Westfälische
Wilhelms-Universität
Münster

ifgi
Institute for Geoinformatics
University of Münster

Optimierungsmöglichkeiten eines webbasierten Geodatenmanagements im Kontext studentischer Projektarbeiten am Beispiel des Portals StudMap14

Bachelorarbeit im Fach Geoinformatik

vorgelegt von Andreas Ohrem

Matrikelnummer 370211

Münster, März 2014

Erstgutachter: Dr. Torsten Prinz

Zweitgutachter: Christian Knoth

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and Extensible Markup Language
API	Application Programming Interface
CD	Compact Disk
CSS	Cascading Style Sheets
DBF	Database File
DGK5	Deutsche Grundkarte im Maßstab 1:5.000
DOM	Document Object Model
EPSG	European Petroleum Survey Group
GDAL	Geospatial Data Abstraction Library
GDI	Geodateninfrastruktur
GeoTIFF	Geographic Tagged Image File Format
GIS	Geoinformationssystem
GML	Geography Markup Language
GPS	Global Positioning System
GPX	Global Positioning System Exchange Format
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
MySQL	My Structured Query Language
OGC	Open Geospatial Consortium
OSM	Open Street Map
PC	Personal Computer
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PNG	Portable Network Graphics
PRJ	Projection File

RGB	Red Green Blue
SHP	ESRI Shapefile
SHX	Shapefile Database Index File
SQL	Structured Query Language
SRS	Spatial Reference System
TFW	Tagged Image File Format World File
TIFF	Tagged Image File Format
TK25	Topographische Karte im Maßstab 1:25.000
UAV	Unmanned Aerial Vehicle
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WCS	Web Coverage Service
WFS	Web Feature Service
WGS84	World Geodetic System 1984
WKT	Well-Known Text
WMS	Web Map Service
WWU	Westfälische Wilhelms-Universität
XML	Extensible Markup Language
ZDM	Zentrum für digitale Medien

Danksagung

Zu Beginn möchte ich für die Beratungsgespräche zur Findung des Themas dieser Arbeit und die Betreuung während der Bearbeitungszeit meinen Gutachtern Dr. Torsten Prinz und Christian Knoth danken.

Zusätzlich möchte ich André Wieghardt für Hilfe bei Arbeiten am Server, Florian Lahn für Ratschläge bei der Einbindung von GeoTiff-Dateien und Korbinian Nagel für programmiertechnische Hinweise und Tipps an der richtigen Stelle danken.

Außerdem vielen Dank an dieser Stelle an meinen Bruder und meine Eltern für die Unterstützung bei dieser Arbeit.

1 Einleitung

Die vorliegende Arbeit zeigt Optimierungsmöglichkeiten für ein webbasiertes Geodatenmanagement hinsichtlich Interoperabilität und Zusammenarbeit im Rahmen studentischer Projektarbeiten auf und setzt einige dieser Möglichkeiten am praktischen Beispiel der Applikation StudMap14 um. StudMap14 ist ein mit Web-Technologien wie JavaScript, PHP, Sencha Ext JS [**1**] und GeoExt [**2**] implementiertes Webportal, das ein Teil der Geodateninfrastruktur des Fachbereichs 14 Geowissenschaften der Universität Münster ist [**3**]. Zudem wird ein GeoServer [**4**] zur Bereitstellung von Layern des OGC-Standards Web Map Service (WMS) [**5**] und die Geospatial Data Abstraction Library (GDAL) [**6**] zur Geodatenverarbeitung verwendet. Es geht in dieser Arbeit nicht um die Neuentwicklung eines Systems, sondern um die Verbesserung und Erweiterung einer bestehenden und sich bereits in Verwendung befindlichen Software.

In Kapitel 2 (Anforderungsanalyse) wird zunächst analysiert, welche Anforderungen beziehungsweise Optimierungsmöglichkeiten bestehen und daraufhin festgelegt, welche von diesen umzusetzen sind. Kapitel 3 (Implementierung) beschreibt zuerst die verwendeten Technologien und geht anschließend auf technische Details der Implementierung der neuen Funktionen ein. Die Funktionsfähigkeit und die Benutzerfreundlichkeit dieser Funktionen wird im Rahmen einer Nutzerstudie in Kapitel 4 evaluiert. Eine Zusammenfassung dieser Arbeit und einen Ausblick auf mögliche zukünftige Erweiterungen bietet Kapitel 5. Dieses Einleitungskapitel stellt im Weiteren die Applikation StudMap14 vor und legt die genauere Motivation zur Erweiterung dieses Systems dar.

1.1 StudMap14

Die Webapplikation StudMap14 bietet die Möglichkeit, in einem einfachen Browser-Geoinformationssystem sowohl offene Geodaten verschiedener Anbieter als auch einzelne nicht frei verfügbare Datensätze zu sichten und grundlegende GIS-Operationen auf diesen durchzuführen.

StudMap14 ist aus dem Universitätsnetz der WWU Münster unter folgender Adresse erreichbar:

http://studmap14.uni-muenster.de/GeoExt/index.html

Ursprünglich entwickelt wurde die Plattform Anfang 2010 von Georg Kaspar, Phillip Keller und Holger Hopman, Studierende am Institut für Geoinformatik beziehungsweise Institut für Landschaftsökologie an der WWU Münster. Hiermit sollte im Zuge des Aufbaus der Geodateninfrastruktur „GDI One 4 All“ des Fachbereichs Geowissenschaften das Angebot eines webbasierten Geodatenmanagements geschaffen werden. Die Entwickler formulieren den Zweck des Systems folgendermaßen: „Über diese [Applikation] soll Studierenden sowohl einfacher Zugriff als auch Interaktion mit gebündelten Geodiensten verschiedener

Art ermöglicht werden.“ [**3**] Eine Weiterentwicklung seit der ersten Implementierung bis zu dieser Arbeit fand hauptsächlich in Form von Wartungsarbeiten und Fehlerbehebungen statt. Um neue Features wurde die Plattform nicht erweitert.

StudMap14 wird vor allem im Bereich Landschaftsökologie genutzt. Zum einen ist das Angebot der zur Verfügung gestellten WMS-Layer dort von Interesse, zum anderen wird der DGK5-Kacheldownload (s. Abschnitt 1.1.2) genutzt, da das Finden einer einzelnen Kachel über die grafische Repräsentation durch StudMap14 im Gegensatz zur Identifizierung über einen Dateinamen deutlich einfacher ist.

1.1.1 Zielgruppe

Der Name „StudMap14“ weist bereits darauf hin, dass die Nutzer-Zielgruppe an erster Stelle aus Studierenden des Fachbereichs 14 Geowissenschaften der WWU Münster besteht. Dieser umfasst die Institute für Geographie (inkl. Didaktik der Geographie), Landschaftsökologie, Geologie und Paläontologie, Mineralogie, Planetologie und Geoinformatik [**7**].

Da nur diese Nutzergruppe das System nutzen darf, ist der Zugriff auf StudMap14 technisch eingeschränkt. Ein Nutzer muss mit Hilfe seiner Universitäts-Nutzerkennung entweder direkt über das Universitätsnetz oder über eine Virtual-Private-Network-Verbindung (VPN) auf die Plattform zugreifen. Diese Einschränkung geschieht vor allem auch aus rechtlichen Gründen, um geschützte Geodaten nicht allgemein verfügbar zu machen.

1.1.2 Bisherige Funktionalität

Der grafische Aufbau von StudMap14 (s. Abbildung 1) wird dominiert von einem zentralen Kartenfenster. Dem Nutzer ist es möglich, mit Pan- und Zoom-Mausgesten auf der Grundkarte zu navigieren und einen gewünschten Kartenausschnitt zu wählen. Zusätzlich enthält das Kartenfenster einen Schieberegler zur Veränderung der Zoomstufe, eine Maßstabsanzeige und eine ausklappbare Miniaturkarte zur Verbesserung der Orientierung des Nutzers.

Das Kartenfenster ist umgeben von verschiedenen Bedienelementen. Auf der linken Seite befindet sich sowohl eine Layerliste mit verfügbaren Open Street Map (OSM) Baselayern und weiteren visualisierten Layern als auch eine Liste von WMS-Layern, die von StudMap14 bereitgestellt werden. Diese Zusammenstellung von WMS-Layern umfasst unter anderem Layer mit Naturschutzgebieten, Bodenkarten, TK25- und DGK5-Blattschnitte, digitale Orthofotos, Höhenkarten und Geobasisdaten für den Bereich Münster und Umgebung. Der allgemeine räumliche Fokus der angebotenen Geodaten liegt hingegen auf Nordrhein-Westfalen und umliegenden Gebieten. Sämtliche Layer bieten ein per Rechtsklick erreichbares Kontextmenü, über das Eigenschaften, Einstellungen und Operationen für den Layer verfügbar sind.

Unter der WMS-Layerliste findet der Nutzer eine Suchfunktion in Form eines Textfeldes, mit dessen Hilfe er in der Lage ist, die OSM-Datenbank über die Nominatim-API [**8**] nach Geoobjekten zu durchsuchen und eine Ergebnisliste mit entsprechender Verlinkung zur Position auf der StudMap14-Karte anzuzeigen.

Der obere Navigationsbereich der Applikation bietet zusammengefasst in Reitern und Kategorien verschiedene Werkzeuge an. Zunächst stehen weitere Tools zur Navigation auf der Karte zur Verfügung. Mit Hilfe des Zoom-Werkzeugs lässt sich ein Rahmen aufziehen, der den neuen Kartenausschnitt definiert. Es lassen sich weiterhin die maximale Kartenausdehnung und ein Gitter mit Längen- und Breitengraden anzeigen. Ein Verlaufswerkzeug bietet die Möglichkeit, Eingaben auf der Karte rückgängig zu machen beziehungsweise zu wiederholen.

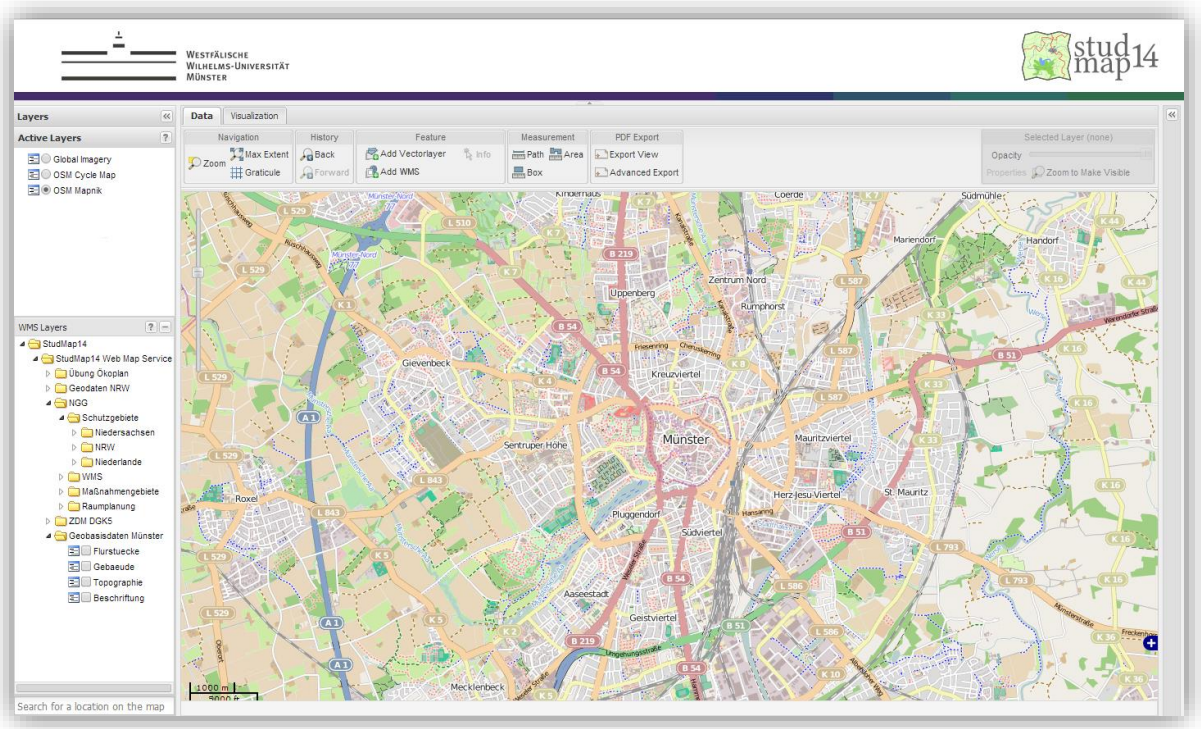

Abbildung 1 *Grafische Benutzerschnittstelle von StudMap14 mit den Elementen „Aktive Layer“ und „WMS-Layer“ auf der linken Seite, einer Toolbar auf der oberen Seite und dem Kartenfenster als zentrales Element (Screenshot vom 8. Februar 2014)*

In der Kategorie „Feature“ ist der Download von DGK5-Kartenkacheln im GeoTiff-Format, deren Blattschnitte als WMS-Layer bereitgestellt sind, über das Info-Tool möglich. Zudem können hier KML-Dateien (von Google entwickeltes Vektor-Geodatenformat) und per GPS aufgezeichnete GPX-Datensätze hochgeladen werden. Diese werden zur Liste der aktiven Layer hinzugefügt und können für die Dauer der Sitzung auf der Karte dargestellt werden. Im Kontextmenü der Layer können Beschriftungen und der grafische Stil der Geoobjekte (Farbe, Linienbreite und –stil, Punktgröße, usw.) angepasst werden.

Mit Hilfe von Messwerkzeugen ist der Nutzer in der Lage, die Länge eines eigens definierten Pfades und den Flächeninhalt von frei gewählten Rechtecken und Polygonen zu bestimmen. Zusätzlich werden die Koordinaten der Vertices als Well-Known Text (WKT) [**9**] in verschiedenen räumlichen Referenzsystemen

und die Bounding Box des definierten Gebiets angezeigt. Der Export des aktuell gewählten, beziehungsweise eines per Rechteck-Werkzeug aufgezogenen Kartenausschnittes als PDF-Dokument ist in der letzten Kategorie des Reiters „Data“ möglich. Sowohl die Ergebnisse der Anwendung der Messwerkzeuge als auch Optionen für den PDF-Export werden in einem neuen Fenster unter der WMS-Layerliste eingeblendet.

Zusätzlich zu diesen ausschließlich zur Sichtung von Daten nutzbaren Funktionen sind im Reiter „Visualization“ Zeichenfunktionen zur Erstellung von Punkten, Pfaden und Polygonen und Modifikationsfunktionen zur Bearbeitung dieser Daten (Auswählen, Verformen, Rotieren und Skalieren) vorhanden. Außerdem können die Attribute Farbe und Transparenz verändert werden.

Rechtsbündig oberhalb der Karte befindet sich außerdem ein Feld mit Angaben zum aktuell ausgewählten Layer. Hier können Eigenschaften des Layers per Klick eingeblendet, der Transparenzgrad des Layers eingestellt und ein Zoom zur optimalen Zoomstufe des Layers durchgeführt werden. Auf der rechten Seite des Kartenfensters lässt sich eine Legende einblenden. Diese erklärt die Symbolik der aktuell visualisierten Layer.

1.2 Optimierungspotentiale und resultierende Fragestellung

In diesem Abschnitt soll geklärt werden, welche Optimierungs- und Verbesserungsmöglichkeiten für StudMap14 existieren und warum eine Weiterentwicklung und Erweiterung von StudMap14 sinnvoll ist. Zur Beantwortung dieser Frage muss zunächst noch einmal der Zweck der Applikation verdeutlicht werden. StudMap14 ist ein Portal, das im Gegensatz zu anderen Geoportalen die spezielle Zielgruppe der Studierenden am Fachbereich 14 hat. Bei studentischen Projektarbeiten und in Kursen an den Instituten sind vor allem Kooperation zwischen den Teilnehmern und Interoperabilität sehr wichtig. Zu diesem Zweck wären folgende anhand von Fallbeispielen verdeutlichte Funktionen von Vorteil.

- Es sollen Geodaten in Zusammenarbeit mit anderen Nutzern in einem webbasierten Geoinformationssystem (wie beispielsweise StudMap14) visualisiert werden und mit bestehenden Geodaten verglichen werden. Wird StudMap14 um Benutzer-Funktionen wie persönliche Layer erweitert, kann ein geteiltes StudMap14-Benutzerkonto erstellt und die zu vergleichenden Geodaten hochgeladen werden. Die zusammenarbeitenden Nutzer können sich mit dem geteilten Account anmelden und haben so ohne großen Aufwand Zugriff auf die Geodaten, welche sie nun komfortabel per Mausklick auf der Karte visualisieren können.

- Per Unmanned Aerial Vehicle (UAV) akquirierte Luftbildaufnahmen im GeoTiff-Format sollen mit bestehenden Luftaufnahmen aus dem WMS-Pool von StudMap14 verglichen werden. Zu diesem Zweck ist die Unterstützung der Einbindung von GeoTiff-Dateien in StudMap notwendig.

- Zwei Nutzer möchten Geodaten austauschen. Das Programm, mit welchem Nutzer 1 die Geodaten von Nutzer 2 öffnen und bearbeiten möchte, unterstützt allerdings das Format nicht, in dem die Daten vorliegen. Hierfür könnte StudMap14 die Plattform zur Konvertierung der Daten zwischen verschiedenen Formaten darstellen. Nach der Benutzung der Funktion zum Teilen eines Layers müssten die Daten nicht mehr mit Hilfe eines weiteren Mediums versendet werden, sondern könnten direkt mit der Export-Funktion von der Plattform heruntergeladen werden.

Diese Funktionalitäten sind in dieser Form bisher nicht vorhanden. Deshalb ist eine Erweiterung von StudMap14 sinnvoll und angebracht. Der Fokus der Optimierung liegt auf der Verbesserung der Interaktion und der Kooperation zwischen Nutzern der Webapplikation im Rahmen studentischer Projektarbeiten und der Optimierung der Interoperabilität und Benutzerfreundlichkeit. Welche Anforderungen aus Benutzersicht an eine solche Erweiterung gestellt werden und welche Ansätze zur Umsetzung dieser Anforderungen es aus Entwicklersicht gibt, soll das nächste Kapitel klären.

2 Anforderungsanalyse

Da aus Gründen der Praktikabilität nicht alle potentiell möglichen Verbesserungen im Rahmen dieser Arbeit umgesetzt werden können, wird hier geklärt, welche dieser Verbesserungsmöglichkeiten zugunsten anderer verworfen werden müssen. Trotzdem werden alle Funktionen hier festgehalten, um als Ideen für eventuelle weitere Arbeiten zu dienen. Die Anforderungsanalyse ist hier in zwei Bereiche unterteilt: Anforderungen aus der Sicht eines Nutzers von StudMap14 und Ansätze zur Umsetzung dieser Anforderungen aus der Sicht des Entwicklers.

2.1 Anforderungen aus Benutzersicht

Aus Sicht eines Nutzers soll das System intuitiv bedienbar sein. Das heißt, alle vorhandenen Features müssen übersichtlich angeordnet und ausreichend erklärt sein. Außerdem sollte es beim Verwenden von Buttons und anderen Interaktionsmöglichkeiten schnell reagieren und im Allgemeinen performant sein. Zudem ist es einem Benutzer wichtig, dass das System zuverlässig und sicher ist. Das heißt, Daten, die er hochgeladen hat, dürfen nicht verloren gehen und persönliche Daten (wie E-Mail Adresse und Passwort) dürfen nicht an Dritte gelangen.

Ein StudMap14-Nutzer hatte bisher die Möglichkeit, verschiedene Web Map Services und Vektorgeodaten in zwei verschiedenen Formaten in der Webapplikation zu sichten. Möchte ein Nutzer allerdings mit anderen Nutzern zusammenarbeiten und die Plattform gemeinsam verwenden, kann dies nur durch das Arbeiten an ein und demselben System geschehen. Eine bessere Möglichkeit bestünde darin, Daten mit anderen Nutzern über StudMap14 als Schnittstelle austauschen zu können. Diese wären so zwar gemeinsam nutzbar, jedoch an unterschiedlichen Geräten, die unter Umständen weit voneinander entfernt sein können.

Auch ist es Nutzern wichtig, Daten, die visualisiert werden sollen, vorab möglichst nicht noch in ein von der Plattform unterstütztes Format konvertieren zu müssen. Aus diesem Grund wäre eine Erweiterung der unterstützten Geodatenformate um häufig genutzte Standards und De-facto-Standards aus Nutzersicht sehr begrüßenswert. Vor allem ist die Visualisierung von Rasterdaten bisher nur via Einbindung eines WMS möglich. Der Aufwand, eigene Rasterdaten zunächst als WMS zu veröffentlichen, um sie danach in StudMap14 visualisieren zu können, ist für einen Nutzer entweder unverhältnismäßig hoch oder sogar überhaupt nicht durchführbar, da entsprechender Zugang zu einem Server notwendig ist. Statt Daten vor der Verwendung in StudMap14 konvertieren zu müssen, wäre es vorteilhaft, sie in Form einer Exportfunktion in einem anderen Format wieder herunterladen zu können als in dem, in welchem sie hochgeladen wurden.

2.2 Ansätze zur Umsetzung der Anforderungen

Um dem Bedarf nach dem Austausch von Geodaten zwischen Nutzern mit StudMap14 als Plattform nachzukommen, müssen zunächst grundlegende Benutzer-Funktionen implementiert werden, um die Identifizierung einzelner Nutzer zu ermöglichen. Zu diesen Funktionen gehören Merkmale wie Registrierung, Login und Logout. Um die Voraussetzungen für einen Austausch von Daten zu schaffen, muss weiterhin eine Möglichkeit gefunden werden, Daten nicht nur temporär hochladen zu können, sondern auch persistent zu speichern. Ist dies der Fall, können Daten über die Nutzung des gleichen Accounts gemeinsam genutzt werden. Denkt man diesen Ansatz weiter, kann auch eine Funktion implementiert werden, mit welcher ein Layer per Klick und darauffolgender Eingabe eines Benutzernamens direkt an das Konto eines anderen Nutzers gesendet werden kann. Zudem kann die bereits vorhandene Export-Funktion von Vektor-Layern zum Download hochgeladener Daten in anderen Formaten verwendet werden. Die Aktualisierung dieser Funktion darf nicht außer Acht gelassen werden, wenn die unterstützten Datenformate erweitert werden.

Die Erweiterung der visualisierbaren Geodaten-Rasterformate betreffend, ist es vor allem angebracht, das GeoTiff-Format zu unterstützen, da es sich bei diesem Format um den De-facto-Standard für Raster-Geodaten handelt [**10**]. Diese Unterstützung kann mit zwei verschiedenen Methoden erreicht werden. Zum einen kann der Datensatz zum Beispiel mit einem GeoServer (s. Abschnitt 3.1.8) zunächst als WMS veröffentlicht und daraufhin auf der Karte visualisiert werden. Diese Aufgabe ist allerdings schwierig zu automatisieren und sollte nicht dem Benutzer zugeschoben werden. Zum anderen ist die direkte Einbindung von Bilddateien in OpenLayers (s. Abschnitt 3.1.3.3) möglich. Hierzu sind zwar einige Vorverarbeitungsschritte notwendig, allerdings handelt es sich hierbei um die einfachere Variante zur Einbindung von GeoTiff-Layern. Der komplette Prozess der Vorverarbeitung und die eigentliche Einbindung ist in Abschnitt 3.2.1.1 beschrieben.

Denkbar wäre auch die Unterstützung des Erdas Imagine IMG-Formates [**11**], mit dem auch multispektrale Rasterdaten gespeichert werden können. Hierzu wäre eine weitreichende Implementierung von Features zur Einstellung der Art und Weise der Darstellung dieser multispektralen Daten bezüglich Belegung der RGB-Farbkanäle notwendig.

Die Anzahl der unterstützten Vektorformate kann durch die Unterstützung von ESRI Shapefiles [**12**] und GML-Dateien erreicht werden. Da es sich beim GML-Format um einen offenen Standard handelt [**13**], wird es von OpenLayers direkt unterstützt. Eine Einbindung ist somit leicht möglich. Shapefiles, die zwar als De-facto-Standard im Vektordatenbereich gelten, allerdings ein proprietäres Format sind, lassen sich jedoch nur durch vorherige Konvertierung in ein von OpenLayers unterstütztes Format darstellen.

Durch die Einbindung weiterer Baselayer kann ebenfalls die Funktionalität von StudMap14 erweitert werden. Mehr verschiedene Baselayer von verschiedenen Anbietern erhöhen den Nutzen der Plattform beim

Vergleichen von Geodaten mit bereits vorhandenen Datensätzen. Hier bieten sich die bereits in der OpenLayers-Bibliothek vorhandenen Google-Baselayer an, welche einfach eingebunden werden können.

Eine Funktion zum Export eines Kartenausschnittes mit allen visualisierten Layern nicht nur wie bisher als PDF, sondern auch beispielsweise als GeoTiff-Datei, könnte Nutzern die Gelegenheit bieten, neue Geodaten als Komposition aus anderen zu generieren.

Um die Funktionalität zur Bearbeitung von Vektorfeatures zu erweitern, sollte eine Funktion zum Löschen und eine Funktion zur Beschriftung von erstellten Features implementiert werden. Zusätzlich könnte man dem Benutzer die Möglichkeit geben, eigens Attribute für Features zu vergeben und mit Werten zu füllen. Dies ließe sich unter anderem durch eine Tabellenansicht aller Attribute eines Layers und durch Anzeige der Attribute in den Eigenschaften eines einzelnen Features visualisieren.

Zusätzlich zu diesen für Nutzer des Portals angebotenen Funktionen wären auch Features für Administratoren denkbar. Beispielsweise wäre eine Funktion zur Sichtung, Versionsverwaltung und Zur-Verfügung-Stellung von Nutzer-Geodaten an andere Nutzer möglich. Hierbei müsste allerdings verschärft auf Datenschutz geachtet werden und das Einverständnis des Nutzers eingeholt werden.

Um die Benutzerfreundlichkeit von StudMap14 zu erhöhen, können die Syntax und die Benennung von Funktions-Elementen in der Toolbar optimiert beziehungsweise eindeutiger gestaltet werden. Da die Funktionalität des Info-Tools auch über die Layer-Eigenschaften vorhanden ist, die über das Rechtsklick-Menü eines Layers erreichbar sind, kann der Info-Button nach dieser Änderung entfernt werden. So können Redundanzen bei der Benutzerführung vermieden werden. Zudem kann die Benutzerfreundlichkeit erhöht werden, indem man Benutzern erlaubt, nicht weiter benötigte Layer wieder aus der Layerliste zu entfernen, um die Übersichtlichkeit zu bewahren.

2.3 Konsequenzen für den Entwickler

Die Benutzeranforderungen sind aus Entwicklersicht grundsätzlich umsetzbar. Wie bereits erwähnt, konnten allerdings nicht alle genannten Funktionen implementiert werden. Größtenteils ist dies auf den für den Rahmen dieser Arbeit insgesamt zu hohen Implementierungsaufwand zurückzuführen. Welche Funktionen umgesetzt wurden und welche nur perspektivisch aufgezeigt werden, ist Tabelle 1 zu entnehmen. Die Wahl der Priorität erfolgt nach Wichtigkeit für den Benutzer in Bezug auf Interoperabilität sowie nach Umsetzungsaufwand. Nähere technische Details zu den umgesetzten Funktionen finden sich im Kapitel zur Implementierung.

Tabelle 1: Umgesetzte und perspektivische Funktionen

umgesetzte Funktionen	**perspektivische Funktionen**
Erweiterung der unterstützten Geodatenformate • GeoTiff-Rasterdaten • Geography Markup Language • ESRI Shapefile	Unterstützung von • Erdas Imagine IMG-Dateien • GeoTiff-Dateien ohne Geoinformationen im Datei-Header mit zugehöriger TIFF World File (s. Abschnitt 4.1)
Benutzer-Funktionen • Login, Logout, Registrierung • persönliche Layer • Layer teilen	Benutzer-Funktionen • persönliche WMS-Layer Administrator-Funktionen
Einbindung der Google-Baselayer	
Erweiterung der Export-Funktion um weitere Formate (GML, GPX)	Kartenausschnitt-Export als KMZ- oder GeoTiff-Datei
mehr Funktionalität für Vektor-Features • Feature löschen • Label hinzufügen/bearbeiten • Änderung der Punktgröße	eigens definierte Feature-Attribute

3 Implementierung

Dieses Kapitel zeigt einerseits die verwendeten Technologien bei der Implementierung auf und legt die zur Optimierung von StudMap14 implementierten Funktionen und Features detailliert dar. Aufgrund der Vielzahl der verwendeten Programmiersprachen und Technologien sind Quellcode-Listings in dieser Arbeit am rechten Rand mit der entsprechenden Programmiersprache beschriftet, in der sie geschrieben sind. Auslassungen in den Quellcode-Listings sind mit `[...]` markiert. Der vollständige und kommentierte Quellcode befindet sich auf der beiliegenden CD.

3.1 Technologien

Da es sich bei StudMap14 um ein webbasiertes Geoinformationssystem handelt, welches mit Hilfe gängiger Webtechnologien implementiert ist, werden unter anderem HTML, CSS, JavaScript, PHP und MySQL verwendet. Größtenteils handelt es sich hierbei um kostenlose, frei verfügbare oder gar Open Source Technologien. Die starke Verknüpfung, das vielfältige Zusammenspiel dieser Technologien und die Art und Weise ihres Einsatzes in der Implementierung von StudMap14 werden in diesem Kapitelabschnitt erläutert.

3.1.1 HTML

Die Hypertext Markup Language (HTML) [**14**] ist eine vom World Wide Web Consortium (W3C) [**15**] standardisierte Auszeichnungssprache zur Erstellung von Webseiten. „Mit HTML lassen sich die Struktur, der Inhalt und das Verhalten eines Dokuments beschreiben bzw. auszeichnen[.] […] HTML-Dokumente sind also reine Textdokumente mit Inhalt (Texte, Überschriften, Bilder, Tabellen, etc.), dem Sie eine Struktur (und somit ein Aussehen) zuweisen.“ [**16**] Diese Auszeichnung geschieht mit Hilfe von Tags, welche ineinander verschachtelt die Struktur eines Dokuments definieren. Derlei ausgezeichnete HTML-Dokumente können mit Webbrowsern interpretiert und angezeigt werden.

In StudMap14 definiert der HTML-Code den grafischen Aufbau des Systems und alle Oberflächenelemente wie beispielsweise Buttons, Panels und Labels. Große Teile des HTML-Quelltextes von StudMap14 werden per JavaScript automatisch generiert. Dies geschieht im Speziellen mit Hilfe der Bibliothek Ext JS von Sencha [**1**] (s. Abschnitt 3.1.3.1).

3.1.2 CSS

Auch Cascading Style Sheets (CSS) [**17**] sind ein vom W3C entwickelter Standard zur Formatierung von Webseiten. Im Gegensatz zu HTML (s. Abschnitt 3.1.1) ist CSS jedoch nicht für die Strukturierung

verantwortlich, sondern für die oberflächliche Gestaltung, in der die Struktur einem Nutzer präsentiert wird. „Ziel ist es, […] den Inhalt von der Gestaltung zu trennen.“ [**16**] Hierzu werden HTML-Tags entsprechende Stilattribute zugewiesen. So lässt sich beispielsweise definieren, dass Überschriften generell im Fettdruck angezeigt werden oder Tabellen mit einem zwei Pixel breiten Rahmen versehen werden sollen.

Analog zur Generierung großer Anteile des HTML-Quelltextes wird auch die CSS-Formatierung in StudMap14 überwiegend von Ext JS übernommen. Nur einige allgemeine Definitionen wie das Aussehen von Überschriften, Hyperlinks und Icons sind in einer eigenen CSS-Datei angegeben.

3.1.3 JavaScript

JavaScript ist eine Skriptsprache, die clientseitig im Browser interpretiert und ausgeführt wird [**16**]. Sie wird genutzt, um während der Laufzeit dynamische Inhalte auf einer Webseite präsentieren zu können. Hierzu wird JavaScript eng mit dem HTML-Quellcode (s. Abschnitt 3.1.1) verbunden. Es kann direkt in HTML-Code eingebunden werden oder aus verlinkten externen JavaScript-Dateien auf diesen zugreifen.

Für StudMap14 ist diese Technologie essentiell, da die vollständige Benutzeroberfläche des Systems mittels JavaScript generiert wird und sämtliche Benutzereingaben (z.B. Klicks auf Buttons, Operationen auf der Karte, Ein- und Ausblenden von Layern, usw.) von JavaScript erkannt werden. Auch die Ausführung entsprechender Konsequenzen dieser Eingaben beruht auf JavaScript. In den folgenden Abschnitten werden die einzelnen JavaScript-Bibliotheken aufgezählt, die neben eigenem JavaScript-Quellcode Verwendung finden, und deren Funktionen für StudMap14 näher erläutert.

3.1.3.1 Sencha Ext JS

Beim Ext-JS-Framework handelt es sich um eine von der Firma Sencha entwickelte proprietäre JavaScript-Bibliothek, die zur Erstellung von Benutzeroberflächen, wie man sie aus Desktopanwendungen kennt, verwendet werden kann [**1**]. Hierzu wird die Benutzeroberfläche nicht, wie üblich, im HTML-Quellcode in Verbindung mit CSS definiert. Vielmehr wird sie im JavaScript-Quellcode mit Hilfe von JavaScript-Objekten festgelegt, die einzelne Komponenten der Oberfläche repräsentieren. Der HTML-Code, welcher diese Komponenten repräsentiert, wird nun automatisch mit Hilfe des Document Object Model (DOM) in den HTML-Quelltext des Dokuments eingebettet. Das Document Object Model ist eine vom W3C entwickelte Schnittstelle, welche „einen einheitlichen Standard dafür [bildet], wie die Elemente eines HTML-Dokuments mit Scriptsprachen anzusteuern sind.“ [**16**]

Ein großer Vorteil der Nutzung einer solchen Bibliothek liegt darin, dass die sehr einfache Einbindung und die Spezifikation des Verhaltens von Buttons, Panels, Labels, Tabs, Formularen und vielen weiteren Benutzerinterface-Elementen möglich sind. Der Nachteil hingegen ist jedoch, dass eine manuelle Manipulation einzelner HTML-Elemente nur unter großem Aufwand möglich ist. Bietet die Bibliothek eine

Funktion zur gewollten Manipulation eines Elements nicht an, muss also ein Workaround gefunden werden, um das gewünschte Ergebnis zu erzielen.

3.1.3.2 GeoExt

Die Open-Source-Bibliothek „GeoExt“ erweitert Ext JS (s. Abschnitt 3.1.3.1) um Funktionalitäten, welche Kartenanwendungen innerhalb eines Ext-JS-Frameworks ermöglichen. Hierzu wird OpenLayers (s. Abschnitt 3.1.3.3) an Ext JS angebunden [**2**]. So können mit GeoExt zum Beispiel Baumstrukturen für Layerlisten, Legenden, Schieberegler zur Steuerung der Layertransparenz, Marker-Pop-ups auf der Karte und ähnliche Elemente im optischen Stil des Ext-JS-Frameworks eingebunden werden. Die Layerlisten am linken Bildschirmrand von StudMap14 sind mit Hilfe der GeoExt-Bibliothek erstellt worden.

3.1.3.3 OpenLayers

Mit der frei verfügbaren JavaScript-Kartenbibliothek „OpenLayers“ sind webbasierte Kartenanwendungen (ähnlich Google Maps oder Open Street Map) zur Anzeige von Geodaten im Browser umsetzbar [**18**]. Die Funktionalität der Bibliothek umfasst unter anderem Einbindung von Baselayern und Layern verschiedener standardisierter Geodatenformate in Raster- beziehungsweise Vektorgestalt. Grundlegende Navigationsoperationen wie Zoomen oder Pannen sind ebenfalls implementiert. Hinzu kommen Features zur Verbesserung der Benutzerfreundlichkeit wie zum Beispiel Pop-ups, Marker, Symbolstile für Vektorfeatures und Handler für Mausgesten auf der Karte und den Features. OpenLayers ist in StudMap14 eng mit der Erweiterung GeoExt (s. Abschnitt 3.1.3.2) verknüpft und ist für alle Elemente der Karte selbst sowie für die Bedienelemente auf dem Kartenfenster verantwortlich.

3.1.4 PHP

Hypertext Preprocessor (PHP, ehem. für Personal Home Page [**19**]) ist eine serverseitige Scriptsprache zur dynamischen Generierung von Webseiten. Hierbei setzt PHP auf das folgende Konzept: Die vom Nutzer per HyperText Transfer Protocol (HTTP) versendete Anfrage an einen Webserver wird auf diesem ausgewertet. Statt wie ohne PHP üblich einfach bereits auf dem Server existierende HTML-Dokumente zurück an den Client zu senden, wird das HTML-Dokument (s. Abschnitt 3.1.1) basierend auf eventuellen Benutzereingaben aus dem HTTP-Header dynamisch generiert und erst dann als Antwort an den Client gesendet (s. Abbildung 2) [**19**].

StudMap14 nutzt PHP unter anderem zum Datenupload. Per Ajax-Request (s. Abschnitt 3.1.5) werden Formularfeldeingaben des Nutzers an ein PHP-Script auf dem Server weitergeleitet. Das PHP-Script verarbeitet die hochgeladenen Daten, speichert sie auf dem Server und gibt Ort und Dateinamen zurück an das aufrufende JavaScript. Dieses kann die hochgeladenen Daten daraufhin anzeigen. Zudem wird PHP zur Anbindung von StudMap14 an die MySQL-Datenbank (s. Abschnitt 3.1.6) verwendet.

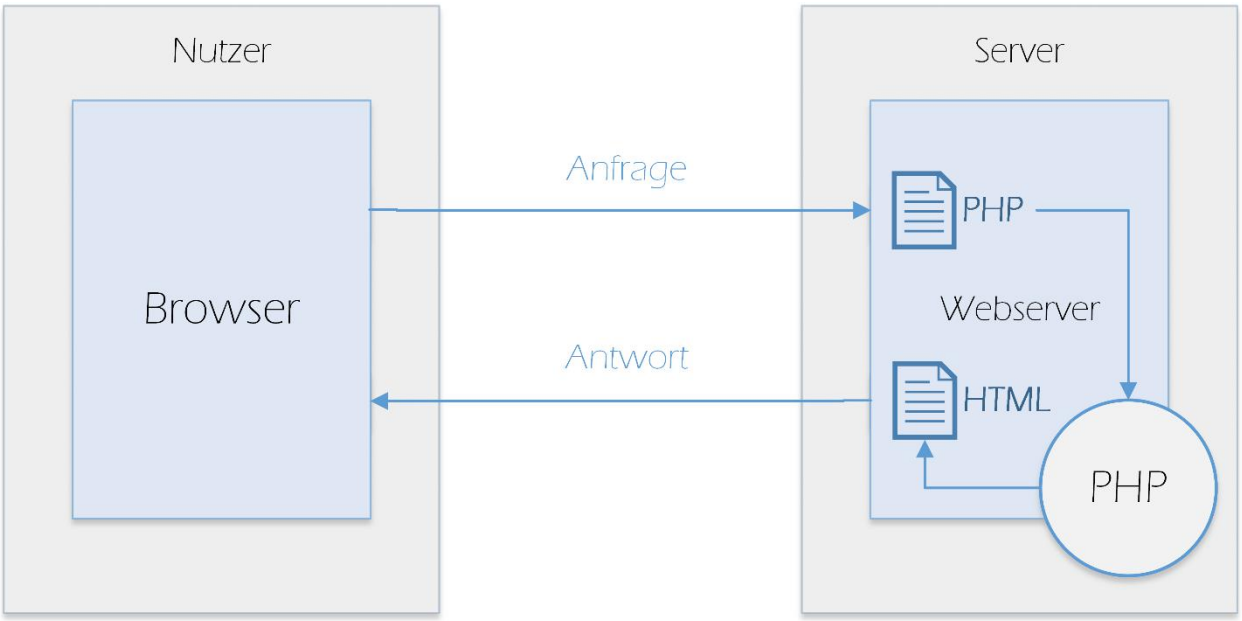

Abbildung 2 *Diagramm zur Veranschaulichung des Client-Server-Modells mit PHP, Verarbeitung der HTTP-Anfrage des Benutzers durch den Server, der dynamisch mit Hilfe eines PHP-Skriptes ein HTML-Dokument generiert und dieses als Antwort an den Browser des Nutzers zurücksendet (eigene Abbildung nach [**19**]).*

3.1.5 Ajax

Mit einer Reihe von Web-Technologien, die unter dem Begriff „Ajax“ (Asynchronous JavaScript and XML) zusammengefasst sind, lassen sich per JavaScript (s. Abschnitt 3.1.3) während der Laufzeit einer Webseite dynamische Inhalte hinzuladen, ohne dass die komplette Seite neu geladen werden muss [**20**]. Um dies zu erreichen, können serverseitige Skripte per Ajax-Request ausgeführt werden. Dies geschieht wie beim üblichen Aufruf einer Webseite mit Hilfe einer HTTP-Anfrage an den Server. Parameter der Anfrage können wie gewohnt im HTTP-Header übergeben werden.

Üblicherweise ist die Rückgabe einer Ajax-Anfrage ein String in der JavaScript Object Notation (JSON), mit welchem Arrays und Objekte sehr praktikabel als Zeichenkette angegeben werden können [**21**]. Dieser kann vom anfragenden JavaScript maschinell interpretiert und ausgewertet werden. Entsprechende Ergebnisse können daraufhin von JavaScript dynamisch in das HTML-Dokument eingefügt werden, um sie dem Nutzer auf der Webseite verfügbar zu machen.

3.1.6 MySQL

MySQL ist eine Variante der Datenbank-Abfragesprache Structured Query Language (SQL) und ein auf Webservern eingesetzter Datenbanktyp. Es handelt sich bei MySQL um ein relationales Datenbanksystem, welches Daten in Tabellen speichert [**16**]. In der Architektur von StudMap14 sorgt die MySQL-Datenbank für die Speicherung der Benutzerdaten und der Informationen zu den hochgeladenen Geodaten eines Benutzers.

3.1.7 GDAL

Die plattformunabhängige Open-Source-Bibliothek „Geospatial Data Abstraction Library“ (GDAL) bietet Funktionen zur Anzeige von Metadaten, Manipulation und Konvertierung räumlicher Rasterdaten [**6**]. Kommandos aus der Befehlsbibliothek der GDAL werden per Konsolenbefehl durchgeführt. Hierdurch ist es möglich, Befehle mit Hilfe der Funktion `shell_exec()` direkt aus dem PHP-Code der Webseite heraus auszuführen (s. Abschnitt 3.2.1.1) und hochgeladene Geodaten eines Nutzers in Formate zu konvertieren, die von OpenLayers (s. Abschnitt 3.1.3.3) unterstützt werden, um sie daraufhin im Kartenfenster der Anwendung als Layer zu visualisieren.

3.1.8 GeoServer

Die Open-Source-Software „GeoServer“ dient der Bearbeitung und dem Austausch von Geodaten über das Internet. Hiermit lassen sich Geodaten in Raster- oder Vektorform mittels der Open Geospatial Consortium (OGC) Standards Web Map Service (WMS), Web Coverage Service (WCS) und Web Feature Service (WFS) veröffentlichen [**4**]. Diese Technologie wird genutzt, um für StudMap14 Raster-Geodaten, aufbereitet in Form von WMS-Layern, bereitzustellen. Dies ist von Vorteil, da die Bildverarbeitung und die Transformation der Rasterdaten in das von StudMap genutzte räumliche Referenzsystem von der GeoServer-Software übernommen wird, denn eine Einbindung von Rasterdaten als OpenLayers Image-Layer stellt sich als schwierig dar (s. Abschnitt 3.2.1.1).

3.2 Funktionserweiterungen und Modifikationen

In diesem Abschnitt wird detailliert auf die Implementierung der einzelnen neuen Funktionen und Features von StudMap14 eingegangen. Wichtige Quellcodeausschnitte illustrieren die programmiertechnische Umsetzung der Funktionen.

3.2.1 Erweiterung der unterstützten Datenformate

Die Erweiterung der Anzahl der unterstützten Datenformate ist ein wesentlicher Teil der Erhöhung der Interoperabilität von StudMap14. Je mehr Datenformate in StudMap visualisiert werden können, desto mehr Nutzer können die Plattform sinnvoll für ihre Zwecke nutzen.

3.2.1.1 GeoTiff

Die grundlegenden Funktionen zum Einbinden von Geodaten waren bereits vor der Erweiterung von StudMap14 vorhanden. Nachdem der Nutzer per Klick auf den Button zum Upload von Vektordaten ein Formular anzeigen lässt und dieses abschickt, wird es per Ajax-Anfrage an den Server gesendet. Dieser speichert die Daten und liefert den angegebenen Layernamen und den Ablageort in Form des Verzeichnisses

und des Dateinamens an den Client zurück, der den Layer visualisiert, indem ein `OpenLayers.Layer.Vector` Objekt mit entsprechenden Stileigenschaften erstellt und an die Karte weitergegeben wird.

Das GeoTiff-Format wird nicht direkt von OpenLayers unterstützt, da dessen Spezifikationen zwar frei verfügbar sind, es sich aber um ein ursprünglich von der „Aldus Corporation“ entwickeltes [**22**] proprietäres Dateiformat handelt, welches Patentrechten bei der Kompression unterliegt [**23**]. Aus diesem Grund kann eine Einbindung von GeoTiff-Rasterlayern nicht direkt durch Hinzufügen eines `OpenLayers.Layer.Image` Objekts zur Karte vorgenommen werden. Stattdessen müssen einige Vorverarbeitungsschritte durchgeführt werden, um dies zu erzielen. Abbildung 3 zeigt eine Übersicht über den kompletten Prozess zur Einbindung von GeoTiff-Dateien, der im Folgenden weiter beschrieben wird.

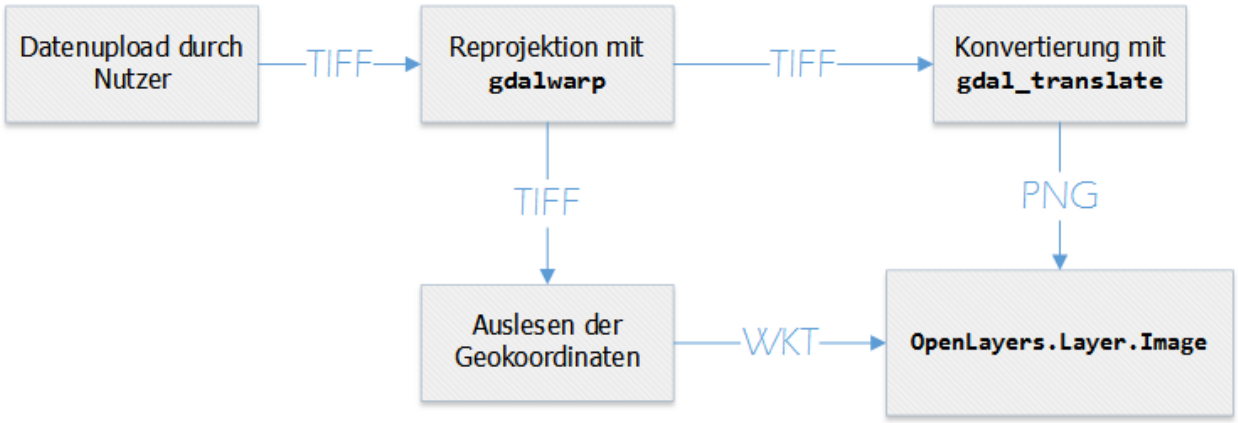

Abbildung 3 *Diagramm zum Prozess der Einbindung von GeoTiff-Dateien vom Dateiupload durch den Nutzer bis zur Erstellung des OpenLayers.Layer.Image Objekts*

Nachdem die Datei auf den Server hochgeladen und in ein zufällig generiertes Unterverzeichnis des `uploads` Verzeichnisses kopiert wurde, wird per PHP-Funktion `shell_exec()` das Programm `gdalwarp` ausgeführt.

```
gdalwarp old_file new_file -t_srs
```

Dieser Befehl soll eine Umprojektion der Datei in das Google-Referenzsystem (EPSG-Code 900913) durchführen, das von StudMap14 als Kartenprojektion verwendet wird. Hierzu wird dem Parameter `-t_srs` (Target Spatial Reference System) der folgende PRF-String [**24**] zur genauen Spezifikation des Referenzsystems angehängt, in welches transformiert werden soll. Tabelle 2 erklärt die folgenden Parameter der Referenzsystemspezifikation. Da $a = b$, handelt es sich beim Referenzellipsoid um eine Kugel.

```
"+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0
+units=m +nadgrids=@null +no_defs towgs84=0,0,0"
```

Die neu erstellte Datei wird mit dem Namenszusatz `_warped` im gleichen Verzeichnis gespeichert. Nach der Umprojektion wird die umprojizierte TIFF-Datei in das PNG-Format konvertiert, um Kompatibilität mit `OpenLayers.Layer.Image` herzustellen. Dies geschieht dem GDAL-Kommando `gdal_translate`.

```
gdal_translate -of PNG –scale -co worldfile=no new_file old_file
```

Da die jetzt erstellte PNG-Datei keine Geoinformationen mehr enthält, müssen diese aus dem reprojizierten GeoTiff ausgelesen werden. Hierfür kommt der Befehl `gdalinfo` zum Einsatz, der die GeoTiff-Headerinformationen anzeigt.

```
gdalinfo filename
```

Tabelle 2: Parameter der Referenzsystemspezifikation (Quellen: [24] und [25])

Parameter	**Wert**	**Erklärung**	
`proj`	`merc`	als Basis zu verwendende Projektion (Mercator, zylindrisch)	
`a`	`6378137`	Länge der großen Halbachse des Referenzellipsoid	
`b`	`6378137`	Länge der kleinen Halbachse des Referenzellipsoid	
`lat_ts`	`0°`	*latitude of true scale* geographische Breite, auf welcher Distanzmessungen korrekt sind	
`lon_0`	`0°`	zentraler Meridian	
`x_0`	`0`	*false easting*	
`y_0`	`0`	*false northing*	
`k`	`1`	Skalierungsfaktor zur Vermeidung von Verzerrungen	
`units`	`m`	Längeneinheit (Meter)	

Eine beispielhafte Ausgabe des Befehls zeigt Quellcode 1. Alle für die weitere Verarbeitung wichtigen Informationen stehen in den letzten fünf Zeilen der Ausgabe. Zur Darstellung des Bildes mit OpenLayers wird die Bounding Box benötigt, welche durch die Geokoordinaten der unteren linken und der oberen rechten Ecke des Bildes definiert wird. Diese werden mit Hilfe der hierfür geschriebenen Funktion `getStringWithinStrings($start, $end, $text)` ermittelt, die den String zwischen zwei definierten Strings aus einem weiteren String extrahiert. Als Start wird `"Lower Left ("` beziehungsweise `"Upper Right ("` und als Ende `")"` verwendet. Mit der Funktion `trim()` werden überflüssige Leerzeichen entfernt, welche die weitere Auswertung der Zahlen erschweren könnten.

Quellcode 1: Beispielhafte Ausgabe des gdalinfo Befehls

```TXT
Driver: GTiff/GeoTIFF
Files: 401110gs_warped.tif
Size is 6407, 6424
Coordinate System is:
PROJCS["unnamed",
    GEOGCS["unnamed ellipse",
        DATUM["unknown",
            SPHEROID["unnamed",6378137,0]],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433]],
    PROJECTION["Mercator_1SP"],
    PARAMETER["central_meridian",0],
    PARAMETER["scale_factor",1],
    PARAMETER["false_easting",0],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]]]
Origin = (849445.212481540050000,6797279.855430781800000)
Pixel Size = (0.514943447748573,-0.514943447748573)
Metadata:
  AREA_OR_POINT=Area
  TIFFTAG_ARTIST=(c) Landesvermessungsamt Nordrhein-Westfalen
  TIFFTAG_DATETIME=03.03.2008 11:44:08
  TIFFTAG_IMAGEDESCRIPTION=4011/10 3406 5760 M³nster Nordost
  TIFFTAG_MAXSAMPLEVALUE=1
  TIFFTAG_MINSAMPLEVALUE=0
  TIFFTAG_RESOLUTIONUNIT=2 (pixels/inch)
  TIFFTAG_XRESOLUTION=400
  TIFFTAG_YRESOLUTION=400
Image Structure Metadata:
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  (  849445.212, 6797279.855) (  7d37'50.51"E, 51d59' 3.33"N)
Lower Left  (  849445.212, 6793971.859) (  7d37'50.51"E, 51d57'57.43"N)
Upper Right (  852744.455, 6797279.855) (  7d39'37.20"E, 51d59' 3.33"N)
Lower Right (  852744.455, 6793971.859) (  7d39'37.20"E, 51d57'57.43"N)
Center      (  851094.834, 6795625.857) (  7d38'43.85"E, 51d58'30.39"N)
```

Die Koordinaten können nun zusammen mit dem Dateipfad und -namen der reprojizierten PNG-Datei und dem Layernamen als JSON-String an das aufrufende JavaScript zurückgegeben werden. Falls die `gdalinfo` Ausgabe keine Koordinaten in Form einer Bounding Box enthält, wird `error` als Rückgabe geliefert. Der vollständige serverseitige Ablauf des Uploads, der Umprojektion und der Konvertierung kann in Quellcode 2 eingesehen werden.

Ist die Rückgabe des JSON-Strings an das aufrufende JavaScript erfolgt, so liest dieses den String und prüft, ob Koordinaten aus der `gdalinfo` Ausgabe ermittelt werden konnten. Falls nicht, wird dem Nutzer mitgeteilt, dass das Referenzsystem seiner GeoTiff-Datei nicht unterstützt wird. Sind die Koordinaten erfolgreich ermittelt worden, wird die Ausgabe an die Funktion `createStudmapRasterLayer(name, path, box)` weitergeleitet, die einen `OpenLayers.Layer.Image` erzeugt, der der Karte hinzugefügt werden kann.

Quellcode 2: Upload, Umprojektion und Konvertierung von GeoTiff-Rasterdateien

file-upload.php — *PHP*

```php
function getStringWithinStrings($start, $end, $text) {
    if ( true == ( $startpos = strpos($text, $start) ) ) {
        if ( true == ( $endpos = strpos($text, $end, $startpos) ) ) {
            $spos = $startpos + strlen($start);

            return substr($text, $spos, $endpos - $spos);
        }
    }
    return false;
}

[...]

$ext = pathinfo($_FILES['gpx-track']['name'], PATHINFO_EXTENSION);

[...]

else if ( strtolower($ext) == 'tif' ) {
    $filename = str_replace('.'.$ext, '', $_FILES['gpx-track']['name']);

    [...]

    $warp = shell_exec('gdalwarp  uploads/'.$random.'/'.$filename.'.'.$ext.'
       uploads/'.$random.'/'.$filename.'_warped.'.$ext.'
       -t_srs
              "+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0
               +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +no_defs
               towgs84=0,0,0"');

    $translate = shell_exec('gdal_translate -of PNG -scale
       -co worldfile=no uploads/'.$random.'/'.$filename.'_warped.'.$ext.'
       uploads/'.$random.'/'.$filename.'_warped.png');

    $gdalinfo = shell_exec('gdalinfo -proj4
       '.$serverPath.'uploads/'.$random.'/'.$filename.'_warped.'.$ext);

    $lowerleft = trim(getStringWithinStrings(
       'Lower Left  (', ')', $gdalinfo));

    $upperright = trim(getStringWithinStrings(
       'Upper Right (', ')', $gdalinfo));

    $bbox = ( $lowerleft != '' && $upperright != '' ) ?
       $lowerleft.', '.$upperright : 'error';

    [...]

    echo '{"success":true,
       "path":"uploads/'.$random.'/'.$filename.'_warped.png",
       "file":"'.$_FILES['gpx-track']['name'].'",
       "name":"'.$_POST['gpx-name'].'",
       "bbox":"'.$bbox.'"}';
}
```

Die Funktion wertet die Koordinatenzeichenkette aus, indem sie sie an den Kommas auftrennt und die einzelnen Koordinatenbestandteile (`left`, `bottom`, `right`, `top`) an ein neues `OpenLayers.Bounds` Objekt übergibt. Dieses wiederum wird zusammen mit dem Layernamen und dem Dateipfad zur PNG-Datei an ein als Rückgabewert der Funktion dienendes `OpenLayers.Layer.Image` Objekt mit der Google-Projektion (EPSG-Code 900913) als Referenzsystem übergeben. Quellcode 3 zeigt die Auswertung der serverseitigen Rückgabe und die Erstellung des Layers.

Quellcode 3: Auswertung der Rückgabe des GeoTiff-Uploads

lib/studmap/upload.js — *JavaScript*

```
fp.getForm().submit({
    url: Paths.fileUpload,
    waitMsg: 'Uploading data ...',
    success: function (fp, o) {
        window.close();
        if (o.result.coords != 'error') {
            msg('Success', 'Processed file "' + o.result.file + '"
              on the server.');
            var path = Paths.fileUploadResultPrefix + o.result.path;
            map.addLayer(createStudmapRasterLayer(o.result.name, path,
              o.result.bbox));

            [...]
        }
        else {
            msg('Error', 'Reference System not supported.');
        }
    }
});

[...]

function createStudmapRasterLayer(name, path, bbox) {
    bbox = bbox.split(', ');
    bbox = new OpenLayers.Bounds(bbox[0], bbox[1], bbox[2], bbox[3]);

    var layer = new OpenLayers.Layer.Image(
        name, path, bbox,
        new OpenLayers.Size(0, 0),
        {
            projection: new OpenLayers.Projection("EPSG:900913"),
            isBaseLayer: false
        }
    );
    return layer;
}
```

Bis der hier beschriebene Ablauf zur Einbindung von GeoTiff-Dateien implementiert war, mussten einige Schwierigkeiten überwunden werden. In einem ersten Versuch sollte die aus der `gdalinfo`-Ausgabe ausgelesene WGS84-Bounding-Box mit Hilfe der JavaScript-Bibliothek Proj4js [**26**] in das Google-Referenzsystem transformiert werden. Dies resultierte in einer um einige hundert Meter nach Nordosten verschobenen Ausgabe des GeoTiff auf der Karte (s. Abbildung 4).

Abbildung 4 *Verschiebung der Darstellung eines GeoTiff-Rasterlayers der Münsteraner Stadtteile St. Mauritz und Rumphorst um einige hundert Meter Richtung Nord-Osten. Die schwarzen Linien stellen die eigentlichen Grenzen des GeoTiff dar.*

Ein weiterer Versuch begann damit, einen in Delphi geschriebenen Algorithmus zur Transformation von Gauß-Krüger Koordinaten in geographische Koordinaten [**27**] in JavaScript zu übersetzen, um so zumindest GeoTiffs zu unterstützen, die in Gauß-Krüger georeferenziert sind. Dieser Algorithmus lieferte jedoch nach der Umwandlung der vom Algorithmus zurückgegebenen geographischen Koordinaten in das Google-Referenzsystem gleiche Ergebnisse wie der Algorithmus aus der Proj4Js-Bibliothek. Die Verschiebung um einige hundert Meter konnte korrigiert werden, indem zusätzlich ein aus [**28**] entnommener Algorithmus zur Umwandlung des Ellipsoid-Systems implementiert wurde.

Die nord-östlichen und süd-westlichen Eckpunkte des GeoTiff lagen nun auf den für sie vorgesehenen Koordinaten. Jedoch musste das Bild zusätzlich um einen kleinen Winkel rotiert und leicht verzerrt werden, um auch die nord-westliche und süd-östliche Ecke an die richtige Position zu bewegen. Die Rotation oder Verzerrung eines Image-Layers ist in OpenLayers allerdings nicht vorgesehen und nur über Umwege machbar, welche weitere Schwierigkeiten mit sich brachten, auf die an dieser Stelle nicht weiter eingegangen werden soll. Aus diesem Grund musste dieser Ansatz verworfen werden und eine Reprojektion des Bildes mit Hilfe von `gdalwarp` vor der Einbindung mit OpenLayers geschehen. Der Vorteil dieser Methode liegt darin, dass das Referenzsystem des GeoTiff automatisch erkannt wird und es nach der Reprojektion immer im gleichen Referenzsystem vorliegt, was eine weitere Verarbeitung erleichtert.

An dieser Stelle soll eine Anmerkung zu GeoTiff-Dateien mit Geoinformationen in einer zusätzlichen TFW-Datei gemacht werden, deren Unterstützung, wie bereits in Abschnitt 2.3 erwähnt, nicht erreicht werden konnte. Um diese Art von GeoTiff zu unterstützen, ist zur Einbindung ein anderes Vorgehen als bei GeoTiff-Dateien mit Geoinformationen im Datei-Header notwendig. Die Umprojektion kann nicht über das Programm `gdalwarp` durchgeführt werden, da das Quellreferenzsystem durch die fehlenden Informationen im Header nicht automatisch vom `gdalwarp`-Programm ausgelesen werden kann. Stattdessen müsste es manuell ermittelt und als PRF-String [**24**] im Parameter `s_srs` des `gdalwarp`-Befehls spezifiziert werden. Dies steigert den Aufwand enorm, weil für jedes mögliche Quellreferenzsystem ein entsprechender PRF-String bereitliegen müsste. Deshalb konnte die Unterstützung dieser Art von GeoTiff nicht erreicht werden.

Ein zusätzliches Problem, das bisher nicht gelöst werden konnte, tritt bei der Darstellung von GeoTiff-Rasterlayern mit Mozilla Firefox auf. Nach dem Einbinden eines GeoTiff-Layers wird das Bild einmal an der richtigen Position angezeigt. Wird jetzt allerdings die Zoomstufe des Kartenausschnittes vom Nutzer verändert oder der Layer deaktiviert und wieder aktiviert, verschwindet der Layer und kann nur durch Aktualisierung der Webseite wiederhergestellt werden. In allen anderen getesteten Browsern (Google Chrome, Opera und Windows Internet Explorer) konnten GeoTiff-Dateien ohne Probleme konsistent dargestellt werden.

3.2.1.2 Geography Markup Language

Vektordateien des auf der Extensible Markup Language (XML) basierenden und vom OGC entwickelten Open Source Formates „Geography Markup Language“ (GML) werden von OpenLayers direkt unterstützt. Aus diesem Grund ist es relativ einfach möglich, die Unterstützung dieses Formats durch StudMap14 zu erreichen. Die bereits vorhandene Funktion `createStudmapVectorLayer(name, path)` unterstützt die Geodatenformate GPX und KML, indem sie einen `OpenLayers.Layer.Vector` erzeugt und diesem das Format `OpenLayers.Format.GPX` beziehungsweise `OpenLayers.Format.KML` zuweist. Diesen Zuweisungen kann einfach im Falle einer hochgeladenen GML-Datei das Format `OpenLayers.Format.GML` hinzugefügt werden. Zusätzlich ist eine Anpassung der Auswertung des Upload-Formulars notwendig, um Fehlermeldungen beim Versuch des Uploads von GML-Dateien zu vermeiden. Die Veränderungen zur Unterstützung des GML-Formates können in Quellcode 4 eingesehen werden.

Quellcode 4: Veränderungen zur Unterstützung des GML-Formats

lib/studmap/upload.js — *JavaScript*

```
if (suffix == "gpx" || suffix == "kml" || suffix == "gml") {
    fp.getForm().submit({
        url: Paths.fileUpload,
        waitMsg: 'Uploading data ...',
        success: function (fp, o) {
            window.close();
            msg('Success', 'Processed file "' + o.result.file + '"
              on the server.');
```

```
            var path = Paths.fileUploadResultPrefix + o.result.path;
            map.addLayer(createStudmapVectorLayer(o.result.name, path));

            [...]
        }
    });
}

[...]

function createStudmapVectorLayer(name, path) {

[...]

    switch (suffix) {
        case 'gpx':
            format = new OpenLayers.Format.GPX();
            break;
        case 'kml':
            format = new OpenLayers.Format.KML();
            break;
        case 'gml':
            format = new OpenLayers.Format.GML();
            break;
    }

[...]

}
```

3.2.1.3 ESRI Shapefile

Die Unterstützung des ESRI-Vektordatenformates „Shapefile“ (SHP) [**12**] gestaltet sich schwieriger als die des GML-Formates, da eine Shapefile-Unterstützung von OpenLayers nicht vorgesehen ist. Abbildung 5 zeigt den Prozess zur Einbindung von Shapefiles schematisch. Es muss (ähnlich wie in Abschnitt 3.2.1.1 für GeoTiff-Rasterdaten beschrieben) eine Konvertierung in ein von OpenLayers unterstütztes Format durchgeführt werden, in diesem Fall in das KML-Format. Dies geschieht mit dem GDAL-Kommandozeilenbefehl `ogr2ogr` [**29**].

```
ogr2ogr -f KML new_file old_file
```

Im Zusammenhang mit ESRI Shapefiles gibt es jedoch noch eine weitere Hürde, die vor der Konvertierung überwunden werden muss. Da Shapefiles aus mehreren zusammengehörigen Dateien bestehen, ist es notwendig, einen benutzerfreundlichen Weg zur Durchführung des Uploads zu finden, sodass die Dateien nicht einzeln hochgeladen werden müssen. Dieses Problem ist lösbar, indem die Dateien vom Nutzer in einem ZIP-Container verpackt und so als einzelne Datei hochgeladen werden. Dieses ZIP-Archiv kann nach dem Upload auf dem Server automatisch mit Hilfe von PHP wieder entpackt und daraufhin gelöscht werden. Damit der Nutzer das ZIP-Archiv nicht als Container für Dateien verwendet, die dem Server Schaden zufügen könnten, werden alle nicht benötigten Dateien (identifiziert anhand ihrer Dateiendung) ebenfalls gelöscht.

Um die fehlerlose Konvertierung durch `ogr2ogr` zu gewährleisten, müssen nun alle zusammengehörigen Dateien den gleichen Dateinamen erhalten. Hierzu werden der Dateiname der SHP-Datei ausgelesen und alle Sonderzeichen aus dem Dateinamen entfernt, um nicht unterstützte Sonderzeichen als Fehlerquelle bei der Konvertierung auszuschließen. Dieser ausgelesene Dateiname findet Verwendung als neuer Name für alle weiteren Dateien, welche dementsprechend umbenannt werden.

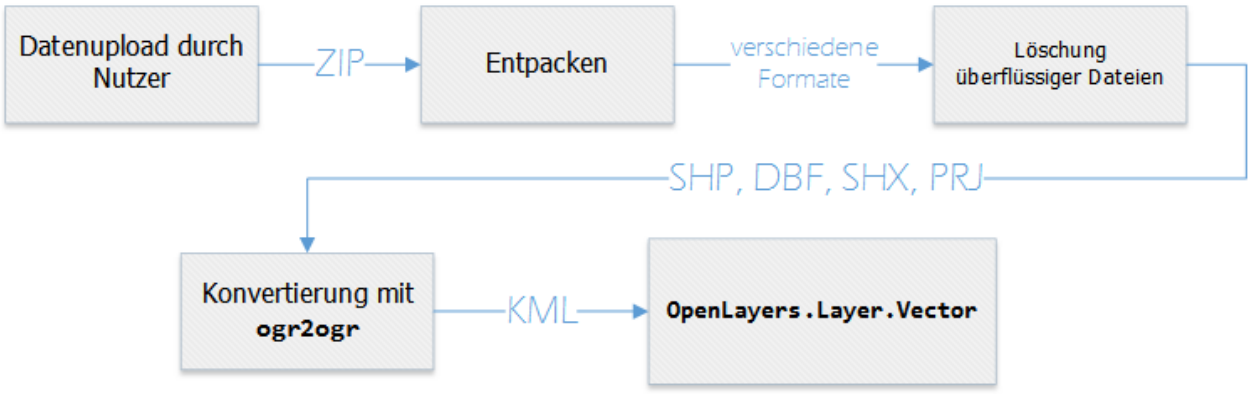

Abbildung 5 *Diagramm zur Verarbeitung hochgeladener ZIP-Archive mit ESRI Shapefiles. Nach dem ZIP-Dateiupload durch den Nutzer wird das Archiv entpackt und überflüssige Dateien gelöscht. Die verbleibenden Dateien (SHP, DBF, SHX und PRJ) werden in eine KML-Datei konvertiert, die als Basis für ein OpenLayers.Layer.Vector Objekt dient.*

Zusätzlich wird überprüft, ob die Anzahl der verbliebenen Dateien „vier“ beträgt. Da alle nicht benötigten Dateien gelöscht wurden, können nur noch die vier Dateiformate SHP (Hauptdatei), DBF (Datenbankdatei), SHX (Indexdatei) [**12**] und PRJ (Projektionsdatei) übrig geblieben sein. Ist die Zahl kleiner, so fehlt mindestens eine der benötigten Dateien. Auch wird überprüft, ob jedes der Dateiformate nur einmal vorkommt. Eine Projektionsdatei ist zwar im Allgemeinen optional, für StudMap14 allerdings obligatorisch, da eine einwandfreie Darstellung von Shapes auf der Karte ohne Beschreibung der verwendeten Projektion nicht gewährleistet werden kann.

Nach der Konvertierung des Shapefiles in das KML-Format gibt der Server dem anfragenden JavaScript den Dateipfad, den Dateinamen der KML-Datei und den Layernamen als JSON-String zurück. Da die Darstellung von KML-Dateien in StudMap14 bereits implementiert ist, kann ein neuer `OpenLayers.Layer.Vector` ohne großen Aufwand mit der bereits vorhandenen Funktion `createStudmapVectorLayer(name, path)` erstellt und der Karte beziehungsweise der Layerliste hinzugefügt werden. Quellcode 5 stellt den serverseitigen Prozess zur Einbindung von Shapefiles dar.

Quellcode 5: ESRI Shapefile Upload

file-upload.php PHP

```php
$ext = pathinfo($_FILES['gpx-track']['name'], PATHINFO_EXTENSION);
if ( strtolower($ext) == 'zip' ) {
    $zip = new ZipArchive;
    $res = $zip->open('uploads/'.$random.'/'.$_FILES['gpx-track']['name']);

    if ($res === true) {
        $zip->extractTo('uploads/'.$random.'/');
        $zip->close();
        unlink('uploads/'.$random.'/'.$_FILES['gpx-track']['name']);
        $numberOfFiles = 0;
        $shapeName = '';
        $allowedFileExtensions = array('shp', 'dbf', 'shx', 'prj');
        $dir = 'uploads/'.$random.'/';

        if ( $dirRes = opendir($dir) ) {
            while ( ( true == $file = readdir($dirRes) ) ) {
                if ( ! in_array($file, array('.', '..') ) && !
                  is_dir($dir.$file) ) {
                    $fileExt = pathinfo($file, PATHINFO_EXTENSION);
                    if ( ! in_array($fileExt, $allowedFileExtensions) ) {
                        unlink('uploads/'.$random.'/'.$file);
                    }
                    else {
                        unset($allowedFileExtensions[
                            array_search($fileExt,
                            $allowedFileExtensions)]);
                            $numberOfFiles++;
                        if ( $fileExt == 'shp' ) {
                            $shapeName = str_replace('.shp', '', $file);
                            $shapeName = preg_replace('/[^A-Za-z]+/', '',
                                  $shapeName);
                        }
                    }
                }
            }
        }
        if ( $dirRes = opendir($dir) ) {
            while ( ( $file = readdir($dirRes) ) !== false ) {
                if ( ! in_array($file, array('.', '..') ) && !
                    is_dir($dir.$file) ) {
                    $fileExt = pathinfo($file, PATHINFO_EXTENSION);
                    rename('uploads/'.$random. '/'.$file,
                           'uploads/'.$random.'/'.$shapeName.'.'.$fileExt);
                }
            }
        }
        if ( $numberOfFiles == 4 ) {
            exec('ogr2ogr -f KML uploads/'.$random.
              '/'.$shapeName.'.kml uploads/'.$random.'/'.$shapeName.'.shp ');
        }

        [...]

        echo '{"success":true, "msg":"'.$numberOfFiles.'",
             "path":"uploads/'.$random.'/'.$shapeName.'.kml",
             "file":"'.$shapeName.'.kml", "name":"'.$_POST['gpx-name'].'"}';
    }
}
```

3.2.2 Benutzer-Funktionen

Unter dem Begriff „Benutzer-Funktionen“ sind hier alle umgesetzten Features erfasst, die den persistenten Geodaten-Upload für einen bestimmten Benutzer ermöglichen. Um dies umzusetzen, müssen Funktionen zur Erstellung eines neuen Benutzerkontos (Registrierung) und zum Login beziehungsweise Logout vorhanden sein. Auch vor der Erweiterung von StudMap14 im Rahmen dieser Arbeit wurden hochgeladene Geodaten bereits persistent auf dem Server gespeichert. Aus diesem Grund muss lediglich eine Möglichkeit gefunden werden, die Ablageverzeichnisse der Geodaten auf dem Server einem bestimmten Benutzer zuzuordnen und die Daten dem Nutzer dementsprechend bei erneutem Login zur Verfügung zu stellen.

Die Umsetzung der drei Funktionen Login, Logout und Registrierung gliedert sich jeweils in drei Teile.

- grafische Benutzerschnittstelle (GUI)
- Behandlung der Benutzereingaben per JavaScript
- serverseitige Überprüfung und Datenbankanbindung

Zur Umsetzung der grafischen Benutzerschnittstelle für die Benutzer-Funktionen ist zunächst das Anlegen eines neuen Reiters oberhalb der Karte notwendig. Hierbei handelt es sich um eine `Toolbar` aus der Ext-JS-Bibliothek, welche dem `item` Array des `TabPanel` hinzugefügt wird, das bisher die bereits vorhandenen Reiter „Data“ und „Visualization“ enthält (s. Quellcode 6).

Quellcode 6: Einfügung der Benutzer-Toolbar in das TabPanel

lib/studmap/ui.js — *JavaScript*

```
var panel = new Ext.TabPanel({
       enableTabScroll : true,
       activeTab : 0,

       items : [ new Ext.Toolbar({
              title : 'Data',
              items : createDataToolbarItems(mapPanel, activeL,
                     accordionPanel)
       }), new Ext.Toolbar({
              title : 'Visualization',
              items : createVisualizationToolbarItems(mapPanel, activeL)
       }), new Ext.Toolbar({
              title: 'User',
              items: createUserToolbarItems(mapPanel, activeL,
                     layerPanel)
       })]
});
```

Dieser Toolbar wiederum wird eine zweispaltige `ButtonGroup` mit den drei Buttons „Login“, „Logout“ und „Sign up“ mit entsprechenden Icons zur Visualisierung der Funktionalität hinzugefügt. Der Logout-Button bleibt deaktiviert, so lange kein Benutzer eingeloggt ist. Das Verhalten der Applikation bei Benutzung der Buttons wird direkt mit einer jeweiligen Handler-Funktion in der Definition der Buttons angegeben. Ein

Klick auf „Login“ oder „Registrierung“ führt zur Öffnung eines neuen Fensters mit einem entsprechenden Formular. Die Betätigung des Logout-Buttons resultiert im umgehenden Logout des Benutzers.

3.2.2.1 Login

Das Login-Formular enthält die zwei Textfelder für den Benutzernamen und das Passwort. Zum Absenden des Formulars ist ein Login-Button vorhanden. Zudem kann das Fenster über einen Abbruch-Button wieder geschlossen werden. Nach der Eingabe der Benutzerdaten startet das JavaScript in der Handler-Funktion des Login-Buttons eine Ajax-Anfrage an den Server, um die Korrektheit der Benutzerdaten zu überprüfen. Die Benutzerdaten werden auf dem Server in einer MySQL-Datenbanktabelle mit folgendem Aufbau gespeichert (s. Quellcode 7).

Quellcode 7: Aufbau der Benutzer-Datenbanktabelle

user.sql — *MySQL*

```mysql
CREATE TABLE IF NOT EXISTS `user` (
  `id` int(15) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `mail` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
);
```

Die Überprüfung der eingegebenen Benutzerdaten erfolgt auf dem Server mit Hilfe eines PHP-Scripts (s. Quellcode 8), das Verbindung zur MySQL-Datenbank aufbaut und eine Auswahl-Abfrage mit dem Benutzernamen und dem Passwort auf der Benutzertabelle ausführt. Wird ein entsprechender Datensatz gefunden, sind die Benutzerdaten korrekt und es kann eine Bestätigung gesendet werden, ansonsten wird eine Fehlermeldung zurückgegeben.

Quellcode 8: Überprüfung der Benutzerdaten nach dem Login (Server)

login.php — *PHP*

```php
$username = ( isset($_POST['username']) ) ?
mysql_real_escape_string($_POST['username']) : '';
$password = ( isset($_POST['password']) ) ? $_POST['password'] : '';

$encPassword = ( isset($_POST['enc']) && $_POST['enc'] == 'true' ) ?
$password : sha1($password);

include('mysqlconnect.php');
$query = mysql_fetch_object(mysql_query('SELECT id FROM user WHERE username =
\''.$username.'\' AND password = \''.$encPassword.'\' LIMIT 1'));

$msg = ( isset($query->id) ) ? 'success' : 'failure';

echo '{"success":true, "msg":"'.$msg.'", "username":"'.$username.'",
"password":"'.$encPassword.'"}';
```

Wenn eine Erfolgsmeldung an das JavaScript (s. Quellcode 9) gesendet worden ist, werden zwei Cookies im Browser gesetzt, die den Benutzernamen und das verschlüsselte Passwort des Benutzers enthalten und nach 14 Tagen ablaufen. Um das Panel mit der persönlichen Layerliste (s. Abschnitt 3.2.2.4) anzuzeigen und die Benutzer-`ButtonGroup` dem eingeloggten Zustand anzupassen, muss die komplette Anwendung mit Hilfe des Befehls `location.reload()` neu geladen werden. Schlägt die Überprüfung der Benutzerdaten vom Server fehl, gibt die Anwendung eine Fehlermeldung an den Nutzer aus.

Quellcode 9: Auswertung der Benutzerdatenüberprüfung

lib/studmap/ui.js — *JavaScript*

```
if (o.result.msg == "success") {
    var expires = new Date(new Date().getTime() + 86400000 * 14);
    document.cookie = "username=" + o.result.username + ";
       path=/;expires=" + expires.toGMTString();
    document.cookie = "password=" + o.result.password + ";
       path=/;expires=" + expires.toGMTString();

    location.reload();
}
else if (o.result.msg == "failure") {
    msg('Login failed', 'Login failed for user ' + o.result.username +
       '.<br>Incorrect combination of username and password.');
}
```

3.2.2.2 Logout

Der Logout wird ohne weitere Nachfrage direkt nach einer Betätigung des Logout-Buttons durchgeführt. Hierzu werden die Cookies `username` und `password` mit neuen Cookies des gleichen Namens überschrieben, deren Ablaufdatum allerdings auf eine Sekunde vor Ausführung des Skripts gesetzt wird. Dies bedeutet, dass die Cookies umgehend automatisch vom Browser gelöscht werden. Weiterhin muss die Beschriftung des Login-Buttons wieder auf „Login“ geändert werden. Der Login- und der Registrierungs-Button werden reaktiviert, der Logout-Button deaktiviert und das Panel zur Anzeige der persönlichen Layer von der linken Seite der Anwendung entfernt. Die Implementierung der Logout-Funktion zeigt Quellcode 10.

Quellcode 10: Behandlung eines Klicks auf den Logout-Button

lib/studmap/ui.js — *JavaScript*

```
var expires = new Date(new Date().getTime() - 1);
document.cookie = "username=; path=/;expires=" + expires.toGMTString();
document.cookie = "password=; path=/;expires=" + expires.toGMTString();
button_login.setText("Login");
buttongroup.setWidth(130);
button_login.setDisabled(false);
button_signup.setDisabled(false);
button_logout.setDisabled(true);
layerPanel.remove(1);
layerPanel.doLayout();
```

3.2.2.3 Registrierung

Zur Registrierung werden dem Benutzer ein Formular mit den Feldern „Benutzername“, „E-Mail Adresse“ und „Passwort“ und ebenfalls zwei Buttons („Registrieren“ und „Abbrechen“) angezeigt. Die Überprüfung der Eingabe einer korrekt formatierten E-Mail Adresse wird mit Hilfe eines regulären Ausdrucks (entnommen aus [**30**]) direkt in der JavaScript Handler-Funktion (s. Quellcode 11) durchgeführt. Zudem wird festgestellt, ob das eingegebene Passwort mit der Passwortwiederholung übereinstimmt. Die Handler-Funktion sendet die Formulardaten per Ajax-Anfrage an den Server, der per PHP-Script (s. Quellcode 12) einen neuen Benutzer anlegt und wie bereits beim Login entweder eine Erfolgs- oder Fehlermeldung zurücksendet. Die entsprechende Meldung für den Nutzer wird verständlich aufbereitet ausgegeben.

Serverseitig wird eine Verbindung zur MySQL-Datenbank aufgebaut und ein neuer Eintrag in der Benutzertabelle angelegt und mit den entsprechenden Benutzerdaten gefüllt. Ein Fehlschlag wird zurückgegeben, falls die MySQL-Anfrage nicht funktionieren sollte oder der Benutzername beziehungsweise die E-Mail Adresse bereits vergeben sind.

Quellcode 11: Behandlung des Registrierungsformulars

lib/studmap/ui.js — *JavaScript*

```
var re =
/^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/;
if ( ! re.test(Ext.getCmp('mail').getValue()) ) {
    msg('Error.', 'The E-Mail address does not seem to be valid.');
}
[...]
else if (fp.getForm().isValid()) {
    fp.getForm().submit({
        url: Paths.signup,
        waitMsg: 'Signing up ...',
        success: function (fp, o) {
            if (o.result.msg == "success") {
                msg('Registration successful.', 'Thank you for creating a
                    StudMap14 account. You can now log in with your user
                    data.');
                window.close();
            }
            else if (o.result.msg == "failure_username") {
                msg('Error.', 'The given username ist already in use.');
            }
            else if (o.result.msg == "failure_mail") {
                msg('Error.', 'The given E-Mail address ist already in
                    use.');
            }
            else {
                msg('Error.', 'Sorry, something went wrong. Please
                    contact an administrator for more information.');
                window.close();
            }}
    });
}
```

Quellcode 12: Registrierung eines neuen Benutzers (Server)

signup.php *PHP*

```php
$username = ( isset($_POST['username']) ) ?
       mysql_real_escape_string($_POST['username']) : '';
$mail = ( isset($_POST['mail']) ) ? mysql_real_escape_string($_POST['mail'])
       : '';
$password = ( isset($_POST['password']) ) ? sha1($_POST['password']) : '';

include('mysqlconnect.php');

$usernameQuery = mysql_fetch_object(mysql_query("SELECT `id` FROM
       `studmap`.`user` WHERE `username`='".$username."'"));
$mailQuery = mysql_fetch_object(mysql_query("SELECT `id` FROM
       `studmap`.`user` WHERE `mail`='".$mail."'"));
[...]

if ( isset($usernameQuery->id) ) {
    echo '{"success":true, "msg":"failure_username"}';
}
else if ( isset($mailQuery->id) ) {
    echo '{"success":true, "msg":"failure_mail"}';
}
else {
    if ( mysql_query("INSERT INTO  `studmap`.`user` ( `username` ,
       `password` , `mail` ) VALUES ( '".$username."',  '".$password."',
       '".$mail."')") ) {
        echo '{"success":true, "msg":"success"}';
    }
    else {
        echo '{"success":false, "msg":"failure"}';
    }
}
```

3.2.2.4 Persönliche Layer

Der Vorteil, StudMap14 als angemeldeter Benutzer zu verwenden, liegt darin, auf einmal hochgeladene Geodaten jederzeit wieder zugreifen zu können. Diese persönlichen Geodaten werden dem Nutzer nach dem Login in einer persönlichen Layerliste unterhalb der aktiven Layerliste angezeigt. Hinzugefügt werden können persönliche Layer über das Kontextmenü eines aktiven Layers oder durch das Hochladen von Geodaten im angemeldeten Zustand.

Bei der persönlichen Layerliste handelt es sich wie bei der aktiven Layerliste um ein Ext JS `TreePanel`, welches einen GeoExt `LayerContainer` enthält. Dieses wird mit Layern gefüllt, indem zunächst ein Ajax-Request Informationen zu persönlichen Layer des Nutzers vom Server abfragt. Vektor- und Rasterdaten werden in separaten MySQL-Tabellen gespeichert, da für Rasterdaten neben dem Verzeichnis, dem Dateinamen und dem Layernamen auch die Bounding Box des Datensatzes in der Datenbank abgelegt ist. Den Aufbau der Datenbanktabellen für die Geodaten zeigen Quellcode 13 und Quellcode 14.

Um alle Layer in der Liste anzuzeigen, wird nun auf jede der Tabellen eine Abfrage ausgeführt, welche alle Datensätze des Benutzers ausgibt (s. Quellcode 15). Diese erhaltenen Daten werden mit Kommata getrennt an das `data`-Element des JSON-Rückgabestrings angehängt. Das Komma am Ende des letzten Datenelements wird nachträglich mit Hilfe der PHP-Funktion `rtrim()` entfernt.

Quellcode 13: Aufbau der Rasterdatentabelle

rasterdata.sql — *MySQL*

```
CREATE TABLE IF NOT EXISTS `rasterdata` (
  `user` int(15) NOT NULL,
  `dir` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `filename` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `layername` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `bbox` text NOT NULL,
  `sharedby` int(15) NOT NULL
);
```

Quellcode 14: Aufbau der Vektordatentabelle

rasterdata.sql — *MySQL*

```
CREATE TABLE IF NOT EXISTS `vectordata` (
  `user` int(15) NOT NULL,
  `dir` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `filename` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `layername` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `sharedby` int(15) NOT NULL
);
```

Quellcode 15: Abfrage der persönlichen Layer aus der Datenbank

personallayers.php — *PHP*

```
$data = '';
$dirs = '';
[...]
$query = mysql_query('SELECT dir, filename, layername, sharedby FROM
       vectordata WHERE user = \''.$id.'\'');
while ( $row = mysql_fetch_object($query) ) {
    $filename = pathinfo($row->filename, PATHINFO_FILENAME);
    $fileext = pathinfo($row->filename, PATHINFO_EXTENSION);
    [...]
    $data .= '"'.$row->dir.'":{"dir":"'.$row->dir.'",
       "filename":"'.$filename.'", "fileext":"'.$fileext.'",
       "layername":"'.$row->layername.'", "sharedby":"'.$sharedUser.'"},';
    $dirs .= $row->dir.',';
}
$query = mysql_query('SELECT dir, filename, layername, bbox, sharedby FROM
       rasterdata WHERE user = \''.$id.'\'');
while ( $row = mysql_fetch_object($query) ) {
    $filename = pathinfo($row->filename, PATHINFO_FILENAME);
    $fileext = pathinfo($row->filename, PATHINFO_EXTENSION);
    [...]
    $data .= '"'.$row->dir.'":{"dir":"'.$row->dir.'",
       "filename":"'.$filename.'", "fileext":"'.$fileext.'",
```

```
        "layername":"'.$row->layername.'", "bbox":"'.$row->bbox.'",
        "sharedby":"'.$sharedUser.'"},';
    $dirs .= $row->dir.',';
}
$data = rtrim($data, ',');
$dirs = rtrim($dirs, ',');
[...]
echo '{"success":true, "msg":"'.$msg.'", "dirs":"'.$dirs.'",
       "data":{'.$data.'}}';
```

Zur Auswertung des Rückgabestrings im aufrufenden JavaScript (s. Quellcode 16) wird der `dirs`-String an den Kommata aufgetrennt und jedes so erhaltene Verzeichnis mittels `for`-Schleife durchlaufen. Hier wird anhand der Dateiendung der in diesem Verzeichnis gespeicherten Geodaten-Datei ermittelt, ob es sich um eine Raster- oder Vektordatei handelt und ein Layer des entsprechenden Typs mit Hilfe der Funktionen `createStudmapVectorLayer(name, path)` beziehungsweise `createStudmapRasterLayer(name, path, bbox)` erstellt werden muss. Dieser Layer wird in ein hierfür erzeugtes `layers`-Array eingefügt, das daraufhin als Inhalt für einen `GeoExt.data.LayerStore` dient, welcher wiederum in einem `GeoExt.tree.LayerContainer` verpackt wird. Bei diesem Container handelt es sich nur um eine abstrakte Repräsentation der Layerliste im Speicher. Um die Liste grafisch als Panel auf der linken Seite der StudMap14 Karte anzeigen zu können, muss zuletzt ein `Ext.tree.TreePanel` erstellt werden, das in das Layer-Panel unter der Liste der aktiven Layer eingefügt wird. Klick-Listener auf den einzelnen Layern sorgen für das Hinzufügen der Layer zu den aktiven Layern und zur Karte. Wie die Liste der persönlichen Layer im StudMap14 Layer-Panel dargestellt wird, zeigt Abbildung 6.

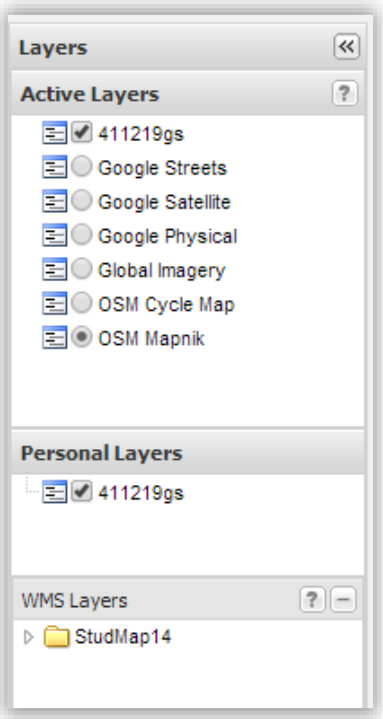

Abbildung 6 Darstellung der Liste der persönlichen Layer auf der linken Seite des Kartenfensters zwischen der Liste der aktiven Layer und der Liste der WMS-Layer.

Quellcode 16: Erstellung des Panels für die persönlichen Layer

lib/studmap/ui.js — *JavaScript*

```
var r = Ext.decode(response.responseText);
if (r.msg == "success") {
    dirs = r.dirs.split(",");
    for (var i = 0; i < dirs.length; i++) {
        var dir = r.data[dirs[i]];
        if (dir) {
            if (dir.fileext == 'kml' || dir.fileext == 'gml' ||
              dir.fileext == 'gpx') {
                var path = 'uploads/' + dir.dir + '/' + dir.filename +
                    '.' + dir.fileext;
                var layer = createStudmapVectorLayer(dir.layername,
                    path);
                layers.push(layer);
            }
            else if (dir.fileext == 'png') {
                var path = 'uploads/' + dir.dir + '/' + dir.filename +
                    '.' + dir.fileext;
                var layer = createStudmapRasterLayer(dir.layername, path,
                    dir.bbox);
                layers.push(layer);
            }
        }
    }

    var availablePersonalLayers = new GeoExt.data.LayerStore({
        layers: layers
    });
    var personalLayerList = new GeoExt.tree.LayerContainer({
        layerStore: availablePersonalLayers
    });

    [...]

    var personalL = new Ext.tree.TreePanel({
        title: 'Personal Layers',
        root: personalLayerList,
        [...]
    });
    layerPanel.insert(1, personalL);
    layerPanel.doLayout();

    return personalL;
}
```

3.2.2.5 Layer mit anderen Nutzern teilen

Zum Teilen eines persönlichen Layers mit einem anderen Benutzer kann das Share-Element im Kontextmenü des Layers verwendet werden. Dies lässt ein Fenster mit einem Formular zur Eingabe eines Benutzernamens erscheinen. Wird dieses Formular abgesendet, startet das JavaScript einen Ajax-Request, welcher die Informationen des Layers in der MySQL-Datenbank kopiert und mit der Benutzer-ID des Empfängers versieht. Eine Fehlermeldung wird ausgegeben, falls es keinen Benutzer mit dem eingegebenen Benutzernamen gibt.

Quellcode 17: Layer teilen

share.php PHP

```php
$username = ( isset($_POST['username']) ) ?
       mysql_real_escape_string($_POST['username']) : '';
$layer = ( isset($_POST['layer']) ) ?
       mysql_real_escape_string($_POST['layer']) : '';
$layername = ( isset($_POST['layername']) ) ?
       mysql_real_escape_string($_POST['layername']) : '';
$shareUsername = ( isset($_POST['shareUser']) ) ?
       mysql_real_escape_string($_POST['shareUser']) : '';

include('mysqlconnect.php');
$usernameQuery = mysql_fetch_object(mysql_query("SELECT `id` FROM
       `studmap`.`user` WHERE `username`='".$username."'"));
$shareUsernameQuery = mysql_fetch_object(mysql_query("SELECT `id` FROM
       `studmap`.`user` WHERE `username`='".$shareUsername."'"));

if ( isset($usernameQuery->id) ) {
    $ext = strtolower(pathinfo($layer, PATHINFO_EXTENSION));
    $path_rel = pathinfo($layer, PATHINFO_DIRNAME);
    $dir = explode("/", $path_rel);
    $random = $dir[1];
    $filename = pathinfo($layer, PATHINFO_FILENAME).'.'.$ext;


    if ( $ext == 'gml' || $ext == 'kml' || $ext == 'gpx' ) {
        mysql_query("INSERT INTO `vectordata` (`user`, `dir`, `filename`,
             `layername`, `sharedby`) VALUES ('".$usernameQuery->id."',
             '".$random."', '".$filename."', '".$layername."',
             '".$shareUsernameQuery->id."')");
    }
    else if ( $ext == 'png' ) {
        $bboxQuery = mysql_fetch_object(mysql_query("SELECT `bbox` FROM
             `studmap`.`rasterdata` WHERE `user`=".$shareUsernameQuery->id."
             AND `filename`='".$filename."'"));
        $bbox = $bboxQuery->bbox;
        mysql_query("INSERT INTO `rasterdata` (`user`, `dir`, `filename`,
             `layername`, `bbox`, `sharedby`) VALUES ('".$usernameQuery->id
             ."', '".$random."', '".$filename."', '".$layername."',
             '".$bbox."', '".$shareUsernameQuery->id."')");
    }
    echo '{"success":true, "msg":"success"}';
}
else {
    echo '{"success":true, "msg":"failure"}';
}
```

3.2.3 Einbindung der Google-Baselayer

Google-Geobasisdaten sind als eigener Layer-Typ in OpenLayers verfügbar. Aus diesem Grund ist das Hinzufügen der Google-Baselayer „Physical“, „Satellite“ und „Streets“ ohne großen Aufwand an der Stelle möglich, an der auch die bereits verfügbaren OSM-Baselayer erstellt und der Karte hinzugefügt werden (s. Quellcode 18).

Quellcode 18: Einbindung der Google-Baselayer

lib/studmap/map.js — *JavaScript*

```
var layerGooglePhysical = new OpenLayers.Layer.Google(
    "Google Physical",
    { type: google.maps.MapTypeId.TERRAIN, numZoomLevels: 22 }
);
Ext.apply(layerGooglePhysical, stateFuncs);
m.addLayer(layerGooglePhysical);

var layerGoogleSattelite = new OpenLayers.Layer.Google(
    "Google Satellite",
    { type: google.maps.MapTypeId.SATELLITE, numZoomLevels: 22 }
);
Ext.apply(layerGoogleSattelite, stateFuncs);
m.addLayer(layerGoogleSattelite);

var layerGoogleStreets = new OpenLayers.Layer.Google(
    "Google Streets",
    { numZoomLevels: 22 }
);
Ext.apply(layerGoogleStreets, stateFuncs);
m.addLayer(layerGoogleStreets);
```

3.2.4 Vektorfeature Funktionen

StudMap14 bietet bisher die Funktion, Punkte, Pfade und Polygone auf der Karte zu zeichnen, zu verändern (umformen, rotieren, skalieren) und umzufärben (s. Abschnitt 1.1.2). Diese Funktionalität wird hier um die in den folgenden Abschnitten beschriebenen Features erweitert.

3.2.4.1 Hinzufügung und Bearbeitung von Labels

Bisher war es nicht möglich, gezeichnete Features auf der Karte zu beschriften. Zur Beschriftung von Vektorfeatures und zur Positionierung der Beschriftung auf der Karte bietet OpenLayers bereits von Haus aus Attribute an. Es müssen also lediglich Möglichkeiten für den Benutzer geschafften werden, um diese Attribute zu setzen. Hierzu wird ein Button mit dem Titel „Edit Label“ in die „Modify Feature“ Buttongroup eingefügt, der ein Formularfenster (s. Abbildung 7) öffnet, in dem der Nutzer Beschriftung, horizontale und vertikale Ausrichtung und den Abstand der Beschriftung vom Feature definieren kann. Diese Einstellungen werden auf die Attribute des aktuell mit dem Select-Werkzeug ausgewählten Features angewandt (s. Quellcode 19).

Quellcode 19: Verarbeitung der Formulareingaben zur Feature-Beschriftung

lib/studmap/ui.js — *JavaScript*

```
var feature = modifyControl.feature;
[...]
feature.attributes.label = Ext.getCmp('label').getValue();
feature.attributes.labelAlignHorizontal =
Ext.getCmp('labelHorizontalAlignment').getValue().inputValue;
```

```
feature.attributes.labelAlignVertical =
Ext.getCmp('labelVerticalAlignment').getValue().inputValue;
feature.attributes.labelXOffset = Ext.getCmp('labelXOffset').getValue();
feature.attributes.labelYOffset = Ext.getCmp('labelYOffset').getValue();
```

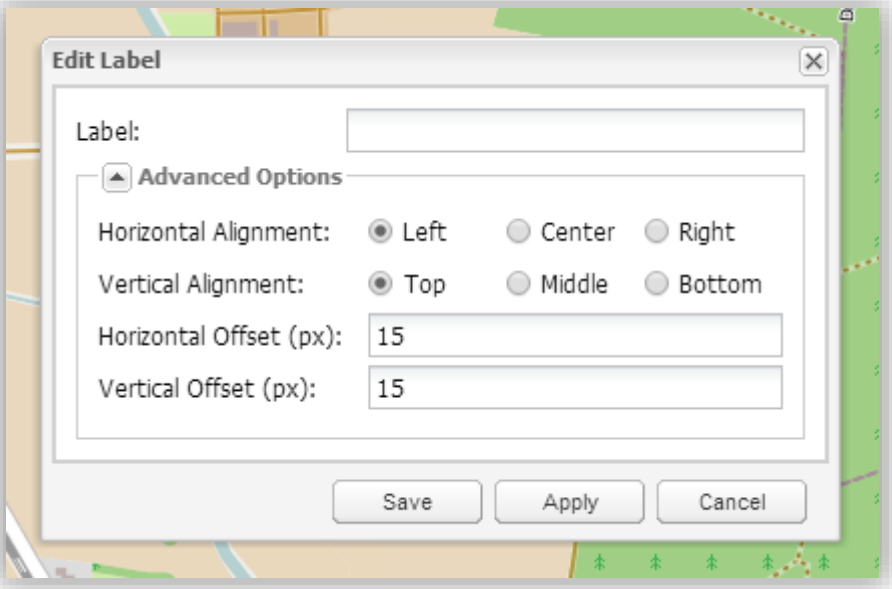

Abbildung 7 *Formularfenster zum Hinzufügen oder Bearbeiten einer Feature-Beschriftung mit Optionen zur Positionierung und Ausrichtung der Beschriftung.*

3.2.4.2 Löschung erstellter Features

Auch eine Funktion zur Entfernung hinzugefügter Features existiert bisher nicht. Um diese Funktion umzusetzen, wird ein Button mit der Beschriftung „Delete“ zur „Modify Feature“ Buttongroup hinzugefügt. Wählt der Nutzer nun ein Feature mit dem Select-Werkzeug aus, kann der Button zum Löschen des entsprechenden Features verwendet werden (s. Quellcode 20).

Quellcode 20: Löschung von Features

lib/studmap/ui.js *JavaScript*

```
drawLayer.removeFeatures(modifyControl.feature);
modifyControl.deactivate();
```

3.2.4.3 Änderung der Punktgröße

Zusätzlich wird eine Funktion zur Änderung der Größe von Point-Features implementiert. Um dieses Attribut einzustellen, wird dem Benutzer, wie auch bei der Einstellung der Transparenz eines Features, ein Schieberegler angeboten, mit dessen Hilfe er eine von sieben Größeneinstellungen auswählen kann. Eine Listener-Funktion reagiert auf Bewegungen des Schiebereglers und setzt das Attribut `studmap_pointsize`

auf den eingestellten Wert (s. Quellcode 21). Dieser Wert wird vom `OpenLayers.Style` Objekt mit Hilfe eines dynamischen Attributs und der `context`-Funktion `getPointSize` ausgelesen und dem Style-Attribut `pointRadius` zugewiesen.

Quellcode 21: Änderung der Punktgröße

lib/studmap/ui.js — *JavaScript*

```
var setAttributeFunc = function(attr, value) {
       for ( var i = 0; i < drawLayer.selectedFeatures.length; i++) {
              var feature = drawLayer.selectedFeatures[i];
              feature.attributes[attr] = value;
       }
       drawLayer.redraw();
};

var sizeSlider = new Ext.Slider({
       width: 150,
       increment: 1,
       minValue: 1,
       maxValue: 7,
       value: 3,
       listeners: {
           change: function (slider, val) {
               currentPointSize = val * 2;
               setAttributeFunc('studmap_pointsize', currentPointSize);
           }
       }
});
```

3.2.5 Optimierung der Benutzerfreundlichkeit

Neben den bereits beschriebenen Kern-Features sind im Rahmen dieser Arbeit einige kleinere Funktionen zur Optimierung der Benutzerfreundlichkeit vom StudMap14 implementiert worden. Diese Funktionen finden hier Erwähnung.

3.2.5.1 Layer löschen

Da hinzugefügte Layer in einem Cookie gespeichert werden und so für kurze Zeit auch sitzungsübergreifend verfügbar sind, kann es problematisch sein, dass Layer bisher nicht durch einen Nutzer manuell aus der Layerliste gelöscht werden können. Ein weiterer Nutzer am selben System kann so Zugriff auf die Daten des vorherigen Nutzers erhalten. Auch aus Gründen der Übersichtlichkeit kann es nützlich sein, wenn Layer aus der Liste entfernt werden können.

Zum Löschen eines Layers wird deshalb ein weiterer Unterpunkt in das Rechtsklick-Kontextmenü eines jeden Layers eingefügt, dessen Listener-Funktion den Layer aus der Liste und von der Karte entfernt (s. Quellcode 22).

Quellcode 22: Layer löschen

lib/studmap/ui.js — *JavaScript*

```
menuC.add(new Ext.menu.Item({
	text: 'Delete',
	handler: this.delete,
	tag: node
}));

[...]

delete: function (item, e) {
	var layer = item.tag.layer;
	map.removeLayer(layer);
}
```

3.2.5.2 Download der DGK5-Kacheln

Der Download der DGK5-Kartenkacheln im GeoTiff-Format ist bisher über das Info-Werkzeug in der Feature Buttongroup möglich. Um dies für den Benutzer intuitiver zu gestalten, wird der Button in „Download“ umbenannt und mit einem intuitiveren Symbol versehen. Die bisherige Funktionalität des Info-Buttons findet sich nun in den Eigenschaften eines Layers wieder, welche per Rechtsklick erreichbar sind.

3.2.5.3 Eingabe eines Kartentitels beim PDF-Export

Beim PDF-Export des aktuellen Kartenausschnittes wurde bisher der Titel „Studmap14 Export“ als Überschrift der Karte angezeigt. Dieser Kartentitel lässt sich nun beim Export vom Nutzer ändern.

Hierzu wird dem Exportfenster, welches nach der Betätigung eines der Export-Buttons erscheint, ein Textfeld zur Angabe des Kartentitels hinzugefügt. Dieses wird beim PDF-Export ausgelesen, um die Kartenüberschrift an den `printProvider` weiterzuleiten (s. Quellcode 23).

Quellcode 23: Kartentiteleingabe beim PDF-Export

lib/studmap/printing.js — *JavaScript*

```
mapTitleTextField: new Ext.form.TextField({
	fieldLabel: 'Map Title',
	value: 'StudMap14 Export',
	autoScroll: true,
	anchor: '100%'
})

[...]

provider.customParams.mapTitle =
	printingWindow.mapTitleTextField.getValue();
```

4 Empirische Auswertung und Evaluation

Um Nutzerfreundlichkeit und Funktionsfähigkeit der umgesetzten Funktionen in Hinsicht auf studentische Projektarbeiten auszuwerten, wurde eine Benutzerstudie durchgeführt.

4.1 Methode und Durchführung

Zur Evaluation wurde ein Bewertungsbogen für die fünf wichtigsten umgesetzten Funktionen (s. Tabelle 3) erstellt. Dieser wurde von potentiellen Benutzern ausgefüllt, welche gebeten wurden, die entsprechenden Funktionen zu testen und jeweils auf einer Skala von 1 bis 10 hinsichtlich Benutzerfreundlichkeit und Funktionsfähigkeit zu bewerten. Zusätzlich hatte jeder Teilnehmer die Möglichkeit, in einem Freitextfeld optional Anmerkungen zur jeweiligen Funktion zu vermerken. Außerdem beinhaltete der Bewertungsbogen am Anfang Felder für das Geschlecht, den Studiengang und das Fachsemester des Teilnehmers und am Schluss ein Freitextfeld mit der Fragestellung, ob die umgesetzten Funktionen im Rahmen studentischer Projektarbeiten oder innerhalb von Kursen von Nutzen sein können.

Tabelle 3: Für den Bewertungsbogen zu testende Funktionen

Funktion	**Hinweis für Teilnehmer**
Download der DGK5 Kartenkacheln	Der Download-Button wird aktiviert, sobald ein WMS-Layer aus dem ZDM DGK5 Ordner aktiviert wird.
Rasterdaten-Upload	Falls keine eigenen Rasterdaten verfügbar sind, können die diesem Fragebogen beiliegenden GeoTiff Dateien verwendet werden.
Vektordaten-Upload (ESRI Shape- und GML-Dateien)	Falls keine eigenen Vektordaten verfügbar sind, können die diesem Fragebogen beiliegenden Dateien verwendet werden.
Benutzer-Funktionen (Registrierung, Login, Logout)	-
Persönliche Layer und Layer teilen	Persönliche Layer sind im angemeldeten Zustand in der linken Navigationsleiste verfügbar. Es handelt sich hierbei um persistent verfügbare Layer, welche nicht nur für die Dauer einer Sitzung verfügbar sind. „Layer teilen“ ist per Rechtsklick auf einen persönlichen Layer erreichbar.

4.2 Auswertung der Ergebnisse

Der Bewertungsbogen wurde von sieben Benutzern ausgefüllt. Alle Teilnehmer haben als Studiengang „Geoinformatik“ und als Geschlecht „männlich“ angegeben. Die Mittelwerte $\overline{x}$ der Benutzerbewertungen auf einer Skala von 1 bis 10 (sehr niedrig bis sehr hoch) und die Standardabweichungen s zeigt Tabelle 4.

Tabelle 4: Ergebnisse der Funktionstests der Benutzerstudie

Funktion	**Benutzerfreundlichkeit**	**Funktionsfähigkeit**
Download der DGK5 Kartenkacheln	$\overline{x} \approx 8{,}29 \quad s \approx 0{,}88$	$\overline{x} = 9 \quad s \approx 1{,}07$
Rasterdaten-Upload	$\overline{x} = 9 \quad s \approx 1{,}07$	$\overline{x} \approx 8{,}71 \quad s \approx 1{,}58$
Vektordaten-Upload (ESRI Shape- und GML-Dateien)	$\overline{x} \approx 8{,}6 \quad s \approx 0{,}90$	$\overline{x} = 10 \quad s = 0$
Benutzer-Funktionen (Registrierung, Login, Logout)	$\overline{x} = 10 \quad s = 0$	$\overline{x} = 10 \quad s = 0$
Persönliche Layer und Layer teilen	$\overline{x} \approx 8{,}86 \quad s \approx 0{,}83$	$\overline{x} = 9 \quad s \approx 1{,}07$

Diese Ergebnisse zeigen seitens der Teilnehmer eine überwiegend positive Bewertung der Benutzerfreundlichkeit und Funktionsfähigkeit. Die Durchschnittsbewertungen der Funktionen liegen alle höher als 8 von 10 Punkten mit einer geringen Standardabweichung nahe 1. Somit sind die umgesetzten Funktionen grundsätzlich von zufriedenstellender Qualität. Herausstechend sind die Ergebnisse zu den Benutzer-Funktionen. Alle Teilnehmer bewerteten sowohl Benutzerfreundlichkeit als auch Funktionsfähigkeit mit 10 von 10 Punkten.

Trotz der allgemein gut ausgefallenen Bewertung gab es auch wenige Ausreißer nach unten (bis zu 5 von 10 Punkten), die vom Teilnehmer im jeweiligen Freitextfeld erklärt wurden. So konnte bei einem Nutzer eine hochgeladene GeoTiff-Datei manchmal nicht angezeigt werden. Ein anderer Nutzer hatte ein ähnliches Problem mit Vektordaten. Für ihn wurde ein Vektor-Layer erst beim Rauszoomen aus der Standard-Zoomstufe sichtbar und auswählbar.

Zudem gab es einige Vorschläge von Nutzern, um das Portal noch weiter zu verbessern. Die ersten drei dieser Vorschläge konnten nach der Auswertung der Nutzerstudie bereits implementiert werden.

- Passwortwiederholungs-Textfeld bei der Registrierung
- Funktion, um einen aktiven Layer zu den persönlichen Layern hinzuzufügen
- „Zoom to Layer“-Funktion für Rasterdaten (von zwei Nutzern angemerkt)
- Anzeige des tatsächlichen Upload-Fortschrittes mit dem Ladebalken beim Geodaten-Upload
- Möglichkeit zur Bearbeitung von geteilten Layern

Das Freitextfeld mit der Fragestellung, ob die umgesetzten Funktionen im Rahmen studentischer Projektarbeiten oder innerhalb von Kursen von Nutzen sein können, füllten alle Teilnehmer der Nutzerstudie mit positiven Anmerkungen aus. Die Funktionen, Layer mit anderen Nutzern teilen und persönliche Layer persistent auf dem Server ablegen zu können, erhöhen laut den Teilnehmern stark den Nutzen für die Zusammenarbeit in Projekten. Auch die intuitive Bedienung und die Plattformunabhängigkeit von StudMap14 als Webapplikation sind positiv aufgefallen. Als negativ wurden nur fehlende Funktionen zum Geoprocessing aufgeführt, die StudMap14 auch als Plattform zur Verarbeitung von Geodaten interessant machen könnten.

5 Diskussion und Ausblick

Zusammenfassend konnten zum einen viele Möglichkeiten zur Verbesserung von StudMap14 hinsichtlich Interoperabilität und Zusammenarbeit zwischen Studierenden des Fachbereichs im Rahmen von Projektarbeiten gefunden und einige davon umgesetzt werden. Dies trägt dazu bei, dass StudMap14 als Geoportal für Studierende des Fachbereichs 14 seine Aufgabe noch besser erfüllt und für mehr verschiedene Nutzungsszenarien einsetzbar ist. Zum anderen wurde in technischer Hinsicht eine Möglichkeit gefunden, GeoTiff-Rasterdaten als OpenLayers Image-Layer einzubinden.

Die Anforderungen, welche aus den drei Fallbeispielen aus dem Einleitungskapitel (s. Abschnitt 1.2) resultieren, konnten größtenteils im Rahmen dieser Arbeit umgesetzt werden. Es gibt nun die Möglichkeit, ein StudMap14-Benutzerkonto zu erstellen und Daten persistent zu speichern. Dies erspart einerseits den erneuten Upload gleicher Daten zu einem späteren Zeitpunkt und kann andererseits dazu beitragen, die Interoperabilität im Hinsicht auf Zusammenarbeit zwischen Nutzern zu erhöhen, wenn diese ein StudMap14-Benutzerkonto teilen.

Das Feature, zum Beispiel UAV-Luftbildaufnahmen in Form von GeoTiff-Dateien mit bestehenden Daten aus dem WMS-Pool von StudMap14 vergleichen zu können, wurde realisiert, indem die Funktion zum GeoTiff-Upload implementiert wurde, hat jedoch noch Verbesserungspotential und funktioniert nicht absolut problemlos (s. Abschnitt 3.2.1.1 und Abschnitt 4.2).

Indem die Anzahl der unterstützten Vektordaten-Formate sowohl beim Upload als auch beim Download von Geodaten auf beziehungsweise von StudMap14 erhöht wurde, kann StudMap nun als die im dritten Fallbeispiel beschriebene Plattform zum Austausch und zur Konvertierung von Geodaten dienen, was wiederum der Verbesserung der Interoperabilität und der Zusammenarbeit in studentischen Projektarbeiten dient.

Für zukünftige Arbeiten zur Verbesserung von StudMap14 können die im Rahmen dieser Arbeit nicht umgesetzten Funktionen in Betracht gezogen werden. Diese beinhalten Funktionen zur Unterstützung von Erdas Imagine IMG-Dateien und GeoTiff-Dateien ohne Header-Geoinformationen mit zugehöriger TFW-Datei als weitere Rasterdaten-Formate, weiterhin Administrator-Funktionen, um Layer zu sichten und zur Verfügung zu stellen, den Kartenausschnitt-Export als KML- oder GeoTiff-Datei und ein Feature zur Einbindung von WMS-Layern als persönliche Layer.

Auch Funktionen zum Geoprocessing sind denkbar, mit denen Nutzer StudMap14 nicht nur als Plattform zur Visualisierung und zum Austausch von Geodaten verwenden könnten, sondern auch in der Lage wären, eigene Geodaten aus hochgeladenen und mit den Vektorfeature-Funktionen erstellten Daten zu generieren.

Die allgemeine Benutzerfreundlichkeit der grafischen Bedienoberfläche könnte ebenfalls in Zukunft zum Beispiel mit mehr Tastenkürzeln für Funktionen, mit noch intuitiverer Bedienung bei der Erstellung und Bearbeitung von Vektor-Features oder mehr visuellem Feedback bei Benutzereingaben weiter optimiert werden.

Erkenntnisse aus dieser Arbeit können zukünftig auch beispielsweise für die Weiterentwicklung des von Ragnar Warrlich entwickelten WADI-Systems [**29**] oder vergleichbarer Geoportale verwendet werden, die auf Ext JS und GeoExt basieren.

Anhang

Inhalte der beiliegenden CD

- *bachelorarbeit_andreas_ohrem.pdf*

  PDF-Version dieser Arbeit.

- *studmap_quellcode.zip*

  Bearbeitete und neu erstellte Dateien für die Erweiterungen befinden sich nur im Hauptverzeichnis und im Verzeichnis *lib/studmap*.

Literaturverzeichnis

[1] Sencha. (2014) Sencha Ext JS. JavaScript Framework for Rich Desktop Applications. [Online, Zugriff am: 10. Februar 2014]. http://www.sencha.com/products/extjs.

[2] GeoExt Community. (2010) GeoExt. JavaScript Toolkit for Rich Web Mapping Applications. [Online, Zugriff am: 10. Februar 2014]. http://geoext.org.

[3] Kaspar, Georg. (2008) Geoportal des Fachbereichs Geowissenschaften. StudMap14. [Online, Zugriff am: 8. Februar 2014]. http://studmap14.uni-muenster.de/joomla/index.php/studmap14.

[4] GeoServer. (2014) GeoServer. Startseite. [Online, Zugriff am: 12. Februar 2014]. http://geoserver.org/display/GEOS/Welcome.

[5] Open GIS Consortium. (2004) OGC Web Map Service Interface. [Online, Zugriff am: 10. März 2014]. http://portal.opengeospatial.org/files/?artifact_id=4756.

[6] GDAL. (2014) GDAL. Geospatial Data Abstraction Library. [Online, Zugriff am: 12. Februar 2014]. http://www.gdal.org.

[7] Dekanat Fachbereich 14 Geowissenschaften. (2010) Fachbereich 14 Geowissenschaften. Organisation. [Online, Zugriff am: 9. Februar 2014]. https://www.uni-muenster.de/Geowissenschaften/organisation/index.html.

[8] Open Street Map Community. (2014) Open Street Map Wiki. Nominatim. [Online, Zugriff am: 8. März 2014]. http://wiki.openstreetmap.org/wiki/Nominatim.

[9] Open Geospatial Consortium. (2011) OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. [Online, Zugriff am: 10. März 2014]. http://portal.opengeospatial.org/files/?artifact_id=25355.

[10] Mahammad, Sazid / Ramakrishnan, R. (2003) GeoTIFF - A standard image file format for GIS applications. [Online, Zugriff am: 10. März 2014]. http://geospatialworld.net/images/pdf/117.pdf.

[11] ERDAS. (2010) ERDAS Field Guide. [Online, Zugriff am: 10. März 2014]. http://geospatial.intergraph.com/Libraries/Tech_Docs/ERDAS_Field_Guide.sflb.ashx.

[12] ESRI. (1998) ESRI Shapefile Technical Description. [Online, Zugriff am: 15. Februar 2014]. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

[13] Open Geospatial Consortium. (2007) OpenGIS® Geography Markup Language (GML) Encoding Standard. [Online, Zugriff am: 10. März 2014]. http://portal.opengeospatial.org/files/?artifact_id=20509.

[14] World Wide Web Consortium. (2014) HTML5. [Online, Zugriff am: 10. März 2014]. http://www.w3.org/TR/html5.

[15] World Wide Web Consortium. (2014) World Wide Web Consortium. [Online, Zugriff am: 10. März 2014]. http://www.w3.org.

[16] Lubkowitz, Mark. (2005) Webseiten programmieren und gestalten, 2. Auflage. Bonn: Galileo Press GmbH.

[17] World Wide Web Consortium. (2010) Cascading Style Sheets (CSS) Snapshot 2010. [Online, Zugriff am: 10. März 2014]. http://www.w3.org/TR/CSS.

[18] OpenLayers. (2014) OpenLayers: Free Maps for the Web. [Online, Zugriff am: 11. Februar 2014]. http://openlayers.org.

[19] Wenz, Christian / Hauser, Tobias. (2006) PHP 5.1. Dynamische Webseiten professionell programmieren. München: Markt+Technik Verlag.

[20] McLaughlin, Brett. (2005) Ajax meistern. Köln: O'Reilly Verlag.

[21] Wenz, Christian. (2007) JavaScript und AJAX, 7. Auflage. Bonn: GmbH, Galieo Press.

[22] Stotz, Dieter. (1995) Computergestützte Audio- und Videotechnik. Multimediatechnik in der Anwendung. Berlin: Springer Verlag.

[23] Fulton, Wayne. (2010) TIFF - Tag Image File Format. [Online, Zugriff am: 12. Februar 2014]. http://www.scantips.com/basics9t.html.

[24] OSGeo. (2014) OSGeo. PROJ.4. [Online, Zugriff am: 12. Februar 2014]. https://trac.osgeo.org/proj/wiki/GenParms.

[25] Iliffe, Jonathan / Lott, Rogar. (2008) Datums and Map Projections. For Remote Sensing, GIS and Surveying, 2nd Edition. Dunbeath, Scotland, UK: Whittles Publishing.

[26] trac. (2009) Proj4js. [Online, Zugriff am: 26. Februar 2014]. http://trac.osgeo.org/proj4js.

[27] Delphi Treff. (2014) Geographische in Gauß-Krüger-Koordinaten umrechnen. [Online, Zugriff am: 26. Februar 2014]. http://www.delphi-treff.de/tipps/mathematik/einheiten/geographische-in-gauss-krueger-koordinaten-umrechnen.

[28] Florian Wetzel. (2014) Umrechnung der Gauß-Krüger-Notation in Längen- und Breitengrade. [Online, Zugriff am: 26. Februar 2014]. http://calc.gknavigation.de.

[29] Warrlich, Ragnar. (2012) Entwicklung eines prototypischen Webservices zur Visualisierung archäologischer Befunde am Beispiel des Projektes Wadi Abu Dom Itinerary (Sudan). B.Sc.-Arbeit. Institut für Geoinformatik. Münster.

[30] StackOverflow Benutzer. (2013) StackOverflow. [Online, Zugriff am: 05. März 2014]. http://stackoverflow.com/questions/46155/validate-email-address-in-javascript.

Eigenständigkeitserklärung

Hiermit versichere ich, dass die vorliegende Arbeit über „**Optimierungsmöglichkeiten eines webbasierten Geodatenmanagements im Kontext studentischer Projektarbeiten am Beispiel des Portals StudMap14**“ selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

Münster, 11. März 2014 _____

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

Münster, 11. März 2014 _____