Fachhochschule Salzburg University of Applied Sciences

Entwicklung des Geoinformationssystems „Historische Bäume und Wälder“

BACHELORARBEIT 2

durchgeführt am Bachelorstudiengang
Informationstechnik & System–Management
Fachhochschule Salzburg GmbH

vorgelegt von:
Simon Back

Studiengangsleiter:	FH-Prof. DI Dr. Gerhard Jöchtl
BetreuerIn:	DI (FH) MMag. Moser Karin

Salzburg, Mai 2011

Simon Back
Schlenkenstrasse 43
83395 Freilassing

Hiermit versichere ich, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Freilassing, den 15. Mai 2011

Simon Back

Danksagung:

Mein größter Dank gebührt meinen Eltern, welche mir das Studium an der Fachhochschule Salzburg ermöglicht haben und mich durchgehend dabei unterstützt haben.

Besonderer Dank geht auch an FH-Prof Univ.- Doz. Mag. Dr. Karl Entacher für die Möglichkeit, das Bachelor-Praktikum an der Fachhochschule zu absolvieren und für die fachliche Betreuung und Unterstützung im Praktikum und bei der Bachlorarbeit.

Bedanken möchte ich mich auch bei DI (FH) MMag. Moser Karin für das Korrekturlesen der Arbeit.

Das Projekt wurde im Rahmen des Sparkling Science Projekts SPA/03-059/GeoWeb "Geoinformationstechnologien basierend auf OpenStreetMap und Google Maps-API", http://www.sparklingscience.at gefördert.

Der vorliegende Text ist auf Basis des LATEX-Templates aus T. Gockel[1], Form der wissenschaftlichen Ausarbeitung, Springer-Verlag, Heidelberg, 2008, erstellt. Das Originaltemplate wurde als Basis für die Erstellung einer Bachelorarbeit am Studiengang ITS angepasst.

[1] http://www.formbuch.de

Kurzzusammenfassung

Diese Arbeit ist Teil des Sparkling Science Projekts SPA/03-059/GeoWeb und behandelt das Thema „Historische Bäume und Wälder“ im Zusammenhang mit geographischen Informationssystemen. Die im Rahmen des Bachelorprojekts programmierte Web-Applikation bietet dem/der BenutzerIn die Möglichkeit kategorisierte Objekte wie z.B. Bäume oder Wälder inklusive detaillierter Informationen, Bilder, Anhänge und Texte zu speichern und unter Verwendung der Google Weltkarte anzeigen zu lassen und mit anderen BenutzerInnen zu teilen. Die Arbeit beginnt mit einem Theorieteil über Datenbankmodellierung, Inhaltsverwaltungssysteme und die Google Maps Programmierschnittstelle. In dem praktischen Teil wird die Umsetzung der Applikation beschrieben.

Abstract

The present thesis is part of the Sparkling Science project SPA/03-059/GeoWeb and addresses the topic „Historical Trees and Forests“ in combination with geographic information systems. The programmed web application offers the user the possibility to save categorized objects such as trees or forests including detailed information, images, attachments and texts and to display them using the Google map and to share with other users. The thesis starts with a theoretical part about database modelling, content management systems and the Google Maps application programming interface. The practical part describes the implementation of the application.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Kapitel 1

Einleitung

Der Beliebtheitsgrad von geographischen Informationssystemen (GIS) im Internet steigt von Tag zu Tag. Derartige GIS werden für sehr ernsthafte Themen, wie z.B. das Analysieren von Ländergrenzen im Krieg[1] verwendet, aber auch für einfache Aufgaben wie z.B. das Planen der nächsten Urlaubsroute oder das Speichern der letzten Fahrradtour. Interessant dabei ist nicht nur der jeweilige Weg zum Ziel, sondern im Besonderen auch Informationen rund um die einzelnen Orte, genannt „Point of Interest" (POI). Dazu gehören z.B. Restaurants, Tankstellen oder Schwimmbäder und im Fall der vorgestellten Applikation Bäume und Wälder. Damit Benutzer dieser Applikationen Informationen finden und verwenden können, müssen diese aber zuerst gesammelt und gespeichert werden. Um diesen Vorgang zu erleichtern wurde die in dieser Arbeit präsentierte Web-Applikation erstellt.

1.1 Zielsetzung der Arbeit

Das Ziel des Projekts „Historische Bäume und Wälder" ist die Entwicklung eines Geoinformationssystems in Form einer Web-Applikation, welche es auf einfache Art und Weise ermöglicht, Geoinformationen zu speichern, zu verwalten und auch wieder abzurufen. Im Detail heißt das, es soll die Möglichkeit bestehen, Bäume, Alleen und Wälder inklusive relevanter Informationen speichern zu können. Anschließend sollen dieses Informationen, dargestellt auf einer topografischen Karte, allen BenutzerInnen zur Verfügung stehen.

1.2 Strukturierung der Arbeit

Die Struktur der Bachelorarbeit stellt sich wie folgt dar: Kapitel 2 beschreibt und erklärt alle Grundlagen, die zum Verständnis der Arbeit notwendig sind. Das beinhaltet die Erklärung von MySQL Spatial Extensions, die Modellierung von Datenbanken und die Google Maps Programmierschnittstelle. Kapitel 3 beschreibt und erläutert die Umsetzung der Applikation. Kapitel 4 rundet abschließend die Bachelorarbeit mit einer Zusammenfassung und einem kurzen Ausblick in die Zukunft ab.

[1] http://www.dnews.de, (29.04.2011)

Kapitel 2

Grundlagen

In diesem Kapitel der Bachelorarbeit werden die grundlegenden Techniken, Funktionen und Hilfsmittel beschrieben und erklärt, welche für die Implementierung der Applikation notwendig waren. Dazu gehören Details zu MySQL und deren Spatial Extensions und Grundlagen zur Datenbankmodellierung.

2.1 MySQL GIS und Spatial Extensions

Das Datenbanksystem MySQL[1] unterstützt mehrere verschiedene Arten von raumbezogenen Erweiterungen. Dabei handelt es sich um eine MySQL Installation, welche Geometrie Datentypen unterstützt und außerdem auch Funktionen für die Berechnung von geometrischen Analysen bereitstellt.

Im Detail handelt es sich dabei um zwei verschiedene Kategorien: Datentypen, welche einzelne Geometriewerte speichern und Datentypen, welche Sammlungen von Geometriewerten speichern können. Unter die erste Kategorie fallen die Datentypen „Geometry“, „Point“, „Linestring“ und „Polygon“. Die zweite Kategorie beinhaltet die Datentypen „Multipoint“, „Multilinestring“, „Multipolygon“ und „Geometrycollection“. Die Deklaration eines Attributs als „Geometry“ ermöglicht die Speicherung von Daten in jedem beliebigen Format der ersten Kategorie, d.h. es ist möglich, alle Einzelwert Datentypen zu speichern.

Als Datenformat unterstützt MySQL das Well-Known Text (WKT) und das Well-Known Binary (WKB) Format. Diese Formate sind aber nur relevant für die Eingabe bzw. Speicherung der geometrischen Daten. Im Falle des Well-Known Text Formats werden die Daten in ASCII Form an die Datenbank übermittelt, während bei dem Well-Known Binary Format die Daten, wie der Name schon sagt, in Binärform übermittelt werden. Das interne Format für die Speicherung der Daten entspricht weder WKT noch WKB [1]. In der folgenden Aufzählung werden die für die Applikation relevanten Datentypen und Formate erklärt.

Der geometrische Datentyp POINT
Der MySQL Datentyp „Point“ speichert genau einen geometrischen Punkt, also ein Objekt. In der Applikation entspricht ein Objekt z.B. einem Nadelbaum, welcher exakte Koordinaten in Form von Längen- und Breitengrad besitzt und nur einmal exis-

[1] http://www.mysql.de, (05.02.2011)

tiert. Längen- und Breitengrad werden in der Datenbank als x- und y-Koordinate in dem geodätischen Referenzsystem World Geodetic System 1984 (WGS 84)[2] gespeichert.

Der geometrische Datentyp LINESTRING
Um mehrere zusammenhängende einzelne Objekte als ein Objekt zu speichern, wird der Datentyp „Linestring“ verwendet. Die Kette der Objekte ist aber nicht in sich geschlossen, das bedeutet, ein „Linestring“ entspricht einer einfachen Linie, definiert durch mindestens zwei Punkte.

Der geometrische Datentyp POLYGON
Im Gegensatz zu dem vorhergehenden Datentyp „Linestring“ ist ein „Polygon“ ein in sich geschlossenes Objekt, welches eine Fläche darstellt. Bezogen auf die Applikation ist dies beispielsweise ein Wald, definiert durch die Fläche von beliebig vielen Bäumen. Jeder Baum stellt einen Eckpunkt des Polygons dar.

Das Well-Known Text Format
Die Applikation verwendet für die geometrischen Daten ausschließlich das Well-Known Text Format, damit die Koordinaten direkt in ASCII Form verwendet werden können. Um die Daten in die Datenbank eintragen zu können ist zusätzlich die MySQL-Funktion „GeomFromText()“ notwendig, welche als Argument einen Geometrietyp im WKT Format entgegen nimmt. Mit dem folgenden Programmcode in Listing 2.1 können die in der Datenbank gespeicherten Koordinaten inklusive dem Namen des Objekts ausgelesen werden. Da die Koordinaten im Binärformat gespeichert werden, müssen die Koordinaten, also Längengrad und Breitengrad, mit Hilfe der MySQL Funktionen „x“ und „y“ extrahiert werden. In Abbildung 2.1 ist das Resultat der SQL Abfrage dargestellt.

```
1 <?php
2   $query =  "SELECT a.name, x(b.gpoint) as 'Längengrad', y(b.gpoint) as
        'Breitengrad' "
3       . "FROM trees_geo_data a, trees_geo_points b "
4       . "WHERE a.id=b.dataID AND b.dataID=125";
5 ?>
```

Listing 2.1: Koordinaten aus Datenbank lesen

name	Längengrad	Breitengrad
Lindenbaum in Oberalm	47.70170058332163	13.098809123012643

Abbildung 2.1: Resultat der Abfrage der Koordinaten

2.2 Datenbankmodellierung

Dies ist eine der wichtigsten Aufgaben bei einer Web-Applikation und sollte deshalb sehr sorgfältig ausgearbeitet werden und auch sehr hoch priorisiert sein. Durch die Model-

[2] http://de.wikipedia.org/wiki/World_Geodetic_System_1984, (29.04.2011)

lierung erhält man ein fertiges Grundkonzept, an das man sich später halten sollte, um unnötig viel Arbeit mit späteren grundlegenden Änderungen aus dem Weg zu gehen.

2.2.1 Die MySQL Engine InnoDB

Die Verwendung der MySQL Storage Engine InnoDB ist darin begründet, dass InnoDB im Gegensatz zu der sonst üblichen Engine MyISAM Transaktionskonzepte unterstützt. Das ist vor allem wichtig, wenn der Fall eintreten kann, dass mehrere Benutzer gleichzeitig einen Datenbankeintrag ändern wollen. Durch die transaktionssichere Engine InnoDB wird dies verhindert, was somit die Mehrbenutzerfähigkeit und die Performance erhöht. [2]

2.2.2 MySQL Workbench

Die von der Oracle Corporation entwickelte Software MySQL Workbench[3] bietet spezielle Funktionen und Werkzeuge zur Modellierung von MySQL-Datenbanken. Der Einsatz dieser Software ist darin begründet, dass sie die Modellierung der Datenbank sehr übersichtlich darstellt und somit jeder Zeit eine einfache Validierung möglich ist. Die Workbench bietet außerdem noch Möglichkeiten zur Verwaltung von Datenbankservern und Benutzern und ist für Windows, Linux und Mac OS verfügbar [3]. In Abbildung 2.2 ist der Startbildschirm von MySQL Workbench dargestellt.

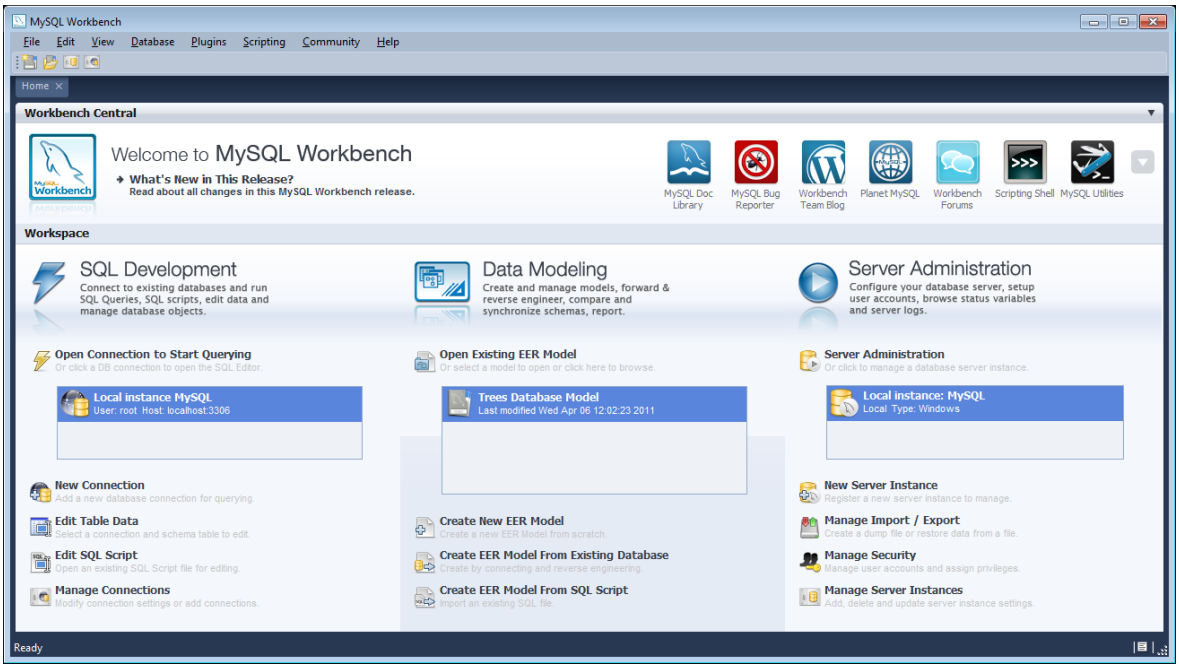

Abbildung 2.2: MySQL Workbench zur Datenbankadministration und -modellierung

[3]https://www.mysql.de/products/workbench, (03.02.2011)

2.2.3 Grundlagen der ER-Modellierung

Die Erstellung eines Entity Relationship Modells erfordert einige Grundlagen, welche im Folgenden in verkürzter Form beschrieben sind. Das Entity Relationship Modell wurde im Jahre 1970 von E.F. Codd eingeführt [2]. Dieses dient dazu, einen Realitätsausschnitt mit fest definierten Methoden und Werkzeugen zu beschreiben. Ein vollständiges ER-Modell enthält immer ein ER-Diagramm und eine Beschreibung der einzelnen Elemente. Das ER-Diagramm setzt sich zusammen aus eindeutig identifizierbaren Objekten der realen oder virtuellen Welt, den sogenannten Entitäten und aus deren Merkmalen, auch genannt Attribute. Zusätzlich kommen noch die Schlüsselattribute und die Beziehungen zwischen den Entitäten hinzu.

Ein Schlüssel ist ein Attribut, welches ein anderes oder mehrere andere Attribute eindeutig identifiziert. Des Weiteren kann ein Schlüssel entweder aus einem Merkmal bestehen oder auch aus mehreren, was dann als zusammengesetzter Schlüssel bezeichnet wird.

Die Beziehung zwischen ein oder mehreren Entitäten werden hauptsächlich durch drei verschiedene Arten definiert. Die erste Beziehungsart beschreibt eine „1:1“ Beziehung, was bedeutet, dass zu jedem Element der Entitätsmenge 1 ein Element der Entitätsmenge 2 existiert. Die zweite Beziehungsart beschreibt eine „1:n“ Beziehung, welche dadurch definiert ist, dass zu jeder Entität der Entitätsmenge 1 mehrere Entitäten der Entitätsmenge 2 gehören. Die dritte der drei wichtigsten Beziehungsarten ist die „n:m“ Beziehung, bei welcher zu n Entitäten der Entitätsmenge 1, m Entitäten der Entitätsmenge 2 gehören. In Abbildung 2.3 sind diese drei Beziehungsarten inklusive der Notationselemente mittels MySQL Workspace dargestellt [2].

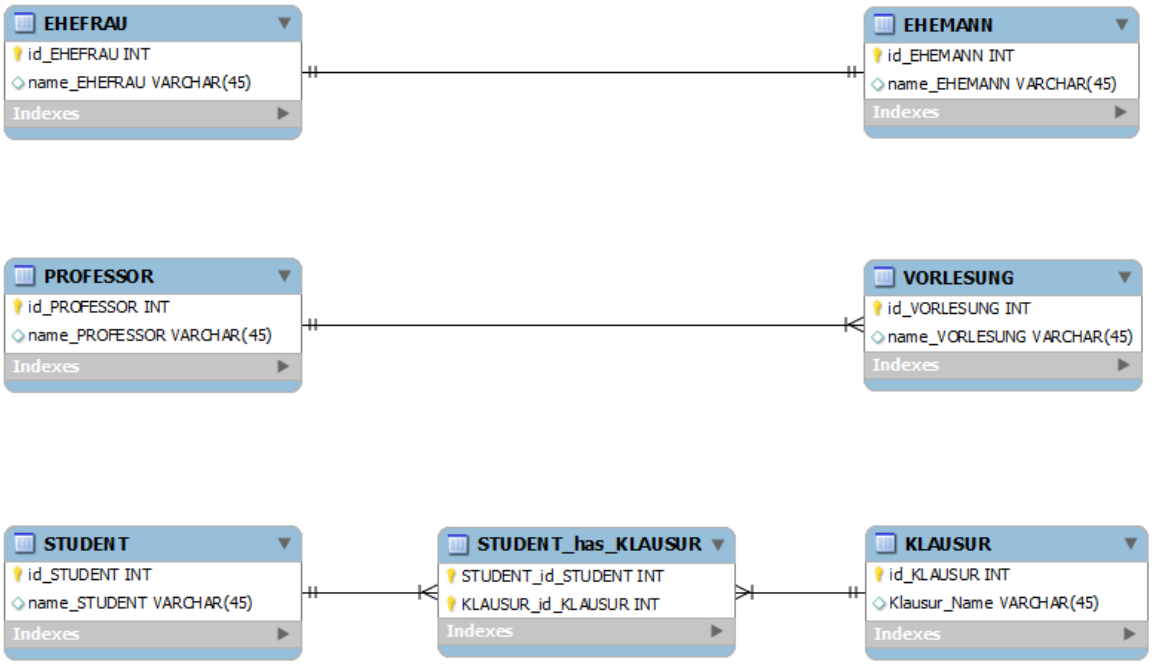

Abbildung 2.3: Die drei wichtigsten Beziehungsarten im ER-Modell

Bemerkung 1: Die „n:m“ Beziehung wird in einem ER-Modell üblicherweise mit einem eigenen Symbol dargestellt. Bei der MySQL Workbench Beziehung in Abbildung 2.3 wird diese allerdings als eigene Verknüpfungstabelle dargestellt.

Bemerkung 2: In dieser Arbeit werden alle Modellierungsschritte mit den drei präsentierten Beziehungsarten abgedeckt. In der ER-Modellierung gibt es jedoch noch weitere Beziehungsarten wie z.B. Vererbung, abhängige - sowie rekursive Beziehungen [2].

Bei der Entwicklung eines ER-Modells sollte grundsätzlich immer schrittweise vorgegangen werden. Im ersten Schritt sind die Entitätsmengen festzulegen. Im nächsten Schritt sind die Beziehungen bzw. Beziehungsarten zwischen den Entitäten zu definieren. Im letzten Schritt sind die Attribute und Schlüssel der Entitätsmengen zu definieren [2].

2.2.4 Physische Umsetzung

Bei der physischen Umsetzung des ER-Modells als MySQL Implementierung werden die Beziehungen im Modell durch Fremdschlüssel der Tabellen interpretiert. Dies stellt sich in der Workbench folgendermaßen grafisch dar:

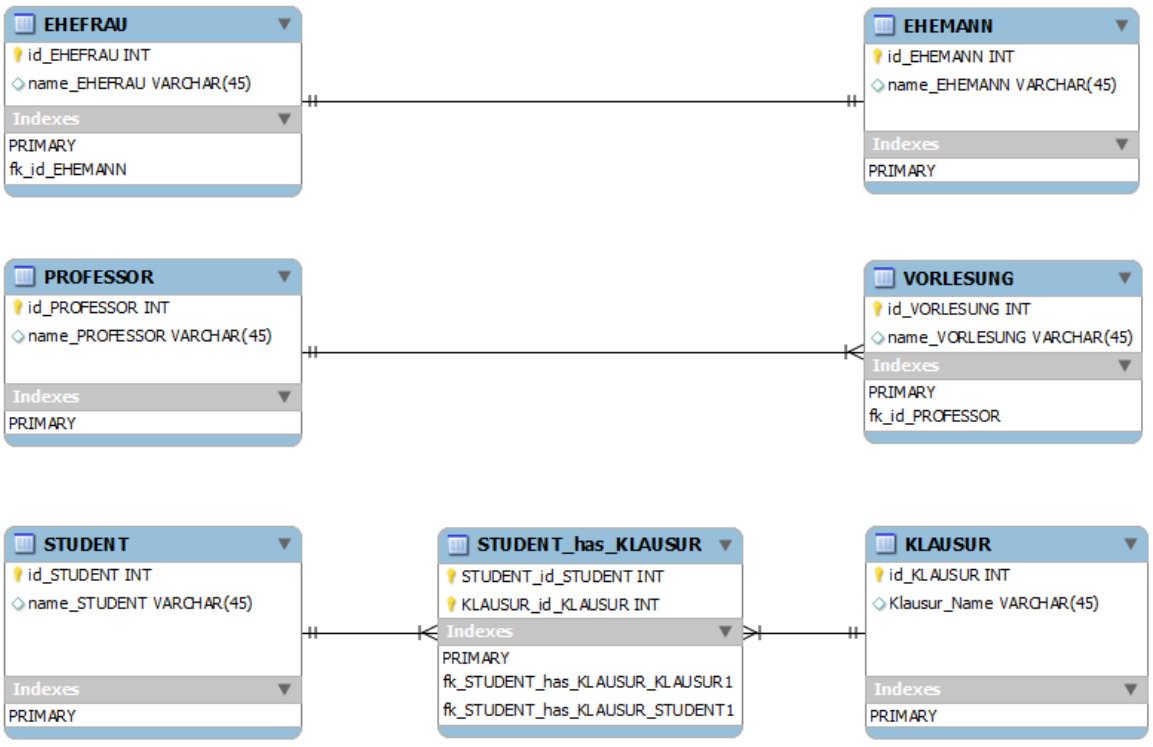

Abbildung 2.4: Die drei wichtigsten Beziehungsarten im ER-Modell mit Fremdschlüsseln

Bemerkung: Andere ER-Programme unterscheiden im Gegensatz zu dem MySQL-spezifischen Workbench strikt zwischen einem konzeptuellen und einem physischen Modell. Dies ist beispielsweise der Fall bei dem Programm PowerDesigner[4] der Firma Sybase

[4]http://www.powerdesigner.de/, (29.04.2011)

und bei dem Toad Data Modeler[5] der Firma Quest Software.

2.3 Content Management System

Ein Content Management System (CMS) ist, wie der Name schon sagt, ein System mit dem man Inhalte verwalten kann. Spricht man von einem CMS - so bezieht man sich in den meisten Fällen auf eine Web-Applikation. Dieses bietet die Möglichkeit eine Internet Applikation auf eine professionelle und effiziente Art und Weise zu aktualisieren und zu warten. Außerdem bietet es weitere Vorteile, welche im Folgenden aufgezählt sind.

- Einfache Bearbeitung von Inhalten
- Verwaltung durch mehrere Benutzer möglich
- Einfache Benutzer- und Rechteverwaltung
- Überwiegend Model View Controller Konzept (siehe Abschnitt 2.3.4)
- Zeiteinsparung durch vereinfachte Funktionen
- Verwaltung ist von überall möglich
- Keine Notwendigkeit von speziellem Fachwissen

2.3.1 Entscheidungsfindung CMS

In der Praxis bieten nicht alle Content Management Systeme alle vorher genannten Vorteile, sondern es müssen Prioritäten gesetzt werden und anschließend ein geeignetes System ausgewählt werden. Die Entscheidungsfindung für ein CMS für die Web-Applikation „Historische Bäume und Wälder“ wurde auf folgende Systeme eingeschränkt: WordPress, Drupal, Typo3 und Joomla. Das CMS Typo3 wurde bereits in der Vorauswahl als weniger geeignet deklariert, da es relativ komplex für die gewünschte Anwendung wäre und es dadurch auch nicht voll ausgenutzt werden könnte. Die drei verbleibenden Systeme wurden beurteilt durch Analyse der jeweiligen Vor- und Nachteile, welche in den folgenden Aufzählungen dargestellt sind.

Vor- und Nachteile des Content Management System Wordpress [6]
Vorteile:

- Sehr benutzerfreundlich
- Sehr gut geeignet für Blogs
- Einfache Suchmaschinenoptimierung
- Viele Plugins verfügbar
- Übersichtliche Benutzeroberfläche

[5] http://www.quest.com/toad-data-modeler/, (29.04.2011)
[6] http://wordpress-deutschland.org, (01.02.2011)

Nachteile:

- Eigentlich nur für Blogs geeignet
- Nicht Entwicklerfreundlich
- Benutzerverwaltung nur mit festgelegten Benutzertypen
- Standard Funktionsumfang sehr gering

Vor- und Nachteile des Content Management System Drupal [7]
Vorteile:

- Sehr zuverlässig und robust
- Flexibel
- Leicht zu erweitern
- Einfache Bedienung
- Suchmaschinenfreundlichkeit
- Sicherheit
- Mehrsprachigkeit

Nachteile:

- Nicht sehr Benutzerfreundlich
- Nicht alle Medien werden unterstützt
- Keine Linkverwaltung
- Gestaltung aufwändig

Vor- und Nachteile des Content Management System Joomla [8]
Vorteile:

- Designer- und entwicklerfreundlich
- Große Community vorhanden
- Sehr gute Rechteverwaltung in neuer Version
- Sehr einfache Installation
- Übersichtliche Administrationsoberfläche
- Sehr viele Plugins und Erweiterungen verfügbar
- Beliebig erweiterbar

Nachteile:

- Einarbeitungszeit notwendig

[7] http://drupal.org, (01.02.2011)
[8] http://www.joomla.de, (01.02.2011)

- Keine automatischen Updates
- Unübersichtlich bei großen Datenmengen

Um anhand der Vor- und Nachteile der Systeme das passende CMS auszuwählen, müssen zuerst die Prioritäten und Anforderungen der Web-Applikation festgelegt werden. Für den Aufbau der Applikation ist es notwendig, dass die Programmierung von Modulen und das Erstellen eines Templates möglich und bestenfalls einfach zu realisieren ist. Des Weiteren ist eine umfangreiche Rechteverwaltung notwendig, um für jede Administrationsschicht der Applikation eine Benutzergruppe erstellen zu können. Abschließend ist es außerdem notwendig, dass die Applikation multilingual betrieben werden kann. Nach dem Vergleich der Anforderungen und Prioritäten mit den einzelnen Vor- und Nachteilen wurde Joomla als favorisiertes Content Management System festgelegt. Natürlich bieten auch die Konkurrenzprodukte die Möglichkeit der Umsetzung dieser Features. Der Entscheidungsprozess wurde allerdings auch von den bisherigen Erfahrungen des Projektteams mit Joomla beeinflusst.

2.3.2 Das Content Management System Joomla

Joomla ist ein in der Programmiersprache PHP (Version 5) programmiertes Web-Content-Management-System, das als „freie Software“ unter der GPU Lizenz verfügbar ist und welches hauptsächlich zur Erstellung und Verwaltung von Internetseiten verwendet wird. Zur Speicherung der Inhalte verwendet Joomla die Datenbank MySQL. Joomla ist seit dem 10. Januar 2011 in der Version 1.6 verfügbar[9] und hat seitdem einige neue Funktionen sowie u.a. eine wesentlich verbesserte Benutzer- und Rechteverwaltung, Mehrsprachigkeit und eine frei definierbare Kategoriestruktur. Joomla ist aufgeteilt in ein „Backend“ und ein „Frontend“. Ersteres ist eine eigenständige Internetseite mit Benutzerauthentifizierung, welche die vollständige Konfiguration und Bearbeitung des Systems selbst und aller Inhalte ermöglicht. Zweiteres ist die Darstellung der eigentlichen Internetseite mit den Inhalten, welche für die Öffentlichkeit bestimmt sind [4].

2.3.3 Die Programmstruktur von Joomla

Die Struktur der Programmlogik des Content Management Systems Joomla ist gegliedert in Komponenten, Module und Plugins. Eine Komponente ist mehr oder weniger ein eigenständiges Programm innerhalb von Joomla und ist vollständig von anderen Komponenten abgegrenzt. In den meisten Fällen gibt es für Komponenten noch eine Verwaltung im „Backend“, also in der Verwaltungs- und Konfigurationszentrale von Joomla.

Module sind eigene Programmteile, welche in Komponenten eingebettet werden. Ein Modul wird normalerweise an einer vorher definierten Position in dem aktivierten Template angezeigt. Des Weiteren können für Module im „Backend“ zur Konfiguration Einstellungen vorgenommen werden wie z.B. der Titel des Moduls, die Position, die Zugriffsebene und auf welchen Seiten das Modul angezeigt werden soll.

Plugins sind im allgemeinen sehr spezifisch und nur für einen bestimmten Zweck gedacht und können wahlweise aktiviert oder deaktiviert werden. Darüber hinaus sind sie

[9] http://www.joomla.org, (30.04.2011)

bestimmten Bereichen zugeordnet und sind meistens nicht konfigurierbar. Ein Beispiel für ein Joomla eigenes Plugin ist die GMail-Authentifizierung, mit welcher es im aktivierten Zustand möglich ist, dass sich BenutzerInnen in den BenutzerInnenbereich der Applikation mit den eigenen Google-Zugangsdaten einloggen können.

2.3.4 Joomla-Module im Detail

Module in Joomla sind nach dem Model-View-Controller (MVC) Muster gegliedert [4]. Das bedeutet, dass es ein Modell gibt, das die benötigten Daten z.B. durch eine Datenbankabfrage zur Verfügung stellt. Zusätzlich existiert der Bereich View, welcher für die Präsentation zuständig ist und die Daten des Modells darstellt. Der Bereich des Controllers ist verantwortlich für die Koordination zwischen Modell und Präsentation und um deren Verwaltung. Der Vorteil dieses Musters ist, dass die Logik und die Darstellung voneinander strikt getrennt sind und somit eine gewisse Ordnung herrscht.

Allerdings ist der Mehraufwand eines solchen Konzepts nicht zu unterschätzen. Um ein Modul in Joomla zu installieren, muss im „Backend“ eine ZIP-Datei hochgeladen werden, welche einer bestimmten Ordnerstruktur entsprechen muss. Diese ist in folgender Aufzählung dargestellt.

- mod_modulname.php
- helper.php
- tmpl/default.php
- mod_modulname.xml

Die Datei mod_modulname.php repräsentiert die Steuerung des Moduls, d.h. diese Datei wird zuerst aufgerufen, nachdem das Modul angezeigt wird. Des Weiteren wird in dieser Datei ein Objekt der „Helper“-Klasse erzeugt, welche sich in der Datei helper.php befindet. Diese Helper Klasse repräsentiert das Modell des MVC Musters, d.h. es sammelt die Daten, bereitet sie für die Präsentation vor und gibt sie anschließend zurück an die Steuerungsdatei. Die Datei default.php im Ordner tmpl ist zuständig für die Präsentation der Daten, welche von dem Modell kommen. Im einfachsten Fall werden die Daten in dieser Datei nur mit einem PHP Befehl ausgegeben. Die letzte erforderliche Datei ist eine XML Datei mit dem Namen mod_modulname.xml. Diese Datei wird bei der Installation ausgelesen und enthält alle Informationen über das Modul wie z.B. den Titel, Autor, Erstellungsdatum, Beschreibung und alle zugehörigen Dateien [5].

2.4 Die Google Maps-Programmierschnittstelle (API)

Ein Application Programming Interface (API) ist eine Schnittstelle, welche die Möglichkeit des Zugriffs auf Daten oder Dienste einer anderen Software bietet. In Bezug auf die Google Maps API ist das die Einbettung der Weltkarte „Google Maps“ von Google und zahlreiche damit verbundene von Google zur Verfügung gestellte Funktionen. Diese ermöglichen es, über verschiedene Dienste Karten zu bearbeiten und Inhalte in Karten einzufügen. Für

Karten können allgemein Kartenobjekte, Kartenereignisse, Kartensteuerelemente und Kartenoverlays definiert werden, und es ist möglich Kartenservices in Anspruch zu nehmen. Unter Kartenobjekt versteht man die Karte selbst und ihre verschiedenen Typen:

- MapTypeId.ROADMAP
  Anzeige der Straßenkartenansicht
- MapTypeId.SATELLITE
  Anzeige der Google Earth Satellitenbilder
- MapTypeId.HYBRID
  Anzeige der Straßenkartenansicht inklusive Satellitenbilder
- MapTypeId.TERRAIN
  Anzeige einer physischen Karte, basierend auf Geländeinformationen

Für die Applikation „Historische Bäume und Wälder“ wurde als Basis die Straßenkartenansicht verwendet, das Implementieren von anderen Karten wäre aber jederzeit möglich.

Kartenereignisse sind Ereignisse, welche auf bestimmte Interaktionen von BenutzerInnen oder auf sogenannte MVC-Statusänderungen reagieren. Erstere werden ausgelöst, wenn ein/eine BenutzerIn z.B. mit der Maus auf ein Objekt klickt oder darüber fährt. Zweitere werden ausgelöst, wenn sich eine Objekteigenschaft ändert. Dies ist beispielsweise der Fall, wenn die Karte vergrößert oder verkleinert wird.

Mit Kartensteuerelementen ist es möglich, die Karte zu steuern. Im Detail handelt es sich überwiegend um Buttons, welche zur Navigation auf der Karte oder zur Veränderung der Ansicht der Karte genutzt werden. Des Weiteren ist es aber auch möglich, eigene Steuerlemente zu platzieren, um eigene Funktionen damit anzusteuern. Dies ist vor allem sehr hilfreich, wenn man dem/der BenutzerIn über einen Button die Möglichkeit geben möchte, die Karte in den Vollbildmodus zu schalten oder die Sichtbarkeit bestimmter Objekte zu aktivieren bzw. zu deaktivieren.

Die Kategorie Kartenoverlays beschreibt Objekte, die sozusagen auf die Karte gesetzt werden können, d.h. sie sind auf der Karte an bestimmten Positionen bzw. Koordinaten sichtbar. Als Objekte werden Marker, Polylinien und Polygone bezeichnet, und zusätzlich ist es auch möglich, eigene Overlays zu definieren. Die zur Verfügung gestellten Kartenservices bieten die Möglichkeit, auf einfache Art und Weise z.B. Entfernungen zu berechnen, Adressen oder Routen zu finden. In den folgenden Kapiteln wird eine Auswahl dieser Funktionen genauer erläutert [6].

2.4.1 Marker, Polylinien und Polygone

Die Google Maps API bietet sogenannte Overlays, welche Objekte darstellen, die mit Optionen definiert werden können und anschließend auf der Karte angezeigt werden. Eine Art von Overlay ist der „Marker“. Dieser kennzeichnet durch ein beliebiges Symbol einen bestimmten durch Koordinaten definierten Ort auf der Karte. Als optionaler Zusatz kann noch ein Titel für den Marker angegeben werden, welcher angezeigt wird, wenn man die Maus über den Marker bewegt.

Eine weitere Art von Overlays ist die „Polylinie“, welche durch mindestens zwei Punkte auf der Karte definiert wird. Diese Punkte sind verbunden durch eine Linie, welche sich

durch die Angabe der Farbe, der Deckkraft und der Liniendicke gestalten lässt. Die dritte und letzte Art der verwendeten Overlays ist das „Polygon“, welches auch durch mehrere geographische Koordinaten definiert wird. Der Unterschied zur „Polylinie“ besteht darin, dass die Punkte immer eine geschlossene Fläche bilden, d.h. der erste Punkt ist zugleich auch der letzte Punkt. Darüber hinaus ist es möglich, für die eingeschlossene Fläche eine Farbe und deren Deckkraft zu definieren.

Für jedes Overlay besteht außerdem die Möglichkeit, ein „InfoWindow“ zu definieren. Dies ist auch eine Google Maps Funktion, welche beispielsweise beim Klicken auf einen Marker ein Fenster in Form einer Sprechblase öffnet, in welchem weitere Informationen, Bilder oder Ähnliches in HTML Form angezeigt werden können. Google Maps bietet noch einige weitere Overlays, welche aber für die Applikation nicht von Bedeutung sind.

2.4.2 Clustering von Objekten

Sobald sich in einem Ausschnitt auf der Karte mehrere Marker befinden, überlagern sich diese und die Darstellung wird unübersichtlich vor allem, wenn man aus der Karte herauszoomt, um die gesamte Karte anzuzeigen. Um diesem Problem vorzubeugen, hat Luke Mahe eine Javascript Funktion programmiert, welche die Marker beim Herauszoomen aus der Karte clustert [7]. Das bedeutet, Marker, die in einer bestimmten Entfernung in einem Gebiet liegen, werden als ein Objekt dargestellt. Für die Cluster Objekte können andere Symbole definiert werden, welche idealerweise signalisieren, dass es sich um mehrere Objekte handelt, die zusammengeführt wurden.

2.4.3 Orts-, Regions-, und Ländersuche auf der Karte

Die Google Maps API bietet eine Klasse, welche Geocodierung ermöglicht. Unter Geocodierung versteht man das Ermitteln der geographischen Koordinaten einer Adresse. Im Detail bedeutet das, dass eine Adresse via Javascript an die Google Maps API geschickt wird und im Erfolgsfall die Koordinaten dieser Adresse zurückgegeben werden.

Kapitel 3

Umsetzung

In den folgenden Abschnitten wird die vorher erklärte und beschriebene Theorie in die Praxis umgesetzt. Das Ziel ist eine Web-Applikation mit einer eingebetteten Google Maps Weltkarte und der Möglichkeit, historische Bäume und Wälder, in Kategorien, eingeteilt in dieser kennzeichnen zu können. BenutzerInnen soll es möglich sein, einzelne Bäume, Alleen mit mehreren Bäumen und ganze Wälder in die Karte einzutragen und zu speichern. Als Zusatzinformation können zu jedem Objekt Informationen über dieses Objekt, Bilder, Anhänge und Texte gespeichert werden. Des Weiteren hat der/die BenutzerIn die Möglichkeit, die selbst erstellten Objekte zu verändern oder zu löschen. Sobald ein Objekt gespeichert wurde, erhalten Administratoren/Administratorinnen eine Nachricht zur Freischaltung dieses Objekts. Nachdem das Objekt geprüft und freigeschalten wurde, kann jeder/jede BesucherIn der Internetseite das Objekt und dessen Zusatzinformationen betrachten.

3.1 Joomla Installation

Für die Installation von Joomla 1.6 wird ein Webserver mit einer MySQL-Datenbank und einer PHP Installation in der Version 5 benötigt. Zuerst muss das gezippte Installationspaket von einer entprechenden Quelle (z.B. joomla.org) heruntergeladen und anschließend entpackt werden. Im Anschluss müssen mittels eines FTP-Programms diese Dateien auf den Speicherplatz des Webservers kopiert werden. Nach diesem Vorgang kann der Installationsvorgang durch das Aufrufen der Internetadresse des Webservers im Internetbrowser gestartet werden. Im ersten Schritt der Installationsroutine muss der/die BenutzerIn die gewünschte Sprache auswählen. Nachdem diese Auswahl getroffen wurde, wird der Webserver auf Kompatibilität in Bezug auf Joomla geprüft, wie in Abbildung 3.1 dargestellt ist. Im Falle einer Inkompatibilität schlägt die Installationsroutine eine Fehlerbehebungsmaßnahme vor. Tritt kein Fehler auf, so muss die im nächsten Schritt angezeigte Lizenzvereinbarung gelesen und akzeptiert werden. In den zwei darauffolgenden Schritten müssen die Einstellungen für die Datenbank und optional für die FTP Verbindung angegeben werden. Im letzten Schritt kann ein Seitentitel, die Email-Adresse des/der Administrators/Administratorin und das Passwort für den Administrationszugang festgelegt werden.

Abbildung 3.1: Joomla 1.6 Installationsroutine - Kompatibilitätsprüfung

3.2 Datenbankmodellierung mit MySQL Workbench

MySQL Workbench, detailliert beschrieben in Abschnitt 2.2.2, ist eine Software zur Modellierung und Verwaltung von MySQL Datenbanken. Die folgende Modellierung 2.2.3 bezieht sich nur auf die Tabellen, welche für die Programmierung der Module für die Applikation notwendig sind. Wie in Abbildung 3.2 zu erkennen ist, ist das Modell in mehrere Bereiche aufgeteilt. Im Folgenden wird die Modellierung detailliert beschrieben.

Die Tabelle „jos_users“ in dem grünen Bereich am unteren, linken Rand ist nur symbolisch in der Modellierung und wird nicht in die Datenbank übernommen, weil diese die Standard BenutzerInnen-Tabelle von Joomla darstellt und somit die Verbindung zwischen den neuen für die Programmierung der Applikation notwendigen Tabellen und den von Joomla benutzten Tabellen herstellt.

Die Tabelle „trees_geo_data“ in dem roten Bereich in der rechten, unteren Ecke ist das Herz der gesamten Modellierung, weil diese mit allen anderen Tabellen direkt oder indirekt in Beziehung steht. Jedes vorhandene Objekt in der Applikation hat einen Eintrag in dieser Tabelle. Die Tabellen in dem blauen, rosa und orangefarbenen Bereich speichern detaillierte Informationen über ein Objekt wie z.B. Objekttyp, Koordinaten und Kategorie. Die Tabelle „trees_icons“ speichert für jede Kategorie das zu verwendende Symbolbild mit Namen und Dateinamen. Der letzte Bereich, eingefärbt in minzgrün, beinhaltet ebenfalls Tabellen mit Zusatzinformationen zu Objekten. Die Tabelle „trees_texts“ speichert von BenutzerInnen eingegebene Texte für Objekte während die Tabelle „trees_images“ Bilder zu den jeweiligen Objekten speichert. Die Tabelle „trees_hlinks“ speichert Internetlinks zu den Objekten. Auf Grund dessen, dass ein Objekt mehrere Texte, Bilder und Internetlinks als Zusatz enthalten kann, sind die jeweiligen Tabellen als eigenständige Tabelle ausgelagert und die identifizierende ID zu dem entsprechenden Objekt wird zusätzlich als „Foreign Key“ gespeichert.

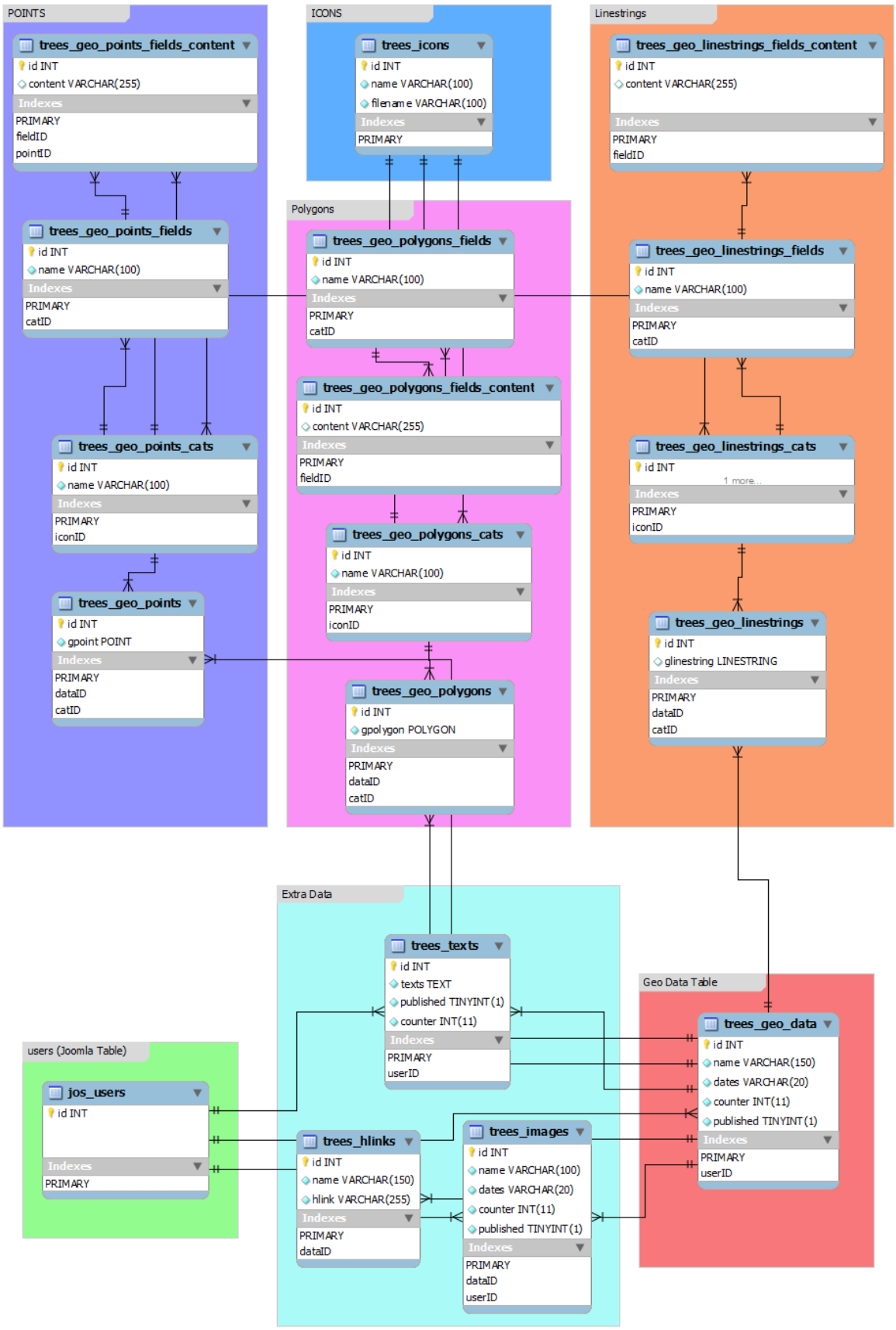

Abbildung 3.2: MySQL Workbench Datenbankmodellierung

3.3 Programmierung der Joomla-Module

Wie schon in Abschnitt 2.3.3 detailliert erläutert, besteht die Programmstruktur von Joomla aus Komponenten, Modulen und Plugins. Für die Programmierung der Applikation „Historische Bäume und Wälder“ wurden insgesamt drei Module programmiert: „Einbetten der Google Weltkarte“, „Erstellen von neuen Objekten“ und „Verwaltung von erstellen Objekten“. Die Ordnerstruktur der Module ist aufgebaut wie in Abschnitt 2.3.4 beschrieben. In den folgenden Unterkapiteln werden die Funktionen der Module unter Verwendung von Programmcodeauszügen beschrieben und erklärt. Die Module befinden sich alle im Frontend der Web-Applikation, d.h. die Internetseite ist komplett wartbar, konfigurierbar und administrierbar, ohne dass sich der/die AdministratorIn in den Joomla-eigenen Backend-Administrationsbereich einloggen muss.

3.3.1 Initialisierung der Google-Weltkarte

Die Darstellung der Google Map wird durch die Javascript-Funktion „initialize“ realisiert, welche in dem folgenden Listing 3.1 auszugsweise dargestellt ist.

```
<script type="text/javascript">
 var latlng = new google.maps.LatLng(latitude, longitude);
 var myOptions = {
   zoom: zoomlevel,
   center: latlng,
   mapTypeId: google.maps.MapTypeId.ROADMAP
 };
 map = new google.maps.Map(document.getElementById('content'), myOptions);
</script>
```

Listing 3.1: Initialisierung der Google Weltkarte

In der Funktion „initalize“ wird zuerst ein neues „google.maps.LatLng“-Objekt erstellt, welches die vorher definierten Koordinaten als Argumente empfängt. Anschließend wird ein „map-options“ Objekt mit den gewünschten Optionen für die Karte erstellt. Dieses wird danach bei der Erstellung des „google.maps.Map“-Objekts als Parameter übergeben. Der andere Übergabewert übergibt eine Referenz auf ein „DIV“ Element mit dem Namen „content“, in dem die Karte angezeigt wird.

Zusätzlich zu der Funktion „initialize“ werden in dieser Funktion zwei Buttons konfiguriert, erkennbar in Abbildung 3.3 in der rechten oberen Ecke, welche auf der Karte angezeigt werden. Der erste Button, mit dem Titel „Configuration“, ermöglicht die Kartendarstellung im Vollbild und der zweite Button, welcher den Titel „Fullscreen“ trägt, ermöglicht die Aktivierung bzw. Deaktivierung der Marker, Polylinien und Polygone auf der Karte. Der Vollbildmodus wird aktiviert, indem die Ebene, in welcher die Karte dargestellt wird, mit Hilfe von Javascript und CSS auf die gesamte Größe des Browserfensters skaliert wird. Die Sichtbarkeit von Objekten wird über ein „Flag“ gesteuert, welches in der PHP-Session des Benutzers gespeichert wird. Nach Ablauf der Session werden wieder alle Objekte angezeigt.

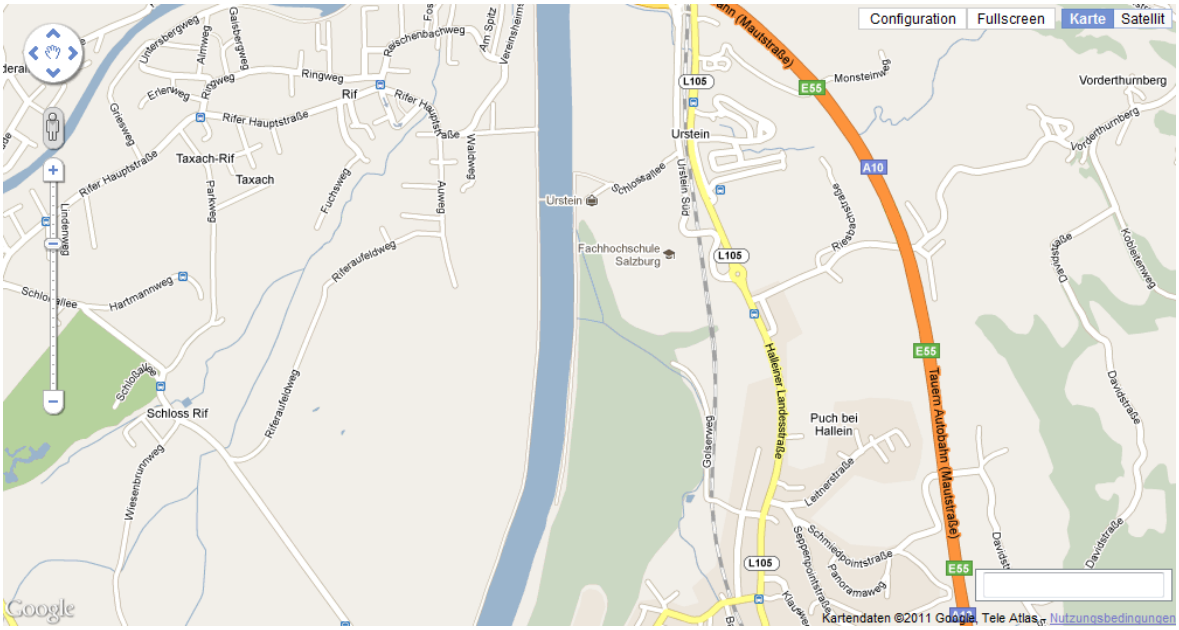

Abbildung 3.3: Google Map mit zusätzlichen Steuerelementen

3.3.2 Suche

Die Suche nach Adressen, Regionen und Ländern auf der Karte, siehe Kapitel 2.4.3 auf Seite 12, ist eine weitere Funktion, die das Modul „Einbetten der Google Weltkarte“ bietet. In Abbildung 3.3 ist in der unteren rechten Ecke das Suchfeld dargestellt, welches die Suchanfragen entgegennimmt.

```
<script type="text/javascript">
 function searchByWord(val)
 {
   if(val.length > 3)
   {
      var suche = val;
      geo.geocode({
             'address': suche }, function(results, status) {

             if (status == google.maps.GeocoderStatus.OK) {
        map.setCenter(results[0].geometry.location);
        map.setZoom(13);
             } else  {
                 alert(val+" konnte nicht gefunden werden");
             }
        });
   }
 }
</script>
```

Listing 3.2: Die Suchfunktion der Karte

Das Suchfeld nimmt Texteingaben entgegen und sucht mit Hilfe des Geocodierungsdienstes von Google, siehe Kapitel 2.4.3, nach der Eingabe. Wie in Listing 3.2 erkennbar, wird

die Suche erst durchgeführt, wenn der eingegebene Text mehr als drei Buchstaben umfasst. Wenn der Geocoder eine positive Rückmeldung mit dem Status „OK“ zurückgibt, wird die Karte mit dem Zoomfaktor 13 auf die von dem Geocoder ermittelte Position zentriert. In dem Fall, dass keine passende Position zu der Eingabe gefunden wird, wird eine Fehlermeldung zurückgegeben und dem/der BenutzerIn im Browser angezeigt.

3.3.3 Hochladen von GPX Dateien

Für diejenigen BenutzerInnen der Applikation, die ein GPS-Gerät besitzen, ist diese Funktion sehr zeitsparend. Die durch das GPS-Gerät erstellte GPX-Datei kann direkt hochgeladen werden und die Daten werden automatisch benutzerbezogen in der Datenbank gespeichert.

```
1  <?php
2  $FileHandle = fopen($tmpFile,"r");
3  while (!feof($FileHandle))
4  {
5    $xmlData = $xmlData.fgets($FileHandle);
6  }
7  fclose($FileHandle);
8  $xml = new SimpleXMLElement($xmlData);
9
10 foreach ($xml->wpt as $waypoint)
11 {
12   //...
13 }
14 ?>
```

Listing 3.3: Hochladen einer GPX-Datei

Die GPX-Datei wird hochgeladen und anschließend mittels PHP Zeile für Zeile ausgelesen und in eine Variable gespeichert, siehe Listing 3.3. Diese Variable wird anschließend mit Hilfe einer „foreach“ Schleife ausgelesen und die Werte werden der Reihe nach in der Datenbank gespeichert. Zusätzlich wird das Erstellungsdatum in Form eines UNIX Zeitstempels und die eindeutige Nummer des Benutzers, welcher die Datei hochgeladen hat, gespeichert. Anschließend können zusätzliche Informationen, Bilder, Anhänge und Texte zu den Punkten hinzugefügt werden, welche aber, genauso wie die Punkte selbst, erst sichtbar sind, wenn sie von einem/einer AdministratorIn kontrolliert und freigeschalten wurden.

3.3.4 Erstellen eines Marker-Objekts

Um einen Punkt auf der Karte zu erstellen und zu speichern, muss der/die BenutzerIn zuerst eine Kategorie auswählen, welche einen Punkt erstellt.

```
1 <script type="text/javascript">
2  var map_center = map.getCenter();
3  marker = new google.maps.Marker({
```

```
    position: map_center,
    map: map,
    title:name,
    draggable:true,
    icon:pIcon
  });
marker.setMap(map);

google.maps.event.addListener(marker, 'dragend', function() {
  updateMarkerInfoWindow(infowindow, marker, id);
  infowindow.open(map, marker);
  latLng = marker.getPosition();
  document.getElementById('pLat').value = latLng.lat();
  document.getElementById('pLong').value = latLng.lng();
});
</script>
```

Listing 3.4: Einen Google Maps Marker erstellen

Anschließend wird mit der in Listing 3.4 dargestellten Javscript Funktion ein Marker auf die Karte gesetzt, welcher beliebig verschoben werden kann. Dieser Funktion wird als Positionsangabe die Kartenmitte, ermittelt durch die Funktion map.getCenter, der Name der Karte, der Name des Markers und die Bilddatei des Icons übergeben. Außerdem wird mit dem Schlüsselwort „draggable“ definiert, dass der Marker sich bewegen lässt. Nach dem Erstellen des Objekts wird eine Funktion definiert, welche bei dem Event „loslassen des Markers“ ausgeführt wird. Diese Funktion beinhaltet das Aktualisieren des zu dem Marker gehörenden Infowindows (siehe Abschnitt 2.4.1), d.h. die aktuellen Koordinaten des Markers werden in dem Infowindow gespeichert, um diese anschließend kontrollieren und speichern zu können. Nach dem der/die BenutzerIn die Positionierung beendet und die Koordinaten gespeichert hat, erscheint ein Dialogfenster, in das die zusätzlichen Informationen eingetragen werden können.

```
<?php
$dataObj       = new stdClass();
$dataObj->id     = NULL;
$dataObj->name     = $name;
$dataObj->dates    = $times;
$dataObj->counter  = 0;
$dataObj->published = 0;
$dataObj->userID   = $uID;

$db->insertObject("trees_geo_data", $dataObj);
$dataID = $db->insertid();

$query = "INSERT INTO trees_geo_points (gpoint, dataID, catID) VALUES (
    GeomFromText('POINT($lat $long)'),'".$dataID."','".$catID."')";
$db->setQuery($query);
$result = $db->query();
$pointID = $db->insertid();

foreach($variables as $var)
{
  if(isset($arrPost[$var]))
  {
    $id = explode("_", $var);
```

```
23      $id = $id[1];
24      $content= $arrPost[$var];
25      $query  = "INSERT INTO trees_geo_points_fields_content (content, fieldID
            , pointID) VALUES ('".$content."','".$id."', '".$pointID."')";
26      $db->setQuery($query);
27      $result = $db->query();
28    }
29  }
30  ?>
```

Listing 3.5: Marker in der Datenbank speichern

Wenn das Formular abgeschickt wird, wird der Punkt inklusive der Zusatzinformationen mit Hilfe von PHP in der Datenbank gespeichert. Die Programmierung ist in Listing 3.5 dargestellt, wobei diesem Codeauszug noch das Speichern der POST Variablen vorangeht. Die Validierung der POST Daten wird von Joomla selbst übernommen.

Um das Konzept des Speicherns nachvollziehen zu können, ist die Abbildung 3.2 und die dazugehörige Erklärung in Kapitel 3.2 unabdingbar. Zuerst wird ein Datenobjekt im Joomla eigenen Format erstellt, welches dann die Daten erhält. Anschließend werden die Daten in die Tabelle „trees_geo_data“ eingetragen. Die ID, welche der Eintrag in dieser Tabelle erhält, wird gespeichert, um im nächsten Schritt in der „trees_geo_points“ Tabelle gespeichert werden zu können. Zusätzlich werden hier die ID der Kategorie und die Koordinaten gespeichert. Die Koordinaten werden in dem MySQL Spatial Format und mit Hilfe der Funktion „GeomFromText“, ausführlich erläutert in Kapitel 2.1 auf Seite 2, gespeichert. Die ID, die dieser Eintrag erhält, wird wieder für den nächsten Eintrag gespeichert. In dem letzten Teil des Codeauszugs wird mit einer foreach Schleife über alle POST Daten, die den Namen „vars“ tragen und damit relevant sind für das Speichern, iteriert und anschließend werden diese in der Datenbank gespeichert. Damit ist das Speichern des Markers abeschlossen. Nun bietet die Applikation dem/der BenutzerIn darüber hinaus die Möglichkeit, Bilder, Anhänge und Texte zu speichern. Dies wird in Kapitel 3.3.7 erläutert.

3.3.5 Erstellen eines Polyline-Objekts

Das Erstellen eines Polyline Objekts ähnelt in den Grundzügen der Erstellung eines einzelnen Markers. Zuerst wird die entsprechende Kategorie ausgewählt und anschließend kann direkt auf der Karte eine Linie gezogen werden. Für diese Funktion bietet Google Maps das Polyline Objekt, beschrieben in Kapitel 2.4.1.

```
1 <script type="text/javascript">
2  line = new google.maps.Polyline({
3    strokeColor: '#ff0000',
4    strokeOpacity: 1.0,
5    strokeWeight: 2
6  });
7  line.setMap(map);
8  google.maps.event.addListener(map, 'click', addNewPointForLinestring);
9 </script>
```

Listing 3.6: Eine Google Maps Polyline erstellen

Um eine Linie auf der Karte einzeichnen zu können, muss ein „google.maps.Polyline“-Objekt erzeugt werden. Diesem werden, wie in Listing 3.6 dargestellt, die Farbe der Linie, die Transparenz der Linie und die Stärke der Linie als Argumente übergeben. Anschließend wird die Linie dem „map“-Objekt zugewiesen und die Google eigene Funktion „addNewPointForLinestring“ für das Event des „Klickens auf die Karte“ definiert. Diese Funktion fügt den zuletzt gesetzten Eckpunkt einem Array hinzu, um später auslesen zu können, welche Positionen die „Polyline“ beinhaltet.

3.3.6 Erstellen eines Polygon-Objekts

Die Implementierung des Erstellens eines Polygon Objektes in der Applikation ist nahezu identisch zu der Erstellung eines Polyline Objektes, die in dem vorherigen Kapitel 3.3.5 erklärt wurde. Trotzdem besteht ein wesentlicher Unterschied, denn das Polygon wird durch das „google.maps.Polygon“ Objekt erstellt. Das Polygon wird immer automatisch geschlossen und besitzt somit eine innere Fläche, für die eine zusätzliche Farbe und deren Transparenz definiert werden kann. Nach dem Hinzufügen eines Eckpunktes wird die Funktion „addNewPointForPolygon“ aufgerufen, welche den neuen Punkt dem Array für das aktuelle Polygon hinzufügt.

3.3.7 Hochladen von Bildern, Anhängen und Texten

Nach der Bestätigung der Position eines Markers, einer Polylinie oder eines Polygons erscheint ein Dialogfenster, welches dem/der BenutzerIn die Möglichkeit bietet, Bilder, Anhänge oder Texte hinzuzufügen. Die Web-Applikation bietet für das Hochladen von Bildern zwei verschiedene Möglichkeiten. Der/Die BenutzerIn kann entweder ein einzelnes Bild über ein normales HTML-Formular hochladen oder den Jumploader verwenden [8]. Letzere Option ist ein Java Applet, das unter Verwendung des HTTP-Protokolls mehrere Dateien nacheinander hochladen kann. Des Weiteren bietet es die Möglichkeit, das Bild vorher zu skalieren oder zu drehen. Bei der Einbindung des Applets in den PHP Programmcode können zusätzlich mehrere Parameter konfiguriert werden, wie z.B. die möglichen Dateiformate, die Darstellungsart und die Adresse, an die die Datei geschickt wird. Das bedeutet, jedes Bild wird an eine URL geschickt, welche anschließend das Bild verarbeitet bzw. speichert. Mittels einer PHP-Session werden vorher die essentiellen Daten für die Verarbeitung der Bilder gespeichert, um diese dann bildbezogen in der Datenbank speichern zu können. Zu diesen Daten gehören unter anderem die ID des Benutzers, welcher das Bild hochgeladen hat, und die ID des Dateneintrags, für welchen das Bild gespeichert werden soll. Vor der Verarbeitung wird außerdem noch die Klasse „thumbnail“ von Uwe Hölzel[1] eingebunden, welche das Skalieren von Bildern mittels PHP möglich macht.

```
1 <?php
2 $fname  = time()."_".$dataID."_".$counter.".jpg";
3 rename("$target_file_path","$fname");
4 copy("$target_file_path","../../images/trees_images/originals/
    $target_file_path");
```

[1] http://deruwe.de/vorschaubilder-einfach-mit-php-realisieren.html, (16.02.2011)

```php
$size = getimagesize("$target_file_path");

$resize = false;
if(($size[0] > 600) || ($size[1] > 600))
{
  $thumbnail = new thumbnail();
  $thumbnail->create("$target_file_path");
  $thumbnail->setQuality(80);
  ($size[0] > $size[1]) ? $thumbnail->resize("600") : $thumbnail->resize(
      false,"600");
  $thumbnail->save($fileName);
  $resize = true;
}
$thumbnail = new thumbnail();
$thumbnail->create("$target_file_path");
$thumbnail->setQuality(80);
$thumbnail->cube(70);
$thumbnail->save($fileNameThumb);

($resize == false) ? copy("$target_file_path","../../images/trees_images/
    normals/$target_file_path");
unlink("$target_file_path");
?>
```

Listing 3.7: Die Verarbeitung eines hochgeladenen Bilds

In Listing 3.7 ist ein Auszug aus dem PHP-Skript zum Verarbeiten der Bilder dargestellt. Nach dem Hochladen einer Datei wird der Dateiname geändert, um dem Problem von doppelt vorkommenden Dateinamen zu entgehen. Der neue Name besteht aus dem aktuellen UNIX-Zeitstempel, der ID des Datenobjekts und dem Wert einer Zählervariable, die die verarbeiteten Bilder zählt. Anschließend wird das Originalbild in den dafür vorgesehenen Ordner gespeichert. Für die Darstellung in der Applikation wird ein Bild mit der maximalen Größe von 600 Pixel und eine verkleinerte Version mit der Größe von 60 Pixel benötigt. Dazu wird zuerst die Bildgröße des Originals ausgelesen und anschließend wird das Bild auf die zwei Bildgrößen skaliert und in den jeweiligen Ordnern gespeichert. Ist das Original nicht größer als 600 Pixel, wird es nur kopiert und nur einmal, für das Vorschaubild mit 60 Pixel, verkleinert. Am Ende wird das hochgeladene Bild, welches sich noch in einem temporären Ordner befindet, gelöscht, und die Daten zu den erstellen Bildern werden in der Datenbank gespeichert.

Das Hochladen von Anhängen erfolgt auf die gleiche Weise wie bei Bildern, mit dem Unterschied, dass das Skalieren und das Erstellen eines Vorschaubilds wegfällt. Das Erstellen eines Textes für ein Objekt wurde realisiert über ein Dialogfenster und ein Textfeld, in das der Text eingegeben werden kann. Um die Textformatierung zu vereinfachen, wurde der plattformunabhängige WYSIWYG („What you see is what you get“) Editor TinyMCE[2] implementiert. Dieser auf Javscript basierende Editor ermöglicht das Formatieren von Texten mit mehreren Buttons und Menüs. Nach dem Einfügen des Textes wird dieser zusammen mit der entsprechenden ID des Objekts in der Datenbank gespeichert.

[2] http://tinymce.moxiecode.com, (20.02.2011)

3.3.8 Darstellen von Marker, Polylinien und Polygon Objekten

Die erstellten Objekte sind alle in der Datenbank gespeichert und werden bei Bedarf aus der Datenbank ausgelesen. Das Auslesen der Daten aus der Datenbank erfolgt mit PHP und anschließend werden die Daten an Javascript „weitergegeben“. Das Verfahren ist für Marker, Polylinien und Polygone fast identisch und wird aus diesem Grund in folgendem Listing 3.8 nur für Marker auszugsweise beschrieben.

```
1  <?php
2  if($session->get( 'showMarkers', '1' ) == "1")
3  {
4    $user    = &JFactory::getUser();
5    $uID     = $user->get('id');
6    $groups  = $user->get('groups');
7    $counter= 0;
8
9    /* Überprüfen der Berechtigung in der SQL-Abfrage */
10   $query = "SELECT a.id as id, a.dataID as dataID, x(gpoint) as x, y(gpoint)
         as y, b.name as category, b.id as catid, c.filename as iconfilename,
       d.name as pname" .
11     " FROM trees_geo_points a, trees_geo_points_cats b, trees_icons c,
         trees_geo_data d" .
12     " WHERE a.catID = b.id AND b.iconID = c.id AND d.id=a.dataID AND (d.
         published=1 OR d.userID='$uID' OR $admin=1)";
13   $rows = $db->setQuery($query);
14   $rows = $db->loadObjectList();
15   foreach ( $rows as $row ) {
16     /* Auslesen aller Bilder des Objekts */
17     $query    = "SELECT id,name FROM trees_images WHERE dataID='$dataID' AND
           (published=1 OR userID='$uID') ORDER BY ordering";
18     $pictures = $db->setQuery($query);
19     $pictures = $db->loadObjectList();
20       $content .= "<tr><td class=\"markerSpacer\"><td></tr><tr>";
21     foreach ( $pictures as $pic ) {
22       $content .= "<td><a href=\"javascript:showImage($pic->id);\"><img src
           =\"images/trees_images/thumbs/$pic->name\" class=\"markerImage\"
           /></a></td>";
23     }
24     $content .= "</tr>";
25
26     $map_output    .=  "<script type=\"text/javascript\">"
27             . "MarkersCreateArray[$counter] = new Array();"
28             . "MarkersCreateArray[$counter]['lat'] = '$row->x';"
29             . "MarkersCreateArray[$counter]['long'] = '$row->y';"
30             . "MarkersCreateArray[$counter]['name'] = '$row->pname';"
31             . "MarkersCreateArray[$counter]['icon'] = '$row->iconfilename';"
32             . "MarkersCreateArray[$counter]['catID'] = '$catID';"
33             . "MarkersCreateArray[$counter]['content'] = '$content';"
34             . "</script>\n";
35     $counter++;
36   }
37 }
38 ?>
```

Listing 3.8: Marker aus Datenbank auslesen

Das Grundprinzip dieser Programmierung in Kurzfassung ist: Die Daten für das jeweilige Objekt und dessen Zusatzinformationen sowie Bilder, Anhänge und Texte werden aus der Datenbank gelesen und mit PHP in ein Javascript Array übertragen. Dieses Array wird anschließend mit Javascript ausgelesen und die Objekte werden auf die Google Map gelegt. Der Grund für diese aufwendige Mischung der Programmiersprachen ist, dass man keine Google Maps Overlays, siehe 2.4.1, mit PHP erzeugen kann. In dem dargestellten Codeauszug in Listing 3.8 wird zuerst überprüft, ob die Anzeige der Marker auf der Karte vom dem/der BenutzerIn aktiviert oder deaktiviert ist. Anschließend wird die ID und die BenutzerInnengruppe mit der Joomla eigenen Funktion „get('id')“ und „get('groups')“ über das Benutzer Objekt ausgelesen, um in dem darauffolgenden SQL Statement prüfen zu können, ob der/die BenutzerIn berechtigt ist, das entsprechende Objekt zu sehen.

Des Weiteren wird mit der Variable „counter“ ein Zähler definiert, welcher notwendig für die Speicherung der Daten in dem Javascript Array ist. Das folgende SQL Statement liest unter anderem die ID der Objekte, die Koordinaten, die Kategorien und die Namen der Objekte aus der Datenbank. Anschließend werden mittels dieser ID alle Bilder, Texte, Anhänge und Datenfelder ausgelesen, welche zu dem Objekt gehören. In dem Listing 3.8 ist nur das Auslesen der Bilder dargestellt. Das Auslesen der zu dem Objekt gehörenden Texte, Anhänge und Datenfelder funktioniert allerdings auf die gleiche Art und Weise. All diese Informationen werden in der Variable „content“ gespeichert, für die spätere Anzeige im Infowindow, siehe Kapitel 2.4.1. Zum Abschluss wird diese Variable und weitere Informationen zu dem Marker von dem PHP-Skript als Javascript Code ausgegeben, um die Informationen in dem Javascript Array „MarkersCreateArray“ speichern zu können. Damit ist der PHP-Code zum Auslesen der Objekte beendet. Nach dem Laden der Applikation werden die Javascript Funktionen „showMarkers“, „showLinestrings“ und „showPolygons“ aufgerufen, welche das vorher gespeicherte Array auslesen und die entsprechenden Objekte mit ihren Informationen auf die Weltkarte legen. Das Erstellen von Markern, Polylinien und Polygonen ist in Kapitel 3.3.4, 3.3.5 und 3.3.6 beschrieben und erklärt.

3.3.9 Darstellung von Bildern, Anhängen und Texten

Bilder, Anhänge und Texte werden mit Hilfe eines Dialogfensters dargestellt, welches mit dem Javascript Framework Mootools[3] realisiert wurde.

```
1  <script type="text/javascript">
2   function showImage(picID){
3     var reqDialog = new MooDialog.Request('modules/mod_treesobject/getPicture.
          php?picID='+picID, null, {
4       'class': 'MooDialog showImage', autoOpen: false });
5     reqDialog.setRequestOptions({
6       onRequest: function(){
7         reqDialog.setContent('loading...');
8       }
9     }).open();
10  }
11  </script>
```

Listing 3.9: Darstellung der Bilder von Objekten

[3] http://mootools.net, (25.02.2011)

Nach dem Klick auf einen Bilderlink eines Objekts wird die Funktion „showImage“ aufgerufen und als Argument wird die ID des Bilds übergeben. Die Funktion „showImage“, dargestellt in Listing 3.9, erstellt ein Dialogfenster dessen Design über CSS gesteuert werden kann. Dieses Fenster holt sich den Inhalt mittels asynchronem Zugriff auf eine URL. Das bedeutet, die Seite muss nicht neu geladen werden, um das Bild anzeigen zu können. Die URL übergibt als HTTP-GET-Argument die eindeutige ID des Bilds an die aufgerufene PHP Datei. Diese wiederum liest den Pfad des Bilds mit Hilfe der ID aus der Datenbank und gibt das Bild aus, welches dann in dem Dialogfenster angezeigt wird. Die Anzeige von Texten und Anhängen erfolgt auf ähnliche Art und Weise: Bei Texten wird der Text angezeigt, bei Anhängen nur der Link zu der Datei, welcher sich dann in einem neuen Fenster öffnet.

3.3.10 Freischalten, Bearbeiten und Löschen von Objekten

Jeder/Jede BenutzerIn hat die Möglichkeit, nach einer kostenlosen Registrierung die Web-Applikation in vollem Umfang zu nutzen. Das heißt, Objekte erstellen, Dateien und/oder Bilder hochladen und Texte hinzufügen. Um trotzdem die Kontrolle über das System zu behalten und um Missbrauch vorzubeugen, gibt es verschiedene Freigabelevel und BenutzerInnenebenen.

- BesucherIn
- Registrierter/Registrierte BenutzerIn
- ModeratorIn
- AdministratorIn

Nach der Registrierung an der Applikation und der Freischaltung durch einen/eine AdministratorIn erhält der/die BenutzerIn den Status „Registriert“ und kann Objekte etc. erstellen. Diese sind danach nur für den/die BenutzerIn selbst sichtbar. Erst nach der Kontrolle und Bestätigung des Objekts und dessen Informationen durch einen/eine ModeratorIn oder durch einen/eine AdministratorIn wird dieses für alle BenutzerInnen sichtbar. Ein/Eine AdministratorIn ist außerdem berechtigt, Benutzer zu löschen und registrierte BenutzerInnen in den Status des/der Moderators/Moderatorin zu erheben.

3.3.11 Bearbeiten, Löschen und Hinzufügen von Kategorien

Um in der Applikation einen Punkt, eine Polylinie oder ein Polygon erstellen zu können, muss vorher eine entsprechende Kategorie erstellt werden. Eine Kategorie kann beispielsweise ein Baum, eine Allee oder ein Wald sein. Das Bearbeiten, Löschen und Hinzufügen dieser Kategorien ist den Administratoren/Administratorinnen vorbehalten. Für jede angelegt Kategorie muss ein Icon festgelegt bzw. hochgeladen werden, um die Kategorie auf der Karte eindeutig kenntlich zu machen.

Kapitel 4

Zusammenfassung und Ausblick

Die beschriebene Web-Applikation gibt den BenutzerInnen die Möglichkeit, entdeckte historische Bäume, Wälder oder auch Alleen mit mehreren historischen Bäumen in einer Weltkarte zu positionieren und mit Zusatzinformationen wie z.B. Größe, Alter, Bilder, Anlagen und/oder Texten zu speichern. Die Objekte können außerdem geändert oder gelöscht werden. Des Weiteren sind Administratoren/Administratorinnen dazu berechtigt, die von registrierten BenutzerInnen erstellten Objekte zu ändern, zu erweitern oder zu löschen. Die Anforderungen an die Applikation, welche vor Beginn des Projekts festgelegt wurden, sind also vollständig erfüllt. Die Applikation wurde so aufgebaut und entwickelt, dass späteren Erweiterungen oder Veränderungen nichts im Wege steht und auch andere Programmierer diese durchführen bzw. implementieren können.

Das Thema geographische Informationssysteme ist ein sich ständig weiter entwickelndes und immer beliebter werdendes technisches Gebiet, und somit wird der Umfang der Applikation wahrscheinlich immer größer und weiter fortgeschritten. Außerdem gibt es in dem Themengebiet der Web-Programmierung immer wieder Veränderungen und Neuerungen, welche auch in dieser Applikation mit Sicherheit implementiert werden könnten. Ein weiteres, zukünftiges Ziel für die Applikation könnte die Vernetzung mit sozialen Netzen wie z.B. Facebook und Twitter sein, um Internetbenutzer auf die Applikation aufmerksam zu machen, die die Web-Applikation „Historische Bäume und Wälder“ noch nicht kennengelernt haben. Des Weiteren ist ein Blog in Planung, welcher aktuelle Neuigkeiten rund um das Thema historische Bäume und Wälder dokumentieren soll. Auch hierbei können soziale Netze eine große Rolle spielen und für eine großflächige Verteilung der Informationen sorgen.

Der Bereich „Mobile Geräte“ kann als Zukunftsvision für die Applikation angesehen werden. Dieses Thema ist in der heutigen Zeit immer wichtiger, weil mehr und mehr Menschen mobile Geräte wie z.B. das iPhone, ein Android-Handy oder ein Windows Phone besitzen und vor allem auch die meiste Zeit das Gerät in unmittelbarer Nähe haben. Aus diesem Grund würde die Programmierung einer App für mobile Geräte, welche ebenfalls die Möglichkeit zur Speicherung von historischen Bäumen und Wäldern bietet, mit Sicherheit die Anwenderzahl und die Beliebtheit erhöhen.

Literaturverzeichnis

[1] Oracle Corporation. Raumbezogene Erweiterungen in MySQL. `http://dev.mysql.com/doc/refman/5.1/de/spatial-extensions.html` (15.02.2011).

[2] H. Faeskorn-Woyke, B. Bertelsmeier, P. Riemer, and E. Bauer. *Datenbanksysteme*. Pearson Education Deutschland GmbH, München, 2007.

[3] Oracle Corporation. MySQL Workbench. `http://www.mysql.de/products/workbench/` (05.02.2011).

[4] Hagen Graf. *Joomla! 1.6 - das Einsteigerbuch*. Pearson Education Deutschland GmbH, München, 2011.

[5] Hagen Graf. *Joomla! 1.5*. Pearson Education Deutschland GmbH, München, 2008.

[6] Google. Google maps javascript api v3. `http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/` (02.02.2011).

[7] L. Mahe. MarkerClusterer for google Maps v3. `http://gmaps-utility-library-dev.googlecode.com/svn/tags/markerclusterer` (10.03.2011).

[8] JMaster. Java file upload applet. `http://jumploader.com` (19.02.2011).

Anhang A

Anhang

A.1 Die grafische Benutzeroberfläche der Web-Applikation

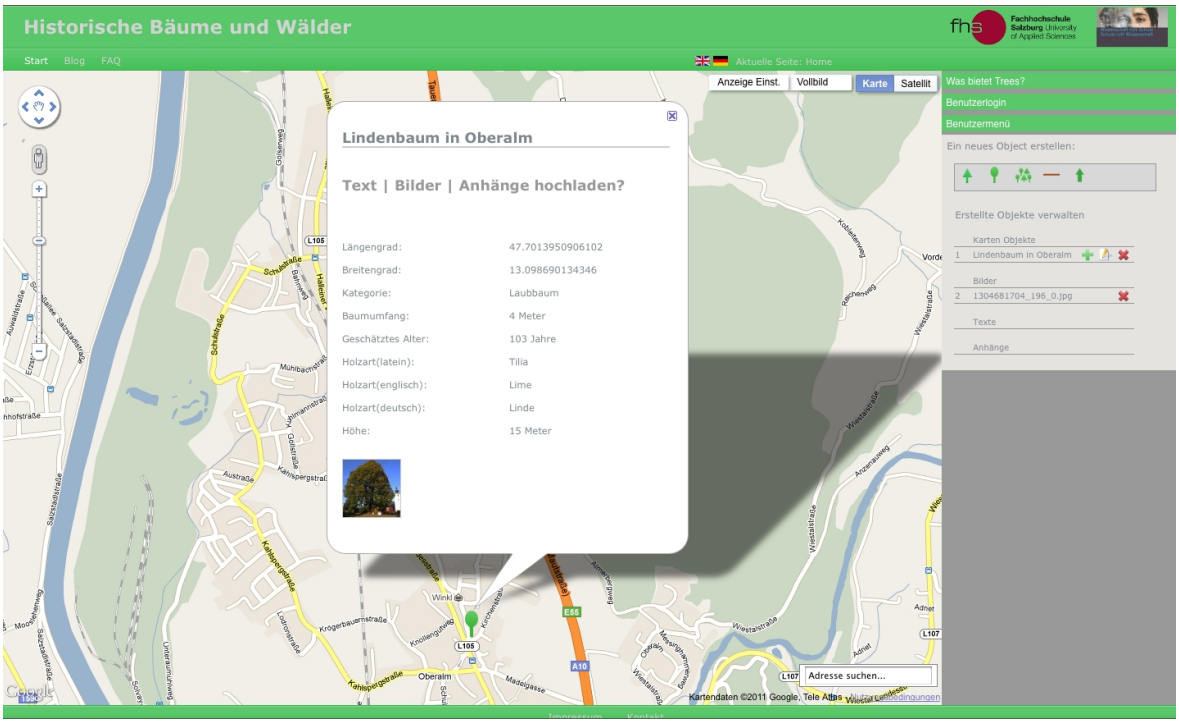

Abbildung A.1: Die Benutzeroberfläche mit Darstellung eines Markers

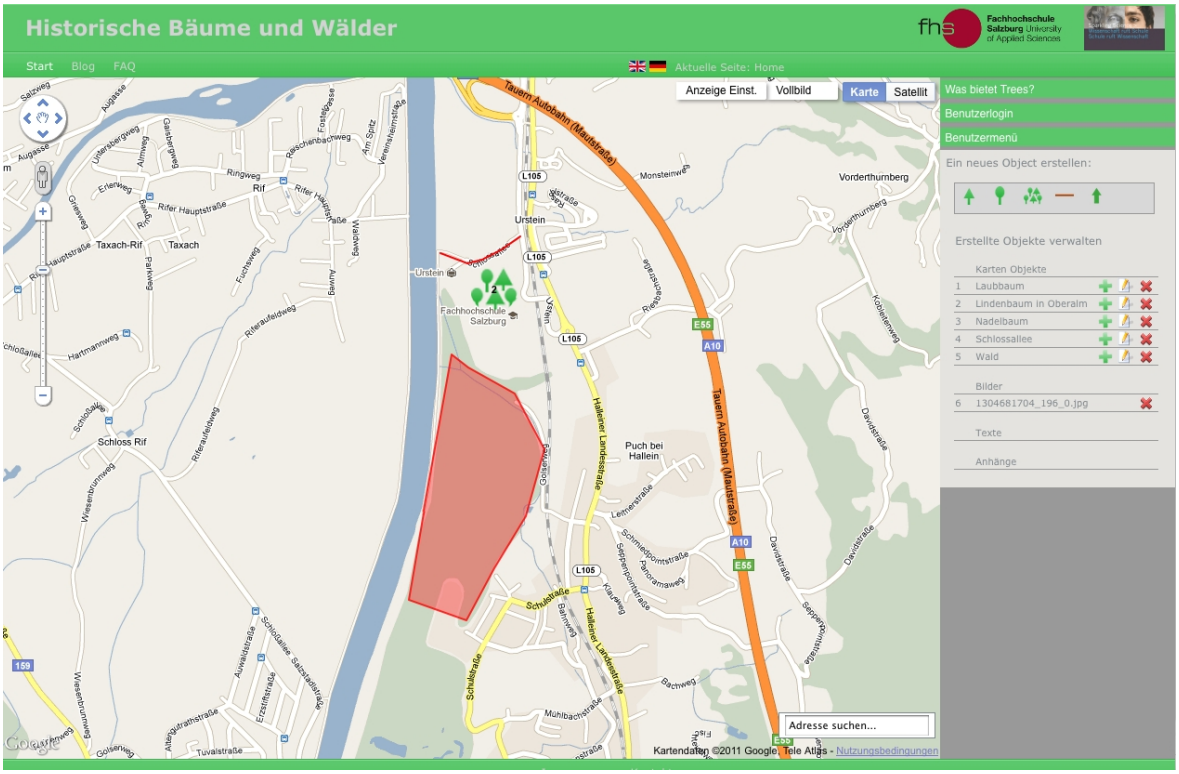

Abbildung A.2: Die Benutzeroberfläche mit verschiedenen Objektarten