige Domain, über welche auf die Meta-Datenbank zugegriffen werden kann, zu ersetzen.

**AUTHORIZIERUNG ZUM ZUGRIFF AUF DIE METADATEN** Um die Meta-Datenbank durch unbefugten Zugriff zu schützen, müssen sich zugelassene Benutzer oder Systeme authentifizieren. Diese Authentifizierung erfolgt durch das *HTTP Basic Authentication*-Verfahren<sup>11</sup>. Der dafür notwendige Base64-codierte Authentifizierungs-String, dem die Zeichenfolge `Basic` vorausgeht, muss in der Konfigurationsoption `metadbApiAuthString` angegeben werden, welche nach der Installation der Kompressions-Komponente auf einen gültigen Wert zu setzen ist. Der (ungültige) Standardwert ist in Listing 33 dargestellt.

```
1  Basic xxxxxxxxxxxxxxxxxxxxxxxxx\=
```

<sup>10</sup> <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>

<sup>11</sup> <https://tools.ietf.org/html/rfc2617>

---

**Listing 33:** Standardwert für die Konfigurationsoption `metadbApiAuthString`

**ENDPUNKT ZUM ABRUFEN DER METADATEN** In der Konfigurationsoption `metadbApiEndpointFetchUrl` kann der Endpunkt der API zur Meta-Datenbank spezifiziert werden, über den technische Metadaten abgerufen werden können. Dieser Wert ist standardmäßig festgelegt auf `https://DOMAIN/metadb-rest-api/digrep/media`.

**ENDPUNKT ZUM SCHREIBEN DER METADATEN** Mithilfe der Konfigurationsoption `metadbApiEndpointSendUrl` kann der Endpunkt der API zur Meta-Datenbank spezifiziert werden, über den technische Metadaten gespeichert werden können. Auch hier wird der Wert `https://DOMAIN/metadb-rest-api/digrep/media` als Standard verwendet, da er in der Regel mit dem Wert für `metadbApiEndpointFetchUrl` übereinstimmt.

**PRÄFIX DER MEDIEN-UIDS** Jede in der Meta-Datenbank registrierte mediale Repräsentation wird durch eine eindeutige sogenannte UID identifiziert. Jede UID beginnt mit einem allen Mediendateien gemeinsamen Präfix, auf welches ein für das Objekt spezifischer alphanumerischer Identifikator folgt. Da zum Starten eines Kompressionsvorgangs durch die Medien-Datenbank nur das alphanumerische Suffix übermittelt wird, muss in der Konfigurationsoption `metadbApiMediaUidPrefix` das konstante Präfix festgelegt werden. Der Standardwert, der gegeben ist durch `http://DOMAIN/metadb/`, muss vor dem ersten Start der Kompressionskomponente geeignet angepasst werden.

## 11.7 Zugriff über die Web-Oberfläche

Auf die Web-Oberfläche kann über jeden Browser zugegriffen werden, indem eine Verbindung mit dem Docker-Container auf dem in der Konfiguration festgelegten Port aufgebaut wird. Auf dem Server kann beispielsweise in der Standardkonfiguration, und sofern der Port durch die Docker-Konfiguration nicht umgeleitet wird, durch die Eingabe der in [Listing 34](#) dargestellten Zeichenfolge die Startseite der Kompressions-Komponente aufgerufen werden.

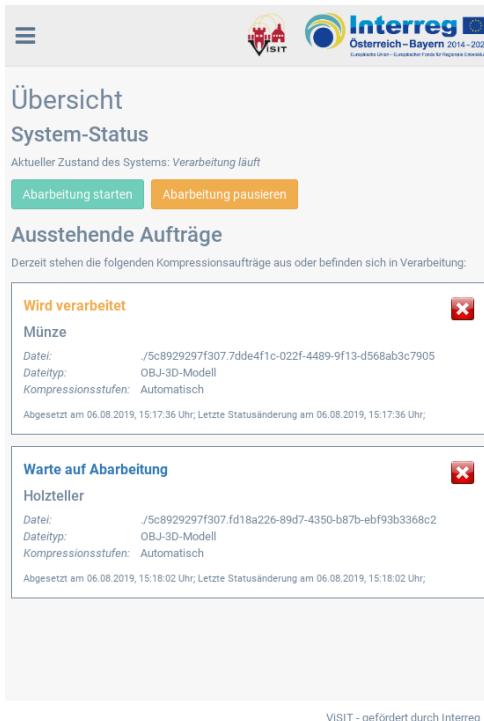
```
1 http://localhost:1613
```

**Listing 34:** Zugriff auf die Web-Oberfläche der Kompressions-Komponente

Die Web-Oberfläche bietet vier verschiedene Ansichten, welche im Folgenden erläutert werden. Zwischen diesen Ansichten kann über die Navigation in der linken Seitenleiste bzw. über das Aufklapp-Menü auf der linken Seite gewechselt werden.

### 11.7.1 Startseite

Die Startseite hat zwei wichtige Funktionen. Zum einen erlaubt sie das Pausieren oder Fortführen der Auftragsverarbeitung, zum anderen gibt sie einen Überblick über die aktuell ausstehenden Kompressions-Aufträge, die sich dort auch abbrechen lassen. Wie in [Abbildung 61](#) deutlich wird, werden zu jedem ausstehenden Auftrag mehrere Informationen angezeigt. Diese umfassen neben dem aktuellen Status des Auftrags der Titel, UID und Dateityp des Objekts, die gewünschten Kompressionsstufen, sowie die Zeitpunkte der Absetzung des Auftrags und der letzten Statusänderung. Über



**Abbildung 61:** Startseite, unter anderem mit einem Überblick über die laufenden und anstehenden Kompressions-Aufträge

die rote Schaltfläche im rechten oberen Bereich eines Auftrags wird dieser nach dem Bestätigen einer Sicherheitsabfrage abgebrochen, sofern sich dieser noch nicht in Verarbeitung befindet. Kompressionsaufträge, die bereits ausgeführt werden, lassen sich nicht abbrechen. Die Auflistung der ausstehenden Kompressionsaufträge erfolgt aufsteigend nach dem Zeitpunkt der Absetzung.

#### 11.7.2 Neuer Auftrag

Über diese Seite, welche in [Abbildung 62](#) dargestellt ist, lassen sich manuelle Kompressionsaufträge absetzen, wobei dies normalerweise direkt beim Hochladen der Medien-Datei von der Medien-Datenbank übernommen wird. Soll dennoch manuell ein Auftrag abgesetzt werden, so müssen in dieser Ansicht die UID des Metadatums (Objekt-Identifikator, objectID) und die UID der digitalen Repräsentation (Medien-Identifikator, mediaID), deren Beziehung in Abbildung [73](#) veranschaulicht wird, angegeben werden. Zusätzlich kann ein Unterverzeichnis angegeben werden (Basis-Pfad), in dem sich die zu komprimierende Datei befindet, wobei dieser Pfad stets in Bezug auf das in der Konfiguration angegebene mediaFileRootDirectory interpretiert wird. Außerdem muss ein Titel für die Datei angegeben werden, der beliebig gewählt werden kann und lediglich zur leichteren Identifizierung des Auftrags innerhalb des Kompressions-Systems dient. Des Weiteren ist der Dateityp zu spezifizieren, wobei hier die Optionen PNG-Bild, JPEG-Bild und OBJ-3D-Modell zur Auswahl stehen.

Für den Fall, dass ein 3D-Modell komprimiert wird, können im darauf folgenden Abschnitt die gewünschten Kompressionsstufen festgelegt werden. Zum Einen steht die Option „Automatisch“ zur Verfügung, die stan-

Abbildung 62: Ansicht zum Absetzen eines neuen Kompressions-Auftrags

dardmäßig ausgewählt ist und alle in der Konfiguration spezifizierten Standard-Kompressionsstufen umfasst. Weitere oder alternative Auflösungsstufen können durch die Wahl des Eintrags „Feste Größe“ und die Angabe der gewünschten Vertex-Anzahl hinzugefügt werden. Sämtliche derzeit angegebenen Auflösungsstufen werden aufgelistet und lassen sich mit Klick auf das Mülltonnen-Symbol wieder entfernen.

#### 11.7.3 Archiv

Über diese Ansicht lässt sich ein Überblick über die zuletzt abgearbeiteten Kompressions-Aufträge erhalten, unabhängig davon, ob die Ausführung erfolgreich war oder fehlgeschlagen ist. Die Einträge werden nach der letzten Statusänderung absteigend sortiert, wobei die Anzahl der dargestellten Einträge in der Konfiguration festgelegt werden kann. Ein Beispiel für diese Ansicht ist in [Abbildung 63](#) zu sehen.

#### 11.7.4 Einstellungen

Auf dieser Seite lassen sich die wichtigsten Konfigurations-Optionen anpassen, wie in [Abbildung 64](#) zu sehen ist. Alle Einstellungen, die auf dieser Seite nicht verfügbar sind, müssen manuell in der Konfigurationsdatei angepasst werden. Für Details zu den einzelnen Optionen wird auf Abschnitt [11.6](#) verwiesen.

Die über die Web-Oberfläche verfügbaren Konfigurationsmöglichkeiten werden im Folgenden aufgeführt, wobei auch die Entsprechung in der Konfigurationsdatei benannt wird. Sämtliche Angaben werden jedoch erst durch das Betätigen der Schaltfläche „Einstellungen speichern“ am Ende der Seite zum Server übertragen und übernommen.

- *Portnummer der API-Schnittstelle:* `apiPort`
- *Maximale Länge der Auftragsliste:* `queueMaxLength`

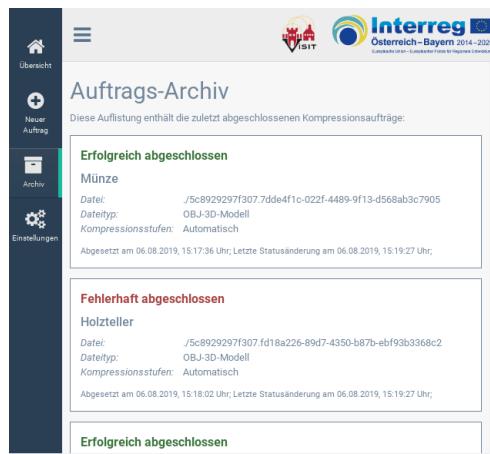


Abbildung 63: Archiv-Ansicht mit einem Überblick über die ausgeführten Kompressions-Aufträge

- *Abarbeitung automatisch beim Starten des Servers beginnen: autostart*
- *API-Zugriffsberechtigte IP-Adressen: accessWhiteListIps.* Die berechtigten IP-Adressen sind einzeln anzugeben und über die Schaltfläche „IP-Adresse hinzufügen“ zu bestätigen. Durch einen Klick auf das Mülltonnen-Symbol können Einträge wieder entfernt werden. Auch hier entspricht die Angabe eines Asterisks („\*“) einer Zugriffsgenehmigung für alle Hosts.
- *Standard-Kompressionsstufen für 3D-Modelle: defaultLevels.* Die Anzahl der Vertices der komprimierten Modelle, die bei der Kompression von 3D-Modellen standardmäßig erstellt werden sollen, sind an dieser Stelle einzeln anzugeben und über die Schaltfläche „Kompressionsstufe hinzufügen“ hinzuzufügen. Auch hier können einzelne Kompressionsstufen durch Betätigen des Mülleimer-Symbols entfernt werden.
- *Kompression der Textur von 3D-Modellen: textureLimits, textureSizes.* Die Anzahl der Schwellwerte und demzufolge unterschiedlicher Texturauflösungen lässt sich über die beiden Schaltflächen „Unterscheidung hinzufügen“ bzw. „Unterscheidung entfernen“ kontrollieren. Die Texturgröße ist als Anzahl der Pixel pro Dimension zu verstehen.
- *Aktionen für die Kompression von Bildern: imageCompressionLevels.* Die für eingehende Bild-Kompressions-Aufträge zu erstellenden Auflösungsstufen sind hier aufgelistet, wobei sich einzelne Einträge durch das Betätigen des Mülleimer-Symbols entfernen lassen. Weitere Kompressionsstufen lassen sich durch die Angabe eines beliebigen Titels, der nur aus Groß- oder Kleinbuchstaben, Ziffern und Binde- oder Unterstrichen bestehen darf, sowie der maximalen Breite und maximalen Höhe des komprimierten Bildes in Pixeln und anschließendes Betätigen der Schaltfläche „Kompressionsstufe hinzufügen“ hinzufügen.

#### 11.8 Zugriff über die API

Sämtliche Funktionen der API stehen bei der standardmäßigen Konfiguration unter dem Basispfad <http://localhost:1613/api> zur Verfügung. Die

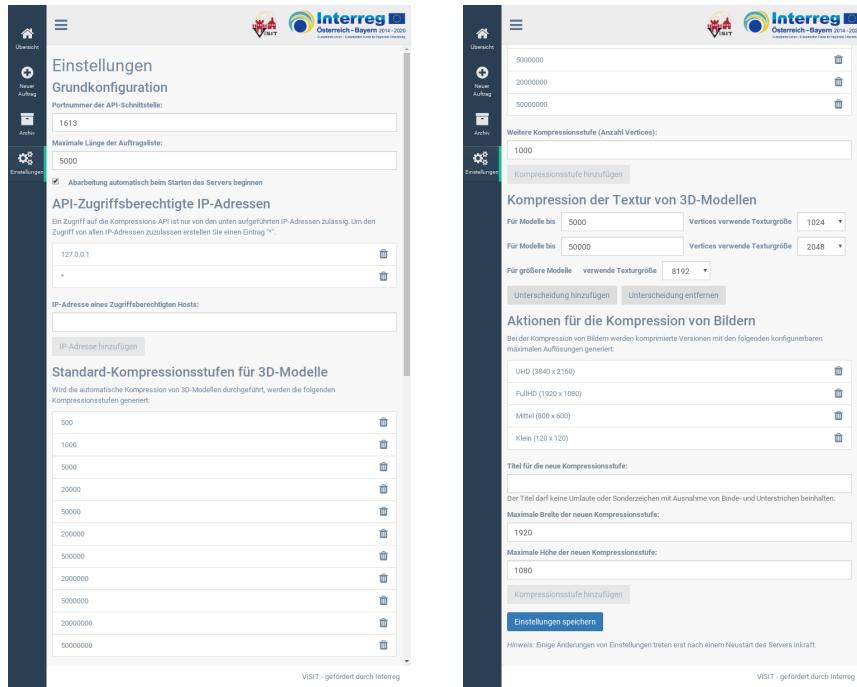


Abbildung 64: Einstellungsmöglichkeiten über die Web-Oberfläche

API bietet unterschiedliche Module an, auf deren Funktionen in diesem Abschnitt eingegangen wird. POST-Parameter sind stets als JSON-Objekt übergeben, ebenso erfolgt die Antwort in Form eines JSON-Objekts.

#### 11.8.1 Aktuelle Kompressions-Aufträge

Für Informationen zu den IDs, dem MIME-Typen oder den Kompressions-Stufen wird auf Abschnitt 11.4 verwiesen. Beim Absetzen wird jedem Kompressions-Auftrag eine ID zugewiesen, welche zum Löschen des Auftrags angegeben werden muss. Der Status eines Auftrags kann einen der folgenden Werte annehmen:

- ENQUEUED: Der Auftrag wurde abgesetzt, mit der Abarbeitung wurde jedoch noch nicht begonnen.
- PROCESSING: Der Auftrag wurde abgesetzt und befindet sich derzeit in Verarbeitung.
- ERROR: Die Verarbeitung des Auftrags wurde fehlerhaft abgeschlossen.
- COMPLETED: Die Verarbeitung des Auftrags wurde erfolgreich abgeschlossen.

**Funktion:** Absetzen eines Kompressions-Auftrags

**HTTP-Methode:** POST

**Pfad:** /jobs/dispatch

**POST-Paramter:**

```

1  {
2      basePath : string, /* Unterverzeichnis, in welchem die Datei abgelegt ist, ansonsten ""
3      */
4      objectUid : string, /* objectTripleID (nicht objectTripleURL) der Medien-Datei */
5      mediaUid : string, /* mediaTripleID (nicht mediaTripleURL) der zu komprimierenden Medien-
6      Datei */

```

```

5   title : string,      /* Beliebiger Titel zur Identifizierung des Kompressions-Auftrags */
6   mimeType : string,  /* MIME-Typ der zu komprimierenden Datei */
7   levels : [string],  /* Array mit den Bezeichnern aller gewuenschten Kompressions-Stufen */
8 }
```

**Listing 35:** POST-Parameter zum Absetzen eines Kompressions-Auftrags

**Antwort:**

```

1 {
2   success : boolean,
3   message : string
4 }
```

**Listing 36:** Antwort auf das Absetzen eines Kompressions-Auftrags

**Funktion:** Auflisten aller noch nicht abgeschlossenen Kompressions-Aufträge

**HTTP-Methode:** GET

**Pfad:** /jobs/dispatch

**Antwort:**

```

1 {
2   success : boolean,
3   message : string,
4   items : [
5     job : {
6       basePath : string, /* Unterverzeichnis, in welchem die Datei abgelegt ist,
7       ansonsten "" */
8       objectUid : string, /* objectTripleID (nicht objectTripleURL) der Medien-Datei */
9       mediaUid : string, /* mediaTripleID (nicht mediaTripleURL) der Medien-Datei */
10      title : string,    /* Beliebiger Titel zur Identifizierung des Kompressions-
11      Auftrags */
12      mimeType : string, /* MIME-Typ der zu komprimierenden Datei */
13      levels : [string], /* Array mit den Bezeichnern aller gewuenschten Kompressions-
14      Stufen */
15      },
16      receivedOn : timestamp, /* UNIX-Timestamp, an dem der Auftrag abgesetzt wurde */
17      id : int,               /* Vom Kompressions-System vergebene ID fuer den Auftrag
18    }
19  ]
```

**Listing 37:** Antwort auf das Auflisten nicht abgeschlossener Kompressions-Aufträge

**Funktion:** Löschen eines Kompressions-Auftrags

**HTTP-Methode:** DELETE

**Pfad:** /jobs/cancel/{ID}, mit {ID} der ID des Kompressions-Auftrags

**Antwort:**

```

1 {
2   success : boolean,
3   message : string
4 }
```

**Listing 38:** Antwort auf das Löschen eines Kompressions-Auftrags

### 11.8.2 Abgeschlossene Kompressions-Aufträge

**Funktion:** Auflisten der letzten abgeschlossenen Kompressions-Aufträge

**HTTP-Methode:** GET

**Pfad:** /archive/jobs/

**Antwort:**

```

1 {
2   success : boolean,
3   message : string,
```

```

4      items : [{  
5          job : {  
6              basePath : string, /* Unterverzeichnis, in welchem die Datei abgelegt ist,  
7              ansonsten "" */  
8                  objectUid : string, /* objectTripleID (nicht objectTripleURL) der Medien-Datei */  
9                  mediaUid : string, /* mediaTripleID (nicht mediaTripleURL) der Medien-Datei */  
10                 title : string, /* Beliebiger Titel zur Identifizierung des Kompressions-  
11                 Auftrags */  
12                 mimeType : string, /* MIME-Typ der zu komprimierenden Datei */  
13                 levels : [string], /* Array mit den Bezeichnern aller gewuenschten Kompressions-  
14                 Stufen */  
15             },  
16             receivedOn : timestamp, /* UNIX-Timestamp, an dem der Auftrag abgesetzt wurde */  
17             id : integer, /* Vom Kompressions-System vergebene ID fuer den Auftrag */  
18         */  
19         state : string, /* Aktueller Status des Auftrags, kann entweder ENQUEUED,  
20                     PROCESSING, ERROR oder COMPLETED sein */  
21         lastStateChange : timestamp /* UNIX-Timestamp der letzten Statusaenderung des  
22         Auftrags */  
23     ]  
24 }

```

**Listing 39:** Antwort auf das Auflisten abgeschlossener Kompressions-Aufträge

#### 11.8.3 Status des Kompressions-Systems

Der Zustand des Kompressions-Systems kann einen der folgenden Werte annehmen:

- **STARTUP:** Das System wird hochgefahren, mit der Abarbeitung von Kompressions-Aufträgen wurde noch nicht begonnen.
- **RUNNING:** Das System ist hochgefahren und bearbeitet Kompressions-Aufträge oder ist bereit dazu.
- **PAUSED:** Das System ist hochgefahren, die Abarbeitung von Kompressions-Aufträgen wurde jedoch pausiert.
- **SHUTTINGDOWN:** Das System wird heruntergefahren, weshalb das Absetzen weiterer Aufträge nicht möglich ist. Jedoch werden noch Kompressions-Aufträge verarbeitet.
- **SHUTDOWN:** Das System wurde heruntergefahren. Es können keine weiteren Aufträge abgesetzt oder verarbeitet werden.

Zum Setzen des Zustands des Kompressions-Systems kann einer der folgenden Werte verwendet werden:

- **RUN:** Setze den Zustand auf RUNNING und beginne mit der Bearbeitung von Kompressions-Aufträgen. Dieses Kommando ist nur in den Zuständen PAUSED oder STARTUP zulässig.
- **PAUSE:** Setze den Zustand auf PAUSED und pausiere damit die Abarbeitung der Kompressions-Aufträge nach dem Abschließen des aktuellen Auftrags. Dieses Kommando ist nur im Zustand PAUSED zulässig.
- **SHUTDOWN\_PROCESS\_QUEUE:** Setze den Zustand auf SHUTTINGDOWN, verarbeite aber vor dem Herunterfahren die gesamte Auftragsliste. Dieses Kommando ist nur im Zustand STARTUP, RUNNING oder PAUSED zulässig.
- **SHUTDOWN\_IMMEDIATELY:** Setze den Zustand auf SHUTTINGDOWN, schließe aber vor dem Herunterfahren den sich aktuell in Verarbeitung befindlichen Auftrag ab. Dieses Kommando ist nur im Zustand STARTUP, RUNNING oder PAUSED zulässig.

- **KILL:** Setze den Zustand auf SHUTDOWN, fahre das System herunter ohne Rücksicht auf ausstehende oder sich in Verarbeitung befindliche Kompressions-Aufträge.

**Funktion:** Abrufen des Systemzustands

**HTTP-Methode:** GET

**Pfad:** /control/state

**Antwort:**

```

1  {
2      success : boolean,
3      message : string,
4      state : string    /* {STARTUP | RUNNING | PAUSED | SHUTTINGDOWN | SHUTDOWN} */
5 }
```

Listing 40: Antwort auf das Abrufen des Systemzustands

**Funktion:** Setzen des Systemzustands

**HTTP-Methode:** PUT

**Pfad:** /control/state

**POST-Paramter:**

```

1  {
2      state : string    /* {RUN | PAUSE | SHUTDOWN_PROCESS_QUEUE | SHUTDOWN_IMMEDIATELY | KILL}
3 }
```

Listing 41: POST-Parameter zum Setzen des Systemzustands

**Antwort:**

```

1  {
2      success : boolean,
3      message : string
4 }
```

Listing 42: Antwort auf das Setzen des Systemzustands

#### 11.8.4 Konfiguration des Kompressions-Systems

Für Details zu den einzelnen Konfigurationsoptionen wird auf Abschnitt [11.6](#) verwiesen.

**Funktion:** Abrufen der Systemkonfiguration

**HTTP-Methode:** GET

**Pfad:** /settings/config

**Antwort:**

```

1  {
2      success : boolean,
3      message : string,
4      config : {
5          apiPort : integer,
6          apiAccessWhitelist : [string],
7          autostart : boolean,
8          queueMaxLength : integer,
9          defaultLevels : [string],
10         textureLevelLimits : [integer],
11         textureLevelSizes : [integer]
12         imageCompressionLevels : [
13             maxWidth : integer,
14             maxHeight : integer,
15             title : string
16         ]
17     }
18 }
19 }
```

Listing 43: Antwort auf das Abrufen der Systemkonfiguration

**Funktion:** Setzen der Systemkonfiguration

**HTTP-Methode:** PUT

**Pfad:** /settings/config

**POST-Paramter:**

```
1 {
2     apiPort : integer,
3     apiAccessWhitelist : [string],
4     autostart : boolean,
5     queueMaxLength : integer,
6     defaultLevels : [string],
7     textureLevelLimits : [integer],
8     textureLevelSizes : [integer]
9     imageCompressionLevels : [
10         maxWidth : integer,
11         maxHeight : integer,
12         title : string
13     ]
14 }
```

Listing 44: POST-Parameter zum Setzen der Systemkonfiguration

**Antwort:**

```
1 {
2     success : boolean,
3     message : string
4 }
```

Listing 45: Antwort auf das Setzen der Systemkonfiguration

## 12 VISIT METADATEN UND DIE SEMANTISCHE DATENBANK

### 12.1 Theoretische Grundlagen für die Semantische Datenbank

Dieses Unterkapitel gibt Einblicke in Teilbereiche des Semantic Webs, um eine theoretische Grundlage für die folgenden technischen Entwicklungen zu geben. Nachdem diese erläutert wurden, wird ebenfalls auf eine spezielle Ausprägung eines Metadatenmodells eingegangen, welches die Struktur für die im ViSIT Projekt verwendeten Metadaten vorgibt: das ViSIT Model **VisMo**.

Die hier angeführten Ausführungen beschränken sich jedoch nur auf jeweilige Grundlagen der Themenkomplexe, welche an manchen Stellen um weiterführende Informationen erweitert werden, wenn dies für den weiteren Verlauf von Nöten ist. Dennoch, falls angestrebt, verweisen wir für ein tieferes Verständnis auf weitere Fachliteratur, wie z.B. [Hit+07].

**SEMANTIC WEB UND RDF DATEN** Das Semantic Web ist eine Art Erweiterung zum eigentlichen World Wide Web, wie wir es aktuell kennen. Dieses ist primär für Menschen ausgelegt, die durch Homepages browsen und dabei entsprechende Informationen durch betrachten und lesen der Homepages erlangen. Diese Informationen sind dadurch jedoch nur für Menschen vorhanden, Maschinen oder Computer können auf die Informationen nicht zugreifen, um mit den entsprechenden Daten arbeiten zu können. Genau hier setzt das Semantic Web an, welches Standardisierungen, Regeln und Prozesse vorgibt, um Homepages und Applikationen so anzupassen, dass eben genau eine (semi-) automatische Informationsverarbeitung für Maschinen möglich wird.

Eine dieser Standardisierungen ist das Resource Description Framework **RDF** [MMo4], welches der de-facto Standart im Semantic Web ist, um Metadaten zu beschreiben. Daten in RDF werden als Graph modelliert und persistiert, welcher aus Knoten und Kanten besteht. Dabei entsteht eine Wissensbasis gefüllt an Informationen. Die Knoten sind hierbei die "Akteure", also diejenigen Entitäten, Sachen, Objekte, Dinge etc., ausgehend vom jeweiligen Anwendungsfall, auf die sich die im Graphen enthaltenen Informationen beziehen (diese Dinge werden im Folgenden weiterhin als "Metadatenentität" bezeichnet). Die Kanten im Graphen beschreiben Beziehungen zwischen den gegebenen Knoten und Eigenschaften der Knoten. Weiterhin sind die Knoten und Kanten durch das Grundprinzip eines **Statements** verbunden, welches eine Kapselung einer elementaren Aussage darstellt. Das Statement ist, ähnlich dem deutschen Satzbau, immer bestehend aus drei Teilen:

**SUBJEKT** Die Metadatenentität repräsentiert als ein Knoten im Graphen, von der die Aussage - und damit das Prädikat - des Statements ausgeht.

**PRÄDIKAT** Die Semantik oder die Bedeutung der Aussage.

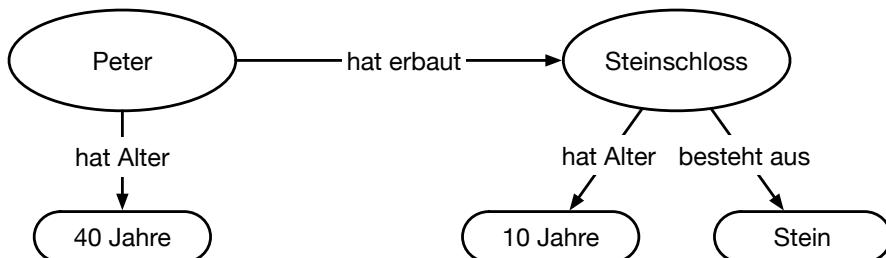
**OBJEKT** Zweierlei Konzepte können das Objekt des Statements bilden: ein weiterer Knoten im Graphen, um das Ziel der Aussage und damit des Prädikats, um eine Relation zwischen zwei Metadatenentitäten/Knoten darzustellen, oder ein fester Wert, um eine Eigenschaft einer Metadatenentitäten/eines Knotens zu charakterisieren.

Zur Verständlichkeit für die Thematik der Aussagen und Statements im Semantic Web Kontext, soll hier ein kurzes, erfundenes Beispiel erläutert werden. Folgende Aussagen bilden die Wissensbasis:

- Peter ist vom Beruf Baumeister.
- Peter ist 40 Jahre alt.
- Peter war am Bau des Steinschlosses beteiligt.
- Das Steinschloss besteht aus Stein.
- Das Steinschloss ist 10 Jahre alt.

Wie oben beschrieben, bestehen die Aussagen jeweils aus Subjekt, Prädikat und Objekt. Als Subjekte agieren die beiden Metadatenentitäten "Peter" und das "Steinschloss", während die Objekte der Aussagen der Beruf "Baumeister", das Material "Stein", zwei "Altersangaben", sowie das "Steinschloss" selbst sind. Semantisch sind die Subjekte und Objekte über die Beziehungen bzw. Eigenschaften einer "Berufszuordnung", zwei "Alterszuordnungen", einer "Materialzuweisung" sowie der "Erbauung" eines Objekts verbunden.

Diese Aussagen können nun in einen Graphen zusammengefasst werden, dessen high-level Illustration in [Abbildung 65](#) zu sehen ist.



**Abbildung 65:** Informationen aus obigen Aussagen, kombiniert als Graph.

**LINKED OPEN DATA GEDANKE** Ein weiterer Eckpfeiler des Semantic Web ist ein weiteres Konzept, das unter dem Namen **Linked Open Data - LOD** bekannt ist. Oft wird dieser Name ebenfalls für das Semantic Web selbst benutzt, die punktgenauen Definitionen überschneiden und ergänzen sich.

Einfach übersetzt zielt LOD auf öffentlich zugängliche Daten ab, die untereinander vernetzt und verlinkt sind. Somit soll es möglich sein, verteilte Datenbanken mit ihren eigenen entsprechenden Wissensbasen, miteinander zu verbinden, um so jedem Beteiligten mehr Informationen zur Verfügung zu stellen, da durch die Verlinkung einzelner Graphen ein großer Gesamtgraph entsteht. Auf diese Weise macht es Sinn, dass jede Wissensbasis ihren eigenen spezialisierten Kontext besitzt. Sollte eine Wissensbasis weitere Informationen aus einem anderen Kontext benötigen, müssen diese Daten nicht auf eigene Hand erforscht und aufbereitet werden, da eine LOD Verbindung zu einer anderen Wissensbasis hergestellt werden kann. Zur weiteren Veranschaulichung dieser Thematik und dessen Vorteile, zeigt der folgende Paragraph zwei Anwendungsfälle im geschichtswissenschaftlichen Kontext.

ZWEI ANWENDUNGSFÄLLE FÜR RDF IM GESCHICHTSWISSENSCHAFTLICHEN KONTEXT Ein erster Anwendungsfall, von dem geisteswissenschaftliche Wissensbasen profitieren können, ist oben bereits kurz angedeutet worden: das Verbinden einer eigenen Wissensbasis mit externen, bereits bestehenden Wissensbasen. Das Erforschen und Erkunden von Wissen benötigt generell in jeglichem Kontext sehr viel Zeit und ebenfalls Pflege der Daten. Daher kommt diesem Anwendungsfall der LOD Gedanke entgegen, da bereits erstellte Wissensbasen und deren Datenbanken öffentlich zugänglich sind.

Gerade generelle Themen oder Kontexte wie Personen, Städte oder Orte werden in vielen geschichtswissenschaftlichen Projekten benötigt, und gerade diese sind in öffentlichen Datenbanken zugänglich. Daher ist es für diese Anwendungsfälle sinnvoll, den eigens entwickelten Anwendungsfall an diese Datenbanken zu knüpfen. Dadurch wird der eigene Zeitaufwand erheblich reduziert und die angebundenen Daten genießen in der Regel außerdem einen hohen Standard, da bereits viele potenzielle Reviews von anderen Nutzern bestehen.

Ein zweiter großer Vorteil davon, geschichtswissenschaftliche Daten in Form von Metadaten und RDF zu persistieren, ist das mögliche Erschließen von vorher nicht bekannten oder erforschten Zusammenhängen der persistierten Objekte. Dazu folgendes (frei erfundenes) Beispiel: Ausgehend von der eigenen Wissensbasis, die die Daten aus [Abbildung 65](#) enthält, sollen nun zwei weitere Wissensbasen angekoppelt werden, welche auf der einen Seite weitere Informationen über Personen und vor allem deren familiärer Beziehungen beinhaltet, und auf der anderen Seite eine Wissensbasis, die mehr Informationen über Gebäude und deren Geschichte beinhaltet. Dies ist in [Abbildung 66](#) visualisiert.

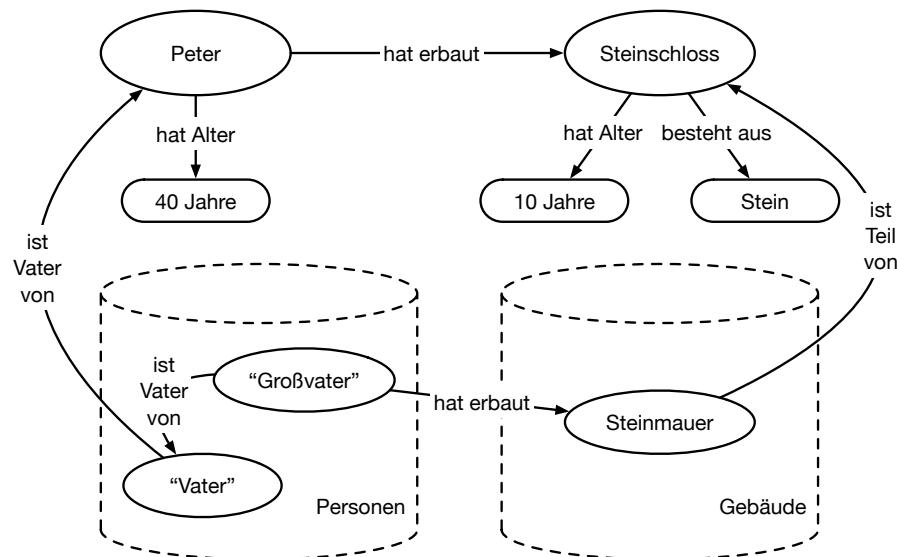


Abbildung 66: Grundlegende eigene Wissensbasis (oben), erweitert um zwei externe Wissensbasen (unten).

In dem Beispiel beinhaltet die eigene Wissensbasis Informationen über "Peter" und das "Steinschloss". Durch die beiden hinzugenommenen Wissensbasen wird Peter aus dem Anwendungsbeispiel mit seinem "Vater", und dieser wiederum mit seinem "Großvater" verbunden (die Namen sind hier zur Einfachheit ersetzt). Zudem wird die "Steinmauer" als ein Teil des Steinschlosses deklariert. Die beiden neuen Wissensbasen enthalten darüber

hinaus bereits implizit eine eigene Verbindung, die semantisch besagt, dass der "Großvater" am Bau der Steinmauer beteiligt ist.

Dadurch erweitern die beiden externen Wissensbasen die eigenen Informationen durch die neu erstellten Relationen. Darüber hinaus jedoch lässt sich so ebenfalls eine neue Erkenntnis in den Daten schliessen: sowohl "Peter" als auch dessen "Großvater" sind direkt oder indirekt am Bau des "Steinschlosses" beteiligt.

**DAS CONTEXTUAL REFERENCE MODEL – CIDOC CRM** Bisher war die technische Beschreibung der semantischen Daten im ViSIT Kontext aus Gründen der Einfachheit sehr flach gehalten. Gemäß den Semantic Web Standards basieren die Metadaten jedoch auf einem Datenmodell, um die Anforderungen des Semantic Webs zu genügen und ebenfalls technische Verarbeitbarkeit zu gewährleisten.

In ViSIT ist die Wahl hierbei auf das **Contextual Reference Model CIDOC CRM** [Doeo3] gefallen, da dies eine der bekanntesten und vorherrschendsten Ontologien im Bereich des kulturellen Erbes ist. Diese Ontologie wird als Basis benutzt, die im folgenden Paragraphen erweitert für den ViSIT Kontext beschrieben wird. Der größte Vorteil dieser Ontologie ist, dass sie sich nicht auf einen speziellen Bereich des kulturellen Erbes fokussiert ist, sondern auf generische Weise komplexe Zusammenhänge und verschiedene Themengebiete abbildet. Zudem solle es möglich sein, andere Ontologien oder Modelle aus dem selben Bereich in diese Ontologie zu überführen, um eine gemeinsam verständliche Wissensbasis zu kreieren.

Das CIDOC CRM wird seit mittlerweile über 10 Jahren von der CIDOC Documentation Standards Working Group<sup>12</sup> und der CIDOC CRM SIG<sup>13</sup> entwickelt, welche beide Arbeitsgruppen von CIDOC<sup>14</sup> sind. Das CIDOC CRM ist 2000 als "Working Draft" bei der ISO/TC46/SC4<sup>15</sup> akzeptiert worden, welcher 2006 schliesslich auch als offizieller Standard [Cida] akzeptiert wurde, und 2014 in eine überarbeitete Version [Cidb] überführt wurde.

In der aktuellen Hauptversion 6.2<sup>16</sup>, die im Mai 2015 veröffentlicht wurde, enthält die Ontologie 89 RDF Klassen und 149 einzigartige Relationen und Eigenschaften, die sich in einer mehrfach ineinander- sowie auseinander verzweigenden Struktur einordnen. Laufend werden ebenfalls Nebenversionen veröffentlicht - die aktuellste Versionsnummer lautet 6.2.3<sup>17</sup>.

**DAS VISIT MODEL – VISMO** Aufbauend auf dem CIDOC CRM wurde eine Ontologie entwickelt, die den kompletten Anwendungsfall des ViSIT Projekt abbilden kann: das **ViSIT Model VisMo**. Der Fokus liegt dabei auf der Darstellung von Architektur-Objekten und Ausstellungsobjekten, die mit Personen oder Gruppen von Personen, Orten sowie zeitlichen Events in Verbindung gesetzt werden, um eine Wissensbasis zu kreieren.

Diesbezüglich sind die Hauptentitäten, die in der ViSIT Datenbank angelegt werden können, die folgenden:

**EREIGNIS (ACTIVITY):** Diese Entität umfasst alle vergangenen und zukünftigen Vorgänge und Geschehnisse in kulturellen, sozialen und physischen Systemen, analog zum "E5\_Event"<sup>18</sup> des CIDOC CRM.

<sup>12</sup> <http://network.icom.museum/cidoc/working-groups/overview/>

<sup>13</sup> <http://network.icom.museum/cidoc/working-groups/crm-special-interest-group/>

<sup>14</sup> <http://network.icom.museum/cidoc/>

<sup>15</sup> <https://www.iso.org/committee/48798.html>

<sup>16</sup> <http://www.cidoc-crm.org/Version/version-6.2>

<sup>17</sup> <http://www.cidoc-crm.org/Version/version-6.2.3>

<sup>18</sup> <http://www.cidoc-crm.org/Entity/E5-Event/Version-6.2>

**BAUWERK (ARCHITECTURE):** Diese Entität bezeichnet alle Arten von Bau-ten, die wie die Objekte als Informationsträger (vgl. "E84\_Information\_Carrier"<sup>19</sup> des CIDOC CRM) betrachtet werden.

**GRUPPE (GROUP):** Diese Entität bezeichnet mehrere Personen, die sich zu einer Gruppierung zusammengeschlossen haben und durch eine glei-che oder ähnliche Tätigkeit miteinander verbunden sind (vgl. "E74\_Group"<sup>20</sup> des CIDOC CRM).

**INSTITUTION (INSTITUTION):** Diese Entität bezeichnet hier alle Arten von organisierten Einrichtungen, die keine natürlichen Personen oder Per-sonengruppen sind, z.B. Museen, Archive, Bibliotheken, Universitäten usw.

**OBJEKT (OBJECT):** Diese Entität bezeichnet alle Arten von Ausstellungsob-jekten aus den Sammlungen von musealen oder museumsähnlichen Institutionen, Archiven etc. , die wie Bauwerke als Informationsträ-ger betrachtet werden (vgl. "E84\_Information\_Carrier"<sup>21</sup> des CIDOC CRM).

**PERSON (PERSON):** Diese Entität bezeichnet analog zur Definition im CI-DOC CRM der "E21\_Person"<sup>22</sup> alle natürlichen Personen, die leben oder bereits verstorben sind sowie Personen, von denen angenom-men wird, dass sie lebten. Dazu zählen historische Persönlichkeiten als auch Personen aus Legenden, Mythen und Sagen.

**ORT (PLACE):** Diese Entität bezeichnet hier ausschließlich über Koordinaten lokalisierbare verschwundene und bestehende Dörfer und Städte.

**LITERATUR (REFERENCE):** Diese Entität bezeichnet alle Arten von niederge-schriebenen und veröffentlichten Texten.

Dabei erfüllt VisMo genau den Zweck, den sich das CIDOC CRM als Ziel gesetzt hat: als eine semantische "Erweiterung" des CIDOC CRM ist der Inhalt, der für VisMo produziert wird, direkt zum größten Teil verständlich und Leser oder Benutzer des Modells können dies intuitiver, auf der Basis der Beschreibungen des CIDOC CRM, verstehen, lesen und benutzen. Dies ist dadurch begründet, dass alle Klassen und viele der Relationen und Eigenschaften durch Vererbung speziellere Konzepte der CIDOC CRM Klas-sen und Relationen/Eigenschaften sind. Nur einzelne Teile des VisMo sind speziell für die Ontologie hinzugefügt worden, immer wenn kein Konzept aus dem CIDOC CRM passend für eine Vererbung war. Abbildung 67 visua-lisiert den Entwicklungsprozess hinter VisMo.

Der erste Schritt bestand dabei in der Sammlung der Kernthemen, die in ViSIT behandelt werden. Aus diesen konnte dann im nächsten Schritt ein grobes Konzept entwickelt werden, welches anschließend in RDF über-tragen werden konnte. Wie oben beschrieben, wurde hierbei von CIDOC CRM Grundklassen und Relationen bzw. Eigenschaften ausgegangen, wel-che dann für den ViSIT Kontext erweitert und angepasst wurden. Als näch-stes konnten dann die erstmals groben Konzepte und Entitäten mit benö-tigten Metadaten bzw. dessen Anforderungen erweitert werden. Die Ergeb-nisse der vorherigen Schritte konnten dann letztendlich in dem Ontologie-Editor *protegé*<sup>23</sup> zusammengeführt werden, um eine RDF/OWL Ontologie

19 <http://www.cidoc-crm.org/Entity/E84-Information-Carrier/Version-6.2>

20 <http://www.cidoc-crm.org/Entity/E74-Group/Version-6.2>

21 <http://www.cidoc-crm.org/Entity/E84-Information-Carrier/Version-6.2>

22 <http://www.cidoc-crm.org/Entity/E21-Person/Version-6.2>

23 <https://protege.stanford.edu/>

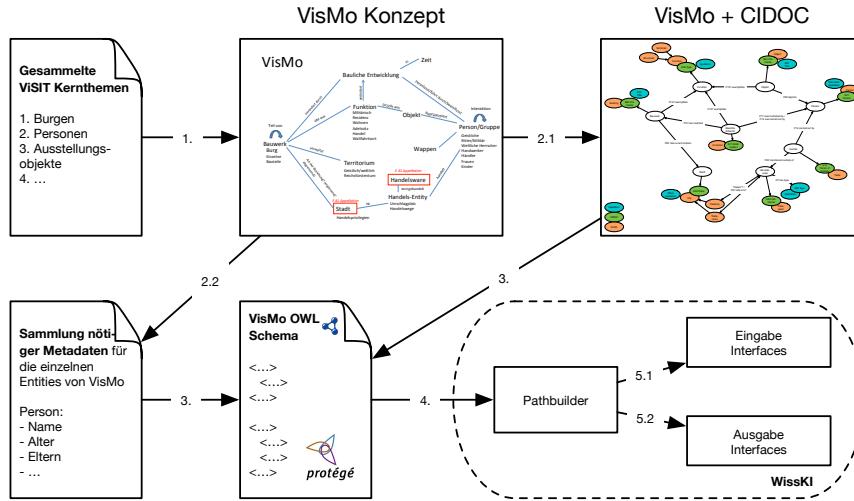


Abbildung 67: Arbeitsprozess hinter der Entwicklung des ViSIT Modells.

zu erstellen. Diese ist in ihrer letzten offiziellen Version in [Listing 49](#) im Appendix zu sehen.

Ebenfalls ist in [Abbildung 67](#) visualisiert, wie und an welcher Stelle die VisMo Ontologie technisch zum Einsatz kommt: sie dient als Input für das sogenannte WissKI Modul, um aus der Ontologie Ein- sowie Ausgabemaschen zu generieren, welche letztendlich vom Endnutzer des ViSIT Systems benutzt werden, um einerseits Daten in die semantische Datenbank einzutragen und diese dann auch wieder auszulesen und anzuzeigen. Der große Vorteil an diesem Prozess ist, dass der Endnutzer keinerlei Wissen über das Semantic Web und seine Technologien benötigt, da der oben beschriebene Prozess davon abstrahiert. Damit schreiben und lesen die Endnutzer im Endeffekt RDF, ohne davon zu wissen. Technische Details zu diesem Prozess sowie WissKI werden in folgenden Unterkapiteln gegeben.

## 12.2 Technische Details zur Semantischen Datenbank

Nachdem [Unterabschnitt 12.1](#) die theoretische Grundlage für die Semantische Datenbank beschrieben hat, fokussiert sich dieses Unterkapitel auf die technischen Aspekte der Datenbank. Dazu zählt in erster Linie die **allgemeine Infrastruktur**, das **Hosting** an der Universität Passau, der **allgemeine Zugriff auf die Datenbank**, getroffene Entscheidungen bezüglich **Security und Zertifizierungen**, sowie die anschließende Beschreibung einzelner Komponenten: dem **CMS Drupal**, dessen Modul **WissKI**, die **ViSIT REST API**, der unterliegende RDF Triplestore **RDF4J** und dessen generelle Funktionalität.

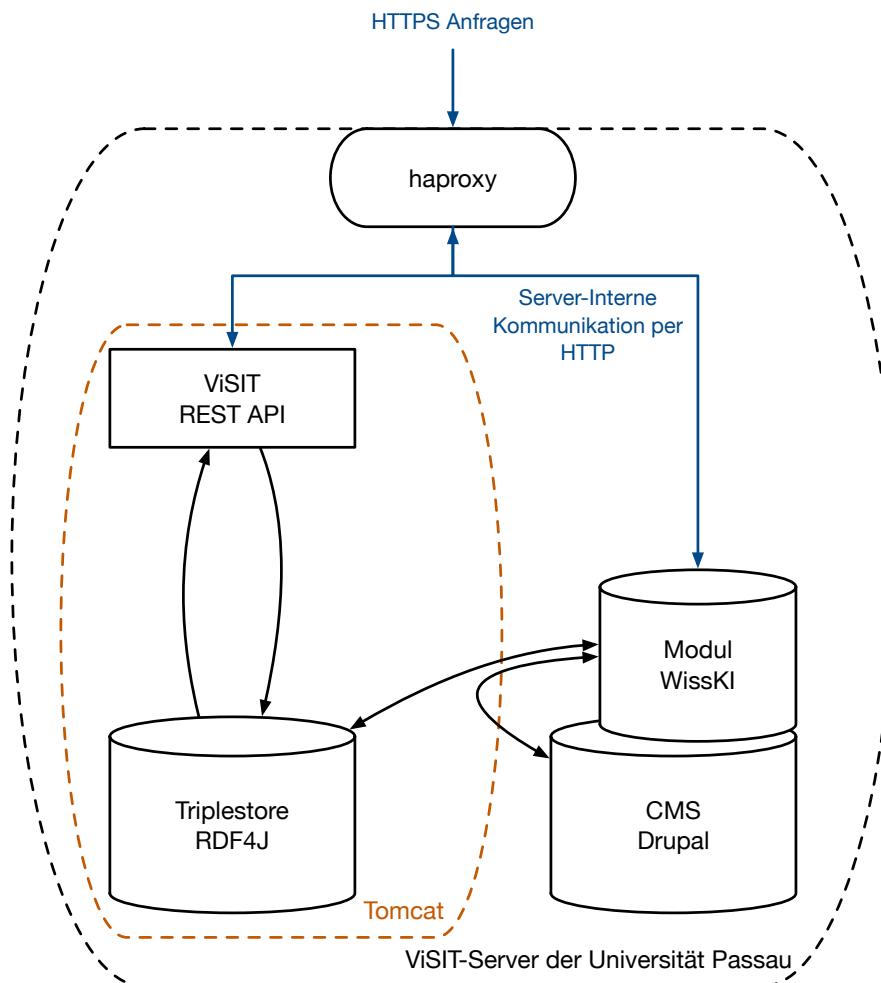
Für die semantische Datenbank wurde zur Projektlaufzeit aus Testzwecken ebenfalls eine Testinstanz ins Leben gerufen, welche eine komplette Spiegelung des damals aktuellen Systems ist. Die beiden Haupt-URLs der Server sind:

- <https://database.visit.uni-passau.de/>
- <https://database-test.visit.uni-passau.de/>

Von diesen beiden Base-URLs ausgehend sind die weiteren Komponenten über folgende URL-Zusätze zu erreichen:

- **Drupal/WissKI**: Base URL + /drupal
- **RDF4J**: Base URL + /rdf4j-workbench
- **Tomcat**: Base URL (ohne Zusatz)
- **ViSIT REST API**: Base URL + /metadb-rest-api
- **API Beschreibung**: Base URL + /metadb-test-api/swagger-ui.html

**INFRASTRUKTUR** Die Semantische Datenbank des ViSIT Projekts ist auf einem virtuellen Server an der Universität Passau installiert. Die allgemeine Infrastruktur ist in [Abbildung 68](#) zu sehen.



**Abbildung 68:** Technische Infrastruktur der Semantischen Datenbank des ViSIT Projekts.

Dessen Hauptkomponenten mit Beschreibung oder Verweis auf das ausführliche Unterkapitel sind die folgenden:

**HAPROXY** Dem virtuellen Server für die ViSIT Infrastruktur ist ein **haproxy**<sup>24</sup> vorgeschaltet. Dieser ist dafür da, die per HTTPS verschlüsselten An-

<sup>24</sup> <http://www.haproxy.org/>

fragen von aussen an den Server entgegen zu nehmen, und intern an die richtigen Komponenten weiterzuleiten. Prinzipiell kann dieser haproxy ebenfalls Anfragen per HTTP entgegen nehmen, leitet diese dann aber automatisch auf den Port für HTTPS weiter. Damit ist sicher gestellt, dass nach aussen nur verschlüsselte Daten versandt werden. Dem haproxy sind für die benötigte Funktionalität zwei backends bekannt: eines für den Tomcat (ViSIT REST API und Triplestore RDF4J) und eines für Drupal bzw. WissKI (welche hintergründig auf einem Apache laufen). Die Verschlüsselung ist durch ein SSL Zertifikat der Universität gewährleistet. Die Konfiguration zum haproxy ist am Server zu finden unter /etc/haproxy.

**TOMCAT** Zur Installation weiterer Komponenten am Server, ist ein **Apache Tomcat**<sup>25</sup> in der Version 8.5.24 installiert. Am Server ist dieser zu finden unter /opt/tomcat8/apache-tomcat-8.5.24.

**VISIT REST API** Diese API wurde eigens für ViSIT entwickelt, um eine Abstraktionsschicht für die unterliegenden Metadaten zu bieten. Während WissKI diese Abstraktion für die wissenschaftlichen Benutzer der Metadaten bildet, ist die API für die technische Anbindung der restlichen Komponenten des ViSIT Projekts zuständig. Die API bietet in erster Linie die Möglichkeit, die RDF Metadaten aus dem Triplestore zu lesen. Zurückgegeben wird das Ergebnis im JSON Format, um eine möglichst breite Verständnis und damit direkte Verwendbarkeit zu gewährleisten. Der zweite große Teil behandelt das Schreiben, Auslesen und Updaten der sogenannten technischen Metadaten: Metadaten, die Informationen zu einem Medien-Objekt im ViSIT Kontext geben. Die REST API ist als eigenständiges Java-Projekt implementiert, welches auf dem ViSIT Server bzw. in dessen Tomcat Installation deployed wird. Eine ausführliche Beschreibung wird in [Unterabschnitt 12.4](#) gegeben.

**TRIPLESTORE RDF4J** Der Triplestore ist für die Persistierung der RDF Daten zuständig. Im ViSIT Kontext ist die Wahl hierfür auf die **RDF4J**<sup>26</sup> Datenbank gefallen, dieser ist jedoch durch jeglichen gleichwertigen Triplestore ersetzbar. Wie bei der REST API beschrieben, wird im ViSIT Kontext weitestgehend möglich von den RDF Daten abstrahiert. Dies wird sowohl durch die REST API und dem WissKI Modul bewerkstelligt. Der RDF4J Triplestore ist am Server bzw. in dessen Tomcat Installation deployed.

**DRUPAL UND WISSKI** Die letzte Komponente der Infrastruktur der Semantischen Datenbank ist eine Kombination aus dem Content Management System **Drupal**<sup>27</sup> und dessen Modul **WissKI - Wissenschaftliche KommunikationsInfrastruktur**<sup>28</sup>. Als Modul baut WissKI auf der Implementierung von Drupal auf und benutzt dessen Funktionalität zum Persistieren von Entities als Inhalt. Zudem, da WissKI aber ebenfalls mit RDF Daten arbeitet, wird ein Triplestore benötigt - wie oben beschrieben. WissKI übernimmt dabei die Synchronisation zwischen den Entities in Drupal und den RDF Daten, sowohl beim Speichern als auch beim Auslesen von Daten. Weiterhin bietet WissKI die Möglichkeit, ein eigenes Datenmodell zu definieren, welches den gesamten

---

<sup>25</sup> <http://tomcat.apache.org/>

<sup>26</sup> <http://rdf4j.org/>

<sup>27</sup> <https://www.drupal.org/>

<sup>28</sup> <http://wiss-ki.eu/>

Datenfluss eine Struktur vorgibt. Aus diesem werden ebenfalls einfache Interfaces generiert, um auf der einen Seite die Daten anzulegen, und auf der anderen Seite auf eine einfache Weise darzustellen. Weitere Details hierzu werden in [Unterabschnitt 12.3](#) beschrieben.

Im Zusammenspiel der obig genannten Komponenten erlaubt das gesamte System der Semantischen Datenbank das Management der semantischen Daten, die für den Kontext des ViSIT Projekts benötigt werden. Der Workflow der Datenbank sieht dabei in etwa wie folgt aus:

- Über die einfachen WissKI Eingabe-Interfaces geben die Kuratoren bzw. Geisteswissenschaftler Informationen und Metadaten in das Gesamtsystem ein.
- Diese Metadaten werden mit der WissKI Funktionalität automatisch ebenfalls in RDF Daten übersetzt, die dem Schema entsprechen, welches bei Installation und Konfiguration des Gesamtsystems erstellt wird (siehe [Unterabschnitt 12.1](#) für das Metadatenmodell CIDOC + VisMo, [Unterabschnitt 12.3](#) für die Konfiguration von WissKI ).
- Für Forschungszwecke können diese angelegten Metadaten dann mit den WissKI Ausgabe-Interfaces betrachtet werden, was ebenfalls die Graphstrukturen hinter den Daten hervorhebt, da in den Interfaces zwischen den einzelnen Entitäten navigiert werden kann.
- Die ViSIT REST API dient zur technischen Anbindung weiterer ViSIT Komponenten, indem die Metadaten auf standardisierte Weise abgefragt werden können.

### 12.3 WissKI – Wissenschaftliche Kommunikationsinfrastruktur

Das WissKI Modul bietet die Möglichkeiten, sowohl RDF Daten zu lesen und zu schreiben - aber auf eine einfache Weise über simpel gehaltene Eingabe- und Ausgabeinterfaces, um den Zugang für Forscher und die Geisteswissenschaftler im ViSIT Kontext zu gewährleisten. Um dies jedoch zu bewerkstelligen, benötigt das Modul verschiedene Konfigurationen und Einstellungen. Unter anderem das wichtigste ist das Definieren der semantischen Struktur der Daten, wie es bereits oben beschrieben wurde.

Für den ViSIT Anwendungsfall ist das technische System der semantischen Datenbank, beschrieben in [Abschnitt 12](#), vollständig konfiguriert und betriebsbereit. Nichtsdestotrotz werden in den folgenden Unterabschnitten die Einstellungen für WissKI erläutert, um für potenziell zukünftige Änderungen eine grundlegende Beschreibung zu geben. Diese Beschreibungen können jedoch nie eine Tiefe und Genauigkeit erreichen, wie sie von den WissKI Entwicklern gegeben werden kann. Deswegen sei hier ebenfalls auf <http://wiss-ki.eu/> verwiesen.

**WISSKI SALZ ADAPTER** Wie ebenfalls bereits in [Unterabschnitt 12.2](#) beschrieben, regelt das WissKI System das Persistieren und Auslesen der im Gesamtsystem angewandten semantischen Daten. Als ein Modul für das CMS Drupal, werden die Daten auf der einen Seite im CMS als Entitäten gespeichert, auf der anderen Seite - da die Daten auf Semantic Web Standards basieren sollen - als RDF Daten in einem Triplestore. WissKI führt hier automatisch die Konvertierung zwischen den beiden Datenbanken durch, ohne dass der Nutzer hier aktiv werden müsste.

Die Verbindung mit dem Drupal CMS geschieht automatisch mit der Installation des WissKI Moduls. Was jedoch konfiguriert werden muss ist die Verbindung des Moduls zum zu verwendenden Triplestore. Dies passiert im sogenannten **WissKI Salz Adapter**.

Wenn das Menü zum bearbeiten der Adapter geöffnet wird, erscheint eine Liste der aktuell definierten Adapter. Für das ViSIT Projekt ist bereits ein Adapter eingerichtet mit dem Namen `visittestrepo`. Grundsätzlich reicht für einen Anwendungsfall wie ViSIT ein Adapter, es können aber natürlich beliebig viele Adapter definiert werden. [Abbildung 77](#) und [Abbildung 78](#) im Appendix zeigen die Konfigurationsmöglichkeiten eines WissKI Salz Adapters, bzw. die Einstellungen die für ViSIT getätigten wurden.

Die wichtigsten Endpunkte bzw. Konfigurationsmöglichkeiten sind die folgenden (die hier nicht erwähnten Punkte können in der Regel auf der Standardkonfiguration bzw. leer gelassen werden):

**ADAPTER NAME:** Der Name des Adapters, mit dem dieser eindeutig identifiziert werden kann.

**WRITEABLE UND PREFERRED LOCAL STORE:** Diese beiden Checkboxen sollten in der Regel immer gesetzt sein, wenn es sich um den Adapter bzw. Triplestore handelt, der hauptsächlich mit dem System arbeiten soll. "Writeable" bedeutet, dass Daten auf dem Triplestore geschrieben werden dürfen, "Preferred Local Store" weist das System an, diesen entsprechenden Adapter als Hauptadapter zu benutzen, falls mehrere definiert sein sollten.

**READ UND WRITE URL:** Dies sind die beiden Einstellungen, die WissKI mit dem Triplestore verbinden. Es sind die beiden URLs des entsprechenden Triplestore, auf die bei diesem lesend bzw. schreibend zugegriffen werden kann. Nur wenn diese beide gesetzt sind, kann das System richtig in Betrieb genommen werden. Die beiden URLs, die in den Bildern gesetzt sind, zeigen also auf den Triplestore, der in der Infrastruktur für die semantische Datenbank installiert wurde. (Zusätzliche Hintergrundinformation: die URLs zeigen hier auf "`http://localhost:8081/...`" und damit auf eine lokale Installation, da sowohl das WissKI /CMS System und der Triplestore auf dem selben Server installiert ist. Die beiden Komponenten kommunizieren lokal miteinander.)

**DEFAULT GRAPH URI:** Für RDF Daten werden eindeutige URI Bezeichner für die Knoten und Kanten des RDF Graphen benötigt. In der Regel erhalten die Knoten, wenn sie für Instanzen bzw. Entitäten stehen, eine zufällig generierte Zeichenkette als URI. Die Default Graph URI wird dann verwendet, um vor diese Zeichenkette gesetzt zu werden. Somit entstehen URI Bezeichner, die auf die Semantic Web Standards passen und auch den eigenen Anwendungsfall besser repräsentieren: so wie im Beispiel für das ViSIT Projekt mit "`http://visit.de/data`". Eine beispielhafte URI wäre also "`http://visit.de/data/5c62c9aab4666`".

**REITER COMPUTE TYPE AND PROPERTY HIERARCHY:** Ein weiterer wichtiger Punkt im Bezug auf das Modell und damit die Struktur der semantischen Daten befindet sich im Reiter mit dem Namen "Compute Type and Property Hierarchy and Domains and Ranges". Öffnet man den Reiter, erhält man die Möglichkeit (nachdem die Checkbox "Re-Compute results" betätigt wurde), durch den Button "Start Reasoning" einen

sogenannten Reasoning Prozess zu starten. Einfach formuliert betrachtet dieser die aktuell definierten Modelle des Systems, um potenziell zusätzliche Informationen hinzuzufügen. Dadurch kann das System auf schnellere Weise arbeiten, da diese Informationen nicht erst im produktiv laufenden Zustand des Systems hinzugefügt werden müssen. Diesen Prozess zu starten ist sehr wichtig, wenn ein **Update oder eine Änderung des Metadatenmodells passiert ist**. Der Prozess kann einige Minuten in Anspruch nehmen, bis er vollständig durchgeführt wurde.

**WISSKI ONTOLOGY** In diesem Teil der Konfiguration kann die unterliegenden Ontologie bzw. das Metadatenmodell für das System definiert werden. Dazu wird zunächst in einem Drop-Down Menu der Adapter ausgewählt, für den dies getan werden soll. Weiterhin muss dann eine RDFS oder OWL Schema Datei in das WissKI System hochgeladen werden. Dazu ist ein entsprechender Button vorgesehen.

Wenn bereits eine Ontologie für einen Adapter existiert, wird diese bzw. vielmehr dessen enthaltene Namensräume angezeigt. Zusätzlich gibt es dann die Möglichkeit, die aktuelle Ontologie zu löschen, womit wieder zum ursprünglichen Zustand - einer nicht vorhandenen Ontologie samt Upload Button - zurückgekehrt wird.

Auf diese Weise kann eine Ontologie ausgetauscht werden. Vorsicht jedoch hier: Beim Austauschen einer Ontologie sollte darauf geachtet werden, dass die alte Ontologie in der neuen Ontologie enthalten ist, damit die aktuell definierten Pfade (siehe nächsten Unterabschnitt) nicht invalidiert werden.

Ein Beispiel für das VisMo Modell, welches für das ViSIT Projekt im entsprechenden WissKI Modul definiert ist, ist in [Abbildung 79](#) im Appendix zu sehen.

**PATHBUILDERS** Die Pfade des WissKI Moduls bilden das eigentliche Herzstück, da ausgehend von diesen Pfaden alle weiteren Komponenten, wie zum Beispiel die Eingabe- sowie Ausgabeinterfaces, generiert werden. Dieser Unterabschnitt wird einen Einblick in die Konfiguration des ViSIT Projekts geben. Wie eingangs zu diesem Kapitel jedoch bereits erwähnt, kann dieser Einblick nie alle Details des Moduls abdecken. Deswegen sei hier nochmals auf <http://wiss-ki.eu/> für detailliertere und ausführlichere Erklärungen verwiesen.

Die erste Übersicht beim Navigieren auf das Pathbuilders Menü listet alle vorhandenen Pathbuilder auf, die aktuell im entsprechenden WissKI System definiert sind. Genauso wie für die Salz Adapter und die Ontologie genügt es aber auch hier, einen Pathbuilder zu benutzen. Der für das ViSIT Projekt konfigurierte Pathbuilder trägt den Namen "visittestrepo\_paths".

Wird dieser editiert, gelangt man in die Übersicht aller in diesem Pathbuilder befindlichen Pfade. Dies ist beispielhaft in [Abbildung 80](#) zu sehen.

Ein WissKI Pfad kann intern eine Gruppe (zur Gruppierung mehrerer Pfade für das selbe Objekt) oder ein wirklicher Pfad sein und besteht prinzipiell aus drei wichtigen Komponenten:

- >ID Eindeutiger Identifikator für den gesamten Pfad innerhalb des WissKI Systems.

**PFAD** Einzelner Knoten für eine Gruppe, oder eine Folge von RDF Knoten, Relationen und Eigenschaften, die den Pfad im RDF Graphen widerspiegeln sollen.

**DATENTYP** Definition des Werts des jeweiligen Pfades. Bei primitiven Datentypen endet der Pfad in einer RDF Eigenschaft, während eine sogenannte “entity reference” eine Relation, also einer Verbindung zu einem weiteren Knoten bzw. einer WissKI Entität entspricht.

Drei Beispiele sollen diesen Sachverhalt weiter erklären. [Abbildung 69](#) zeigt zwei Pfade, wie sie für das ViSIT Projekt definiert wurden: der obere “Pfad” ist eine Gruppe für das allgemeine Museumsobjekt. Dessen ID ist “Object”, während der Pfad nur auf “<http://visit.de/ontologies/vismo/Object>” gesetzt ist. Dies lässt sich dadurch erklären, dass die Gruppe für den Ursprungsknoten eines dieser Entitäten steht, welcher den RDF Typen “<http://visit.de/ontologies/vismo/Object>” besitzen soll. Weiterhin benötigt eine Gruppe keinen Datentypen.



**Abbildung 69:** Zwei Beispiel-Pfade aus der WissKI Konfiguration des ViSIT Projekts.

Der zweite Pfad in [Abbildung 69](#) ist nun ein wirklicher Pfad und steht high-level für den bezeichnenden Titel eines Ausstellungsobjekts. Die ID des Pfads ist “Object\_identifiedBy\_Title”, während der Pfad aus zwei Knoten und einer Relation besteht:

- Da sich der Pfad bzw. der zugehörige Titel auf ein Ausstellungsobjekt beziehen soll, ist der erste Knoten von dem der Pfad ausgeht “<http://visit.de/ontologies/vismo/Object>”.
- An diesen Knoten schließt sich dann die Relation “[ecrm:P1\\_is\\_identified\\_by](#)” an.
- Der Zielknoten dieses Pfads ist ein weiterer Knoten: “[ecrm:E35\\_Title](#)”.

Was für den zweiten Pfad noch fehlt ist der zugehörige Datentyp, welcher in der Ansicht in [Abbildung 69](#) ebenfalls zu sehen ist: da ein Titel angegeben werden soll, definiert der Pfad eine Eigenschaft am Ende des Pfades (nicht in der Übersicht zu sehen) und benötigt damit einen primitiven Datentypen “text/plain”. Editiert man diesen Pfad, öffnet sich die Maske, die in [Abbildung 70](#) zu sehen ist. Dort können viele Einstellungen zum Pfad editiert werden und zusätzlich ist die Angabe der letztendlichen Eigenschaft durch die Eingabe “Datatype Property” möglich. In diesem Beispiel ist der letzte Teil des Pfads somit “[http://erlangen-crm.org/170309/P3\\_has\\_note](http://erlangen-crm.org/170309/P3_has_note)”.

Wird die aktuelle Einstellung des Pfads gespeichert, öffnet sich die zweite Maske zur Konfiguration eines WissKI Pfads. Ausgehend von dem aktuellen Beispiel eines Objekt-Titels ist dies in [Abbildung 71](#) zu sehen. Die ersten vier Einstellungen werden automatisch vom System gesetzt, und sollten in der Regel nicht angepasst werden müssen. Wichtig sind die vier unteren Einstellungen, welche den Datentypen des aktuell betrachteten Pfads definieren, indem zuerst der allgemeine Datentyp gesetzt wird und in den folgenden drei Einstellungen lässt sich einstellen, wie dieser Datentyp später in der Eingabemaske des WissKI Systems aussehen soll. In unserem Beispiel

The screenshot shows a web-based form for editing a semantic path. At the top, there's a 'Name' field containing 'Object\_identifiedBy\_Title' with a note 'Machine name: object\_identifiedby\_title'. Below it is a 'Path Type' dropdown set to 'Path'. A question 'Is this Path a group?' has a 'no' option selected. A status message '► REASONER HAS RUN. CACHE IS PREPARED' is displayed. The main area is a table with two columns: 'STEP' and 'EDIT'. There are four rows in the table:

STEP	EDIT
<a href="http://visit.de/ontologies/vismo/Object">http://visit.de/ontologies/vismo/Object</a>	-
<a href="http://erlangen-crm.org/170309/P1_is_identified_by">http://erlangen-crm.org/170309/P1_is_identified_by</a>	-
<a href="http://erlangen-crm.org/170309/E35_Title">http://erlangen-crm.org/170309/E35_Title</a>	-
please select	-

Below the table are sections for 'Datatype Property' (set to [http://erlangen-crm.org/170309/P3\\_has\\_note](http://erlangen-crm.org/170309/P3_has_note)) and 'Disambiguation Point' (set to 'Concept 2: [http://erlangen-crm.org/170309/E35\\_Title](http://erlangen-crm.org/170309/E35_Title)'). A blue 'Save' button is at the bottom.

Abbildung 70: Erste Maske zum Editieren eines WissKI Pfads.

kann ein Text in einem einfachen Textfeld angelegt werden. Zusätzlich dazu kann die Kardinalität für das entsprechende Feld festgelegt werden.

Abbildung 72 zeigt weiterhin ein Beispiel für einen Pfad, welcher eine Relation zwischen zwei Datenbankobjekten definiert. In diesem Falle handelt es sich semantisch um eine Beziehung zwischen Teilobjekten, also dass ein Objekt ein Teil eines zweiten Objekts ist. Um dies zu tun ist der Datentyp "Entity reference" nötig, und dass der Pfad in der entsprechend zu referenzierenden Klasse endet: wie in diesem Fall "<http://visit.de/ontologies/vismo/Object>".

Zwei weitere wichtige Punkte der WissKI Konfiguration sind in obigen Beispielen zu sehen:

**ANORDNUNG DER PFADE** Die Anordnung der Pfade spielt eine wichtige Rolle in der Konfiguration eines WissKI Systems. Dies ist auf der linken Seite in Abbildung 80 und der Übersicht der Pfade eines Pathbuilders zu sehen. Die Pfade können dort (durch drag&drop der "Kreuzchen" links neben einer Pfad-ID) in verschiedene Reihenfolgen bzw. Gruppierungen gebracht werden, um somit die Zugehörigkeit eines Pfads zu einer Gruppe, bzw. sogar einer Gruppe zu einer Obergruppe zu definieren. Dies wird getan, indem ein Pfad "unter" und dann "eine Ebene nach rechts" geschoben wird, wie dies im Beispiel für die ersten beiden Pfade zu sehen ist (ebenfalls in Abbildung 69 zu sehen). Eine Untergruppe zum Objekt bildet zum Beispiel der Pfad mit der ID "Object\_Dating", welche ihrerseits wieder vier Pfade unter sich zusammen führt. Dies ist wichtig, da das Setzen eines dieser Pfade nicht jeweils einen eigenen Zwischenknoten (mit dem RDF Typen "<http://visit.de/ontologies/vismo/Dating>") erzeugen soll, sondern alle vier Pfade den selben Zwischenknoten benutzen sollen.

**DISAMBIGUIERUNG** Die Disambiguierung bezieht sich - seicht formuliert - ähnlich wie die oben beschriebene Anordnung und Untergruppen indirekt auf die korrekte Zuweisung von Knoten zu ihren entsprechenden Objekten. Die Disambiguierung ist in den Pfdaden durch die rot geschriebenen Teile zu sehen. Sie weist das hinterliegende System an, dass alles was ab diesem Punkt im Pfad definiert ist, einzigartig im

Pathbuilder \*

visittestrepo\_paths

Name of the pathbuilder.

Path \*

object\_identifiedby\_title

Name of the path.

**CHOOSE FIELD**

Titel (f961978b85ac335b73e52837f090be9d)

Select an existing field from bundle Object (b15d6693d3ba884f733c2cce4cd6b015)

f961978b85ac335b73e52837f090be9d

ID of the mapped Field.

Type of the field that should be generated. \*

Text (plain)

Type for the Field (Textfield, Image, ...)

Type of form display for field

Textfield

Widget for the Field – If there is any.

Type of formatter for field

Plain text

Formatter for the field – If there is any.

Cardinality

Unlimited

Save Delete

Abbildung 71: Zweite Maske zum Editieren eines WissKI Pfads.



Abbildung 72: Beispiel-Pfad aus der WissKI Konfiguration des ViSIT Projekts für eine Relation zwischen zwei Entitäten der Datenbank.

System gespeichert werden soll. Dadurch können keine Duplikate mit genau der selben Konfiguration entstehen. Dies lässt sich einfach am obigen Beispiel erläutern: die Disambiguierung für den Objekttitle beginnt ab dem „ecrm:E35\_Title“ Knoten, der weiterhin nur die RDF Eigenschaft „[http://erlangen-crm.org/170309/P3\\_has\\_note](http://erlangen-crm.org/170309/P3_has_note)“ besitzt. Durch die Disambiguierung an dieser Stelle wird das System keinen zweiten Knoten erstellen, der den selben Titel in der Notiz-Eigenschaft besitzt. Damit wird sichergestellt, dass zum Beispiel keine zwei Objekte mit dem Titel „Mona Lisa“ erstellt werden können. Technisch verweist das System im Hintergrund bei Verweis auf diesen Knoten somit immer auf genau diesen einen Knoten – wie beschrieben werden *keine* Duplikate davon erstellt. Die Disambiguierung eines Pfads kann in der Maske mit dem Feld „Disambiguation Point“ vorgenommen werden, die in [Abbildung 70](#) zu sehen ist.

Die Informationen und die Konfiguration des Pathbuilders wird vom WissKI System weiterhin genutzt, um Ein- sowie Ausgabeinterfaces für die Hauptobjekte des Pathbuilders zu erzeugen. Über die Create-Seite des WissKI Systems (zu sehen in [Abbildung 81](#) im Appendix) ist unter anderem das Eingabe-Interface für die Ausstellungsobjekte (Object) zu erreichen, dieses ist in [Abbildung 82](#) im Appendix zu sehen. Nachdem Objekte über dieses erstellt wurden, können diese per Navigate- und Find-Seite des WissKI Systems eingesehen werden. Ein Beispiel für das Ausgabeinterface einer Partisanen aus dem ViSIT Projekt ist in [Abbildung 83](#) im Appendix zu sehen.

## 12.4 Technischer Zugang zu den Metadaten – die ViSIT REST API

Das oben beschriebene WissKI System ist entworfen, um im ViSIT Kontext Zugang für Geisteswissenschaftler, Museumsmitarbeiter und allgemein forschenden Personen zu schaffen. Weiterhin war es im ViSIT Projekt aber ebenfalls nötig, einen technischen Zugang zu den über WissKI erzeugten Daten zu gewährleisten. Diesen technischen Zugang verwenden weiter verarbeitende Komponenten des Projekts, beschrieben in den Kapiteln 4 bis 8. Ziel ist es ebenfalls, die Informationen der RDF Daten zu vermitteln, ohne dass der Empfänger mit RDF oder anderen semantischen Technologien in Berührung kommen muss.

Um dies auf standardisierte Weise durchzuführen, ist die Wahl auf eine serverseitige REST API gefallen, die mit den weiteren technischen Komponenten via HTTP kommunizieren kann. Die REST API ist in Java geschrieben und mit dem Spring Framework<sup>29</sup> umgesetzt. Die Entwicklung ist in einem Repository im allgemeinen ViSIT Projekt Bitbucket: <https://bitbucket.org/visit2016/metadb-rest-api/>. Eine direkte Kommunikation der aktuell installierten Version der REST API ist stets (sowohl auf dem produktiven Server als auch dem Testsystem) unter der URL Endung “.../-metadb-rest-api/swagger-ui.html” zu finden.

Dieses Kapitel beschreibt weiterhin die allgemeine Umsetzung der API, dessen Hintergründe, sowie technische Eigenheiten wie Datenmodelle usw. Jedoch sind *die wichtigsten technischen Details zur Verwendung der ViSIT REST API* in [Unterabschnitt 12.5](#) beschrieben. Dort werden unter anderem wichtige Skripte zur Inbetriebnahme der API erklärt, sowie der Prozess des Deployments erläutert.

Thematisch lässt sich die REST API in folgende Teile aufteilen, die unterschiedliche Aufgabenbereiche übernehmen:

**DIGITAL REPRESENTATIONS:** Die Digital Representations - zu deutsch digitale Repräsentationen - stellen die Verbindungen zwischen Metadaten und zugehörigen Mediendaten dar. Hierzu sind sie ein Eintrag in der Metadatenbank, die zu entsprechenden Entitäten hinzugefügt werden, und dabei verschiedene technische Details zu den Mediendaten beinhaltet. Die REST API bietet hierfür die Möglichkeiten, über HTTP entsprechende Digital Representations zu erzeugen, schreiben, bearbeiten, und zu löschen. Den Repräsentationen liegt folgendes (RDF) Modell, aufgezeigt in [Abbildung 73](#), zugrunde:

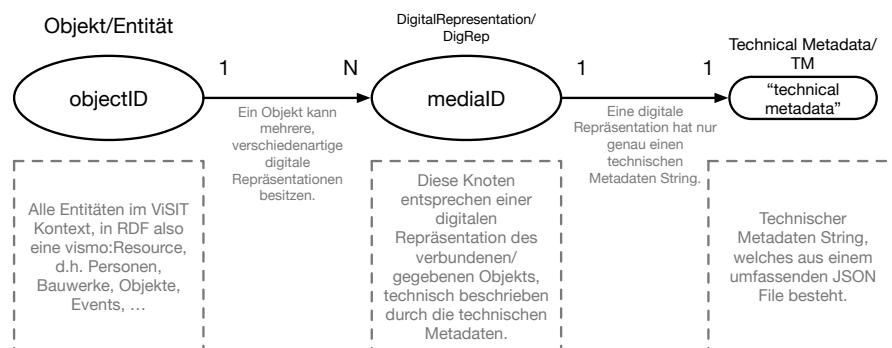


Abbildung 73: Modell der digitalen Repräsentationen.

<sup>29</sup> <https://spring.io/>

Die möglichen API Funktionen sind, in [Tabelle 5](#) die folgenden (die API Pfade sind relativ zur jeweiligen REST API angegeben):

HTTP	API Pfad	Request Params	Kurzbeschreibung
GET	/digrep/media	id	Gibt TM String des mit der "id" spezifizierten DigRep Knotens zurück.
PUT	/digrep/media	id, newData	Updated den TM String durch "newData" des per "id" spezifizierten DigRep Knotens.
DELETE	/digrep/media	id	Löscht den durch die "id" spezifizierten DigRep Knoten.
GET	/digrep/object	id	Gibt alle TM Strings und deren DigRep Knoten IDs zurück.
POST	/digrep/object	id	Legt einen neuen DigRep Knoten für das Objekt mit der gegebenen "id" an.
DELETE	/digrep/object	mediaId, objectId	Löscht den DigRep Knoten mit der ID "mediaID", zugehörig zum Objekt mit der ID "objectId".

**Tabelle 5:** API Funktionen für die Digital Representations.

**OBJEKTE:** Dieser Bereich der API dient hauptsächlich zum Auslesen der Datenbankobjekte bzw. Entitäten, die über WissKI in der Metadatenbank angelegt wurden. Diese API Schnittstelle erfüllt die Anforderung, die bereits am Anfang dieses Unterkapitels beschrieben wurde: die Gewährleistung des technischen Auslesens der Metadaten in einem *nicht-Semantic Web Format*, in diesem Falle JSON. Dieses JSON beinhaltet alle Informationen, die entsprechend als RDF Metadaten auf der Datenbank vorhanden sind. [Listing 50](#) im Appendix gibt eine Art Schema für diesen Output an. Dort werden alle möglichen Felder mit ID, Datentyp und den potenziell referenzierten Typen einer Referenz angegeben. Zum Auslesen der Objekte bietet die API zwei mögliche API Funktionen aus [Tabelle 6](#):

HTTP	API Pfad	Request Params	Kurzbeschreibung
GET	/object	id	Gibt die JSON Repräsentation des Objekts mit der ID "id" zurück.
GET	/wisskiobject	wisskipath	Gibt die JSON Repräsentation des Objekts zurück, welches dem WissKI View Path "wisskipath" entspricht.

**Tabelle 6:** API Funktionen zum Auslesen von Objekten.

Wichtig zu wissen beim Auslesen der Objekte ist, dass die angesprochene ID der API die RDF ID des Knotens der entsprechenden Entität ist (z.B. "<http://visit.de/data/5c4ae02a2d569>"), während der sogenannte "WissKI View Path" derjenige URL Pfad ist, der ange-

zeigt wird, wenn eine Entität per WissKI Oberfläche betrachtet wird (z.B. "<https://database.visit.uni-passau.de/drupal/wisski/navigate/405/view>").

Weiterhin bietet dieser Bereich der API noch ein zusätzliches Quality of Life Feature: das Zurückgeben der Anzahl an vorhandenen Digital Representations und damit vorhandenen Mediendaten, bezogen auf ein Objekt. Dieses Objekt kann der API, wie beim Auslesen, per RDF ID oder per WissKI View Pfad übergeben werden.

#### (UPLOAD UND DOWNLOAD FÜR EXCEL:) TODO add?

Für die REST API ist eine Verschlüsselung umgesetzt, die dem Konzept der "basic authentication"<sup>30</sup> folgt. Mit dieser werden die sogenannten "unsicheren" Operationen der API, also diejenigen, die Daten verändern, erzeugen, und löschen dürfen, von unzulässigem Zugriff geschützt. Diese sind im ViSIT Kontext diejenigen Operationen, die sich auf die DigitalRepresentations beziehen. Alle rein lesenden Zugriffe auf die Metadatenbank sind gemäß dem Linked Open Data Gedanken offen nach aussen.

Für die gesicherten Zugänge der API sind auf dem ViSIT Server verschiedene "User" samt Passwort angelegt, welche auf die jeweils verarbeitenden Applikationen gemünzt werden, dementsprechend die Tablet und Fernrohr Apps, sowie die Kompressions-Komponente. Diese Liste an Usern und Passwort sind am entsprechenden Server im eigenen Bereich des "visit" Accounts zu finden und nur für entsprechende Administratoren zugänglich. Die Datei hat den Namen "users.csv" und muss **vor dem Start bzw. dem Deployment** der REST API vorhanden sein, damit diese darauf Zugriff hat.

#### 12.5 Wichtige Technische Charakteristika der Entwicklung und den Betrieb der Semantischen Datenbank

In diesem Unterabschnitt werden wichtige technische Details zum Betrieb der ViSIT REST API erläutert. Diese sind wichtig für die Nutzung der API.

**PYTHON SCRIPT ZUM GENERIEREN DER SPARQL QUERY TEMPLATES WICHTIG:** Das im Folgenden beschriebene Script muss **vor Deployment** der REST API ausgeführt werden.

Für den Betrieb der REST API, vor allem zum Auslesen der unterliegenden Metadaten, sind sogenannte SPARQL Query Templates von Nöten. Diese können als Vorlagen für die Abfrage aus der Metadatenbank verstanden werden. Diese Query Templates werden automatisch aufbauend auf dem in WissKI definierten Paths erstellt. So kann, wenn sich die WissKI Pfad Konfiguration ändert, das Script erneut angestoßen werden, um auf die Veränderung zu reagieren und damit die neu hinzugefügten Pfade mit in die Anfragen aufgenommen werden. Dieser Prozess ist in einem Phyton Script gekapselt, welches auf dem ViSIT Server ausgeführt werden kann.

Wichtigste Information die das Script benötigt sind die Pfad Konfigurationen des WissKI Systems. Diese können als Datei direkt im WissKI System, im Pathbuilder downloadet werden. [Abbildung 74](#) zeigt diese Funktionalität mit dem "Create Exportfile" Button.

Nach Betätigen des Buttons wird eine aktuelle Datei in die darüber stehende Liste aufgenommen. Durch Drücken dieser kann die Datei anschließend gespeichert werden.

<sup>30</sup> Erklärt z.B. auf <https://swagger.io/docs/specification/authentication/basic-authentication/>

Das Python Script befindet sich auf dem Server im Account "visit" im Ordner "python" und heißt "CreateSPARQLTemplatesFromPathsXML.py". Die Pfad-Datei muss umbenannt werden zu "paths.xml" und muss sich im selben Ordner befinden. Das Ausführen des Scripts kann mit folgendem Befehl ausgeführt werden:

```
1 | sudo python3 CreateSPARQLTemplatesFromPathsXML.py
```

**Listing 46:** Befehl zum Starten des Python Script zum Erstellen der Query Templates der ViSIT Metadatenbank.

Die Ergebnisse des Scripts werden automatisch in den Ordner "Templates" im Hauptordner des Accounts "visit" gespeichert, und automatisiert von der REST API benutzt.

EINSTELLUNGEN ZUM DEPLOYMENT DER REST API Wie oben bereits beschrieben, ist die REST API in einem Repository<sup>31</sup> des ViSIT Haupt-Repositories angelegt.

In der Regel ist das entsprechende Projekt in einem "offline Teststatus" bezüglich seiner Konfiguration. Dies bedeutet, dass lokale Tests ausgeführt werden können, um die Funktionen der API lokal zu überprüfen. Sollte die REST API produktiv auf einem Server deployed werden, müssen folgende Änderungen in der Konfiguration durchgeführt werden:

- Einbinden der springframework Dependency: In der "pom.xml" Maven Konfigurationsdatei des Projekts muss die Dependency für das Artefakt "spring-boot-starter-tomcat" auf den scope "provided" gesetzt werden. Die entsprechende Einstellung ist in Abbildung 75 zu sehen. Wichtig ist, dass Zeile 4 **nicht** kommentiert ist.
- Verbindung zum entsprechenden Triplestore: Weiterhin ist es nötig, die REST API mit dem Triplestore zu verbinden, mit dem das aktuell verwendete System, im speziellen das WissKI System, arbeitet. Diese Verbindung wird in der REST API in der Konfigurationsdatei "application.properties" gesetzt. Dabei muss der query und der update Endpunkt URLs des jeweiligen Triplestores angegeben werden. Im offline Modus sind die beiden URLs auf "none" gesetzt, während sie für den Produktiv-Modus der API auf den entsprechenden Triplestore eingestellt werden müssen. In Abbildung 76 ist die Konfiguration für den offline Modus zu sehen, während ebenfalls zwei vordefinierte URL Pärchen zu sehen sind, welche für den ViSIT Test- und Produktiv-Server stehen.

## 12.6 Semantische Datenbank - FAQ und häufig auftretende Probleme

Nachdem bis jetzt alle Details bezüglich der Metadatenbank und dessen Umgang erläutert wurden, führt dieses Unterkapitel eine Zusammenfassung von bekannten "Problemen" in Kombination mit Lösungen an, die im produktiven Betrieb der Datenbank entstehen können. Bei Auftreten eines dieser Probleme sollte die Datenbank durch Einsatz der aufgeführten Lösung wieder in einen funktionierenden Zustand überführt werden können.

---

<sup>31</sup> <https://bitbucket.org/visit2016/metadb-rest-api/src/master/>

**WISSKI CACHE FLUSH** Bei manchen Konfigurations-Änderungen kann es passieren, dass diese erst “online” geschaltet werden, wenn der Cache des Drupal Systems geleert wird. Dies ist zum Beispiel der Fall, wenn ein Thumbnail für ein Objekt in der Metadatenbank hochgeladen wird. Der produktive Server ist so konfiguriert, dass einmal am Tag ein solcher Cache Flush durchgeführt wird. Sollten die jeweiligen Informationen durch die Konfiguration jedoch sofort benötigt werden, kann ein Cache Flush auch per Hand am Server ausgeführt werden. Dazu muss in das Verzeichnis der Drupal Installation gewechselt werden (`/var/www/html/drupal`) und folgender Befehl ausgeführt werden:

```
1 | drush cr
```

**Listing 47:** Befehl für einen Cache Flush eines Drupal Systems.

Sollte dieser Befehl aus irgendwelchen Gründen nicht funktionieren, kann die selbe Funktionalität über andere Wege durchgeführt werden. Hierzu folgender Link: <https://www.drupal.org/docs/7/administering-drupal-7-site/clearing-or-rebuilding-drupals-cache>.

**RECOMPUTE HIERARCHY IN WISSKI** Wenn die dem WissKI System unterliegende Ontologie ausgetauscht oder upgedated wird, kann es sein, dass das System über diese Ontologie eine neue Hierarchie generieren muss. Das neue Berechnen der Ontologie kann in der Übersicht des SALZ Adapters (siehe [Unterabschnitt 12.3](#), Funktionalität ganz unten aufklappbar) angestoßen werden.

**UPDATES VON DRUPAL, MAINTENANCE MODE** Von Zeit zu Zeit wird eine neue Version von Drupal released. Dies ist für die Metadatenbank theoretisch irrelevant, wenn es nur ein Minor Update ist (diese sollten aber im Bezug auf aktuelle Software trotzdem eingespielt werden). Ein Major Update führt in der Regel jedoch dazu, dass das Drupal System in den Maintenance Mode versetzt wird - ein Status in dem das System aus Sicherheitsgründen offline geschaltet wird. Deswegen ist hier ein Update durchzuführen.

Updates des Drupal Systems können (in seltenen Fällen) entweder im Drupal Interface selbst durchgeführt werden (Reiter “Extend”, dann auf Link für “available updates” drücken). In der Regel führt man die Updates (auch von anderen Modulen) direkt am Server durch. Dafür in das Verzeichnis der Drupal Installation wechseln (`/var/www/html/drupal`) und folgenden Befehl ausführen:

```
1 | sudo composer update --with-all-dependencies
```

**Listing 48:** Befehl zum Updaten einer Drupal Installation.

Möglicherweise müssen im Konflikt stehende Abhängigkeiten von Hand gelöst werden. Dazu Schritt für Schritt die einzelnen Dependencies updaten und am Ende nochmals obigen Befehl ausführen.

**RDF4J WORKBENCH ACCESS** Um direkt auf dem unterliegenden Triplestore zu arbeiten, kann auf die RDF4J Workbench zugegriffen werden (URL: <https://database.visit.uni-passau.de/rdf4j-workbench>). In manchen Fällen leitet das Aufrufen der Homepage zu einem Prompt, der nach der URL, sowie Benutzer und Passwort verlangt. Für den Server ist kein User angelegt, deswegen können die beiden Felder für Benutzer und Passwort leer gelassen werden. Jedoch ist manchmal die angegebene URL inkorrekt, da

“`http`” anstatt “`https`” in der URL steht. Dieses muss angepasst werden, dann kann auf den Triplestore zugegriffen werden.

**MAVEN ABHÄNGIGKEITEN IM REST API PROJEKT** Im Java Projekt für die Vi-SIT REST API passiert ab und an der Fall, dass die Abhängigkeiten des Projekts nicht richtig interpretiert werden, was dazu führt, dass einige Packages in den implementierten Klassen als “nicht vorhanden” angezeigt werden. Durch ein erneutes Downloaden der Abhängigkeiten (durch das Kommando “Reimport” oder falls dies nicht hilft “Update Maven Indices” in der obersten `pom.xml` des Projekts) sollte sich dieses Problem lösen lassen.

M  
Name of the Pathbuilder-Tree.

Which adapter does this Pathbuilder belong to?

---

### IMPORT TEMPLATES

**Pathbuilder Definition Import**

Path to a pathbuilder definition file.

**Set default mode to**

What should the fields and groups mode be set to?

---

### EXPORT TEMPLATES

- [rdf4jpathbuildertest\\_20180315T204613](#)
- [visittestrepo\\_pathes\\_20180808T170845](#)
- [visittestrepo\\_pathes\\_20180905T182510](#)
- [visittestrepo\\_pathes\\_20181128T104132](#)
- [visittestrepo\\_pathes\\_20190107T110029](#)
- [visittestrepo\\_pathes\\_20190109T143959](#)
- [visittestrepo\\_pathes\\_20190123T151843](#)
- [visittestrepo\\_pathes\\_20190131T143855](#)

[visittest\\_20180315T204613\\_05420](#)

Abbildung 74: WissKI Pathbuilder Ansicht, die den Download der Pfad-Datei anbietet.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
    <!-- Uncomment above setting to produce a productive instance -->
</dependency>
```

Abbildung 75: pom.xml Konfiguration zum produktiven Deployment der REST API.

```
# -----
# Properties for the Ann04j, whether to connect or not connect to the database
#
# Use these to connect to the PRODUCTIVE DB
#visit.rest.sparql.endpoint.query=https://database.visit.uni-passau.de/rdf4j-server/repositories/visittestrepo
#visit.rest.sparql.endpoint.update=https://database.visit.uni-passau.de/rdf4j-server/repositories/visittestrepo/statements

# Use these to connect to the test DB
#visit.rest.sparql.endpoint.query=https://database-test.visit.uni-passau.de/rdf4j-server/repositories/visittestrepo
#visit.rest.sparql.endpoint.update=https://database-test.visit.uni-passau.de/rdf4j-server/repositories/visittestrepo/statements

# Use these to NOT connect to the DB
visit.rest.sparql.endpoint.query=none
visit.rest.sparql.endpoint.update=none
```

**Abbildung 76:** Einstellungen zum Verbinden des Triplestores. Hier gezeigt: lokale Konfiguration, während die Konfiguration für Produktiv- und Testsystem oben vorhanden aber auskommentiert ist.

## 13 APPENDIX

```

1  <?xml version="1.0"?>
2  <rdf:RDF xmlns="http://visit.de/ontologies/vismo/"
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:ns="http://www.w3.org/2003/06/sw-vocab-status/ns#"
5      xmlns:owl="http://www.w3.org/2002/07/owl#"
6      xmlns:xml="http://www.w3.org/XML/1998/namespace"
7      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
8      xmlns:skos="http://www.w3.org/2004/02/skos/core#"
9      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
10     xmlns:wot="http://xmlns.com/wot/0.1/"
11     xmlns:foaf="http://xmlns.com/foaf/0.1/"
12     xmlns:dc="http://purl.org/dc/elements/1.1/">
13     <owl:Ontology rdf:about="http://visit.de/ontologies/vismo/">
14         <owl:versionIRI rdf:resource="http://visit.de/ontologies/vismo/0.4.5/" />
15         <owl:imports rdf:resource="http://erlangen-crm.org/170309/" />
16         <owl:imports rdf:resource="http://xmlns.com/foaf/0.1/" />
17     </owl:Ontology>
18
19
20
21     <!--
22     ///////////////////////////////////////////////////
23     //
24     // Object Properties
25     //
26     ///////////////////////////////////////////////////
27     -->
28
29
30
31
32
33     <!-- http://visit.de/ontologies/vismo/containsEntry -->
34
35     <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/containsEntry">
36         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
37         <owl:inverseOf rdf:resource="http://visit.de/ontologies/vismo/isEntryIn"/>
38         <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
39         <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/ReferenceEntry"/>
40         <rdfs:comment>Reference from a vismo:Reference to a contained vismo:ReferenceEntry.</
41         rdfs:comment>
42         <rdfs:label>contains entry</rdfs:label>
43     </owl:ObjectProperty>
44
45
46     <!-- http://visit.de/ontologies/vismo/employsTraderoute -->
47
48     <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/employsTraderoute">
49         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
50         <owl:inverseOf rdf:resource="http://visit.de/ontologies/vismo/forTrade"/>
51         <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Trade"/>
52         <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/Traderoute"/>
53         <rdfs:comment>Refers from a vismo:Trade to the vismo:TradeRoute that the trade is
54         fulfilled on.</rdfs:comment>
55         <rdfs:label>employs traderoute</rdfs:label>
56     </owl:ObjectProperty>
57
58
59     <!-- http://visit.de/ontologies/vismo/endLocation -->
60
61     <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/endLocation">
62         <rdfs:subPropertyOf rdf:resource="http://visit.de/ontologies/vismo/routeLocation"/>
63         <rdfs:comment>Refers from a traderoute to the vismo:City that represents the ending
64         point for the route.</rdfs:comment>
65         <rdfs:label>end location</rdfs:label>
66     </owl:ObjectProperty>
67
68
69     <!-- http://visit.de/ontologies/vismo/entryIsAbout -->
70
71     <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/entryIsAbout">
72         <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
73         <owl:inverseOf rdf:resource="http://visit.de/ontologies/vismo/referencedByEntry"/>

```

```

74      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/ReferenceEntry"/>
75      <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
76      <rdfs:comment>Reference from a vismo:ReferenceEntry to a vismo:Resource, associating
77      the describing nature of the associated vismo:Reference.</rdfs:comment>
78      <rdfs:label>entry is about</rdfs:label>
79      </owl:ObjectProperty>
80
81
82      <!-- http://visit.de/ontologies/vismo/forTrade -->
83
84      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/forTrade">
85          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
86          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Traderoute"/>
87          <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/Trade"/>
88          <rdfs:comment>Refers from a vismo:TradeRoute to the vismo:Trade resource that
89          illustrates the trading on the given route.</rdfs:comment>
90          <rdfs:label>for trade</rdfs:label>
91      </owl:ObjectProperty>
92
93
94      <!-- http://visit.de/ontologies/vismo/hasDigitalRepresentation -->
95
96      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/hasDigitalRepresentation">
97          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
98          <owl:inverseOf rdf:resource="http://visit.de/ontologies/vismo/representsDigitally"/>
99          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
100         <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/DigitalRepresentation"/>
101         <rdfs:comment>Links from a vismo:Resource (so an object that can be further specified
102         in the ViSIT context) to a digital representation of it, e.g. a picture that shows the
103         respective resource, a 3D model, etc.</rdfs:comment>
104         <rdfs:label>has digital representation</rdfs:label>
105     </owl:ObjectProperty>
106
107
108      <!-- http://visit.de/ontologies/vismo/interactionSource -->
109
110      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/interactionSource">
111          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
112          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/MiscellaneousInteraction"/>
113          <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
114          <rdfs:label>interaction source</rdfs:label>
115      </owl:ObjectProperty>
116
117
118      <!-- http://visit.de/ontologies/vismo/interactionTarget -->
119
120      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/interactionTarget">
121          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
122          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
123          <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/MiscellaneousInteraction"/>
124
125          <rdfs:label>interaction target</rdfs:label>
126      </owl:ObjectProperty>
127
128
129      <!-- http://visit.de/ontologies/vismo/interstation -->
130
131      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/interstation">
132          <rdfs:subPropertyOf rdf:resource="http://visit.de/ontologies/vismo/routeLocation"/>
133          <rdfs:comment>Refers from a traderoute to the vismo:City that represents a
134          interstation for the route.</rdfs:comment>
135          <rdfs:label>interstation</rdfs:label>
136      </owl:ObjectProperty>
137
138
139      <!-- http://visit.de/ontologies/vismo/isEntryIn -->
140
141      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/isEntryIn">
142          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
143          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/ReferenceEntry"/>
144          <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/Reference"/>

```

```

145      <rdfs:comment>Reference from a vismo:ReferenceEntry to its encompassing
146      vismo:Resource.</rdfs:comment>
147      <rdfs:label>is entry in</rdfs:label>
148      </owl:ObjectProperty>
149
150
151      <!-- http://visit.de/ontologies/vismo/partOfTradeRoute -->
152
153      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/partOfTradeRoute">
154          <rdf:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
155          <owl:inverseOf rdf:resource="http://visit.de/ontologies/vismo/routeLocation"/>
156          <rdf:domain rdf:resource="http://visit.de/ontologies/vismo/Place"/>
157          <rdf:range rdf:resource="http://visit.de/ontologies/vismo/Traderoute"/>
158          <rdfs:comment>Refers from a vismo:City to a/multiple vismo:TradeRoute resource,
159          indicating the given city is part of a trade route and therefore its associated trade.</
160          rdfs:comment>
161          <rdfs:label>part of trade route</rdfs:label>
162      </owl:ObjectProperty>
163
164
165      <!-- http://visit.de/ontologies/vismo/reference -->
166
167      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/reference">
168          <rdf:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
169          <owl:inverseOf rdf:resource="http://visit.de/ontologies/vismo/referencedBy"/>
170          <rdf:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
171          <rdf:range rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
172          <rdfs:comment>Issues that the referenced vismo:Reference contains further and
173          descriptive information about the given vismo:Resource entity.</rdfs:comment>
174          <rdfs:label>reference</rdfs:label>
175      </owl:ObjectProperty>
176
177
178      <!-- http://visit.de/ontologies/vismo/referencedBy -->
179
180      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/referencedBy">
181          <rdf:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
182          <rdf:domain rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
183          <rdf:range rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
184          <rdfs:comment>Refers to the vismo:Resource entities that reference this
185          vismo:Reference and therefore this entity contains further and descriptive information
186          about the resource entities.</rdfs:comment>
187          <rdfs:label>referenced by</rdfs:label>
188      </owl:ObjectProperty>
189
190
191      <!-- http://visit.de/ontologies/vismo/referencedByEntry -->
192
193      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/referencedByEntry">
194          <rdf:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
195          <rdf:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
196          <rdf:range rdf:resource="http://visit.de/ontologies/vismo/ReferenceEntry"/>
197          <rdfs:comment>Reference from a vismo:Resource to a given vismo:ReferenceEntry,
198          indicating that the associated vismo:Reference contains information about the former.</
199          rdfs:comment>
200          <rdfs:label>referenced by entry</rdfs:label>
201      </owl:ObjectProperty>
202
203
204      <!-- http://visit.de/ontologies/vismo/representsDigitally -->
205
206      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/representsDigitally">
207          <rdf:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
208          <rdf:domain rdf:resource="http://visit.de/ontologies/vismo/DigitalRepresentation"/>
209          <rdf:range rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
210          <rdfs:comment>Refers from a given digital representation (a picture, 3D model, etc.)
211          back to the vismo:Resource that it originally represents.</rdfs:comment>
212          <rdfs:label>represents digitally</rdfs:label>
213      </owl:ObjectProperty>
214
215      <!-- http://visit.de/ontologies/vismo/routeLocation -->
216
217      <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/routeLocation">
```

```

216   <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topObjectProperty"/>
217   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Traderroute"/>
218   <rdfs:range rdf:resource="http://visit.de/ontologies/vismo/Place"/>
219   <rdfs:comment>Refers (with different sub-properties) from a vismo:TradeRoute to a
220   vismo:City that is located on said route.</rdfs:comment>
221   <rdfs:label>route location</rdfs:label>
222 </owl:ObjectProperty>

223
224
225 <!-- http://visit.de/ontologies/vismo/startLocation -->
226
227 <owl:ObjectProperty rdf:about="http://visit.de/ontologies/vismo/startLocation">
228   <rdfs:subPropertyOf rdf:resource="http://visit.de/ontologies/vismo/routeLocation"/>
229   <rdfs:comment>Refers from a traderroute to the vismo:City that represents the starting
230   point for the route.</rdfs:comment>
231   <rdfs:label>start location</rdfs:label>
232 </owl:ObjectProperty>

233
234
235 <!--
236 ///////////////////////////////////////////////////
237 //
238 // Data properties
239 //
240 ///////////////////////////////////////////////////
241 -->

242
243
244
245 <!-- http://visit.de/ontologies/vismo/buildingHistory -->
246
247 <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/buildingHistory">
248   <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
249   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Architecture"/>
250   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
251   <rdfs:comment>Property used to (freely) describe the building history of a
252   vismo:Architecture entity.</rdfs:comment>
253   <rdfs:label>building history</rdfs:label>
254 </owl:DatatypeProperty>

255
256
257 <!-- http://visit.de/ontologies/vismo/comment -->
258
259 <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/comment">
260   <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
261   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
262   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
263   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">This is a
264   comment about a given vismo entity.</rdfs:comment>
265   <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://visit.
266   de/ontologies/vismo</rdfs:isDefinedBy>
267   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">comment</
268   rdfs:label>
269 </owl:DatatypeProperty>

270
271 <!-- http://visit.de/ontologies/vismo/description -->
272
273 <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/description">
274   <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
275   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
276   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
277   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">This property
278   defines a (historical) description for a vismo entity.</rdfs:comment>
279   <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://visit.
280   de/ontologies/vismo/</rdfs:isDefinedBy>
281   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">description</
282   rdfs:label>
283 </owl:DatatypeProperty>

284 <!-- http://visit.de/ontologies/vismo/entryPages -->
285
286 <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/entryPages">
```

```

287      <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
288      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/ReferenceEntry"/>
289      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
290      <rdfs:comment>The range of pages that a given vismo:ReferenceEntry references of a
291      vismo:Reference entity.</rdfs:comment>
292      <rdfs:label>entry pages</rdfs:label>
293    </owl:DatatypeProperty>
294
295
296    <!-- http://visit.de/ontologies/vismo/helpfulLinks -->
297
298    <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/helpfulLinks">
299      <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
300      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
301      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
302      <rdfs:comment>This property is used to conveniently collect links to online resources
303      that contain further information of the associated vismo:Resource.</rdfs:comment>
304      <rdfs:label>helpful links</rdfs:label>
305    </owl:DatatypeProperty>
306
307
308    <!-- http://visit.de/ontologies/vismo/iconography -->
309
310    <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/iconography">
311      <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
312      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
313      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
314      <rdfs:comment>A property to associate iconography to a given vismo:Resource entity.</
315      rdfs:comment>
316      <rdfs:label>iconography</rdfs:label>
317    </owl:DatatypeProperty>
318
319
320    <!-- http://visit.de/ontologies/vismo/innerDescription -->
321
322    <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/innerDescription">
323      <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
324      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Architecture"/>
325      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
326      <rdfs:comment>Property used to describe the interior of a vismo:Architecture entity.</
327      rdfs:comment>
328      <rdfs:label>inner description</rdfs:label>
329    </owl:DatatypeProperty>
330
331
332    <!-- http://visit.de/ontologies/vismo/keyword -->
333
334    <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/keyword">
335      <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
336      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
337      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
338      <rdfs:comment>This property is used to address keywords for a vismo:Resource entity.
These refer to more general topics that can be addressed to anything out of the VisMo
domain, for example "Trade", "War", "Peace", or overall
temporal associations.</rdfs:comment>
      <rdfs:label>keyword</rdfs:label>
339    </owl:DatatypeProperty>
340
341
342
343    <!-- http://visit.de/ontologies/vismo/literature -->
344
345    <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/literature">
346      <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
347      <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
348      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
349      <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">This property
350      defines a literature entry that contains further information about the given vismo
entity.</rdfs:comment>
      <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://visit.
de/ontologies/vismo/</rdfs:isDefinedBy>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">literature</
351      rdfs:label>
352    </owl:DatatypeProperty>
353
354
355

```

```

356
357      <!-- http://visit.de/ontologies/vismo/outerDescription -->
358
359      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/outerDescription">
360          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
361          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Architecture"/>
362          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
363          <rdfs:comment>Property used to describe the exterior of a vismo:Architecture entity.</rdfs:comment>
364          <rdfs:label>outer description</rdfs:label>
365      </owl:DatatypeProperty>
366
367
368
369      <!-- http://visit.de/ontologies/vismo/pages -->
370
371      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/pages">
372          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
373          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
374          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
375          <rdfs:comment>Number of pages of a given vismo:Reference entity.</rdfs:comment>
376          <rdfs:label>pages</rdfs:label>
377      </owl:DatatypeProperty>
378
379
380
381      <!-- http://visit.de/ontologies/vismo/publisher -->
382
383      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/publisher">
384          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
385          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
386          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
387          <rdfs:comment>The publisher of the given vismo:Reference entity.</rdfs:comment>
388          <rdfs:label>publisher</rdfs:label>
389      </owl:DatatypeProperty>
390
391
392
393      <!-- http://visit.de/ontologies/vismo/series -->
394
395      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/series">
396          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
397          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
398          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
399          <rdfs:label>series</rdfs:label>
400      </owl:DatatypeProperty>
401
402
403
404      <!-- http://visit.de/ontologies/vismo/superordinateTitle -->
405
406      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/superordinateTitle">
407          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
408          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Title"/>
409          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
410          <rdfs:comment>Textual String to name the title of the superordinate reference collection that incorporates the associated vismo:Reference entity.</rdfs:comment>
411          <rdfs:label>superordinate title</rdfs:label>
412      </owl:DatatypeProperty>
413
414
415
416      <!-- http://visit.de/ontologies/vismo/technicalMetadata -->
417
418      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/technicalMetadata">
419          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
420          <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/DigitalRepresentation"/>
421          <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
422          <rdfs:comment>Refers from a vismo:DigitalRepresentation to a JSON formatted String that represents the technical metadata that is produced by the process that creates given digital representation.</rdfs:comment>
423          <rdfs:label>technical metadata</rdfs:label>
424      </owl:DatatypeProperty>
425
426
427
428      <!-- http://visit.de/ontologies/vismo/thumbnail -->
429
430      <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/thumbnail">
431          <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>

```

```

432   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
433   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
434   <rdfs:comment>A thumbnail generated for the respective digital representation. In
435   general a base 64 encoding.</rdfs:comment>
436   <rdfs:label>thumbnail</rdfs:label>
437 </owl:DatatypeProperty>

438
439
440 <!-- http://visit.de/ontologies/vismo/volume -->
441
442 <owl:DatatypeProperty rdf:about="http://visit.de/ontologies/vismo/volume">
443   <rdfs:subPropertyOf rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
444   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/Reference"/>
445   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
446   <rdfs:comment>Number of volumes of a given publication series.</rdfs:comment>
447   <rdfs:label>volume</rdfs:label>
448 </owl:DatatypeProperty>

449
450
451
452 <!-- http://www.w3.org/2002/07/owl#topDataProperty -->
453
454 <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#topDataProperty">
455   <rdfs:domain rdf:resource="http://visit.de/ontologies/vismo/DigitalRepresentation"/>
456   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
457 </rdf:Description>

458
459
460
461 <!--
462 //////////////////////////////////////////////////////////////////
463 // Classes
464 //
465 //////////////////////////////////////////////////////////////////
466 -->

467
468
469
470
471 <!-- http://visit.de/ontologies/vismo/Activity -->
472
473 <owl:Class rdf:about="http://visit.de/ontologies/vismo/Activity">
474   <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E7_Activity"/>
475   <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
476   <rdfs:comment>Activities in the ViSIT context are any type of timely historical event
477   that can contribute a timely frame for associated ViSIT concepts. For example &quot;
478   World War II&quot;, &quot;The battle for town x&quot;, etc.</rdfs:comment>
479   <rdfs:label>Activity</rdfs:label>
480 </owl:Class>

481
482
483 <!-- http://visit.de/ontologies/vismo/Architecture -->
484
485 <owl:Class rdf:about="http://visit.de/ontologies/vismo/Architecture">
486   <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E53_Place"/>
487   <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E84_Information_Carrier
488   "/>
489   <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
490   <rdfs:comment>Architecture in the ViSIT context describes every building or
491   architectural production that has been erected by mankind in some way.</rdfs:comment>
492   <rdfs:label>Architecture</rdfs:label>
493 </owl:Class>

494
495
496 <!-- http://visit.de/ontologies/vismo/BishopricAffiliation -->
497
498 <owl:Class rdf:about="http://visit.de/ontologies/vismo/BishopricAffiliation">
499   <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
500   <rdfs:comment>This class comprises headwords for bishopric affiliations for
501   vismo:Architecture resources.</rdfs:comment>
502   <rdfs:label>Bishopric Affiliation</rdfs:label>
503 </owl:Class>

504
505 <!-- http://visit.de/ontologies/vismo/Country -->

```

```

506
507      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Country">
508          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E53_Place"/>
509          <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
510          <rdfs:label>Country</rdfs:label>
511      </owl:Class>
512
513
514      <!-- http://visit.de/ontologies/vismo/Dating -->
515
516      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Dating">
517          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E52_Time-Span"/>
518          <rdfs:comment>More specific class of the E52_TimeSpan and used in the ViSIT context
519          to give temporal associations with various entities.</rdfs:comment>
520          <rdfs:label>Dating</rdfs:label>
521      </owl:Class>
522
523
524      <!-- http://visit.de/ontologies/vismo/Description -->
525
526      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Description">
527          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
528          <rdfs:comment>Descriptions comprise characteristic types for objects in the domain of
529          museums. Therefore these are for example "painting", "oil painting";
530          &quot;chest&quot;; etc.</rdfs:comment>
531          <rdfs:label>Description</rdfs:label>
532      </owl:Class>
533
534
535      <!-- http://visit.de/ontologies/vismo/DigitalRepresentation -->
536
537      <owl:Class rdf:about="http://visit.de/ontologies/vismo/DigitalRepresentation">
538          <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
539          <rdfs:comment>A digital representation symbolises a multimedia representation of a
540          vismo:Resource that can be illustrated in some way. These incorporate pictures, videos,
541          audio files, and 3D models in particular.</rdfs:comment>
542          <rdfs:label>Digital Representation</rdfs:label>
543      </owl:Class>
544
545
546      <!-- http://visit.de/ontologies/vismo/Function -->
547
548      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Function">
549          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
550          <rdfs:comment>Functions relate to semantical and functional properties that are
551          inherited by vismo:Object as well as vismo:Architecture and their vismo:Structural
552          Evolution resources. Both an object as well as an architecture could exert "
553          military" functions.</rdfs:comment>
554          <rdfs:label>Function</rdfs:label>
555      </owl:Class>
556
557
558      <!-- http://visit.de/ontologies/vismo/GeographicalAffiliation -->
559
560      <owl:Class rdf:about="http://visit.de/ontologies/vismo/GeographicalAffiliation">
561          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
562          <rdfs:comment>This class comprises headwords for geographical affiliations for
563          vismo:Architecture resources.</rdfs:comment>
564          <rdfs:label>Geographical Affiliation</rdfs:label>
565      </owl:Class>
566
567
568      <!-- http://visit.de/ontologies/vismo/Group -->
569
570      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Group">
571          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E74_Group"/>
572          <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
573          <rdfs:comment>This more general class will comprise different groups of people that
574          are associated in the VisMo context. In the first instance these are WorkingGroups (
      Werkst\"aften) and joint practices (Soziet\"at).</rdfs:comment>
          <rdfs:label>Group</rdfs:label>
      </owl:Class>

```

```

575      <!-- http://visit.de/ontologies/vismo/GroupDescription -->
576
577      <owl:Class rdf:about="http://visit.de/ontologies/vismo/GroupDescription">
578          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
579          <rdfs:comment>This Class comprises descriptive types for all kinds of groups that
580              are associated in the VisMo context, such as Werkstatt and Soziet\"a t.</rdfs:comment>
581          <rdfs:label>Group Description</rdfs:label>
582      </owl:Class>
583
584
585
586      <!-- http://visit.de/ontologies/vismo/HistoricalChange -->
587
588      <owl:Class rdf:about="http://visit.de/ontologies/vismo/HistoricalChange">
589          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E9_Move"/>
590          <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
591          <rdfs:label>HistoricalChange</rdfs:label>
592      </owl:Class>
593
594
595
596      <!-- http://visit.de/ontologies/vismo/InscriptionType -->
597
598      <owl:Class rdf:about="http://visit.de/ontologies/vismo/InscriptionType">
599          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
600          <rdfs:comment>This E55_Type describes the type of an Inscription, done on various
601              vismo:Object entities.</rdfs:comment>
602          <rdfs:label>Inscription Type</rdfs:label>
603      </owl:Class>
604
605
606
607      <!-- http://visit.de/ontologies/vismo/Institution -->
608
609      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Institution">
610          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E53_Place"/>
611          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E74_Group"/>
612          <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
613          <rdfs:comment>An institution in the ViSIT context is primarily used for museums,
614              which inherit both the properties of a E53_Place as well as a E74_Group. This is
615              necessary to make instances of this class be able to represent a spatial entity as well
616              as an entity that can for example hold vismo:Objects.</rdfs:comment>
617          <rdfs:label>Institution</rdfs:label>
618      </owl:Class>
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645

```

```

646    </owl:Class>
647
648
649
650    <!-- http://visit.de/ontologies/vismo/Object -->
651
652    <owl:Class rdf:about="http://visit.de/ontologies/vismo/Object">
653        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E84_Information_Carrier"
654            "/>
655        <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
656        <rdfs:comment>Objects in the ViSiT context subsume all sorts of items that are
657        displayed in a museum.</rdfs:comment>
658        <rdfs:label>Object</rdfs:label>
659
660    </owl:Class>
661
662
663    <!-- http://visit.de/ontologies/vismo/OrderAffiliation -->
664
665    <owl:Class rdf:about="http://visit.de/ontologies/vismo/OrderAffiliation">
666        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
667        <rdfs:comment>This class comprises headwords for order affiliations for
668        vismo:Architecture resources.</rdfs:comment>
669        <rdfs:label>Order Affiliation</rdfs:label>
670
671    </owl:Class>
672
673
674    <!-- http://visit.de/ontologies/vismo/Person -->
675
676    <owl:Class rdf:about="http://visit.de/ontologies/vismo/Person">
677        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E21_Person"/>
678        <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
679        <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
680        <rdfs:label>Person</rdfs:label>
681
682    </owl:Class>
683
684
685    <!-- http://visit.de/ontologies/vismo/Place -->
686
687    <owl:Class rdf:about="http://visit.de/ontologies/vismo/Place">
688        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E53_Place"/>
689        <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
690        <rdfs:comment>All cities, towns, settlements etc. of some sort are subsumed under
691        this class.</rdfs:comment>
692        <rdfs:label>Place</rdfs:label>
693
694    </owl:Class>
695
696
697    <!-- http://visit.de/ontologies/vismo/Profession -->
698
699    <owl:Class rdf:about="http://visit.de/ontologies/vismo/Profession">
700        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
701        <rdfs:comment>Professions subsume all roles, employments, titles, authorities, etc.
702        for persons that are inherent in the cultural heritage domain.</rdfs:comment>
703        <rdfs:label>Profession</rdfs:label>
704
705    </owl:Class>
706
707
708    <!-- http://visit.de/ontologies/vismo/Reference -->
709
710    <owl:Class rdf:about="http://visit.de/ontologies/vismo/Reference">
711        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E84_Information_Carrier"
712            "/>
713        <rdfs:comment>Used in the cultural use case of Visit as a reference to various
714        textual information objects that contained further and descriptive information about a
715        given Visit resource.</rdfs:comment>
716        <rdfs:label>Reference</rdfs:label>
717
718    </owl:Class>
719
720
721
722    <!-- http://visit.de/ontologies/vismo/ReferenceEntry -->
723
724    <owl:Class rdf:about="http://visit.de/ontologies/vismo/ReferenceEntry">
725        <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E84_Information_Carrier"
726            "/>

```

```

717     <rdfs:comment>A Reference Entry contains further information about the reference of a
718     vismo:Resource in a given vismo:Reference entity, like the page numbers for example.</
719     rdfs:comment>
720         <rdfs:label>Reference Entry</rdfs:label>
721     </owl:Class>

722
723     <!-- http://visit.de/ontologies/vismo/ReferenceType -->
724
725     <owl:Class rdf:about="http://visit.de/ontologies/vismo/ReferenceType">
726         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
727             <rdfs:comment>Summarises the various types that references in the cultural heritage
728             domain can have.</rdfs:comment>
729             <rdfs:label>Reference Type</rdfs:label>
730     </owl:Class>

731
732
733     <!-- http://visit.de/ontologies/vismo/Resource -->
734
735     <owl:Class rdf:about="http://visit.de/ontologies/vismo/Resource">
736         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E1_CRM_Entity"/>
737         <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
738             <rdfs:comment>A vismo:Resource adds descriptive functionality to the resources used
739             in the ViSIT context, therefore adding the possibilities of adding comments,
740             descriptions, as well as literature information to the given resource.</rdfs:comment>
741             <rdfs:label>Resource</rdfs:label>
742     </owl:Class>

743
744     <!-- http://visit.de/ontologies/vismo/Room -->
745
746     <owl:Class rdf:about="http://visit.de/ontologies/vismo/Room">
747         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E53_Place"/>
748         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E84_Information_Carrier
749             "/>
750             <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
751                 <rdfs:comment>A room in the classic sense. Can only be associated with its
752                 vismo:Architecture entity that contains it.</rdfs:comment>
753                 <rdfs:label>Room</rdfs:label>
754     </owl:Class>

755
756     <!-- http://visit.de/ontologies/vismo/SacralBuilding -->
757
758     <owl:Class rdf:about="http://visit.de/ontologies/vismo/SacralBuilding">
759         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
760             <rdfs:comment>A further type characterisation for vismo:Architecture entities,
761             classifying them by a sacral type.</rdfs:comment>
762             <rdfs:label>Sacral Building</rdfs:label>
763     </owl:Class>

764
765
766     <!-- http://visit.de/ontologies/vismo/SecularBuilding -->
767
768     <owl:Class rdf:about="http://visit.de/ontologies/vismo/SecularBuilding">
769         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
770             <rdfs:comment>A further type characterisation for vismo:Architecture entities,
771             classifying them by a secular type.</rdfs:comment>
772             <rdfs:label>Secular Building</rdfs:label>
773     </owl:Class>

774
775
776     <!-- http://visit.de/ontologies/vismo/StructuralEvolution -->
777
778     <owl:Class rdf:about="http://visit.de/ontologies/vismo/StructuralEvolution">
779         <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E11_Modification"/>
780         <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
781             <rdfs:comment>A vismo:StructuralEvolution changes a vismo:Architecture entity in some
782             way. A change in its basic vismo:Function can thereby be established. For example a
783             castle that changes from its military function to a museum.</rdfs:comment>
784             <rdfs:label>StructuralEvolution</rdfs:label>
785     </owl:Class>

```

```

786      <!-- http://visit.de/ontologies/vismo/Technique -->
787
788      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Technique">
789          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
790          <rdfs:comment>Techniques subsume the naming of production processes, which have the
791          result of producing an vismo:Object that are associated with the cultural heritage
792          domain.</rdfs:comment>
793          <rdfs:label>Technique</rdfs:label>
794      </owl:Class>

795
796
797      <!-- http://visit.de/ontologies/vismo/Title -->
798
799      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Title">
800          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E35_Title"/>
801          <rdfs:comment>A Class to comprise a title in combination with a superordinate title
802          of a reference collection that contains this vismo:Reference entity.</rdfs:comment>
803          <rdfs:label>Title</rdfs:label>
804      </owl:Class>

805
806
807      <!-- http://visit.de/ontologies/vismo/Trade -->
808
809      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Trade">
810          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E9_Move"/>
811          <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
812          <rdfs:comment>This class subsumes a trade of some or more vismo:TradeGood entities. A
813          trade should always be associated with a vismo:TradeRoute.</rdfs:comment>
814          <rdfs:label>Trade</rdfs:label>
815      </owl:Class>

816
817
818      <!-- http://visit.de/ontologies/vismo/TradeGood -->
819
820      <owl:Class rdf:about="http://visit.de/ontologies/vismo/TradeGood">
821          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E55_Type"/>
822          <rdfs:comment>Tradegoods subsume titled names for the goods that are transported and
823          sold on vismo:TradeRoute objects.</rdfs:comment>
824          <rdfs:label>Tradegood</rdfs:label>
825      </owl:Class>

826
827
828      <!-- http://visit.de/ontologies/vismo/Traderoute -->
829
830      <owl:Class rdf:about="http://visit.de/ontologies/vismo/Traderoute">
831          <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
832          <rdfs:comment>A vismo:TradeRoute encompasses several vismo:City entities that are
833          associated with the vismo:Trade that is associated with the given trade route. These
834          cities can thereby be starting or end location, as well as an intermediate station.</
835          rdfs:comment>
836          <rdfs:label>Traderoute</rdfs:label>
837      </owl:Class>

838
839
840      <!-- http://visit.de/ontologies/vismo/WorkingGroup -->
841
842      <owl:Class rdf:about="http://visit.de/ontologies/vismo/WorkingGroup">
843          <rdfs:subClassOf rdf:resource="http://erlangen-crm.org/170309/E74_Group"/>
844          <rdfs:subClassOf rdf:resource="http://visit.de/ontologies/vismo/Resource"/>
845          <rdfs:comment>Frauke :)</rdfs:comment>
846          <rdfs:label>WorkingGroup</rdfs:label>
847      </owl:Class>
848
849  </rdf:RDF>

```

Listing 49: VisMo Ontologie in der letzten (englischen) Version.

```

1  {
2      "Object": {
3          "type": "http://visit.de/ontologies/vismo/Object",
4          "object_identifiedby_title": "string",
5          "object_has_description": "string",
6          "object_exemplifies_function": "string",
7          "object_inventory_number": "string",

```

<b>Adapter Name *</b>	visittestrepo	Machine name: visomotes
The human-readable name of this adapter. This name must be unique.		
<b>Description</b>	visittestrepo	
<p>The text will be displayed on the <i>adapter collection</i> page.</p> <p><input checked="" type="checkbox"/> Writable</p> <p>Is this Adapter writable?</p> <p><input checked="" type="checkbox"/> Preferred Local Store</p> <p>Is this Adapter the preferred local store?</p> <p><b>Read URL</b></p> <pre>http://localhost:8081/rdf4j-server/repositories/visittestrepo</pre> <p><b>Write URL</b></p> <pre>http://localhost:8081/rdf4j-server/repositories/visittestrepo/statements</pre> <p><input type="checkbox"/> Use graph independent rewriting rewrite queries, so that remote SPARQL stores with non-standard dataset handling do always answer right</p> <p><b>Default Graph URL *</b></p> <pre>http://visit.de/data/</pre> <p>Graph URL that is used to store triples in by default. May also be used as a base for new entity URLs.</p> <p><b>Ontology graphs</b></p>		

Graphs that are considered to be containing ontology information. These are used to compute class and property information like hierarchies, domain/range, etc. Leave empty if system automatically detect the graphs.

**Abbildung 77:** Übersicht der Konfiguration eines WissKI Salz Adapters, Teil 1.

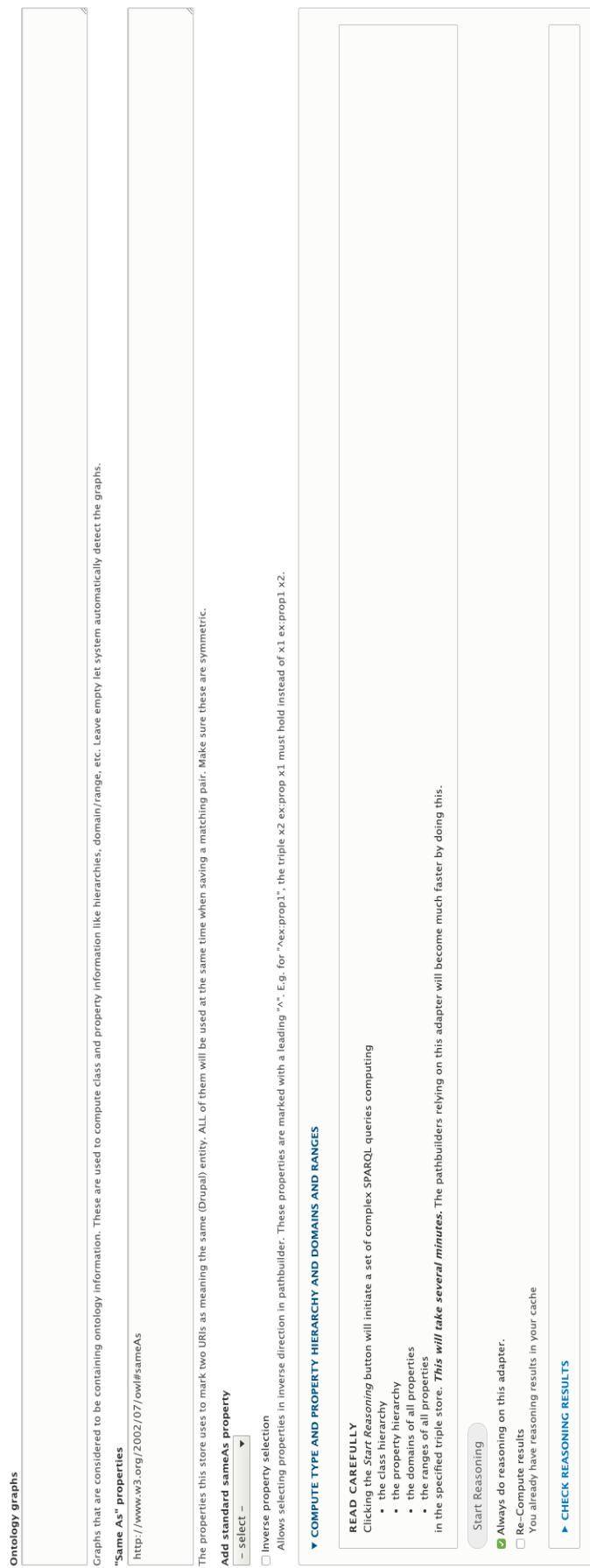


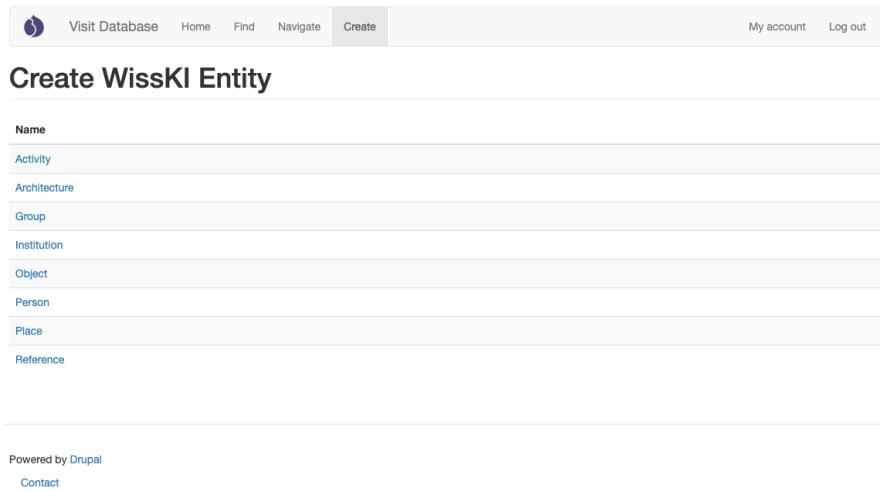
Abbildung 78: Übersicht der Konfiguration eines WissKI Salz Adapters, Teil 2.

Select the store for which you want to load an ontology.			
Currently loaded Ontology:			
Name	Iri	Version	Graph
<a href="#">http://visit.de/Ontologies/vismo/</a>	none	<a href="http://visit.de/ontologies/vismo/0.4.5/">http://visit.de/ontologies/vismo/0.4.5/</a>	<a href="https://a.uiguu.se/Qb0jthNQ28qD.owl">https://a.uiguu.se/Qb0jthNQ28qD.owl</a>
<a href="#">http://erlangen-crm.org/170309/</a>	none	none	<a href="http://erlangen-crm.org/170309/">http://erlangen-crm.org/170309/</a>
<a href="#">http://xmins.com/foaf/0.1/</a>	none	none	<a href="http://xmins.com/foaf/0.1/">http://xmins.com/foaf/0.1/</a>
<a href="#">Delete Ontology</a>			
Short Name	URI		
ref	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>		
skos	<a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a>		
protege	<a href="http://protege.stanford.edu/plugins/owl/protege@">http://protege.stanford.edu/plugins/owl/protege@</a>		
xsp	<a href="http://www.owl-ontologies.com/2005/08/07/xsp.owl#">http://www.owl-ontologies.com/2005/08/07/xsp.owl#</a>		
owl	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>		
xsd	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>		
swrl	<a href="http://www.w3.org/2003/11/swrl#">http://www.w3.org/2003/11/swrl#</a>		
ecrm	<a href="http://erlangen-crm.org/170309/">http://erlangen-crm.org/170309/</a>		
swrlb	<a href="http://www.w3.org/2003/11/swrlb#">http://www.w3.org/2003/11/swrlb#</a>		
rdfs	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>		
crm	<a href="http://cidoc-crm.org/cidoc-crm/">http://cidoc-crm.org/cidoc-crm/</a>		
xml	<a href="http://www.w3.org/XML/1998/namespace">http://www.w3.org/XML/1998/namespace</a>		
schema	<a href="http://schema.org/">http://schema.org/</a>		
terms	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>		
ns	<a href="http://www.w3.org/2003/06/sw-vocab-status/ns#">http://www.w3.org/2003/06/sw-vocab-status/ns#</a>		
wot	<a href="http://xmins.com/wot/0.1/">http://xmins.com/wot/0.1/</a>		
fns:f	<a href="https://xmins.com/fns:f/n1/">https://xmins.com/fns:f/n1/</a>		

**Abbildung 79:** Detailansicht einer definierten Ontology im WissKI System am Beispiel VisMo für das ViSIT Projekt.

Edit Pathbuilder: visitstestrepo_paths						
Home > Administration > Configuration > WissKI > Pathbuilders		Pathbuilders				
		<a href="#">+ Add Path</a> <a href="#">+ Add Existing Path</a>				
PATH						
TITLE	PATH	ENABLED	FIELD TYPE	CARDINALITY	WEIGHT	OPERATIONS
+ Object	Group [http://visit.de/ontologies/vismo/Object]	<input checked="" type="checkbox"/>		Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_identifiedBy_Title	http://visit.de/ontologies/vismo/Object -> ecrm:P1_isIdentifiedBy -> ecrm:E35_Title	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_has_Description	http://visit.de/ontologies/vismo/Object -> ecrm:P2_has_Type -> http://visit.de/ontologies/vismo/Description	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_exemplifies_Function	http://visit.de/ontologies/vismo/Object -> ecrm:P137_exemplifies -> http://visit.de/ontologies/vismo/Function	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_Dating	Group [http://visit.de/ontologies/vismo/Object -> ecrm:P160_has_temporal_Projection -> http://visit.de/ontologies/vismo/Dating]	<input checked="" type="checkbox"/>		Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_Dating_start	http://visit.de/ontologies/vismo/Object -> ecrm:P160_has_temporal_Projection -> http://visit.de/ontologies/vismo/Dating	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_Dating_end	http://visit.de/ontologies/vismo/Object -> ecrm:P160_has_temporal_Projection -> http://visit.de/ontologies/vismo/Dating	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_Dating_sometime	http://visit.de/ontologies/vismo/Object -> ecrm:P160_has_temporal_Projection -> http://visit.de/ontologies/vismo/Dating	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_Dating_century	http://visit.de/ontologies/vismo/Object -> ecrm:P160_has_temporal_Projection -> http://visit.de/ontologies/vismo/Dating	<input checked="" type="checkbox"/>	List (text)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_composedOf_Object	http://visit.de/ontologies/vismo/Object -> ecrm:P46_is_composed_of -> http://visit.de/ontologies/vismo/Object	<input checked="" type="checkbox"/>	Entity reference	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_PartOf_Object	http://visit.de/ontologies/vismo/Object -> ecrm:P46i_forms_Part_of -> http://visit.de/ontologies/vismo/Object	<input checked="" type="checkbox"/>	Entity reference	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_description	http://visit.de/ontologies/vismo/Object	<input checked="" type="checkbox"/>	Text (plain, long)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_producedBy_Production	Group [http://visit.de/ontologies/vismo/Object -> ecrm:P108i_wasProduced_by -> ecrm:E12_Production]	<input checked="" type="checkbox"/>		Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Object_employs_Material	http://visit.de/ontologies/vismo/Object -> ecrm:P108i_wasProduced_by -> ecrm:E12_Production -> ecrm:P126_employed -> ecrm:E57_Material	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>
+ Production_use_d_Technique	http://visit.de/ontologies/vismo/Object -> ecrm:P108i_wasProduced_by -> ecrm:E12_Production -> ecrm:P32_used_general_technique -> http://visit.de/ontologies/vismo/technique	<input checked="" type="checkbox"/>	Text (plain)	Unlimited	Unlimited	<a href="#">Edit</a> <a href="#">▼</a>

Abbildung 80: Übersicht der ersten Pfade des für das ViSIT Projekt definierten Pathbuilders.



**Abbildung 81:** Ausgangs-Interface zum Erzeugen einer Hauptentität im WissKI System.

```

8   "object_description": "string_long",
9   "object_comment": "string_long",
10  "object_keyword": "string",
11  "object_iconography": "string",
12  "object_literature": "string_long",
13  "object_current_owner": "entity_reference (http://visit.de/ontologies/vismo/Institution)",
14
15  "object_current_location": "entity_reference (http://visit.de/ontologies/vismo/Institution)",
16  "object_currentlocation_arch": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
17  "object_composedof_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
18  "object_partof_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
19  "object_tookpartin_activity": "entity_reference (http://visit.de/ontologies/vismo/Activity)",
20  "object_depicts_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
21  "object_depicts_architecture": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
22  "object_depicts_place": "entity_reference (http://visit.de/ontologies/vismo/Place)",
23  "object_depicts_activity": "entity_reference (http://visit.de/ontologies/vismo/Activity)",
24
25  "object_helpfullinks": "string",
26  "object_thumbnail": "image",
27  "object_prefentifier_inscriptio": {
28    "type": "http://erlangen-crm.org/170309/E34\_Inscription",
29    "inscription_text": "string",
30    "inscription_has_type": "string",
31    "inscription_signature": "string",
32    "inscription_mounting": "string",
33    "inscription_date": "string"
34  },
35  "object_producedby_production": {
36    "type": "http://erlangen-crm.org/170309/E12\_Production",
37    "object_employs_material": "string",
38    "production_used_technique": "string",
39    "production_doneby_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
40    "production_doneby_group": "entity_reference (http://visit.de/ontologies/vismo/Group)",
41    "production_tookplaceat_place": "entity_reference (http://visit.de/ontologies/vismo/Place)",
42    "production_dating": {
43      "object_prod_dating_start": "string",
44      "object_prod_dating_end": "string",
45      "production_date_sometime": "string",
46      "object_prod_dating_century": "list_string"
47    }
48  },
49  "object_has_dimension": {
50    "type": "http://erlangen-crm.org/170309/E54\_Dimension",
51    "dimension_has_measurementunit": "string",
52    "dimension_hasvalue": "string"
53  }
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
954
955
956
956
957
958
958
959
959
960
961
962
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
```

Home » Create WissKI Entity

<b>TITEL</b>	<input type="text"/> <input type="button" value="Show row weights"/>
<b>OBJEKTEZEICHNUNG</b>	<input type="text"/> <input type="button" value="Show row weights"/>
<b>FUNKTION</b>	<input type="text"/> <input type="button" value="Show row weights"/>
<b>OBJEKT BESTEHT AUS: (TEILOBJECTEN)</b>	<input type="text"/> <input type="button" value="Show row weights"/>
<b>TEILOBJECT VON: (OBJEKT)</b>	<input type="text"/> <input type="button" value="Show row weights"/>

Geben Sie hier den Titel des Objekts ein, das Sie anzeigen wollen, z.B. "Einzug der Kriemhild".

Fügen Sie hier möglichst präzise und schlagwortartig ein, um was für einen Objekttyp es sich handelt, z.B. "Gemälde", "Turramtung" oder "Henkelkrug". Achtung! Für den Titel eines Objekts (z.B. "Mona Lisa") steht das Feld "Titel" zur Verfügung.

Hier können Sie beschreiben, was das Objekt ursprünglich für eine Funktion hatte und in welchen Zusammenhangen es genutzt wurde, z.B.: Spezifikation einer Bügeleisflasche als "Bierflasche" oder eines Henkelkrugs als "Essigkrug".

**ALLGEMEINE OBJEKTDATIERUNG**  
   
 In dieser Gruppe können Sie Angaben zur allgemeinen Objektdatierung machen.

Abbildung 82: Eingabe-Interface für ein Ausstellungsobjekt im ViSIT WissKI System.

The screenshot shows a web-based application interface for managing historical objects. At the top, there's a navigation bar with links for 'Visit Database', 'Home', 'Find', 'Navigate', 'Create', 'My account', and 'Log out'. Below the navigation is a breadcrumb trail: 'Navigate / Object / Partisane (587)'. The main title is 'Partisane (587)'. Underneath, there are tabs for 'View', 'Edit' (which is selected), 'Delete', 'Triples', and 'Devel'. A detailed list of object properties follows:

- Objektbezeichnung**: Partisane
- Prunkpartisane**
- Inscription**
- Text**: I.P.D.G.E.P.S.R.-I.P.E.G.D.L.
- Anbringung**: Über dem Passauer Wolf
- Datering**: 1698
- Inventarnummer**: 00045
- Darstellung**: Passauer Wolf, Wappen des Fürstbischofs Johann Philipp Graf von Lamberg
- Standort (Museum)**: Oberhausmuseum
- Herstellung**
- Material**: Eisen
- Holz**
- Technik**: Geschmiedet
- Objekt besteht aus: (Teilobjekten)**: Wappen des Fürstbischofs Johann Philipp von Lamberg
- Standort (Bauwerk)**
- Veste Oberhaus**
- Maße**
- Abmessung**: Breite
- Wert**: 17,7 cm
- Abmessung**: 215
- Wert**: 215 cm
- Allgemeine Objektdatierung**: freie Datierung

Below the properties, there's a section for 'Allgemeine Objektdatierung' with the option 'freie Datierung'.

Abbildung 83: Beispiel Ausgabe-Interface einer Partisane des ViSIT WissKI Systems.

```

51 },
52 "object_transferred_custody": {
53   "type": "http://erlangen-crm.org/170309/E10_Transfer_of_Custody",
54   "custody_receiving_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
55   "custody_receiving_group": "entity_reference (http://visit.de/ontologies/vismo/Group)",
56   "custody_from_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
57   "custody_from_group": "entity_reference (http://visit.de/ontologies/vismo/Group)",
58   "object_toc_dating": {
59     "object_toc_dating_exact": "datetime",
60     "object_toc_dating_start": "string",
61     "object_toc_dating_end": "string",
62     "object_toc_dating_sometime": "string",
63     "object_toc_dating_century": "list_string"
64   },
65   "object_dating": {
66     "type": "http://visit.de/ontologies/vismo/Dating",
67     "object_dating_start": "string",
68     "object_dating_end": "string",
69     "object_dating_sometime": "string",
70     "object_dating_century": "list_string"
71   },
72   "object_refentry": {
73     "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
74     "object_refentry_pages": "string",
75     "object_refentry_in_reference": "entity_reference (http://visit.de/ontologies/vismo/
76     Reference)"
77   },
78   "object_digitalrepresentation": {
79     "type": "http://visit.de/ontologies/vismo/DigitalRepresentation",
80     "object_dr_technicalmetadata": "string_long"
81   },
82 },
83 "Person": {
84   "type": "http://visit.de/ontologies/vismo/Person",
85   "person_idby_actorappel": "string",
86   "person_hastype_profession": "string",
87   "person_firstname": "string",
88   "person_lastname": "string",

```

```

89   "person_pseudonym": "string",
90   "person_alternatename": "string",
91   "person_carries_title": "string",
92   "person_comment": "string_long",
93   "person_description": "string_long",
94   "person_keyword": "string",
95   "person_iconography": "string",
96   "person_parentof_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
97   "person_ischildof_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
98   "person_ownerof_architecture": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
99   "person_motiv_arch_production": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
100  "person_carriedout_arch_prod": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
101  "person_infld_arch_production": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
102  "person_motiv_structevol": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
103  "person_carriedout_structevol": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
104  "person_infl_structevol": "entity_reference (http://visit.de/ontologies/vismo/Architecture)",
105  "person_depictedon_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
106  "person_participatedin_activity": "entity_reference (http://visit.de/ontologies/vismo/Activity)",
107  "person_receivedcustody_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
108  "person_lostcustodyof_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
109  "person_helpfullinks": "string",
110  "person_thumbnail": "image",
111  "person_birth": {
112    "type": "http://erlangen-crm.org/170309/E67\_Birth",
113    "person_mother": "entity_reference (http://visit.de/ontologies/vismo/Person)",
114    "person_father": "entity_reference (http://visit.de/ontologies/vismo/Person)",
115    "person_birthplace": "entity_reference (http://visit.de/ontologies/vismo/Place)",
116    "person_birth_dating": {
117      "person_birth_dating_exact": "datetime",
118      "person_birth_dating_start": "string",
119      "person_birth_dating_end": "string",
120      "person_birth_dating_sometime": "string"
121    }
122  },
123  "person_death": {
124    "type": "http://erlangen-crm.org/170309/E69\_Death",
125    "person_deathplace": "entity_reference (http://visit.de/ontologies/vismo/Place)",
126    "person_death_dating": {
127      "person_death_dating_exact": "datetime",
128      "person_death_dating_start": "string",
129      "person_death_dating_end": "string",
130      "person_death_dating_sometime": "string"
131    }
132  },
133  "person_marriage": {
134    "type": "http://visit.de/ontologies/vismo/Marriage",
135    "marriage_partner_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
136    "marriage_begin_dating": {
137      "marriage_begin_dating_exact": "datetime",
138      "marriage_begin_dating_start": "string",
139      "marriage_begin_dating_end": "string",
140      "marriage_begin_dating_sometime": "string"
141    },
142    "marriage_end_dating": {
143      "marriage_end_dating_exact": "datetime",
144      "marriage_end_dating_start": "string",
145      "marriage_end_dating_end": "string",
146      "marriage_end_dating_sometime": "string"
147    }
148  },
149  "person_refentry": {
150    "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
151    "person_refentry_pages": "string",
152    "person_refentry_in_reference": "entity_reference (http://visit.de/ontologies/vismo/Reference)"
153  },
154  "person_digitalrepresentation": {
155    "type": "http://visit.de/ontologies/vismo/DigitalRepresentation",
156    "person_dr_technicalmetadata": "string_long"

```

```

157     }
158   },
159   "Architecture": {
160     "type": "http://visit.de/ontologies/vismo/Architecture",
161     "architecture_idby_title": "string",
162     "arch_sacraltype": "string",
163     "arch_has_seculartype": "string",
164     "arch_bishopricaffiliation": "string",
165     "arch_geographicaffiliation": "string",
166     "arch_orderaffiliation": "string",
167     "architecture_description": "string_long",
168     "architecture_comment": "string_long",
169     "architecture_keyword": "string",
170     "architecture_icongraphy": "string",
171     "architecture_innerdescription": "string_long",
172     "architecture_outerdescription": "string_long",
173     "architecture_depictedby_object": "entity_reference (http://visit.de/ontologies/vismo/
174       Object)",
175     "architecture_buildinghistory": "string_long",
176     "arch_currentlyholds_object": "entity_reference (http://visit.de/ontologies/vismo/Object)
177       ",
178     "architecture_exemplify_function": "string",
179     "architecture_location_place": "entity_reference (http://visit.de/ontologies/vismo/Place)
180       ",
181     "arch_currentowner_person": "entity_reference (http://visit.de/ontologies/vismo/Person)",
182     "arch_currentowner_group": "entity_reference (http://visit.de/ontologies/vismo/Group)",
183     "arch_currentowner_institution": "entity_reference (http://visit.de/ontologies/vismo/
184       Institution)",
185     "architecture_contains_arch": "entity_reference (http://visit.de/ontologies/vismo/
186       Architecture)",
187     "architecture_fallswithin_arch": "entity_reference (http://visit.de/ontologies/vismo/
188       Architecture)",
189     "architecture_tookpartin_activity": "entity_reference (http://visit.de/ontologies/vismo/
190       Activity)",
191     "architecture_helpfullinks": "string",
192     "architecture_thumbnail": "image",
193     "arch_producedby_production": {
194       "type": "http://erlangen-crm.org/170309/E12_Production",
195       "production_motivatedby_person": "entity_reference (http://visit.de/ontologies/vismo/
196         Person)",
197       "production_carriedoutby_person": "entity_reference (http://visit.de/ontologies/vismo/
198         Person)",
199       "production_inflby_person": "entity_reference (http://visit.de/ontologies/vismo/Person)
200       ",
201       "arch_prod_motivatedby_group": "entity_reference (http://visit.de/ontologies/vismo/
202         Group)",
203       "arch_prod_carriedoutby_group": "entity_reference (http://visit.de/ontologies/vismo/
204         Group)",
205       "arch_prod_inflby_group": "entity_reference (http://visit.de/ontologies/vismo/Group)",
206       "arch_production_dating": {
207         "arch_prod_dating_start": "string",
208         "arch_prod_dating_end": "string",
209         "arch_production_sometime": "string",
210         "arch_prod_dating_century": "list_string"
211       }
212     },
213     "arch_modifiedby_structevolution": {
214       "type": "http://visit.de/ontologies/vismo/StructuralEvolution",
215       "structuralevolution_idby_title": "string",
216       "structuralevolution_description": "string_long",
217       "structuralevolution_comment": "string_long",
218       "structevol_exemplifies_function": "string",
219       "structevol_motivatedby_person": "entity_reference (http://visit.de/ontologies/vismo/
220         Person)",
221       "structevol_carriedoutby_person": "entity_reference (http://visit.de/ontologies/vismo/
222         Person)",
223       "structevol_influencedby_person": "entity_reference (http://visit.de/ontologies/vismo/
224         Person)",
225       "structevol_motivby_group": "entity_reference (http://visit.de/ontologies/vismo/Group),
226
227       "structevol_carriedoutby_group": "entity_reference (http://visit.de/ontologies/vismo/
228         Group)",
229       "structevol_inflby_group": "entity_reference (http://visit.de/ontologies/vismo/Group)",
230       "arch_structevol_dating": {
231         "arch_structevol_dating_start": "string",
232         "arch_structevol_dating_end": "string",
233         "arch_evol_dat_sometime": "string",
234         "arch_structevol_dating_century": "list_string"
235       }
236     }
237   }
238 }
```

```

220     "arch_refentry": {
221         "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
222         "arch_refentry_pages": "string",
223         "arch_refentry_in_reference": "entity_reference (http://visit.de/ontologies/vismo/
224             Reference)"
225     },
226     "architecture_digitalrepresentati": {
227         "type": "http://visit.de/ontologies/vismo/DigitalRepresentation",
228         "architecture_dr_techmetadata": "string_long"
229     }
230 },
231 "Place": {
232     "type": "http://visit.de/ontologies/vismo/Place",
233     "place_idby_placeappel": "string",
234     "place_description": "string_long",
235     "place_comment": "string_long",
236     "place_keyword": "string",
237     "place_iconography": "string",
238     "place_holds_architecture": "entity_reference (http://visit.de/ontologies/vismo/
239         Architecture)",
240     "place_witnessed_activity": "entity_reference (http://visit.de/ontologies/vismo/Activity)
241         ",
242     "place_wasbirthplaceof_person": "entity_reference (http://visit.de/ontologies/vismo/
243         Person)",
244     "place_wasdeathplaceof_person": "entity_reference (http://visit.de/ontologies/vismo/
245         Person)",
246     "place_isdepictedby_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",

247     "place_helpfullinks": "string",
248     "place_thumbnail": "image",
249     "place_refentry": {
250         "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
251         "place_refentry_pages": "string",
252         "place_refentry_in_reference": "entity_reference (http://visit.de/ontologies/vismo/
253             Reference)"
254     },
255     "place_digitalrepresentation": {
256         "type": "http://visit.de/ontologies/vismo/DigitalRepresentation",
257         "place_dr_techmetadata": "string"
258     }
259 },
260 "Institution": {
261     "type": "http://visit.de/ontologies/vismo/Institution",
262     "institution_idby_appel": "string",
263     "institution_ownerof_arch": "entity_reference (http://visit.de/ontologies/vismo/
264         Architecture)",
265     "institution_fallswithin_place": "entity_reference (http://visit.de/ontologies/vismo/
266         Place)",
267     "institution_address": "string",
268     "institution_owns_catalog": "entity_reference (http://visit.de/ontologies/vismo/Reference)
269         ",
270     "institution_loc_catalog": "entity_reference (http://visit.de/ontologies/vismo/Reference)
271         ",
272     "institution_helpfullinks": "string"
273 },
274 "Group": {
275     "type": "http://visit.de/ontologies/vismo/Group",
276     "group_idby_actorappel": "string",
277     "group_keyword": "string",
278     "group_iconography": "string",
279     "group_produced_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
280     "group_ownerof_architecture": "entity_reference (http://visit.de/ontologies/vismo/
281         Architecture)",
282     "group_motiv_arch_production": "entity_reference (http://visit.de/ontologies/vismo/
283         Architecture)",
284     "group_carriedout_arch_production": "entity_reference (http://visit.de/ontologies/vismo/
285         Architecture)",
286     "group_infl_arch_production": "entity_reference (http://visit.de/ontologies/vismo/
287         Architecture)",
288     "group_motiv_structevol": "entity_reference (http://visit.de/ontologies/vismo/
289         Architecture)",
290     "group_carriedout_structevol": "entity_reference (http://visit.de/ontologies/vismo/
291         Architecture)",
292     "group_infl_structevol": "entity_reference (http://visit.de/ontologies/vismo/
293         Architecture)",
294     "group_receivedcustodyof_object": "entity_reference (http://visit.de/ontologies/vismo/
295         Object)",
296     "group_lostcustodyof_object": "entity_reference (http://visit.de/ontologies/vismo/Object)
297         ",
298     "group_refentry": {
299

```

```

280     "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
281     "group_refentry_pages": "string",
282     "group_refentry_in_reference": "entity_reference (http://visit.de/ontologies/vismo/
283     Reference)"
284   },
285   "Reference": {
286     "type": "http://visit.de/ontologies/vismo/Reference",
287     "reference_keyword": "string",
288     "reference_has_type": "string",
289     "reference_publisher": "string",
290     "reference_series": "string",
291     "reference_volume": "integer",
292     "reference_pages": "integer",
293     "reference_catalog_owner": "entity_reference (http://visit.de/ontologies/vismo/
294     Institution)",
295     "reference_catalog_location": "entity_reference (http://visit.de/ontologies/vismo/
296     Institution)",
297     "reference_title": {
298       "type": "http://visit.de/ontologies/vismo/Title",
299       "reference_title_title": "string",
300       "reference_title_superordinate": "string"
301     },
302     "reference_producedby_production": {
303       "type": "http://erlangen-crm.org/170309/E12_Production",
304       "production_authormame": "string",
305       "production_year": "integer",
306       "ref_production_placeofpub": "string"
307     },
308     "reference_entry": {
309       "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
310       "reference_entry_pages": "string",
311       "reference_entry_about_activity": "entity_reference (http://visit.de/ontologies/vismo/
312       Activity)",
313       "reference_entry_about_arch": "entity_reference (http://visit.de/ontologies/vismo/
314       Architecture)",
315       "reference_entry_about_object": "entity_reference (http://visit.de/ontologies/vismo/
316       Object)",
317       "reference_entry_about_place": "entity_reference (http://visit.de/ontologies/vismo/
318       Place)",
319       "reference_entry_about_group": "entity_reference (http://visit.de/ontologies/vismo/
320       Group)",
321       "reference_entry_about_person": "entity_reference (http://visit.de/ontologies/vismo/
322       Person)"
323     },
324     "reference_catalog_dating": {
325       "type": "http://visit.de/ontologies/vismo/Dating",
326       "catalog_exhibition_start": "datetime",
327       "catalog_exhibition_end": "datetime"
328     }
329   },
330   "Activity": {
331     "type": "http://visit.de/ontologies/vismo/Activity",
332     "activity_idby_title": "string",
333     "activity_description": "string_long",
334     "activity_comment": "string_long",
335     "activity_keyword": "string",
336     "activity_iconography": "string",
337     "activity_hadparticipant_person": "entity_reference (http://visit.de/ontologies/vismo/
338     Person)",
339     "activity_used_architecture": "entity_reference (http://visit.de/ontologies/vismo/
340     Architecture)",
341     "activity_used_object": "entity_reference (http://visit.de/ontologies/vismo/Object)",
342     "activity_tookplaceat_place": "entity_reference (http://visit.de/ontologies/vismo/Place)",

343     "activity_isdepictedby_object": "entity_reference (http://visit.de/ontologies/vismo/
344     Object)",
345     "activity_helpfullinks": "string",
346     "activity_thumbnail": "image",
347     "activity_dating": {
348       "type": "http://visit.de/ontologies/vismo/Dating",
349       "activity_dating_exact": "datetime",
350       "activity_dating_end": "string",
351       "activity_dating_start": "string",
352       "activity_dating_sometime": "string"
353     },
354     "activity_refentry": {
355       "type": "http://visit.de/ontologies/vismo/ReferenceEntry",
356       "activity_refentry_pages": "string",
357     }
358   }
359 
```

```
346     "activity_refentry_in_reference": "entity_reference (http://visit.de/ontologies/vismo/  
347     Reference)"  
348 },  
349 "activity_digitalrepresentation": {  
350     "type": "http://visit.de/ontologies/vismo/DigitalRepresentation",  
351     "activity_dr_techmetadata": "string"  
352 }  
353 }
```

**Listing 50:** JSON “Schema” der ViSIT Daten, die über die REST API ausgespielt werden.

## REFERENCES

- [Bot+10] Mario Botsch u. a. *Polygon mesh processing*. AK Peters/CRC Press, 2010.
- [Cida] ISO 21127:2006 - *Information and Documentation – A Reference Ontology for the Interchange of Cultural Heritage Information*. Standard. International Organization for Standardization, Sep. 2006.
- [Cidb] ISO 21127:2014 - *Information and Documentation – A Reference Ontology for the Interchange of Cultural Heritage Information*. Standard. International Organization for Standardization, Sep. 2014.
- [Doe03] Martin Doerr. "The CIDOC Conceptual Reference Module: An Ontological Approach to Semantic Interoperability of Metadata". In: *AI Magazine* 24.3 (2003), S. 75.
- [GH97] Michael Garland und Paul S Heckbert. "Surface simplification using quadric error metrics". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, S. 209–216.
- [GH98] Michael Garland und Paul S Heckbert. "Simplifying surfaces with color and texture using quadric error metrics". In: *Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE. 1998, S. 263–269.
- [Hit+07] Pascal Hitzler u. a. *Semantic Web: Grundlagen*. Springer-Verlag, 2007.
- [MM04] Frank Manola und Eric Miller. *RDF Primer*. W3C Recommendation. W3C, Feb. 2004. URL: <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.