Cranach Digital Archive (CDA) Dokumentation

Das CDA ist ein Forschungsprojekt und dient der digitalen Erschließung der Gemälde eines der bedeutendsten Maler der deutschen Renaissance - Lucas Cranach der Ältere.

Technologische Auswahl

Infrastruktur		
Server	Ubuntu 14.04.5 LTS	https://www.ubuntu.com/
Webserver	Apache/2.4.7 (Ubuntu)	http://httpd.apache.org/
Datenbank	MySQL v5.*	https://www.mysql.com/
Repository	Git v1.9.*	https://git-scm.com/
Image Zoom Modul	IIPImage v2.*	https://github.com/ruven/iipsrv

Tabelle 1: Auswahl der technologischen Infrastruktur

Programmiersprachen		
JavaScript	ES6/7 to ES5 conversion	http://babeljs.io/
PHP	PHP 7.*	http://php.net/

Tabelle 2: Auswahl der Programmiersprachen

Modulverarbeitung		
Node.js	v7.10.*	https://nodejs.org/en/
Node Packet Manager	NPM 4.*	https://www.npmjs.com/
Webpack	v2.2	http://webpack.github.io/docs/what-is-webpack.html

Tabelle 3: Auswahl Modulverarbeitung

Frameworks und Bibliotheken		
React	v15.*	https://facebook.github.io/react/
jQuery	v3.*	https://jquery.com/
DataTables	v1.10.*	https://datatables.net/
Bootstrap	v3.3.7	http://getbootstrap.com/

IIPMooViewer	v2.*	https://github.com/ruven/iipmooviewer

Tabelle 4: Auswahl weiterer Frameworks und Libraries

Code Konventionen

JavaScript	Eslint-config-airbnb
PHP	Linter-phpcs

Tabelle 5: Auswahl Konventionen

Grundsätzliche Architektur

Das CDA ist in einem Programmierstil basierend auf dem Konzept der Objektorientierung (OOP) entwickelt. Durch die Aufteilung in Klassen bleibt das Portal flexibler, sowie besser skalierbar.

Um auch in JavaScript eine Art Objektorientierung zu gewährleisten wird mit dem Babel-Transpiler ECMAScript-6/7-Code in ECMAScript-5-Code umgewandelt.

Weiterhin wird das Prinzip der „seperation of concerns“ – sprich der klaren Trennung der Aufgaben – verfolgt. Hierfür ist die Struktur des Portals in vier Hauptdirektiven unterteilt.

- paintings
- archival-docs
- literature
- shared

Verwendete Design-Pattern

Model View Controller (MVC):
MVC ist ein Muster zur Trennung von Software in drei Komponenten - Datenmodell (engl. model), Präsentation (engl. view) und Programmsteuerung (engl. controller).

Modell (model)
Das Modell enthält die darzustellenden Daten. Diese werden aus der Datenbank abgefragt und gespeichert. Es ist von Präsentation und Steuerung unabhängig.

Präsentation (view)
Die Präsentationsschicht ist für die Darstellung der benötigten Daten aus dem Modell und die Entgegennahme von Benutzerinteraktionen zuständig. Sie kennt sowohl ihre Steuerung als auch das Modell, dessen Daten sie präsentiert, ist aber nicht für die Weiterverarbeitung der vom Benutzer übergebenen Daten zuständig.

Steuerung (controller)
Die Steuerung verwaltet eine oder mehrere Präsentationen, nimmt von ihnen Benutzeraktionen entgegen, wertet diese aus und agiert entsprechend. Hier werden auch die Variablen in der richtigen Sprache ausgewertet und der jeweiligen „View“ zugeschrieben. Zu jeder Präsentation existiert eine eigene Steuerung.

Single-Page-Anwendung (SPA)

Im Zuge der voranschreitenden Entwicklung von JavaScript und den daraus resultierenden Vorteilen wird im CDA das MVC-Pattern immer mehr abgelöst und durch eine SPA ersetzt. Als Single-Page-Webanwendung (englisch single-page application) wird eine Webanwendung bezeichnet, die aus einem einzigen HTML-Dokument besteht und deren Inhalte dynamisch nachgeladen werden.

Während im CDA das *Archivalien*- und *Publikations-Modul* noch aus einem Mix von MVC und dem dynamischen Nachladen der Inhalte mittels jQuery besteht, verwendet das *Paintings*-Modul nur noch den Teil „Modell" des MVC-Patterns um Daten zu erhalten und übernimmt die Steuerung und Präsentation auf Client-Seite selbst. Hierfür wird in JavaScript die React-Bibliothek zum Auswerten und Darstellen der Inhalte verwendet.

Module

Das Internetportal ist in drei große, mit einander verknüpfte, Kategorien unterteilt:

- Gemälde (paintings)
- Archivalien (archival-docs)
- Publikationen (literature)

Root-Directory

Im Wurzelverzeichnis sind ausschließlich Konfigurationsdateien abgelegt.

Die Webpackkonfiguration sorgt dafür, dass die JavaScript-Dateien sowie jegliche andere Ressourcen und Medien für die Benutzung im Browser transformiert und gebündelt werden. Dies ist vor allem notwendig um ECMA-Script-6 und -7 Standards in ein browserverständliches ECMA-Script-5 zu übersetzen.

Package.json ist eine Konfigurationsdatei die nicht nur wichtige Metadaten über das Projekt enthält. Neben Angaben zu Namen, Beschreibung, Autor, Version, Lizenz und Keywords definiert das File auch Abhängigkeiten der Entwicklung, die mit dem NPM-Packet-Manager installiert werden sollen. Soweit möglich werden hier alle externen JavaScript Frameworks und Libraries als „dependencies" hinterlegt.

Gemälde

Das *Paintings*-Modul ist das Herzstück des Cranach Digital Archives. Hier werden sämtliche Daten zu den einzelnen Werken, sowie dazu gehörige Bilder, ermittelt und dargestellt. Der Bereich wird mit einer Single-Page Anwendung verwirklicht (s.o.).

Die Ordnerstruktur in den Gemälden ist wie folgt aufgebaut:

- *paintings*
  - *assets*
    - *ajax*
    - *css*
    - *lang*
    - *php*
    - *js*
  - *mvc*
  - *js*
    - *grid.components*
    - *entry.components*

- *templates*

paintings
Die Gemäldedirektive beinhält Dateien die das Modul starten. Das File index.php bindet die Serverseitige Abwicklung des Model-View-Controller Konzepts ein und initiiert diese. Der Einstiegspunkt für das Schnüren des Webpack-Pakets ist die Datei paintings.main.js - hier wird der Hauptcontroller geladen, der die Steuerung des Gemäldemoduls übernimmt.

mvc
Der Ordner hält alle Dateien die für das MVC-Pattern notwendig sind.

templates
Html-Strukturen die über die Steuerung des MVC-Musters in die Präsentation geladen werden sind in dieser Direktive hinterlegt.

js
Der Pfad ist in zwei Subkategorien unterteilt. Während sich die Grid-Komponente um die Gallerieansicht kümmert, ist die Entry-Komponente für die Verarbeitung eines einzelnen Objekts (Gemälde) zuständig. Die Controller steuern die jeweilige Auswahl der Komponente.

assets
Dateien welche die Hauptsteuerung sowie das MVC unterstützen befinden sich in den Assets. Stylesheets, Suchalgorithmen und andere unterstützende Entwicklungen, die speziell nur für das Gemäldemodul sind, werden hier hinterlegt.

Datenpacket eines Objekts (Gemälde)

Die Daten eines Gemäldes werden über einen XML-Html-Request asynchron aus einer MySql-Datenbank abgegriffen. Ein Datenpacket erstreckt sich über eine Abfrage von über 20 Datenbanktabellen und wird je nach Sprache und diversen *n zu m – Relationen* ausgewertet und final an die Anfrage in einem Array zurück gegeben.

```php
<?php
array(
  'title' => array(
    'current' =>,
    'alt' => array(
      'title' =>,
      'type' =>,
      'remarks' =>
    )
  ),
  'attribution' => array(
    'current' =>,
    'alt' => array(
      'attr' =>,
      'function' =>,
      'prefix' =>,
      'suffix' =>,
      'name_type' =>,
      'other_name' =>,
      'remarks' =>,
      'date' =>
    )
  ),
  'dating' => array(
    'current' =>,
    'alt' => array(
      'dating' =>,
      'event' =>,
      'begin' =>,
      'end' =>,
      'remarks' =>
    )
  ),
```

Abbildung 1: Daten Titel, Zuschreibung und Darstellung

Die ersten drei Container im Datenpacket enthalten den Titel, die Zuschreibung und die Datierung des ausgewählten Gemäldes.

Jedes Element hat einen aktuell festgelegten Wert *current* und weitere Alternativen. Beispielsweise kann ein Objekt über die Jahre – je nach Forschungs- und Wissenstand – oft mehrere Zuschreibungen bekommen. Dies wird in den jeweiligen Arrays abgebildet.

```php
  'owner' =>,
  'repository' =>,
  'location' =>,
  'dimensions' =>,
  'support' =>,
  'signature' =>,
  'originalInscription' =>,
  'inscriptions' =>,
  'description' =>,
  'provenance' =>,
  'exhibitions' =>,
```

Abbildung 2: Daten aus MultipleTables

Als nächsten folgen eine Reihe von Daten die alle eine 1 zu 1 Beziehungen, also genau einen Wert pro Objekt, haben.

Viele dieser Elemente befinden sich in der Datentabelle *MultipleTables*.

```
'publications' => array(
  'id' =>,
  'publication' =>,
  'page' =>,
  'figure' =>,
  'catalogue' =>,
  'remarks' =>,
  'refNr' =>,
  'title' =>,
  'subtitle' =>,
  'heading' =>,
  'journal' =>,
  'series' =>,
  'volume' =>,
  'edition' =>,
  'placePubl' =>,
  'yearPubl' =>,
  'numOfPages' =>,
  'date' =>,
  'copyright' =>,
  'authors' =>,
  'publisher' =>
),
```

Abbildung 3: Daten-Container Publikationen

Die Publikationen speisen sich aus einem eigenen Literatur-Modul und sind sehr umfassend.

In der Gemäldeansicht wird ein großer Teil der Informationen abgebildet, die volle Ansicht aller Daten zu den Publikationen sind in der CDA eigenen Literaturansicht zugänglich.

```
'interpretation' => array(
  'interpretation' =>,
  'date' =>,
  'author' =>
),
'connectedWorks' => array(
  'objNr' =>,
  'frNr' =>,
  'title' =>,
  'repo' =>,
  'name' =>,
  'minithumb' =>
),
```

Abbildung 4: Daten Interpretationen und verwandte Werke

Jedes Gemälde kann mehrere Interpretationen und verwandte Werke haben.

```
81      'reports' => array(
82        'id' =>,
83        'surveyType' =>,
84        'project' =>,
85        'condition' =>,
86        'treatment' =>,
87        'treatmentDate' =>,
88        'entered' =>,
89        'author' =>,
90        'files' =>,
91        'restoration' => array(
92          'mid' =>,
93          'purpose' =>,
94          'type' =>,
95          'remarks' =>,
96          'textField' =>
97        ),
98        'operators' = array(
99          'oid' =>,
00          'role' =>,
01          'operator' =>
02        )
03      )
```

In den *reports* werden alle Untersuchungen bezüglich dem Material und der Technik, sowie der Erhaltungszustand und die Restaurierungsgeschichte festgehalten.

Abbildung 5: Daten zu den Untersuchungen am Gemälde

```
104     'relatedWorks' => array(
105       'objNr' =>,
106       'remarks' =>,
107       'thumb' =>,
108       'title' =>,
109       'repository' =>,
110     )
111   );
```

Gemälde die in Relation zu einander stehen werden in der Datentabelle *Linkage* festgehalten.

Abbildung 6: Daten von Relationen die zu einem Gemälde bestehen

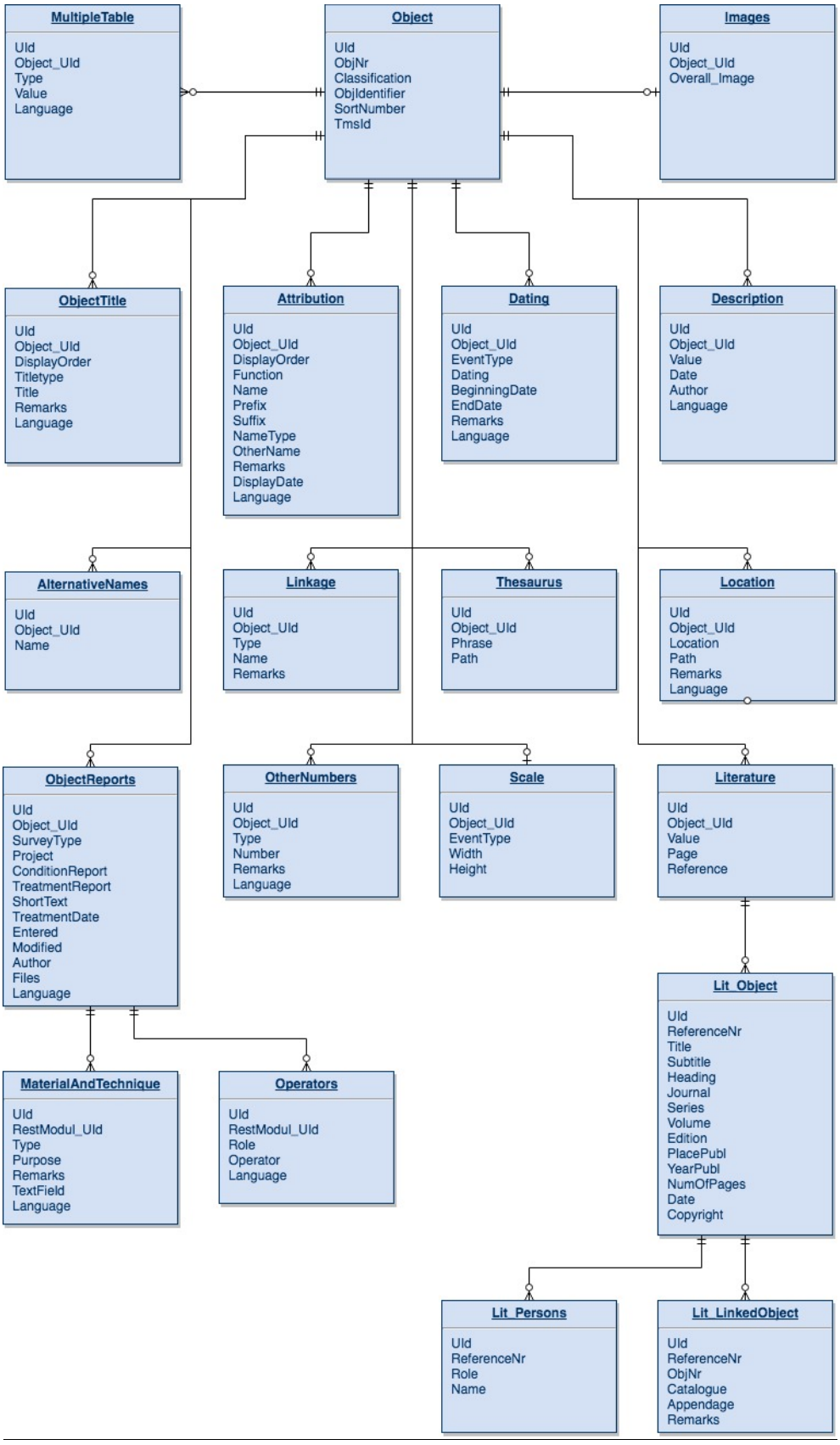
Bildunterschriften (Metadaten)

Die Informationen zu den einzelnen Bildern, sei es Gesamtansicht, Detailaufnahme und vieles mehr, befinden sich in einer gesonderten Datenbank – deswegen ist sie auch in dem Entity Relationship Model (s.u.) nicht aufgeführt. Die Daten hierfür werden je nach aktuell ausgewähltem Bild abgefragt und dargestellt.

```
58          $sql = "SELECT * FROM Metadata\n"
59            . "WHERE Name LIKE '$this->imgName'";
70
71          $ergebnis = mysql_query($sql, $this->con);
72
73          while ($row = mysql_fetch_object($ergebnis)) {
74              if ($row->Language == 'Deutsch') {
75                  $resultDE = array(
76                      "file-type-de" => $row->FileType,
77                      "image-description-de" => $row->ImageDesc,
78                      "image-created-de" => $row->ImageCreated,
79                      "image-date-de" => $row->ImageDate,
80                      "image-source-de" => $row->ImageSrc
81                  );
82              } else {
83                  $resultEN = array(
84                      "file-type-en" => $row->FileType,
85                      "image-description-en" => $row->ImageDesc,
86                      "image-created-en" => $row->ImageCreated,
87                      "image-date-en" => $row->ImageDate,
88                      "image-source-en" => $row->ImageSrc
89                  );
90              }
91          }
92
93          return array_merge($resultDE, $resultEN);
```

Abbildung 7: Abfrage Bildunterschriften

Datenbank Modell – Entity Relationship Model



SQL Phrase Index (Sphinx) Search

Sphinx ist ein Such-Server für die Volltext-Suche auf Open Source-Basis, der vor allem darauf ausgelegt ist, einen performanten und schnellen Zugriff auf Suchergebnisse zu liefern. In der Suchmaschine enthalten sind ein Indexer, ein Suchdaemon und ein Kommandozeilen-Suchtool.

Konfiguration

Sphinx wird durch die Datei */etc/sphinxsearch/sphinx.conf* konfiguriert. Als Minimum müssen eine Datenquelle und ein Index eingerichtet werden, um die Suchmaschine zu verwenden.

Konfiguration einer Quelle

Um einen Index anlegen zu können, benötigt Sphinx eine Quelle. Im Cranach Archiv wird eine MySQL-Datenbank als Datenquelle verwendet.

Abbildung 8: Auszug aus sphinx.conf - Quelle

Die Datenbank-Tabelle *SphinxIndex* ist eine *View*, auch Sicht genannt, dies ist eine logische Beziehung zwischen Datensätzen, auf die direkt zugegriffen werden kann, ohne die innere Struktur des *Views* zu kennen. Dieses Konstrukt wird auch als virtuelle Tabelle bezeichnet.

Abbildung 9: Server-Pfad und Script-File

Auf dem CDA-Server befindet sich die Source-Datei *sphinxIndex.sql* unter */home/mkpacc/SCRIPTS/sql-scripts*. Die Ausführung dieses Scripts erstellt die Ansicht der Tabelle (View).

Konfiguration eines Index

Hier wird nun der eigentliche Index konfiguriert. Der Index wird auf der Festplatte, beziehungsweise auf dem Server, abgelegt und stellt die Datenbasis der Suche dar. Indiziert werden nur die Felder, die vom Typ her Text enthalten. Alle anderen Attribute können nicht indiziert werden.

```
index cda_search
{
    source              = cda_search
    path                = /var/lib/sphinxsearch/data/cda_search
    morphology          = none

    min_word_len        = 1
    min_prefix_len      = 0
    min_infix_len       = 3
    charset_table       = 0..9, A..Z->a..z, _, a..z, U+410..U+42F->U+430..U+44F, U+430..U+44F,U+C5->U+E5, \
                          U+E5, U+C4->U+E4, U+E4, U+D6->U+F6, U+F6, U+16B, U+0c1->a, U+0c4->a, U+0c9->e, U+0cd->i, \
                          U+0d3->o, U+0d4->o, U+0da->u, U+0dd->y, U+0e1->a, U+0e4->a, U+0e9->e, U+0ed->i, U+0f3->o, \
                          U+0f4->o, U+0fa->u, U+0fd->y, U+104->U+105, U+105, U+106->U+107, U+10c->c, U+10d->c, \
                          U+10e->d, U+10f->d, U+116->U+117, U+117, U+118->U+119, U+11a->e, U+11b->e, U+12E->U+12F, \
```

Abbildung 10: Auszug aus sphinx.conf - Index

Die Indexierung der Felder wird mit folgendem Befehl durchgeführt:
sudo indexer --config /etc/sphinx/sphinx.conf --all --rotate

Sollte es in der Datenbank neue noch nicht indizierte Werte geben, muss der Indexer erneut ausgeführt werden.

Suchabfragen in PHP

Um eine Suche mit Sphinx in PHP zu starten ist es zunächst notwendig die *SphinxApi* einzubinden. Diese befindet sich in *shared > lib > php*.

Die Klasse *Sphinxsearch*:

- Startet die Suchabfrage unter Berücksichtigung der gegebenen Parameter
- Setzt nötige Filter wie z.B. den Sprachfilter
- Gruppiert die Ergebnisse nach einem gewünschten Wert
- Berechnet eine Gewichtung

In der Klasse *QueryHandler*:

- Behandelt die Suchergebnisse die aus der Abfrage geliefert werden
- Fragt anhand gelieferter Nummern (Ids) die Basisdaten des Objekts ab
- Ermittelt die jeweilige Kategorie in der die Suche einen Treffer geliefert hat