IIIF : Deliverable 4: IIIF image endpoint bij meemoo

Inhoudsopgave

Inleiding

Dit document geldt als deliverable 4 in het IIIF-project fase 2 (IIIF-2) van VKC in samenwerking met meemoo. De deliverable is het geschreven resultaat van Taak 2 in Werkpakket 2, de omschrijving van een te realiseren IIIF-compliant image endpoint voor het ontsluiten door VKC van archiefbeelden bij meemoo.

Doelstelling

Doel van Taak 2 is het ontwikkelen door meemoo van een IIIF Image API endpoint op materiaal uit het MAM-systeem. De endpoint zal worden gebruikt door VKC voor de ontsluiting van de collectie in via de VKC IIIF manifests en viewer. Op termijn zal meemoo de endpoint breder inzetten, zowel voor derden als voor eigen gebruik. Bij het bepalen van de specificaties, minimale vereisten en architectuur is hier rekening mee gehouden.

Over het VKC-2 project

De Vlaamse Kunstcollectie (VKC) werkte reeds eerder samen met meemoo aan de uitbouw van een digitaal ecosysteem. Er werd een (proef)opstelling gebouwd waarbij metadata en beelden via een zogenaamde Datahub, en IIIF-Imagehub en -Imageserver vanuit de registratiesystemen en de DAMS voor hergebruik worden beschikbaar gemaakt in de Arthub met IIIF-viewer.

Het IIIF-2 project bouwt verder op deze samenwerking door vanuit concrete use cases verbeteringen aan te brengen met het oog op twee belangrijke doelen: de maximale geautomatiseerde aggregatie van content en het maximaal geautomatiseerd ter beschikking stellen van die content voor hergebruik.

Het project voorziet in een nauwere aansluiting op de infrastructuur en het archiefsysteem van meemoo, de verkenning door VKC van IIIF-annotaties en IIIF-multi-layerviewers, en de opmaak van een metadatahandboek. Het systeem zal ook worden uitgetest op hybride collecties.

Over IIIF

IIIF of het International Image Interoperability Framework staat voor:

- een set van open standaarden die contentbeheerders toelaat hun digitale collecties en bijhorende informatie te ontsluiten
- een community: LAM's, softwarebedrijven, developers en anderen die samen de IIIF standaarden omschrijven, ontwikkelen, testen en propageren

Het volledige IIIF-raamwerk bestaat uit vier API's (application programming interfaces), waarvan de twee belangrijkste de Image API en de Presentation API zijn. De API's bieden een gestandardiseerde methode om bestanden te beschrijven en aan te leveren of uit te wisselen via het internet.

1. Image API voor het aanbieden in al dan niet bewerkte vorm (rotatie, zoom, fragment, kleuraanpassing, formaat) van bestanden en de technische metadata (info.json)
2. Presentation API voor de beschrijving in json-formaat (manifests) van de structuur en/of layout van een object, een collectie van objecten of gerelateerde objecten
3. Content Search API voor het doorzoeken van transcripties (OCR), vertalingen, annotaties van een via IIIF aangeboden bron
4. Authentication API voor het afschermen van de bronnen voor niet geauthenticeerd gebruik (bearer token)

Er zijn nog API's in ontwikkeling:

1. Change Discovery API die het machines eenvoudiger maken om IIIF-bronnen te vinden, bv. voor de ontwikkeling van (thematische) zoekmachines en portaalapplicaties
2. Content State API waarmee (een deel) van de inhoud van een IIIF Presentation API bron doorgegeven kan worden aan applicaties, ongeacht hun verschillende gebruikersinterfaces en mogelijkheden

Naast louter afbeeldingen kunnen via de IIIF API dus ook weergavegebaseerde of meer bepaald

structurele metadata uitgewisseld worden over gestructureerde objecten. Deze metadata kan bijvoorbeeld puur het object beschrijven of kan annotaties (open annotations) op en over het object bevatten.

Scope

Technische oplossing

De use case is het bouwen van een IIIF Image API 2.1 compliant endpoint voor de ontsluiting en download van hoog-kwalitatieve afbeeldingen, inclusief de nodige technische en administratieve (rechten) metadata, bewaard in het meemoo-archiefsysteem. De endpoint moet zonder meer kunnen ingeschakeld worden in het VKC-ecosysteem voor gebruik in de Imagehub.

De oplossing dient ook als proefopstelling voor meemoo met het oog op een bredere inzet van IIIF voor materiaal van meerdere aanbieders dat bij meemoo wordt bewaard en ontsloten mag of moet worden en dient dus voldoende schaalbaar te zijn om grotere en diverse collecties zonder grote drempels aan te kunnen.

Inhoudelijk aanbod

In het VKC-2 project zal een afgelijnde set bestanden aangeboden worden van:

- De aangesloten musea bij VKC, i.c. de 5 Schone Kunsten musea
- Collecties Brugse musea (in Art in Flanders is dit de algemene collectie *Musea Brugge*)

Aangezien meemoo met de opstelling op termijn mogelijk meer collecties wil ontsluiten, zowel voor VKC als voor derden, is het relevant voorbij het huidige aanbod van VKC (i.c. de Image Hub) en Art in Flanders te kijken naar zowel de maximaal potentiële omvang nu als de verwachtte aanwas. Met andere woorden is schaalbaarheid een belangrijk aandachtspunt.

Aan de hand van de metadata in het meemoo-archiefsysteem voor de silo Lukas/Art in Flanders (AIF) kan een exhaustieve en maximale lijst en omvang bepaald worden van de te ontsluiten werken. Hieruit blijkt dat voor bovenstaande aanbieders momenteel 15905 werken ontsloten worden op Art in Flanders.

Table 1. Totale omvang ontsloten en niet-ontsloten archiefbeelden in de Lukas/AIF tenant in het meemoo archiefsysteem, cijfers per aanbieder (februari 2021).

Collectie	**Aantal items in archief**	**Items online in AIF**	**wacht op metadata of mag niet online**
Musea Brugge	13176	5481	7695
KMSKA	7406	7270	136
MSK Gent	2065	1502	563
Mu.ZEE	1111	866	245
M Leuven	2177	786	1391

Collectie	Aantal items in archief	Items online in AIF	wacht op metadata of mag niet online
TOTAAL	25935	15905	10030
Bestandsgrootte	Gemiddeld		
van	200 MB per file	oudste opnames	250 MB ** gemiddelde voor hele archief-tenant
tot		600MB per file	nieuwste opnames

Bestaande implementaties

Het VKC ecosysteem (fase 1)

Verkenning

Samen met VKC verkende meemoo verschillende technische componenten voor de bouw van de IIIF-infrastructuur. Bij de selectie van de componenten werd ook rekening gehouden met de bestaande infrastructuur van VKC. In de eerste fase van het project, van 2018 tot 2019, werden onderstaande acties uitgevoerd:

- De verkenning van verschillende IIIF-beeldservercomponenten in de internationale academische en erfgoedwereld en de selectie van de meest bruikbare component in functie van de hierna genoemde compilatie;
- De compilatie van dergelijke IIIF-componenten in een proefopstelling van een beeldinfrastructuur, complementair met de arthub;
- De koppeling, in een proefopstelling, van dergelijke IIIF-beeldinfrastructuur aan de VKC-datahub;
- Een beknopte analyse van de koppeling met de onderbouw;
- De publicatie van de technische informatie van deze proefopstelling op Github en de bekendmaking ervan op relevante nationale en internationale fora;
- Tegelijk het voorbereiden van de musea en andere beeldenleveranciers op de noodzakelijke IIIF-metadata.

Selectie van componenten

In het voorjaar van 2019 zette meemoo samen met de VKC, artinflanders.be en andere beeldpartners de eerste stappen naar een vlotte, efficiënte en geautomatiseerde ontsluiting van beeldmateriaal in de tweede fase van dit project. Daarbij werd rekening gehouden met de (technische) aanbevelingen (de zogenaamde modelarchitectuur) uit de Blauwdruk gedistribueerd beeldbeheer die eerder was opgesteld.

Centraal in deze fase stond de implementatie van de IIIF-specificaties in online services. Deze open specificaties laten toe om de interacties tussen eindconsument, de toepassing waarin beelden worden gevisualiseerd, en de communicatie met de onderliggende online services te standaardiseren en te automatiseren. Verschillende technische componenten werden verkend voor

de bouw van de IIIF-infrastructuur. De keuze van de componenten moet het eenvoudig maken om de beeldinfrastructuur ook in de toekomst te beheren.

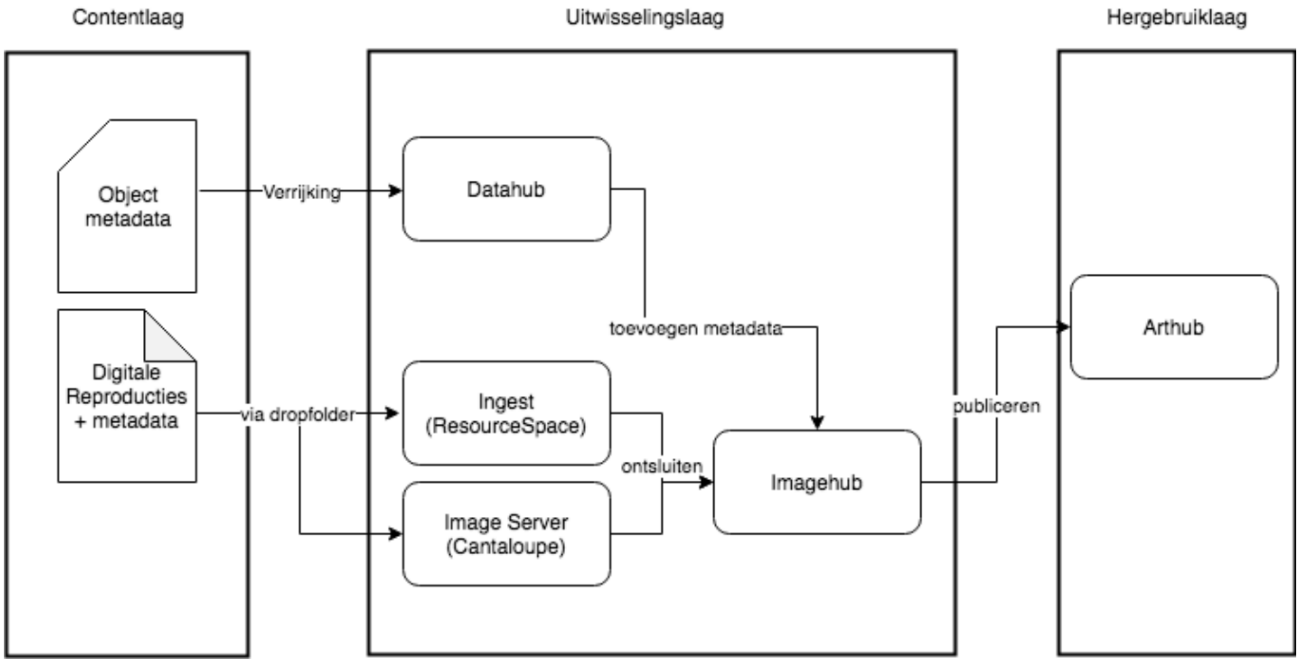

Figure 1. Schema van de gekozen componenten voor de IIIF-proefopstelling

- Als ingestcomponent is gekozen voor ResourceSpace, een software die al door een aantal partners van VKC gebruikt wordt voor de ontsluiting van hun beelden. ResourceSpace is een DAM-systeem (beeldbeheersysteem) waarmee de beelden opgenomen kunnen worden in de proefopstellingen en waarin vervolgens de metadata uit de Datahub gekoppeld kunnen worden aan de beelden.
- Als IIIF-beeldenserver is gekozen voor Cantaloupe, omwille van de mogelijkheid om de metadata die ingebed wordt in de beelden te behouden in de kopieën van de afgeleiden. Via Cantaloupe zijn de beelden uitwisselbaar via de IIIF Image API 2.
- Er werd ook een Imagehub ontwikkeld om de beelden en hun metadata via de IIIF Presentation API te ontsluiten (via IIIF-manifests). Deze webapplicatie is gemaakt in hetzelfde framework waarmee de Datahub ontwikkeld werd (PHP/Symfony).
- De IIIF-manifests zelf worden getoond in de Universal Viewer. Dit is een beeldenviewer die IIIF-manifests kan weergeven.

Samen vormen deze componenten de IIIF-proefopstelling. Testen gebeurde met tweehonderd beelden uit de collecties van de partnermusea van de VKC. De aangepaste versie van de Arthub ontsluit nu beelden via de Universal Viewer door een koppeling met de Imagehub.

Verdere details over de opzet en architectuur van het VKC-ecosysteem zijn te vinden in Deliverable 1, Architectuur van de proefopstelling, van het VKC-1 project.

Nieuws van de Groote Oorlog (meemoo)

Voor de website Nieuws van de Groote Oorlog (NVDGO) heeft meemoo (destijds VIAA) een IIPImage server, of iipsrv, opgezet voor de ontsluiting van multipage kranten en andere documenten via een OpenLayers3 canvas-viewer op de site. Als protocol wordt in de viewer Zoomify gebruikt, maar ook

IIP en IIIF zijn ondersteund.

> Voorbeeld detailpagina NVDGO: L'indépendance belge | Nieuws van de Groote Oorlog (hetarchief.be)

Voor meer informatie en documentatie over de IIIF-implementatie voor NVDGO zie:

- IIPImage en IIIF - Het Archief (Publiek) - meemoo documentation (atlassian.net)
- Handleiding IIIF-beelden gebruiken - Interactie - meemoo documentation (atlassian.net)

Deze image server is oorspronkelijk opgezet als dedicated endpoint voor de NVDGO website. Om ook uitwisseling met derden mogelijk en eenvoudiger te maken is op basis van een initiële vraag van de "Boekentoren" (UGent) een aantal aanpassingen doorgevoerd:

- De URL werd herschreven zonder querystring (?IIIF=...) naar meemoo.be)
- De CORS-instellingen werden gewijzigd zodat cross-domain requests toegestaan zijn;
- Het lokale media path van het document werd vertaald naar de gemeenschappelijk bekende identifier, in dit geval de external_id bij meemoo (pid).

Hiervoor werd een Python script geschreven dat via nginx-specifieke X-Accel-Redirect headers de request doorstuurt naar de relevante URL op http://images.hetarchief.be.

Resultaat is een IIIF Image API 2.0 met level 1 compliancy voor een 300.000 documenten.

Example 1. Image information request: NvdGO

https://images.hetarchief.be/iipsrv/?IIIF=/media/5/S/S2aTZOTWggiebVVpZpSTJPWd/wp9t14vr7s_19140404_0001/info.json


```
{
  "@context": "http://iiif.io/api/image/2/context.json",
  "@id":
"http://images.hetarchief.be/iipsrv/?IIIF=/media/5/S/S2aTZOTWggiebVVpZpSTJPWd/wp9t
14vr7s_19140404_0001",
  "protocol": "http://iiif.io/api/image",
  "width": 4633,
  "height": 5959,
  "sizes": [{
    "width": 144,
    "height": 186
  }, {
    "width": 289,
    "height": 372
  }, {
    "width": 579,
    "height": 744
  }, {
    "width": 1158,
    "height": 1489
  }, {
    "width": 2316,
    "height": 2979
  }],
  "tiles": [{
    "width": 256,
    "height": 256,
    "scaleFactors": [1, 2, 4, 8, 16, 32]
  }],
  "profile": ["http://iiif.io/api/image/2/level1.json", {
    "formats": ["jpg"],
    "qualities": ["native", "color", "gray", "bitonal"],
    "supports": ["regionByPct", "regionSquare", "sizeByForcedWh", "sizeByWh",
"sizeAboveFull", "rotationBy90s", "mirroring"],
    "maxWidth": 5000,
    "maxHeight": 5000
  }]
}
```

Example 2. Image information request: meemoo IIIF

https://iiif.meemoo.be/wp9t14vr7s_19140404_0001/info.json


```
{
  "@context": "http://iiif.io/api/image/2/context.json",
  "@id": "https://iiif.meemoo.be/wp9t14vr7s_19140404_0001",
  "protocol": "http://iiif.io/api/image",
  "width": 4633,
  "height": 5959,
  "sizes": [{
    "width": 144,
    "height": 186
  }, {
    "width": 289,
    "height": 372
  }, {
    "width": 579,
    "height": 744
  }, {
    "width": 1158,
    "height": 1489
  }, {
    "width": 2316,
    "height": 2979
  }],
  "tiles": [{
    "width": 256,
    "height": 256,
    "scaleFactors": [1, 2, 4, 8, 16, 32]
  }],
  "profile": ["http://iiif.io/api/image/2/level1.json", {
    "formats": ["jpg"],
    "qualities": ["native", "color", "gray", "bitonal"],
    "supports": ["regionByPct", "regionSquare", "sizeByForcedWh", "sizeByWh",
"sizeAboveFull", "rotationBy90s", "mirroring"],
    "maxWidth": 5000,
    "maxHeight": 5000
  }],
  "rights": "https://nieuwsvandegrooteoorlog.hetarchief.be/nl/gebruiksvoorwaarden"
①
}
```


① Uit het voorbeeld blijkt dat image langs twee routes via IIIF kan opgevraagd worden. Hoewel beide requests afgehandeld worden door dezelfde image server wijkt de info.json via http://iif.meemoo.be af. Zie meemoo iiif-mapping op Github.

Example 3. Image request

https://iiif.meemoo.be/wp9t14vr7s_19140404_0001/full/full/0/default.jpg

Specificaties

Functioneel

Image server

- We bieden minstens de Image API 2.1 aan, we voorzien op korte termijn de mogelijkheid om versie 3.0 te ondersteunen
- We streven een level 2 compliance na, echter indien level 1 voldoende functionaliteit biedt voor VKC kan hiermee worden volstaan (https://iiif.io/api/image/3.0/compliance)
- De bestanden worden door de image server aangeboden als jpeg en optioneel indien mogelijk in png-formaat
- Afgeleiden kunnen optioneel in hun geheel gedownload worden, bij voorkeur als TIFF
- De archiefbestanden (on_tape) worden niet als download beschikbaar gemaakt, maar kunnen aangevraagd worden bij meemoo via de bestaande kanalen
- Er wordt geen presentation API aangeboden in het kader van dit project
- Er wordt in het kader van dit project geen authenticatie voorzien

Table 2. Features MVP en roadmap

_	**VKC-2 mvp**	**Roadmap meemoo**
Image API	2.1	3.0
Level compliancy	1	2
Default output	jpg	jpg, png
Download full image	ja, als jpg	ja, als afgeleide
Download archival image	nee	nee
Presentation API	-	3.0
Authentication API	-	TBD

Beeldbestanden

Afgeleiden

Het gebruiken van archiefmasters is om verschillende redenen praktisch niet haalbaar. De bestanden zijn zeer groot wat de load op de image server en bandbreedte impacteert en het raadplegen onnodig vertraagd, de kwaliteit is ook hoger dan nodig bij het eenvoudig raadplegen op

een gewoon beeldscherm. Bovendien bevatten de beelden doorgaans randen en kleurkaarten die enkel relevant zijn in een archivistische context of voor hergebruik. Standaard worden daarom afgeleide bestanden voor ontsluiting en raadpleging gemaakt.

Bij het omzetten van de archiefbestanden naar afgeleide bestanden worden de volgende vuistregels gehanteerd:

- De resolutie (ppi) is maximaal 300 ppi voor werken in het publieke domein, afhankelijk van de originele waarde in het bronbestand. Voor werken die auteursrechtelijk beschermd zijn, worden de op het moment van creatie van het afgeleide beeld geldende hergebruiksvoorwaarden gerespecteerd.
- De afbeeldingen worden herschaald tot een baseline resolutie (grootte) van 5000 pixels voor de kortste zijde, de maximale boven- en ondergrens is daarbij de originele resolutie, bestanden worden niet "opgeblazen" indien ze kleiner zijn, maar kunnen bij heel grote bestanden wel verkleind worden.
- De beelden worden bijgesneden zodat randen, kleurkaarten kaders niet meer worden getoond.
- Bestanden worden aangeboden op en voor het web in sRGB, tegenover AdobeRGB voor de originelen.
- Embedded metadata(XMP, Exif, ICC) worden in zoverre in de bron beschikbaar en mogelijk, in de afbeeldingen opgeslaan.

De te ontsluiten collectie is zowel qua type als qua fysieke afmeting zeer divers, voor de hele VKC-collectie spreken we over heel kleine objecten van een paar centimeter tot wandtapijten, of een wandkaart van ettelijke meters hoog en breed. De afmeting in pixels van het digitale beeld is niet per se in verhouding en is afhankelijk van de leeftijd, de fotograaf, het opnametoestel, etc. Voor sommige grote werken zijn verschillende foto’s aan elkaar geplakt (stitching). Andere zijn gevat in 1 foto. Relatief gesproken kan een groter fysiek werk in aantal pixels kleiner of gelijk zijn aan een fysiek kleiner werk.

Om te vermijden dat (heel) grote werken te fel herschaald worden ten opzichte van kleine werken met een in verhouding hoger aantal pixels, zal binnen dit project bekeken worden in welke mate een relatie bestaat tussen de fysieke en digitale afmetingen op basis waarvan herschaling dynamisch kan gebeuren. We onderzoeken dit aan de hand vaan een aantal typewerken zoals *Pierrot et Squelette en Jaune* van James Ensor.

CAUTION

Bij het herschalen moet onderscheid worden gemaakt tussen werken in public domain en werken die nog onder het auteursrecht vallen. In de overeenkomst die de VKC momenteel met de auteursrechtenorganisatie Sabam heeft, worden volgende hergebruiksvoorwaarden vooropgesteld voor werken die onder het auteursrecht vallen: de resolutie van het gereproduceerde beeld mag niet meer dan 640x480 pixels zijn en een resolutie van maximum 72dpi hebben.

Voor de creatie van de afgeleiden starten we met een manuele workflow die eenvoudig kan bijgesteld worden om uiteindelijk te komen tot een automatiseerbare workflow. Om zowel de

workflow voor de creatie van afgeleide beelden als de specificaties an sich te testen beperken we ons in eerste instantie tot de omzetting van de originele testset aan beelden die nu reeds beschikbaar zijn in de IIIF-viewer in de VKC Arthub. Hierbij zal worden onderzocht welke een haalbare workflow is voor de aanmaak van de afgeleide beeldbestanden en in welke mate dit proces geautomatiseerd kan worden. Indien nodig kunnen bovenstaande specificaties dan ook bijgewerkt worden op basis van voortschrijdend inzicht.

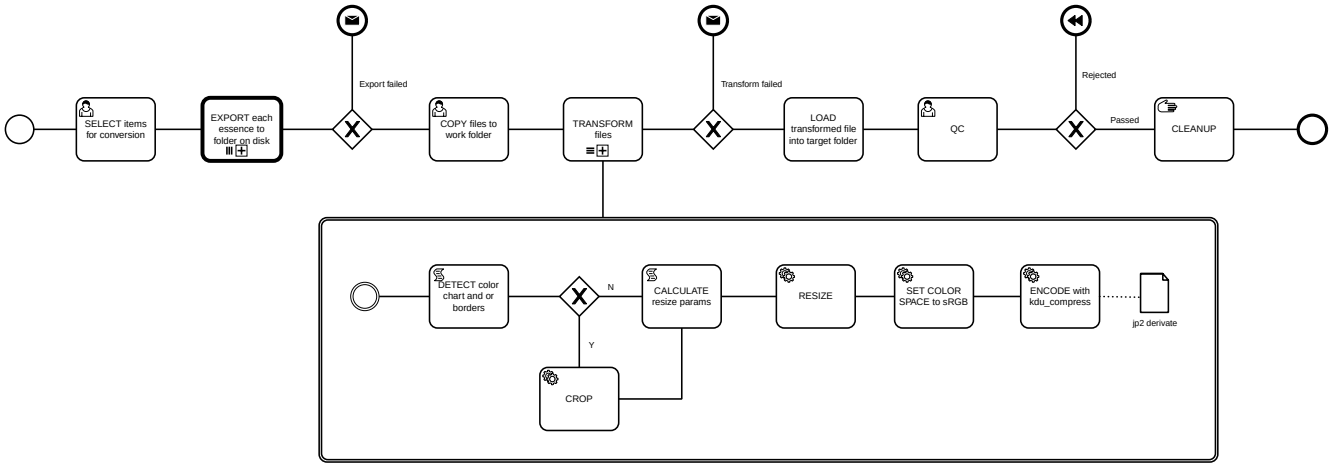

Figure 2. Voorbeeld manuele workflow voor creatie van jp2 afgeleide beeldbestanden

Bestandsformaat

De twee meest gebruikte bestandsformaten voor het aanbieden van kwalitatieve hoogresolutiebestanden via een image server zijn piramidale tiff's (pTIFF) en JPEG2000 (jp2) bestanden. Beide bestandsformaten zijn uitermate geschikt voor grote, hoogkwalitatieve bestanden met verschillende resoluties en grote kleurdieptes, kunnen metadata bevatten, bieden ruime compressiemogelijkheden (lossless en lossy) en ondersteuning voor meerdere colorspaces. De formaten worden als volgt omschreven:

> **JPEG 2000** is an image coding system that uses state-of-the-art compression techniques based on wavelet technology and offers an extremely high level of scalability and accessibility. Content can be coded once at any quality, up to lossless, but accessed and decoded at a potentially very large number of other qualities and resolutions and/or by region of interest, with no significant penalty in coding efficiency. — jpeg.org

> **TIFF** is a tag-based file format for storing and interchanging raster images. It serves as a wrapper for different bitstream encodings for bit-mapped (raster) images. The different encodings may represent different compression schemes and different schemes for color representation (photometric interpretation). — Library of Congress

De keuze voor een bestandsformaat voor ontsluiting hangt af van verschillende factoren:

- **Encodering**: bronbestanden converteren naar jp2 is doorgaans eenvoudiger (1 stap) en sneller (resources), mits gebruik van een goede encoder zoals kdu_compress (Kakadu). De creatie van

piramidale tiffs is door de band trager. Voor een eenmalige conversie van een in aantal beperkte collectie is de impact beperkt. Wanneer regelmatig grotere batches moeten worden omgezet is de keuze voor jp2 evidenter.

- **Decodering**: jp2-bestanden moeten in tegenstelling tot tiffs on-the-fly gedecodeerd worden door de image server. Dit zorgt voor een kleine overhead bij het aanbieden van jp2-bestanden die echter beperkt kan worden gehouden door de performante Kakadu library te gebruiken. Anderzijds biedt jp2 de mogelijkheid tot progressive transmission waarbij eerst een lagere resolutie wordt getoond tot de hogere resoluties ingeladen zijn.
- **Compressie en bestandsgrootte**: afhankelijk van de complexiteit en resolutie van een beeldbestand enerzijds en anderzijds de gekozen compressie-methode kunnen zowel ptiffs als jp2's kleiner zijn dan de andere. Zie voorbeeld Getty. Wanneer kwaliteit(sverlies) een belangrijk aandachtpunt is dan biedt jp2 het voordeel van een lossless compressie met een visueel gelijkwaardige kwaliteit van een uncompressed TIFF voor eeen bestand van kleinere omvang. Het verschil in grootte zonder merkbaar kwaliteitsverlies van jp2 tegenover tiff is vooral uitgesproken bij compressie van grotere bestanden met hogere resoluties. Voor een vergelijking van bestandsgroottes na omzetting zie Appendix A.
- **Resolutie**: jp2 wordt opgeslagen in de hoogste resolutie maar kan door de gebruikte wavelettechnologie op verschillende (lagere) resoluties gedecodeerd worden. Tiff moet worden opgeslagen met de gewenste resolutielagen, de weergave is beperkt tot deze resoluties.

> Gelet op bovenstaande vergelijking, in combinatie met de expertise bij meemoo met de omzetting en het gebruik van jpeg2000, is gekozen om de afgeleide bestanden in een gecomprimeerd jp2-formaat op te slaan voor gebruik in de image server.

Identifiers

De URI van een typische *image request* is https://example.com/iiif/`identifier` Aan de hand van de identifier in de uri weet of kan de image server achterhalen (*resolve*) welk beeld in de *response* teruggeven moet worden. Het formaat van de identifier kan vrij gekozen worden (ARK, URN, filename, …) zolang de link met en naar een beeld achterhaalbaar, uniek en persistent is.

Doel is om op een eenduidige manier aan de hand van een gekende pid een beeld te kunnen opvragen via de image API van meemoo zonder dat hiervoor nog de tussenstap van een zogenaamde resolver nodig is. Vaak wordt de identifier gebruikt in de bestandsnaam van het beeld dat wordt opgevraagd.

In het VKC-project zijn verschillende identifiers gekend voor dezelfde werken, afhankelijk van het systeem waar de collectie en de metadata van de beelden beheerd worden, van historische inventarisnummers tot persistente digitaal-archiefnummers.

In het archiefsysteem van meemoo is elk werk een intellectuele entiteit (IE). Een IE heeft een unieke *persistent identifier* (pid), die onveranderlijk is doorheen de hele keten van registratie, digitalisering, archivering tot ontsluiting, we gebruiken hiervoor noid in eigen formaat, vb. `wp9t14vr7s`. Catalogusnummers en overige lokale identifiers van de collectiebeheerders worden, indien gekend en gewenst, bewaard in de metadata van de IE. Ze worden echter beschouwd als niet-autoratieve metadata.

CAUTION	In het geval van enkelvoudige objecten is de relatie tussen de IE en het gearchiveerde media-object 1-op-1. De pid kan herleid worden naar 1 mediabestand. Meervoudige objecten hebben echter meerdere objecten die samen een IE vormen. Zoals kranten waarbij elke pagina een afzonderlijk bestand is. Elk media-object dat deel uitmaakt van eenzelfde IE heeft daarom naast de pid een unieke external_id die bestaat uit de pid van het IE met een suffix gescheiden door een *underscore*, bijvoorbeeld: `wp9t14vr7s_19140404_0001`, pagina 1 van krant `wp9t14vr7s`. Voor enkelvoudige objecten is de external_id gelijk aan de pid.

De external_id uit het meemoo-archiefsysteem biedt een hoge mate van zekerheid op vlak van uniciteit en persistentie met voldoende granulariteit, namelijk tot op digitaal object. Om zonder omweg de identifier te kunnen gebruiken in de image endpoint zal deze toegepast worden in de naamgeving van de afgeleide beeldbestanden:{identifier}.{ext}, bijvoorbeeld: `wp9t14vr7s_19140404_0001.jp2`.

De meemoo-identifiers moeten uiteraard bekend zijn in de datahub van VKC, zonder de nood aan bijkomende resolvers en mappings. Met andere woorden zal VKC de meemoo-identifier in de metadata van bijvoorbeeld kmska:494 moeten bijhouden om de image en information requests te kunnen doen. In het kader van dit project levert meemoo hiervoor de nodige data met de metadata-export waarbij wordt onderzocht of en hoe die structureel kan worden uitgewisseld via de API's.

Vertaald naar een image information request:

https://iiif.meemoo.be/wp9t14vr7s_19140404_0001/info.json

info.json

Via de info.json biedt de image server informatie over het beeld, voornamelijk technische gegevens maar ook informatie over rechten of services kunnen hierin worden opgenomen.

De structuur van een request is: {scheme}://{server}\{/prefix}/{identifier}/info.json, de response is een JSON-LD. Om vlot te werken moet CORS (Cross Origin Resource Sharing) toegestaan zijn en moeten de correcte Accept of Content-type headers gebruikt worden in respectievelijk de request of de response.

In onderstaande tabel wordt de gewenste opbouw van de info.json weergegeven. Dit zou moeten volstaan binnen de scope van dit project en verder. Let wel, https://iiif.io/api/image/3.0/#5-image-information

Table 3. info.json properties en hun omschrijving: overgenomen van iiif.io

Property	Compliance	Omschrijving
`@context`	Required	The context document that describes the semantics of the terms used in the document. This must be the URI: http://iiif.io/api/image/2/context.jsonfor version 2.1 of the IIIF Image API. This document allows the response to be interpreted as RDF, using the JSON-LDserialization.
`@id`	Required	The base URI of the image as defined in URI Syntax, including scheme, server, prefix and identifier without a trailing slash.
`protocol`	Required	The URI http://iiif.io/api/imagewhich can be used to determine that the document describes an image service which is a version of the IIIF Image API.
`width`	Required	The width in pixels of the full image content, given as an integer.
`height`	Required	The height in pixels of the full image content, given as an integer.
`sizes`	Optional	A set of height and width pairs the client should use in the sizeparameter to request complete images at different sizes that the server has available. This may be used to let a client know the sizes that are available when the server does not support requests for arbitrary sizes, or simply as a hint that requesting an image of this size may result in a faster response. A request constructed with the w,hsyntax using these sizes *must* be supported by the server, even if arbitrary width and height are not.

Property	Compliance	Omschrijving
`tiles`	Optional	A set of descriptions of the parameters to use to request regions of the image (tiles) that are efficient for the server to deliver. Each description gives a width, optionally a height for non-square tiles, and a set of scale factors at which tiles of those dimensions are available.
`profile`	Required	A list of profiles, indicated by either a URI or an object describing the features supported. The first entry in the list *must* be a https://iiif.io/api/image/2.1/#profile-descriptionvoor de properties.
`attribution`	Optional	Text that *must* be shown when content obtained from the Image API service is displayed or used. It might include copyright or ownership statements, or a simple acknowledgement of the providing institution. The value *may* contain simple HTML as described in the HTML Markup in Property Valuessection of the Presentation API.
`license`	Optional	A link to an external resource that describes the license or rights statement under which content obtained from the Image API service may be used.
`logo`	Optional	A small image that represents an individual or organization associated with the content. Logo images *must* be clearly rendered when content obtained from the Image API service is displayed or used. Clients *must not* crop, rotate, or otherwise distort the image.

Voorbeeld info.json

```
{
  "@context": "http://iiif.io/api/image/2/context.json",
  "@id": "http://www.example.org/image-service/abcd1234/1E34750D-38DB-4825-A38A-
B60A345E591C",
  "protocol": "http://iiif.io/api/image",
  "width": 6000,
  "height": 4000,
  "sizes": [{
    "width": 150,
    "height": 100
  }, {
    "width": 600,
    "height": 400
  }, {
    "width": 3000,
    "height": 2000
  }],
  "tiles": [{
    "width": 512,
    "scaleFactors": [1, 2, 4, 8, 16]
  }],
  "profile": ["http://iiif.io/api/image/2/level2.json"],
  "attribution": "Provided by Example Organization",
  "logo": "http://example.org/images/logo.png",
  "license": "http://rightsstatements.org/vocab/InC-EDU/1.0/"
}
```

Authenticatie en authorisatie

Authenticatie en authorisatie zijn niet binnen scope van dit project. meemoo behoudt weliswaar de mogelijkheid om in een later stadium authenticatie te voorzien al dan niet op basis van de IIIF Authentication API in combinatie met de eigen accesstoken-authenticatie services. In dat geval zal de afnemer (i.c. VKC) voldoende documentatie krijgen om de authenticatie te integreren. Hiervoor dient eerst een grondige analyse te worden uitgevoerd.

Technisch

Standaardisatie

meemoo standardiseert in de mate van het mogelijke haar services op het vlak van programmeertalen, software, RTE, tools voor deployment, gebruik van (open) standaarden, etc. Om tot een onderhoudbare oplossing te komen wordt hierbij best zo nauw mogelijk aangesloten.

Ter informatie sommen we de typische toepassingen en software bij meemoo op:

- Programmeertaal: Python (in mindere mate Ruby)

- Database: PostgreSQL
- Webserver en proxy: nginx+
- Authenticatie en authorisatie: eigen IdP met authenticatie op basis van SAML2.0 en OAuth met LDAP als directory
- Runtime environment: containers op Openshift (Cloud) of VMWare (on premise)
- Deployment: CI/CD met Jenkins (pipelines), Puppet met Foreman (indien VM)
- Versioning: Git (Github)
- Logging: stdout naar Elasticsearch/Kibana
- Queue: RabbitMQ
- Reporting: Data Warehouse + Tableau of PowerBI
- Storage: Object Store (S3) of persistent local storage (VM)

Daarnaast streeft meemoo naar het maximaal inpassen van nieuwe ontwikkelingen in haar eigen eco-systeem. Voor de platformen gericht op ontsluiting en interactie is een architectuur uitgewerkt die herhaalbaar en breed inzetbaar is: de zogenaamde Shared Components. Een voorbeeld van een applicatie die volledig met de componenten is gebouwd is Het Archief voor Onderwijs.

Het systeem omvat een volledige en modulaire middleware op vlak van ETL, opslag, zoekindex, metadata-API, authenticatie en media delivery, waaruit op basis van de nood componenten kunnen worden ingezet voor de bouw van een toepassing:

- Syncrator: taakgebaseerde ETL-service die de synchronisatie van metadata uit het MAM (de bron) naar de lokale datastore (het doel) verzorgt
- (Meta)datastore: een postgresql database die onder meer de metadata van de te ontsluiten archiefitems bevat en ook applicatiespecifieke tabellen kan bevatten
- Indexer: een ETL service die de synchronisatie tussen database en index regelt op basis van database triggers en webhooks
- ElasticSearch als standaard index die de zoekfunctionaliteit in de platformen mogelijk maakt
- Hasura GraphQL voor naadloos gebruik bovenop een postgresql database. Dit vormt de basis-API waarmee de frontendapplicaties met de datastore kunnen connecteren.
- IDM: een volledige Identity en Access Management stack:
  - een LDAP directory
  - een SAML identity provider
  - een self-service user management applicatie
  - een account manager voor beheerders
- Event logger
- Media services zoals play ticketing en streaming

meemoo-infrastructuur

Hosting en deployment

De image server kan gehost worden bij meemoo op 2 manieren

- Virtualisatie in een VM in het eigen datacenter in Oostkamp (DCO)
- Virtualisatie in containers op Openshift in IBM Cloud

Gelet op onderstaande punten wordt gekozen voor de eerste oplossing:

- Netwerkverkeer en bandbreedte zijn goedkoper in DCO dan in de Cloud
- Storage is eveneens goedkoper en makkelijk uitbreidbaar in DCO (zie *Opslag*)
- Het netwerkverkeer verloopt via onze eigen nginx proxy met flexibiliteit onder meer op vlak van URL rewrites en image cache

Voor de deployment en management van de software rekenen we op de combinatie Puppet en Foreman. De codebase wordt bewaard en geversioneerd in een Git repository van meemoo.

Opslag

File storage voldoet bij voorkeur aan de volgende vereisten:

- low-latency
- high-availability
- low-cost
- scaleable

De twee door meemoo gebruikte oplossingen zijn een eigen Swarm Object Store met S3-connector waarop onder meer de on_disk archiefbestanden en browse copies van het MAM worden bewaard of local disk storage in het datacenter. Beide oplossingen voldoen aan bovenstaande vereisten. Echter, image servers hebben bij voorkeur de bestanden dichtbij en hoewel er S3-oplossingen voor verschillende image servers bestaan zal dit toch trager zijn dan lokale I/O door de overhead van het netwerkverkeer (GET). Aangezien de image server toch in het datacenter wordt gehost en de afgeleide bestanden louter en alleen voor de image server bestaan is het logischer deze te hosten in ons eigen datacenter.

Om een inschatting te kunnen maken van de nodige opslag is op een aantal testbestanden een conversie uitgevoerd. Voor de creatie van de jp2 bestanden is kdu_compress van Kakadu gebruikt volgens een profiel van het Bodlean.

```
$ kdu_compress -i input.tif -o output.jp2
  Clevels=6 Clayers=6 \
  "Cprecincts={256,256},{256,256},{128,128}" \
  "Stiles={512,512}" \
  Corder=RPCL \
  ORGgen_plt=yes \
  ORGtparts=R \
  "Cblk={64,64}" \
  -jp2_space "sRGB" \
  Cuse_sop=yes \
  Cuse_eph=yes \
  -flush_period 1024 \
  Creversible=yes|no \
  -rate -|3
```

- Creversible: of de compressie omkeerbaar is (lossless) of niet (lossy)
- rate: de mate van compressie. een integer of enkel dash (geen)

Hieruit blijkt een reductie van de bestandsgrootte van:

- full, lossless ⇒ -43%
- bijgeknipt en resized, lossless ⇒ -84%
- bijgeknipt en geresized, lossy ⇒ -98%

Voor het berekenen van de onder- en bovengrens voor benodigde schijfruimte vertrekken we van de totale omvang van de VKC tenant in het meemoo MAM: 32502 tiffs van gemiddeld 250 MB per tiffs = 8.125TB

Table 4. Schatting minimaal benodigde opslagruimte, afhankelijk van formaat

formaat	**reductie**	**nodige schijfruimte**
lossless jp2	(8.125/100)*16	1.3 TB
lossy jp2	(8.125/100)*2	162.5 GB

Ter vergelijking, de ca. 300.000 pagina's op NVDGO zijn samen 2.4TB, gemiddeld 8MB per afbeelding.

Monitoring en logging

Applicaties bij meemoo worden gemonitored. Hieronder verstaan we enerzijds het monitoren van de toestand van de applicatie en anderzijds wat er in die toestand gebeurt:

- monitoring: draait mijn applicatie, is ze gezond, zijn er issues
- logging: wat doet mijn applicatie, welke acties worden uitgevoerd

De Image endpoint is geen uitzondering. In praktijk houdt dit in dat we de gebruikte resources, de health en uptime en eventuele errors in de gaten houden via PRTG (sensors).

Typische voorbeelden van sensoren zijn:

- CPU, RAM en Disk usage
- Connectiviteit (intern en extern)
- HTTP health & liveness checks
- Certificaatgeldigheid (indien van toepassing)

De developer guidelines van meemoo worden hierbij gehanteerd. Deze kunnen opgevraagd worden of geraadpleegd op: logging guidelines voor developers.

De toepassing logt via stdout naar een ELK-stack (Elasticsearch-Logstash-Kibana). Kort samengevat moeten de log messages voldoen aan de volgende voorwaarden:

- Single-line JSON*
- Standaard gestructureerd formaat en labels
- Logs worden weggeschreven naar stdout/err
- De x-correlation-id van de request wordt gebruikt als trace of span-id
- Healthz en andere monitoring checks worden niet gelogd

Optioneel biedt meemoo de mogelijkheid voor het loggen van user events, de zogenaamde event-logger. Aangezien de endpoint voornamelijk bevraagd zal worden door andere services (machines) wordt deze vorm van loggen gezien als buiten scope van dit project en louter vermeld ter informatie.

Performantie

De proefopstelling zal dienen om de baseline performantie te meten en eventuele verbeteringen door te voeren of een voorstel te doen, vb. voor een redundante HA setup, proxy cache, etc. Dit kan op basis van de gegevens uit monitoring en debug logging, maar we voorzien ook de optie van geautomatiseerde load en stress testing. De aandacht richt zich hierbij op:

- Throughput (bandbreedte)
- nRequests/sec
- Response Time
- Up Time

Documentatie

Alle applicaties bij meemoo worden beschreven en gedocumenteerd op de Confluence wiki van meemoo in de *Software Factory*. Daarnaast bevat elke code repository een README. Handleidingen en technische documentatie worden afhankelijk van de nodige toegang bewaard op de wiki of indien nuttig ontsloten via developer.meemoo.be of het partnerportaal van meemoo.

Worden minstens gedocumenteerd:

- technische details omgeving (uri's, auth, software, dependencies, repo, etc.)

- verantwoordelijke en medewerkers
- installatieprocedure
- links naar externe documentatie
- API calls en responses

Architectuur

Diagram (eenvoudig)

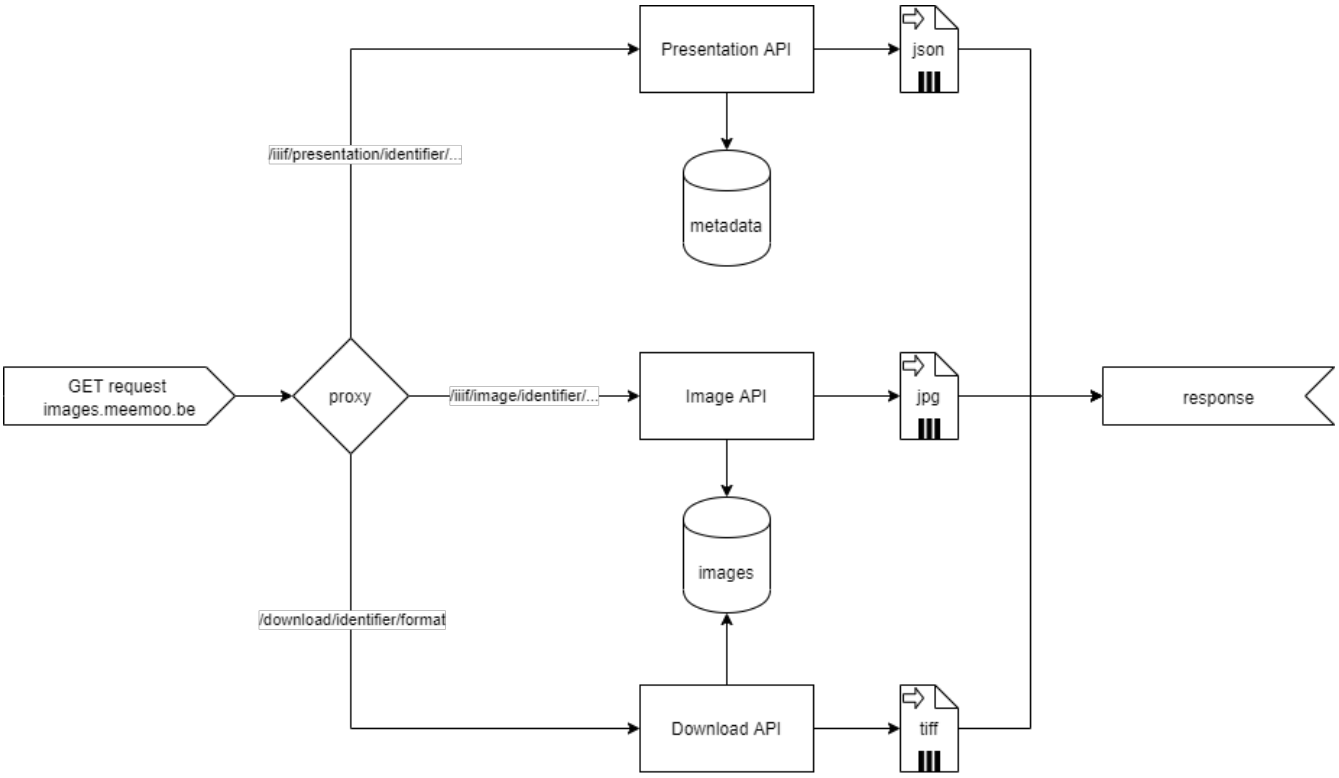

Figure 3. Eenvoudige voorstelling van IIIF-architectuur (to be) bij meemoo

Componenten

Er zijn drie specifieke softwarecomponenten, maar alleen 1 is in scope van dit project:

1. Image API: de IIIF compliant image server die beelden levert aan een image viewer
2. *Download API: een custom applicatie die een hoogresolutiebestand levert in een bepaald formaat*
3. *Presentation API: een custom applicatie die IIIF compliant manifests levert*

HTTP Proxy

Requests verstuurd naar https://images.meemoo.be worden opgevangen op de nginx proxy-server van meemoo. De proxy routeert het verkeer op basis van het path naar de gewenste onderliggende applicatie of onderliggende functie en geeft de respectievelijke responses terug aan de "requester". Indien nodig worden tijdens dit proces URL's vertaald en herschreven en headers toegevoegd.

Daarnaast fungeert de proxy ook als content cache voor de image server. Dit gebeurt bijvoorbeeld voor de jpeg-afbeeldingen die door image server van NVDGO worden aangemaakt. De disk cache van de proxy is een aanvulling op de memcached cache van de image server.

Image API

IIIF beschrijft een set aan verplichte en optionele voorwaarden en aanbevelingen waaraan een API moet voldoen. Bijvoorbeeld op vlak van de request URI's, response formats en metadata. Het voornaamste en quasi enige doel van de image server is het aanleveren van afbeeldingen voor (her)gebruik in een image viewer of andere toepassingen, op basis van de parameters in de zogenaamde image request request.

Example 4. Image request

```
https://images.meemoo.be/iiif/<identifier:external_id>.<ext>/{region}/{size}/{rotation}/{quality}.{format}
```

Daarnaast moet de API ook kunnen antwoorden op een image information request. De response moet een json (of JSON-LD in versie 3.0)bestand teruggeven: info.json. Naast een set aan verplichte en optionele technische informatie kan dit bestand ook metadata bevatten over rechten en licenties en aan het beeldbestand gerelateerde services zoals authenticatie. Zie het hoofdstuk over info.json.

Example 5. Image information request

```
https://images.meemoo.be/iiif/\{identifier:external_id}/\{info.json}
```

Hoewel de API strikt omschreven is kunnen IIIF-compliant Image servers sterk verschillen op het vlak van gemak van installatie en configuratie, snelheid en features, roadmap en ondersteuning en de gebruikte programmeertaal (Ruby, Python, PHP, Java, C++, ...). Zie IIIF/awesome-iiif voor een uitgebreide lijst van "mature" IIIF-compliant image servers.

Voor dit document is geen technische benchmark verricht op basis van geïnstalleerde image servers. Een recente vergelijking en benchmark van image servers werd uitgevoerd door Getty en gerapporteerd in Getty Common Image Service Research & Design Report (2018). IIPImage komt hier zowel voor tiff als voor jp2 (indien met Kakadu als decoder) als betrouwbaar en consistent performant naar boven.

Typische pluspunten die naar voren worden geschoven zijn:

- Bewezen technologie. IIPImage wordt breed ingezet in productieomgevingen tot zeer grote omvang en bij grote spelers
- Jpeg2000 (jp2) ondersteuning met zowel Kakadu als de verbeterde OpenJpeg
- Stabiele prestaties en betrouwbare performantie ook met grote datasets van hogere kwaliteit (C++)
- Eenvoudige installatie(vereisten): PHP+webserver met fastcgi zoals nginx, Apache

Spreken ook in het voordeel:

- Configureerbaar via variabelen
- Matuur en nog steeds actief ontwikkeld en ondersteund
- Online beschikbaarheid van documentatie, gebruikersforum, code
- Ondersteuning voor Image API 2.1 en sinds kort ook 3.0:

> IIPImage is now fully version 3 compliant. This commit 1d6c0d5 adds a new server directive IIIF_VERSION which allows you to set the IIIF Image API version (2 or 3). Essentially this affects the info.json output.

- level 2 compliancy door toevoeging van png als output format

Er zijn echter ook minpunten:

- Embedded metadata in tiles is enkel voor tiff
- Ondersteuning voor de Image API versie 3.0 is nog maar zeer recent
- 1 hoofd-maintainer (maar wel actieve community)
- Soms veel tijd tussen releases (maar die zijn dan wel stabiel)

Aangezien meemoo reeds ervaring heeft met de installatie en opzet van een IIPImage server, inclusief Kakadu, en deze al een paar jaar stabiel in productie draait en gelet op de maturiteit en boven beschreven voordelen zal verder worden gebouwd op deze expertise.

De standaardvereisten voor de installatie van de IIPImage software (ipsrv) zijn:

- OS: Debian
- Installatie: autoconf of via apt-get (Debian 7+)
- Webserver: Apache+fcgi module (mod_fastcgi) + Nginx proxy (rewrites etc.)
- Libraries: libtiff, zlib, kakadu
- Configuratie: startup variables, httpd.conf, clean URL's en logging (syslog)

Voor meer opties en configuratiemogelijkheden zie: https://iipimage.sourceforge.io/documentation/server/

Download API

IIPImage voorziet als default enkel de "download" van het volledige bestand in jpeg- of png-formaat. Het is niet mogelijk het jp2-bestand of een rasterformaat zoals tiff te downloaden via de IIIF endpoint. Indien er voldoende vraag is naar beelden in de verschillende bestandsformaten en of andere transformatie te downloaden kan dit worden opgevangen met een kleine webapplicatie die op basis van de identifier een afgeleide in een bepaald formaat als download aanbiedt.

Tenzij dit wordt gezien als een ernstige regressie van de huidige setup van VKC, behoort deze API niet tot de scope van het project, maar dit kan op termijn wel deel uitmaken van de meemoo-setup.

Example 6. Mogelijke download request

```
https://images.hetarchief.be/download/\{identifier:external-id}.{format:jpg|jp2|tiff|png]}
```

Presentation API

De presentation API is niet in scope van dit project aangezien de bestaande API binnen het VKC-ecosysteem zal worden gebruikt. Op termijn wil meemoo wel een presentation API (3.0) voorzien, gebruik makend van de eigen shared components (meemoo-ecoysteem) voor ETL, metadata storage en API endpoint.

Impact

Voor de realisatie van de in dit document beschreven opstelling wordt een impact verwacht op:

Infrastructuur

- aanmaken en configureren van VM's
- allocatie van storage
- voorzien en configureren netwerkconnectiviteit
- DNS-configuratie
- nginx-configuratie
- monitoring en maintenance

Devops

- installatie en configuratie van de IIPImage software
- deployment configuratie: Openshift en Jenkins
- ontwikkeling custom componenten en scripts
- automatisering workflow afgeleide beeldbestanden
- monitoring en maintenance

Business

- workflow aanmaak en controle afgeleide beeldbestanden (bewerking, bijsnijden en verkleinen) ism devops
- QA en testen van de opstelling
- metadatabeheer
- opvolging

Conclusie

meemoo voorziet in het kader van het IIIF-project fase 2 van VKC een IIIF Image API. De API zal in eerste instantie een aanbod van De Schone Kunsten Musea ontsluiten voor gebruik door VKC. VKC voorziet in de presentation API. Op termijn zal de endpoint breder ingezet worden met een groter

aanbod en zowel voor gebruik door derden als voor eigen gebruik van meemoo.

De endpoint zal bestaan uit een IIPImage image server met lokale opslag en bereikbaar via een nginx proxy-server op images.meemoo.be. De implementatie gebeurt volledig op de infrastructuur van en door meemoo.

Als ontsluitingsformaat wordt jpeg2000 (jp2) gehanteerd. Er wordt een workflow uitgewerkt waarmee de archiefbestanden vlot kunnen worden omgezet naar dit formaat. De afgeleide bestanden worden op de meemoo-infrastructuur bewaard.