Berufsakademie Sachsen
Staatliche Studienakademie Leipzig

Umkreissuche und die geographische Darstellung der Ergebnisse

Praxisarbeit in der Fachrichtung Informatik
im 4. Semester

Eingereicht von: Christian Gräfe
Demmeringstaße 8b
04177 Leipzig
Seminargruppe: IT05
Matrikelnummer: 205830

Betreuer: Prof. Dr. habil. Harald Englisch
Beratung Medizin-Informatik
Burghausener Str. 18
04178 Leipzig, OT Böhlitz-Ehrenberg

Leipzig, 14.07.07

Inhaltsverzeichnis

1 Einführung / Problemstellung

In der heutigen Konsum- und Informationsgesellschaft ist es immer wichtiger geworden, zu wissen, wo und wie man Informationen findet. Wo gibt es die neuesten Nachrichten? Wofür steht eigentlich der Begriff „...“? Wie komme ich am schnellsten an mein Ziel? Für die meisten Fragen dieser Art gibt es bereits gute Lösungen im Internet, so zum Beispiel spiegel.de, wikipedia.de oder map24.de.

Um eben so eine Frage geht es auch in dieser Praxisarbeit, nämlich das Finden von Adressen in meiner Nähe bzw. in der Nähe eines bestimmten Ortes. Diese Art von Suche wird als Umkreissuche bezeichnet. Um dieses Thema nicht zu groß zu fassen, wird das prinzipielle Vorgehen und die Umsetzung einer solchen Umkreissuche am Beispiel einer Kliniksuche im Bundesland Sachsen demonstriert.

Im Internet gibt es bereits Kliniksuchen, wie zum Beispiel klinik-lotse.de oder deutsches-krankenhaus-verzeichnis.de. Die meisten Seiten bieten allerdings nur eine Suche nach dem Ort oder der Postleitzahl an. Man bekommt nur ungenaue Angaben über die Entfernung. Wenn man es genau wissen will, muss man erst die Strecke im Routenplaner berechnen lassen. Es wäre doch viel praktischer, wenn man die genaue Entfernungsangabe und am besten gleich noch die Route zu der Klinik als Ergebnis der Suche bekommt. Die Umsetzung so einer Suche wird in dieser Praxisarbeit erläutert. Dabei wird der Punkt Routenplaner ausgelassen, da die Karten-API, welche in diesem Beispiel verwendet wird, so eine Routenplanung mit einem Funktionsaufruf erledigen kann.

Ein weiterer Punkt ist die Darstellung der Ergebnisse in einer übersichtlichen Karte. Fast alle Kliniksuchen geben zu den Ergebnissen nur die Adresse an, aber keine Darstellung der Adresse in einer Karte. Die Anzeige einer Karte mit den Ergebnissen wird ebenfalls Bestandteil dieser Praxisarbeit sein.

Damit das erarbeitete Ergebnis problemlos weiterverwendet werden kann, soll, wenn möglich, eine Trennung von Code und Design vorgenommen werden.

Nach der Beschreibung der vorhandenen Gegebenheiten und der zum Einsatz kommenden Techniken folgt die Erläuterung, auf welche Art die vorhanden Daten verarbeitet und angezeigt werden. Im letzten Kapitel befinden sich eine Auswertung des Ergebnisses sowie Anmerkungen zu einer möglichen Erweiterung dieser Lösung.

2 Vorbetrachtungen

2.1 Gegebenheiten

Wie bereits in der Einleitung erwähnt, beschränkt sich die Umkreissuche auf Kliniken im Raum Sachsen. Die Adress- und Kontaktdaten der Kliniken befinden sich in einer Tabelle in einer MySql-Datenbank. In dieser Tabelle befinden sich nicht nur Kliniken, sondern auch noch weitere medizinische Einrichtungen. Diese Tabelle wird von der Internetseite www.gesundheit-sachsen.de für die Adresssuche genutzt. Damit am Quell-code des Gesundheitsportales „Gesundheit-Sachsen“ keine Änderungen vorgenommen werden müssen, bleibt die Struktur der Adresstabelle so wie sie ist. Um für die Kliniken allerdings weitere Informationen bereit zu stellen, wird eine zweite Tabelle angelegt, die diese Informationen unter dem gleichen Primärschlüssel wie die Adresstabelle speichert.

Die Internetseite wird auf einem Linux-System mit installiertem Apache, MySql und PHP gehostet (LAMP). Da sich auf dem Server PHP in der Version 5 befindet, können objektorientierte Ansätze für die Umsetzung einer Lösung genutzt werden. Für die Internetseite wurde die Domain www.kliniksuche-sachsen.de registriert.

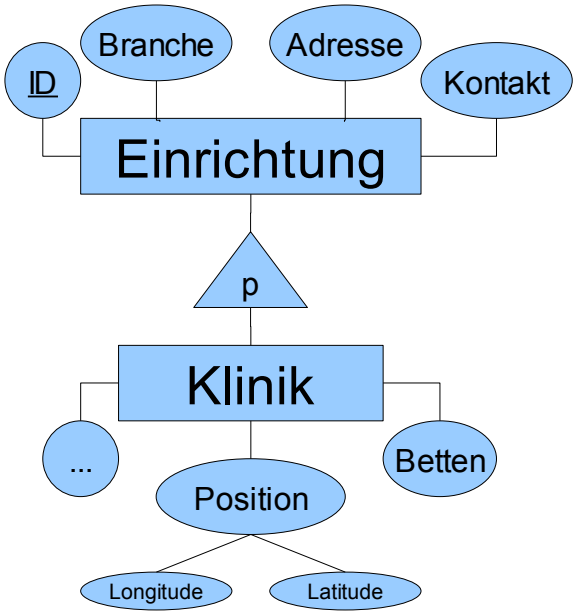

Abbildung 1: vereinfachtes ERD der Kliniken

2.2 Die verwendeten Techniken

2.2.1 Karten-API (Map24)

Für die Darstellung der Suchergebnisse und für die Bestimmung von geographischen Koordinaten wird eine externe Software benötigt. Zur Darstellung von Kartendaten gibt es mehrere Anbieter. Da das Ganze allerdings so wenig wie möglich kosten soll, beschränkt sich das Angebot auf zwei Anbieter. Das sind zum einen Google[1] und zum anderen Map24[2]. Beide Firmen bieten eine freie API (Application Programming Interface) ihrer Kartensoftware an, wobei kaum Unterschiede zwischen den beiden APIs bestehen.

Die Nutzung der APIs wird dahingehend beschränkt, dass nur eine bestimmte Anzahl an Transaktionen erlaubt ist. Mit Transaktionen ist das Abfragen von Koordinaten, das Berechnen einer Route und ähnliches gemeint. Beide APIs haben eine sehr ausführliche Dokumentation ihrer Funktionen und Objekte. Des Weiteren gibt es zahlreiche Tutorials zur Nutzung der jeweiligen API. Damit die Schnittstellen genutzt werden können, wird ein kostenloser Lizenzschlüssel benötigt, welcher allerdings nur für eine Domain gilt. Die APIs werden durch JavaScript genutzt. Es muss nur eine JavaScript-Datei, welche auf dem Server des Kartenanbieters liegt, mithilfe des HTML-Tags <script> in die Seite eingebunden werden, damit man auf die Funktionen und Objekte der API zugreifen kann.

Die Wahl der API fiel schließlich auf die Map24 API, da sie ausgereifter wirkt, einen besseren Routenplaner besitzt und zum Teil genauere Daten liefert, vor allem bei der Bestimmung von Hausnummern.

1 http://www.google.com/apis/maps/
2 http://devnet.map24.com

2.2.2 AJAX (JavaScript + XML + XMLHttpRequest)

Die Verwendung von JavaScript, asynchronem Nachladen von Seitenelementen mittels XMLHttpRequest und XML sind keine neuen Techniken. Allerdings hat die Zusammenarbeit dieser Techniken einen Namen bekommen: **AJAX**.[3]

JavaScript[4] ist eine objektbasierte Skriptsprache zum dynamischen Darstellen bzw. Ändern von Inhalten einer Webseite. Mit JavaScript kann das *Document Object Model* (DOM)[5] der aktuell geöffneten Internetseite geändert werden. Wenn der Browser eine HTML-Datei öffnet, wird der HTML-Code in das DOM übersetzt und dann angezeigt. Das DOM beinhaltet die Baumstruktur der HTML-Seite und kann jederzeit, durch das Einfügen oder Entfernen von Knoten mittels JavaScript, erweitert bzw. verringert werden.

Das DOM ist nicht nur für HTML, sondern auch für **XML** (*Extensible Markup Language*)[6] gedacht. XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird bevorzugt für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell beim Austausch von Daten über das Internet.

Bei dem klassischen Aufruf einer Internetseite, wird eine HTTP-Anfrage an den Webserver gestellt und der Nutzer wartet, dass der Server die angeforderte Webseite liefert. Wenn der Nutzer auf einen Link klickt, erfolgt dies von Neuem. Der Nutzer muss also immer warten, bis die Seite vom Server geladen ist. Um eben dieses Warten zu vermeiden, kann man solche Abfragen mittels **XMLHttpRequest**[7] asynchron ausführen. Mithilfe des XMLHttpRequest-Objektes kann eine Anfrage an den Webserver gerichtet werden. Wenn das Ergebnis der Anfrage an den Browser übermittelt wurde, wird eine vorher definierte Call-Back-Funktion ausgeführt. Diese Funktion ändert dann das DOM der Webseite und fügt die neuen Daten ein. Das XMLHttpRequest-Objekt kann das vom Server übermittelte Ergebnis als Plaintext oder als in DOM umgewandeltes XML ausgeben.

3 http://de.wikipedia.org/wiki/Ajax_(Programmierung)
4 http://de.wikipedia.org/wiki/JavaScript
5 http://de.wikipedia.org/wiki/Document_Object_Model
6 http://de.wikipedia.org/wiki/Extensible_Markup_Language
7 http://www.galileocomputing.de/openbook/javascript_ajax/18_ajax_002.htm

2.2.3 MySql, PHP, Webservice

Die Adressdaten für die Kliniken befinden sich, wie bereits erwähnt, in einer **MySql** Datenbank, welche auf einem anderen Server, als das eigentliche Projekt liegt. Bei MySql handelt es sich um ein Relationales Datenbankverwaltungssystem[8]. Das MySql Datenbankmanagementsystem (DBMS) ist vor allem im Bereich der Internetanwendungen sehr verbreitet. Das DBMS wird meist in Kombination mit Linux, Apache und PHP eingesetzt (LAMP)[9].

Bei **PHP** handelt es sich um eine serverseitig interpretierte Skriptsprache, die vor allem bei der Erzeugung von dynamischen Webseiten zum Einsatz kommt. PHP verfügt nativ über Funktionen und Klassen für die Arbeit mit Datenbanken und XML-Daten. Für eine objektorientierte Programmierung (OOP) sollte möglichst PHP Version 5 verwendet werden, da erst ab Version 5 grundlegende OOP-Konzepte wie Kapselung der Daten und Destruktoren unterstützt werden. Des Weiteren sind einige wichtige Teile der Standardbibliothek, wie etwa die DOM-API, seit PHP 5 objektorientiert.[10]

Da man mit JavaScript nicht direkt zu einer Datenbank verbinden kann, nutzt man so genannte **Webservices**[11]. JavaScript sendet mittels XMLHttpRequest eine Anfrage an einen Webservice und erhält von diesem als Ergebnis Daten in einer XML-Struktur. Diese XML-Daten kann JavaScript auswerten und in die aktuelle Seite einbinden. In meinem Beispiel, wird der Webservice ein PHP-Skript sein, welches die eigentliche Umkreissuche durchführt, die Ergebnisse der Suche in eine XML-Struktur umwandelt und diese ausgibt bzw. übermittelt.

8 http://de.wikipedia.org/wiki/MySQL
9 http://de.wikipedia.org/wiki/LAMP
10 http://de.wikipedia.org/wiki/PHP
11 http://de.wikipedia.org/wiki/Webservice

3 Umsetzung

3.1 Nutzung der Seite

Nach dem Aufrufen der Internetseite sieht der Benutzer auf der linken Seite eine Karte von Sachsen und rechts ein Formularfeld für die Eingabe seiner Adresse, des Suchradius und seiner Klinikauswahl (Allgemein, Rehakliniken, Privatkliniken). Im Formularfeld befindet sich ebenfalls ein Button „Suchen“, mit einem Klick auf diesen Button startet die Suche.

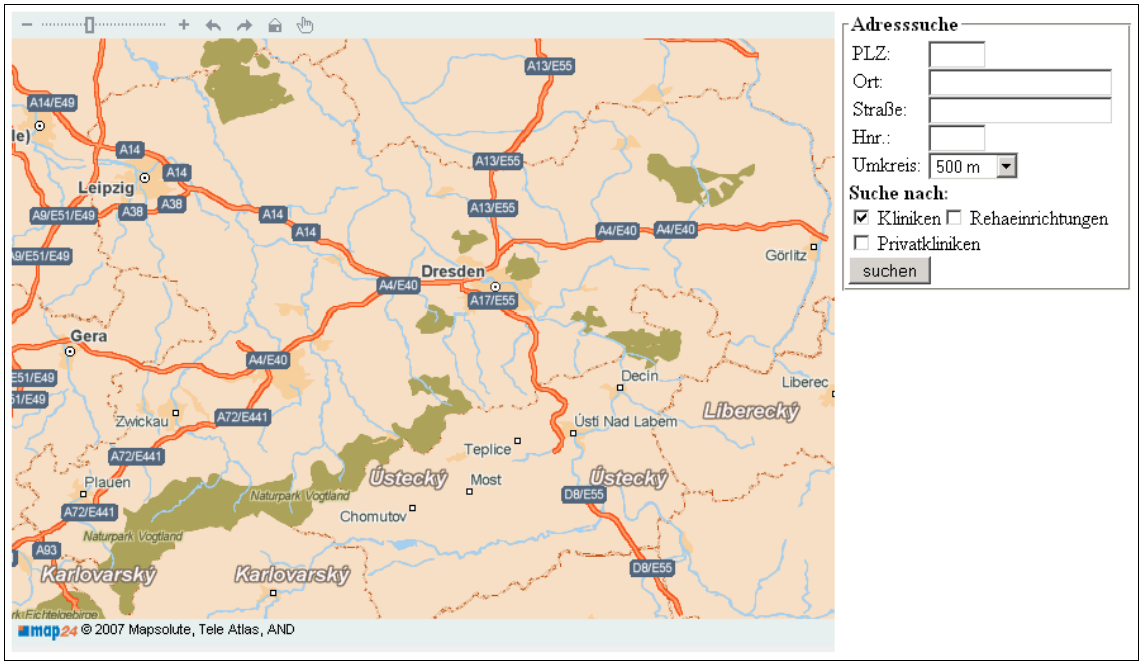

Abbildung 2: Nach dem Aufruf der Seite

Nach Erfolg der Suche ist nicht mehr die gesamte Karte von Sachsen zu sehen, sondern nur der Kartenausschnitt mit dem entsprechenden Umkreis. Auf der Karte sieht der Benutzer seine angegebene Adresse, dargestellt durch einen roten Punkt und einen durchsichtigen roten Kreis, welcher den Suchradius widerspiegelt. In diesem Kreis befinden sich nun, je nach Ergebnis der Suche, die Kliniken, dargestellt durch unterschiedlich farbige Punkte. Fährt der Benutzer mit dem Mauszeiger über den Punkt einer Klinik, werden unter dem Adressformular Informationen zu der ausgewählten Klinik angezeigt.

Unterhalb der Karte ist nach der Suche eine Legende zu sehen, welche die Bedeutung der unterschiedlich eingefärbten Punkte erläutert. Unter der Legende befindet sich die gesamte Ergebnisliste, wobei die Kliniken nach der Entfernung zum Suchpunkt (eingegebene Adresse des Benutzers) aufsteigend sortiert sind.

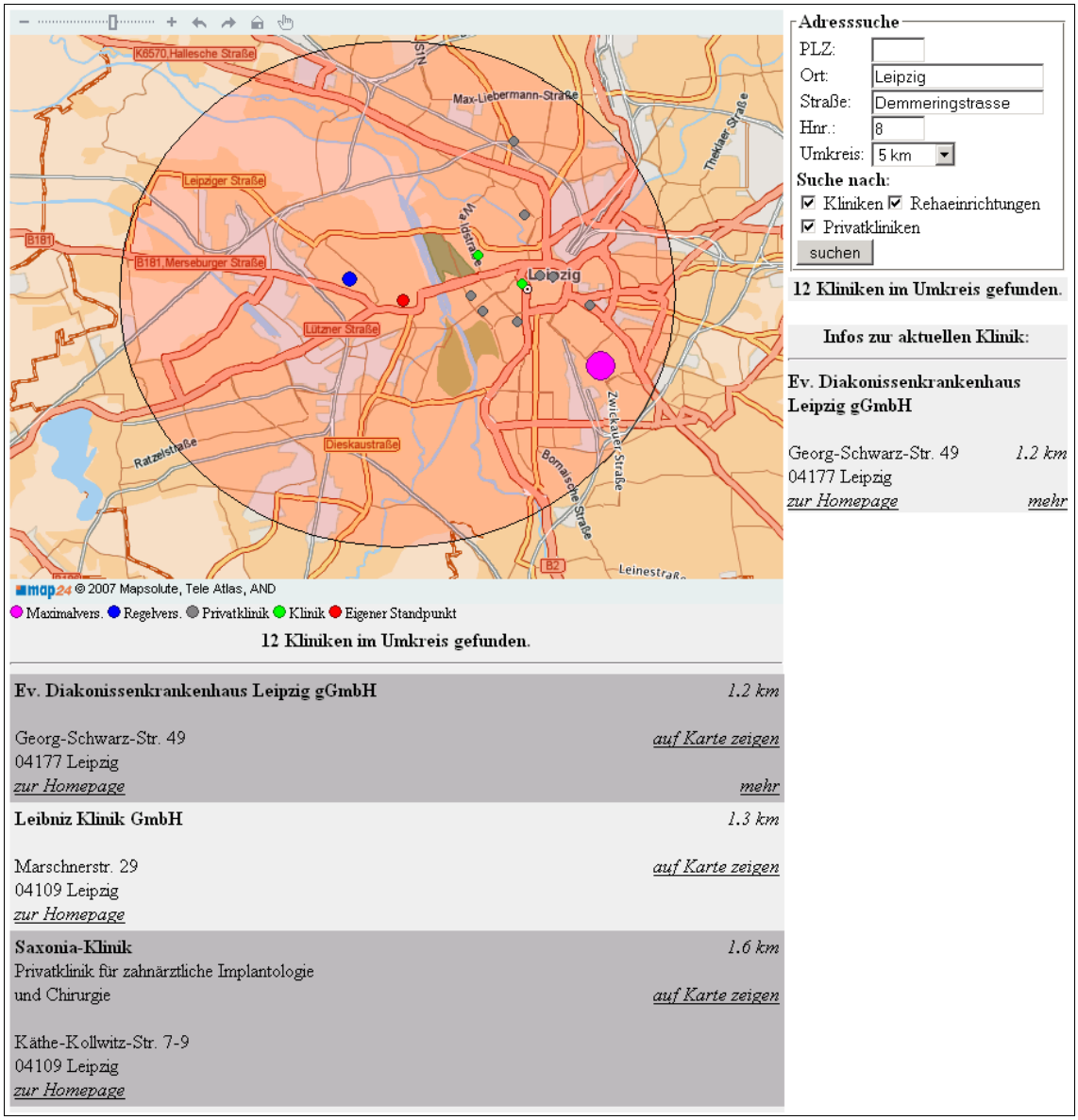

Abbildung 3: Nach der Suche (Auszug)

Zu jeder Klinik wird die Bezeichnung, die Adresse und ein Link zur Homepage angezeigt. Für Kliniken, die im „Sächsischen Krankenhausregister“ eingetragen sind, wird ein Link „mehr“ angezeigt, welcher auf eine Detailseite auf der Internetseite www.krankenhausregister-sachsen.de verweist.

In der Ergebnisliste steht zusätzlich der Link „auf Karte anzeigen“ zur Verfügung. Mit einem Klick auf diesen Link, wird die entsprechende Klinik auf der Karte zentriert und herangezoomt.

Der gesamte Programmablauf erfolgt auf einer Seite, welche permanent angezeigt wird. Es erfolgt kein HTTP-Refresh. Jede Änderung des Inhaltes wird dynamisch mittels Java-Script realisiert.

3.2 Allgemeiner Programmablauf

Zu der im vorangegangenem Kapitel beschriebenen Nutzung der Seite folgt nun der interne Programmablauf bei dem Durchführen einer Suche.

Die JavaScript-Funktionen für die Internetseite sind auf mehrere Dateien verteilt. So gibt es zum Beispiel eine Datei „xmlhttp.js“. Diese beinhaltet die Funktion *getXmlHttp()*. Da das Erzeugen eines XMLHttpRequest-Objekts nicht in allen Browsern gleich ist, kümmert sich die Funktion *getXmlHttp()* darum, dass das Objekt je nach Browser erzeugt wird. Im Firefox und Opera sowie dem Internet Explorer (IE) 7 wird das XMLHttpRequest-Objekt durch *new XMLHttpRequest()* erzeugt, in früheren Versionen des IE allerdings durch *new ActiveXObject("Msxml2.XMLHTTP")*.

Weitere Dateien sind „geocode.js“ für das Geocoding der eingegebenen Adresse (3.3.1), „XmlToObject.js“ für das Umwandeln der XML Daten in ein JavaScript-Objekt (3.3.3) und „show_results.js“ für das Anzeigen der Ergebnisliste (3.4). Am Wichtigsten ist allerdings das Skript, welches direkt vom Map24-Server eingebunden wird, in ihm befindet sich die eigentliche Karten-API.

Beim Durschführen einer Suchanfrage geschieht folgendes:

1. Die eingegebene Adresse wird an den Map24-Server gesendet und dieser liefert ein Array mit möglichen Koordinaten.
2. Das Array mit den Koordinaten wird gefiltert und die ersten Koordinaten als Ausgangspunkt genommen.
3. Ausgehend von diesem Startpunkt wird eine Umkreissuche in dem vorher definierten Radius gestartet.
4. Die Umkreissuche liefert XML-Daten mit den Kliniken im Umkreis und ihre Entfernung zum Startpunkt. Die Kliniken sind aufsteigend nach ihrer Entfernung sortiert.
5. Die XML Daten werden in ein JavaScript-Array von Klinik-Objekten umgewandelt. Das Klinik-Objekt ist ein selbst definiertes Objekt, welches eine ähnliche Struktur wie die XML-Daten hat.
6. Die Kliniken werden auf der Karte und in der Ergebnisliste angezeigt.

3.3 Backend

3.3.1 Geocoding der eingegebenen Adresse

Mit einem Klick auf den Suchen-Button wird die Umkreissuche gestartet. Damit die Suche aber auch ein Ergebnis liefern kann, muss erst die eingegebene Adresse in Koordinaten umgewandelt werden.

Mit einem Klick auf „Suchen“ werden die Daten aus den Formularfeldern ausgelesen und in Variablen gespeichert. Es wird überprüft ob die Stadt bzw. die Postleitzahl eingegeben ist und ob mindestens eine Klinikart zur Suche ausgewählt wurde.

Als Zweites geschieht allerdings nicht das Umwandeln der Daten, sondern das Entfernen der Ergebnisse einer vorangegangenen Suche. Die Objekte auf der Karte, wie zum Beispiel die Anzeige des Umkreises und der eigene Standpunkt, werden entfernt. Die div-Container für die Ergebnisliste, für die Legende und für die aktuelle Klinik werden, durch das Entfernen aller Kindknoten, geleert.

Erst jetzt werden die Adressdaten an die Funktion *geocode()* zum Umwandeln übergeben. Hier werden die Daten asynchron verarbeitet. Die Funktion selber liefert kein Ergebnis zurück, stattdessen übergibt man der Funktion als Variable eine CallBack-Funktion. Diese CallBack-Funktion wird nach der Umwandlung der Adressdaten in Koordinaten mit dem Ergebnis der Umwandlung als Parameter aufgerufen.

Die Funktion *geocode()* übergibt die Adressdaten an einen Webservice von Map24 und setzt als CallBack-Funktion *filterLocations()*. In dieser Funktion werden die Ergebnisse, die der Map24 Webservice liefert, nochmals nach Postleitzahl und Straße gefiltert. Der Map24-Geocode-Webservice liefert für jede Anfrage eine Auswahl an in Frage kommenden Ergebnissen. Gibt man zum Beispiel nur eine Postleitzahl ohne Ort an, werden alle Orte gefunden, die diese PLZ haben. Die Filterfunktion ruft schließlich die CallBack-Funktion, welche an *geocode()* übergeben wurde, auf und übergibt die gefilterten Ergebnisse.

Aus dem Array mit möglichen Ergebnissen wird schließlich, der Einfachheit halber, das erste Ergebnis als Ausgangspunkt für die Suche angenommen und auf der Karte durch einen Roten Punkt dargestellt. Der Suchradius wird ebenfalls um diesen Punkt dargestellt. Nun kann die Umkreissuche gestartet werden.

3.3.2 Kliniken im Umkreis suchen und als XML ausgeben

Die eigentliche Umkreissuche erfolgt in einem PHP-Skript, welches auf dem gleichen Webserver liegt wie die HTML- und JavaScript-Dateien. Dieses PHP-Skript wird mittels JavaScript und XMLHttpRequest aufgerufen. Als Parameter werden die Koordinaten des Ausgangspunktes, der Umkreis und einige weitere Filterkriterien per GET, also direkt in der URL, übergeben.

Im PHP-Skript wird eine Verbindung zur Datenbank hergestellt und alle Kliniken werden, mit Hilfe des SQL Befehls *„SELECT * FROM kliniken LEFT JOIN daten1 ON kliniken.id = daten1.id“* (vgl. Kapitel 2.1), ausgelesen. Der SQL-Befehl kann je nach den angegeben Filterkriterien mit Hilfe von *WHERE* variiert werden.

Es wird von allen Kliniken die Entfernung (Luftlinie) zum Startpunkt ermittelt. Befindet sich die Klinik im geforderten Radius, wird sie mit ihren Daten (Adresse, Kontakt usw.) und der Entfernung in einem Array gespeichert. Dieses Array wird anschließend, nachdem alle Entfernungen berechnet wurden, aufsteigend sortiert. Die Klinik mit der kürzesten Entfernung steht nun als Erstes.

Aus diesem sortierten Array werden nun die XML-Daten erzeugt. Dies geschieht unter Verwendung der PHP internen DOM-API. Da die API objektorientiert arbeitet, ist die Erstellung des XML-Baumes sehr einfach möglich.

Als Erstes wird ein Objekt vom Typ *DOMDocument* erzeugt, dieses Objekt ist der Ausgangspunkt des späteren XML und stellt wichtige Funktionen für die Arbeit mit Knoten zur Verfügung. So kann mittels *createElement()* ein Knoten erzeugt werden und dieser anschließend mit *appendChild()* an entsprechender Stelle im Baum eingefügt werden. Ein Beispiel, für die XML-Struktur der Klinikdaten findet sich im Anhang.

```
// XML-Objekt und Wurzel
$xml = new DOMDocument('1.0','UTF-8');
$root = $xml->createElement("daten");
$xml->appendChild($root);

// Datensatz für Klinik anlegen
$ds = $xml->createElement("ds");
$root->appendChild($ds);
...

// XML ausgeben
echo $xml->saveXML();
```

Sind alle Kliniken in den Baum eingefügt, kann das XML ausgegeben werden.

3.3.3 Ein JavaScript-Objekt für die XML-Daten

Ändert sich der Status der HTTP-Anfrage, welche vom XMLHttpRequest-Objekt aus aufgerufen wurde, wird eine CallBack-Funktion ausgeführt. Auf diese Weise kann überprüft werden, ob die Anfrage an das PHP-Skript erfolgreich war, und ob das Ergebnis schon vorliegt. Sobald das PHP-Skript das Ergebnis übermittelt hat, steht in der Variable *responseXML* des XMLHttpRequest-Objektes die Schnittstelle zum DOM der Daten bereit.

Damit das XMLHttpRequest-Objekt für weitere Anfragen zur Verfügung steht und das Handling der Klinikdaten einfacher vonstatten geht, werden die Daten in einem Array gespeichert. Für diesen Zweck wird ein Klinikobjekt definiert, welches die Daten einer Klinik strukturiert zur Verfügung stellt. Das Array beinhaltet schlussendlich die Klinikdaten in Form dieser Klinikobjekte.

Objekte in JavaScript werden, ähnlich wie Funktionen, mit dem Schlüsselwort *function* definiert. Membervariablen des Objektes werden in der Objektdefinition mit *this* gekennzeichnet. Um eine Instanz des Objektes zu erzeugen, wird der *new* Operator benötigt.

```
// Definition des Objektes
function klinik()
{
      this.name=...
      this.zusatz1=...
      this.zusatz2=...
      ...
}

// Erzeugen einer Instanz
var eineKlinik=new klinik();
```

Das Klinikobjekt enthält zusätzlich zu den Klinikdaten noch weitere Variablen. So wird zum Beispiel die Position auf der Karte gleich in einem Map24.Location-Objekt gespeichert. Des Weiteren enthält das Klinikobjekt eine Methode *printInfos()*, welche die Daten der Klinik an einer bestimmten Position im HTML ausgibt. Der Funktion wird ein DOM-Knoten als Parameter übergeben, unter diesem Knoten werden die Daten als Kind-Knoten eingefügt.

```
<html>
<head>
<script type="text/javascript" language="javascript">
      var eineKlinik=new klinik();
      var target=document.getElementById("infos");
      eineKlinik.printInfos(target);
</script>
</head>
<body>
      <div id="infos"></div>
</body>
</html>
```

3.4 Frontend - Darstellen der Ergebnisse

Sind alle Daten im Array erfasst, können diese nun angezeigt werden. Als Erstes werden die Kliniken in der Karte angezeigt. Dazu werden die Map24.Location-Objekte aller Kliniken, mit Hilfe ihrer Methode *commit()*, auf der Karte veröffentlicht. Die Kliniken sind auf der Karte durch unterschiedlich farbige Punkte dargestellt.

Abhängig von der Versorgungsart der gefundenen Kliniken, wird unter der Karte eine Legende eingeblendet, welche die unterschiedlichen Farben der Kliniken erläutert. Es werden nur die Farben angezeigt, die auch wirklich auf der Karte vorkommen. Die Legende wird in einem vorher definierten, leerem div-Bereich mittels JavaScript dynamisch erzeugt.

Unter dieser Legende wird eine Liste mit allen gefundenen Kliniken, geordnet nach Entfernung, ausgegeben. Dabei steht die Klinik mit der geringsten Entfernung an erster Stelle. Wie bereits bei der Legende werden die Daten in einem vorher definierten, leerem div-Bereich mittels JavaScript dynamisch in das DOM eingefügt. Dies erledigt die im vorangegangen Kapitel erwähnte Funktion *printInfos()*. Die Ergebnisse werden zur besseren Orientierung mit einer wechselnden Hintergrundfarbe dargestellt. [siehe dazu auch: Abbildung 3: Nach der Suche (Auszug)]

Den Kliniken auf der Karte ist ein MouseOver-Event zugeordnet. Fährt der Benutzer mit der Maus über eine Klinik auf der Karte, werden auf der rechten Seite, unter dem Suchformular, die Daten der Klinik angezeigt. In diesem Bereich befindet sich ein weiterer div-Container. Dem MouseOver-Event ist eine JavaScript Funktion zugeordnet, welche die *printInfos()* der aktuellen Klinik mit diesem div-Bereich als Ziel ausführt.

4 Auswertung des Endergebnisses

Alles in allem ist dieses erarbeitete Ergebnis nur ein Zwischenschritt. Durch den modularen Aufbau des Ganzen ist es relativ einfach möglich die Suche zu erweitern bzw. den Aufbau der Seite zu ändern. Es ist also weitestgehend eine Trennung von Code und Darstellung gelungen. Dies ist vor allem dem Einsatz moderner Technologien wie AJAX und Webservices zu verdanken.

Einige Optionen, wie zum Beispiel die Routenplanung, sind noch nicht implementiert, lassen sich aber ohne größere Probleme hinzufügen. Die Map24-API stellt, gerade was Routenplanung betrifft, gute Schnittstellen zur Verfügung.

Größtes Manko der jetzigen Lösung ist wohl das Design. Da die Internetseiten von kliniksuche-sachsen.de aber nur als Konzeptstudie gedacht sind, wurde auf die Optik keinen Wert gelegt. Wie bereits erwähnt, lässt sich das Aussehen und die Anordnung der Elemente, unabhängig von der sich dahinter verbergenden Logik, verändern.

Das Ziel der Arbeit, die Machbarkeit nachzuweisen, ist erreicht. Mit verändertem Design kann nun im nächsten Schritt die Lösung auch in anderen Projekten eingesetzt werden.

Literaturverzeichnis

[1] Google Maps API (22.06.07)
Internet: http://www.google.com/apis/maps/

[2] Mapsolute Developer Network (22.06.07)
Internet: http://devnet.map24.com/

[3] Wikipedia: Ajax (Programmierung) (22.06.07)
Internet: http://de.wikipedia.org/wiki/Ajax_(Programmierung)

[4] Wikipedia: JavaScript (22.06.07)
Internet: http://de.wikipedia.org/wiki/JavaScript

[5] Wikipedia: Document Object Model (22.06.07)
Internet: http://de.wikipedia.org/wiki/Document_Object_Model

[6] Wikipedia: Extensible Markup Language (22.06.07)
Internet: http://de.wikipedia.org/wiki/Extensible_Markup_Language

[7] Galileo Computing: JavaScript und AJAX – 18.2 AJAX Technik
http://www.galileocomputing.de/openbook/javascript_ajax/18_ajax_002.htm

[8] Wikipedia: MySql (22.06.07)
Internet: http://de.wikipedia.org/wiki/MySQL

[9] Wikipedia: LAMP (22.06.07)
Internet: http://de.wikipedia.org/wiki/LAMP

[10] Wikipedia: PHP (22.06.07)
Internet: http://de.wikipedia.org/wiki/PHP

[11] Wikipedia: Web Service (22.06.07)
Internet: http://de.wikipedia.org/wiki/Webservice

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AJAX	**A**synchronous **J**avaScript **a**nd **X**ML
API	**A**pplication **P**rogramming **I**nterface – Programmschnittstelle
DBMS	**D**aten**b**ank **M**anagement **S**ystem
DOM	**D**ocument **O**bject **M**odel - API zu XML und HTML Dokumenten
ERD	**E**ntity-**R**elationship-**D**iagramme – für die Darstellung von Relationen
HTML	**H**yper**t**ext **M**arkup **L**anguage
HTTP	**H**ypertext **T**ransfer **P**rotocol – Protokoll zum Übertragen von Daten
IE	**I**nternet **E**xplorer – Webbrowser von Microsoft
LAMP	**L**inux **A**pache **M**ySql **P**HP
OOP	**o**bjekt**o**rientierte **P**rogrammierung
PHP	**P**HP: **H**ypertext **P**reprocessor (ursprünglich: Personal Home Page Tools)
SQL	**S**tructured **Q**uery **L**anguage – Abfragesprache für Datenbanken
XML	E**x**tensible **M**arkup **L**anguage – Auszeichnungssprache für Daten

Abbildungsverzeichnis

Anhang

Beispiel eines erzeugten XML-Dokumentes mit einer Klinik

```
<daten>
    <ds id="2003321115754745170">
        <name>Klinikum St. Georg gGmbH</name>
        <zusatz1>Akadem. Lehrkrankenhaus</zusatz1>
        <zusatz2>der Universität Leipzig</zusatz2>
        <pos>
            <long>742.82366943359</long>
            <lat>3083.02246093750</lat>
        </pos>
        <dist/>
        <adress>
            <zip>04129</zip>
            <city>Leipzig</city>
            <street>Delitzscher Str.</street>
            <hno>141</hno>
        </adress>
        <contact>
            <vorwahl>0341</vorwahl>
            <tel>909-0</tel>
            <fax>909-2155</fax>
            <web>http://www.sanktgeorg.de</web>
            <email>info@sanktgeorg.de</email>
        </contact>
        <details>
            <betten>1050</betten>
            <tp>85</tp>
        <versorgung>S</versorgung>
            <traeger>O</traeger>
            <skhr_id>69</skhr_id>
        </details>
    </ds>
</daten>
```

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Praxisarbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtliche oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht, noch einer anderen Prüfungsbehörde vorgelegt.

Leipzig, 14.07.07 _____

Christian Gräfe