Pflichtenheft

Projekt: Enhancing discovery of geospatial datasets in data repositories

Auftraggeber: Daniel Nüst und Prof. Dr. Edzer Pebesma - Westfälische Wilhelms-Universität Münster

Auftragnehmer: A²HL²

Datum:

November 2018

Version:

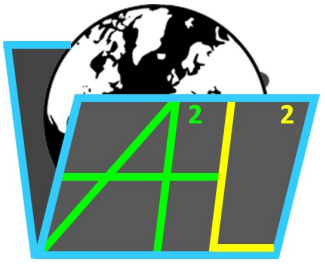
2.0

Autoren / Projektteam:

Henry Fock
Anika Graupner
Lukas Jahnich
Lia Kirsch
Aysel Tandik

Kontaktmail:

a_tand01@wwu.de



Inhalt

1. Zielbestimmung

Neue Technologien und neue (globale) Fragestellungen führen zu einer steigenden Zahl wissenschaftlicher Analysen, die Geodaten verwenden. Geodaten selbst heben sich von anderen Daten ohne räumliche Bezüge ab und sind einzigartig darin, Arbeiten und Menschen über Disziplinen hinweg zu verbinden.
Das Projekt soll es ermöglichen, basierend auf einem bereits bestehenden Geodatenkataloges, der als FOSS (Free and Open Source Software) zur Verfügung steht, die Suche nach Geodatensätzen bzw. nach deren Metadaten zu vereinfachen, indem dem Nutzer ähnliche Datensätze vorgeschlagen werden. Zudem soll es dem Nutzer möglich sein, Metadaten aus eigenen Geodatensätzen zu extrahieren und in dem Katalog zu speichern.
Für Entwickler und Betreiber von Datenkatalogen und Repositorien selbst würde diese Aufgabe eine zu große Hürde darstellen, da diese meist nicht über das erforderliche Fachwissen verfügen, die Ähnlichkeiten zwischen den (Meta-)Daten zu bewerten.
Durch das Projekt soll letztendlich die Lücke zwischen Geodatenformaten, Geodatenkatalogen und Ähnlickeitsmessungen geschlossen werden und der Zugang zu Informationen und Daten mit raum-zeitlichen Bezügen vereinfacht werden.

1.1 Musskriterien

- Suche nach Metadaten von Geodatensätzen über die API
- Berechnung der Ähnlichkeit der (Meta-)Datensätze
- Anzeigen von ähnlichen Geodatensätzen im Suchergebnis
- Vergleich von zwei Metadatensätzen auf Basis ihrer BoundingBox über die API
- Einfügen der Geodaten in ein Inputfeld zur Extraktion und Speicherung der Metadaten in ein Inputfeld
- lokale Extraktion der Metadaten aus Geodatensätzen über ein CLI-Tool
- parallele Extraktion der Metadaten aus mehreren Geodatensätzen über ein CLI-Tool
- Speicherung von Metadaten von Geodatensätzen in der Datenbank
- Verwalten von Metadaten von Geodatensätzen in einer Datenbank
- Aufrufen einer Kartenanwendung über die API
- Suche nach Metadaten von Geodatensätzen über eine Kartenanwendung, über die man sich Metadaten zu Geodatensätzen anzeigen lassen kann, die in einem bestimmten Gebiet liegen

1.2 Wunschkriterien

- Hochladen und Speicherung von Geodatensätzen über ein Inputfeld in einer Datenbank
- Änderung von Geodatensätzen

1.3 Abgrenzungskriterien

- es wird der Geodatenkatalog PyCSW verwendet und kein Repositorium

2. Produkteinsatz

Die Anwendung dient zur Suche von Geodatensätzen über deren Metadaten. Zudem sollen ähnliche Datensätze zu dem gesuchten Datensatz angezeigt werden.

2.1 Anwendungsgebiete

Das Projekt erweitert den Geodatenkatalog des Kunden um die genannten Funktionen, um diesen auch gegenüber Wettbewerbern zu verbessern und die Arbeit folgender Zielgruppen zu erleichtern.

2.2 Zielgruppen

- Wissenschaftler: Das Projekt erleichtert die Suche nach ähnlichen oder verwandten Arbeiten (während der Recherche, beim Schreiben von Manuskripten, beim Auswerten von Beiträgen), da durch das Anzeigen der ähnlichen Datensätze das Teilen von Informationen beschleunigt wird. Auch können Arbeiten verglichen werden, auf eventuelle Fehler geprüft werden und neue Herangehensweisen entdeckt werden, wenn mit anderen Wissenschaftlern in den Austausch gegangen wird.

- Betreiber von Repositorien und Katalogdiensten: Das Projekt kann als Orientierung dienen, um bestimmte Funktionen in eigene Dienste zu integrieren bzw. um Ideen für eigene Funktionen zu sammeln.

- Archivare / Datenerhalter: Der Zugang zu Suchergebnissen (Daten, Artikeln) wird durch das Projekt verbessert, sodass Daten und Informationen einfacher geteilt und verbreitet werden können und Daten schon vor der Verwendung durch die Ähnlichkeitsmessungen in Beziehung gesetzt werden.

2.3 Betriebsbedingungen

Die Anwendung kann über jedes Gerät mit Internetzugang ausgeführt werden, das auch Docker installiert hat, z.B. PC, Laptop, Notebook. Auf Mobiltelefonen bzw. Smartphones ist eine Installation von Docker nicht möglich.

3. Funktionale Anforderungen

3.1 Extraktion

FE001: Das Programm unterstützt die folgenden Datenformate zur Extraktion von Metadaten: GeoPackage, NetCDF, GeoJSON, Shapefile, CSV on the Web, ISO 19xxx, GeoTIFF.

FE002: Es soll für den Nutzer möglich sein, über ein CLI-Tool den räumlichen Umfang der Daten zu extrahieren. Dies soll mit verschiedenen Detailleveln verwirklicht werden:

- bei Geodatensätzen mit mehreren Geometrien wird eine Bounding Box als räumlicher Umfang ausgegeben
- bei einzelnen Punkten soll nur der Punkt ausgeben werden
- bei einzelnen Linien werden die Endpunkte der Linie ausgegeben

Die Extraktion ist an das jeweilige Format angepasst. Dafür werden verschiedene Module verwendet:

Module, die in Python integriert sind.	Externe Module.
- csv - os - sys - json - from xml.etree import ElementTree as ET - from os import listdir - from os.path import isfile, join - from xml.dom.minidom import parse	- from osgeo import gdal, ogr, osr - click - netCDF4 - pandas - pygeoj - pyshp - xarray - ogr2ogr

FE003: Es soll für den Nutzer möglich sein, über ein CLI-Tool den zeitlichen Umfang der Daten zu extrahieren. Die zeitlichen Daten werden, falls sie vorhanden sind, ausgelesen und in ein einheitliches Format umgewandelt. Das gewählte Format entspricht dem Timestamp-Format ISO8601.

Die Extraktion ist an das jeweilige Format angepasst. Dafür werden verschiedene Module verwendet:

Module, die in Python integriert sind.	Externe Module.
- csv - os - sys - json - math - functools - from xml.etree import ElementTree as ET - from os import listdir - from os.path import isfile, join - from xml.dom.minidom import parse	- from osgeo import gdal, ogr, osr - click - netCDF4 - pandas - pygeoj - pyshp - xarray - ogr2ogr - DateTime

FE004: Der Nutzer kann den räumlichen und zeitlichen Umfang aller unterstützten Daten aus einem Datenverzeichnis extrahieren. Das Ergebnis wird in einem zweidimensionalen Array oder als CSV gespeichert**.**

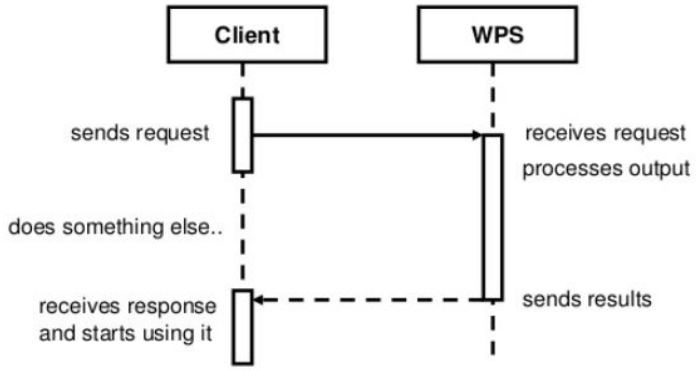
Datei 1	Datei 2	Datei 3
Boundingbox [lonMin, latMin, lonMax, latMax]	Boundingbox [lonMin, latMin, lonMax, latMax]	Boundingbox [lonMin, latMin, lonMax, latMax]
Zeitlicher Umfang [start, ende, intervall]	Zeitlicher Umfang [start, ende, intervall]	Zeitlicher Umfang [start, ende, intervall]

FE005: Ein Nutzer kann für eine Datei über einen API Aufruf die Extraktion der Metadaten anfordern, sodass die Metadaten des Datensatzes sofort aktualisiert werden. Über die API wechselt der Benutzer auf eine Seite mit einem Inputfeld, in das veränderte Geodatensätze eingefügt werden können, um den Metadatensatz des Geodatensatzes zu aktualisieren. Dabei wird zuerst überprüft, ob es schon einen Metadatensatz mit dem gleichen Namen wie den eingefügten Geodatensatz gibt. Falls ja, wird der Metadatensatz aktualisiert, falls nein, wird ein neuer Metadatensatz eingefügt.

FE006: Wenn ein oder mehrere Geodatensätze hochgeladen werden, wird die Extraktion der Metadaten automatisch gestartet und die Metadaten werden in der Datenbank gespeichert. Die Geodatensätze werden allerdings nicht gespeichert.

FE007: Metadatensätze sollen weiterhin extrahiert werden, während gleichzeitig neue Datensätze erstellt werden. Dieses Vorgehen soll in einem unabhängigen Prozess verwirklicht werden. Dieser Punkt lässt sich bei unserem Vorgehen nicht verwirklichen, da wir keine Geodatensätze speichern und auch keine neuen in pycsw erstellen können.

FE008: Die client-server Kommunikation soll event-basiert gestaltet werden. Das heißt, wenn der Nutzer z.B. eine Extraktion startet, kann er während die Extraktion im Hintergrund läuft, pycsw normal weiter verwenden. Wenn die Extraktion fertig gestellt wurde, erhält er eine Push-Benachrichtigung. Die folgende Abbildung verdeutlicht den Vorgang:



(Quelle:https://image.slidesharecdn.com/2014-01-22-wps-standard-140227034627-phpapp01/95/the-web-processing-service-standard-benjamin-pro-20-638.jpg?cb=1393474074)

3.2 API

FA001: Alle Nutzerbezogene Funktionalitäten sind über RESTful http API „endpoints“ verfügbar:

/?service=CSW&version=2.0.2...	
...&request=GetCapabilities	Auflistung der Funktionalitäten, die der Dienst beinhaltet.
...&request=GetRecordByID&id=<id>	Suche nach einem (Meta-)Datensatz über die verzeichnisspezifische ID. Antwort enthält die einzelnen Metadaten des Datensatzes und eine sortierte Liste mit höchstens 20 ähnlichen Datensätzen und ihrem Ähnlichkeitswert.
...&request=GetRecordByID&id=<id>&similar=<n>	Suche nach einem (Meta-)Datensatz über die verzeichnisspezifische ID. Antwort enthält die einzelnen Metadaten

	des Datensatzes und eine sortierte Liste mit höchstens n ähnlichen Datensätzen und ihrem Ähnlichkeitswert.
…&request=GetSimilarityBBox&id1=<id>&id2=<id>	Vergleicht zwei Datensätze über ihre verzeichnisspezifischen IDs nur auf Basis ihrer Boundingboxen.
…&request=ExtractMetadata	Wechsel auf die Seite mit dem Inputfeld, mit dem Metadaten aus eingefügten Geodaten extrahiert und in der Datenbank gespeichert werden (Transaction insert)..
…&request=GetMap	Wechsel auf die Seite der Kartenanwendung.

FA002: API endpoints geben gültige JSON als Antwort zurück, inklusive der Errors. Dies wird durch “application/json” erfüllt.

FA003: API endpoints nutzen angemessene HTTP Statuscodes, die pycsw bereits enthält:

200	- OK → wird ausgegeben, wenn die Metadaten erfolgreich integriert wurden → wenn die Ähnlichkeitsberechnung erfolgreich durchgeführt wurde
400	- InvalidValue: Invalid property value - OperationParsingFailed: Bad request - OperationNotSupported: Not Implemented - MissingParameterValue: Bad Request - InvalidParameterValue: Bad Request - VersionNegotiationFailed: Bad Request - InvalidUpdateSequence: Bad Request - OptionNotSupported: Not Implemented - NoApplicableCode: Internal Server Error
403	- OperationProcessingFailed: Server Processing Failed
404	- NotFound

FA004: Raumbezogene Daten aus der API sind in GeoJSON (RFC 7946) kodiert.

Beispiel GeoJSON Kodierung nach RFC 7946
(Quelle: https://tools.ietf.org/html/rfc7946):

```
{
   "type": "FeatureCollection",
   "features": [{
      "type": "Feature",
      "geometry": {
         "type": "Point",
         "coordinates": [102.0, 0.5]
      },
      "properties": {
         "prop0": "value0"
      }
   }, {
      "type": "Feature",
      "geometry": {
         "type": "LineString",
         "coordinates": [
            [102.0, 0.0],
            [103.0, 1.0],
            [104.0, 0.0],
            [105.0, 1.0]
         ]
      },
}
```

FA005: Erweiterte Metadaten, die beispielsweise extrahierte raumbezogene und zeitliche Informationen beinhalten, sind in den regulären Metadaten-Dokumenten mit inbegriffen. Das heißt Metadaten, wie z.B. Titel, Autor usw. die bereits fertig in den Geodatensätzen enthalten sind, werden auch extrahiert und zusammen mit den räumlichen und zeitlichen Umfängen in einem Dokument gespeichert.

FA006: Mit dem hinzugefügten Parameter „similar = n“ zu einer Anfrage bzw. in einem Endpunkt um Datensätze zu lesen, bekommt der Nutzer die IDs und die Ähnlichkeitswerte für n ähnliche Datensätze. Wenn der Nutzer diesen Parameter nicht hinzufügt, werden alle IDs und Ähnlichkeitswerte angezeigt, die dem Mindestwert der Ähnlichkeit entsprechen, der unter 3.3 Ähnlichkeitsberechnung definiert ist. Es werden höchstens 20 ähnliche Datensätze (also die mit den 20 ähnlichsten Werten) angezeigt. Wenn der vom Benutzer eingegebene Wert “n” höher ist, als die dem Mindestwert entsprechenden ähnlichen Datensätze, werden weniger Datensätze als “n” für den Benutzer angezeigt.

3.3 Ähnlichkeitsberechnung

Bestimmung des Ähnlichkeitswertes / Ähnlichkeitsberechnung:

Die Datensätze werden über ihre gespeicherten Metadaten verglichen. Sobald ein neuer Metadatensatz in der Datenbank gespeichert wird, wird dieser automatisch mit allen anderen Metadatensätzen verglichen und das Ergebnis wird in einer anderen Tabelle “Similarities” in der Datenbank mit den folgenden Spalten gespeichert:

record1	record2	total_similarity	geospatial_extend	temporal_extend	dataformat	autor	title

In der Spalte record1 befindet sich die ID des neu hinzugefügten Metadatensatzes, in der Spalte record2 befindet sich die ID des Datensatzes, der schon vorhanden war, d.h. wenn der neue Datensatz hinzugefügt wird, werden so viele neue Zeilen der Tabelle “Similarities” hinzugefügt, wie es bereits bestehende Datensätze gibt.
Wenn ein Metadatensatz aktualisiert wird, werden in der Tabelle “Similarities” alle Zeilen gelöscht, die die ID des aktualisierten Datensatzes in record1 oder record2 beinhalten. Der aktualisierte Datensatz wird dann erneut mit allen schon bestehenden Datensätzen verglichen.
Alle Metadaten (also die Spalten geospatial_extend, temporal_extend, dataformat, author, title), die verglichen werden, bekommen einen maximal-Score, wie wichtig sie für die Berechnung der Ähnlichkeit sind. Diese Metadaten haben zudem gegebenenfalls noch Unterklassen, durch die der letztendliche Score für jede dieser Spalten berechnet wird. Diese werden aber nicht in extra Spalten gespeichert und sind nur für den internen Algorithmus notwendig.
In der Spalte total_similarity werden dann die Werte der anderen Spalten zusammen gerechnet, der maximale Wert läge hier bei 100%, dieser wird aber nie erreicht werden (siehe FS001). Zwei Datensätze gelten als ähnlich, wenn sie einen Mindestwert in der Spalte “total_similarity” von 51% erreichen.
Durch diese Vorgehensweise soll verhindert werden, dass ein Algorithmus, der erst bei einer Anfrage über die API zu laufen beginnt, das System verlangsamt.

Die Tabelle auf der nächsten Seite veranschaulicht die Vergabe der Werte für die Berechnung der Ähnlichkeit zweier (Meta-)Datensätze.

Verglichene Metadaten	**Gewichtung in %**	**Bemerkungen**
geospatial_extent	40	Da es um den Vergleich von Metadaten zu Geodatensätzen geht, ist der räumliche Umfang natürlich am höchsten bewertet.
Überschneidung der Boundingboxen	15	Daten liegen ungefähr im gleichen Gebiet. Es wird die sich überschneidende Fläche berechnet und desto näher diese am Wert der größeren der beiden Flächeninhalte ist, desto höher ist die Ähnlichkeit. Wenn die Schnittfläche dem Wert der größeren Boundingbox entspricht, liegt der Wert bei 15.Wenn es keinen Schnitt gibt, liegt der Wert bei 0.
Ähnlichkeit der Flächeninhalte	10	Ähnliche Größen der Flächeninhalte könnten auf ähnliche Inhalte der Geodaten schließen lassen (z.B. Polygone Ländergrenzen).Wenn die Flächen gleich sind, beträgt der Wert 10.Wenn unterschiedliche Detaillevel (siehe FE002) verglichen werden, beträgt der Wert 0.
räumliche Nähe	10	Zwei Messstationen innerhalb einer Stadt können interessant sein. 10 Punkte, wenn der Ort (Zentrum) der selbe ist. Je weiter diese sich entfernen, desto kleiner ist der Wert
Gleicher Geometrietyp	5	Auch ein gleicher Geometrietyp könnte auf ähnliche Inhalte der Geodaten schließen lassen (z.B. Linien für Straßen). Gleich = 5 Ungleich = 0
temporal_extent	30	Geodatensätze haben oft auch einen zeitlichen Bezug, deshalb wird der zeitliche Umfang am zweit höchsten bewertet.
Überschneidung von Zeitreihen	10	Wichtig, wenn z.B. Daten für einen bestimmten Zeitraum gesucht werden. Es wird die sich überschneidende Zeitreihe berechnet und desto näher diese am Wert der größeren der beiden Zeitreihen ist, desto höher ist die Ähnlichkeit. Vollständige Überschneidung: 10 Keine Überschneidung: 0
Länge der Messungen	10	Muss beachtet werden, da z.B eine sehr kurze Zeitreihe in einer längeren liegen könnte, der zeitliche Umfang deshalb bei z.B. 30% läge, obwohl der eigentliche Schnitt nur sehr klein ist. Gleiche Länge: 10
Durchschnittliches Zeitintervall	10	Intervalle können sich unterscheiden, Messungen können z.B. sekündlich, stündlich oder einmal pro

		Woche über einen längeren Zeitraum durchgeführt worden, in der Masse aber gleich sein.
dataformat	20	Gleiche Datenformate beinhalten ähnliche Arten von Daten.
Gleichheit Vektorformat / Rasterformat	10	Ein Datenformat für Geodaten unterscheidet sich in erster Linie schon mal in Formaten für Vektordaten und Rasterdaten, weshalb das hier auch berücksichtigt werden muss. Gleich = 10 Ungleich = 0
Gleicher Dateityp	10	Zudem sollte der Gesamtwert für “dataformat” höher sein, wenn letztendlich zweimal genau der gleiche Dateityp vorliegt. Gleich = 10 Ungleich = 0
title	5	Ein ähnlicher Titel gibt Aufschluss über den Inhalt der Daten. Je exakter sich die Wörter überschneiden, desto höher ist der Wert. Vollständige Überschneidung: 5 Keine Überschneidung: 0
author	5	Die Wahrscheinlichkeit, dass sich ein Autor mit ähnlichen Daten befasst, ist relativ hoch. Je exakter sich die Wörter (also die Namen der Autoren, es kann auch mehrere geben) überschneiden, desto höher ist der Wert. Vollständige Überschneidung: 5 Keine Überschneidung: 0

Weitere Anforderungen an die Ähnlichkeit(sberechnung):

FS001: Es gibt einen API Endpoint, der zwei verzeichnisspezifische IDs als Parameter von zwei verschiedenen Datensätzen beinhaltet, die nur auf Basis ihrer Boundingboxen verglichen werden sollen. Der Ähnlichkeitswert hierfür wurde bereits bei der Speicherung des neueren Metadatensatzes berechnet und in der Tabelle “Similarities” unter der Spalte “geospatial_extend” gespeichert.

FS002: Ein API endpoint stellt eine sortierte Liste mit ähnlichen Datensätzen für eine Repositorium spezifische Datensatz ID. Es werden höchstens 20 ähnliche Datensätze (also die mit den 20 ähnlichsten Werten) angezeigt.

FS003: Der Ähnlichkeitswert ist im Intervall [0,1[normalisiert. Werte gegen 0 weisen hierbei auf eine geringe Ähnlichkeit und Werte gegen 1 auf eine hohe Ähnlichkeit hin.
Die 1 ist aus dem Intervall ausgeschlossen, da es sich bei einem Wert von eins um einen vollständig ähnlichen bzw. um den gleichen Datensatz handeln würde. Ein Datensatz wird aber auch nie mit sich selbst verglichen (siehe FS004).

FS004: Der Input Datensatz wird nicht in der Ähnlichkeitsliste mit aufgeführt. In der Tabelle “Similarities” gibt es deshalb keine Zeile, bei der die ID in der Spalte “record1” der ID in der Spalte “record2” entspricht.

FS005: Liegt ein gleicher Datentyp vor, so ist der Ähnlichkeitswert höher als bei nicht gleichen Datentypen (siehe Ähnlichkeitsberechnung).

3.4 UI

FU001: API endpoints geben gültige JSON als Antwort zurück.

FU002: Über einen API Endpunkt (request=ExtractMetadata) gelangt der Nutzer auf eine HTML Seite mit einem Inputfeld, in das die Geodatensätze eingefügt werden können, aus denen Metadaten extrahiert werden sollen.

FU003: Über einen API Endpunkt (request=GetMap) gelangt man auf eine HTML Seite mit einer Kartenansicht (Bonus Aufgabe). Diese beinhaltet einen Geocoder, um an gewünschte Orte zu gelangen.
Bei der Karte selber handelt es sich um eine Heatmap, die eine Übersicht über vorhandene (Meta-)Daten in dem angezeigten Kartenfenster gibt. Der Nutzer kann in einen Bereich der Karte klicken, sodass sich eine Liste öffnet, in der die Metadatensätze für diesen Bereich angezeigt werden. Der Bereich ist abhängig von einem eingestellten Radius in Abhängigkeit von der Stelle, auf die der Benutzer klickt. Ein Mockup zur Anwendung befindet sich unter “3.6 Bonuseigenschaft”.

FU004: Das Client Tool zur Extraktion der Metadaten lässt sich extern über eine Konsole nutzen, die Daten werden dann direkt im Konsolenfenster angezeigt.

FU005: Die Sprache der Benutzeroberfläche ist auf Englisch.

3.5 Konfiguration

FC001: Alle Konfigurationen von zusätzlicher Funktionalität sind über Textdateien z.B. YAML Format möglich und idealerweise mit dem Konfigurationsmechanismus von der Basissoftware realisierbar.
Konfigurationen können über zusätzliche Textdateien ausgeführt werden, die in verschiedenen Formaten vorliegen können. Die Konfigurationen sollten allerdings den Konfigurationseigenschaften von PyCSW entsprechen, die in der Dokumentation von PyCSW aufgeführt sind.

FC002: Die Konfiguration ist spätestens nach dem Neustart des Dienstes aktiv.

3.6 Bonus Eigenschaft

“Geospatial search and map-based browsing of records.”

Die Oberfläche zur Bonuseigenschaft ist bereits unter 3.4 UI, FE002 beschrieben.
Hierbei handelt es sich um eine HTML Seite und ein zugehöriges JavaScript. Bei der Karte handelt es sich um eine Mapbox Karte mit Heatmap Funktionalität. Die Heatmap wird durch eine GeoJSON erstellt, in der sich alle in pycsw gespeicherten Daten befinden.

Über einen API Endpunkt (request=GetMap) gelangt man auf eine HTML Seite mit einer Kartenansicht (Bonus Aufgabe). Diese beinhaltet einen Geocoder, um an gewünschte Orte zu gelangen.
Bei der Karte selber handelt es sich um eine Heatmap, die eine Übersicht über vorhandene (Meta-)Daten in dem angezeigten Kartenfenster gibt. Der Nutzer kann in einen Bereich der Karte klicken, sodass sich eine Liste öffnet, in der die Metadatensätze für diesen Bereich angezeigt werden. Der Bereich ist abhängig von einem eingestellten Radius in Abhängigkeit von der Stelle, auf die der Benutzer klickt:

Author	Name der Datei	Boundingbox	Zeitspanne
Peter	Messdaten in Blablabla	[3,4,6,7]	2015 - 2016
Hans	lalalalalalala	[5,3,8,6]	1995 - 2002

4. Testdaten

Für jedes unterstützte Dateiformat werden mehrere Testdaten zur Verfügung gestellt. Diese werden von öffentlichen Quellen genommen. Sie beinhalten sowohl korrekte als auch inkorrekte Daten.

Link zu den Daten:

https://uni-muenster.sciebo.de/s/QFj5pzm7AzxAh1f?path=%2FTestdaten-A%C2%B2HL%C2%B2

Datenquellen:

geojson, kml, csv, shapefile	http://geojson.io
csv	https://kunden.dwd.de/weste/xl_register.jsp
shapefile	https://download.geofabrik.de/europe/germany/nordrhein-westfalen.html
geotiff	https://pangaea.de/?q=geotiff
geopackage	https://figshare.com/search?q=geopackage&searchMode=1
geojson	https://gist.github.com/aaronlidman/7894176?short_path=d0d520b#file-featurecollection-geojson
netcdf	https://www.unidata.ucar.edu/software/netcdf/examples/files.html https://zenodo.org/search?page=1&size=20&q=netcdf&file_type=nc&access_right=open
gml	http://worldmap.harvard.edu/data/geonode:mypolygon_px6 https://github.com/gephi/gephi/wiki/Datasets https://themes.jrc.ec.europa.eu/file/view/107486/example-of-gml-file-conformant-to-inspire-land-cover-vector-application-schema

5. Qualitätsanforderungen

Produktqualität	Sehr Gut	Gut	Normal	Nicht Relevant
Funktionalität				
Angemessenheit			x	
Richtigkeit		x		
Interoperabilität		x		
Ordnungsmäßigkeit		x		
Sicherheit		x		
Zuverlässigkeit				
Reife			x	
Fehlertoleranz		x		
Wiederherstellbarkeit		x		
Benutzbarkeit				
Verständlichkeit	x			
Erlernbarkeit		x		
Bedienbarkeit	x			
Effizienz				
Zeitverhalten		x		
Verbrauchsverhalten			x	
Änderbarkeit				
Analysierbarkeit		x		
Modifizierbarkeit		x		
Stabilität		x		
Prüfbarkeit		x		

6. Nicht-funktionale Anforderungen

6.1 Wartbarkeit

Die Software wird als Open Source Software unter einer OSI genehmigten Lizenz veröffentlicht unter Berücksichtigung der Lizenzen anderer verwendeter oder erweiterter Software (siehe 7.4 Lizenzen).
Die Dokumentation zum Aufbau, zur Konfiguration und zur Installation des Systems muss in einer Markdown-formatierten Dokumentationsdatei bereitgestellt werden, wobei die entsprechende Markierung für Listen, Links usw. verwendet wird.

6.2 Benutzerfreundlichkeit

Das System muss eine intuitive Verwendung unterstützen, so dass die Zielgruppen ihre Hauptfunktionen (ohne Bonusfeature) ohne weitere Dokumentation nutzen können. Deshalb beinhaltet die Anwendung z.B. ein Inputfeld zur Extraktion der Metadaten, da ein Inputfeld im Gegensatz zu einem Endpunkt den Arbeitsablauf des Nutzers vereinfacht, weil es visuell vorstellbar und interaktiver ist, da der Nutzer auf einen Knopf drückt und nicht selber Befehle eingeben muss. Der Endpunkt, mit dem man auf die Seite des Inputfeldes gelangt, ist für den Nutzer bereits vorgefertigt.

6.3 Leistung

- die API-Aufruf Antwortzeit eines Aufrufes an einen Geodatenkatalog mit dem Anzeigen von ähnlichen Datensätzen soll höchstens doppelt so lange dauern wie die Antwortzeit eines Aufrufes ohne das Anzeigen ähnlicher Datensätze

- das Aufrufen einer Seite soll in unter fünf Sekunden umgesetzt werden können

- komplexe Seiteninhalte können unabhängig voneinander geladen werden

- interaktive Visualisierungen sollen in unter drei Sekunden reagieren können

→ die Leistungen werden mit Hilfe von Unittests getestet

7. Technische Produktumgebung

7.1 Software

Betriebssysteme:

- Windows 8, 10
- MacOS 10.13.6

Basissoftware:

- PyCSW 2.2.0

Datenbank:

- SQLite3

Browserkompabilität:

- Mozilla Firefox 60.0 oder höher
- Google Chrome 70.0.3538 oder höher
- Opera 57.0.3098.76

Verwaltungsplattform:

- GitHub

7.2 Hardware

- Laptop, Notebook
- PC

7.3 Produktschnittstellen

- RESTful HTTP API endpoints
- CLI-Tools

7.4 Lizenzen

- GitKraken non-commercial use
- Public domain (SQLite3)
- Permissive FOSS license (MapBox)
- X/MIT (GDAL)

Unsere Lizenz: X/MIT (höchste Lizenz der oben aufgelisteten)

8. Projektmanagement und Zeitplan

8.1 Projektmanagement

Für unser Projekt orientieren wir uns an einer Mischung aus traditionellem und agilem Projektmanagement. Wir haben versucht, die Zeitpunkte für unsere Aufgaben bereits im voraus zu planen. Allerdings sprechen und planen wir jede Aufgabe vor Beginn dieser durch. Die entsprechenden Teammitglieder versuchen dann, die Aufgabe zu erfüllen und schreiben am Ende gegebenenfalls schon Testfälle. Dann werden die fertigen Ergebnisse den anderen Mitgliedern vorgestellt und über Verbindungen und mögliche Änderungen im Bezug auf andere Aufgaben gesprochen. Somit versuchen wir, trotz der begrenzten Zeit iterativ vorzugehen.

Wir treffen uns regelmäßig zweimal in der Woche (Mittwochs ab 10:00, Donnerstags ab 11:00). Mittwochs wird zunächst geklärt, wie weit jeder mit seiner momentanen Aufgabe gekommen ist und ob es Probleme gibt:

> Da uns Kommunikation im Team sehr wichtig ist, sollte immer eine Atmosphäre geschaffen sein, in der jeder rechtzeitig Bescheid geben kann, wenn bei seiner Aufgabe eine Problem besteht. Die anderen Mitglieder haben dann je nach freier Zeit die Möglichkeit, sich mit dem genannten Problem auseinanderzusetzen.
> Falls nach längerer intensiver Sucher noch keine Lösungen gefunden wurden, werden wir auf andere Teams zugehen und fragen, ob bei diesen ähnliche Probleme aufgetreten sind und/oder ob diese Ideen für Lösungsansätze haben. Die Kommunikation erfolgt entweder persönlich oder über die öffentlichen Mattermost Chats (pycsw, python).
> Falls auch über diesen Weg keine Lösung gefunden wurde, wenden wir uns an den Auftraggeber, Daniel Nüst. Auch hier erfolgt die Kommunikation entweder persönlich Mittwochs um 14:00 Uhr in der regelmäßigen Sprechstunde oder über Mattermost bzw. per E-Mail.

Fertiggestellte Aufgaben werden den anderen Gruppenmitgliedern vorgestellt. Gegebenenfalls werden neue Aufgaben verteilt. Unsere detaillierte Aufgabenplanung machen wir in Glo von GitKraken. Wir arbeiten hauptsächlich alleine oder in Zweier- bis Dreiergruppen an einzelnen Aufgaben.
Ab 14:00 Uhr nutzen wir die Fragestunde, um offene Fragen direkt beim Auftraggeber zu klären bzw. um diesen auf den neuesten Stand zu bringen.
Neben Mittwoch und Donnerstag kann sich jeder die restliche Zeit, die er für diese Woche noch am Projekt arbeiten will oder muss, zuhause selbstständig einteilen. Offene Fragen stellen wir uns dann gegenseitig in unserer Whatsapp Gruppe oder in Mattermost, je nachdem, welches Format sich besser eignet.

8.2 Zeitplan

Der nachfolgende Zeitplan enthält aktuell unsere geplanten Aufgaben mit den voraussichtlichen Zeiträumen, in denen diese fertiggestellt werden sollen. Die Aufgaben sind aufsteigend nach ihren Startzeitpunkten geordnet. Der Zeitplan wird nach und nach bearbeitet bzw. weiter ausgefüllt. Die verantwortlichen Teammitglieder werden je nach Auslastung und Know-How spätestens vor Beginn einer Aufgabe eingeteilt. Zudem helfen uns die Spalten “IST-Beginn” und “IST-Ende” dabei zu erkennen, falls wir die Zeit falsch einkalkuliert haben, sodass wir rechtzeitig reagieren können. Ist eine Aufgabe nicht rechtzeitig erledigt muss sich erneut mit ihr beschäftigt und nach der Lösung für das Problem gesucht werden. Die erste Weihnachtsferienwoche wird nicht für das Projekt eingeplant. Die zweite Weihnachtsferienwoche dient als Puffer für Aufgaben und Probleme, die noch nicht gelöst werden konnten. Es wird aber versucht werden, den Zeitplan möglichst einzuhalten.
Wie lange die einzelnen Aufgaben dauern können wir nur schätzen, da niemand vorher mit exakt so einer Aufgabe betraut war. Wir als Team bringen aber bereits folgendes Vorwissen und folgende Kompetenzen aus Studium und bereits bestehenden Arbeitsverhältnissen mit:

- Erfahrungen aus dem Geosoftware I Modul
- Arbeiten mit Testfällen
- Managen eines Teams
- Arbeiten im Team
- Arbeiten mit Datenbanken

Auf der nachfolgenden Seite befindet sich der Zeitplan.

Aufgabe	SOLL-Beginn	IST-Beginn	SOLL-Ende	IST-Ende	Bemer-kungen
Start des Projektes	17.10.2018	17.10.2018	-	-	
Pflichtenheft erstellen	17.10.2018	17.10.2018	31.10.2018	28.11.2018	Die Frist wurde vom Auftraggeber verlängert.
Aufsetzen von PyCSW	17.10.2018	17.10.2018	24.10.2018	31.10.2018	
CLI-Tool räumlicher Umfang	24.10.2018	24.10.2018	05.12.2018		
SQlite3 Datenbank erstellen und Verbindung aufbauen	24.10.2018	31.10.2018	14.11.2018	14.11.2018	
API	07.11.2018	07.11.2018	19.12.2018		
CLI-Tool zeitlicher Umfang	14.11.2018	14.11.2018	05.12.2018		
Ähnlichkeits-berechnung	05.12.2018		19.12.2018		
Extraktion der Metadaten und Speicherung in der Datenbank	05.12.2018		19.12.2018		
Bonus Feature	09.01.2019		16.01.2019		
Dokumentation zum Aufbau, zur Konfiguration und zur Installation des Systems	16.01.2019		23.01.2019		
Projektende und Präsentation	30.01.2019	30.01.2019	30.01.2019	30.01.2019	

→ Testfälle werden während des Projektes geschrieben und gehören zu jeder Aufgabe (bei der Testfälle nötig sind) dazu. Deshalb sind diese in der Tabelle nicht extra aufgeführt.