Hochschule Neubrandenburg
University of Applied Sciences

Hochschule Neubrandenburg

Studiengang Geoinformatik

Konzeption eines Systems für die Aufnahme von Gebäudedaten für das Wegeleitsystem WeLeS

urn:nbn:de:gbv:519-thesis2009-0302-4

Bachelorarbeit

vorgelegt von: *Marcel Klingberg*

Zum Erlangen des akademischen Grades

„Bachelor of Engineering“ (B.Eng.)

Erstprüfer: Prof. Dr.-Ing. Andreas Wehrenpfennig

Zweitprüfer: Dipl.-Informatiker Jörg Schäfer

Eingereicht am: 23.10.2009

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den 23.10.2009

Unterschrift

……

Marcel Klingberg

Kurzfassung

Das WeLeS erleichtert die Orientierung innerhalb eines Gebäudes. Grundlage des Wegeleitsystems bilden Gebäudedaten die in einer Datenbank verwaltet werden. Der derzeitige Prozess der Gebäudedatenerfassung ist sehr aufwendig. Um den Prozess der Datenerfassung zu erleichtern wurde ein Konzept erarbeitet. Das Konzept ermöglicht die webbasierte Datenerfassung im WeLeS. Dafür sieht das Konzept einige Änderungen der Datenbank vor. Die Funktionsweise des Konzeptes wird mit Hilfe eines Prototyps demonstriert.

Stichworte

Wegeleitsystem, AOWIs, WeLeS, PHP, MySQL, SVG

Abstract

The WeLeS makes easier the orientation within a building. Basis of the guide-system are building data. This data are managed in a database. The current process of collecting building data is very wasteful. To facilitate the process of data collection a concept was developed. The concept allows the web-based data collection in the WeLeS. Therefore the concept provides for some changes of the database. The functionality of the concept is demonstrated by using a prototype.

Keywords

guide-system, AOWIs, WeLeS, PHP, MySQL, SVG

.

Inhaltsverzeichnis

1 Einführung

Die Grundlage für die vorliegende Bachelorarbeit bildete die Entwicklung eines **A**uskunfts-, **O**rientierungs- und **W**ege**l**eit**s**ystems (**AOWIs**) im Rahmen zweier Diplomarbeiten [1] [2] für das Dietrich-Bonhoeffer-Klinikum in Neubrandenburg. Wie aus den Diplomarbeiten hervorgeht, handelt es sich beim AOWIs um ein Geoinformationssystem (GIS), genau genommen kann hier von einem Gebäude-Informationssystem (GebIS) gesprochen werden. Ein GebIS ist: *„eine Sammlung von Informationen und Methoden zur Erfassung, Verwaltung, Analyse und Präsentation von Daten mit Gebäudebezug“* [3]. Gemäß dieser Definition enthält das AOWIs Gebäudedaten, um folgenden Aufgaben nachzukommen:

- Informationen eines gesuchten Zieles bereitzustellen (Auskunft)
- die Lage des gesuchten Zieles zu beschreiben (Orientierung)
- eine Karte mit entsprechenden Angaben zur Zielführung bereitzustellen.

Im Rahmen einer Belegarbeit [4] wurde das AOWIs erweitert und auf Open Source Technologie portiert. Das so entwickelte **W**eg**ele**it**s**ystem(**WeLeS**) wurde um einen Suchalgorithmus für eine variable Wegfindung erweitert. Die Daten beider Systeme beziehen sich nur auf das Dietrich-Bonhoeffer-Klinikum in Neubrandenburg. Für die Verwaltungen eines neuen Gebäudekomplexes müssten die Daten für das AOWIs bzw. WeLeS neu gewonnen und das System neu eingerichtet werden. Um diesen Vorgang zu erleichtern, wird im Rahmen der vorliegenden Bachelorarbeit ein Konzept für ein Programmmodul entwickelt, mit dem es möglich ist, Daten auf der Grundlage von Gebäudegrundrissen online im System zu erfassen. Dabei werden Untersuchungen angestellt, ob bereits vorhandene Lösungen existieren.

2 Analyse des Themas und der zugrundeliegenden Wegeleitsysteme

2.1 Was ist zu tun?

Die Hauptaufgabe der beiden Wegeleitsysteme besteht darin, auf der Grundlage von Gebäudekarten, einem Besucher die Orientierung innerhalb eines Gebäudekomplexes zu erleichtern. Der Besucher wählt ein Ziel aus, und als Ergebnis erhält er eine Wegbeschreibung wie er dieses Ziel erreichen kann. Die erforderlichen Daten werden aus einer Datenbank gelesen. Zu diesen Daten zählen die Gebäudegrundrisse, sowie die Zielwege und Wegbeschreibungen. Alle diese Daten müssen im Vorfeld erfasst und in der Datenbank bereitgestellt werden. Das Erfassen der Daten für die Wegeleitsysteme ist ein aufwendiger Arbeitsprozess (Anhang 1). Anregungen, wie die Daten direkt im einen Wegeleitsystem ermittelt werden können, wurden in einer der Diplomarbeiten [1] zum AOWIs gegeben.
Dabei wird vorgeschlagen, die erforderlichen Graphiken aus einer stilisierten CAD Zeichnung zu erstellen. Die so erstellte Graphik soll als Grundlage der Digitalisierung dienen. Über ein Auswahlmenü erhält der Benutzer die Möglichkeit, die zu digitalisierende Punktart auszuwählen. Als zugrundeliegende Technologie zur Bestimmung der Koordinaten und Digitalisierung wurde JavaScript genannt.
Die Idee, auf einer Graphik die Gebäudegrundrisse und Zielwege zu digitalisieren, soll mit dieser Bachelorarbeit realisiert werden. Die Grundlage zur Entwicklung eines Programmmoduls bildet die Entwicklung des WeLeS. Im Rahmen der Analyse werden folgende Fragen geklärt:

- Wie funktionieren beide Wegeleitsysteme?
- Welche Technologien werden eingesetzt?
- Wie können Orientierungspunkte erfasst werden?
- Welche Arten von Orientierungspunkten gibt es?
- Welche Anforderungen können an ein Programmmodul gestellt werden?
- Welche Lösungsansätze ergeben sich daraus?

Anhand der Bearbeitung dieser Fragen kann in einem anschließenden Schritt ein Konzept zu einem Programmmodul erarbeitet werden.

2.2 Grundprinzipien beider Wegeleitsysteme

2.2.1 Grundprinzip des AOWIs

Das AOWIs erteilt Auskünfte über ein gesuchtes Ziel innerhalb eines Gebäudekomplexes. Damit ein Nutzer sein gesuchtes Ziel erreichen kann, werden die entsprechenden Daten aus einer Datenbank ermittelt, und als Ergebnis erhält der Nutzer eine generierte Karte mit allen wichtigen Informationen. Um das zu erreichen, werden die für die graphische Darstellung in SVG (2.3.2) benötigten Daten aus einer Datenbank abgefragt. Änderungen an den Koordinaten in der Datenbank wirken sich automatisch auf die generierten graphischen Elemente und Wegführung aus. Um neue graphische Elemente hinzuzufügen brauchen nur neue Koordinaten in die Datenbank eingefügt werden. Realisiert wird das AOWIs über eine Client-Server Lösung. Der Client Browser startet die Abfragen und stellt die Ergebnisse da. Die erforderlichen Koordinaten für die graphische Darstellung mit SVG wurden über eine Digitalisierung im CAD-System gewonnen. In den Diplomarbeiten [1] und [2] wurden die entsprechenden Schritte angeführt, die erforderlich sind, um ein Gebäudekomplex (im Beispiel der Diplomarbeiten das Klinikum) in der Datenbank zu speichern. Dabei wird deutlich, dass das AOWIs für jeden neuen Gebäudekomplex (z.B. Museum, Universität) neu aufgesetzt und die entsprechenden Schritte der Digitalisierung neu durchgeführt werden müssen. Für die Pflege und Wartung des AOWIs, sind entsprechende Werkzeuge vorgesehen. So ist es möglich, neue Ziele anzulegen, Ziele zu löschen, Ziele umzubenennen, Ziele umzusetzen, und neue Wegpunkte hinzuzufügen.

2.2.2 Grundprinzip des WeLeS

Das in der Belegarbeit auf Open Source entwickelte WeLeS funktioniert auf derselben Grundlage wie das AOWIs. So werden im WeLeS dieselben Grundrissdaten des Klinikums verwendet, die im Rahmen des AOWIs Projektes gewonnen wurden. Das bedeutet, um im WeLeS einen neuen Gebäudekomplex aufzunehmen, müssen dieselben Arbeitsschritte durchgeführt werden wie im AOWIs. Problematisch im WeLeS ist, dass alle Scripte ausschließlich auf das Beispiel des Klinikums zugeschnitten sind. Somit ist das WeLeS nur eine exemplarische Umsetzung der Grundphilosophie des AOWIs auf Basis von Open Source Technologie. Hierbei gilt zu beachten, dass für das WeLeS die ursprüngliche Datenbank des AOWIs umstrukturiert wurde. Werkzeuge für eine Wartung und Pflege konnten im Rahmen der Belegarbeit nicht realisiert werden.

2.3 Eingesetzte Technologien

Bei beiden Wegeleitsystemen handelt es sich um Webanwendungen. Das bedeutet, dass der Nutzer keine weitere Software für den Einsatz der Wegeleitsysteme installieren muss, da diese nur über den Webbrowser aufgerufen wird. Die grundlegende Technologie zur Darstellung von Inhalten in einem Browser ist HTML. HTML steht für **H**yper**T**ext **M**arkup **L**anguage [5] und ist eine Seitenbeschreibungssprache. Mit dieser Sprache ist es möglich, Texte mit Hilfe zusätzlicher Befehle (Tags) zu beschreiben und zu strukturieren. Es ist aber nicht nur möglich Texte darzustellen, sondern auch Multimediainformationen.

HTML hat aber auch Grenzen. So ist es mit reinen HTML nicht möglich, einfache Formulare auf Vollständigkeit zu überprüfen, oder einfach nur 2 Werte zu addieren. Das Werkzeug, dass solche Aufgaben realisiert, heißt JavaScript. JavaScript ist eine Programmiersprache die innerhalb von HTML eingebettet, sowie als externe Datei in HTML eingebunden werden kann. Der Browser ist in der Lage, die Quellzeilen von JavaScript bei entsprechenden Aufrufen zu interpretieren und auszuführen. Mit JavaScript ist es beispielsweise möglich, die Position der Mouse auf einem Bild abzufragen. Dadurch spielt JavaScript auch für den Einsatz im Programmmodul eine bedeutende Rolle.

HTML und JavaScript sind beides Technologien, die auf der Clientseite des Browsers ausgeführt werden. So ist es nicht möglich, mit diesen Technologien dynamisch auf Anfragen des Benutzers zu reagieren und entsprechend eine Website zu generieren bzw. Daten in einer Datenbank zu speichern. Da das AOWIs dynamisch auf die Anfragen von Nutzern reagieren muss, reichen diese Technologien nicht mehr aus. Wie bereits unter Punkt 2 erläutert, basiert das AOWIs auf einer Client-Server Lösung. Das bedeutet, alle Anfragen des Clients werden von einem Server entgegengenommen, verarbeitet und die Ergebnisse in Form eines HTML-Dokuments zurück zum Client gesendet. Die folgende Abbildung soll dies verdeutlichen.

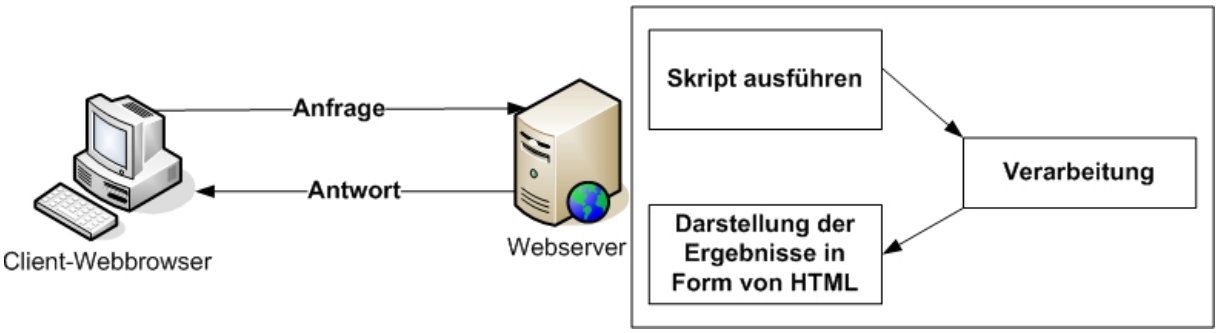


Abb. 1 Client-Server System

Zu den serverseitigen Programmiersprachen zählen unter anderem PHP, ASP, ASP.NET, PERL, Java, Ruby, Python.

2.3.1 Programmierung

Die Programmierung des AOWIs erfolgte aufgrund der Voraussetzungen im Dietrich-Bonhoeffer-Klinikum mit Microsoft ASP. Ziel bei der Entwicklung des WeLeS war es, das AOWIs auf Open Source Technologien zu portieren. Somit kommt im WeLeS für die serverseitige Programmierung PHP [6] zum Einsatz. PHP ist eine weit verbreitete Programmiersprache für die Entwicklung von dynamischen Webseiten. Dabei werden die PHP Funktionalitäten als Modul im laufenden Webserverprozess geladen. Damit das möglich ist, muss der Webserver so konfiguriert werden, dass der PHP Interpreter entsprechende PHP Scripte auswerten kann.

2.3.2 Graphische Darstellung

Die Entwicklung des AOWIs führte dazu, SVG als Graphikformat zur Darstellung der Grundrisse und Wege zu nutzen. SVG steht für **S**calable **V**ector **G**raphics und ist ein XML (**E**xtensible **M**arkup **L**anguage) basierter Standard für die Darstellung von Vektorgraphik im Internet [7]. Standardisiert wurde SVG ab September 2001 durch das W3C (World Wide Web Consortium). Der Einsatz von SVG hat für das AOWIs folgende Gründe [1]:

- einfache Skalierbarkeit ohne „Verpixelung“
- einfache Transformation und Translation
- einfaches Übereinanderlegen von Grafikelementen
- Zusammensetzen von größeren Grafikelementen durch Verwendung bereits vorhandener Elemente
- Gruppierung zusammengehöriger Grafikelemente, die gemeinsam einfach bearbeitet werden können
- Formatierung von Elementen und Gruppen über CSS (**C**ascading **S**tyle **S**heets)
- Einbinden von Rastergraphiken (JPEG, PNG, GIF)
- Entwicklung eigener Schriftarten
- Zugriff auf Elemente von außen über das DOM (**D**ocument **O**bject **M**odel)

Der letzte Punkt dieser Aufzählung ist besonders wichtig für die Entwicklung des Programmmoduls. Der Zugriff auf das DOM eines SVG Dokuments kann beispielsweise mit JavaScript erfolgen. Im WeLeS kommt ebenfalls SVG für die graphische Ausgabe zum Einsatz.

2.3.3 Die Datenbank

Eine Datenbank ist nach [8]: „Eine Sammlung von zusammenhängenden (in Beziehung zueinander stehenden Daten, …)… . Die Datenbank kennzeichnet die zentrale Komponente eines Geoinformationssystems.“ Nach [9] erfolgt der Zugriff auf die Datenbank über das Datenbankmanagementsystem (DBMS). Es hat die Aufgabe: „die einheitliche Beschreibung, die sichere Verwaltung und die schnelle Abfrage der Datenbank“ zu ermöglichen. Weitverbreitet sind relationale Datenbanksysteme (RDBMS) z.B. Oracle, DB2, MySQL, Access. In beiden Wegeleitsystemen kommen daher solche Systeme zum Einsatz. Das RDBMS im AOWIs ist Microsoft Access und im WeLeS MySQL. Die folgenden Unterpunkte beschreiben beide Systeme näher.

2.3.3.1 AOWIs Datenbank

Basis des AOWIs ist eine Microsoft Access Datenbank bestehend aus 6 Tabellen:

- Tabelle ZIELDATEN
- Tabelle GRUNDRISSE
- Tabelle WEGPUNKTE
- Tabelle WEGE
- Tabelle BILDER
- Tabelle TRANSFORMATION

Die Beziehung der Tabellen werden durch folgendes ER- Model dargestellt [1].

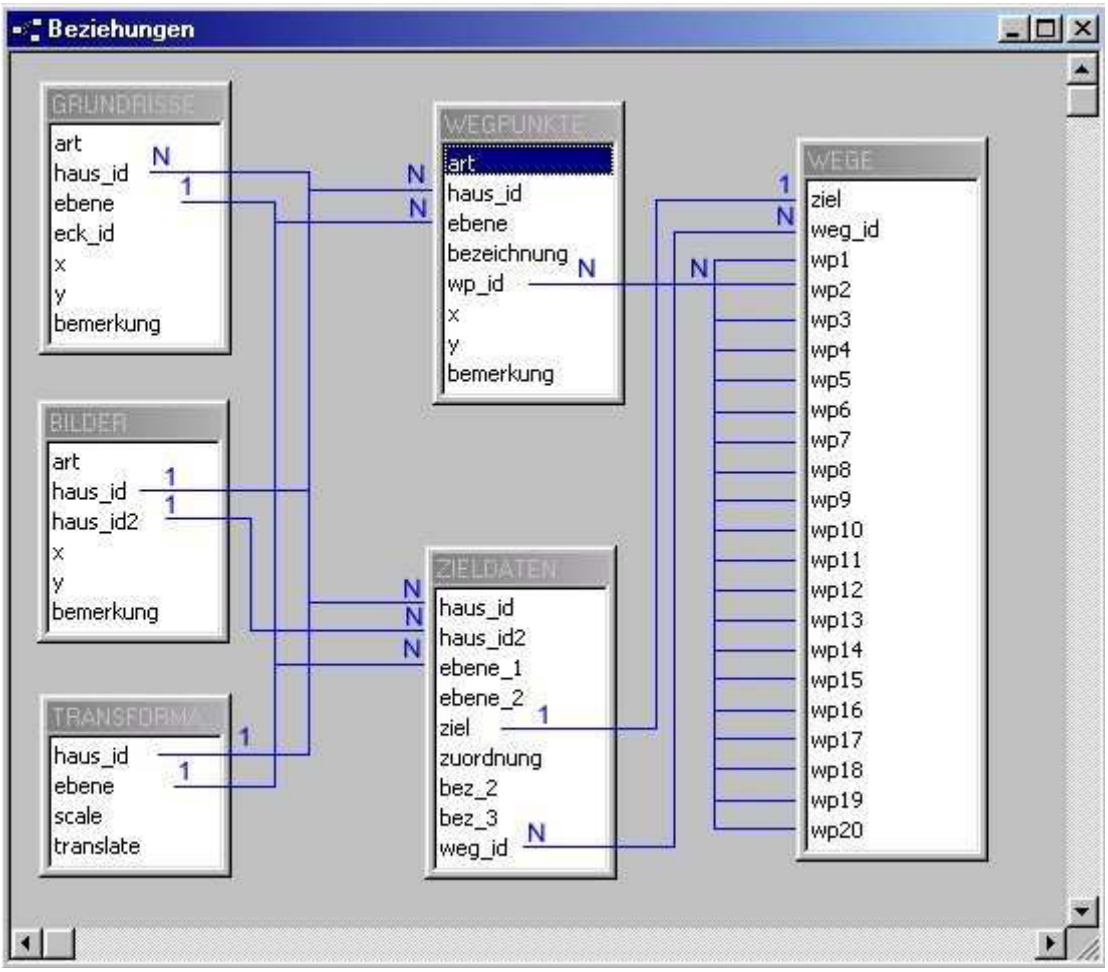


Abb. 2 ER- Model AOWIs

Da das AOWIs für die vorliegende Bachelorarbeit keine weitere Rolle spielt, wird an dieser Stelle nicht weiter auf die Datenbank eingegangen. Weitere Informationen zu dieser Datenbank können aus [1] und [2] entnommen werden.

2.3.3.2 WeLeS Datenbank

Für die Entwicklung des WeLeS wurde die ursprüngliche Datenbank umstrukturiert. Die Tabellen wurden nicht 1 zu 1 in MySQL überführt, sondern es wurden nur die benötigten Daten entnommen. Grundlage des WeLeS bildet eine MySQL Datenbank mit 4 Tabellen:

- Tabelle GRUNDRISS
- Tabelle EBENEN
- Tabelle HAUS
- Tabelle WEGPUNKTE

Die Tabellen WEGE und TRANSFORMATION werden nicht mehr benötigt, da die entsprechenden Informationen in anderen Tabellen integriert werden bzw. die Tabelle WEGE wird im WeLeS durch einen Suchalgorithmus (Anhang 2) ersetzt. Für die SVG Darstellung sind die Tabellen GRUNDRISS, EBENEN, HÄUSER und WEGPUNKTE verantwortlich. Grundlage für den Suchalgorithmus bildet die Tabelle WEGPUNKTE. Die Struktur der Datenbank kann Abb. 3 entnommen werden.

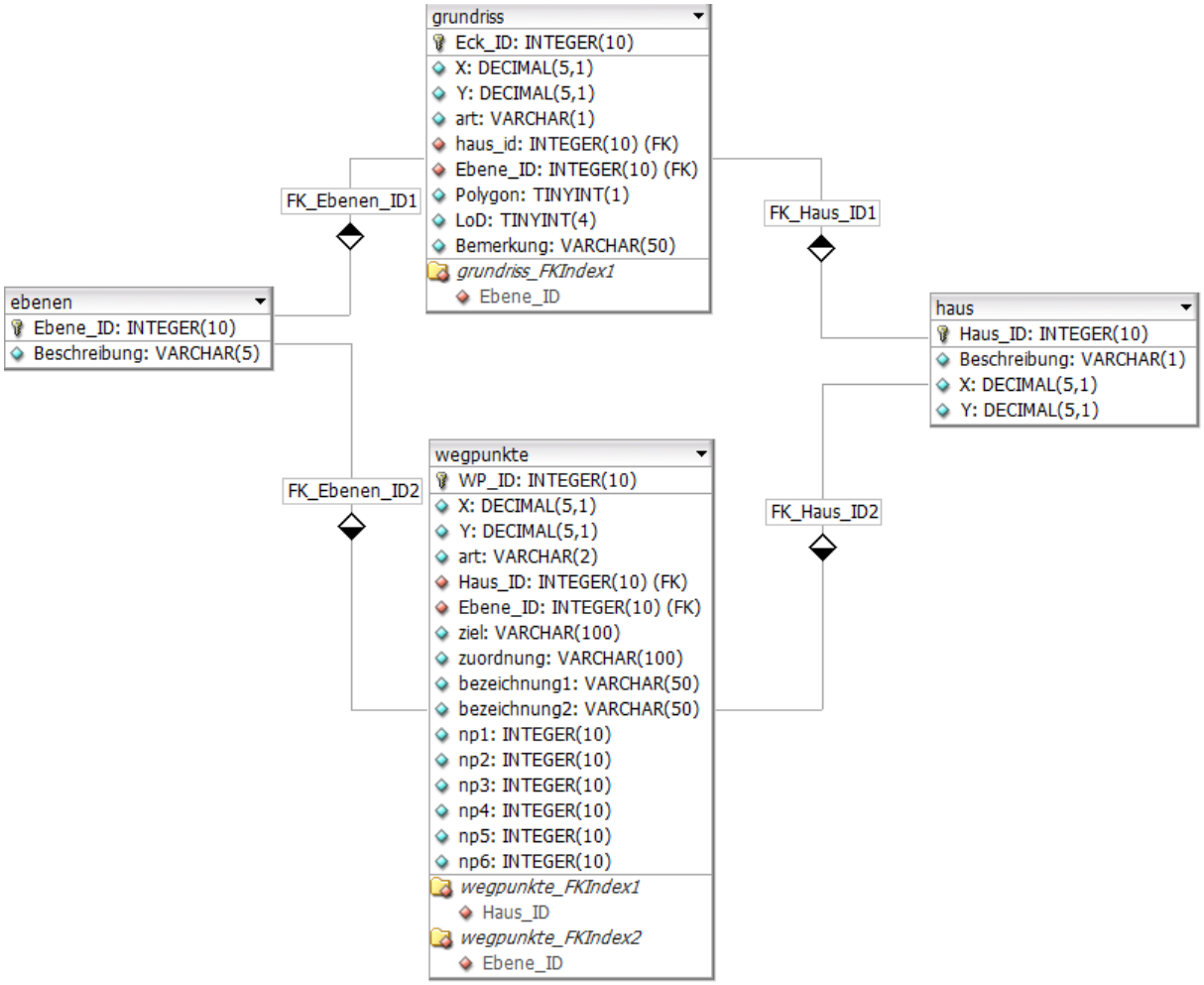


Abb.3 ER- Modell WeLeS

Die Eckpunkte der Häuser werden 2 dimensional (x, y) in der Tabelle GRUNDRISS gespeichert. Über das Attribut *art* können die Punkte weiter unterteilt werden (H steht für Hauspunkt, G für Gangpunkt, I für Innenleben und O für Orientierung). Über die Fremdschlüssel *Ebenen_ID* und *Haus_ID* kann festgestellt werden, in welchem Haus und auf welcher Etage sich ein Punkt befindet. Um Öffnungen wie Türen oder Gänge besser herauszustellen, wurde im WeLeS das Attribut *Polygon* neu eingeführt. Neu ist auch das Attribut *LoD*, das für Level of Detail steht. Mit *LoD* kann bestimmt werden, wann ein Punkt gezeichnet werden soll und wann nicht. Zur Verbesserung der Struktur wurden die Attribute *haus* und *ebenen* aus der ursprünglichen Microsoft Access Datenbank in eigenen Tabellen ausgelagert. Die Tabelle

HAUS enthält neben der *Id* und Beschreibung auch Angaben für die Platzierung der Hausbeschriftung auf der generierten SVG Darstellung. Die Etagen der Gebäude werden in der Tabelle EBENEN verwaltet. Die Tabelle WEGPUNKTE dient zur Speicherung der Wegpunkte für den Suchalgorithmus. Aus der Tabelle ZIELDATEN der ursprünglichen Microsoft Access Datenbank werden im WeLeS die Attribute *ziel*, *zuordnung*, *bezeichnung1* und *bezeichnung2* erstellt. Um Nachbarpunkte, die für den Suchalgorithmus wichtig sind zu verwalten, enthält die Tabelle WEGPUNKTE die Attribute *np1* bis *np6*. Diese Attribute speichern die Id's der benachbarten Punkte.

2.3.4 Verbesserung der Datenbank

Aufgrund der Analyse der Datenbank ergeben sich einige Änderungsvorschläge für eine Weiterentwicklung des WeLeS.
In der Datenbank werden bisher für die Verwaltung der Punkte zwei Attribute *x* und *y* gebraucht. Über das Attribut *LoD* wird entschieden, wann ein Punkt gezeichnet werden soll. Da aus den gespeicherten Punkten in SVG Polygone oder Polylines erzeugt werden, bietet sich eine Verwaltung der Punkte unter Verwendung von Geometriedatentypen an. Dazu zählen unter MySQL beispielsweise `Point`, `LineString`, `Polygon` u. a. Geometriedatentypen. So ist es möglich die komplette Darstellung des Grundrisses in Polygonen zu speichern. Diese würde den Aufwand der Programmierung für die Darstellung vereinfachen.
Die Analyse ergab außerdem, dass alle vorhandenen Tabellen vom MySQL Tabellentyp MySAM *(*Anhang 3*),* sind. Diese hat Auswirkung auf die Verwendung der vorhandenen Fremdschlüssel (Foreigen-Key) der Tabellen, da MySAM Tabellen keine Integritätsregeln (Foreigen-Key-Regeln) unterstützen [6]. Die Integritätsregeln prüfen bei Veränderungen von Datensätzen, wie sich diese Änderungen auf Tabellen auswirken, die über einen Fremdschlüssel verwiesen werden. Da bei Änderungen der MySAM Tabellen keine Kontrollen von MySQL vorgenommen werden, kann diese leicht zur Inkonsistenz der Datenbank führen. Ein Beispiel dafür findet sich in Tabelle EBENEN. Über die Tabelle EBENEN sollten eigentlich die Etagen der Gebäude verwaltet werden. Leider ist die Tabelle für die Beispieldaten aus dem Klinikum leer. Trotzdem befinden sich in den Tabellen GRUNDRISS und WEGPUNKTE in den Fremdschlüsseln aus der Tabelle EBENEN Werte. Würden die Integritätsregeln überprüft werden, wären keine Abfragen möglich, da die Fremdschlüssel auf eine leere Tabelle verweisen. Dieses bedeutet auch, dass bei der Programmierung keine entsprechenden Kontrollen durchgeführt werden. Bei der Programmierung können zwar Kontrollen vorgenommen werden, wenn von der Datenbank aus keine Integritätsregeln überprüft werden, aber auch hier können sich Fehler einschleichen. Der einzige Tabellentyp, der Integritätsregeln in MySQL prüft, ist InnoDB. Daher wäre es von Vorteil, aus dem Gesichtspunkt der Integritätskontrollen alle Tabellen auf InnoDB umzustellen. Ein weiterer Vorteil von InnoDB ist die Un-

terstützung von Transaktionen. Transaktionen haben die Aufgabe, bei einem Mehrbenutzerzugriff die Inkonsistenz der Daten zu verhindern. Erst wenn eine Transaktion bestätigt wird, werden die Änderungen am Datenbestand wirksam, ansonsten kann die Transaktion widerrufen werden. Transaktionen sind ein weiterer Kontrollmechanismus, um einen inkonsistenten Datenbestand zu verhindern.
Der Einsatz von InnoDB bringt aber auch einige Nachteile mit sich. So unterstützt InnoDB keine Volltextsuche. Des Weiteren verlangsamt der Einsatz von InnoDB den Einsatz des Systems, da bei allen einfüge und lösch Operationen die Integritätsregeln überprüft werden. Bei einem geringen Datenbestand spielt diese aber noch keine so große Rolle. Ein weiterer Nachteil ist, dass ein Online Backup kostenpflichtig ist, ansonsten ist ein Backup nur möglich, wenn die Datenbank heruntergefahren wird.
Eine weitere Veränderung der Datenbank betrifft die Tabelle WEGPUNKTE. Bisher konnten dort nur maximal 6 Nachbarpunkte zugeordnet werden. Um den Datenbankentwurf zu verbessern, könnten diese Attribute in einer extra Tabelle verwaltet werden. Die Tabelle WEGPUNKTE würde dann jeden Punkt wie bisher verwalten, aber die Zuordnung der Nachbarpunkte erfolgt dann in der extra Tabelle. Diese Tabelle würde dann auch die Grundlage für den Suchalgorithmus bilden.
Für die Verwaltung mehrerer Gebäudekomplexe müsste die Datenbank ebenfalls um eine Tabelle erweitet werden. Die Beziehung zwischen Gebäudekomplex und Haus würde dann über eine 1:N Beziehung realisiert werden.
Für die Verwaltung von Grundrissen könnte die Datenbank auch um eine zusätzliche Tabelle erweitert werden.

Im Rahmen der Bachelorarbeit können nicht alle Änderungsvorschläge für eine mögliche Weiterentwicklung untersucht und umgesetzt werden. Es werden daher nur Änderungen vorgenommen, die zum Bearbeiten der Aufgabenstellung notwendig sind.

2.4 Erfassung von Orientierungspunkten

Die Grundlage zur Erfassung von Orientierungspunkten bildet eine Graphik, die den Grundriss abbildet. Das bedeutet, dass der Grundriss in Form einer Rastergraphik vorliegt. Das Grundelement einer Rastergraphik ist das Pixel (*engl. picture element*) [10]. Das Pixel ist nicht nur ein einzelner Bildpunkt, sondern eine gleich große Rasterzelle. Die Pixel der Rastergraphik sind in Matrixform (x, y) angeordnet, wobei der Ursprung i. d. R. die linke obere Ecke der Graphik ist. Aufgrund dieser Struktur kann bei einer Rastergraphik nicht nach Punkt, Linie und Fläche unterschieden werden. Aus der Rastergraphik können nur Eigenschaften der Pixel gewonnen werden, dazu zählen die Grau-. und Farbwerte sowie die Intensität. Diese Eigenschaften der Pixel spielen für die Erfassung von Orientierungspunkten keine Rolle. Die erforderlichen Orientierungspunkte müssen über eine manuelle Erfassung auf der Rastergraphik gewonnen werden. Die so ermittelten Koordinaten beziehen sich in erster Linie auf das Bildkoordinatensystem. Sofern erforderlich, können die ermittelten Bildkoordinaten in jedes gewünschte Koordinatensystem über entsprechende Transformationen überführt werden. Außerdem ermöglicht SVG die Definition beliebiger Koordinatensysteme. Das Ziel der manuellen Erfassung von Punkten besteht darin, Daten für die Generierung einer vektorbasierten Ausgabegraphik zu gewinnen. Als Ausgabeformat kommt SVG (2.3.2) zum Einsatz. In den Diplomarbeiten [1] und [2] werden die Schritte beschrieben, wie die Koordinaten der Punkte für das AOWIs ermittelt wurden. Grundlage hierfür bildet die Digitalisierung der Grundrisse mit Hilfe eines CAD-Systems. Die beschriebenen Schritte zeigen, wie aufwendig der Prozess der Punktaufnahme für einen Gebäudekomplex ist. Der Prozess der Digitalisierung soll zukünftig nur noch online im WeLeS erfolgen. Hierbei werden keine Vorgaben gemacht, ob die Graphik mit Hilfe eines CAD-System erstellt wird oder nicht.

Die Orientierungspunkte auf der Rastergraphik ergeben sich aus der Definition der Datenbank (2.3.3). So enthält die Tabelle GRUNDRISS das Attribut *art*, wo zwischen Hauspunkt (H), Gangpunkt (G), Innenleben (I) und Orientierung (O) unterschieden wird.

Für eine fehlerfreie Darstellung des Gebäudes ist, wie oben erwähnt, auf die Reihenfolge der aufzunehmenden Punkte zu achten.

Auch für die Definition von Zielwegen innerhalb eines Grundrisses sind in der Datenbank verschiedene Arten von Punkten vorgesehen. Diese Wege werden in der Tabelle WEGPUNKTE verwaltet, die ebenfalls das Attribut *art* besitzt. Unterschieden werden die Wegpunkte nach Aufzug (AU), Hauseingang (HE), Hauptkreuzung (HK), Nebenkreuzung (NK), Startpunkt (SP), Treppenhaus (TR), Zielpunkt (ZP), und Zwischenpunkt (ZW).

Der Arbeitsbereich des Programmmoduls muss die Verarbeitung der Rastergraphik für die Erfassung von Orientierungspunkten unterstützen. Die Erfassung der Punkte und die Weiterverarbeitung der gewonnenen Daten bilden den zentralen Kern des Programmmoduls.

2.4.1 Probleme bei der Punkterfassung

Wie oben erwähnt, ergeben sich die Punktarten aus der Definition der Datenbank. Die Unterscheidung der Punktarten kann für einen einfachen Sachbearbeiter recht schwer werden. Die richtige Darstellung der Grundrisse und Zielwege ist zum einen abhängig von der Reihenfolge der aufgenommenen Punkte und zum anderen von der Art der Punkte. Zudem müssen Angaben zu dem jeweiligen Haus und Ebene gemacht werden. Erschwerend kommt noch hinzu, dass bei der Aufnahme der Grundrisspunkte (H, G, I, O) Angaben für die Attribute *Polygon* und *LoD* gemacht werden müssen. Mögliche Angaben für Polygon sind 0, 1, 2, um Öffnungen wie Türen und Gänge bei der Darstellung von Wänden besser herauszustellen. Beim Attribut *LoD* sind ebenfalls Angaben von 0, 1, 2 erforderlich. Wobei bei einem Wert von 2 dafür gesorgt wird, dass ein Punkt nicht gezeichnet wird. Diese Angaben erschweren eine manuelle Datenerfassung der Grundrisspunkte. Hinzu kommt, dass bei Ausführung der PHP Scripte für die Darstellung der Umrisse der Häuser (H), die Umrisse nur mit Hilfe der *ebenen_id* 3 dargestellt werden. Das bedeutet, dass für die Darstellung jeder Ebene der Umriss der Ebene 3 gezeichnet wird. Dies dürfte eigentlich gar nicht möglich sein, da *ebenen_id* ein Fremdschlüssel ist und die Tabelle EBENEN leer ist (2.3.3.3). Für einen neuen Gebäudekomplex heißt das, entweder wird der Umriss nur einmal aufgenommen und enthält das Attribut *ebenen_id 3* oder es wird der Quelltext und die Datenbank dahingehend angepasst, dass zu jeder Ebene der Umriss gezeichnet wird. Wobei zu überlegen ist, wie die graphische Ausgabe eines Zielweges über mehrere Etagen erfolgt.
Die beste Möglichkeit zur Darstellung der Grundrisse bildet das Speichern der Daten mit Hilfe der Geometriedatentypen von MySQL (2.3.3.3). Hierbei braucht der Nutzer sich keine Gedanken über die Art, und die Bedingungen zur Darstellung der Punkte machen.
Weitere Probleme gibt es bei der Bedeutung von zwei Wegpunktarten: Hauptkreuzung HK und Nebenkreuzung NK. Es ist nirgends dokumentiert, was für eine Rolle beide Punktarten für den Algorithmus spielen. Hierbei muss außerdem erwähnt werden, dass keine Gewichtung der Punkte bei der Bestimmung des Zielweges vorgenommen wird. So kann keine Aussage getroffen werden, ob ein generierter Zielweg auch wirklich für jeden zugänglich ist oder nicht. Die hier beschriebenen Probleme bei einer Punktaufnahme zeigen, dass ein Programmmodul zur Aufnahme von Orientierungspunkten nur dann effektiv funktioniert, wenn das Konzept der Datenbank überarbeitet wird. Nachfolgend werden weitere Anforderungen aufgeführt, die an das Programmmodul gestellt werden.

2.5 Anforderung an das Programmmodul

Ziel der Bachelorarbeit ist die Beschreibung eines Programmmoduls, mit dem es möglich ist, die wesentlichen Orientierungspunkte eines Gebäudes auf einer Graphik zu erfassen und im WeLeS zu verwalten. Damit dies möglich ist, muss ein solches Programmmodul einige Anforderungen erfüllen. Die Anforderungen ergeben sich zum einen aus dem Einsatz des Programmmoduls und zum anderen aus dem Aufbau, der Struktur und den im WeLeS eingesetzten Technologien.

2.5.1 Einsatz des Programmmoduls

Der Einsatz des Programmmoduls sollte im Verwaltungsbereich liegen. Der Aufbau und die Bedienung des Moduls sollten so gestaltet sein, dass jeder Sachbearbeiter es mit einfachen EDV Kenntnissen bedienen kann. Mit dem Modul haben die Sachbearbeiter die Möglichkeit, die Grundrisse zu bearbeiten und die erfassten Punkte im Programmsystem einzupflegen. Folgendes Use Case Diagramm soll einen Überblick über die zur Verfügung stehenden Funktionen des Programmmoduls für die Sachbearbeiter geben.

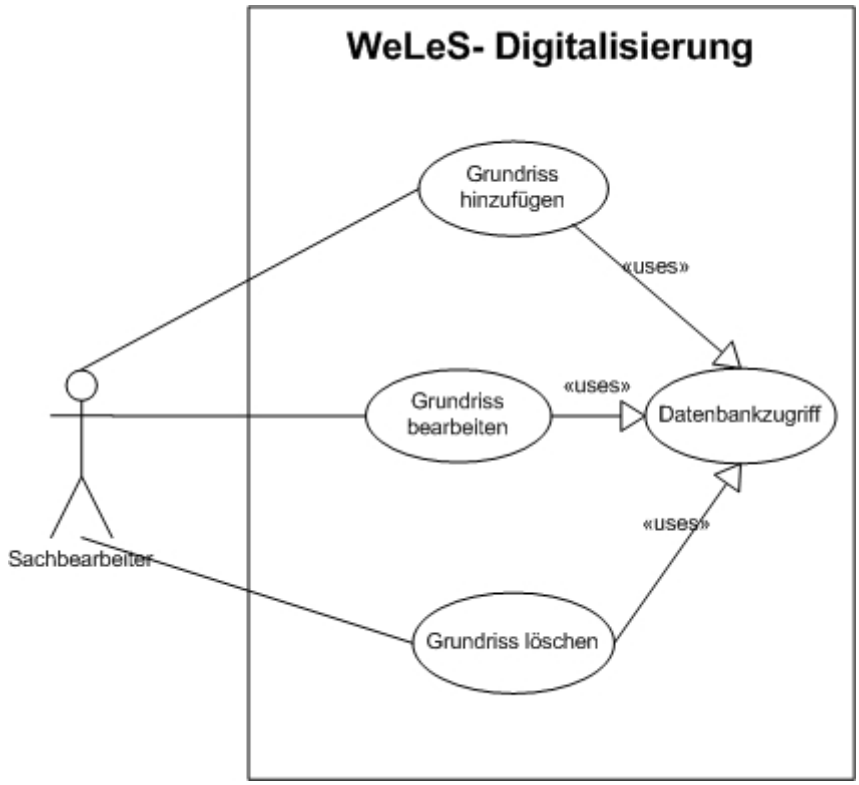


Abb. 4 Use Case Diagramm zum Programmmodul

Das in Abb. 4 dargestellte Use Case Diagramm beschreibt nur die Sicht eines Sachbearbeiters auf das zu entwickelnde Programmmodul. Bei all diesen Arbeiten braucht der Sachbe-

arbeiter Zugriff auf die Datenbank. Die Bedienung des Moduls sollte klar und verständlich sein. Sobald ein Grundriss nicht mehr zur Bearbeitung benötigt wird, sollte dieser entfernt werden können.
An den zugrundeliegenden Grundrissdaten ergeben sich ebenfalls Anforderungen bezüglich der Auflösung und des Maßstabes. Alle Grundrissdaten eines Gebäudekomplexes sollten mit derselben Auflösung und im selben Maßstab vorliegen. Auf diese Weise ist sichergestellt, dass alle Grundrisse und der dazugehörige Zielweg eines Gebäudekomplexes gleich dargestellt werden können. Aufgrund desselben Maßstabes der Grundrissdaten können bei der Zielführung auch Angaben zur Entfernung in Meter gemacht werden.
Da es sich bei Grundrissdaten von Gebäuden einer Firma oder Institution um sensible Informationen handelt, sollten Sicherheitsvorkehrungen zum Schutz der Daten getroffen werden. Das Hochladen der Dateien und der Zugriff könnten beispielsweise über eine Benutzerverwaltung geregelt werden.

2.5.2 Anforderungen zur technischen Umsetzung

Zu den wichtigsten Anforderungen zur technischen Umsetzung des Programmmoduls zählt die Unterstützung der im WeLeS eingesetzten Technologien (2.3).
Grundlage der Entwicklung des WeLeS bildet die Programmierung in PHP. Für ein Programmmodul zur Erfassung von Orientierungspunkten ist es daher erforderlich, dass es die Programmierung in PHP unterstützt. Die Verwaltung der Daten im WeLeS erfolgt wie unter 2.3.3 erläutert in einer MySQL Datenbank. Somit muss das Programmmodul den Zugriff auf eine MySQL Datenbank unterstützen. Die Datenbank muss so aufgebaut sein, dass eine Inkonsistenz der Daten verhindert wird (2.3.4).

Um Grundrisse zu bearbeiten, muss das Programmmodul einen übersichtlichen Arbeitsbereich definieren, in dem es möglich ist, Orientierungspunkte zu erfassen und in der Datenbank einzupflegen. Der Nutzer muss dabei die Reihenfolge der aufzunehmenden Orientierungspunkte berücksichtigen, da dies Auswirkung auf die graphische Darstellung bei der Zielführung in SVG hat.
Bei einem bereits existierenden Programm zur Erfassung von Punkten in Karten oder Grundrissen, kann nicht davon ausgegangen werden, dass alle erwähnten Anforderungen unterstützt werden. Es gilt zu berücksichtigen, dass das AOWIs sowie das WeLeS Eigenentwicklungen sind. Deshalb wird bei einer evtl. existierenden Lösung die Anpassungsfähigkeit des Programms an das WeLeS untersucht. Bei der Anpassungsfähigkeit muss untersucht werden, ob Änderungen am Quelltext möglich sind. Wenn Änderungen möglich sind, wird untersucht, inwieweit der Quelltext angepasst werden muss, um Orientierungspunkte zu erfassen und in die WeLeS Datenbank einzupflegen. Wichtigste Voraussetzung für eine existierende

Lösung ist die Unterstützung von PHP und MySQL. Die folgende Übersicht soll die Anforderungen an einem Programmmodul zur Erfassung von Orientierungspunkten zusammenfassen:

- Einsatz im Verwaltungsbereich
- einfacher Aufbau und Bedienbarkeit
- Grundrissdaten eines Komplexes mit derselben Auflösung und selbem Maßstab
- Unterstützung der im WeLeS verwendeten Technologien
    - HTML, PHP, MySQL, SVG
- Konsistenz der Daten gewährleisten
- Arbeitsbereich für die Erfassung von Orientierungspunkten
- Reihenfolge bei der Punktaufnahme

Bei einer existierender Lösung:

- Unterstützung der im WeLeS verwendeten Technologien
- Anpassung des Programms an das WeLeS

2.6 Lösungsansätze

Aus den Anforderungen können zwei verschiedene Lösungsansätze abgeleitet werden. Auf der einen Seite besteht die Möglichkeit einer eigenen Entwicklung des Programmmoduls, auf der anderen Seite könnte auf ein vorhandenes Programm zurückgegriffen werden. Beide Lösungsansätze werden im Folgenden untersucht.

2.6.1 Eigene Entwicklung

Eine eigene Entwicklung bezieht den gesamten Prozess von der Planung bis zur Realisierung des Programmmoduls mit ein. Die eigene Entwicklung des Programmmoduls kann daher perfekt in das WeLeS integriert werden. So kann bei der Entwicklung auf die im WeLeS eingesetzten Technologien zurückgegriffen und die Struktur der Datenbank den Anforderungen des Moduls angepasst werden. Die Entwicklung einer eigenen Lösung ist in der Regel aufwendiger als auf eine vorhandene Lösung, so fern vorhanden, zurückzugreifen. Im Rahmen der Bachelorarbeit kann daher nur ein Konzept und ein Prototyp entwickelt werden.

2.6.2 Existierende Programmlösung

Als Ergebnis der Recherche zur einer existierenden Lösung kann Mapbender [11] [12] erwähnt werden. Mapbender erfüllt die unter 2.6.2 genannten Voraussetzungen für vorhandene Programmlösungen. Entwickelt wurde Mapbender in PHP und JavaScript. Mapbender ist ein Open Source Projekt der Open Source Geospatial Foundation (OSGeo). Daher ist es möglich, Änderungen am Quelltext von Mapbender vorzunehmen. Die benötigten Daten von Mapbender werden in einer Datenbank verwaltet, dabei kann MySQL oder PostgreSQL/PostGIS zum Einsatz kommen. Diese Daten werden zur Laufzeit aus der Datenbank dynamisch gelesen, daher erinnert Mapbender stark an ein CMS-System.
Die Hauptaufgabe von Mapbender ist die Darstellung und Verwaltung von Karten, die über weltweite Kartendienste zur Verfügung gestellt werden. Der Zugriff auf diese bereitgestellten Karten erfolgt über OGC standardisierter Dienste wie z.B. WMS, WFS.

Die folgende Abbildung veranschaulicht das Hinzufügen von Kartendiensten in Mapbender.

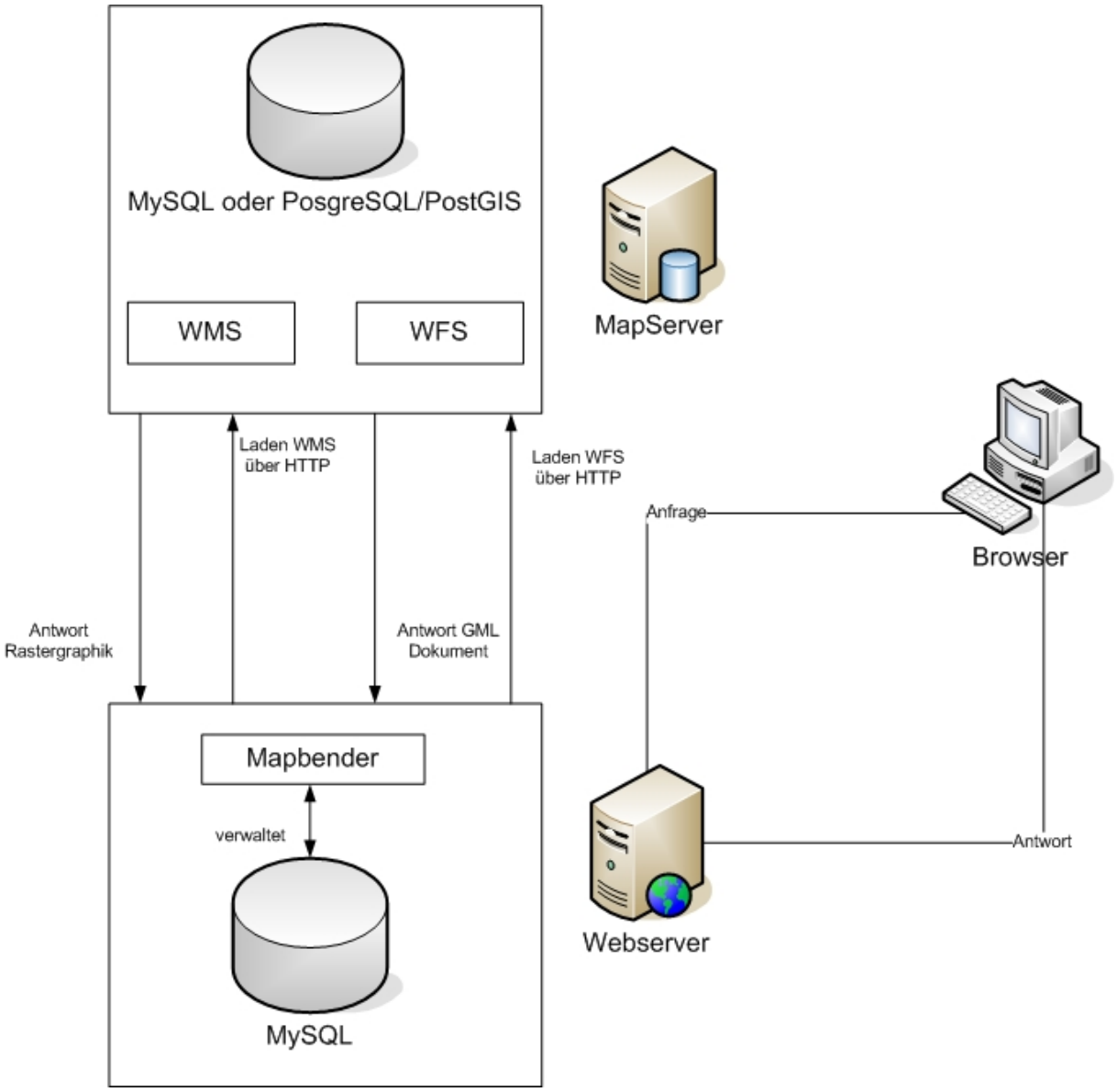


Abb. 5 Einsatz von WFS und WMS in Mapbender

Diese Abbildung resultiert aus der Beschreibung der einzelnen Kartendienste in der offiziellen Dokumentation von Mapbender [12]. Die folgenden Unterpunkte beschreiben kurz die Kartendienste WMS und WFS.

2.6.2.1 WMS

WMS [13] steht für ***W**eb **M**ap **S**ervice* und beschreibt eine Schnittstelle zum Austausch von Karten und Daten über das HTTP Protokoll. Das Hinzufügen eines neuen Kartendienstes erfolgt über die Anfrage *getCapabilities-URL*. Das Ergebnis dieser Anfrage ist ein XML Dokument von einem MapServer, der diesen Dienst anbietet. Dieses XML Dokument enthält alle benötigten Informationen, die Spezifikationen des Servers beschreiben. Diese Informationen werden in der zugrundeliegenden Datenbank von Mapbender gespeichert, und der Dienst ist in Mapbender verfügbar. Der Aufruf von Karten erfolgt über *getMap*, als Ergebnis wird eine Karte in Form einer Rastergraphik geliefert. Der Zugriff auf den geladenen Dienst kann über eine, in Mapbender integrierte, Benutzerverwaltung geregelt werden.

2.6.2.2 WFS

WFS [14] steht für ***W**eb **F**eature **S**ervice* und beschreibt eine Schnittstelle die es ermöglicht, Änderungen an den Features einer Karte vorzunehmen. Ergebnis einer Anfrage ist ein GML Dokument. Mapbender nutzt WFS für folgende Aufgaben [12]:

- Suchmodul
- Digitalisierung
- Räumliche Suche
- WFS Informationen über Tooltip an den Geometrien

Für die Nutzung einer Digitalisieroberfläche wird WFS mit Transaktionsfähigkeit (WFS-T) benötigt. Mapbender enthält für die Digitalisierung eine Beispielanwendung Namens *gui_digitize*. Folgende Abbildung demonstriert das Digitalisieren mit *gui_digitize* am Beispiel des Kummerower Sees.

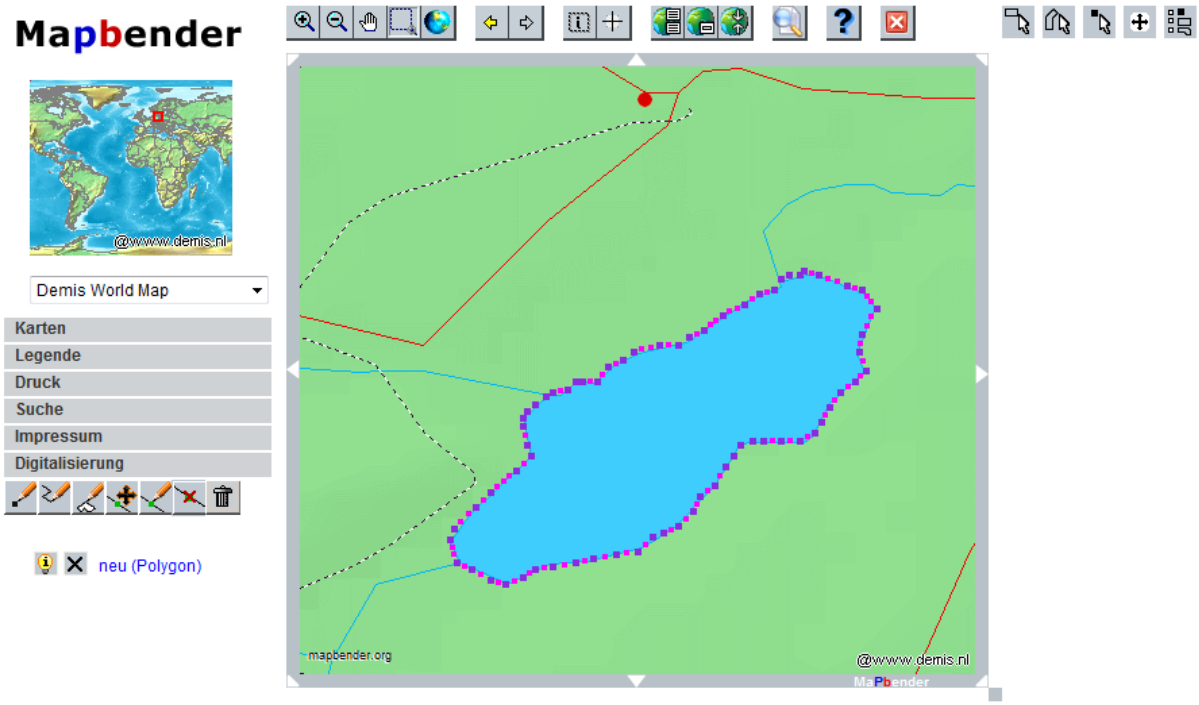


Abb. 6 Beispieldigitalisierung mit Mapbender

Die Beispielanwendung *gui_digitize* bietet die Möglichkeit Punkte, Linien, Polygone und Stützpunkte auf einer Karte zu digitalisieren (siehe unten). In Abb. 6 wurde der See mit Hilfe eines Polygons digitalisiert.

2.6.2.3 Arbeitsbereich von Mapbender

Der Arbeitsbereich beschreibt die Schnittstelle zwischen System und Benutzer. In Mapbender werden dem Benutzer einige Werkzeuge zur Verfügung gestellt.

Abb. 7 obere Werkzeugleiste

Die obere Werkzeugleiste beinhaltet z.B. Zoom und Pan Funktionen sowie die Möglichkeit Koordinaten anzeigen zu lassen. Der Benutzer hat auch die Möglichkeit, Einstellungen an den WMS Daten vorzunehmen und neue WMS Daten hinzuzufügen. Diese Werkzeugleiste ist in dieser oder in einer etwas veränderten Form immer Bestandteil der Darstellung des ausgewählten Kartendienstes.

Zur Digitalisierung stellt Mapbender dem Benutzer eine weitere Werkzeugleiste zu Verfügung.

Abb.8 Digitalisierungswerkzeuge

Für die Digitalisierung bietet Mapbender eine Auswahl zwischen Punkt, Linie, Polygon und Stützpunkten an.

Mapbender bringt zur Bearbeitung von Karten sehr interessante Werkzeuge mit. All diese Werkzeuge könnten bei einem möglichen Einsatz von Mapbender mit genutzt werden. Zudem kann gesagt werden, dass Mapbender die im WeLeS eingesetzten Technologien unterstützt und ein Werkzeug zur Digitalisierung von Punkten bereitstellt.

2.7 Fazit

Wie schon erwähnt, bringt eine eigene Entwicklung einen hohen Arbeitsaufwand mit sich. Bestimmte Werkzeuge wie Zoomen, Pan und Digitalisierung sind in Mapbender vorhanden und müssen bei einer eigenen Entwicklung berücksichtigt werden. Vorteil der eigenen Entwicklung ist die perfekte Integration des Moduls im WeLeS. Mapbender ist zwar ein interessantes Programm zur online Verwaltung und Bearbeitung von Karten, aber von einer Integration im WeLeS sollte abgeraten werden. Dies hat folgende Gründe.

Um Mapbender nutzen zu können, ist der Einsatz eines MapServers erforderlich (Abb. 5). Dieser Server muss die Grundrisskarten in Form eines WFS-T Dienstes zur Verfügung stellen, da Mapbender diesen Dienst zur Digitalisierung benötigt. Der Einsatz von WFS funktioniert nur mit Vektordaten, da WFS eine Anfrage mit einem GML Dokument beantwortet. Rasterdaten lassen sich nicht mit diesem Format verarbeiten. Die Grundlage zur Erfassung von Orientierungspunkten bildet aber eine Rastergraphik. Daher müssten alle Grundrisse so umgewandelt werden, so dass sie über den WFS-T Dienst in Mapbender geladen werden können. Bei der Bearbeitung der Grundrissdaten müssen die von der OGC bestimmten Spezifikationen des WFS berücksichtigt werden. Dieser Prozess ist wahrscheinlich aufwendiger, als das Bestimmen der Punkte für das AOWIs.

Wenn für das Digitalisierungsmodul extra ein MapServer betrieben wird, ist die Frage berechtigt, warum nicht das gesamte System über einen MapServer betrieben wird. Ein MapServer allein reicht aber bei weitem noch nicht aus. Mapbender müsste so angepasst werden, dass die verschieden Punktarten digitalisiert und gespeichert werden können. Die Änderung am Quelltext ist zwar nur einmal erforderlich, doch ist auch dieser Prozess sehr aufwendig.

Der Einsatz eines MapServers beeinträchtigt auch die Forderung nach einer einfachen Benutzung des WeLeS, einschließlich des Digitalisierungsmoduls. So ist entsprechendes Fachwissen der Sachbearbeiter für die Installation, Konfiguration und Administration des MapServers, sowie für Installation und Bedienung von Mapbender erforderlich.

Als Fazit der Analyse kann der Einsatz einer vorhandenen Programmlösung ausgeschlossen werden. Der Aufwand, diese zu integrieren ist weitaus größer als ein eigenes Modul zu entwickeln. Das eigene Modul kann perfekt an das bestehende System angepasst und genutzt werden. Die Anforderungen zur Entwicklung des Moduls wurden im Rahmen der Analyse besprochen. Basierend darauf kann ein Konzept des Programmmoduls erarbeitet werden.

3 Konzept des Programmmoduls

In der Analyse wurde deutlich, dass es schwer wird ein Programmmodul für das bisherige WeLeS zu entwickeln. Die Aufnahme der Orientierungspunkte für die aktuelle Datenbank ist äußerst unübersichtlich. Aus diesem Gesichtspunkt heraus, werden bei der Erarbeitung des Konzeptes einige Veränderungen im WeLeS mit aufgenommen. Die Verwaltung mehrerer Gebäudekomplexe wird bei der Erarbeitung des Konzepts nicht berücksichtigt, da das Konzept sich in erster Linie auf alle wichtigen Änderungen bezieht, die für die Aufnahme von Orientierungspunkten notwendig sind. Die Erarbeitung des Konzeptes dient als Grundlage für die Entwicklung eines Programmmoduls. Dabei werden die in der Analyse genannten Anforderungen berücksichtigt (2.5). Die Erarbeitung des Konzeptes beinhaltet 3 Schwerpunkte:

1. Verwaltung der Grundrissdaten
2. Aufbau des Arbeitsbereichs
3. Speicherung der aufgenommenen Punkte

Die Aufgabe des Programmmoduls besteht darin, Daten in die Datenbank aufzunehmen. Um die Daten auf einem aktuellen Stand zu halten, sollte es eine Möglichkeit geben, Daten löschen zu können. Das Hinzufügen und Löschen von Daten ist nicht ungefährlich, daher sollte nicht jeder Sachbearbeiter diese Rechte erhalten. Um den Verlust von Daten zu verhindern, wäre es wichtig Backup Strategien zu Implementieren. Die Benutzerverwaltung und die Realisierung von Backup Strategien spielen für die Erarbeitung des Konzeptes im Rahmen dieser Bachelorarbeit keine Rolle. Dennoch sei an dieser Stelle darauf hingewiesen, dass diese Punkte für einen erfolgreichen Einsatz des WeLeS nicht vernachlässigt werden dürfen.

3.1 Verwaltung der Grundrissdaten

Die Grundrissdaten liegen in Form einer Rastergraphik vor, die die Grundlage zur Erfassung der Orientierungspunkte bildete (2.4). Die Bearbeitung der Graphiken erfolgt online im WeLeS, somit muss das WeLeS Zugriff auf diese Graphiken erhalten. Prinzipiell ist es möglich, Rastergraphiken in einer Datenbank zu verwalten. Dazu stellt MySQL den Datentyp BLOB für die Speicherungen von Binärdaten zur Verfügung. Da Rastergraphiken recht groß werden können und dies sich negativ auf die Performance der Datenbank auswirken kann, sollte davon abgesehen werden, die Grundrissgraphiken in einer Datenbank zu speichern. Besser ist es, die Graphiken in einem separaten Verzeichnis auf dem Webserver zu verwalten und nur die Dateinamen und Pfadangaben in der Datenbank zu speichern. Die Graphikdateien

können dann im Arbeitsbereich zur Weiterverarbeitung eingebunden werden. Da ein Gebäudekomplex aus mehreren Grundrissen besteht, muss eine Schnittstelle definiert werden, die die Verwaltung mehrerer Grundrissdateien ein und desselben Komplexes beschreibt. Diese Schnittstelle sollte folgende Struktur berücksichtigen.

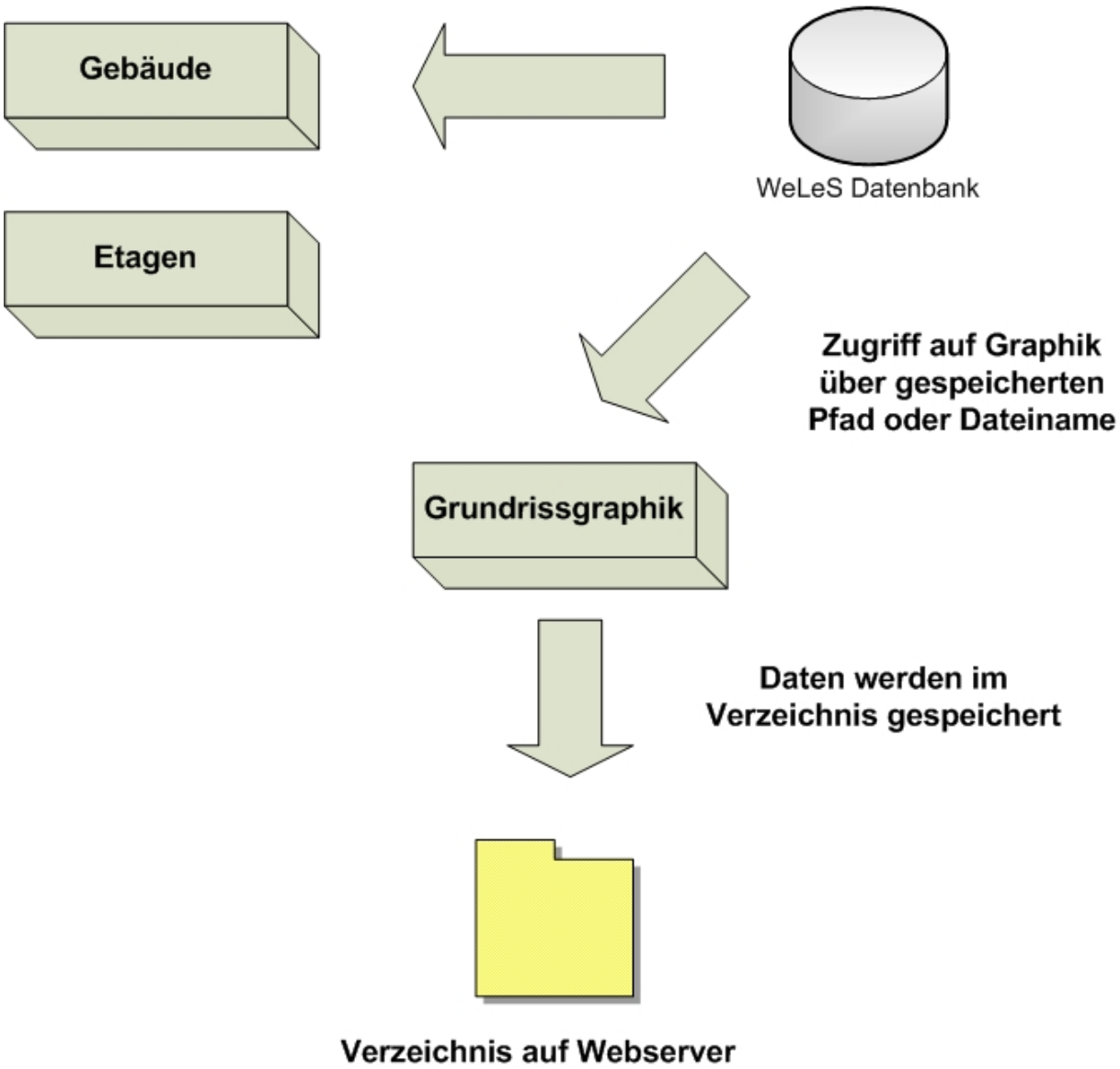


Abb.9 Verwaltung von Grundrissen im WeLeS

Wie in Abb. 9 dargestellt, werden Informationen zu den Gebäuden eines Gebäudekomplexes sowie den Etagen in der Datenbank gespeichert. Um diese Schnittstelle zu realisieren, müssen entsprechende Änderungen im WeLeS vorgenommen werden. Um dies zu erreichen, muss die Datenbank um eine Tabelle erweitert werden.

3.1.1 Datenbankerweiterung

Um alle Informationen der Graphik zu verwalten, muss die Datenbank um eine Tabelle GRAPHIK erweitert werden. Zu den gespeicherten Informationen zählen:

Attribut	Beschreibung
Id	dient zur eindeutigen Identifizierung der Grundrissdatei.
Name	Dateiname
Länge	Breite in Pixel
Breite	Höhe in Pixel
Maßstab	Maßstabszahl
Beschreibung	optional zusätzlich Angabe zur Datei.

Tabelle 1 zusätzliche Tabelle GRAPHIK

Beim Laden der Grundrissgraphik sollten Informationen über die Zugehörigkeit des Hauses und der Ebene aus der Tabelle gelesen werden. Diese Informationen werden bei der Erfassung und Speicherung der Punkte gebraucht. Daher wird die Beziehung zwischen der Tabelle HAUS und der Tabelle GRAPHIK über eine 1:N Beziehung realisiert. Jedes Haus besteht aus mehreren Grundrissgraphiken, und mehrere Grundrissgraphiken gehören zu einem Haus. Die Beziehung zwischen der Tabelle EBENEN und der Tabelle GRAPHIK wird über eine 1:1 Beziehung realisiert. Dadurch wird ausgedrückt, dass zu jeder Ebene eines Hauses auch wirklich nur eine Grundrissgraphik existiert. Eine ER- Model der kompletten Datenbank kann unter 3.4.1 entnommen werden.
Bevor ein Grundriss im System erfasst wird, sollte die Datei im Arbeitsverzeichnis des Webservers kopiert werden. Sobald ein Grundriss nicht mehr benötigt wird, sollte dieser aus dem Arbeitsverzeichnis und der Datenbank gelöscht werden.

3.1.2 Anforderungen der Graphikdateien

Die Graphiken die zu einem Gebäudekomplex gehören, sollten in derselben Auflösung und im gleichen Maßstab vorliegen. Dies vereinfacht die Weiterverarbeitung der Punkte und kann unerwünschte Effekte bei der graphischen Ausgabe verhindern. Das Programmmodul sollte auch in der Lage sein, aus den aufgenommenen Punkten ungefähre Entfernungsangaben zu berechnen. Für die Berechnung ist die Angabe des dpi Wertes erforderlich, wobei dpi für ***d**ots **p**er **i**nch* steht und Punkte pro Zoll (1 Zoll = 25,4mm) bedeutet. Mit diesem Wert wird die Punktdichte auf einem Bild angegeben. Um aus den Pixelkoordinaten Entfernungen zu berechnen sind daher folgende Schritte erforderlich:

1. Abstand der Punkte in Pixel ermitteln
2. Abstand durch den dpi Wert des Bildes dividieren
3. Multiplikation mit dem Maßstab und Umrechen in Meter

Folgendes Beispiel soll dies verdeutlichen.

$$geg:$$
$$\overline{p_1 p_2} = 136 Pixel$$
$$dpi = 96$$
$$Maßstab = 1:200$$

$$Berechnung:$$
$$\left(\left(\frac{136}{96}\right) * 25{,}4mm\right) * \frac{200}{1000} = \underline{\underline{7{,}2m}}$$

Um diese Schritte umzusetzen, müsste der dpi Wert in der Tabelle GRAPHIK als Attribut gespeichert werden.

Zur besseren Bearbeitung der Grundrissgraphiken im Arbeitsbereich, sollten diese innerhalb eines SVG Dokumentes eingebunden werden. Das Einbinden von externen Graphiken erfolgt in SVG mit dem `<image>`- Tag [7].

3.1.3 Fazit

Die Schnittstelle der Verwaltung der Grundrissdaten enthält folgende Spezifikationen:

- Arbeitsverzeichnis auf dem Webserver für die Graphiken
- Verwaltung der Graphikdateien in der Datenbank
- Verwaltung von Gebäuden und Etagen in der Datenbank

Um die Schnittstelle zu realisieren, müssen Änderungen an der Datenbank vorgenommen werden (3.1.1). Die Verwaltung der Grundrissdaten bildet die Grundlage zur Weiterverarbeitung der Graphiken für den Sachbearbeiter. Der Sachbearbeiter bearbeitet die Grundrissgraphiken im Arbeitsbereich. Dieser Bereich wird im folgenden Abschnitt beschrieben.

3.2 Arbeitsbereich

Der Arbeitsbereich beschreibt die Schnittstelle zur Bearbeitung der Grundrissdaten. Um eine Graphik zu bearbeiten, wird der Grundriss über die zuvor beschriebene Schnittstelle in den Arbeitsbereich geladen. Im Arbeitsbereich werden dem Sachbearbeiter eigene Werkzeuge zur Verfügung gestellt, mit denen er seine Aufgaben erfüllen kann. Die Aufgaben des Sachbearbeiters sind dabei folgende:

- Erfassung aller Punkte zur Beschreibung des Grundrisses
- Erfassung von Zielpunkten

3.2.1 Aufbau

Der Arbeitsbereich sollte übersichtlich gestaltet sein. So befinden sich die Werkzeuge zum Bearbeiten wie gewohnt auf der linken Seite. Die Darstellung des Grundrisses sollte mittig angeordnet sein. Der Aufbau könnte wie folgt aussehen.

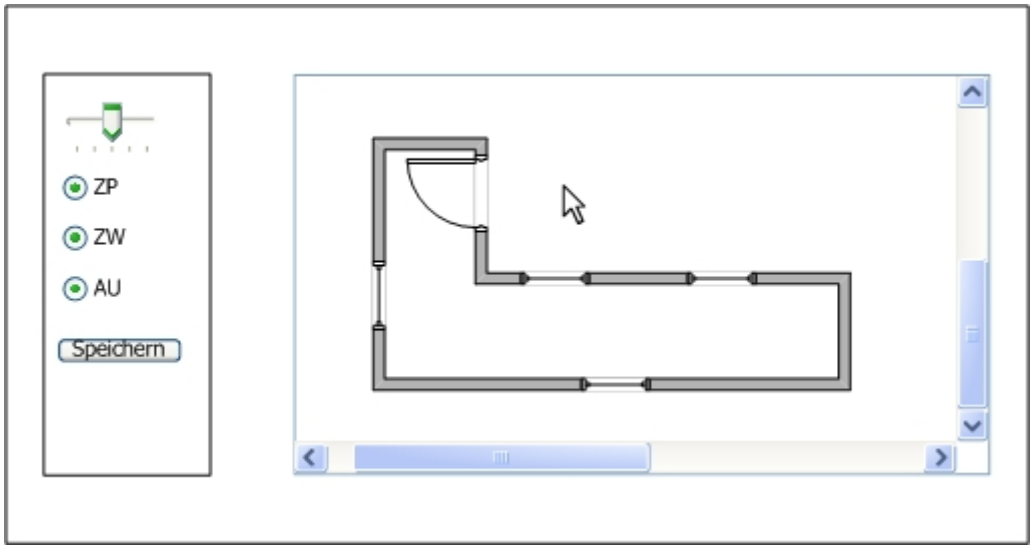


Abb.10 Skizze des Arbeitsbereiches

Diese Skizze veranschaulicht, wie der Arbeitsbereich aufgebaut sein könnte. Dabei werden keine Designvorgaben gemacht, da das Design der Webseiten vom Einsatz des Wegeleitsystems abhängt. Die Skizze zeigt, wo der Grundriss und die Werkzeuge im Arbeitsbereich platziert werden können. Ziel ist es, den Arbeitsbereich möglichst einfach zu gestalten, damit der Sachbearbeiter sich sofort zurechtfindet und sich nicht lange einzuarbeiten braucht.

Zu den Werkzeugen der Bearbeitung zählen das Zoomen, das Verschieben der Graphik beim Zoomen (Pan), die Auswahl der Punktart sowie das Speichern der hinzugefügten Punkte.

3.2.2 Änderungen an der Datenbank

Wie unter Punkt 2.4.1 erwähnt, ist es bei der aktuellen Definition der Datenbank äußerst kompliziert, die Arten der Grundrisspunkte zu erfassen. Um dem Sachbearbeiter die Aufnahme der Punkte zu erleichtern, sollte die Datenbank dahingehend verändert werden. Zur Aufnahme von Grundrisspunkten (H, I, G, O) sollte die Tabelle GRUNDRISS überarbeitet werden. Dabei wird die Unterscheidung der Punktart sowie die Angaben für die Attribute *Polygon* und *LoD* weggelassen. Ein Punkt wird nicht mehr mit den Attributen *x*, *y* abgebildet. Stattdessen werden die Umrisse und Räume eines Grundrisses direkt als `POLYGON` bzw. als `LineString` in der Datenbank abgebildet. Gänge müssten gar nicht erst erfasst werden, da diese sich aus den Zwischenräumen der erfassten Räume und Grundrisslinien ergeben. Um Türen bei der generierten Karte anzudeuten, werden die Punkte eines Raumes in SVG als `<polyline>` gezeichnet. Ein `<polyline>`- Tag zeichnet sich dadurch aus, dass es im Gegensatz zum `<polygon>`- Tag nicht geschlossen wird. Daher muss bei der Aufnahme der Punkte lediglich auf die Reihenfolge der aufzunehmenden Punkte geachtet werden. Die Tabelle Grundriss könnte nach den genannten Änderungen wie folgt aussehen:

Attribut	Beschreibung
Id	dient zur eindeutigen Identifizierung des erfassten Polygons
Polygon	Punkte (x, y) des erfassten Polygons
Art	Entweder Grundriss(g), oder Raum (r)
Beschreibung	Optionale nähere Beschreibung des Polygons (z.B. Umriss, Raum, Raumnummer).

Tabelle 2 überarbeitete Tabelle GRUNDRISS

Bei der Erfassung von Zielpunkten und Zielwegen spielt allerdings die Punktart eine wichtige Rolle. Daher darf bei dieser Art der Erfassung die Punktart nicht entfallen. Hier sollte die Datenbank dahingehend angepasst werden, dass die Zuordnung der Wegpunkte in einer extra Tabelle erfolgt. Prinzipiell ist es auch möglich, die Verwaltung der Ziel- und Wegpunkte mit Hilfe des Geometriedatentyps `POINT` zu realisieren. Für diese Änderungen müssten die PHP Scripte des Suchalgorithmus angepasst werden.
Die so durchgeführten Veränderungen erleichtern zum einen das Erfassen der Punkte sowie die Darstellung eines Gebäudes bei einer späteren Suchanfrage.

3.2.3 Punktaufnahme

Da das Erfassen der Orientierungspunkte auf der Clientseite im Browser erfolgt, muss hier JavaScript zum Einsatz kommen. Mit JavaScript ist es möglich, innerhalb eines HTML bzw. SVG Dokumentes Elemente über das DOM hinzuzufügen oder zu löschen. Das Aufnehmen der Punkte kann über das `onclick` Ereignis in JavaScript geregelt werden. Um dieses Ereignis auszulösen braucht nur auf die entsprechende Region der Graphik geklickt zu werden. Mit Hilfe von JavaScript könnte ein Punkt innerhalb des SVG Dokuments generiert werden, sowie die Pixelkoordinaten zur weiteren Verarbeitung zwischengespeichert werden. Folgendes Aktivitätsdiagramm soll dies veranschaulichen.

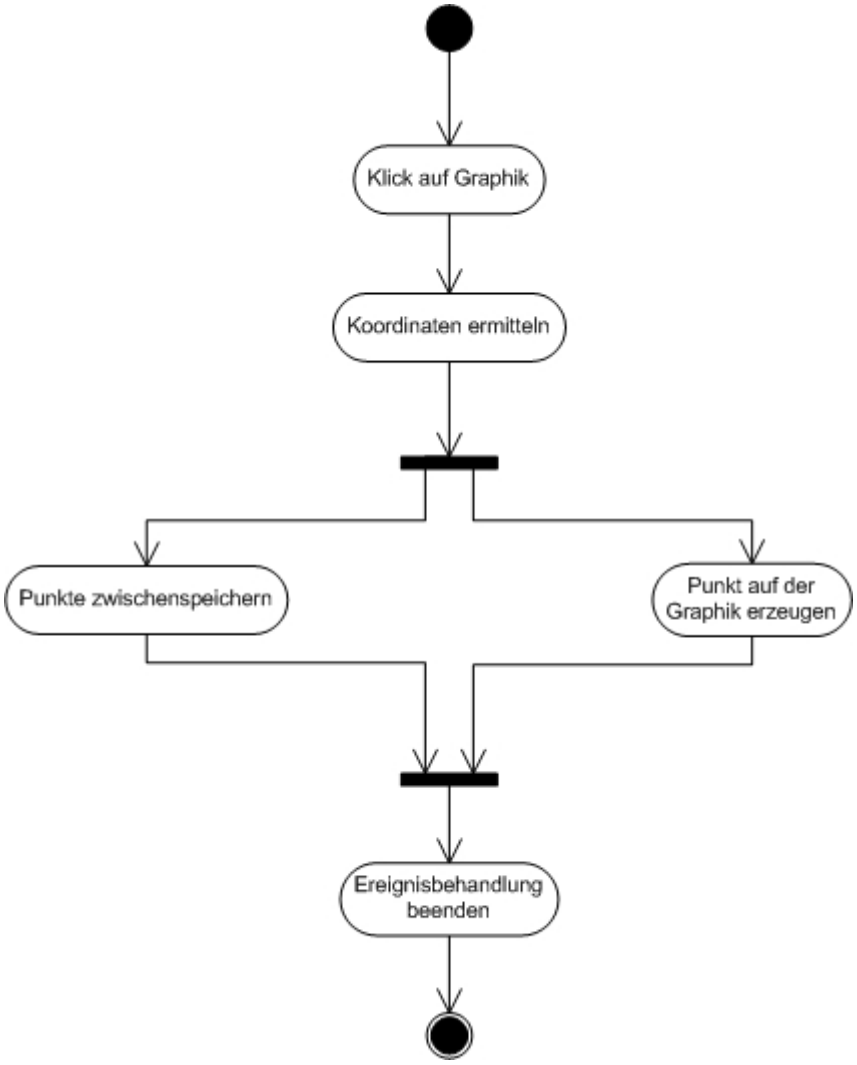


Abb. 11 Aktivitätsdiagramm zur Aufnahme von Grundrisspunkte

Die Koordinaten der Grundrisspunkte sollten auf der Clientseite zwischengespeichert werden. Damit die Grundrisspunkte endgültig in der Datenbank erfasst werden können, soll der Sachbearbeiter die Punkte bewusst speichern. So lange die Gebäudegrundrisse für alle Eta-

gen des Gebäudes gleich sind, würde es ausreichen, diesen nur einmal zu erfassen. Hierfür müsste ein einheitliches Konzept für die graphische Ausgabe definiert werden. Ein solches Konzept sollte alle möglichen Fälle beschreiben die bei der graphischen Ausgabe auftreten können.

Beim Hinzufügen neuer Ziel- und Wegpunkte gibt es einiges mehr zu beachten. Bei der Aufnahme von Zielpunkten und Zielwegen müssen zu den Koordinaten auch die Punktarten erfasst werden. Dazu zählt beispielsweise das Hinzufügen von Ziel- und Wegpunkte über mehrere Etagen. Diese wird mit der Punktart Treppe (TR) und Aufzug (AU) realisiert. Soll ein Zielweg zu einem anderen Gebäude führen, erfolgt dies über die Punktart Hauseingang (HE). Zur besseren Unterscheidung der Punktart, wäre es günstig diese in unterschiedlichen Farben darzustellen. Die so generierten Punkte dienen der besseren Übersicht. Da für die Aufnahme neuer Zielpunkte die Id's der Nachbarpunkte erfasst werden müssen, sollte das Programmmodul ein Algorithmus enthalten, der in der Lage ist die Nachbarpunkte automatisch zu ermitteln. Alle vorhandenen Ziel- und Wegpunkte werden in der Graphik angezeigt. Wird ein neuer Ziel- oder Wegpunkt hinzugefügt, kann der Sachbearbeiter diesen Punkt über eine Linie mit einen vorhandene Punkt verbinden. Beim Verbinden wird die Id des vorhandenen Punktes ermittelt. Die Id des Punktes wird über die Position der Mouse abgefragt. Dabei wird über die aktuelle Position der Mouse (x, y) die Id des dazugehörigen Punktes ermittelt. Über die Id wird der vorhandene Punkt als Nachbar des neuen Punktes hinzugefügt. Umgekehrt wird der neue Punkt über seine Id als Nachbar des vorhandenen Punktes hinzugefügt. Die Umsetzung der Nachbarschaftsbeziehungen wird erst beim Speichern in der Datenbank realisiert (3.3.2) Dieses Verfahren kann für weitere Nachbarpunkte wiederholt werden. Der Ablauf dieses Verfahrens soll durch folgendes Aktivitätsdiagramm dargestellt werden.

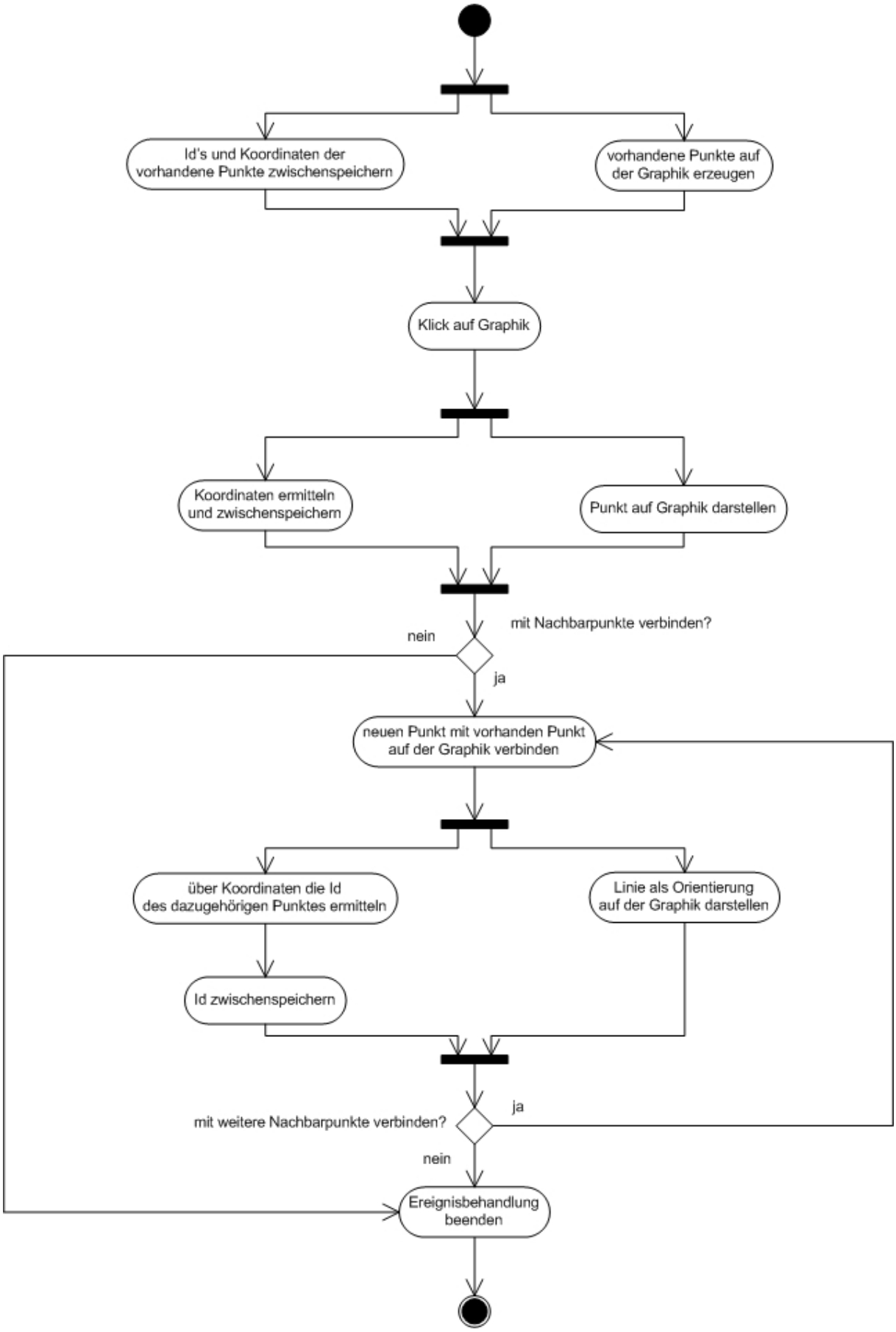


Abb. 12 Aktivitätsdiagramm zur Aufnahme von Ziel-, und Wegpunkte

Dieses Vorgehen erleichtert dem Sachbearbeiter das Erfassen von neuen Ziel- und Wegpunkten, da er sich keine Gedanken machen muss, welche weiteren Angaben für das Erfassen neuer Punkte zusätzlich zu treffen sind, damit der Suchalgorithmus funktioniert. Jeder Ziel- und Wegpunkt sollte einzeln erfasst und anschließend bewusst gespeichert werden.

Solange die neuen Punkte nicht endgültig gespeichert werden, hat der Sachbearbeiter die Möglichkeit, diese wieder zu löschen, ohne das im Hintergrund Löschoperation in der Datenbank durchgeführt werden. Beim Löschen eines Punktes im Arbeitsbereich werden alle zwischengespeicherten Werte wieder gelöscht. Mit dem endgültigen Speichern der Punkte beschäftigt sich der 3. Teil des Konzeptes (3.3).
Um die Punktaufnahme zu erleichtern, kann ein Raster über die Graphik gelegt werden. Dieses Raster kann ebenfalls mit SVG generiert werden. Der Abstand der Rasterlinien könnte z.B. 50 Pixel betragen. Dieses Raster könnte beispielsweise so aus sehen.

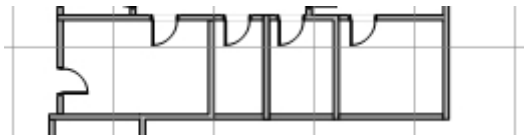


Abb. 13 Beispiel eines Raster mit 50 Pixel Abstand

Abb. 13 zeigt, wie ein Raster auf die zu bearbeitende Grundrissgraphik gelegt wird. Die Probleme, die bei der Punktaufnahme auftreten können, werden im nächsten Unterpunkt besprochen.

3.2.4 Probleme bei der Punktaufnahme

Problematisch wird die Aufnahme von Punkten bei Etagen, die aus mehreren Graphikdateien besteht. Hierfür muss nach der Bearbeitung der Graphiken eine Translation durchgeführt werden, da es ansonsten bei der Ausgabe der Umrisse und Zielwege zu Überschneidungen kommen kann. Für diesen Fall sollte der Arbeitsbereich eine Möglichkeit bereitstellen, entsprechende Angaben bei der Bearbeitung zu machen. Es gibt zwei Möglichkeiten, die Translation durchzuführen. Entweder wird bei der Grundrissgraphik der Koordinatenursprung verschoben, oder der Koordinatenursprung bleibt unverändert und alle aufgenommenen Punkte werden verschoben. Translationen werden in SVG mit dem Attribut `transfom` und den zugehörigen Wert `translate(tx, ty)` durchgeführt. Liegen die Graphiken im selben Maßstab und in derselben Auflösung vor, reicht eine Verschiebung des Koordinatenursprungs auf der x Achse aus. Wird der Koordinatenursprung verschoben, muss dafür gesorgt werden, dass die Graphik innerhalb des Arbeitsbereiches nicht verschoben wird.
Ein ähnliches Problem ergibt sich, wenn ein Gebäudekomplex aus mehreren Gebäuden besteht. Auch hier könnte es bei graphischen Ausgaben zu Überschneidungen kommen. Eine Möglichkeit dies zu umgehen wäre, alle Gebäude in einer Grundrissgraphik darzustellen. Eine weitere Möglichkeit wäre auch hier wieder, eine Translation durchzuführen. Wobei auch in diesem Fall entsprechende Angaben im Arbeitsbereich gemacht werden müssten und entschieden werden muss, wie die Translation realisiert wird.
Ein weiteres Problem betrifft die Genauigkeit bei der Erfassung der Punkte. Bei der Erfassung eines Raumes ist es gut möglich, nicht immer die gleichen horizontalen und vertikalen Pixel aller Ecken zu treffen. Um die Genauigkeit bei der Erfassung zu erhöhen, besteht in SVG die Möglichkeit zu zoomen. Dabei sollte aber berücksichtigt werden, dass sich beim zoomen das Bildkoordinatensystem in Relation zur Graphik verändert. Um dies auszugleichen, müssen im Hintergrund Koordinatentransformationen berechnet werden.

3.2.5 Fazit

Der Arbeitsbereich stellt den wichtigsten Teil des Programmmoduls bei der Erfassung von Orientierungspunkten dar. Ziel bei der Realisierung des Arbeitsbereich sollte sein, diesen übersichtlich und funktional zu gestalten. So ist der Sachbearbeiter in der Lage, ohne lange Einarbeitungszeit seine Aufgaben durchzuführen. Um dies zu realisieren, müssen im Hintergrund einige wichtige Änderungen bei der Verwaltung der Daten in der Datenbank durchgeführt werden (3.2.2). Diese Änderungen betreffen in erster Linie das Speichern der aufgenommenen Punkte. Mit der Realisierung der Schnittstelle zum Speichern der neu aufgenommenen Punkte befasst sich der nächste Abschnitt.

3.3 Verwaltung der erfassten Punkte

Das Aufnehmen der Punkte (3.2.3) steht im engen Zusammenhang mit dem dauerhaften Speichern der Punkte in der Datenbank. Daher beschreibt dieser Teil des Konzeptes, wie das Speichern der Punkte effektiv umgesetzt werden kann. Dabei spielen die unter 3.2.2 genannten Änderungen der Datenbank eine wichtige Rolle. Das Speichern der Punkte soll von dem Sachbearbeiter bewusst durchgeführt werden. Das bewusste Speichern erfolgt über einen entsprechenden Button. Über das `onclick` Ereignis des Buttons wird der Speichervorgang ausgelöst. Im Unterschied zum Speichern der Grundrisspunkte, soll jeder einzelne Ziel- und Wegpunkt erfasst und gespeichert werden.

3.3.1 Datenaustausch zwischen Client und Server

Das Aufnehmen der Punkte erfolgt, wie oben beschrieben, auf der Clientseite. Die Verarbeitung der Daten zum Speichern erfolgt auf der Serverseite. Somit müssen die aufgenommenen Punke dem ausführenden PHP Script auf der Serverseite übergeben werden. Für die Übergabe von Daten auf der Clientseite zum Server stellte das HTTP Protokoll zwei Methoden bereit GET und POST [15]. Bei GET werden die zu übergebenen Daten an die aufrufende URL angehängt. Ein Beispiel für die GET Methode sieht wie folgt aus.

```
http://www.beispielseite.de/rechner.php?var1=9&var2=9
```

In diesem Beispiel werden dem PHP Script auf dem Server zwei Variablen übergeben. Werden zu viele Zeichen übergeben, beantwortet der Webserver die Anfrage mit dem Fehlercode 413 *Request Entity Too Large*. Damit ein Sachbearbeiter beliebig viele Punkte aufnehmen kann, sollte daher die Übergabe der Werte mit POST erfolgen. Bei der POST Methode werden die Parameter nicht an der URL angehängt, sondern über einen eigenen Kanal dem Server übergeben. Die POST Methode bietet sich bei der Übergabe von Formulardaten an den Server besonders gut an, da Formulare oft sehr viele Daten enthalten. Werte, die nicht unbedingt vom Sachbearbeiter gesehen werden müssen, die aber bei der Verarbeitung trotzdem wichtig sind, können ebenfalls übergeben werden. Dafür können in HTML versteckte Formularelemente definiert werden. Die Syntax solcher Elemente sieht wie folgt aus:

```
<input type="hidden" name="unsichtbar" value=" ">
```

Im Programmmodul könnte dies beispielsweise die Id eines Haus bzw. einer Ebene sein.

Die Übergabe der Daten könnte wie folgt aussehen:

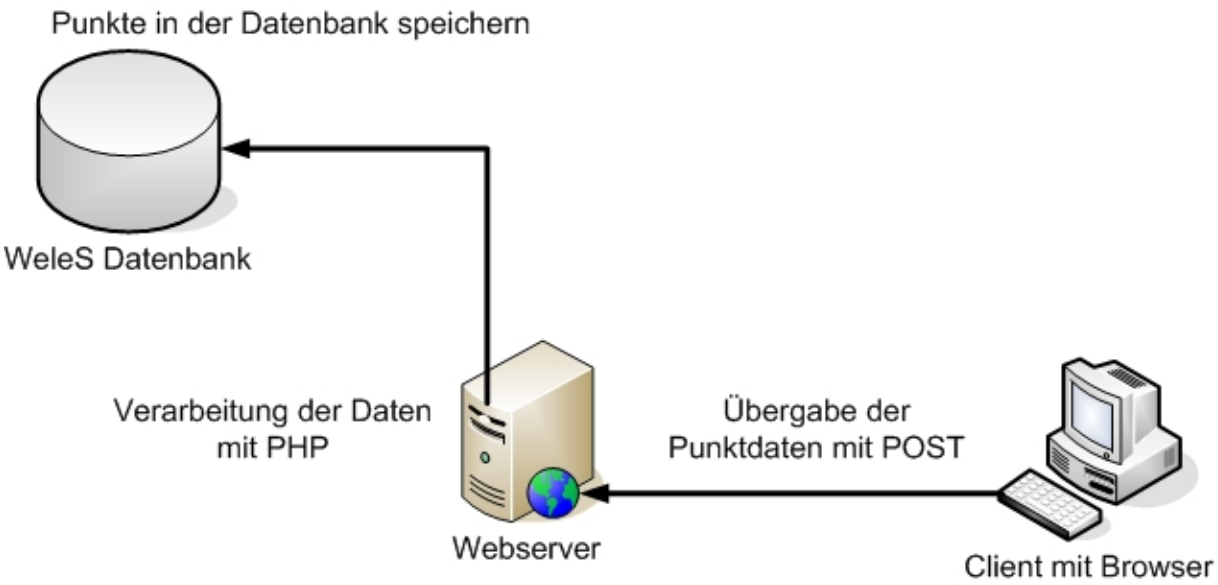


Abb. 14 Beispiel der Datenübergabe mit POST

Abb. 14 zeigt eine einfache Übergabe von Daten mit der POST Methode an den Server. Im Programmmodul erfolgt diese jedoch in mehreren Schritten, da einige Werte auf der Clientseite zwischengespeichert werden:

1. Übergabe der zwischengespeicherten Daten mit POST an den Server
2. dynamisches Generieren eines Formulars mit den erfassten Werten
3. Senden des Formulars an den Client
4. Bearbeiten des Formulars auf der Clientseite
5. Abschicken des Formulars mit der POST Methode an den Server
6. Verarbeitung der Formulardaten auf dem Server
7. Speichern der Daten in der Datenbank

Zwischen Punkt 5 und 6 sollte eine Überprüfung der Formulardaten erfolgen. Dies kann zum einen auf der Clientseite mit JavaScript beim Absenden, oder zum anderen nach dem Absenden auf der Serverseite mit PHP erfolgen.

3.3.2 Speichern von Grundrisspunkte

Bevor die Punkte endgültig gespeichert werden, sollte der Nutzer die Möglichkeit erhalten, Änderungen und evtl. Beschreibungen zu den einzelnen Punkten vorzunehmen. Dazu muss dynamisch in PHP ein Formular generiert werden. Beim Abschicken des Formulars müssen die Eingaben auf Fehler überprüft werden.

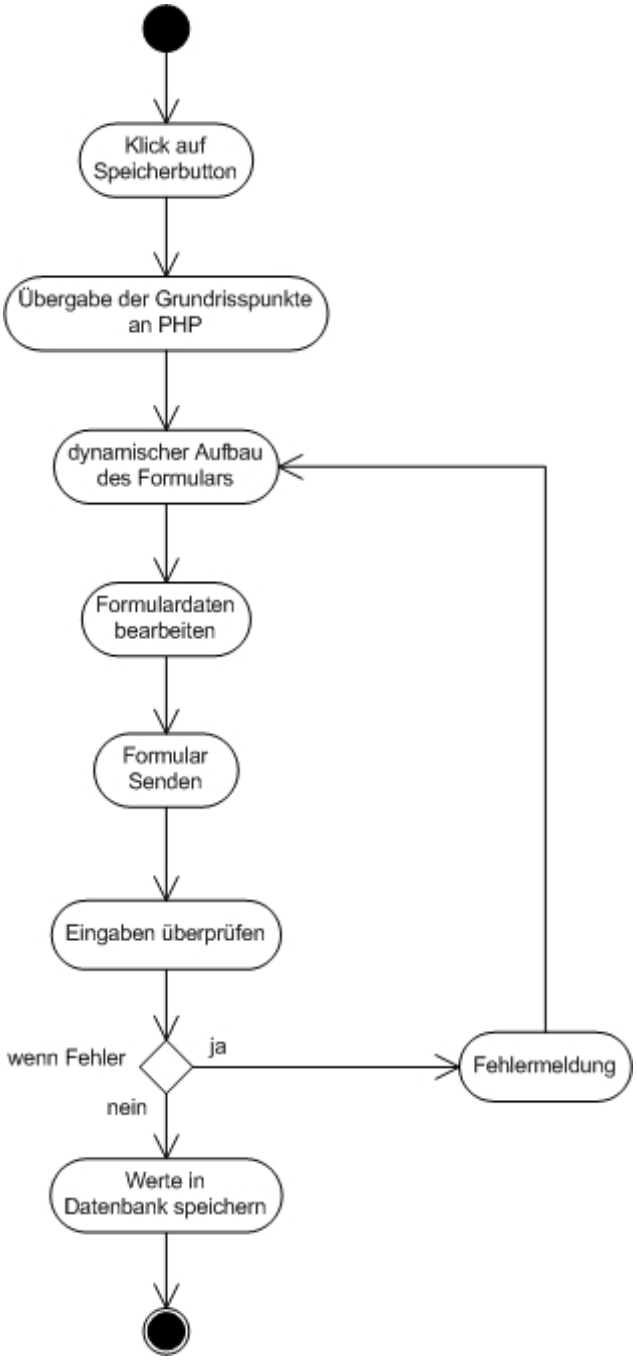


Abb. 15 Aktivitätsdiagramm zum Speichern von Grundrisspunkte

3.3.3 Speichern von Ziel- und Wegpunkte

Bevor ein neuer Ziel- oder Wegpunkt gespeichert wird, werden automatisch alle dazugehörigen Nachbarpunkten bei der Punktaufnahme ermittelt (3.2.3). Bevor ein Punkt bewusst über einen Button gespeichert wird, sollte zusätzlich abgefragt werden, ob dieser wirklich gespeichert werden soll. Im Anschluss daran sollte der Sachbearbeiter die Möglichkeit erhalten, über ein Formular Angaben über Punktart, Ziel, Zuordnung und Bezeichnung machen zu können. Zum einen wird der Punkt mit allen zugehörigen Attributen gespeichert, zum anderen erfolgt eine Zuordnung des Punktes zu seinen Nachbarpunkten. Außerdem müssen bei allen infrage kommenden Nachbarpunkten der neue Punkt als Nachbar hinzugefügt werden. Auch dieser Prozess soll durch ein Aktivitätsdiagramm veranschaulicht werden.

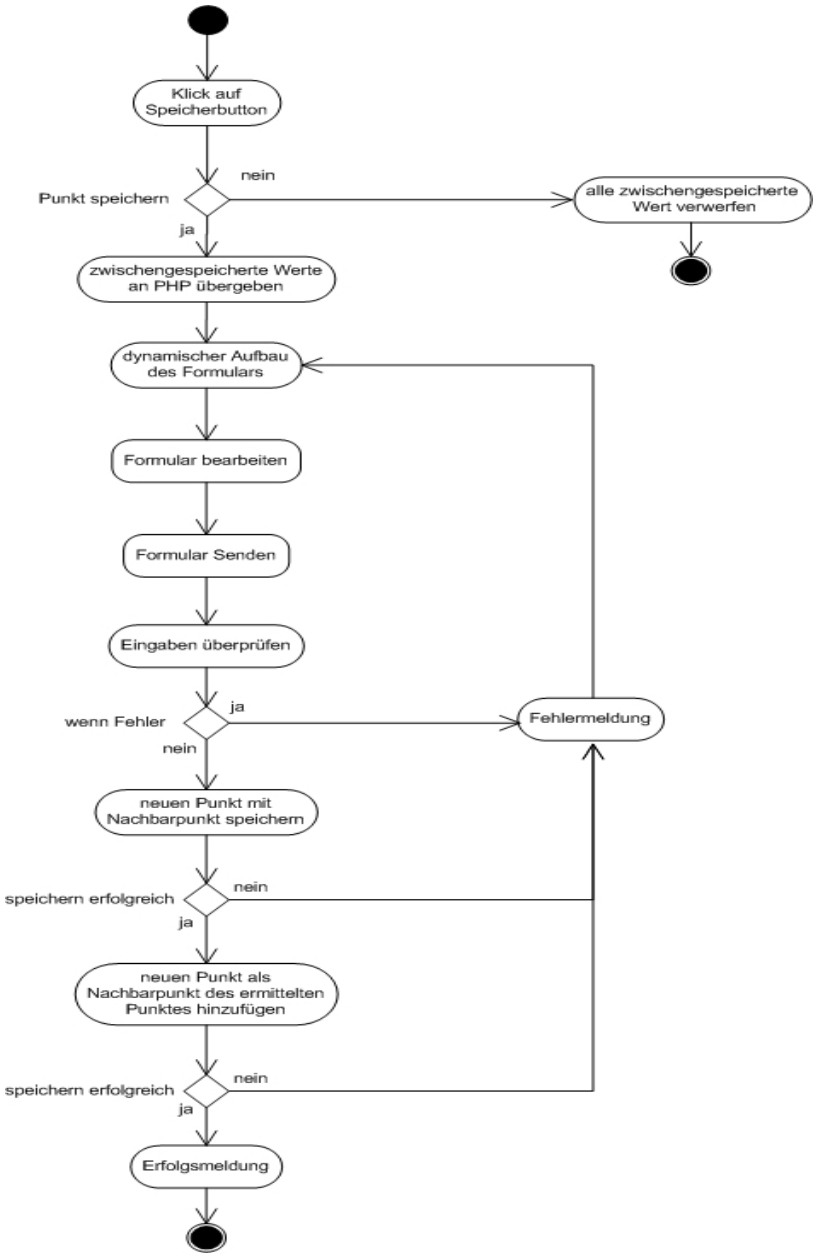


Abb. 16 Aktivitätsdiagramm zum Speichern von Ziel- und Wegpunkte

3.3.4 Fazit

Das Speichern der Punkte beschreibt den letzten Teil des Konzeptes und schließt sich direkt an die Erfassung der Punkte im Arbeitsbereich an. Die Übergabe der Formulardaten sollten mit der POST Methode erfolgen, da dort mehr Daten übergeben werden können. Das Speichern der Punkte soll bewusst vom Sachbearbeiter durchgeführt werden, wobei die Ziel- und Wegpunkte einzeln zu speichern sind. Der gesamte Vorgang zum Speichern erfolgt in 7 Schritten:

1. Übergabe der zwischengespeicherten Daten mit POST an den Server
2. dynamisches Generieren eines Formulars mit den erfassten Werten
3. Senden des Formulars an den Client
4. Bearbeiten des Formulars auf der Clientseite
5. Abschicken des Formulars mit der POST Methode an den Server
6. Verarbeitung der Formulardaten auf dem Server
7. Speichern der Daten in der Datenbank

Das dynamische generieren des Formulars kann mit Hilfe der AJAX Technik [18] realisiert werden. AJAX beschreibt die asynchrone Datenübertragung zwischen Client und Server, um eine Webseite nicht neu laden zu müssen. Die zwischengespeicherten Daten werden über AJAX an ein PHP Script übergeben, ohne das der Arbeitsbereich neu geladen werden muss. Die Übergabe der Daten erfolgt auch bei AJAX mit Hilfe der POST Methode. Das PHP Script erstellt aus den übergebenen Daten ein Formular. Über dieses Formular können die Daten noch manuell bearbeitet werden. Anschließend können die Daten in die Datenbank geschrieben werden.

3.4 Fazit des Konzeptes

3.4.1 Veränderungen der Datenbank

Die Erarbeitung des Konzeptes hat gezeigt, dass einige Änderungen im WeLeS notwendig sind. In erster Linie betreffen diese Änderungen die zugrundeliegende Datenbank. Diese Änderungen führen zu einer Überarbeitung des ER- Modells der Datenbank.

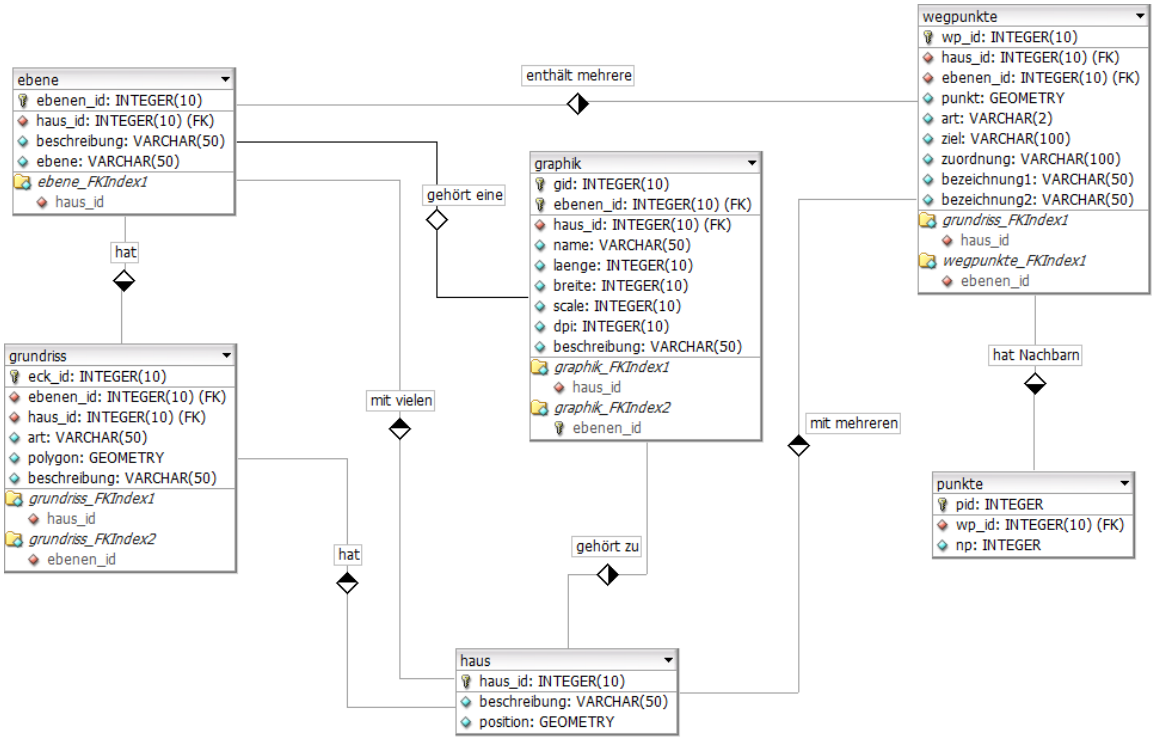


Abb. 17 überarbeitetes ER- Modell

Wie in Abb. 17 dargestellt, enthält die überarbeitete Datenbank 2 neue Tabellen, die Tabellen GRAPHIK und PUNKTE. Die Tabelle GRAPHIK resultiert aus der unter 3.1. beschriebenen Schnittstelle zur Grundrissverwaltung. Um die ursprüngliche Beschränkung von 6 Nachbarpunkten aufzuheben und um den Entwurf zu verbessern, wurde die Zuordnung der Ziel- und Wegpunkte in die Tabelle PUNKTE ausgegliedert. Diese neue Tabelle PUNKTE ist kein Resultat der Erarbeitung des Konzeptes, sondern soll zur Verbesserung des WeLeS beitragen.

Die größte und wichtigste Veränderung betrifft die Tabelle GRUNDRISS und ist ein Resultat der benutzerfreundlicheren Punktaufnahme, die unter 3.2 beschrieben wurde. In der Tabelle GRUNDRISS wurden zum einen die Attribute *x*, *y* und *LoD* weggelassen und zum anderen die Bedeutung des Attributs *Polygon* verändert. Im Attribut *Polygon* können jetzt die Grundrisspunkte als Polygon und Polyline gespeichert werden. Dafür sorgt ein MySQL Geometrie-

datentype (entweder `LineString` oder `POLYGON`). Zu empfehlen wäre der Geometriedatentype `LineString`, da dieser für die Abfrage der Punktdaten nützlichere Funktionen bereitstellt (Anhang 4). Im Attribut *art* werden nur noch 2 verschiedene Punktarten unterschieden (Grundriss (G) und Raum(R)).
Auch das Speichern der Ziel- und Wegpunkte kann mit dem Geometriedatentyp `POINT` realisiert werden. Die Position der Beschriftung der Häuser aus der Tabelle HAUS könnte ebenfalls mit `POINT` angegeben werden. Diese Änderung wäre aber optional.

3.4.2 Mögliche Probleme bei der Darstellung

Ein Problem bei der Verwaltung der Punkte mit Geometriedatentypen ergibt sich bei der Generierung der graphischen Ausgabe in SVG. In SVG werden die Grundrisse und Zielwege entweder mit dem `<polyline>-` Tag oder `<polygon>-` Tag gezeichnet. In beiden Fällen werden die Koordinatenpaare im Attribut *`points`* aufgereiht. Das bedeutet, dass alle Punkte eines gespeicherten `LineStrings` abgefragt und anschließend im SVG Tag eingefügt werden müssen. Eine erste Möglichkeit könnte folgende Schritte umfassen:

1. Ermitteln aller Punkte zu einem `LineString`
2. mit Hilfe der MySQL `POINTN` Funktion jeden einzelnen Punkt abfragen
    - Rückgabewert der `POINTN` Funktion sieht wie folgt aus: `POINT(102 500)`
3. über Stingverarbeitung den x und y Wert rausziehen
4. Ergebnis zurückgeben
5. einzelne Punkte in SVG Tag einfügen

Diese erste Möglichkeit erfordert im Hintergrund mehrere Datenbankabfragen, was sich evtl. negativ auf die Performance des Systems auswirken kann. Grund dafür ist die `POINTN` Funktion mit der jeder einzelne Punkt aus dem Rückgabestring abgefragt werden muss.
Eine zweite Möglichkeit wäre den kompletten Rückgabestring so zu bearbeiten, dass alle nicht benötigten Zeichen aus dem String entfernt werden. Der Rückgabestring sieht wie folgt aus: `LINESTRING(227 87,617 87,617 287,466 287,466 371)`. Aus diesem String müssen alle Buchstaben und Klammern entfernt werden. Im Anschluss kann der String im `<polyline>-` Tag oder `<polygon>-` Tag des SVG Dokumentes eingefügt werden.

3.4.3 Zusammenfassung

Ziel des Konzeptes ist die Beschreibung eines Programmmoduls zur Erfassung von Orientierungspunkten. Das Modul hat die Aufgabe, die Erfassung der Punkte recht einfach zu gestalten, so dass jeder zukünftige Sachbearbeiter des Systems ohne lange Einarbeitung in der Lage ist, Punkte zur erfassen. Die Funktionsweise des Konzeptes soll durch einen einfachen Prototyp im folgenden Kapitel demonstriert werden.

4 Demonstration des Konzeptes an einem Prototyp

Ziel des Prototyps ist es, die Funktionsweise des erarbeiteten Konzeptes zu demonstrieren. Dabei werden die wichtigsten, im Konzept genannten, Änderungen des WeLeS umgesetzt. Die wichtigsten Funktionen des Programmmoduls ergeben sich aus der Erarbeitung des Konzeptes und sind:

- die Verwaltung der Grundrissgraphiken
- die Punktaufnahme (Digitalisierung)
- das Speichern der Punkte

Hauptaufgabe des Prototyps ist die Aufnahme von Orientierungspunkten. Dabei fließen die unter 2.5.2 genannten Anforderungen zur technischen Umsetzung ebenfalls bei der Entwicklung des Prototyps mit ein. So bilden HTML, PHP, MySQL und SVG die grundlegenden Technologien. Die Erfassung der Punkte erfolgt mit Hilfe von JavaScript auf der Clientseite. Die Übergabe der Daten zum Server erfolgt über die POST Methode. Der Prototyp soll vor allem zeigen, welchen Vorteil die Verwendung von Geometriedatentypen bei der Punktaufnahme mit sich bringt. Das bedeutet, dass die Daten aus dem zuvor entwickelten WeLeS Projekt für das Klinikum in der aktuellen Form nicht weiter verwendet werden können. Außerdem wird für den Prototyp die graphische Ausgabe vereinfacht. So entfallen zusätzliche Angaben wie die Ausgabe einer textlichen Beschreibung und Vergrößerung des Zieles vorerst im Prototyp. Diese Angaben sollten für eine endgültige Entwicklung aber wieder aufgenommen werden, da die Beschreibung des Zielweges für einen Besucher äußerst wichtig ist. Grundlage zur Aufnahme von Orientierungspunkten bilden im Prototyp einfache Test JPEG Dateien. Erfasst werden auf diesen Dateien die Pixelkoordinaten. Um die Funktionsweise des Suchalgorithmus im Prototyp nicht zu verändern, bleibt bei der Entwicklung die Verwaltung der Ziel- und Wegpunkte in der Datenbank unverändert. In diesem Zusammenhang wird die im Konzept genannte automatische Ermittlung eines Nachbarpunktes zu einem Ziel- und Wegpunkt im Prototyp nicht realisiert (3.2.3). Somit müssen die Nachbarpunkte im Prototyp vom Sachbearbeiter vorerst automatisch eingegeben werden. Dies muss aber bei einer endgültigen Entwicklung automatisiert werden, da es bei einem Gebäudekomplex für den Sachbearbeiter zu schwierig und unübersichtlich ist, alle Nachbarpunkte zu erfassen. Den ersten Schritt bei der Entwicklung bildet der Entwurf des Prototyps.

4.1 Entwurf

Den Beginn der Entwicklung des Prototyps bildet die Beschreibung des derzeitigen Funktionsumfangs durch ein Use Case Diagramm.

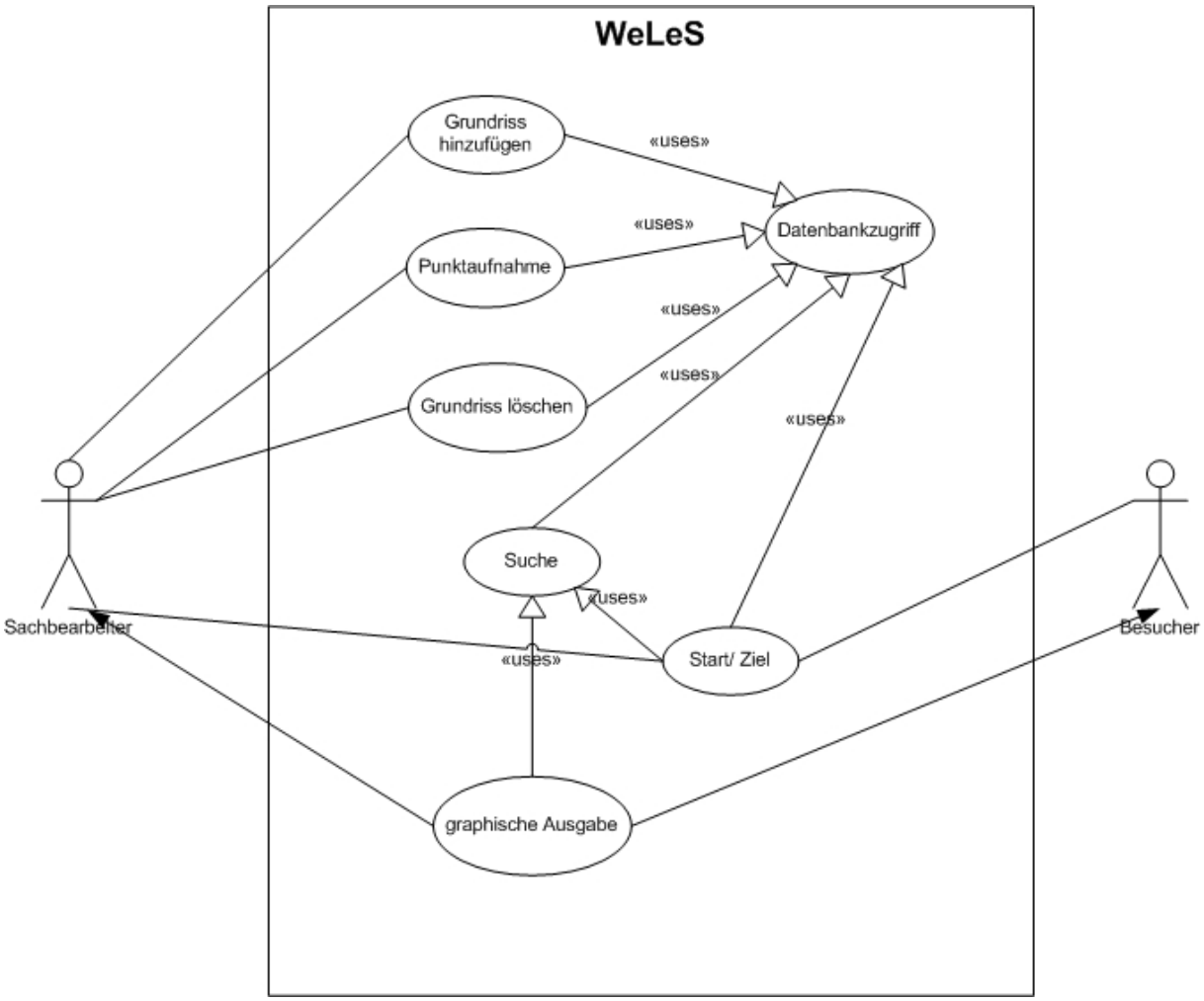


Abb. 18 Use Case Diagramm des Prototyps

Das Use Case Diagramm beschreibt den derzeitigen Funktionsumfang des WeLeS Prototypen. Dabei wird deutlich, dass es 2 Arten von Nutzer gibt, zum einen den Sachbearbeiter der Punkte im System einpflegt, und zum anderen den Besucher des Gebäudekomplexes der Suchabfragen zur Auskunft durchführt. Im Prototyp wird keine Unterscheidung zwischen den beiden Benutzergruppen vorgenommen. Bei einer Weiterentwicklung des WeLeS sollte daher eine Benutzerverwaltung eingeführt werden. Das System sollte 3 verscheiden Benutzerarten unterscheiden (Administrator, Sachbearbeiter, Besucher). Der Administrator sollte vollen Zugriff auf das System zur Pflege und Wartung erhalten, der Sachbearbeiter Lese- und Schreibrechte erhalten, der Besucher nur Leserechte.

Die Funktionen des Prototyps lassen sich ebenfalls in 2 Gruppen einteilen:

- Funktionen zur Datenerfassung
- Suchfunktion mit graphischer Ausgabe

Alle Funktionen die für die Datenerfassung wichtig sind, bilden zusammen das neu entwickelte Programmmodul zur Erfassung von Orientierungspunkten. Die Suchfunktion mit graphischer Ausgabe bildet den bisherigen Funktionsumfang des WeLeS.
Beide Funktionen sollen durch ein Aktivitätsdiagramm näher beschrieben werden.

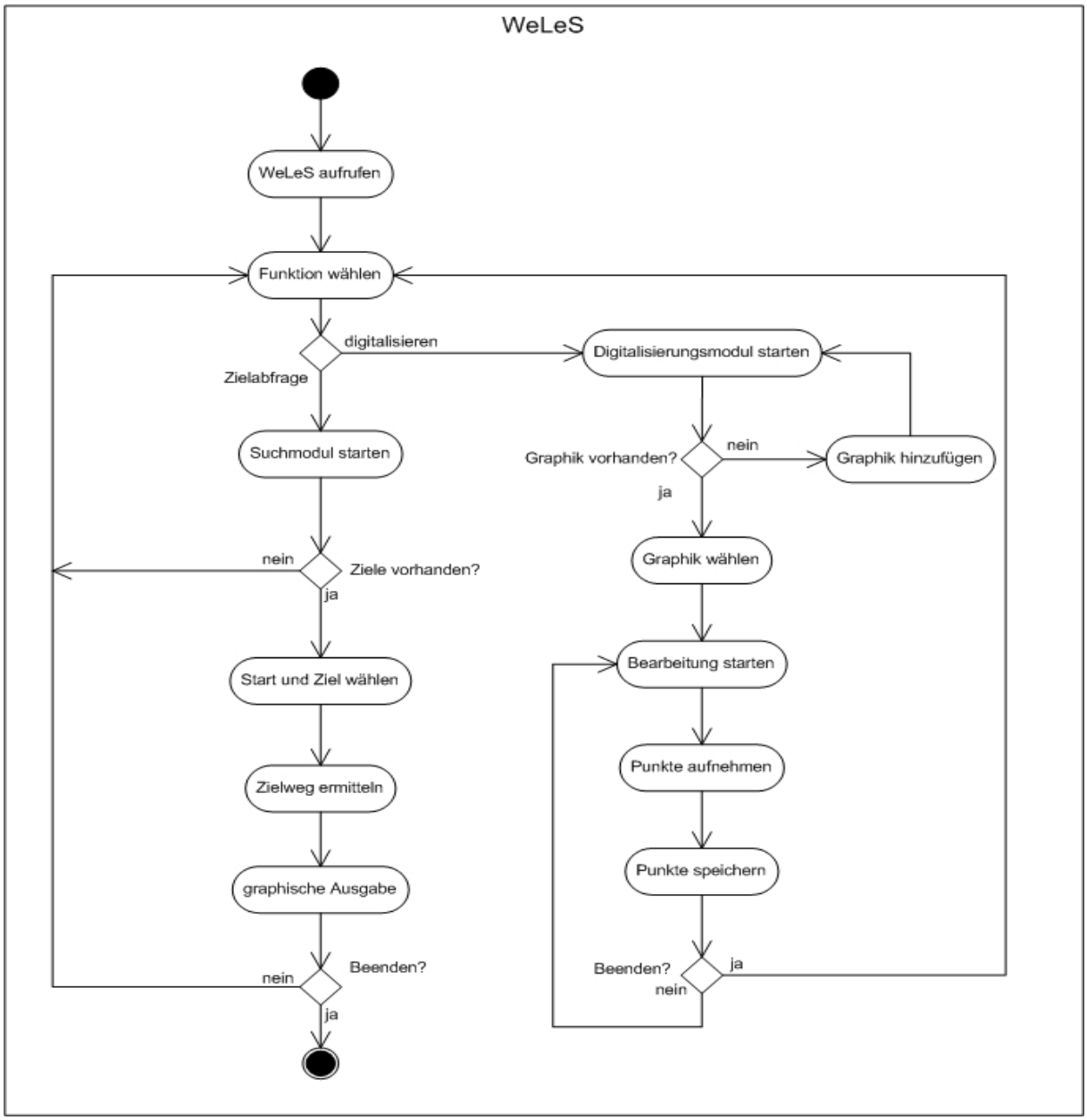


Abb. 19 Aktivitätsdiagramm des Prototyps

Nach dem Aufruf des WeLeS im Browser hat der Nutzer die Möglichkeit, auf der Startseite die gewünschte Funktion auszuwählen. Beim Start des Digitalisierungsmoduls, wird überprüft ob Graphiken zum Bearbeiten vorhanden sind. Sind keine Graphiken vorhanden, muss der Benutzer diese erst hinzufügen. Beim Hinzufügen einer Graphik sollten Angaben zu den Tabellen HAUS und EBENE in der Datenbank schon vorhanden sein. Ist dies nicht der Fall, sollten entsprechende Angaben nachgeholt werden. Die Punktaufnahme kann sooft durchgeführt werden, bis der Sachbearbeiter diese abbricht. Nach Beenden der Punktaufnahme wird der Sachbearbeiter auf die Startseite des WeLeS umgeleitet. Der Ablauf bei der Suchabfrage hat sich im Vergleich zum bisherigen WeLeS nicht verändert. Hierbei werden für die Suche Start und Ziel ausgewählt. Aus der Datenbank werden die entsprechenden Informationen ermittelt, sowie eine Karte für die Ausgabe der Zielführung generiert.
Der letzte Schritt beim Entwurf des Prototyps bildet das ER- Model der Datenbank. Dieses sieht wie folgt aus.

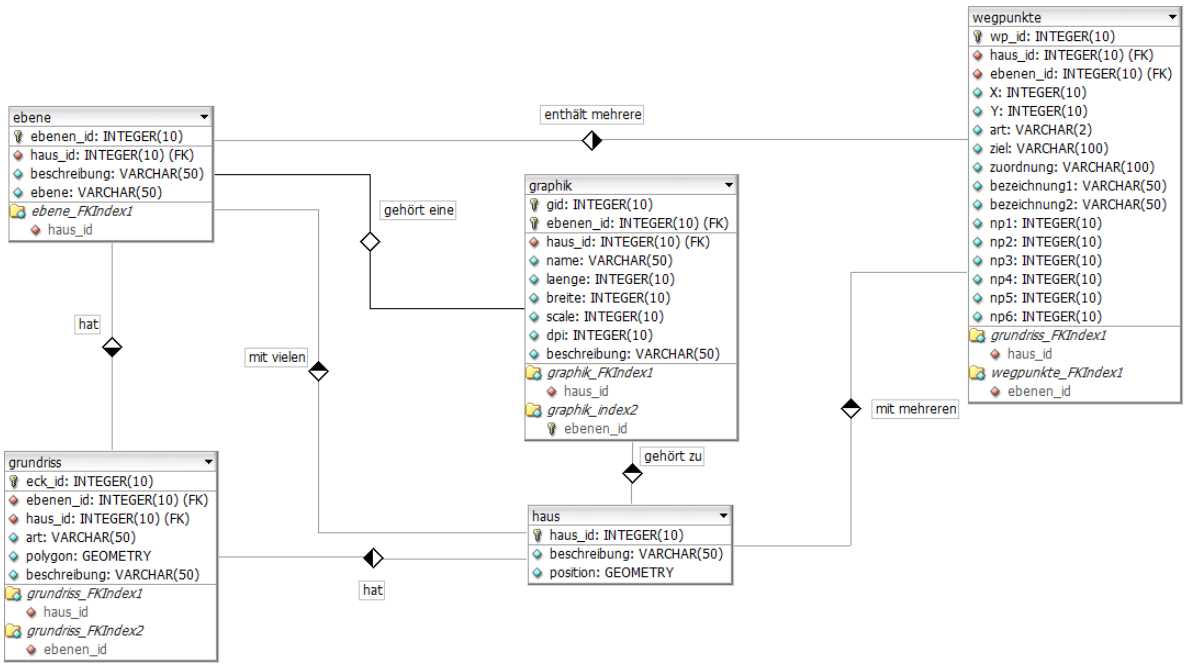


Abb. 20 ER- Model des Prototyps

Das ER- Model zeigt, dass fast alle im Konzept erarbeiteten Änderungen in der Datenbank für den Prototyp umgesetzt werden, bis auf die Verwaltung der Ziel- und Wegpunkte in der Tabelle WEGPUNKTE. Die Verwaltung der Grundrisspunkte in der Tabelle GRUNDRISS erfolgt mit Hilfe des Geometriedatentyp `LineString`. Der folgende Abschnitt beschreibt die Umsetzung und Beschreibung des hier erarbeiteten Entwurfes.

4.2 Umsetzung und Beschreibung des Prototyp

Die Realisierung des Prototyps erfolgt mit Hilfe der Software Distribution XAMPPlite [16] in der Version 1.6.8. Zu dieser Distribution gehört unter anderen:

- Apache 2.2.9
- MySQL 5.0.67
- PHP 5.2.6
- phpMyAdmin 2.11.9.2

Eingesetzte Browser:

- Opera 10.0
- Firefox 3.5.3
- Internet Explorer 7,8
    - Adobe SVG Viewer 3.03.0.94

Das Layout der Webseiten des Prototyps ändert sich nicht im Vergleich zu dem bisher entwickelten WeLeS. Hinzu kommt nur eine Fußzeile mit Links zur Startseite oder zur vorherigen Seite. Um ein einheitlicheres Design zu gewährleisten und die Pflege einfach zu gestalten, werden die Formateigenschaften in einer separaten CSS Datei gespeichert [17]. Diese Datei kann in jeder Webseite des WeLeS eingebunden werden. Die Referenz der CSS Datei erfolgt im HTML Dateikopf und sieht im WeLeS wie folgt aus.

```
<link rel='stylesheet' type='text/css'href='/neuweles/digi/Bilder/layout.css'>
```

Alle erforderlichen Dateien des Programmmoduls werden der Übersicht halber in einem separaten Verzeichnis gespeichert. Im Prototyp heißt dieses Verzeichnis `digi`.
Die Startseite (`index.php`) des WeLeS wird im Browser aufgerufen. Auf der Startseite erhält der Sachbearbeiter die Möglichkeit, die Digitalisierungs- bzw. die Suchfunktion zu starten. Sofern kein Start und Ziel im WeLeS vorhanden sind, bleiben die Dropdown Menüs der Suchfunktion leer.

4.2.1 Digitalisierungsfunktion

Diese Funktion beschreibt das neu entwickelte Programmmodul im Rahmen dieser Bachelorarbeit. Im Prototyp werden folgende Funktionen realisiert:

- Hinzufügen neuer Häuser eines Gebäudekomplexes
- Hinzufügen neuer Ebenen eines Hauses
- Hinzufügen neuer Grundrissgraphiken
- Digitalisierung der Grundrissgraphiken
    - Hinzufügen von Grundrisspunkten
    - Hinzufügen von Ziel- und Wegpunkten

Aufgerufen wird das Programmmodul über die `tool.php` im `digi` Unterverzeichnis des WeLeS. Auf dieser Seite können die gewünschten Funktionen aufgerufen werden.

Abb. 21 Funktionsauswahl im Programmmodul

Abb. 21 stellt die Auswahl der Funktionen im Prototyp dar. Im Folgenden werden die einzelnen Funktionen kurz beschrieben.

4.2.1.1 Haus und Ebene hinzufügen

Diese beiden Funktionen bilden den ersten Schritt, um die im Konzept beschriebene Grundrissverwaltung zu realisieren. Mit Hilfe dieser Angaben kann eine Grundrissgraphik einem Haus und einer Ebene zugeordnet werden. Mit einem Klick auf den Button `Hinzufügen` wird entweder die `formHaus.php` oder die `formEbene.php` gestartet. Beide Scripte generieren die entsprechenden Eingabeformulare. Zu beachten ist, dass beim Hinzufügen einer Ebene schon ein Haus in der Datenbank vorhanden sein muss, da sonst die Ebene keinem Haus zugeordnet werden kann. Das Speichern der Formulardaten in der Datenbank wird mit

Hilfe der `hausDb.php` und `ebeneDb.php` Scripte realisiert. Im Anschluss daran kann eine Grundrissgraphik in der Datenbank hinzugefügt werden.

4.2.1.2 Grundrissgraphik hinzufügen

Der zweite Schritt zur Realisierung der Grundrissverwaltung beschreibt das Hinzufügen einer Graphikdatei selbst. Bevor die Graphik in der Datenbank aufgenommen werden kann, sollte diese im Arbeitsverzeichnis des WeLeS gespeichert werden. Im Prototyp befinden sich die Beispieldaten im Verzeichnis `/digi/bilder/grundriss/`. Beim Hinzufügen einer Graphik wird nur der Dateiname in der Datenbank gespeichert. Wie der Zugriff auf die Graphikdateien für die Digitalisierung erfolgt, wird weiter unten beschrieben. Für den Aufbau des Eingabeformulars sorgt das `formRiss.php` Script. Aufgerufen wird das Script über den Button `Hinzufügen`.

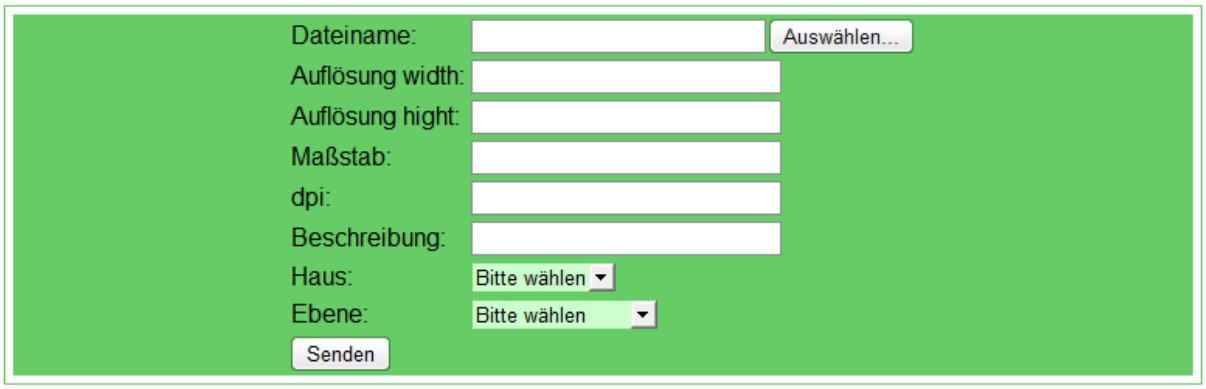


Abb. 22 Formular zum Hinzufügen von Grundrissgraphiken

Die Formulardaten werden über das `rissDb.php` Script in der Datenbank gespeichert. Anschließend können die Graphikdateien bearbeitet werden. Zur Bearbeitung der Graphiken stehen 2 Funktionen zu Verfügung:

- Grundrisspunkte digitalisieren
- Ziel- und Wegpunkte digitalisieren

4.2.1.3 Punktaufnahme

Die Punktaufnahme erfolgt im Arbeitsbereich. Genauer gesagt handelt es sich um zwei Arbeitsbereiche, einer für die Aufnahme von Grundrisspunkten und ein weiterer für die Aufnahme von Ziel- und Wegpunkte. Die Merkmale zu diesem Bereich wurden im Konzept erarbeitet (3.2). Für den Prototyp wurden dabei nur die wichtigsten Werkzeuge realisiert. Zu diesen Werkzeugen zählen:

- das Hinzufügen und Löschen von neuen Punkten
- die Auswahl von Punktarten beim Hinzufügen von Ziel- und Wegpunkten
- Speichern der Punkte

Nicht realisiert:

- die Zoom und Pan Funktionen
- Grundrissbearbeitung auf mehreren Graphikdateien (3.2.4)

Der Arbeitsablauf zur Punktaufnahme ist im Prototyp noch recht umständlich. Zur besseren Orientierung dient die Anzeige der aktuellen Position der Mouse.
Damit auf der Grundrissgraphik Punkte aufgenommen werden können, muss diese in den Arbeitsbereich eingebunden werden. Dabei wird die Rastergraphik innerhalb eines SVG Dokumentes eingebunden. Dies erfolgt in der PHP Klasse `grundriss()`, die sich in der gleichnamigen Datei `grundriss.php` befindet (Anhang 6).
Diese Klasse enthält zwei Methoden, `build()` und `bulidZiel()`. Beide Methoden haben die Aufgabe, die ausgewählte Grundrissgraphik als externe Graphik in einer SVG Datei einzubinden. Die `build()` Methode erstellt die SVG Datei für die Aufnahme von Grundrisspunkten. Und die `buildZiel()` Methode die SVG Datei für die Aufnahme von Ziel- und Wegpunkten. Bei beiden Methoden wird der Dateiname der Grundrissgraphik, die Länge und Breite in Pixel ermittelt und als Link über das `<image>` Element innerhalb des SVG Dokuments eingebunden. Der Rückgabewert beider Funktionen ist der SVG Dateiname. Dieser Dateiname wird in der `digi.php` bzw. `digi_ziel.php` über das `<object>` Element im jeweiligen Arbeitsbereich eingebunden. Innerbalb der SVG Dateien wird im `<defs>` Bereich ein Raster gebildet, dass über die Graphik gelegt wird. Das Raster hat einen Abstand in x und y Richtung von 50 Pixel. Dies kann im Quelltext nach Belieben angepasst werden.

Die Schnittstelle zum Hinzufügen neuer Punkte auf der SVG Graphik innerhalb der Webseite wird über JavaScript realisiert. Der Übersicht halber wird der JavaScript Code für das Hinzufügen von Grundrisspunkten sowie Ziel- und Wegpunkte in zwei externe Dateien (`grundriss.js, zielpunkte.js`) ausgelagert und in dem HTML Dokument des entsprechenden Arbeitsbereiches eingebunden (Anhang 5). Das Einbinden der JavaScript Dateien erfolgt wie schon das Einbinden der CSS Datei im HTLM Dateikopf. Der Einsatz von JavaScript hat folgende Vorteile:

- Zugriff auf das DOM des HTML und SVG Dokumentes
- Reaktion auf Ereignisse (Event) im Browser (z.B. `onlcick, onmouseover, onmouseout, onmousemove`)

Beide Dateien sind fast identisch. Der Unterschied in der `zielpunkte.js` besteht darin, dass hier die Punktarten der Ziel- und Wegpunkte unterschieden werden. Jede Punktart erhält zur besseren Unterscheidung bei der Aufnahme eine eigene Farbe.
Um auf die jeweiligen Benutzerereignisse zu reagieren wird im `<body>` Tag des HTML Dokuments eines Arbeitsbereiches die `Init()` Funktion aufgerufen. In dieser Funktion wird als erstes die eingebundene SVG Datei als SVGDocument- Objekt der globalen Variable `SVGDoc` zugewiesen. Über diese Variable kann jetzt auf die Elemente des Objektes zugegriffen werden. Anschließend wird in der `Init()` Funktion eine `listen()` Funktion aufgerufen, die die eigentliche Ereignisbehandlung übernimmt. Dort werden die jeweiligen EventListenern (Ereignis-Lauschern) [18] im SVG Dokument angehängt. Hierbei sollte die `listen()` Funktion eigentlich die Funktionsweise der verschiedenen Browser unterstützen. Leider funktioniert das Digitalisierungsprogrammmodul des Prototyps nicht im Internet Explorer. Was auf dem im Internet Explorer verwendeten Adobe SVG Viewer [19] zurückzuführen ist. Damit ein neuer Punkt aufgenommen werden kann, wird eine EventListenern für das gesamte SVG Dokument implementiert. Dieser EventListenern wartet auf das `onclick` Event und ruft dann die Funktion `addPoint()` auf. Diese Funktion übernimmt die wichtigste Aufgabe beim Hinzufügen eines neuen Punktes. Als erstes werden Koordinaten der Mouse beim Klick ermittelt. Über die `createElementNS()` Methode [7] wird ein neues `<circle>` Element an der Position des Klicks generiert. Mit der `setAttriuteNS()` Methode werden dem neuen Element seine Attribute hinzugefügt. Zu diesen Attributen zählen:

- eine Id zur eindeutigen Identifizierung des Punktes
- Koordinaten des Mittelpunktes (cx, cy)
- der Radius (r)
- style Attribute zum Füllen des Kreises mit einer Farbe

Um das neue Element im SVG Dokument einzuhängen und so auf der Graphik sichtbar werden zu lassen, wird die `appendChild()` Methode aufgerufen. Dem übergeordneten Element „`myContainer`" wird das neue Element als Kind angehängt und so auf der Graphik im Arbeitsbereich sichtbar. Im Anschluss an diesen Prozess ruft die Funktion `addPoint()` die nächste Funktion `addText()` auf. Dieser Funktion werden als Parameter die Koordinaten der Mouse Position übergeben. Die Funktion hängt als Informationen für den Sachbearbeiter die Id und Koordinaten des neuen Punktes als Text Elemente im HTML Dokument ein. Dabei werden ähnliche Methoden verwendet wie beim Hinzufügen des neuen Punktes im SVG Dokument. Zusätzlich werden die Koordinaten des Punktes, sowie im Fall von Ziel- und Wegpunkte die Punktart, in globale Arrays gespeichert. Um den Mousezeiger im Arbeitsbereich, wie in einem CAD Programm, als Fadenkreuz darzustellen, wird im SVG Dokument der Befehl `cursor="crosshair"` verwendet. Leider wird auch dieser Befehl vom Adobe SVG Viewer im Internet Explorer nicht unterstützt.

Ein weiteres `onclick` Event wird ausgelöst, sobald der Nutzer auf den Speicherbutton klickt. Über diesen Button wird die `ajax()` Funktion aufgerufen. Diese Funktion ist für die Generierung des Formulars zur Speicherung der aufgenommenen Punkte zuständig. Der Name der Funktion steht für die eingesetzte AJAX Technik [18]. Mit Hilfe der AJAX Technik werden Daten zum Server gesendet, ohne den Arbeitsbereich neu im Browser laden zu müssen. Dabei werden die global gespeicherten Koordinaten und Punktarten den `werte.php` bzw. `wegWerte.php` Scripten übergeben. Beide Scripte generieren Eingabeformulare für die Punktdaten und geben sie zurück. Die so generierten Formulare werden dann über die Funktion `outWerte()` als Popup ausgeben. Mit den Formularen erhält der Sachbearbeiter die Möglichkeit, Daten zu bearbeiten und weitere Daten einzugeben. Mit dem Senden der Formulare werden die Formulardaten über zwei PHP Scripte (`grundrissDb.php, wegpunkteDb.php`) in der Datenbank gespeichert.

Ein weiterer EventListenern überwacht das Löschen von neu hinzugefügten Punkten im Arbeitsbereich. Um einen Punkt zu löschen, muss der Sachbearbeiter auf den neu hinzugefügten Punkt klicken. Das so ausgelöste `oncklick` Event ruft die Funktion `removePoint()` auf. Diese Funktion ermittelt die Id des Punktes. Über die ermittelte Id wird der Punkt von der Grundrissgraphik im Arbeitsbereich sowie der Informationstext zu diesem Punkt im zugehörigen HTML Dokument gelöscht. Des weiterem löscht die Funktion alle Punktdaten aus den globalen Arrays.

Um die aktuelle Position der Mouse permanent dem Nutzer anzuzeigen, überwacht ein dritter EventListener das `onmousemove` Event. Dieses Event verfolgt jede Bewegung der Mouse und gibt mit Hilfe der Funktion `position()` die aktuelle Position der Mouse im HTML Dokument aus.

Alle diese Funktionen haben die Aufgabe, das Digitalisieren der Orientierungspunkte so einfach wie möglich zu gestallten. Der Arbeitsbereich zur Aufnahme von Grundrisspunkten sieht im Prototyp wie folgt aus.

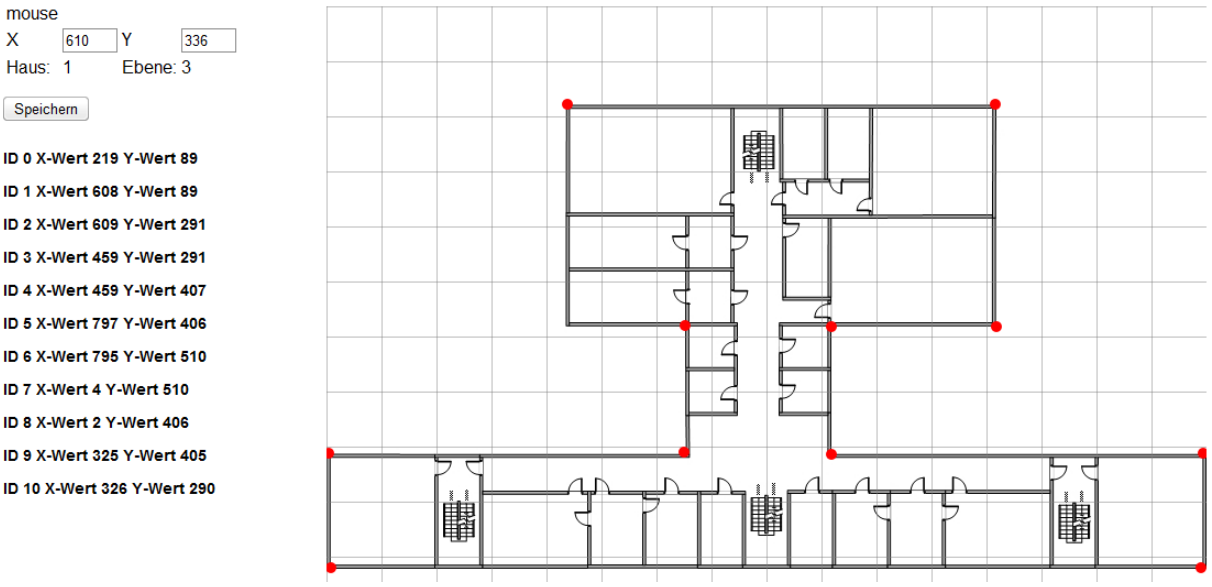


Abb. 23 Arbeitsbereich der Grundrisspunkte

Links oben wird die aktuelle Position der Mouse angezeigt. Damit der Sachbearbeiter weiß zu welchem Haus und Ebene der Grundriss gehört, werden die Werte aus der Datenbank ermittelt und angezeigt. Jeder neu aufgenommene Punkt wird im HTML Dokument auf der linken Seite mit der Id und den Koordinaten dargestellt. Angaben zur Punktart brauchen hier nicht gemacht werden. Die Art der Grundrisspunkte kann im generierten Datenformular ausgewählt werden.

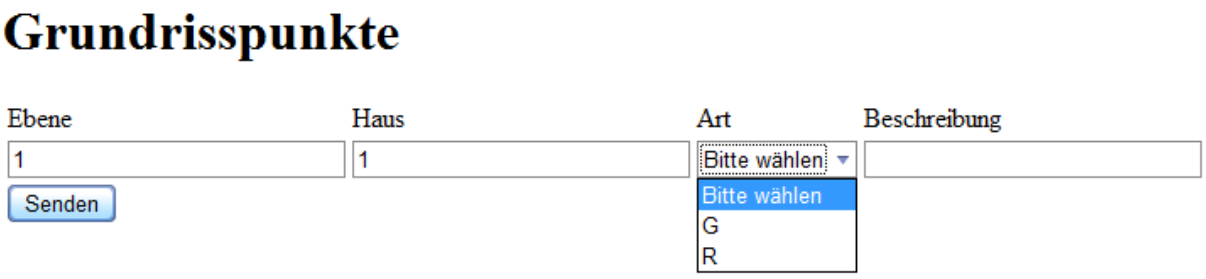


Abb. 24 Auswahl der Grundrisspunktart

Da die Grundrisspunkte als `LineString` gespeichert werden, entfällt in diesem Formular die Angabe der Koordinaten. Die Koordinaten der Grundrisspunkte werden im `werte.php` Script zu einem String zusammengefasst und über eine Sessionvariable in PHP an das `grundrissDb.php` Script übergeben.

Der Arbeitsbereich für die Aufnahme der Ziel- und Wegpunkte sieht wie folgt aus.

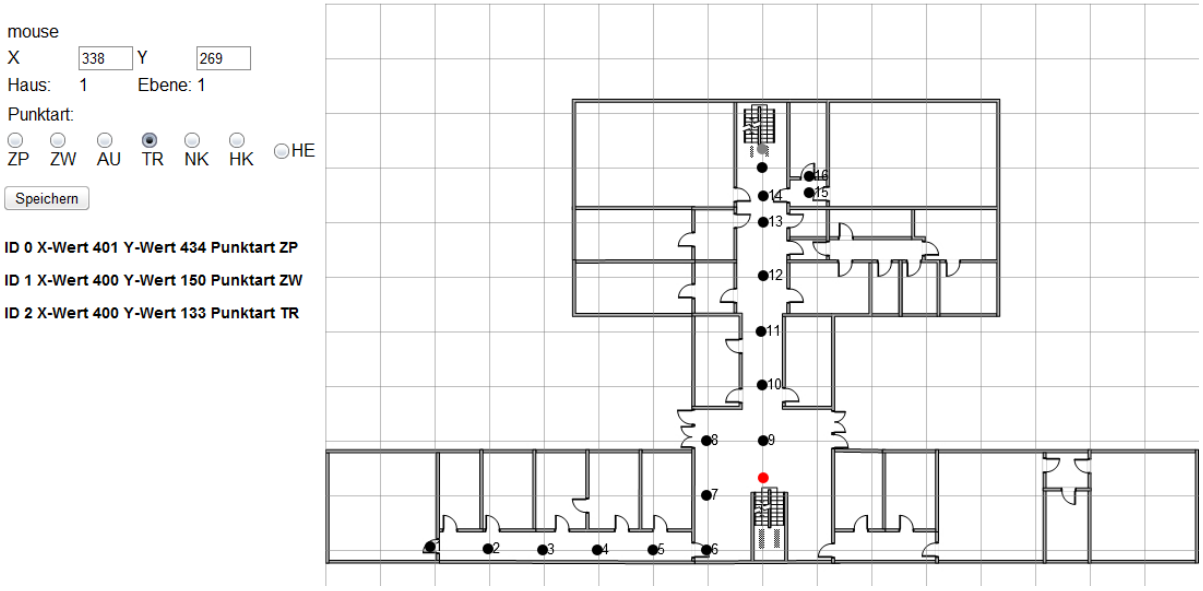


Abb. 25 Arbeitsbereich der Ziel- und Wegpunkte

Bei der Generierung der SVG Graphik zu einer Etage wird automatisch überprüft, ob schon Ziel- und Wegpunkte vorhanden sind. Sind welche vorhanden, werden diese, wie in Abb. 25 in der Graphik dargestellt. Da beim Hinzufügen von neuen Ziel- und Wegpunkten die Ermittlung nicht automatisiert wurde, erfolgt die Eingabe der Nachbarpunkte manuell. Dabei dürfen aber nicht die angezeigten Id Werte der Punkte aus dem Arbeitsbereich verwendet werden. Die Zählung der *wp_id* für die Ziel- und Wegpunkte wird durch ein auto increment erhöht. Daher müssen die *wp_id* Werte aus der Datenbank als Nachbarpunkte manuell eingegeben werden.

Wegpunkte

Haus_id	Ebene	x-Werte	y-Werte	Art	Ziel	Zuordnung	Bezeichnung1	Bezeichnung2	np1	np2	np3	np4	np5	np6
1	3	400	400	ZW	0	0	0	0	23	31	0	0	0	0
1	3	401	350	ZW	0	0	0	0	30	32	0	0	0	0
1	3	400	300	ZW	0	0	0	0	31	33	0	0	0	0
1	3	400	250	ZW	0	0	0	0	32	34	0	0	0	0
1	3	400	200	ZW	0	0	0	0	33	35	0	0	0	0
1	3	400	180	ZW	0	0	0	0	34	36	0	0	0	0
1	3	351	177	ZP	R.1.50	1.OG	Server	0	35	0	0	0	0	0

Senden

Abb. 26 Manuelle Eingabe von Ziel- und Wegpunkte

Wie in Abb. 26 ersichtlich ist, ist das Erfassen von Ziel- und Wegpunkte noch recht umständlich und muss bei einer weiteren Entwicklung automatisiert werden.

4.2.2 Suchfunktion

Die Suchfunktion wird in der `index.php` über den Startbutton gestartet. Auf der nächsten Seite, der `welse_start.php`, kann der Sachbearbeiter über entsprechende Dropdown Menüs die Startposition und das gewünschte Ziel auswählen. Die Suche wird über den Klick auf dem Suchebutton gestartet.

Bei der Suchfunktion des Prototyps wurde einzig der Suchalgorithmus aus dem bisher entwickelten WeLeS übernommen. Alle für den Suchalgorithmus relevanten Dateien konnten ohne Änderungen übernommen werden, da die Abfragen an der Tabelle WEGPUNKTE unverändert bleiben. Verändert wurde lediglich die graphische Ausgabe. Die graphische Ausgabe musste deshalb verändert werden, da die Grundrisspunkte anders aus der Datenbank abgefragt werden als bisher. Grund dafür ist der veränderte Datentyp in der Tabelle GRUNDRISS zur Speicherung der Punktdaten. Die Abfrage der Grundrisspunkte erfolgt in der Klasse `building()` (Anhang 7) mit der Methode `ground_plan()`. Alle Räume zu einem Grundriss werden mit der Funktion `room()` abgefragt. Die Methode `point_exc()` ermittelt die einzelnen Koordinaten aus dem Rückgabestring der Datenbankabfragen. Die Instanz zur Klasse `building()` wird im `svg.php` Script gebildet. Dieses Script generiert die graphische Ausgabe des Grundrisses und des Zielweges in einer SVG Datei. Eingebunden wird diese SVG Datei in der `suche.php`. Die graphische Ausgabe erfolgt stark vereinfacht, eine textliche Beschreibung, das Zoomen des Zielbereiches und das Zeichnen des Etagenwechsels wurden weggelassen. Die Darstellung des Zielweges aus Abb. 26 sieht im Prototyp folgendermaßen aus.

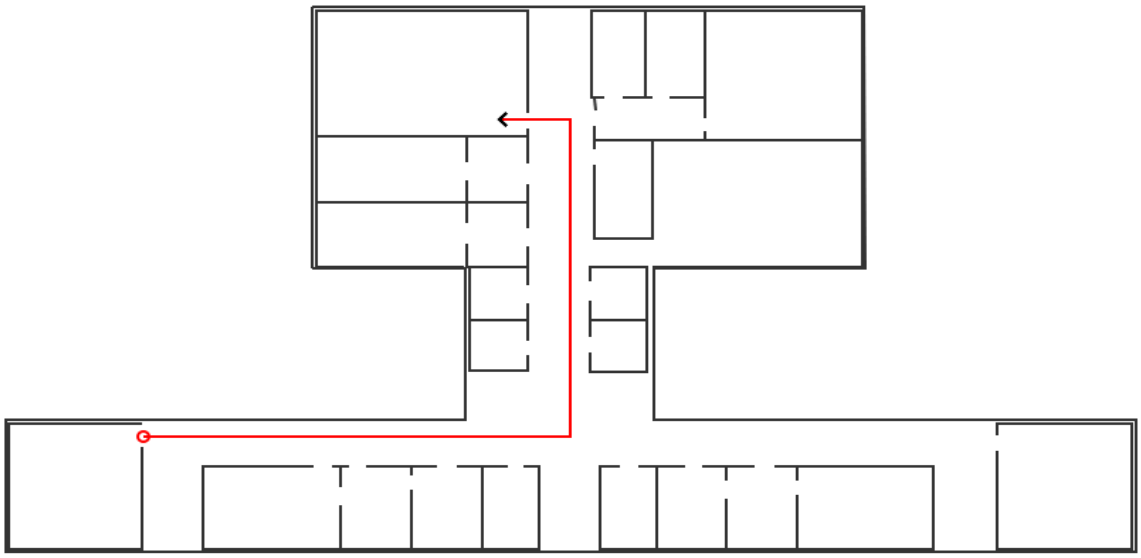


Abb. 27 Darstellung eines Zielweges im Prototyp

4.3 Fazit zum Prototyp

Ziel des Prototyps ist es, die grundlegenden Funktionen zu demonstrieren und dabei zu beschreiben, wie sie funktionieren könnten. Zusammenfassend kann gesagt werden, dass die grundlegenden Funktionen des Programmmoduls im Prototyp umgesetzt wurden. So ist es mit dem Prototyp möglich, Grundrissgraphiken hinzuzufügen, Punkte aufzunehmen und diese in der Datenbank zu speichern. Es muss hierbei berücksichtigt werden, dass die realisierten Funktionen noch nicht vollständig ausgereift sind. Die graphische Ausgabe der Suchfunktion ist stark vereinfacht. Eine Übersicht zu den Quelltextdateien kann Anhang 8 entnommen werden.

5 Fazit und Ausblick

Ziel dieser Bachelorarbeit war das Erarbeiten eines Konzeptes, um Orientierungspunkte online auf der Basis von Gebäudegrundrissen zu erfassen. Dabei wurden im Rahmen der Analyse beide schon entwickelten Wegeleitsysteme (AOWIs, WeLeS) kurz vorgestellt. Es wurde untersucht, welche Anforderungen zur Aufnahme von Orientierungspunkten an ein Programmmodul gestellt werden müssen. Bei den Anforderungen wurde zwischen einer Weiterentwicklung des WeLeS und einer existierenden Programmlösung unterschieden. Wie die Analyse zeigt, ist es vorteilhaft, das bisherige WeLeS ohne das Einbinden von existierenden Programmen weiterzuentwickeln. Die Analyse zeigt auch, dass einige Änderungen am WeLeS vorgenommen werden müssen. Diese Änderungen beziehen sich in erster Line auf die Datenbank. Auf der Grundlage der Analyse folgte im Anschluss die Erarbeitung des Konzeptes. Hier wurden einige Änderungen, die für das Programmmodul wichtig sind, aufgenommen. Das Konzept beinhaltet 3 Schwerpunkte:

1. Verwaltung der Grundrissdaten
2. Aufbau des Arbeitsbereichs
3. Speicherung der aufgenommenen Punkte

Während der Erarbeitung wurde geklärt, wie diese Schwerpunkte realisiert werden können. Im Anschluss des Konzeptes wurden die grundlegenden Funktionen mit Hilfe eines Prototyps demonstriert.

5.1 Verbesserungen

Für die Unterscheidung zwischen Besucher, Sachbearbeiter und Administrator wäre eine Benutzerverwaltung notwendig. Denkbar wäre ein Link auf der Startseite, der auf eine Anmeldemaske verweist. Über die Anmeldung gelangen der Sachbearbeiter und der Administrator in den Backendbereich. Im Backendbereich werden alle Funktionen für die Wartung und Pflege des WeLeS bereitgestellt. Dazu zählen zum einen die Verwaltung der Daten in der Datenbank, und zum anderen die Digitalisierungsfunktion.
Die Startseite der Digitalisierungsfunktion könnte genauso aussehen wie im Prototyp. Dabei wird zwischen der Aufnahme von Grundrisspunkten sowie Ziel- und Wegpunkte unterschieden. Die Auswahl der Grundrissgraphikdatei erfolgt dabei wie im Prototyp über ein Dropdown Menü. Der Arbeitsbereich könnte noch übersichtlicher gestaltet werden sowie weitere Funktionen wie das Zoomen und Pan implementiert werden.
Wichtig bei der Aufnahme von Ziel- und Wegpunkten ist die automatisierte Ermittlung der Nachbarpunkte. Diese ermöglicht dem Sachbearbeiter ein einfaches Hinzufügen und Speichern von neuen Ziel- und Wegpunkten.
Zu verbessern wäre die graphische Ausgabe, da diese sich im Prototyp auf das nötigste beschränkt. Hierbei sollte der Ansatz aus der bisherigen Entwicklung (Beschreibung des Weges, das Vergrößern des Zieles sowie die einheitlich und vereinfachte Darstellung des Grundrisses) mit der im Prototyp umgesetzten Änderungen der Datenbank (veränderte Datentypen), der Datenabfrage und graphischen Generierung der Polylines kombiniert werden. Insbesondere ist hier die Umsetzung der graphischen Ausgabe bei einer Suchabfrage über mehrere Etagen und Häuser zu erwähnen, die für die Entwicklung des Prototyps nicht relevant war. Im Prototyp werden für jede Etage die Grundrisspunkte abgefragt und gezeichnet. Vorteilhaft wäre es den Grundriss eines Hauses nur einmal aufzunehmen. Alle Räume einer Etage könnten dann innerhalb des Grundrisses gezeichnet werden. Bei einem Zielweg über mehrere Etagen muss unterschieden werden, welche Räume gezeichnet werden. Dieses sollte aber nicht über ein zusätzliches Attribut, ähnlich dem *LoD* Attribut der bisherigen Datenbank, gelöst werden. Denkbar wäre es, im Vorfeld zu überprüfen, ob der Zielweg über mehrere Etagen verläuft. Ist das der Fall, könnte mit einer entsprechenden Schraffur des Zielweges und Symbole für Treppen und Fahrstühle ein Etagenwechsel angezeigt werden.
Bei einer Weiterentwicklung sollte außerdem untersucht werden, inwieweit die Zielwege gewichtet werden können. Das bedeutet, dass Personalwege und Wege für die Öffentlichkeit mit unterschieden werden können. Dies könnte mit einem zusätzlichen Attribut in der Tabelle WEGPUNKTE gelöst werden. Mögliche Werte wären Ö (öffentlich) und P (Personal).
Die Stringverarbeitung zur Ermittlung der Koordinaten ist im Prototyp recht umständlich gelöst. Für die weitere Entwicklung wäre es besser, aus dem kompletten Rückgabestring der

Datenbankabfrage alle nicht benötigten Zeichen mit Hilfe der `explode` Funktion zu entfernen. Ein einfaches Beispiel befindet sich in der `Building.php` als Kommentar.
Damit bei einer Mehrfachnutzung der Suchfunktion die Ausgabedateien nicht gegenseitig überschrieben werden, könnte ein Zeitstempel (Ymdhis) dem Dateiname hinzugefügt werden. Diese Dateien sollten nur temporär auf dem Webserver gespeichert werden. Für einen übersichtlichen Quelltext und eine objektorientierte Programmierung der Datenbankabfragen könnte die mysqli Schnittstelle im WeLeS zum Einsatz kommen.
Probleme gibt es bei den Funktionen des Arbeitsbereiches in den verschiedenen Browsern. Im Internet Explorer 8 wird der Grundriss zwar dargestellt, aber die JavaScript Funktionen zum Aufnehmen eines neuen Punktes werden nicht ausgeführt. Im Firefox 3.5.3 funktioniert zwar das Aufnehmen der Punkte, aber nicht das komplette Löschen der Punkte. Beim Löschen verschwindet zwar der Punkt auf der Graphik, aber er wird nicht aus dem Zwischenspeicher gelöscht, und im HTML Dokument bleiben die Informationen zu dem Punkt erhalten. Der einzige Browser, in dem alle Funktionen des Prototyps funktionieren, ist Opera 10.0. Das bedeutet für eine Weiterentwicklung, dass alle JavaScript Funktionen auf die einzelnen Browser abgestimmt werden müssen.

Zum Abschluss kann gesagt werden, dass diese Bachelorarbeit ein wichtiger Schritt in Richtung Weiterentwicklung des WeLeS ist. Bis das WeLeS eine vollfunktionstüchtige Webapplikation ist, sind noch einige Entwicklungsschritte notwendig. Der nächste Schritt könnte darin bestehen, das erarbeitete Konzept des Programmmoduls zur Aufnahme von Orientierungspunkten vollständig im Gesamtkonzept des WeLeS zu integrieren.

Glossar

AJAX	**A**synchronous **J**avaScript **a**nd **X**ML asynchrone Datenübertragung zwischen Client und Server, um die Webseite nicht neu laden zu müssen
CAD	**C**omputer **A**ided **D**esign
CMS	Content Management System
CSS	Stylesheets dient der Formatierung einzelner HTML Elemente
GML	**G**eography **M**arkup **L**anguage
http	**H**yper**t**ext **T**ransfer **P**rotokoll regelt die Kommunikation zwischen Browser und Client
JavaScript	clientseitige Programmiersprache für HTML Seiten
JPEG	**J**oint **P**hotographic **E**xperts **G**roup Rastergraphikformat
MySQL	quelloffenes Datenbankmanagementsystem (MySQL Referenzhandbuch 1.4)
OGC	**O**pen **G**eospatial **C**onsortium http://www.opengeospatial.org/
PHP	Hypertext Preprocessor Programmiersprache für die Entwicklung von dynamischen Webseiten
WFS	**W**eb **F**eature **S**ervice
WMS	**W**eb **M**ap **S**ervice
Xamppe	Zusammenstellung freier Software (www.apachefriends.org)

Quellenverzeichnis

[1] Entwicklung von Auskunfts-, Orientierungs- und Wegeleitsystemen- Wartung, Laufenhaltung und Weiterentwicklungen; Stefan Martin 2005

[2] GRUNDPRINZIPIEN DES AUFBAUES VON INTERAKTIVEN WEBBASIERTEN AUSKUNFTS-, ORIENTIERUNGS- UND WEGELEITSYSTEMEN; Eberhard Schmiedel 2006

[3] Grundlagen der Geo-Informationssysteme Band 2; R. Bill 2. Auflage 1999 Herbert Wichmann Verlag

[4] Beleg im Fach B126 Informationsmanagement Thematik: Wegeleitsystem; Uwe Kickstein, Max Noja, Sebastian Nelson 2008

[5] Jetzt lerne ich Webseiten programmieren und gestalten; Christian Wenz, Tobias Hauser; Markt und Technik 2004

[6] PHP 5.3 & MySQL 5.1 Grundlage, Programmiertechniken, Beispiele; Michael Kofler, Bernd Öggl; ADDISON-WESLEY 2008

[7] Visualisieren von Geodaten mit SVG im Internet Band 1; Nicole Ueberschär/ Andre M. Winter 2006, Herbert Wichmann Verlag

[8] Lexikon der Geoinformatik; Ralf Bill/ Marco L. Zehner; Herbert Wichmann Verlag 2001

[9] Geodatenbanksysteme in Theorie und Praxis; Thomas Brinkhoff; 2. überarbeitete und erweiterte Auflage 2008; Herbert WichmannVerlag

[10] Grundlagen der Geo-Informationssysteme Band 1; R. Bill 4. Auflage Herbert Wichmann Verlag

[11] www.mapbender.org (Zugriff: 08.10.2009)

[12] Mapbender Dokumentation; Stand: 31.10.2008; http://www.mapbender.org/Tutorials#Documentation (Zugriff: 08.10.2009)

[13] http://www.opengeospatial.org/standards/wms (Zugriff: 08.10.2009)

[14] http://www.opengeospatial.org/standards/wfs (Zugriff: 08.10.2009)

[15] TCP/IP Das bhv Taschenbuch; Thomas Zwolsky;1. Auflage 2004

[16] http://www.apachefriends.org/de/xampp.html (Zugriff: 08.10.2009)

[17] SELFHTML: Version 8.1.2 vom 01.03.2007; Stefan Münz

[18] JavaScript und AJAX von Christian Wenz Das umfassende Handbuch; Galileo Computing – <openbook> 2007

[19] http://www.adobe.com/svg/viewer/install/mainframed.html (Zugriff: 08.10.2009)

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Anhang 1 – Erfassung der Punkte im AOWIs

Im folgendem werden die Schritte aus der Diplomarbeit[1] zitiert, die erforderlich sind, um Koordinaten für das AOWIS zu generieren. Die Schritte stammen aus dem Kapitel 4.2. ***„Die innovative Lösung: Generierung von Koordinaten für SVG“***

Dort wurden folgende Schritte beschrieben:

„Nachfolgend wird beschrieben, welche neue Lösung erarbeitet wurde, um Koordinaten für die grafische Darstellung von Gebäudegrundrissen und für Wegpunkte zu gewinnen.

Die Grundlage von Koordinaten in SVG ist ein rechtwinkliges regelmäßiges zweidimensionales Koordinatensystem, dessen Ausgangspunkt mit den Koordinaten 0,0 in der linken oberen Ecke platziert ist. Die x-Achse erstreckt sich horizontal nach rechts, die y-Achse vertikal nach unten.
Im Bezug auf das Auskunfts-, Orientierungs- und Wegeleitsystem sollten zur Vereinfachung, im Unterschied zu einem regulären kartesischen Koordinatensystem mit vier Quadranten, nur positive Werte für Koordinaten vergeben werden. Dies hat beispielsweise Auswirkungen auf eine Anpassung an CAD-Systeme.

Die Maßeinheiten des Koordinatensystems können Millimeter (mm), Zentimeter (cm), inch (Zoll) und Pixel (px) sein.
Die Arbeit mit Millimetern ist besonders günstig in Hinblick auf künftige Papierausdrucke.
Ein Werkzeug (Software), welches die Koordinaten beispielsweise von Endpunkten der Linien oder Eckpunkten von Polygonen ausgibt, wurde nicht gefunden.[1]
Für Wegpunkte wurde mittels JavaScript ein Anzeigeelement selbst geschaffen.

Da jedoch, wie später zu erläutern ist, in jedem Fall eine Grundrisszeichnung des betreffenden Komplexes erforderlich ist, wurde ein anderer Weg neu entwickelt.

- **Schritt 1**

Im ersten Schritt wird eine vollständige Grundrisszeichnung der Gebäude, Einrichtungen, inneren Strukturen der Gebäude und der Außenanlagen auf der Grundlage von Bauplänen

[1] Vgl. Anhang A.1 „Authoring Tools“

und / oder anderen Karten in einem CAD-System digitalisiert, welches eine Exportfunktion in das jeweils aktuelle DXF-Format besitzt.
Zu diesem Zweck wurde ein älteres AutoCAD-Release benutzt. Es kann aber auch eine beliebige andere Version verwendet werden.

Der Umfang der darzustellenden Objekte richtet sich nach der Zielstellung des Projektes und soll keinesfalls einen Bauplan ersetzen.
Es ist möglich, aber nicht erforderlich, jeden Raum oder nicht für die Öffentlichkeit bestimmte Aufzüge darzustellen. Ebenso nicht die innere Struktur von geschlossenen Abteilungen, die durch die Öffentlichkeit nicht betreten werden sollen.
Ggf. ist ein Zeichnung der Umgebung des Geländes hinsichtlich der Straßen- und Parkplatzorganisation erforderlich.

Eine maßstabsgerechte Darstellung ist nicht (unbedingt) erforderlich, da es in einem solchen Projekt nicht um die Optimierung von Wegen in einem Routing oder um Entfernungsmessungen, sondern (mit Ausnahmen[2]) um die stilisierte Darstellung von vorgegebenen Wegen geht.
Sollte sich der Gebäudekomplex vertikal über Ebenen erstrecken, könnten die Zeichnungen für die einzelnen Ebenen aus der Grundzeichnung durch Kopie und Nachbearbeitung der inneren Strukturen abgeleitet werden.
Auch hier sollten Objekte, die nicht für die Öffentlichkeit bestimmt sind, fortgelassen werden, also auch Gebäude des Gesamtkomplexes, die diese Ebene nicht haben.

- **Schritt 2**

Über diese digitalisierte Grundrisszeichnung ist ein regelmäßiges sichtbares Raster zu legen (Vgl. Anhang B.1 Abbildung B.1.2).

- Berechnung des Rasters

Im Hinblick auf günstige Dezimalwerte für Koordinatenangaben in SVG bei einem Ausdruckformat auf A4 im Querformat sollten die Zeicheneinheiten für die Querausdehnung der Zeichnung durch den Wert 56 und für die Längsausdehnung durch den Wert 42 geteilt werden.[3]
Ein angemessener oder geforderter Rand sollte außerdem vorgesehen werden.

[2] Vgl. Kapitel 8 „Weiterentwicklungen“
[3] Blattgröße 297 * 210 Millimeter durch (spätere) 5 Millimeter Rastergröße / bedeutet auch: Verhältnis 4:3

Aus diesen vorläufigen Werten wird sicherlich ein Kompromiss zu finden sein, der das Seitenverhältnis 4:3 repräsentiert.
Der Abstand der Rasterlinien, in Zeicheneinheiten, ist danach zu berechnen.

Wenn das Raster über der Zeichnung vorhanden ist und dieses den gesamten Komplex einschließlich geforderten Druckrand abdeckt, muss jetzt eine Verhältniszahl zwischen dem gezeichneten Rasterabstand und der Zahl 5 errechnet werden.[4]
Nach diesem Verhältnis ist die gesamte Zeichnung (alle Layer eingeschaltet!) einheitlich in x- und y-Richtung zu variieren (ausdehnen bzw. schrumpfen). Daraus resultiert, dass der Rasterabstand 5 Zeicheneinheiten beträgt.
Der Punkt 0,0 der gesamtem Zeichnung einschließlich dieses Rasters muss die linke obere Ecke der Gesamtzeichnung sein.
Die Zeichnung ist spätestens nach der Überlagerung durch das Koordinatensystem mit dem Abstand 5 Zeicheneinheiten dahingehend zu translatieren.
Damit sind in dieser Zeichnung die Koordinatenwerte für x und y z.B. von Gebäudeecken, Wegpunkten u.s.w. in 5-er Schritten (auf dem Raster) bzw. 2,5-er Schritten (zwischen den Rasterlinien) ablesbar, wenn Schritt 3 ausgeführt wurde.
Zu beachten ist, dass aufgrund der eingangs beschriebenen Form des Koordinatensystems bei der Darstellung in SVG beim y-Wert, in AutoCAD, das negative Vorzeichen fortzulassen ist.

Dies ist die Grundlage für die Datenbank, aus der, wie vorstehend angedeutet, die Linien- und Polygonelemente der Grafik in SVG-Format gelesen und dargestellt werden können.

- **Schritt 3**

In der Zeichnung mit dem Raster 5 Zeichnungseinheiten kann jetzt eine Generalisierung / Stilisierung der Gebäudeumrisse, Ecken des benötigten Innenausbaues, Orte von Fahrstühlen sowie Treppenhäusern usw. durch Verlegung der Schnittpunkte auf die Rasterpunkte oder (in die Mitte) zwischen diesen vorgenommen werden.
Damit wird erreicht, dass benötigte Koordinaten für die SVG in Integer-Werten bzw. in Dezimalwerten mit höchstens einer Nachkommastelle (x,5) vorliegen.
Diese Art der Generalisierung ist zur Koordinatenbestimmung und Eintrag in die Datenbank günstiger, da übersichtlich und sicherer gegen Schreibfehler.

[4] für ein gewähltes Raster 5 * 5 Millimeter für z.B. Dietrich-Bonhoeffer-Klinikum, Neubrandenburg

Diese Zeichnung ist demzufolge die Grundlage zur Ableitung der Werte für die Datenbank hinsichtlich

- Koordinaten für Gebäudeecken,
- Koordinaten für Wegpunkte innerhalb und außerhalb von Gebäuden,
- Koordinaten sonstiger notwendiger Punkte, die in Darstellungen benötigt werden, wie Aufzüge, Treppenhäuser, Patientenwartezonen, Behandlungsräume, Sprechzimmer, Probenannahmen etc.

Die ID in der Datenbank für die Gebäudeumringe und Begrenzungen der inneren Flure eines Gebäudes sind streng in der Reihenfolge der Punkte im Uhrzeigersinn festzulegen, damit für die Punktkoordinaten in der SVG-Programmierung die richtige Zeichenreihenfolge eingehalten wird.

Damit sind zwei Darstellungen vorhanden:

- eine an Baupläne und andere Kartengrundlagen angelehnte Zeichnung der wesentlichsten Elemente des Komplexes; ggf. mit äußerer Umgebung zur Übersicht
- eine Zeichnung, welche die Punkte und deren (ablesbare) Koordinaten enthält, die dynamisch oder statisch in SVG-Objekte umgewandelt werden sollen.

Um spätere Aktualisierungen zu ersparen, müssen die eingetragenen Punkte (x- und y-Koordinaten) so vollständig wie nötig sein, d.h.: diese Arbeit wird nur einmal vorgenommen (Datensammlung).
Veränderungen ergeben sich nur bei Neu- und entscheidenden Umbauten. Vorgenannte Schritte sind im Stadium der Datenaufnahme für ein Projekt angesiedelt.

- **Konvertierung von CAD-Zeichnungen in SVG**

Wie bereits erwähnt, ist eine Neuzeichnung von Objekten und Komplexen für das AOWIs mit den recherchierten SVG-Generatoren oder Authoring-Tools (Vgl. Anhang A.1) nicht machbar, da es keine Möglichkeit der grafischen Digitalisierung und koordinatengerechter grafischer Eingabe von primitiven Zeichnungselementen gibt.

Das Konvertieren von vorhandenen Zeichnungen in das SVG-Format ist jedoch mit dem professionellen Grafikprogramm CorelDraw ab Version 10 möglich.
CorelDraw verfügt einerseits über Importfunktionen aller gängigen Grafikformate der Linien pixel- und vektororientiert, u.a. auch für das AutoCAD-Format DWG und das AutoCAD-

Dateiaustauschformat DXF. Dreidimensionale Konstruktionen werden allerdings nicht dreidimensional wiedergegeben.
Andererseits schreibt CorelDraw SVG-Grafik-Quelltexte mit seiner implementierten Exportfunktion.

Bei diesem Export gilt es jedoch Folgendes zu beachten:

- Es ist nur sinnvoll, vektororientierte Zeichnungen zu konvertieren. Pixelbilder werden 1:1 übernommen und nur in Quelltextzeilen eingebettet. Es kann zwischen den Formaten JPEG, GIF und PNG (den „Internet-Grafikformaten“) gewählt werden.
- Die entstehenden SVG-Quelltexte sind, wie bei Programmgeneratoren typisch, sehr umfangreich und schwer zu analysieren; demzufolge schwierig zu korrigieren bzw. anzupassen. Beispielsweise werden geometrische primitive Elemente (Linien, Polylinien, Polygone) aus dem grundlegendsten SVG-Element path[5] kompliziert zusammengesetzt, obwohl es eindeutig definierte einfache Schreibweisen zur Definition der Elemente line, polyline und rect sowie polygon gibt. Eindeutig wären nur Kreise (circle mit Koordinate für den Mittelpunkt und einem Radius) in einem Quelltext zu notieren. Jedoch werden durch den Generator Kreise ebenfalls aus dem Grundelement path kompliziert zusammengesetzt.
- Dimensionslose Punkte werden nicht dargestellt.
- Im Computer nicht installierte Schriftarten werden in „Glyphen“ umgesetzt und ebenfalls durch das Grundelement path als vektorielle Kurven übersetzt[6]. Im Quelltext sind derartige Elemente nicht als Text zu erkennen. Deshalb wird empfohlen, die Standardschriftarten Arial und Times New Roman zu benutzen, die in jedem Computer mit Microsoft Betriebssystemen installiert sein dürften.
- CorelDraw Version 10 benutzt beim Export als Koordinatenwerte inch (Zoll).
  Die CorelDraw Versionen 11 und 12 arbeiten mit dem (durchschaubaren) Wert Millimeter. Demzufolge ist keine Umrechnung in den Quelltexten erforderlich, wenn auf das deutsche Druckformat Bezug genommen werden soll.[7]

Es ist also möglich, die o.a. AutoCAD-Übersichtszeichnung komplett in einen SVG-Quelltext zu konvertieren, wenn nachstehende Schritte und Bedingungen eingehalten werden:

- Konzentrierung aller später in SVG darzustellenden Objekte im AutoCAD-Layer 0 (Farben für die Objekte können beibehalten werden, wenn mit konkreter Farbe gezeichnet wurde oder die Farbe geändert wurde; also nicht: Farbe von Layer)

[5] Vgl. Anhang A.1
[6] diese „Texte“ werden im WWW tatsächlich als Texte interpretiert
[7] Wenn eine neue CorelDraw-Suite erscheint, wird die Vorgängerversion als Freeware freigegeben und z.B. zu neu gekauften Computern kostenlos beigegeben. Demzufolge ist die Version 11 gegenwärtig kostenlos einsetzbar.

- Löschen aller nicht benötigten Objekte
- Löschung der damit überflüssigen Layer
- Generierung einer neuen Zeichnung in CorelDraw mit dem gewünschten Papierformat für den Druck (DIN-Größe und Hoch- bzw. Querformat)
- Import der AutoCAD-Zeichnung
- proportionales „Ziehen“ in die CorelDraw-Vorlage, bis die gewünschte Dimension erreicht ist
- Exportfunktion in SVG-Format.

Wenn unter Einhaltung dieser Schritte die beschriebene AutoCAD-Zeichnung mit dem Beispielrasterabstand 5 Zeichnungseinheiten verwendet wird und in CorelDraw auf eine Vorlage mit Raster 5 deckungsgleich „gezogen“ wird, sind in SVG die in AutoCAD ermittelten Koordinaten gültig.
Zu empfehlen ist, einen charakteristischen Punkt (Gebäudeecke, extra gezeichnetes Kontrollelement - „Passpunkt“) zu haben, der auf den entsprechenden Gitterpunkt von CorelDraw „gezogen“ wird.

Eine Konvertierung von CAD-Zeichnungen in das SVG-Format wird nur empfohlen, um z.B. eine komplette Zeichnung in SVG darzustellen, die nicht wesentlich korrigiert werden muss. Auf die Auswirkungen von Konvertierungen auf die SVG-Quelltexte wurden oben ausführlich hingewiesen.“

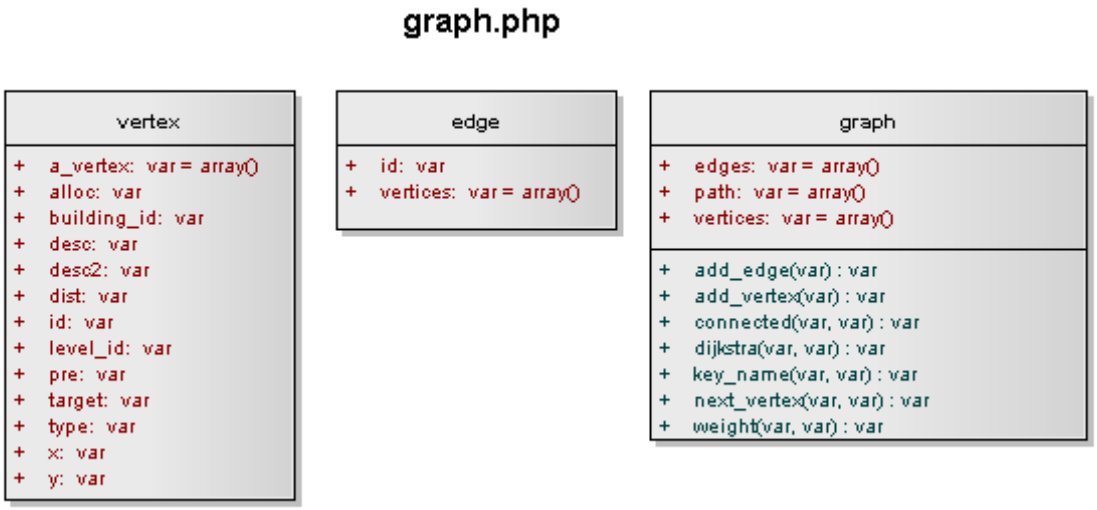
Anhang 2 – Der Dijkstra Algorithmus im WeLeS

Dieser Algorithmus ist einer der bekanntesten bei der Berechnung des kürzesten Weges. Eine Beschreibung zur Funktionsweise des Algorithmus findet sich unter [3] auf der Seite 30 sowie unter [8] auf der Seite 360.
Im WeLeS hat der Algorithmus folgende Aufgabe[4]:

„Der Suchalgorithmus soll dem System ermöglichen, dem Benutzer den kürzesten Weg zwischen 2 frei wählbaren Orten zu liefern.
Für einen Suchalgorithmus wird meistens ein geschlossener Graph benötigt, auf dem sich die dazugehörigen Knoten befinden. Um einen Graphen zu erzeugen, werden die Wegpunkte aus der Datenbank genutzt. Allerdings können die Wegpunkte in ihrer vorliegenden Form noch nicht für den Aufbau eines Graphen benutzt werden, da keine Informationen zu Nachbarschaftsbeziehungen zwischen den Punkten vorhanden sind. Diese müssen der Tabelle Wegpunkte noch in Form von Nachbarpunkten hinzugefügt werden. Die Umsetzung des Suchalgorithmus erfolgt mit PHP. Dabei wird eine objektorientierte Modellierung der Elemente angestrebt."

Realisiert wurde der Algorithmus im WeLeS wie folgt[4]:
„Die Umsetzung des Graphen und dem damit verbundenen Suchalgorithmus erfolgt mit dem PHP – Skript **graph.php**. Dieses enthält 3 Klassen zur Abbildung eines geschlossenen Graphen (**graph**), der Knoten auf dem Graphen (**vertex**), sowie der Kanten eines Graphen (**edge**).
Um dem Benutzer die Möglichkeit zu geben, auf seinem Weg nur Fahrstühle zu benutzen, falls die Etage gewechselt werden muss, gibt es zusätzlich noch das Skript **graph_no_stairs.php**. Dies ist eine abgewandelte Form des **graph.php** Skripts, bei dem Punkte vom Typ Treppe nicht für die Suche berücksichtigt werden.

Klassendiagramm des Graphen



Diese Darstellung zeigt die Klassen des **graph.php** Skripts.
Die Knoten des Graphen werden mit der Klasse **vertex** abgebildet. Dabei sind alle Eigenschaften der Wegpunkte als Attribute vorhanden. Funktionen besitzt diese Klasse nicht.
Die Klasse **edge** bildet eine Kante eines Graphen ab. Die Attribute sind eher rudimentär, da die Klasse für den Suchalgorithmus nicht genutzt wird und mehr als Vorlage für Weiterentwicklungen dienen soll.
Die Klasse **graph** besitzt als Attribute nur 3 Arrays zur Aufnahme von Datenelementen. Wichtiger sind hier die Funktionen der Klasse. Es gibt 2 Funktionen zur Datenaufnahme, sowie die Suchfunktion, und einige Funktionen, die innerhalb der Suchfunktion genutzt werden.
Die Suchfunktion basiert auf dem Algorithmus von Dijkstra und ist als Funktion in die Klasse **graph** integriert. Dieser Algorithmus findet den kürzesten Weg von einem Startpunkt zu einem beliebigen Punkt auf einem geschlossenen, kantengewichteten Graphen. Die Gewichte dürfen dabei nicht negativ sein. In der Klasse **graph** sind die Arrays **vertices** und **edges** enthalten, um die Knoten bzw. Kanten des Graphen aufzunehmen. Es wird davon ausgegangen, dass die Gesamtheit der Knoten bzw. Kanten einen geschlossenen Graphen ergeben. Das Array **path** speichert die Knoten des kürzesten Pfades. Um dem Graphen Knoten oder Kanten hinzuzufügen, gibt es die Funktionen **add_vertex(vertex)** und **add_edge(edge)**, die Objekte der Klassen **vertex** bzw. **edge** als Parameter übernehmen.
Die Klasse **vertex** nimmt die Wegpunkte aus der Datenbank auf. Jedes Objekt enthält alle Informationen über den Wegpunkt, sowie die Variablen **dist** und **pre**, die für den Suchalgorithmus gebraucht werden. Die Nachbarpunkte jedes Wegpunktes werden im Array **a_vertex** gespeichert.

Die Klasse **edge** nimmt 2 Objekte vom Typ **vertex** auf und bildet damit eine Kante des Graphen. Diese Klasse ist nur vorhanden, um eventuell eine Suche auf Basis von Kanten zu ermöglichen.“

Anhang 3 – MySQL Tabellentypen

MySQL unterscheidet verschiedene Tabellentypen.

Tabellentyp	Eigenschaften
MyISAM	• Standard Tabellentyp in MySQL • dient der Verwaltung von nicht-transaktionssichere Tabellen • Daten können schnell gespeichert und Abgerufen werden • bietet die Möglichkeit der Volltextsuche • unterstützt keine Foreigen-Key-Regeln
InnoDB	• unterstützt Transaktionen • unterstützt Foreigen-Key-Regeln • keine Volltextsuche
MEMORY	• Tabellen werden im Arbeitsspeicher • sehr schneller Zugriff auf Datensätze • Verlust der Daten wenn MySQL beendet wird • nicht-transaktionssichere
MERGE	• mehrere identische MyISAM- Tabellen können wie eine einzige Tabelle behandelt werden • nicht-transaktionssichere

Eine komplette Ausführung zu den Tabellentypen kann den offiziellen MySQL 5.1. Referenz-handbuch entnommen werden. Siehe dazu folgenden Link: http://dev.mysql.com/doc/refman/5.1/de/index.html .

Anhang 4 – LineString Funktionen

Übersicht zu den LineString Funktionen in MySQL

Funktionsname	Beschreibung
`EndPoint()`	Gibt den letzten Punkt eines LineStrings zurück
`GLength()`	Ermittelt die Länge eines LineStrings
`NumPoints()`	Gibt die Anzahl der im LineString enthaltenen Punkte zurück
`PointN(`*`LS`*`,n)`	Ermittelt den Punkt an der Position `n`
`StartPoint()`	Gibt den 1. Punkt des LineString zurück
`IsClosed()`	Prüft ob Startpunkt gleich Endpunkt ist

Mehr Information und Beispiel finden sich im MySQL 5.1. Referenzhandbuch.

Anhang 5 – Quelltext der externen JavaScript Dateien

grundriss.js

```
var SVGDoc = null;
var z = 0;
var p = 0;
var x_werte = new Array();
var y_werte = new Array();
var p_art = new Array();
var http = null;
var h_id = 0;
var eb_id = 0;
	/*
	* Funktion übergibt die Koordinaten aller Punkte per POST an werte.php
	* PHP Script erstellt das Formular
	*/
	function ajax()
	{

		if (window.XMLHttpRequest)
		{
			http = new XMLHttpRequest();
		} else if (window.ActiveXObject)
		{
			http = new ActiveXObject("Microsoft.XMLHTTP");
		}
		if(http !=null)
		{
			hid  = document.getElementById("hid").value;
			ebid = document.getElementById("ebid").value;
			var parameter =
"x_werte="+x_werte+"&y_werte="+y_werte+"&hid="+hid+"&ebid="+ebid;
			http.open("POST", "werte.php", true);
			http.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
			http.send(parameter);
			http.onreadystatechange = outWerte;
		}
	}
	/*
	* Ausgabe des in der Ajax Funktion erstellten Formulars
	*/
	function outWerte()
	{
		if (http.readyState == 4)
		{
		var fenster = window.open("","Punktaufnehmen", "width=800,height=600" );
		fenster.document.open();
		fenster.document.write(http.responseText);
		}
	}
	/*
	* Funktion überwacht die aktuelle Position der Mouse
	* Position wird im HTML Dokument ausgeben
	*/
	function position(evt)
	{
		 document.getElementById("positionX").value = evt.clientX;
		 document.getElementById("positionY").value = evt.clientY;
	}
	/*
	* Hinzufügen der Punkte im HTML Dokument, zu besseren Orientierung
	*/
	function addText(px,py)
	{
		var x=px;
		var y=py;
		// Runden der Koordinaten
		newX = Math.round(x);
		newY = Math.round(y);
		var out = document.getElementById("neu");
		var myContent = document.createElement("h3");
		myContent.setAttribute("id",z);
		out.appendChild(myContent);
		var myID = document.createTextNode(" ID ");
```

```
                myContent.appendChild(myID);
                out.appendChild(myContent);
                var myID1 = document.createTextNode(z);
                myContent.appendChild(myID1);
                out.appendChild(myContent);
                var myText3 = document.createTextNode(" X-Wert ");
                myContent.appendChild(myText3);
                out.appendChild(myContent);
                var myText4 = document.createTextNode(newX);
                myContent.appendChild(myText4);
                out.appendChild(myContent);
                var myText5 = document.createTextNode(" Y-Wert ");
                myContent.appendChild(myText5);
                out.appendChild(myContent);
                var myText6 = document.createTextNode(newY);
                myContent.appendChild(myText6);
                out.appendChild(myContent);
                // Speichern der Koordinaten im jeweiligen globalen Array
                x_werte.push(newX);
                y_werte.push(newY);
                // ID Zähler wird erhöht
                z++;
        }

        /*
        * Löschen des in SVG erzeugten Punktes
        * Zugriff erfolgt über die entsprechende ID
        */
        function removePoint(evt)
        {
                var obj = evt.target;
                var  Att_id = evt.target.getAttributeNS(null,"id");
                var remove = obj.parentNode.removeChild(obj);
                alert("ID: "+Att_id);
                var obj = document.getElementById(Att_id);
                var k = document.getElementById(Att_id).removeNode(true);
                // Löschen der Werte aus dem x und y Array
                x_werte.splice(Att_id,1,"re");
                y_werte.splice(Att_id,1,"re");
        }
        /* Hinzufügen neuer Punkte auf der Graphik
        * Ermitteln der Mouse Position
        * an dieser Stelle wird ein Kreis mit dem Radius 5 erstellt
        * Position wird an die Funktion addText weitergeben
        * Anschließend wird der ID Zähler (p) um ein erhöht
        */
        function addPoint(evt)
        {
                var newSVGOb = "http://www.w3.org/2000/svg";
                var newCircle =document.createElementNS(newSVGOb,"circle");
                newCircle.setAttributeNS(null,"id",p);
                newCircle.setAttributeNS(null,"cx",evt.clientX);
                newCircle.setAttributeNS(null,"cy",evt.clientY);
                newCircle.setAttributeNS(null,"r",5);
                newCircle.setAttributeNS(null,"fill","red");
                SVGDoc.getElementById("myContainer").appendChild(newCircle);
                addText(evt.clientX,evt.clientY);
                // ID Zähler um ein erhöhen
                p++;
        }

        /*
        * Die Funktion reagiert auf entsprechende Ergeignisse die vom Benutzer ausgeführt wer-
den
        * Dabei sollten die verscheidene Browser berücksitig werden
        */
        function listen()
        {
                // IE: läuft nur nicht ???????
                // Opera funktoiniert
                if(SVGDoc.attachEvent)
                {
                // Klick event zum  Hinzufügen eines Punktes auf der Grafik
                SVGDoc.getElementById("grundriss").attachEvent("onclick",addPoint);
                // Event zum verfolgen der aktuellen Mouse Position
                SVGDoc.getElementById("grundriss").attachEvent("onmousemove",position);
                // Klick event auf einen Hinzugefügten Punkt zum löschen
                SVGDoc.getElementById("myContainer").attachEvent("onclick",removePoint);
```

```
		}else if(SVGDoc.addEventListener)
		{
		// Klick event zum  Hinzufügen eines Punktes auf der Grafik
		SVGDoc.getElementById("grundriss").addEventListener("click",addPoint, true);
		// Event zum verfolgen der aktuellen Mouse Position
		SVGDoc.getElementById("grundriss").addEventListener("mousemove",position,
	true);
		// Klick event auf einen Hinzugefügent Punkt zum löschen
	SVGDoc.getElementById("myContainer").addEventListener("click",removePoint, true);
		}
	}
	/*
	* Startfunktion
	*/
	function Init()
	{
		// Zuweisen der SVG Datei in einer Variable
		SVGDoc = document.getElementById("svg").getSVGDocument();
		listen(); // aufruf der Evtenlistener Funktion
	}
```

zielpunkte.js

```
		var SVGDoc = null;
		var paint = "red";
		var art = "ZP";
		var digi_art = null;
		var z = 0;
		var p = 0;
		var x_werte = new Array();
		var y_werte = new Array();
		var p_art = new Array();
		var http = null;
		var h_id = 0;
		var eb_id = 0;
		/*
		* Funktion prüft welche Punktart in HTML Dokument ausgewählt wurde und setzt
die jeweilige Farbe und Art
		*/
		function check_art()
		{
			if(document.punktart.art[0].checked == true)
			{
				paint = "red";
				art = "ZP";
			}else if(document.punktart.art[1].checked == true)
			{
				paint = "black";
				art = "ZW";
			}else if(document.punktart.art[2].checked == true)
			{
				paint = "blue";
				art = "AU";
			}else if(document.punktart.art[3].checked == true)
			{
				paint = "grey";
				art = "TR";
			}else if(document.punktart.art[4].checked == true)
			{
					paint = "yellow";
					art = "NK";
				}else if(document.punktart.art[5].checked == true)
				{
					paint = "yellow";
					art = "HK";
				}else if(document.punktart.art[6].checked == true)
				{
					paint = "orange";
					art = "HE";
				}else
				{
					document.punktart.art[0].checked == true;
				}
		}

		/*
		* Funktion übergibt die Koordinaten aller Punkte per POST an wegWerte.php
```

```
		* PHP Script erstellt das Formular
		*/
		function ajax()
		{
			if (window.XMLHttpRequest)
			{
				http = new XMLHttpRequest();
			} else if (window.ActiveXObject)
			{
				http = new ActiveXObject("Microsoft.XMLHTTP");
			}
			if(http !=null)
			{
				h_id  = document.getElementById("hid").value;
				eb_id = document.getElementById("ebid").value;
				var                              parameter                                =
"x_werte="+x_werte+"&y_werte="+y_werte+"&art="+p_art+"&hid="+h_id+"&ebid="+eb_id;
				http.open("POST", "wegWerte.php", true);
				http.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
				http.send(parameter);
				http.onreadystatechange = outWerte;
			}

		}

		/*
		* Ausgabe des in der Ajax Funktion erstellten Formulars
		*/
		function outWerte()
		{
			if (http.readyState == 4)
			{

				var       fenster       =       window.open("","Punktaufnehmen",
"width=800,height=600" );
				fenster.document.open();
				fenster.document.write(http.responseText);

			}
		}

		/*
		* Funktion überwacht die aktuelle Position der Mouse
		* Position wird im HTML Dokument ausgeben
		*/
		function position(evt)
		{
				document.getElementById("positionX").value = evt.clientX;
				document.getElementById("positionY").value = evt.clientY;

		}

		/*
		* Hinzufügen der Punkte im HTML Dokument, zu besseren Orientierung
		*/
		function addText(px,py)
		{
			var x=px;
			var y=py;
			// Runden der Koordinaten
			newX = Math.round(x);
			newY = Math.round(y);
			var out = document.getElementById("neu");
			var myContent = document.createElement("h3");
			myContent.setAttribute("id",z);
			out.appendChild(myContent);
			var myID = document.createTextNode(" ID ");
			myContent.appendChild(myID);
			out.appendChild(myContent);
			var myID1 = document.createTextNode(z);
			myContent.appendChild(myID1);
			out.appendChild(myContent);
			var myText3 = document.createTextNode(" X-Wert ");
			myContent.appendChild(myText3);
			out.appendChild(myContent);
			var myText4 = document.createTextNode(newX);
			myContent.appendChild(myText4);
```

```
                        out.appendChild(myContent);
                        var myText5 = document.createTextNode(" Y-Wert ");
                        myContent.appendChild(myText5);
                        out.appendChild(myContent);
                        var myText6 = document.createTextNode(newY);
                        myContent.appendChild(myText6);
                        out.appendChild(myContent);
                        var myText7 = document.createTextNode(" Punktart ");
                        myContent.appendChild(myText7);
                        out.appendChild(myContent);
                        var myText8 = document.createTextNode(art);
                        myContent.appendChild(myText8);
                        out.appendChild(myContent);
                        // Speichern der Koordinaten und Art des Punktes im jeweiligen globalen
Array
                        x_werte.push(newX);
                        y_werte.push(newY);
                        p_art.push(art);
                        // ID Zähler wird erhöht
                        z++;
                }

                /*
                * Löschen des in SVG erzeugten Punktes
                * Löschen der Ausschrift im HTML
                * Zugriff erfolgt über die entsprechende ID
                */
                function removePoint(evt)
                {
                        var obj = evt.target;
                        var  Att_id = evt.target.getAttributeNS(null,"id");
                        var remove = obj.parentNode.removeChild(obj);
                        alert("ID: "+Att_id);
                        var obj = document.getElementById(Att_id);
                        var k = document.getElementById(Att_id).removeNode(true);
                        // Löschen der Werte aus dem x und y Array
                        x_werte.splice(Att_id,1,"re");
                        y_werte.splice(Att_id,1,"re");
                        p_art.splice(Att_id,1,"re");
                }

                /* Hinzufügen neuer Punkte auf der Graphik
                * Ermitteln der Mouse Position
                * an dieser Stelle wird ein Kreis mit dem Radius 5 erstellt
                * Position wird an die Funktion addText weitergeben
                * Anschließend wird der ID Zähler (p) um ein erhöht
                */
                function addPoint(evt)
                {
                        var newSVGOb = "http://www.w3.org/2000/svg";
                        var newCircle =document.createElementNS(newSVGOb,"circle");
                        newCircle.setAttributeNS(null,"id",p);
                        newCircle.setAttributeNS(null,"cx",evt.clientX);
                        newCircle.setAttributeNS(null,"cy",evt.clientY);
                        newCircle.setAttributeNS(null,"r",5);
                        newCircle.setAttributeNS(null,"fill",paint);
                        SVGDoc.getElementById("myContainer").appendChild(newCircle);
                        addText(evt.clientX,evt.clientY);
                        p++;
                }
                /*
                * Die Funktion reagiert auf entsprechende Ergeignisse die vom Benutzer ausge-
führt werden
                * Dabei sollten die verscheidene Browser berücksitig werden
                */
                function listen()
                {
                        // IE: läuft nur nicht ???????
                        // Opera funktoiniert beide
                        if(SVGDoc.attachEvent)
                        {
                        // Klick event zum  Hinzufügen eines Punktes auf der Grafik
                        SVGDoc.getElementById("grundriss").attachEvent("onclick",addPoint);
                        // Event zum verfolgen der aktuellen Mouse Position
                        SVGDoc.getElementById("grundriss").attachEvent("onmousemove",position);
                        // Klick event auf einen Hinzugefügent Punkt zum löschen
                        SVGDoc.getElementById("myContainer").attachEvent("onclick",removePoint);
                        }else if(SVGDoc.addEventListener)
```

```
                {
                // Klick event zum  Hinzufügen eines Punktes auf der Grafik
                SVGDoc.getElementById("grundriss").addEventListener("click",addPoint,
true);
                // Event zum verfolgen der aktuellen Mouse Position
        SVGDoc.getElementById("grundriss").addEventListener("mousemove",position,
true);
        // Klick event auf einen Hinzugefügten Punkt zum löschen
        SVGDoc.getElementById("myContainer").addEventListener("click",removePoint,
true);
                }

        }

        /*
        * Startfunktion
        */
        function Init()
        {
                // Zuweisen der SVG Datei in einer Variable
                SVGDoc = document.getElementById("svg").getSVGDocument();
                listen();// aufruf der Evtenlistener Funktion
        }
```

Anhang 6 – Quelltext der Klasse grundriss()

```
class grundriss
{

	/**
	 * grundriss::build() Methode erstellt die SVG Datei für die Aufnahme von Grundriss-
punkte
	 *
	 * @param mixed $id   = ID der ausgewählten Graphik Datei
	 * @param mixed $w           = Referenz für die Länge
	 * @param mixed $h           = Referezn für die Breite
	 * @param mixed $hid  = ID des Hauses
	 * @param mixed $ebid = ID der Ebene die zu der Graphik gehört
	 * @return     $d_name        = Der SVG Dateiname
	 */
	function build($id,&$w,&$h,&$hid,&$ebid)
	{
	require_once ($_SERVER['DOCUMENT_ROOT'] . "/neuweles/conf/config.inc");
	$sql = "SELECT ebene_ebenen_id,haus_id,name,laenge,breite FROM graphik where
gid=".$id.";";
	require ($_SERVER['DOCUMENT_ROOT'] . "/neuweles/conf/dbconn.inc");
	while ($row = mysql_fetch_array($ergebnis))
	{
	$hid  = $row['haus_id'];
		$ebid = $row['ebene_ebenen_id'];
		$name = $row['name'];
		$w    = $row['laenge'];
		$h    = $row['breite'];

	}

		$d_name = "grundriss.svg"; //Dateiname
		$url = $_SERVER['DOCUMENT_ROOT'] . '/neuweles/digi/bilder/tmp/' . $d_name;
		$datei = fopen($url, "w+");
		$grundSVG = '
		<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
		<svg  xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ev="http://www.w3.org/2001/xml-events">

		<defs>
		<pattern id="myHatch" patternUnits="userSpaceOnUse" x="0" y="0" width="50"
height="50">
		<g>
			<rect x="0" y="0" width="50" height="50" fill="none"/>
			<path d="M0,50 v -50 h 50" stroke="gray" fill="none" stroke-width="1"/>
		</g>
		</pattern>
		</defs>
		<g id="grundriss" cursor="crosshair" >
		 <image x="0" y="0" xlink:href="/neuweles/digi/bilder/grundriss/'.$name . '"
width="'.$w .'" height="'.$h.'" />
		<rect x="0" y="0" height="'. $h .'" width="'. $w .'" fill="url(#myHatch)"
stroke="none" />
		</g>
		<g id="myContainer"/>
		</svg>';
		fwrite($datei, $grundSVG);
		fclose($datei);
		return $d_name;
	}

	/**
	 * grundriss::buildZiel() Methode erstellt die SVG Datei für die Aufnahme von Ziel-,und
Wegpunkte
	 *
	 *
	 * @param mixed $id   = ID der ausgewählten Graphik Datei
	 * @param mixed $w           = Referenz für die Länge
	 * @param mixed $h           = Referezn für die Breite
	 * @param mixed $hid  = ID des Hauses
	 * @param mixed $ebid = ID der Ebene die zu der Graphik gehört
	 * @return     $d_name        = Der SVG Dateiname
	 */
```

```
        function buildZiel($id,&$w,&$h,&$hid,&$ebid)
        {

        require_once ($_SERVER['DOCUMENT_ROOT'] . "/neuweles/conf/config.inc");
        $sql = "SELECT ebene_ebenen_id,graphik.haus_id,name,laenge,breite FROM graphik,haus
where gid=".$id.";";
        require ($_SERVER['DOCUMENT_ROOT'] . "/neuweles/conf/dbconn.inc");
        while ($row = mysql_fetch_array($ergebnis))
        {
                $hid  = $row['haus_id'];
                $ebid = $row['ebene_ebenen_id'];
                $name = $row['name'];
                $w    = $row['laenge'];
                $h    = $row['breite'];

        }
                $d_name = "digiZiel.svg"; //Dateiname
                $url = $_SERVER['DOCUMENT_ROOT'] . '/neuweles/digi/bilder/tmp/'.$d_name;
                $datei = fopen($url, "w+");
                $grundSVG = '
                <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
                <svg  xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ev="http://www.w3.org/2001/xml-events">

                <defs>
                <pattern id="myHatch" patternUnits="userSpaceOnUse" x="0" y="0" width="50"
height="50">
                <g>
                <rect x="0" y="0" width="50" height="50" fill="none"/>
                <path d="M0,50 v -50 h 50" stroke="gray" fill="none" stroke-width="1"/>
                </g>
                </pattern>
                </defs>

                <g id="grundriss" cursor="crosshair" >
                <image x="0" y="0" xlink:href="/neuweles/digi/bilder/grundriss/'.$name . '"
width="'.$w .'" height="'.$h.'" />
                <rect x="0" y="0" height="'.$h.'" width="'.$w.'" fill="url(#myHatch)"
stroke="none" />
                </g>
                <g id="WP"/>';
                $sql = "SELECT WP_ID,x,y,art FROM wegpunkte,graphik where weg-
punkte.ebenen_id=".$ebid." and graphik.ebene_ebenen_id=".$hid.";";
                require ($_SERVER['DOCUMENT_ROOT'] . "/neuweles/conf/dbconn.inc");
                $num_rows = mysql_num_rows($ergebnis);
                if($num_rows != 0)
                {
                        while ($row = mysql_fetch_array($ergebnis))
                        {
                                $grundSVG =$grundSVG.'<circle id="'.$row['WP_ID'] .'"
cx="'.$row['x'] .'" cy="'.$row['y'] .'" r="5" />
                                <text x="'. (($row['x'])+4) .'" y="'. (($row['y'])+4) .'"
style="font-size:10pt;font-family:Arial"> '. $row['WP_ID'] .' </text>';
                        }

                }

                $grundSVG =$grundSVG.'
                <g id="myContainer"/>
                </svg>';
                fwrite($datei, $grundSVG);
                fclose($datei);
                return $d_name;

        }

}
```

Anhang 7 – Quelltext der Klasse building()

```
<?php

/**
 * @author Marcel Klingberg
 */

class Building
{
	private $ebid;
	private $hid;

	/**
	 * Building::__construct() Konstruktor der Klasse
	 *
	 * @param mixed $eb = Referenz auf die ebene_id
	 * @param mixed $hi = Referenz auf die haus_id
	 */
	function __construct(&$eb,&$hi)
	{

		$this->ebid=$eb;
		$this->hid =$hi;
	}




	/**
	 * Building::point_exc() Methode für die Stringverarbeitung, um die einzelen Punkte aus
dem Rückgabestring der DB Abfrage zu ermitteln
	 * Möglich wäre auch die nicht benötigten Zeichen aus dem Kompletten Rückgabestring zu
entferenen
	 * @param mixed $string = Referenz auf den Rückgabestring
	 * @param mixed $xW = Referenz auf die X Koordinate eines Punktes
	 * @param mixed $yW = Refrenz auf die Y Koordinate eines Punktes
	 */
	function point_exc(&$string,&$xW,&$yW)
	{
		 $pos1 = strpos($string,"(");
		 $pos2 = strpos($string," ");
		 $pos3 = strpos($string,")");

		 $xStart = ($pos1+1);
		 $xLength = ($pos2-$pos1);
		 $xW = substr($string,$xStart,$xLength);

		 $yStart = ($pos2+1);
		 $yLength = ($pos3-$pos2)-1;
		 $yW = substr($string,$yStart,$yLength);
	    //echo $x.",".$y."<br>";

	}


	/**
	 * Building::Ground() Methode um mit Hilfe des <polygon> Tag den Grundriss zu erstellen
// im Prototypen nicht mehr genutzt
	 *
	 * @return String der Koordinaten des Polygon zum Zeichen in SVG
	 */
	function Ground()
	{


		// Variable zum Zwischenspeichern der Polygone
		$x;$y;
		/*
		 * Wenn mehrere Häuser zu eine Komplex gehören, muss erst die Anzahl der Häuser
ermittelt werden
		 * anschließend müssen die Grundriss der Häuser gezeichnet werden
		 * Im Beispiel wird nur ein Haus verwendet
		 */
		// Anzahl der Grundrisse zum einem Haus ermitteln
		require_once ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/config.inc");
```

```
		$sql="SELECT AsText(polygon) as Poly from grundriss,ebene,haus where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this->hid." and grundriss.ebene_ebenen_id = ebene.ebenen_id and art='G';";
		require ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/dbconn.inc");

		while ($row = mysql_fetch_array($ergebnis))
		{
			$ko_string = $row['Poly'];
		}

		// Anzahl der Polygonpunkte zum Grundriss ermitteln
		$sql="SELECT NumPoints(polygon) as anzahl from grundriss,ebene,haus where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this->hid." and grundriss.ebenen_id = ebene.ebenen_id and art='G';";
		require ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/dbconn.inc");

		while ($row = mysql_fetch_array($ergebnis))
		{
			$ko_anz = $row['anzahl'];
		}

		// Auswahl jedes einzelnen Punktes aus dem Polygon
		for($i=1;$i<=$ko_anz;$i++)
		{
			$sql="SELECT AsText(POINTN(polygon,".$i.")) as punkt from grundriss,ebene,haus where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this->hid." and grundriss.ebene_ebenen_id = ebene.ebenen_id and art='G';";
		   require ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/dbconn.inc");
		   $result = mysql_result($ergebnis,0);
		   $this->point_exc($result,$x,$y);

		  //echo $xWert[$i]."<br>";
		  if($i==$ko_anz)
		  {
			$polygon=$polygon.($x)." ".($y);
		  }else
		  {
			$polygon=$polygon.($x)." ".($y).",";
		  }


	  }
	  return $polygon;

	}

	/**
	 * Building::ground_plan() Methode um mit Hilfe des <polyline> Tag den Grundriss zu erstellen
	 *
	 * @param mixed $house = Referenz für die Beschriftung des Hauses
	 * @param mixed $hx = Referenz für die X Koordinate der Position der Beschriftung
	 * @param mixed $hy = Referenz für die X Koordinate der Position der Beschriftun
	 * @return String mit den Koordinaten der Polyline zum Zeichen in SVG
	 */
	function ground_plan(&$house,&$hx,&$hy)
	{
		$x;
		$y;
		$eck = array();
		$numPoint = array();
		$i = 0;
		require_once ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/config.inc");
		$sql="SELECT beschreibung, X(position), Y(position) FROM haus where haus_id=".$this->hid.";";
		require ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/dbconn.inc");

		while($row=mysql_fetch_array($ergebnis))
		{
			$house=$row['beschreibung'];
			$hx=$row['X(position)'];
			$hy=$row['Y(position)'];

		}

		$sql ="SELECT eck_id,NumPoints(polygon) as anzahl from grundriss,ebene,haus where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this->hid." and grundriss.ebenen_id = ebene.ebenen_id and art='G';";
```

```
                require ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/dbconn.inc");
                // anzahl der gefunden Datensätze
                $num_rows = mysql_num_rows($ergebnis);
                        // Verwaltung der ID's und Anzahl der Punkte in zwei Arrays
                while($row=mysql_fetch_array($ergebnis))
                {
                        $eck[$i]=$row['eck_id'];
                        $numPoint[$i]=$row['anzahl'];
                        $i++;
                }
                // äußere Schleife mit der Anzahl der Datensätze
                for($j=0;$j<=$num_rows;$j++)
                {

                        // innere Schleife mit der Anzahl der Punkte innerhalb eines Datensatzes
                        for($i=1;$i<=$numPoint[$j];$i++)
                        {
                                $sql="SELECT AsText(POINTN(polygon,".$i.")) as punkt from grun-
driss,ebene,haus where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this-
>hid." and grundriss.ebenen_id = ebene.ebenen_id and art='G' and eck_id=".$eck[$j].";";
                                require ($_SERVER['DOCUMENT_ROOT']
."/neuweles/conf/dbconn.inc");

                                $result = mysql_result($ergebnis,0);
                                // Strungverarbeitung um einen einzelen Punkte rauszuziehen
                                $this->point_exc($result,$x,$y);

                                        // Bildung eines String mit allen Punkten
                                        // : um die einzelnen Datensätze im String zu unterschei-
den
                                        if($i==$numPoint[$j])
                                        {
                                        $polyline=$polyline.($x)." ".($y).":";
                                        }else
                                        {
                                                $polyline=$polyline.($x)." ".($y).",";
                                        }
                        }

                 }


                 mysql_free_result($ergebnis);
                 // Rückgabe des Datenstrings
                 return $polyline;
        }

        /**
         * Building::room() Methode zum Zeichen der Räume einer Etage mit Hilfe des <polyline>
Tag
         *
         * @return String mit den Koordinaten der Polyline zum Zeichen in SVG
         */
        function room()
        {
                $x;
                $y;
                $eck = array();
                $numPoint = array();
                $i=0;

                // Ermitteln der ID und Anzahl des Datensätze
                require_once ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/config2.inc");
                $sql ="SELECT eck_id,NumPoints(polygon) as anzahl from grundriss,ebene,haus
where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this->hid." and grun-
driss.ebenen_id = ebene.ebenen_id and art='R';";
                require ($_SERVER['DOCUMENT_ROOT'] ."/neuweles/conf/dbconn.inc");
                // anzahl der gefunden Datensätze
                $num_rows = mysql_num_rows($ergebnis);

                // Verwaltung der ID's und Anzahl der Punkte in zwei Arrays
                while($row=mysql_fetch_array($ergebnis))
                {
                        $eck[$i]=$row['eck_id'];
                        $numPoint[$i]=$row['anzahl'];
                        $i++;
                }
```

```
                // äußere Schleife mit der Anzahl der Datensätze
                for($j=0;$j<=$num_rows;$j++)
                {

                        // innere Schleife mit der Anzahl der Punkte innerhalb eines Datensatzes
                        for($i=1;$i<=$numPoint[$j];$i++)
                        {
                                $sql="SELECT AsText(POINTN(polygon,".$i.")) as punkt from grun-
driss,ebene,haus where grundriss.ebenen_id = ".$this->ebid." and grundriss.haus_id = ".$this-
>hid." and grundriss.ebenen_id = ebene.ebenen_id and art='R' and eck_id=".$eck[$j].";";
                                require ($_SERVER['DOCUMENT_ROOT']
."/neuweles/conf/dbconn.inc");

                                $result = mysql_result($ergebnis,0);
                                // Strungverarbeitung um einen einzelen Punkte rauszuziehen
                                $this->point_exc($result,$x,$y);

                                        // Bildung eines String mit allen Punkten
                                        // : um die einzelnen Datensätze im String zu unterschei-
den
                                        if($i==$numPoint[$j])
                                        {
                                        $polyline=$polyline.($x)." ".($y).":";
                                        }else
                                        {
                                                $polyline=$polyline.($x)." ".($y).",";
                                        }
                        }

                 }


                 mysql_free_result($ergebnis);
                 // Rückgabe des Datenstrings
                 return $polyline;

                }



}
?>
```

Anhang 8 – Übersicht der Quelltextdateien des Prototyps

Die folgende Tabelle stellt eine gesamt Übersicht der Quelltextdateien des Prototyps dar.

Datei	Beschreibung
`index.php`	• Startdatei des Systems • Auswahlmöglichkeit zwischen Digitalisierungs- und Suchfunktion
`tool.php`	• wird über den Button Bearbeiten der `index.php` aufgerufen • Startseite des Digitalisierungsmoduls • Auswahl der möglichen Funktionen des Digitalisierungsmoduls
`formHaus.php`	• wird über die `tool.php` aufgerufen • Formular zum Hinzufügen eines Haus
`hausDb.php`	• übernimmt die Formulardaten der `formHaus.php` und fügt diese in die Datenbank ein
`formEbene.php`	• wird über die `tool.php` aufgerufen • Formular zum Hinzufügen einer Ebene eines Hauses
`ebeneDb.php`	• Übernimmt die Formulardaten der `formEbene.php` und fügt diese in die Datenbank ein
`formRiss.php`	• wird über die `tool.php` aufgerufen • Formular zum Hinzufügen einer neuen Grundrissgraphik
`rissDb.php`	• übernimmt die Formulardaten der `formRiss.php` und fügt diese in die Datenbank ein
`del_grundriss.php`	• wird über die `tool.php` aufgerufen • Formular zum Löschen einer Grundrissgraphik aus der Datenbank • Löscht die Graphik nicht aus dem Arbeitsverzeichnisses
`digi.php`	• wird über die `tool.php` aufgerufen • Aufbau des Arbeitsbereiches zur Aufnahme von Grundrisspunkten • bindet die `grundriss.js` für die Aufnahme ein
`werte.php`	• Formular zum Hinzufügen der Grundrisspunkte (G, R)
`grundriss.Db.php`	• übernimmt die Formulardaten der `werte.php` und fügt

	diese in die Datenbank ein
digi_ziel.php	• wird über die tool.php aufgerufen • Aufbau des Arbeitsbereiches zur Aufnahme von Ziel-, und Wegpunkten • bindet die zielpunkte.js für die Aufnahme ein
wegwerte.php	• Formular zum Hinzufügen von Ziel-, und Wegpunkten
wegpunkteDb.php	• übernimmt die Formulardaten der wegwerte.php und fügt diese in dieDatenbank ein
grundriss.php	• enthält die Klasse grundriss() • generieren der SVG Datei für beide Arbeitsbereiche
weles_start.php	• Start der Suchfunktion des WeLeS • Auswahl des Start-, und Zielpunktes • Etagenwechsel in Prototyp nicht berücksichtigt
suche.php	• Starten der Zielwegsuche • Präsentation des Ergebnisses • einbinden der svg.php
svg.php	• generiert die Ausgabe SVG Datei • bindet die Pfad.php ein
Pfad.php	• generiert den Pfad für die SVG Ausgabedatei • Quelltext wurde nicht verändert • Bindet die position.php ein
position.php	• positioniert Symbole und Pfad ohne sich zuschneiden • Quelltext wurde nicht verändert
graph.php graph_no_stairs.php	• beide bilden den Graphen und realisieren den Suchalgorithmus mit Treppe oder ohne • Quelltext wurde nicht verändert
mysql_graph.php mysql_graph_no_stairs.php	• SQL Abfrage des Graphen mit oder ohne Treppe • Quelltext wurde nicht verändert
Building.php	• enthält die Klasse Building() zu Abfrage der Gebäudeinformationen aus der Datenbank