GeoBib

GEOREFERENZIERTE ONLINE-BIBLIOGRAPHIE FRÜHER HOLOCAUST- UND LAGERLITERATUR

SYSTEMDESIGN AUF BASIS DER ANFORDERUNGSANALYSE AUS AP2 (M6.1R)

(VERARBEITUNGSPIPELINE, ENTWICKLUNGSPROZESS, SYSTEMARCHITEKTUR, DATENBANKDESIGN, DOKUMENTATION)

Ines Schiller, Bastian Entrup

30. Sept 2013

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Systemevaluation

An dieser Stelle werden verschiedene Systeme aufgeführt. Diese sind ein Auszug aus dem großen Repertoire, die es insgesamt gibt. Es werden Content Management Systeme, Map-Server und Datenbanken betrachtet. Diese wurden zu Beginn des GeoBib-Projektes in die engere Wahl genommen und daher miteinander verglichen und evaluiert. Eine Ausnahme bildet das letzte Unterkapitel (Kapitel 0); an dieser Stelle wird die Software nicht gegenübergestellt. Dieses Kapitel behandelt Software, die für die weitere Anwendung als Anwendungstools dienen.

1.1 Content Management System (CMS)

Content Management Systeme werden immer häufiger als Grundlagensysteme von Webseiten genutzt, da sie einen einfachen und schnellen Aufbau durch Templates und Themes ermöglichen. Durch Templates wird der Aufbau der Seite definiert. Mit Themes wird das Design an sich definiert. Einige Systeme sind relativ komplex und bieten dem Nutzer durch Plug-ins eine komfortable Lösung der Webseitengestaltung. An dieser Stelle werden einige CM-Systeme vorgestellt und miteinander verglichen. Die Auswahl des CMS für das GeoBib-Projekt wird anhand von bestimmten Kriterien durchgeführt, die beim Vergleich der Produkte Betrachtung finden.

1.1.1 Cartaro auf Basis von Drupal

Drupal ist ein relativ großes CMS. Durch eine Vielzahl an verschiedenen Modulen und Themes lassen sich der Aufbau und das Design von Drupal verhältnismäßig schnell und unkompliziert verändern und anpassen (Drupal – Weblink). Spezielle Designwünsche müssen meist durch eigene Erweiterungen umgesetzt werden. Diese setzen voraus, dass der Aufbau und das Verhalten von Drupal bekannt sind. Wird ein Theme angepasst, müssen bestimmte Dinge beachtet werden, da sonst das veränderte Theme nach Updates wieder mit dem ursprünglichen Theme überschrieben wird. Module, die nicht nach den genauen Programmierregeln von Drupal erstellt werden, führen zu Problemen oder können erst gar nicht eingebaut werden (Drupal API – Weblink).

Ein weiterer Punkt sind die verschiedenen Versionen von Drupal. Jede Version bedingt es, dass ältere Module und Themes angepasst werden müssen. Somit gibt es einige Module, die für bestimmte Dinge nutzbar wären, aber vom Ersteller nicht mehr weiterentwickelt werden. Daher kann es sein, dass man dieses Modul nicht nutzen kann, da ggf. eine neuere Version genutzt wird (Drupal – Weblink).

Die Dokumentation in Drupal ist ein weiterer nicht trivialer Punkt. Es gibt eine Hauptseite, auf der man alle Module und Themes findet. Teilweise ist die Verwendung von Modulen gut und einfach beschrieben, andere Teile wiederum nicht. Daher ist an einigen Stellen eine längere Einarbeitungsphase nötig.

Cartaro ist ein recht umfangreiches Modul für Drupal. Es soll ermöglichen, Karten darzustellen und diese mit Vektor- und/oder Rasterdaten zu erweitern. Diese Daten werden über den GeoServer, der zusätzlich installiert werden muss, verwaltet und nutzbar gemacht. Somit kann ein Web-GIS erstellt werden (Cartaro - Weblink).

Die Installation von Cartaro ist nicht sehr einfach gelöst, da es nicht über die in Drupal eingebettete Funktion geschehen kann. Updates sind daher auch nur durch einen direkten Zugang zum Server möglich. Durch diese Hindernisse, ist es schwer mit dem Modul zu arbeiten. Die Bedienung an sich ist auch gewöhnungsbedürftig, da eine gute Dokumentation fehlt und man in den Fehlerberichten und Supportanfragen suchen muss. Die Installation des GeoServers ist dagegen recht gut beschrieben. Aber auch hier müssen ein paar Dinge direkt serverseitig angepasst werden. Dies fällt so schwer ins Gewicht, dass das gute Design der Kartendarstellung die Nachteile nicht aufwiegt. Wie eine geographische Suche mit dem Modul umgesetzt werden könnte, konnte auf Grund der Probleme bei der Installation und der Komplexität der Anwendung nicht getestet werden.

1.1.2 PmWiki

PmWiki ist ein relativ einfach gehaltenes CMS. Es ist ein freies wiki-basiertes System unter der *GNU General Public License* (GNU GPL). Ziel von PmWiki ist, die Installation und Konfiguration einfach und es für viele Anwendungen nutzbar zu halten (PmWiki – Weblink).

Es gibt eine Vielzahl an Modulen und Templates. Die Module heißen bei PmWiki Cookbooks und können durch einfaches Hinzuladen und Einbinden in die Konfigurationsdatei aktiviert werden. Das Anpassen der Cookbooks ist ohne Probleme möglich. Hier werden nur Programmierkenntnisse der jeweiligen Programmiersprache benötigt. Meist sind nur PHP- und ggf. JavaScript-Kenntnisse erforderlich. Eine Erstellung neuer Cookbooks ist mit Programmierkenntnissen gut umsetzbar, da keine größeren Einarbeitungszeiten und Programmierungen von Schnittstellen/Programmanbindungen nötig sind.

Die Administration wird vom Nutzer über die Benutzeroberfläche durchgeführt. Updates des gesamten Systems und Einbindungen von zusätzlichen Cookbooks müssen aber auf Serverseite durchgeführt werden. Die Verwendung der Cookbooks wird dann über einen Code in der Seite geregelt. Jedes Cookbook hat eine bestimmte Art und Weise wie diese angesprochen wird. Manche Cookbooks beeinflussen nur Seiten und manche können das ganze Verhalten des kompletten Wikis verändern.

1.1.3 Zusammenfassung

Tabelle 1 stellt einige Angaben zu Drupal und PmWiki gegenüber. Im Großen und Ganzen unterscheiden sich die beiden CMS nicht sehr stark. Der ausschlaggebende Punkt für die Wahl des CMS liegt in der Erweiterbarkeit. Hier hat PmWiki mit seiner relativ leichten und kleinen Entwicklungsumgebung einen Vorteil. Erweiterungen, bei PmWiki Cookbooks genannt, können ohne große Kenntnisse der PmWiki-Struktur erstellt und in das Wiki eingebaut werden. Somit können auch komplexere Cookbooks entstehen, ohne sich wie bei Drupal lange in die Struktur des CMS einarbeiten zu müssen. Daher wird für dieses Projekt das CMS PmWiki Verwendung finden.

Tabelle 1: Drupal versus PmWiki

	Drupal	**PmWiki**
Anzahl Module	Viele (> 22600)	Viele (etwa 400)
Anzahl Tempates	Viele (> 1700)	Viele (> 100)
Erweiterbar	Ja (schwierig)	Ja (leicht)
Programmiersprache	Vorwiegend PHP	Vorwiegend PHP
Lizenz	GNU GPL	GNU GPL
Aktuelle Version	Drupal 7	PmWiki 2.2.53
Systemvoraussetzungen	Apache oder Microsoft IIS PHP 5.2 und höher MySQL 5.0 oder höher oder PostgreSQL 8.3 oder höher oder SQLite	Irgendein Webserver PHP 5.2 (min. 4.3.x)
Security	Passwords User-Authentifikation SPAM-Protection (als Modul)	SPAM-Protection Passwords User-Authentifikation

1.2 Map-Server

Ein Map-Server ist ein Kartenserver, mit dem man benutzerspezifische und individuelle Karten im Internet bereitstellen kann (Geoinformatik-Service – Weblink). Viele Dienste bieten nicht nur die Veröffentlichung von Geodaten an, sondern auch das Editieren dieser. Hier wird meist auf die Standards des OGC (Open Geospatial Consortium) zurückgegriffen. Sie beschreiben den Austausch und die Art und Weise wie die Daten bereitgestellt werden können und sollten. Dazu gehören Dienste wie der Web Map Service und der Web Feature Service (OGC – Weblink).

1.2.1 GeoServer

Der GeoServer ist ein Open-Source-Software-Server, der auf der Basis von Java geschrieben wurde. Diese freie Server-Software läuft unter der GNU General Public Liecense (GPL). Weiterhin besitzt er Importe aus Softwareprodukten, die von der Apache Software Foundation veröffentlicht werden und unter der Apache License Version 2.0 und 1.1 stehen. Mit Hilfe des GeoServers wird dem Nutzer das Editieren und Veröffentlichen von Geodaten ermöglicht. Die Geodaten werden über Open-Standards der OGC zur Veröffentlichung zur Verfügung gestellt. Dazu gehören der Web-Feature-, der Web-Coverage- und der Web-Map-Service. Der GeoServer beinhaltet einige Features wie zum Beispiel die Anbindung an verschiedene Datenbanken, unteranderem MySQL und PostgreSQL. Die Installation des Servers ist relativ einfach, da sie unter Linux durch Einbinden eines WAR-Files in den Apache Tomcat schnell und unkompliziert durchgeführt werden kann (GeoServer – Weblink).

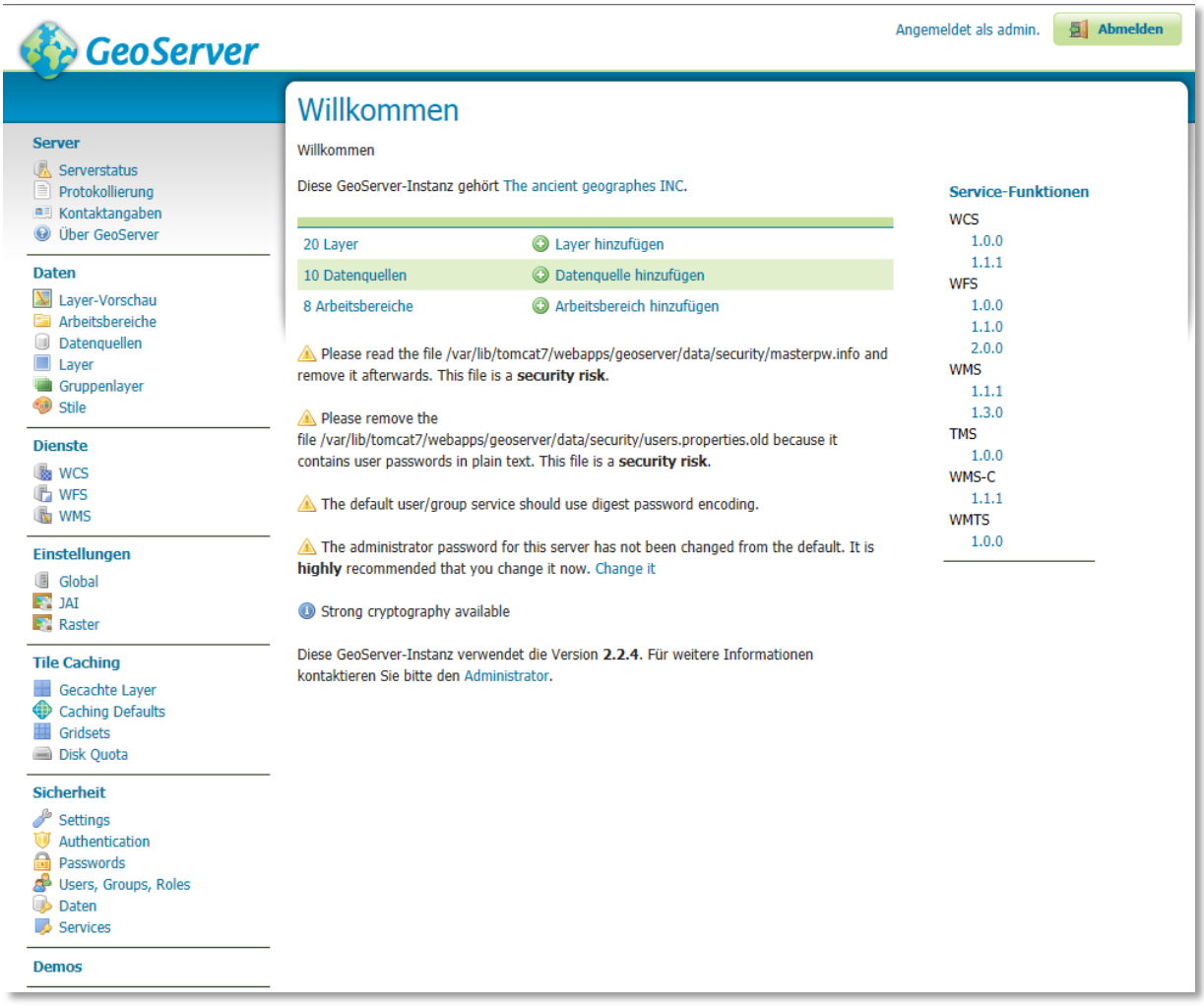

Abbildung 1: GeoServer

1.2.2 UMN MapServer

Der MapServer der University of Minnesota (UMN) ist eine Open-Source-Platform für die Veröffentlichung von Geodaten und eine interaktive Web-Mapping-Applikation. Dieser Server steht unter der MIT-style License. Das Projekt wird vom PSC (Project Steering Commitee) verwaltet und betreut. Es ist von der OSGeo (Open Source Geospatial Foundation) anerkannt und heute als Projekt der OSGeo verzeichnet. Der Server wird durch Entwickler auf der ganzen Welt gewartet und verbessert. Es werden auch die OGC-Standards wie WMS, WFS und WCS unterstützt. Weiterhin unterstützt der MapServer viele verschiedene Sprachen zur Erweiterung des Servers. Dazu gehören PHP, Python, Java und .NET. Weiterhin ist er auf Linux, Windows und weiteren Plattformen einsetzbar. Der UMN MapServer arbeitet auf der Basis eines Mapfiles. Dieses wird zur Konfiguration mit dem Server benötigt und bildet das Herzstück des ganzen Systems (MapServer 2013). Dies ist in Abbildung 2 verdeutlicht.

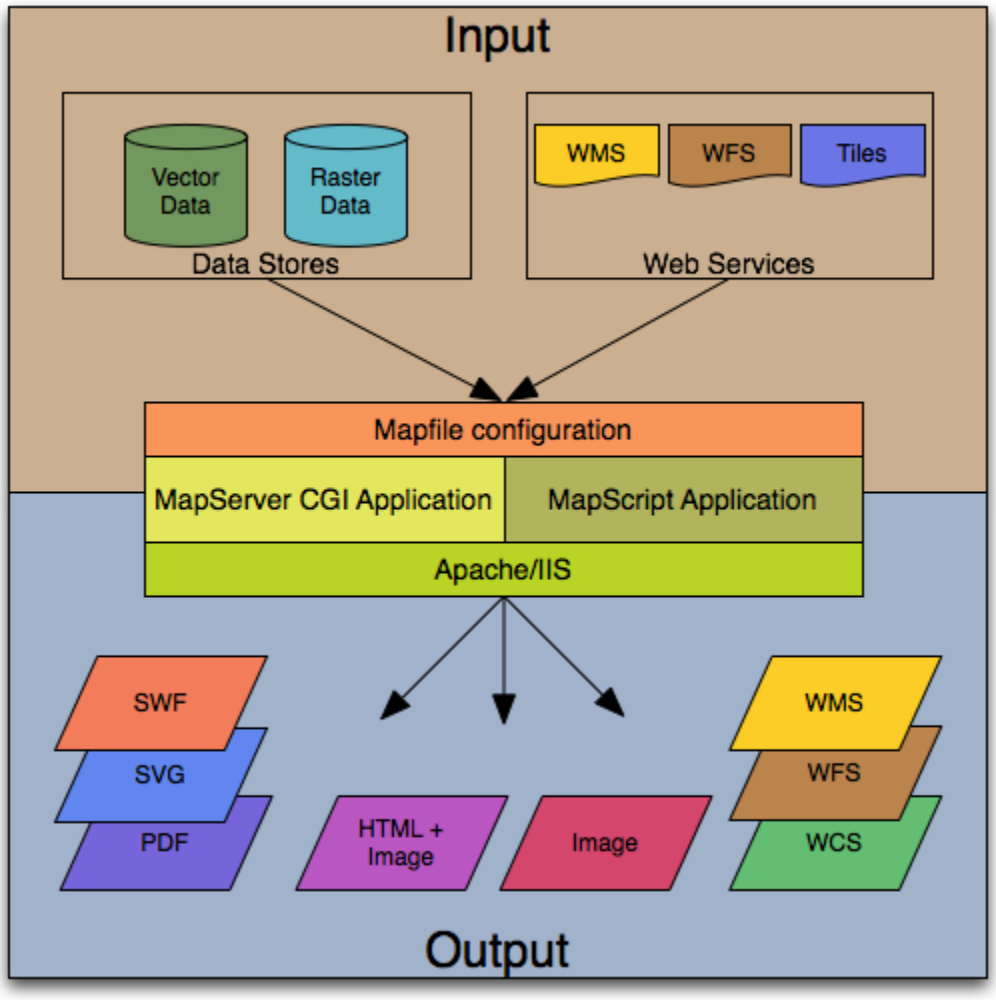

Abbildung 2: MapServer Input – Output (Quelle: MapServer 2013)

1.2.3 Zusammenfassung

Der UMN MapServer und der GeoServer sind zwei relativ ähneliche Produkte. Beide haben ihre Vor- und Nachteile. Ein Nachteil wird meist in der Verwendung des Map Files und der Verwendung von CGI gesehen. Der GeoServer bietet mit seiner Vielzahl an Webservices eine große Schnittstellenvielfalt. Am Ende fiel die Entscheidung auf den GeoServer.

Tabelle 2: GeoServer versus UMN Mapserver

	UMN Mapserver	GeoServer
Lizenz	MIT-style Lizenz	GNU GPL Version 2.0
Webservices	WMS, WFS, WMC, WCS, Filter Encoding, SLD, GML, SOS, OM	WMS, WFS, WCS, WPS, WFS-T, OWS, REST (Ausgabe jeweils mit vielen verschiedenen Datenformaten)
Unterstützte Programmiersprachen	PHP, Python, Perl, Ruby, Java, .NET	PHP, Python, Java, Ruby, cURL
Schnittstelle für die Ausführung	CGI	J2EE
Mindestbestandteile	Map File Geographische Daten HTML-Seiten MapServer CGI Web/HTTP Server	OpenLayers JRE Apache Tomcat
Systemvoraussetzungen	Web-(HTTP)-Server	Apache Tomcat Oracle Java SE 6 oder neuer
Datenformate	ESRI Shape-File PostgreSQL-PostGIS etc. GML MapInfo etc.	ESRI Shape-File PostGIS, Oracle etc. VPF, MYSQL, MapInfo Cascading WFS GeoTIFF, Image Pyramides etc. JPEG2000, GML etc.
Projektionen	On-the-fly (Proj.4)	Über OpenLayers

1.3 Datenbanken

Es gibt viele verschiedene Datenbanksysteme. Sie werden teilweise nur für spezielle Anwendungen genutzt. An dieser Stelle werden zwei der am häufigsten genutzten und freien Open-Source-Lösungen dargestellt.

1.3.1 MySQL-Datenbank

MySQL ist eine freie Open-Source Datenbank, die von der Oracle Corporation vertrieben wird. MySQL ist sowohl frei unter der GNU-Lizenz zu bekommen als auch unter kommerzieller Lizenz (MySQL – Weblink).

Diese Datenbank ist weltweit sehr verbreitet und man kann sagen: „Jeder kennt MySQL“. Sie ist eine relationale Datenbank, läuft unter vielen Betriebssystemen und ist recht einfach zu pflegen. MySQL wird häufig für Webservices verwendet und daher auch im Zusammenhang mit PHP (MySQL – Weblink). Die SQL-Anweisungen sind einfach handhabbar. Die Datenbank an sich kann über verschiedene Systeme, wie zum Beispiel phpMyAdmin, extern verwaltet und bearbeitet werden (phpMyAdmin – Weblink). MySQL an sich hat eine relativ hohe Geschwindigkeit und einen geringen Speicherbedarf. Durch Installation vieler Erweiterungen kann das Gegenteil eintreten, da diese entweder nicht zusammenpassen oder sie nicht alle Fähigkeiten der MySQL-Datenbank (PHP-FAQ – Weblink) unterstützen.

Die Speicherung von räumlichen Daten ist über eine Erweiterung möglich. Diese Erweiterung arbeitet auf Basis des OpenGIS-Geometriemodells. In diesem werden Geometrie-Typen definiert wie Point, Curve und LineString. Die Geometrie-Typen wurden nach dem Vorschlag der OGC erstellt und beruhen auf dem OpenGIS Geometry Model (MySQL – Weblink).

1.3.2 PostgreSQL-Datenbank

PostgreSQL ist ein Open-Source und objektrelationales Datenbanksystem. PostgreSQL setzt auf Zuverlässigkeit, Datenintegrität und Korrektheit des Produktes. Es läuft auf vielen verschiedenen Betriebssystemen wie Linux und Windows. Weiterhin ist es ACID-kompatibel, was bedeutet, dass JOINS, VIEWS, FOREIGN KEY und vieles mehr unterstützt wird. PostgreSQL unterstützt zusätzlich fast alle SQL:2008 Datentypen und kann über verschiedene native Programmierschnittstellen erweitert werden. Programmiersprachen wären zum Beispiel C/C++, Java, .NET, Python und viele mehr (PostgreSQL – Weblink).

PostgreSQL besitzt eine kleine Menge nützlicher Erweiterungen. Diese können in verschiedensten Programmiersprachen verfasst werden. Dies ist eine Besonderheit von PostgreSQL. Die Dokumentation der Datenbank ist in einem Handbuch zusammengefasst und es gibt auch eine kleine Community, die umfangreiche Hilfestellung gibt. Es gibt zwei gute Tools, die für die Administration von PostgreSQL verwendet werden können: einerseits phpPgAdmin und anderseits pgAdmin. PostgreSQL ist umfangreich und hat umfangreiche Möglichkeiten. Bei großer Last übertrifft das heutige PostgreSQL häufig sogar die schnelle MySQL-Datenbank (PHP-FAQ – Weblink).

Die Erweiterung zur Nutzung von räumlichen Daten in einer PostgreSQL-Datenbank ist PostGIS. PostGIS ermöglicht es, geographische Objekte mit Hilfe von ortsbezogenen Abfragen in SQL zu erhalten. Dazu gehören viele verschiedene Funktionen wie Analysefunktionen auf Raster- und Vektordaten, Raster-Map-Algebra, Import und Export von Shape-Files (PostGIS – Weblink). Weitere Funktionen, die PostGIS beinhaltet, sind auf http://www.postgis.net beschrieben.

1.3.3 Zusammenfassung

Grundsätzlich sind die beiden Datenbanksysteme relativ gleich stark. Was die eine nicht kann, kann die andere durch andere Features ausgleichen. Ausschlaggebend sind eher die GIS-Implementierung und die Verwendung von XML. Über PostGIS kann PostgreSQL relativ schnell und einfach Geodaten verarbeiten. Da noch nicht klar ist, ob vielleicht doch einige Teile in XML in der Datenbank vorgehalten werden, ist auch dies ein Punkt, der im Vorhinein beachtet werden muss. Tabelle 3 zeigt einen Ausschnitt aus der Leitungsfähigkeit der beiden Datenbanksysteme.

Tabelle 3: MySQL versus PostgreSQL

	MySQL	**PostgreSQL**
Lizenz	GPL, Proprietär	ProgreSQL
Architektur	Relationales DB-Modell	Objektrealtionales DB-Modell
Interface	SQL	GUI, SQL
Access Control	Native Network Encryption, Patch Access, Run Unprivileged	Audit, Brute-force Protection, Enterprise Directory Compatibility, Native Network Encryption, Password Complexity Rules etc.
Indizes	Full-text, Hash, R-/R+ Baum	Bitmap, Expression, Full-text, GIN, GiST, Hash, Partial, R-/R+ Baum, Reverse
Andere Objekte	Externe Routinen, Funktionen, Trigger etc.	Data Domain, externe Routinen, Funktionen, Trigger etc.
Datenbank Funktionen	Blobs und Clobs, Inner Joins, Inner Selects, Merge Joins, Outer Joins, Union	Blobs and Clobs, Common Table Expression, Except, Inner Joins, Inner Selects, Intersect, Merge Joins, Outer Joins, Parallel Query, Union, Windowing Funktionen
Besondere Datentypen	Datetime, GIS Datentypen, Set	Numeric, Character, Character varying, Interval, GIS Datentypen, Polygon, XML etc.
Und vieles mehr … (vgl. FindTheBest – Weblink))		

1.4 Weitere Software

An dieser Stelle werden weitere Softwareprodukte aufgezeigt. Sie müssen in Verbindung mit den schon ausgewählten Tools verwendet werden oder müssen aus Gründen von Softwarevoraussetzungen verwendet werden.

1.4.1 OpenLayers

OpenLayers ermöglicht es, dynamische Karten auf Webseiten einzubinden. Es können Karten-Tiles, Marker und einiges mehr angezeigt werden. Diese können über Webservices abgefragt werden oder von anderen Datenquellen stammen, wie zum Beispiel KML-Dateien oder Shape-Files. OpenLayers ist eine freie Open-Source JavaScript-basierende Bibliothek, die unter 2-clause BSD-License (FreeBSD) steht (OpenLayers – Weblink).

1.4.2 Apache Webserver

Als Webserver kommen zwei Apache-Produkte in Frage. Einmal der Apache Tomcat, eine Open-Source-Software Implementierung auf Java-Servlets und Java-Server-Pages-Technologien sowie der HTTP-Server von Apache, ein Open-Source-HTTP-Server (Apache Projekte – Weblink).

Der Apache Tomcat ist ein Webserver und steht unter der Apache Lizenz Version 2. Er soll in Zusammenarbeit von Best-of-Breed-Entwicklern aus der ganzen Welt entwickelt werden. Die Java-Server-Spezifikationen werden unter dem Java-Community-Prozess entwickelt (Apache Tomcat – Weblink).

Der Apache HTTP-Server hat das Ziel, einen sicheren, effizienten und erweiterbaren Server bereitzustellen. Er soll HTTP-Dienste bieten, die unter den aktuellen HTTP-Standards laufen. Die Popularität des Apache Webservers im Internet ist seit April 1996 sehr hoch (Apache HTTP-Server – Weblink).

Auf Grund der Wahl des GeoServers als Mapserver wird als HTTP-Server der Apache Tomcat verwendet. Die Einbindung des GeoServers ist sehr einfach, da nur durch Hinzufügen einer WAR-Datei der GeoServer im Apache Tomcat installiert wird (GeoServer Installation – Weblink). Weiterhin wird der Apache HTTP-Server verwendet, da über den Tomcat keine PHP-Anwendungen laufen und dieser nur mit einigen Änderungen auf dem Port 80 erreichbar gemacht werden kann. Somit fungiert der Apache HTTP-Server als Schnittstelle zwischen PHP-Anwendungen und Tomcat-Anwendungen.

2 Systemarchitektur

Anhand der evaluierten Systeme wird an dieser Stelle die Systemarchitektur beschrieben. In diesem Abschnitt wird es um das Datenbanksystem, die Webserver und das gewählte CMS gehen. Weiterhin werden die Gründe für die Auswahl einzelner Bestandteile nochmals zusammengefasst und herausgestellt. Zusätzlich geht es um das Zusammenspiel der verschiedenen Komponenten und deren Platz im Verarbeitungsprozess.

Abbildung 3 zeigt die Systemarchitektur. Als Mittelpunkt des Ganzen steht der Webserver. Als Betriebssystem wird LINUX verwendet. Dieser Server stellt die Schnittstelle zwischen dem Nutzer und der Webseite dar. Als Webseitengrundsystem wird das CMS PMWIKI verwendet. Dieses ist über Schnittstellen mit der Datenbank und dem GEOSERVER verbunden. Der GEOSERVER selbst wird auch eine Verbindung zur Datenbank haben, da die Vektordaten auch in der verwendeten PostgreSQL-Datenbank vorgehalten werden.

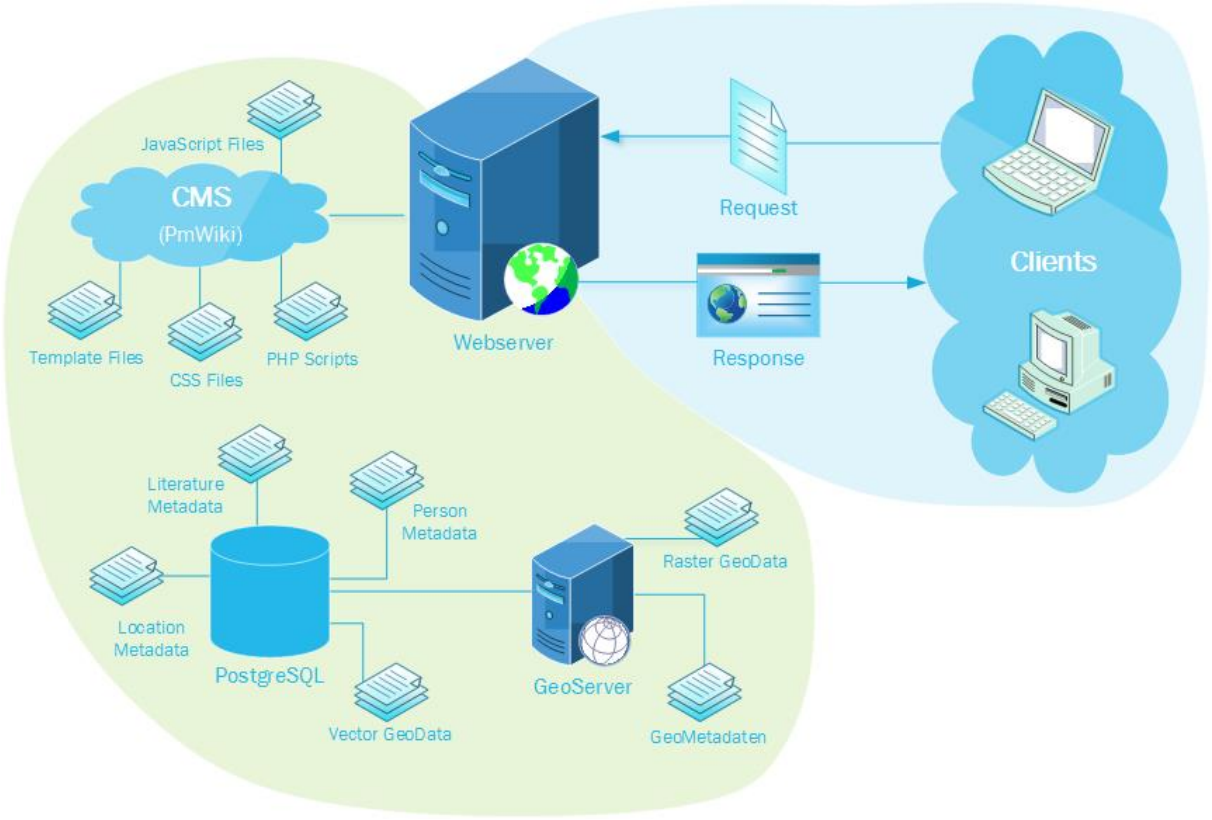

Abbildung 3: Systemarchitektur (Client-Server-Struktur)

2.1 Webserver

Der Webserver besteht aus mehreren Teilen, die an dieser Stelle näher beschrieben werden. Zum Webserver gehören der Mapserver, das Content Management System und der HTTP-Webserver.

Der **GeoServer** ist der Mapserver, der in diesem Projekt Verwendung findet. Er wird die Rasterdaten vorhalten und über eine Schnittstelle mit den Vektordaten in der Datenbank verbunden sein. Über Abfragen werden verschiedenste Layer für das CMS zur Verfügung gestellt.

Das **Content Management System** bildet die Nutzerschnittstelle. Sie gibt das Webseitendesign wieder und ist für den Nutzer die Plattform zur Interaktion mit den Daten. Über verschiedenste Abfragen werden Daten aus der Datenbank und dem Geoserver zur Verfügung gestellt. Für das GeoBib-Projekt wird PMWIKI verwendet, da die Designstruktur der Webseite leicht über Templates abbildbar ist. Des Weiteren können zusätzliche Plug-ins (Cookbooks) relativ einfach über eine Konfigurationsdatei eingebunden werden. Somit ist die Erweiterung des CMS für das GeoBib-Projekt im Gegensatz zu DRUPAL unkomplizierter, da nur die Programmiersprache PHP beherrscht werden muss und keine Developer-Struktur, wie bei DRUPAL.

Der **Webserver** bildet das Grundsystem der Webserverstruktur. Er wird für die Kommunikation zwischen Internet und Webseite genutzt. Weiterhin laufen die Datenbank und der GEOSERVER über diesen HTTP-Webserver. Abbildung 4 gibt diesen Aufbau wieder. Es wird sichtbar, dass alle *Requests* und *Responses* über den HTTP-Webserver laufen. Nur Beispiel 3 bildet eine Ausnahme. Hier wird angenommen, dass der GEOSERVER auch auf einem anderen Webserver liegt. Da JavaScript direkt im Browser verarbeitet wird, gibt es keine Verbindung zum HTTP-Webserver.

Es wird bei der Verwendung des Webservers eine Kombination von Apache Tomcat und HTTP-Apache-Server genutzt. Der HTTP-Apache-Server ist an dieser Stelle für die Kommunikation aller Anfragen, die über das Web gestellt werden, zuständig. Der Apache Tomcat hingegen wird alleine für den GeoServer und die PostgreSQL-Datenbank betrieben.

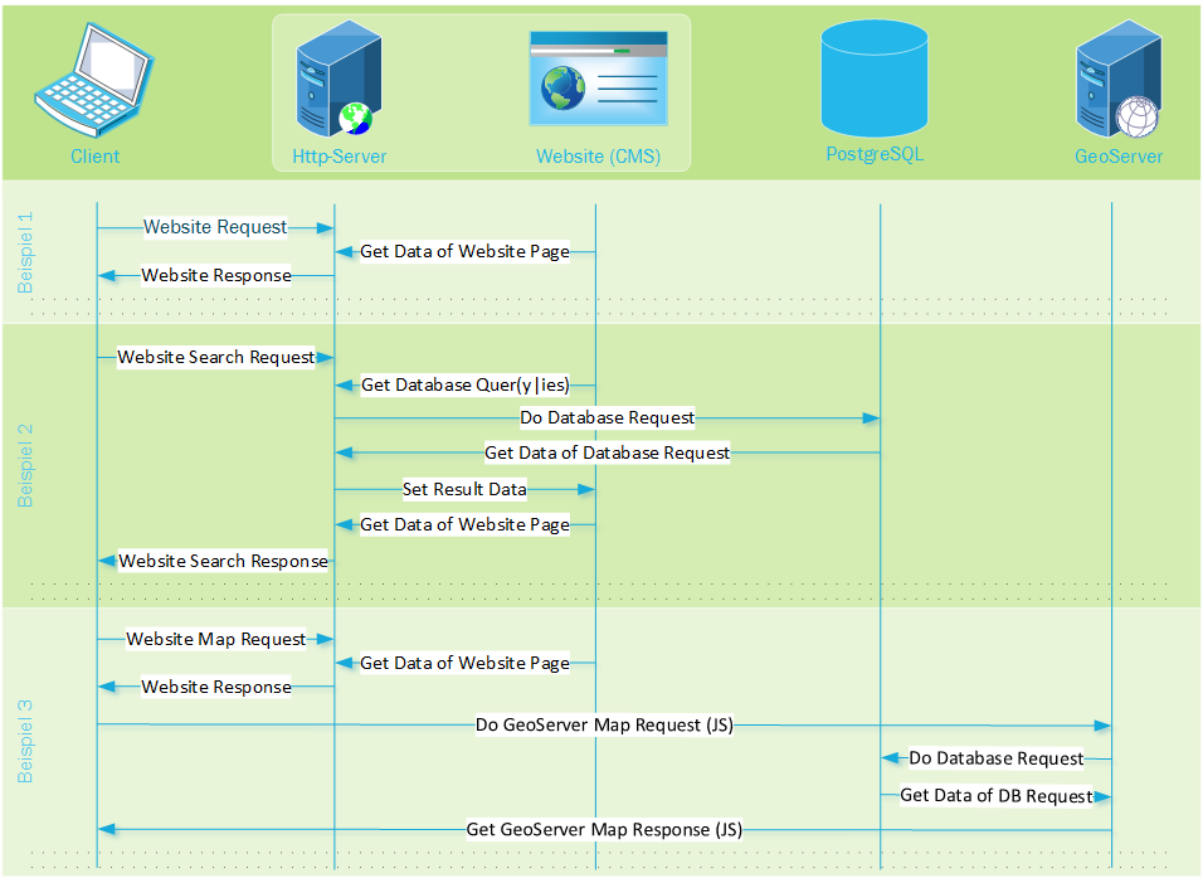

Abbildung 4: Request-Response-Modell (Sequenzdiagramm)

An dieser Stelle wird die Aufteilung in **interne und externe Struktur** nicht beschrieben. Interne Strukturen sind alle Verbindungen, die nur lokal laufen, also auf *Localhost* liegen. Externe sind dagegen Verbindungen ins World Wide Web. Sie können erst im fortgeschrittenen Entwicklungsprozess abgebildet werden, da für diese Strukturen Tests ausgeführt werden müssen und diese noch nicht durchgeführt werden können.

2.2 Datenbank

Das Datenbankdesign wird auf einer **PostgreSQL**-Datenbank aufgesetzt. PostgreSQL ist für die Umsetzung der im GeoBib-Projekt verwendeten Daten gut nutzbar, da mit der Erweiterung von **PostGIS** auch geographische Sachverhalte und Daten abgebildet werden können.

Weiterhin ist in einer PostgreSQL-Datenbank die Verwendung von referentieller Integrität, Regeln und der Volltextsuche möglich. Dieses sollte neben der Einhaltung der ersten, zweiten und dritten Normalform[1] in der Struktur des Datenbankdesigns Verwendung finden.

Die PostgeSQL-Datenbank kann weiterhin über verschiedene Applikationen administriert werden. Neben phpPgAdmin und pgAdmin III kann sie auch über das freie GIS QUANTUMGIS mit Daten gespeist werden. Weiterhin ist die Verbindung von Datenbank und GEOSERVER ohne Erweiterung dessen möglich.

Im Gegensatz zu MySQL, welches sehr häufig im Web Verwendung findet, müssten einige Erweiterungen eingespielt werden. Dies kann zu Problem mit dem Handling und der Performanz führen, da nicht alle Erweiterungen immer für genau die gerade verwendete Version zur Verfügung stehen. Um dies zu vermeiden, wird von vornherein auf die PostgreSQL-Datenbank gesetzt. Weiterhin werden bei MySQL alle Vektordaten in eine Tabelle gespeichert, das lässt die Datenbank zwar schmaler aussehen und der Zugriff könnte schneller sein, aber die Übersicht geht verloren. Weiterhin werden so die referentielle Integrität und die Normalformen schnell verletzt (PHP-FAQ – Weblink, FindTheBest – Weblink).

Im GeoBib-Projekt bildet die Datenbank das Herz der Datenhaltung. In ihr werden alle Daten aus den Bibliographien und die Vektordaten vorgehalten. Durch Vernetzung von Vektor- und Literatur-Metadaten soll die Abfragestruktur vereinfacht werden. Die Datenbank wird über verschiedene Schnittstellen vom CMS und vom GeoServer genutzt werden. Da PostgreSQL verwendet wird, handelt es sich bei dem Datenbankmodell um ein objekt-relationales Datenbankmodell. Dieses beinhaltet zusätzlich zum relationalen Datenmodell Konzepte der Objektorientierung, dazu gehören Methoden der Vererbung und benutzerdefinierte Datentypen (Wiki FH Köln – Weblink). Dies soll auch bei dem Aufbau des Datenbankdesigns beachtet werden.

[1] 1. Normalform: der Wertebereich aller Attribute ist atomar 2. Normalform: 1. NF + Attribute A ist von Attribute B (funktional) abhängig 3. Normalform: 2. NF + Ist A ein Oberschlüssel, wenn B kein primes Attribut ist (Quelle: http://www.computerlexikon.com/begriff-normalform?highlight=Normalform)

2.3 Content Management System

Als Grundlage für die einzelnen Seiten wird das PmWiki-CMS verwendet, da es ohne große Einarbeitungszeit auf die Bedürfnisse des GeoBib-Projektes angepasst werden kann. Dies gilt nicht nur für die Funktionalität, sondern auch für die Erstellung des Designs.

Das PmWiki wird die Webseite repräsentieren und über Cookbooks eine Anbindung an die Datenbank und den GeoServer bekommen. Das PmWiki selbst wird im Webserverbereich eingebunden werden. Die Cookbook-Struktur und deren Verbindungen zu anderen Systembestandteilen wird im Systemdesign näher erläutert (Kapitel 0). Das CMS bildet mit dem Webserver somit das Zentrum der gesamten Serverarchitektur. Das PmWiki selbst besitzt eine relativ feste Ordnerstruktur (Abbildung 5). Die selbst erstellten Cookbooks werden unter dem Ordner „cookbook“ gespeichert und in die Konfigurationsdatei eingebunden. Alle Cookbooks, die von PmWiki.org installiert werden, kommen auch in diesen Ordner. Das eigene Template wird unter „pub/skins“ abgelegt werden.

PmWiki-Ordnerstruktur	Beschreibung
local	Konfigurationsscripte
cookbook	Recipes/Add-ons etc.
docs	Dokumentation
pub	öffentliche Dateien
pub/css	CSS-Systlesheet-Dateien
pub/guiedit	
pub/skins	Layout-Templates
scripts	PmWiki-Scripts
wikilib.d	Default-PmWiki-Seiten

Abbildung 5: PmWiki-Ordnerstruktur

3 Systemdesign

In diesem Abschnitt werden die einzelnen Designstrukturen von Datenbank, GEOSERVER, Webserver und PmWiki genauer beschrieben. Dazu gehören der allgemeine Aufbau und die Struktur in der die späteren Daten eingebunden werden sollen. Weiterhin werden schon im Vorfeld bekannte Regeln und Beziehungen festgelegt und beschrieben.

3.1 Datenbank

Das **Datenbankdesign** ist für die Inhalte der **Literatur-Metadaten** schon erfasst. Dies zeigt Abbildung 6 im Detail. Es sind mehrere Gruppen von Entitäten[2] zu unterscheiden. Zum einen die Haupt-Entitäten, welche das Grundgerüst des Designs bilden und zum anderen mehrere Unter-Entitäten und Eigenschaftsdefinitionen, die zur Verfeinerung und Verknüpfung der Haupt-Entitäten benötigt werden. Unter Eigenschaften einer Entität werden die Zeilen der Entität genannt, die diese beschreiben wie FIRSTNAME und LASTNAME in der Entität PERSON. In den jeweiligen Entitäten wurde besonders auf die Einhaltung der ersten bis dritten Normalform geachtet. Weiterhin ist die Einhaltung der Datenkonsitenz/-integrität und der referentiellen Integrität[3] im Datenbankdesign ein Kriterium für die Strukturierung. Die jeweiligen Relationen zwischen den Entitäten und die Fremdschlüsselverbindungen werden in Tabelle 4 gezeigt.

Die **Haupt-Entitäten** sind in diesem Datenbankdesign PERSON, GENRE, LANGUAGE, ANNOTATOR, WORK und PLACE. Sie bilden die wesentlichen Eigenschaften der jeweiligen Einträge ab. Jede Entität hat einen Primärschlüssel[4], welcher einen eindeutigen Wert besitzt.

Die **Unter-Entitäten** bestehen aus verschiedenen Entität-Typen. Als erstes gibt es die **Zwischentabellen**. Sie verbinden Haupt-Entitäten miteinander. Die Inhalte bestehen nur aus Primärschlüsseln und Fremdschlüsseln[5]. Der Name der Tabelle beschreibt die Verbindung zwischen den Haupt-Entitäten, wie beispielsweise PERSON_CATEGORIES, welche eine Verbindung zwischen den Entitäten PERSON und CATEGORY ist und somit

[2] Engl. Entity: Ein eindeutiges, individuelles und identifizierbares Objekt, über das Informationen gespeichert werden. (Quelle: http://www.computerlexikon.com/begriff-entity-relationship-model?highlight=entity)
[3] http://www.informatikzentrale.de/referentielle-integritaet.html
[4] Engl. Primarykey: Datenfeld, das in einer Tabelle die Datensätze eindeutig beschreibt. (Quelle: http://www.computerlexikon.com/begriff-prim%C3%A4rschl%C3%BCssel)
[5] Engl. Foreignkey: Ist ein Attribut, das auf einen anderen Datensatz in einer anderen Relation verweist. (Quelle: http://www.informatikzentrale.de/fremdschluessel.html)

die Kategorien, die eine Person sein kann abbildet. Zwischentabellen werden somit verwendet, wenn, wie in diesem Beispiel, eine Person mehreren Kategorien zugeordnet werden kann. Eine weitere Unter-Entität sind **Zusatztabellen**, die einen bestimmten Eintrag einer Haupt-Entität erweitern. Die kann man beispielsweise bei der Entität ALT_WORK_NAMES sehen. Sie besitzt als Primärschlüssel die ID eines WORK-Eintrags und ein weiteres Feld NAME. Beide bilden den Primärschlüssel, wobei NAME kein Fremdschlüssel ist, da es nur eine Erweiterung des Namens des WORK-Eintrags angibt. Als nächstes gibt es eine Entität, die für die **Vererbung** verwendet wird. AUTHOR_EDITOR beschreibt eine Spezialisierung von PERSON. Eine PERSON kann somit ein AUTOR oder ein EDITOR sein. Beide Spezialisierungen besitzen alle Felder unter PERSON und unter AUTOR_EDITOR. Eine Person, die weder ein Autor noch ein Editor eines Werkes ist, hat hingeben nur die Felder aus der Entität PERSON.

Eigenschaftsdefinitionen können Enumerations (Aufzählungen) und Arrays (Auflistungen) sein. Sie hängen an speziellen Entität-Eigenschaften wie SEX der Entität PERSON und NAME der Entität CATEGORY. Sie sind somit eine feste Auswahl an Daten, die ein Eintrag an dieser Stelle haben kann. Der Unterschied zwischen Enumerations und Arrays ist die Erweiterbarkeit. Enumerations können und sollten nach Anlegen nie erweitert werden, daher wird dies nur für das Geschlecht einer Person verwendet. Kategorien und Genre können beliebig verändert und erweitert werden und werden somit als Arrays vorgehalten.

Die **Entität PLACE** besitzt zusätzlich eine besondere Funktion. Sie wird die Verbindung zwischen den Geodaten und den Literaturdaten darstellen. Wie genau diese Verbindung später abgebildet wird, wird erst im nächsten Meilenstein (M6.4R) beschrieben, da bis zu diesem Zeitpunkt noch keine Verbindung der Daten auf Datenbankebene erstellt und definiert wurde.

Tabelle 4: Kreuztabelle - Relationen & Fremdschlüssel der Entitäten

	person	person_category	person_places	alt_person_names	category	work	work_author	work_editors	work_annotators	work _places	workpublication_places	work_languages	alt_work_names	annotator	languages	genre	place	publisher
person		x	x	x													o	
person_category	o				o													
person_places	o																o	
alt_person_names	o																	
category		x																
work							x	x	x	x	x	x	x			o		o
work_author	o					o												
work_editors	o					o												
work_annotators						o								o				
work _places						o											o	
workpublication_places						o											o	
work_languages						o									o			
alt_work_names						o												
annotator									x									
languages												x						
genre						x												
place	x		x							x	x							
publisher						x												

Legende

Relation: x

Fremdschlüssel: o

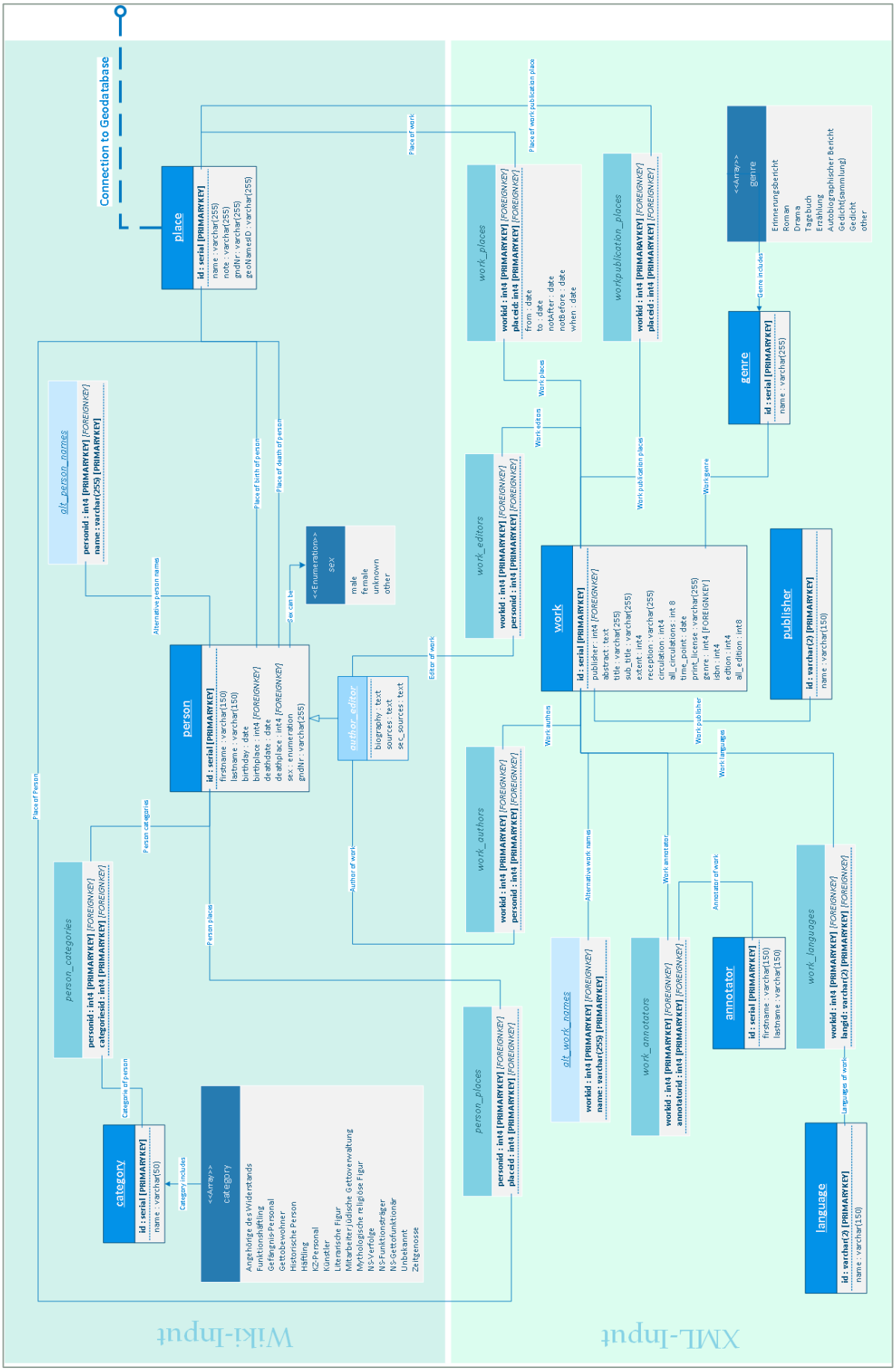

Abbildung 6: ER-Diagramm – Literatur- und Personen-Metadaten

3.2 Webserver

Der Webserver an sich bildet das Herzstück des ganzen Systems. Er beinhaltet die Hauptkomponenten des Projektes. Durch die Apache-Webserver, das CMS und die PostgreSQL-Datenbank werden alle Abfragen bezüglich der Nutzerabfragen verarbeitet. Das CMS, PmWiki, wird verschiedene Cookbooks bekommen, die die speziellen Such- und Seitenanfragen auf der GeoBib-Webseite verarbeiten und wiedergeben. An dieser Stelle wird aber nicht auf die Cookbooks eingegangen, sondern anhand einer Grafik ein grober Arbeitsablauf aufgezeigt. Dieser Ablauf ergibt sich aus dem Aufbau des Webservers. Abbildung 7 stellt diesen Ablauf dar.

Anfragen auf der Webseite gehen generell über die PostgreSQL-Datenbank bzw. die PmWiki-Pages holen sich den Inhalt für ihre Seiten aus dieser. Das Ergebnis wird dann auf die Art der Ausgabe geprüft. Sobald eine Karte ausgegeben werden soll, werden die Daten aus der Datenbank über spezielle Karten-Cookbooks verarbeitet. Gleiches gilt für die Textausgabe, die über Template-Cookbook-Ausgaben laufen. Für die Kartenausgabe wird OPENLAYERS verwendet.

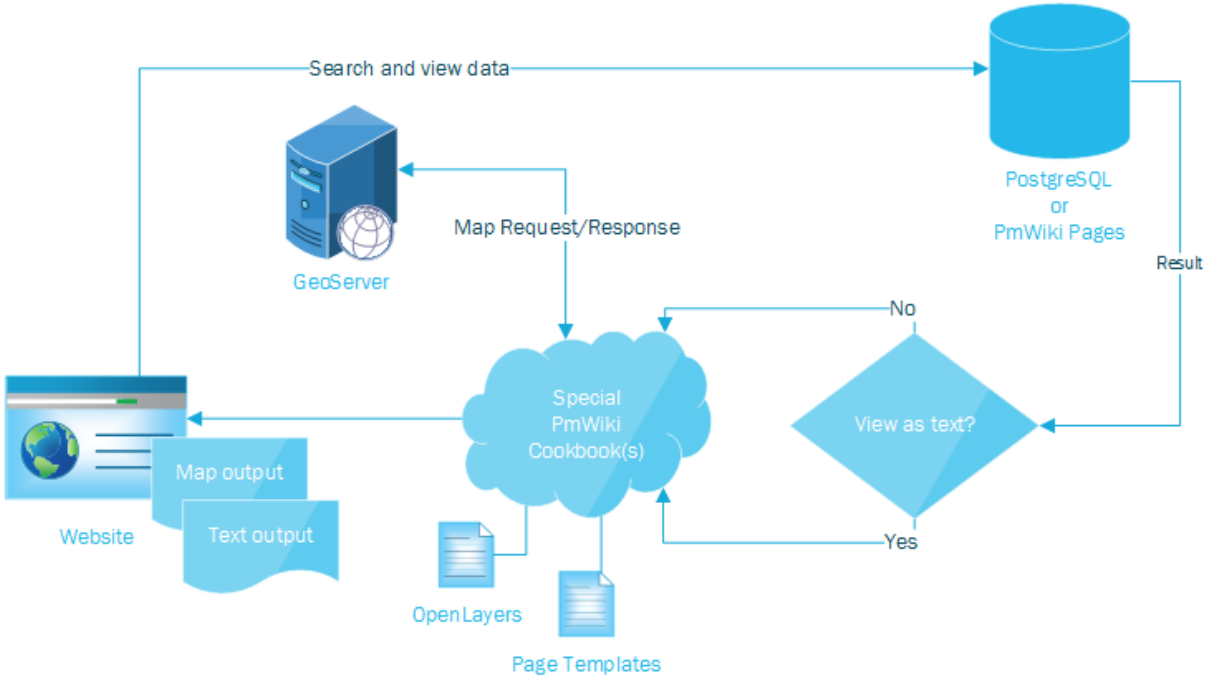

Abbildung 7: Ablaufstruktur Webseite und Datenbank/PmWiki-Seiten

3.3 PmWiki

Im PmWiki gibt es keine grundsätzlichen Designstrukturen, die beschrieben werden müssen. Wichtiger sind die Strukturen der Cookbooks. Sie gliedern sich in drei Kategorien auf: die Suche, die Literatur-Metadaten und OpenLayers (Abbildung 5). Die Cookbooks sollten so strukturiert werden, dass sie auch außerhalb des Projektes in einem anderen PmWiki Verwendung finden können. Daher ist eine konkrete Mark-up-Struktur zu entwickeln. Diese wird an dieser Stelle aber nicht aufgezeigt, sondern in einer Weiterführung dieses Meilensteins beschrieben (M6.4R). Dazu gehört auch die Beschreibung der Template-Struktur des GeoBib-Designs. Sie wird nach den Strukturen, die in M2.1R beschrieben wurden, erstellt.

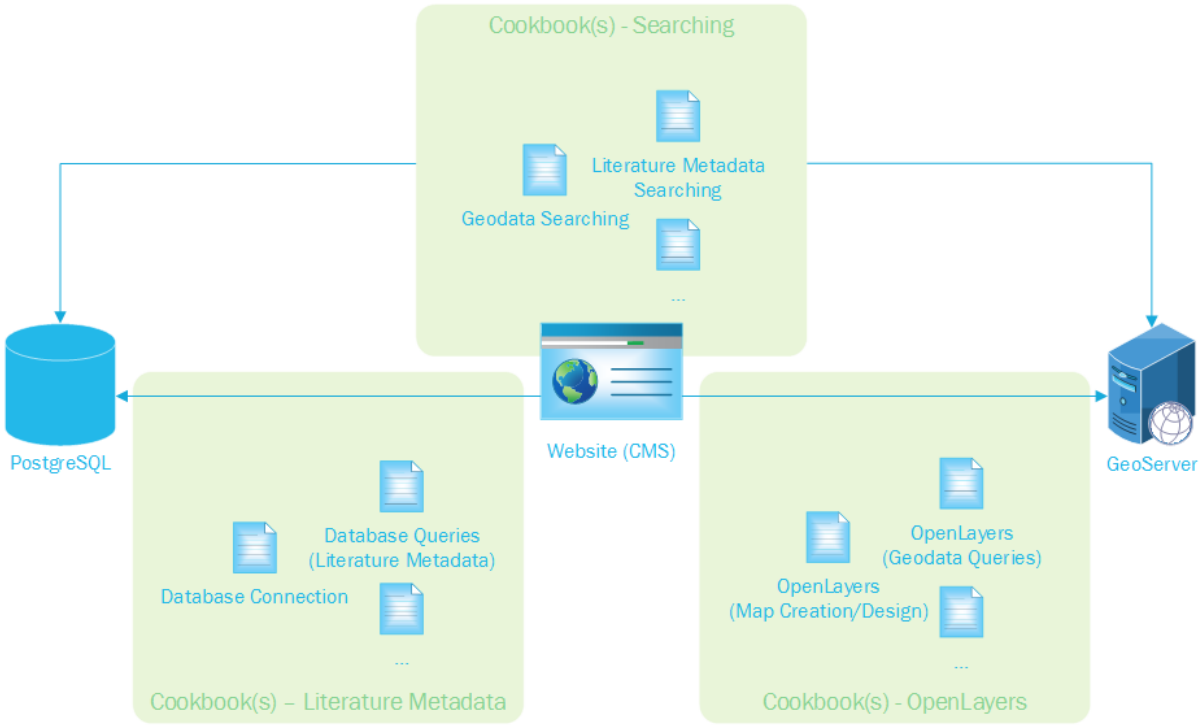

Abbildung 8: PmWiki-Cookbook-Struktur

3.4 GeoServer

Der GeoServer wird grundsätzlich nach der Installation nicht verändert. Unter dem Datenordner wird lediglich eine Dateistruktur für die einzelnen Datenpakete angelegt. Dazu gehören Metadaten, die die Raster- und Vektordaten beschreiben, und die Rasterdaten selbst. Die Rasterdaten werden nach einem bestimmten Schema benannt und gruppenweise in Unterordner abgelegt. Die Vektordaten werden in der PostgreSQL-Datenbank abgelegt. Jedes Shape wird in einer Tabelle hinterlegt. Die einzelnen Shape-Bestandteile bilden jeweils einzelne Zeilen der Tabelle. Somit entsteht eine relativ große Kollektion an Tabellen. Auch hier wird auf die gleiche Namensvergabe wie bei den Rasterdaten zurückgegriffen. Eine genaue Vorgabe kann aber erst im Laufe des Projektes beschrieben werden, da noch nicht klar ist, welche Vektor- und Rasterdaten am Ende genutzt werden (M5.2R) und der Name eine eindeutige und rückführbare Kennung sein soll.

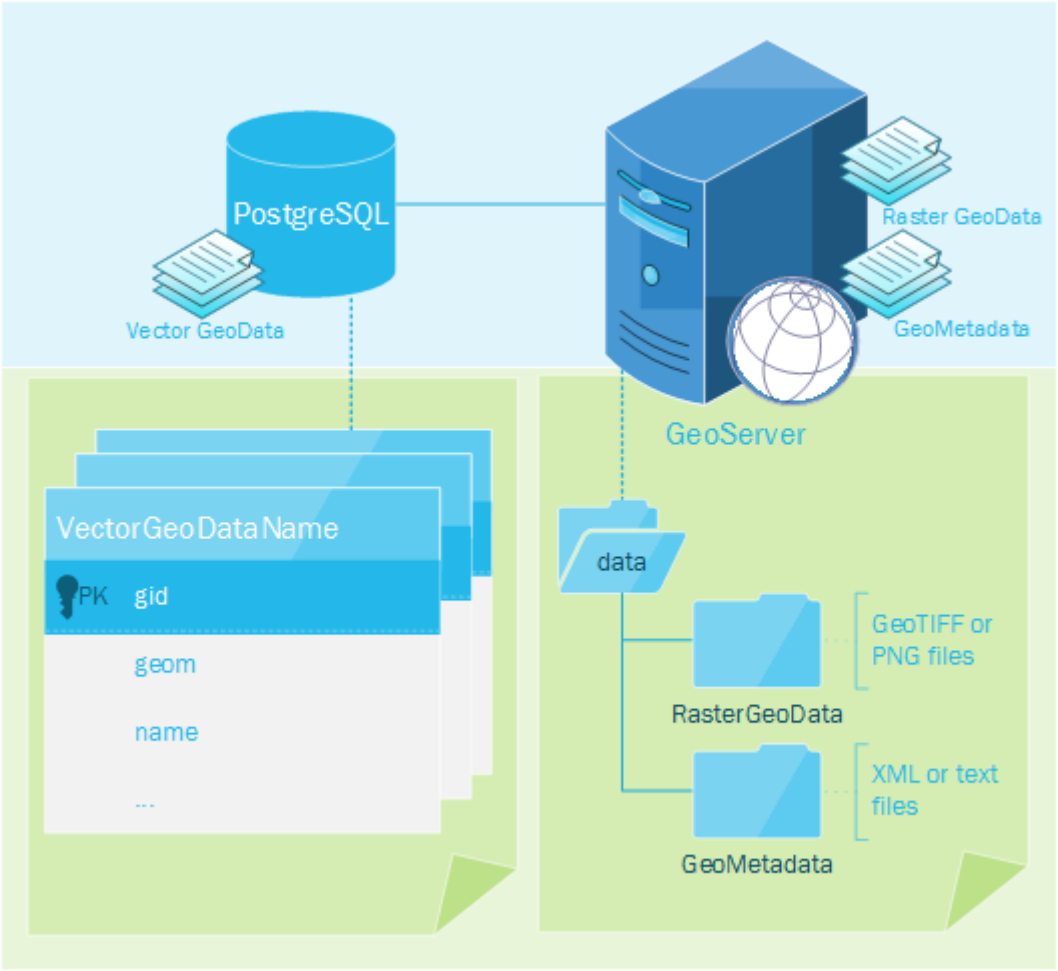

Abbildung 9: GeoServer-Design

4 Verarbeitungspipeline

In diesem Kapitel werden einzelne Verarbeitungsschritte zu den Punkten Literartur-Metadaten und Geodaten beschrieben. Es geht um den Ablauf zur Verarbeitung der Daten und darum, welche Prozesse unter bestimmten Bedingungen angestoßen werden müssen.

4.1 Literatur-Metadaten

Um dem Endnutzer des GeoBib-Systems die Daten in der gewünschten Form aufbereiten zu können, durchlaufen die Informationen, die aus den Texten und zu den Texten gesammelt werden, mehrere Verarbeitungsschritte, die schematisch in Abbildung 10 dargestellt sind.

Aus den relevanten Texten, die im Rahmen des Projekts gesammelt und bearbeitet werden, werden sowohl bibliographische als auch inhaltliche Informationen, zum Beispiel zum Autor, der Handlung und den handlungsrelevanten Orten, extrahiert und um Ergebnisse von historischen und literaturwissenschaftlichen Recherchen ergänzt (M3.1R).

Die Daten, die den Text direkt betreffen, werden in jeweils einer XML-Datei gespeichert und auf einem Server vorgehalten, während die Informationen zu Personen und Orts-Entitäten in einem separaten Wiki gesammelt und gespeichert werden (M4.1S). In einem nächsten Arbeitsschritt müssen die hier aus Gründen des einfacheren Umgang mit den Daten und des Workflows vorher getrennt gehaltenen Daten wieder zusammengeführt werden. Referenzen aus den XML-Dateien auf Entitäten des Wikis müssen aufgelöst werden. Zusätzlich müssen in der Datenbank anschließend die Daten zu Texten, Personen und Orten gespeichert und die entsprechenden Relationen gemäß des Matchings gesetzt werden. Orte können nun über ihre GND[6]-ID bzw. ihre GeoNames-ID einem Punkt auf der Karte zugeordnet werden und stellen somit die Verbindung zum Kartenmaterial und den damit verbundenen Informationen dar. Die Datenbank soll wiederum über die Webseite durchsucht werden können und somit die verbundenen Informationen für den Endnutzer zur Verfügung gestellt werden.

[6] Gemeinsame Normdatei

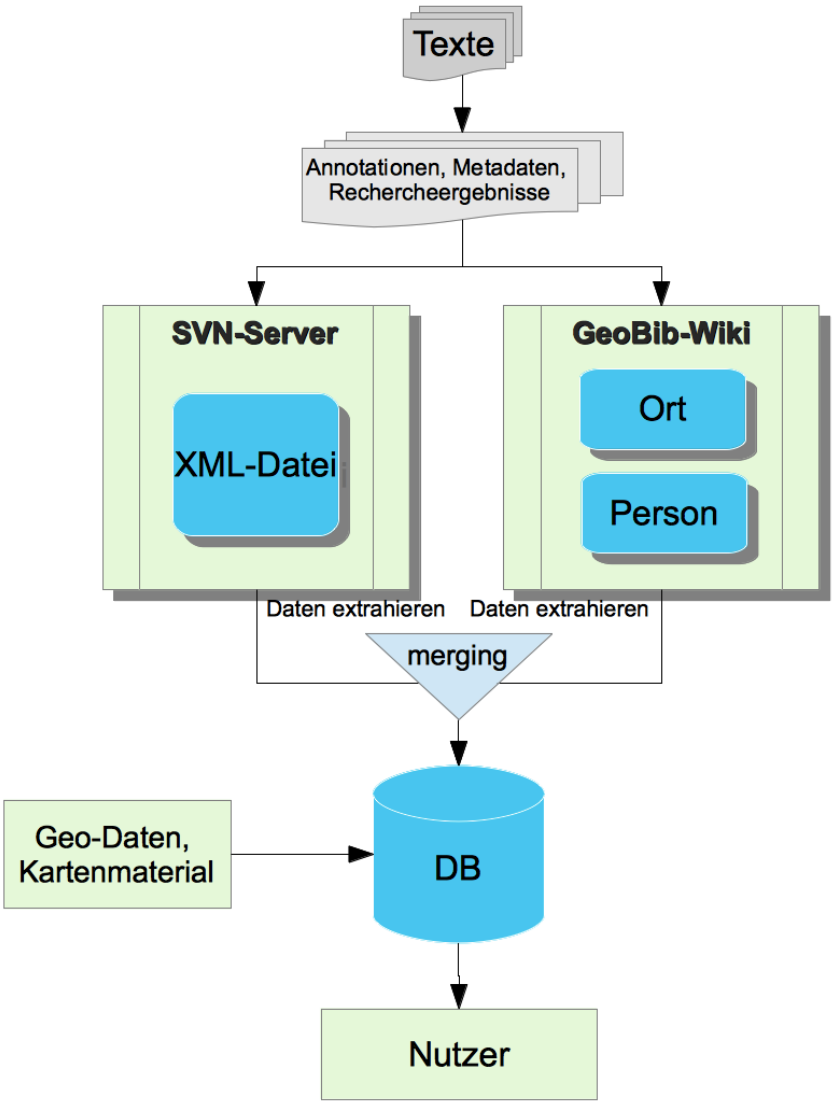

Abbildung 10: Verarbeitungspipeline - Literatur-Metadaten

4.2 Geodaten

Die Geodaten müssen teilweise verschiedenste Verarbeitungsschritte durchlaufen, bis sie über den GEOSERVER zur Verfügung gestellt werden können. An dieser Stelle werden verschiedene Prozesse beschrieben und der Ablauf an Beispielen dargestellt.

Der allererste Schritt wird die Erstellung eines Grunddatensatzes sein. Diese Daten werden in einer Projektion vorliegen. Grundsätzlich wurde sich auf das Koordinatensystem mit dem EPSG-Code 3857 geeinigt. Dieser EPSG-Code beschreibt das Koordinatensystem WGS84 mit sphärischer Mercatorprojektion.

Die ersten Schritte, die an Rasterdaten durchgeführt werden, sind Analyseschritte. Hier ist es wichtig zu wissen, ob die Rasterdaten eine Projektion besitzen und wenn ja, welche. Ist keine Projektion wie WGS84 oder UTM angegeben, so müssen die Rasterdaten zuvor georeferenziert werden. Das bedeutet, dass jedem Punkt auf der Karte ein Punkt in einem Koordinatensystem bzw. auf einer schon georeferenzierten Karte, also Daten mit Projektion, zugewiesen werden. Diese Punkte heißen Referenzpunkte. Weiterhin bekommt das zu georeferenzierende Bild durch die Verlinkung über die Punkte nach der Berechnung dasselbe Koordinatensystem wie der Grunddatensatz zugewiesen. Nach der Georeferenzierung muss die angepasste Karte noch mit den Grunddaten, auf die sie angeglichen wird, verglichen werden. Dieser Vorgang muss so genau wie möglich durchgeführt werden, da es möglichst wenig Unterschiede zwischen den beiden Daten geben sollte. Das bedeutet, dass Städte, Grenzen usw. übereinanderliegen müssen.

Ist ein Rasterdatensatz schon georeferenziert, hat aber noch nicht die gewünschte Projektion, muss diese durch Tools wie QGIS oder die GeoTools angepasst werden. Als Referenzsystem wird wieder EPSG: 3857 verwendet. Nachdem die Rasterdaten aufeinander angepasst und die Genauigkeit geprüft worden ist, können diese digitalisiert werden. Somit werden aus Rasterdaten Vektordaten erstellt. Diese Vektorisierung ist für die spätere Anzeige in der Karte der GeoBib-Webseite notwendig.

Sind Vektordaten vorhanden, müssen diese mit dem Grunddatensatz abgeglichen werden. Der Abgleich erfolgt über die Überlagerung der Daten in einem GIS. Dazu müssen die Daten zuvor wieder in ein Koordinatensystem überführt werden, so denn diese noch nicht in EPSG:3857 vorliegen. Die Genauigkeitsabschätzung mit folgender Anpassung an den Grunddatensatz kann auf verschiedenste Weise durchgeführt werden. Einerseits, kann der zu korrigierende Datensatz verschoben werden, falls nur eine Verschiebung nötigt ist. Sobald aber auch Drehung und Zerrung dazu kommen, muss eine Nachdigitalisierung vorgenommen werden.

Zusätzlich wird es Orte und Plätze geben, die in keiner Karte vorhanden oder in einem Gazeteer verortet sind. Daher müssen weitere POIs aufgenommen und Shapes erstellen werden, die diese Daten erfassen. Dazu gehören Orte wie Niemandsland, dieser Ort wird mehrfach vorhanden sein, aber immer mit einem speziellen Kommentar oder Hinweis, wie zwischen Belgien und Frankreich. Sind diese Orte in etwa eingrenzbar, werden Flächen oder Punkte dafür erstellt und in eine Shape gespeichert.

4.3 Zusammenspiel von Annotationen und Geodaten

Die beiden Datenpakete, Literatur-Metadaten und Geodaten, werden nicht nur über die Datenbank zusammenspielen, sondern auch im Kontext der Suche durch Abfrage-Algorithmen und Konzepte. Grundsätzlich ist auf Datenbankebene die Verbindung über die Entität PLACE des Datenbankdesigns gegeben. Die Vernetzung der beiden Datenpakete wird wie vorher erwähnt aber erst im Meilenstein M6.4R genauer erläutert. Die detaillierten Konzepte und deren Umsetzung im Bereich der Suche werden später im Meilenstein M6.3S erläutert und zeigen das genaue Zusammenspiel von Literatur-Metadaten und Geodaten auf.

Literaturverzeichnis

M2.1R (2013): Anforderungsanalyse und –definition: Literaturwissenschaftliches und didaktisches Nutzungskonzept, GUI-Konzept, Such- und Findbarkeitskonzept

M3.1R (2013): Zwischenbericht zur Beschaffung des Materials

M4.1S (2013): Annotationsschema für die bibliographischen Metadaten, die inhaltliche Verschlagwortung sowie die Geodaten

M5.2R (2013): Bestandserhebung zu verfügbaren digitalen geografischen Grundlagenkarten

M6.3S (unpublished): Bereitstellung eines System-Prototyps

M6.4R (unpublished): Systemdokumentation inkl. Hosting- und Datensicherungskonzept

MapServer (2013): MapServer Documentation – Release 6.2.1. MapServer Open Source Web Mapping. 07. Juli 2013.

Weblinks

Apache HTTP-Server – Weblink:

http://httpd.apache.org/ (gesehen: 23.07.2013)

http://projects.apache.org/projects/http_server.html (gesehen: 23.07.2013)

Apache Projekte – Weblink:

http://projects.apache.org/ (gesehen: 23.07.2013)

Apache Tomcat – Weblink:

http://tomcat.apache.org/ (gesehen: 23.07.2013)

http://projects.apache.org/projects/tomcat.html (gesehen: 23.07.2013)

Cartaro – Weblink:

http://cartaro.org/ (gesehen: 08.07.2013)

https://drupal.org/project/cartaro (gesehen: 08.07.2013)

Drupal – Weblink:

https://drupal.org/ (gesehen: 23.07.2013)

https://drupal.org/project/themes (gesehen: 23.07.2013)

https://drupal.org/project/modules (gesehen: 23.07.2013)

https://drupal.org/documentation/develop (gesehen: 23.07.2013)

Drupal API – Weblink:

https://api.drupal.org/api/drupal (gesehen: 23.07.2013)

FindTheBest – Weblink:

http://database-management-systems.findthebest.com/compare/30-43/MySQL-vs-PostgreSQL (gesehen: 23.07.2013)

Geoinformatik-Service – Weblink:

http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=-166418015 (gesehen: 12.06.2013)

GeoServer – Weblink:

http://geoserver.org/display/GEOS/Welcome (gesehen: 23.07.2013)

http://docs.geoserver.org/stable/en/user/index.html (gesehen: 08.07.2013)

GeoServer Installation – Weblink:

http://docs.geoserver.org/stable/en/user/installation/war.html (gesehen: 23.07.2013)

OGC – Weblink:

http://www.opengeospatial.org/ (gesehen: 08.07.2013)

OpenLayers – Weblink:

http://openlayers.org (gesehen: 08.07.2013)

MySQL – Weblink:

http://www.mysql.de (gesehen: 13.06.2013)

http://www.mysql.com (gesehen: 13.06.2013)

http://dev.mysql.com/doc/refman/5.1/de/spatial-extensions.html (gesehen: 08.07.2013)

PHP-FAQ – Weblink:

http://www.php-faq.de/q-db-vergleich.html (gesehen: 08.07.2013)

phpMyAdmin – Weblink:

http://www.phpmyadmin.net/ (gesehen: 08.07.2013)

PmWiki – Weblink:

http://www.pmwiki.org/ (gesehen: 12.06.2013)

PostGIS – Weblink:

http://postgis.net (gesehen: 08.07.2013)

PostgreSQL – Weblink:

http://www.postgresql.org (gesehen: 08.07.2013)

Wiki FH Köln – Weblink:

http://wikis.gm.fh-koeln.de/wiki_db/Category/ObjektrelationaleDB (gesehen: 23.07.2013)