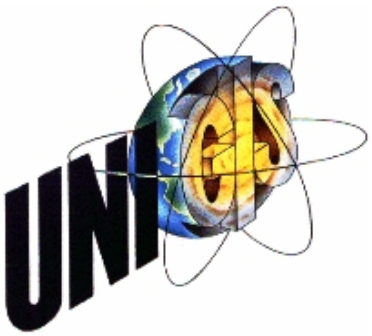


Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„WebGIS“
als Medium zur Umsetzung neuer Nutzungskonzepte im Immobilienmanagement

vorgelegt von

Maria Daedelow
U1408, UNIGIS MSc Jahrgang 2008

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Münster, 28.02.2011

Erklärung

Ich versichere, diese Master Thesis selbstständig, ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit die wörtlich oder sinngemäß übernommen wurden, sind entsprechend gekennzeichnet.

Münster, Februar 2011

.....

Maria Daedelow

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Fragestellung, wie Kulturschaffende und Verwalter öffentlicher Immobilien gemeinsam von der Kommunikation mittels eines WebGIS profitieren können. Die angespannte Finanzlage der Kommunen führt im öffentlichen Immobilienmanagement zu der Entwicklung neuer Nutzungskonzepte. Diese Konzepte schließen die Misch- und die Zwischennutzung von (partiell) leerstehenden Räumen und Gebäuden mit ein. Kulturschaffende eignen sich aufgrund ihrer Flexibilität und Innovationsbereitschaft besonders als Nutzer solcher Angebote.
Um die Kommunikation zwischen beiden Seiten zu erleichtern und einen Überblick über das Raumangebot und den Grad der Auslastung zu erhalten, wird nach der umfassenden Darlegung der aktuellen technischen Voraussetzungen mit dieser Arbeit ein Konzept und ein Lösungsvorschlag für eine Open-Source-Anwendung nach OGC-Standards vorgestellt. Auf Basis von OpenLayers wird ein WebGIS-Client unter Einbeziehung der JavaScript-Bibliotheken ExtJs und GeoExt den Anforderungen entsprechend angepasst.

Abstract

This thesis adresses the issue how persons engaged in the cultural sector and official property administration can benefit from each other using WebGIS technologies. Today's municipal financial situation calls for the development of new utilization concepts. These concepts include mixed usage and intermediate usage of (partially) disused rooms and buildings. Because of their flexibility and innovativeness, persons engaged in the cultural sector qualify as potential users of such proposals. To simplify communication between both sides and to gain information of property capacities and the degree of utilization, this work introduces a concept for an Open Source application according to OGC standards, after outlining the technological preconditions. A WebGIS client based on OpenLayers is being customized using ExtJs and GeoExt JavaScript libraries.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
AJAX	Asynchronous JavaScript and XML
CAFM	Computer Aided Facility Management
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
EPSG	European Petroleum Survey Group
GIS	GeoInformationsSystem
GML	Geography Markup Language
GRASS	Geographic Resources Analysis Support System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
KML	Keyhole Markup Language
MIME-Typ	Multimedia Internet Media Type
OGC	Open Geospatial Consortium
OGF	Open GRASS Foundation
OSGeo	Open Source Geospatial Foundation
OSM	OpenStreetMap
PHP	Hypertext Preprocessor bzw. Personal Home Page Tools
SE	Symbology Encoding
SLD	Styled Layer Descriptor
SMIL	Synchronized Multimedia Integration Language
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
Tab.	Tabelle
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WFS	Web Feature Service
WKT	Well Known Text
WMS	Web Map Service
WPS	Web Processing Service

WWW	World Wide Web
XHR	XMLHttpRequest
XML	Extensible Markup Language
XSD	XML Schema Definition
XSS	Cross-Site Scripting

1 Einleitung

In Zeiten knapper Haushaltskassen sind Kommunen bestrebt, in möglichst vielen Bereichen Einsparungen zu treffen. Ein besonders großes Einsparpotenzial wird dabei den städtischen Immobilien zugesprochen. Es gilt, den Immobilienbedarf zu evaluieren, gebundenes Kapital durch Flächenverkauf liquide zu machen und durch neue Nutzungskonzepte Leerstände zu verringern. Zu solchen Nutzungskonzepten gehören unter anderem auch die Zwischen- und die Misch- (oder Mehrfach-) Nutzung, welche sich in besonderem Maße für kulturelle Zwecke eignen.

Kulturschaffende benötigen Experimentier- und Erprobungsflächen, um ihre Kreativität ausleben zu können. Sie gelten als flexibel und innovativ und können als Auslöser von Standortentwicklungsprozessen zu einer Aufwertung von einzelnen Immobilien oder ganzen Stadtvierteln beitragen.

Die Frage ist, wie beide Seiten zueinander finden können. Hierzu bedarf es eines geeigneten Kommunikationsmittels. Der stetige Fortschritt der Internettechnologien und der Boom von interaktiven Web 2.0-Anwendungen wie Google Maps sowie das gesteigerte Interesse und die Netzkompetenz der Nutzer ermöglichen die Entwicklung einer Lösung auf Basis von WebGIS-Strukturen.

Zielsetzung

Ziel der vorliegenden Arbeit ist es, ein Konzept für eine WebGIS-Anwendung, die Daten über verfügbare öffentliche Immobilien bereitstellt und den Anforderungen von Kulturschaffenden als Nutzer und der Kommune als Immobilieneigentümer und Kulturförderer nachkommt, zu entwickeln. Es soll ein Anforderungsprofil erstellt werden, das beinhaltet, welche Daten und Funktionen für eine solche Anwendung benötigt werden. Des Weiteren soll in einer Beispiellösung gezeigt werden, wie das Konzept in Anwendung umgesetzt werden kann und ob alle konzeptionellen Anforderungen mit den aktuellen technischen Möglichkeiten eines WebGIS erfüllt werden können.

Gliederung der Arbeit

Die Arbeit gliedert sich in 7 Kapitel. Ausgehend von der *Definition des Kulturbegriffs (2.1)* und einem kurzen Abriss über die *Bedeutung von Kultur* für die Stadt im Allgemeinen und dem Potenzial als *Standortfaktor (2.2)*, werden die *Anforderungen* von Kulturschaffenden an den von ihnen benötigten *Raum (2.3)* geschildert.

Kapitel 3 gibt einen Überblick über *kommunales Immobilienmanagement*. Die Kommune bestimmt als Eigentümer einer großen Anzahl von Liegenschaften über deren Nutzung. Es werden die *Funktionen öffentlicher Immobilien (3.1)* und die *Aufgabenfelder* des kommunalen Immobilienmanagements (*3.2*) aufgeführt, auf den Bedarf an *Optimierung (3.3)* hingewiesen sowie Nutzungskonzepte wie Zwischen- und Mischnutzung betrachtet.

In *Kapitel 4* wird auf das Internet als Kommunikationsmedium eingegangen und der Begriff *WebGIS* erläutert.

Kapitel 5 befasst sich eingehend mit den für ein WebGIS relevanten *Basistechnologien*. Diese umfassen die *Client-Server-Architekur (5.1)*, die Internet-Standards von *W3C* und *IETF* (Kommunikationsprotokolle und Scriptsprachen) sowie die *OGC-Spezifikationen* WMS *(5.4.1)*, WFS *(5.4.2)*, GML *(5.4.3)*, SLD *(5.4.6)*, Filter Encoding *(5.4.5)*, Simple Feature Access *(5.4.4)* und WPS *(5.4.7)*.

In *Kapitel 6* wird ein Konzept für die Umsetzung eines WebGIS, welches die Kommunikation zwischen Kulturschaffenden und Ansprechpartnern von Immobilienmanagement / Kulturförderung fördern soll, vorgestellt. Das Konzept beinhaltet ein *Anforderungsprofil (6.1)*, die einzelnen Komponenten der *Systemarchitektur* und einen Lösungsvorschlag für eine *Beispielanwendung (6.6)* mit OpenLayers in Kombination mit ExtJs und GeoExt.

Kapitel 7 schließlich fasst die vorhergehenden Kapitel noch einmal zusammen, zieht ein Fazit und wagt einen Ausblick auf künftige Entwicklungen.

2 Kultur

2.1 Kulturbegriff

Kultur im weitesten Sinne beschreibt die Gesamtheit aller vom Menschen selbst hervorgebrachten Gestaltungsformen, aller Lebensformen. Dies umfasst materielle Gestaltungsformen der Umwelt (z.B. Architektur, bildende Kunst) und schließt abstrakte Gebilde wie das Wissen und die Nutzung gesetzmäßig ablaufender Naturprozesse (Wissenschaft und Technik), Ideen und Wertvorstellungen (z.B. Religion, Moral, Sinngebung) mit ein (vgl. HILLMANN 1994).
Kultur im engeren Sinne meint Kunst (Theater, Musik, bildende Kunst, Literatur, Film) als Ausdrucksform und Kommunikationsmittel von Kultur.

2.2 Bedeutung von Kultur

„Kultur ist nicht nur das Ferment, das Städte bewohnbar und anziehend macht; sie ist [...] die wichtigste Chance, um Gesellschaft kommunikations- und sprachfähig zu halten. [...] Mehr denn je gilt: Ohne Kultur ist keine Stadt [...].“ (SAUBERZWEIG 1989).

Kultur als Standortfaktor

Bedingt durch den Wandel von der Industrie- zur Dienstleistungsgesellschaft rücken für Unternehmen neben den sogenannten harten Standortfaktoren wie Infrastruktur und Lohnkosten zunehmend weiche Faktoren wie Wohn- und Lebensqualität, Kulturangebot und Freizeitmöglichkeiten ins Blickfeld standortstrategischer Entscheidungen.

> „Die weichen Standortfaktoren haben [...] an Gewicht gewonnen. „Weich“ heißt, daß diese Faktoren nicht so knallhart von den Betrieben kalkuliert werden, wie z.B. Lohnkosten, Transportkosten [...] usw. Sie können nicht nach einem Investitionsplan hergestellt werden und wirken eher aufs Gemüt. Die unverbrauchte Landschaft gehört dazu, das Wetter und das städtische Ambiente. Das alles werde vom High-Tech-Flügel der Lohnarbeiterschaft so sehr geschätzt, daß es als Standortfaktor für moderne Industrie gilt.“ (HÄUßERMANN & SIEBEL 1987)

Kultur als Auslöser von Standortentwicklungsprozessen

Kultur kann als Auslöser von Standortentwicklungsprozessen fungieren, wie folgt beschrieben.

Leer stehende Gebäude in vernachlässigten Vierteln werden von Kulturschaffenden als kostengünstige Räume für die Entfaltung ihrer kreativen Möglichkeiten entdeckt, als Ateliers, Proberäume, Werkstätten etc. genutzt. Ausstellungen, Konzerte und andere Veranstaltungen führen zu einer Belebung des Viertels. In der Folge siedeln sich Cafés, Kneipen und Einzelhandel an. Der Standort wird durch die kulturelle Belebung aufgewertet, das Image verbessert sich (vgl. WEHRLI-SCHINDLER 2002, BECKER 2009).

BECKER 2009 fasst zusammen: „Eine vielfältige und aktive Kunstszene erhöht die Standortattraktivität im Wettbewerb um gut ausgebildete und finanzkräftige Bevölkerungsgruppen sowie den Kulturtourismus. Hiervon wiederum profitieren Gastronomie- und Beherbergungsbetriebe."

Kulturwirtschaft als eigener Bilanzierungsbereich

Dass Kultur wirtschaftlich gesehen mehr als ein Imagefaktor ist, belegt ein vom Bundesministerium für Wirtschaft und Technologie (BMWi) herausgegebener Forschungsbericht des aus dem Jahr 2009. Demnach kann die Kultur- und Kreativwirtschaft als eigenes Wirtschaftsfeld angesehen werden, welches „als Wachstumsbranche zu etablieren ist" (BMWI 2009).

Zu den Kernbranchen der Kulturwirtschaft gehören die Musikwirtschaft, der Buchmarkt, der Kunstmarkt, die Filmwirtschaft, die Rundfunkwirtschaft, der Markt für darstellende Künste, die Designwirtschaft, der Architekturmarkt, der Pressemarkt, der Werbemarkt sowie die Software- / Games-Industrie.

Die Kultur- und Kreativwirtschaft gilt als Quelle für Innovationen. Die Branche ist gekennzeichnet durch zukunftsorientierte Arbeits- und Geschäftsmodelle und die Nutzung moderner Technologien, vor allem im Bereich der Informations- und Kommunikationstechnologien (BMWI 2009).

Laut BMWI 2009 erwirtschaftete die Branche 2006 in Deutschland einen Bruttowertschöpfungsbetrag von 61 Mrd. EUR, dies entspricht in etwa 2,6 % vom

Bruttoinlandsprodukt. Im Vergleich zu den anderen Branchen positioniert sie sich damit auf Platz 3 hinter der Maschinenbau- (74 Mrd. EUR) und der Automobilindustrie (71 Mrd. EUR).

Im Jahr 2006 arbeiteten in der Branche 938.000 Menschen (Abb. 1). Verglichen mit anderen Branchen ist der Anteil Selbständiger und freier Mitarbeiter besonders hoch. Wenn auch die gesamtwirtschaftliche Position der Branche positiv zu sehen ist, kann die wirtschaftliche Situation der einzelnen Kulturschaffenden jedoch eher als kritisch bewertet werden, so brauchen Kreative oft ein zweites finanzielles Standbein.

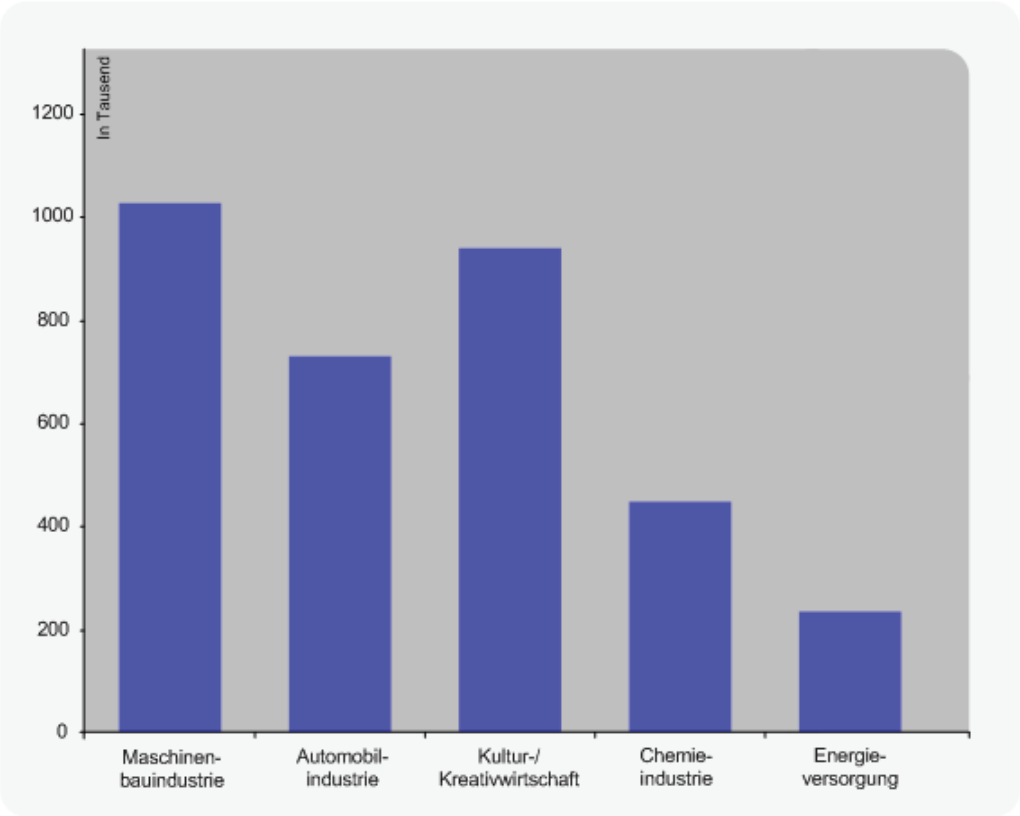


Abb. 1 Erwerbstätige der Kultur- und Kreativwirtschaft im Branchenvergleich 2006.
Verändert nach BMWi 2009.

2.3 Raum für Kulturschaffende

Kulturschaffende benötigen Raum, um sich kreativ entwickeln und entfalten zu können. KLAUS 2006 betont die Bedeutung von „Institutionen zur Ausbildung von talentierten, kreativen jungen Menschen […] für die Entwicklung der Kulturwirtschaft". Institutionen wie Kulturzentren sind Orte, die „kulturwirtschaftliche Innovationen" hervorbringen und stellen als Treffpunkte und Freiräume eine Voraussetzung zur

Entfaltung kultureller Innovation und der Entwicklung kreativer Städte (vgl. KLAUS 2006).

Kulturschaffende lassen sich in zwei Gruppen einteilen. Die eine Gruppe fasst jene zusammen, welche in ihrer Freizeit kreativen Tätigkeiten wie singen oder tanzen nachgehen und dafür gelegentlich, vielleicht ein- oder zweimal pro Woche einen Raum benötigen.

Die andere Gruppe steht für diejenigen, die Kunst professionell betreiben, also als Erwerbstätige der Kultur- und Kreativwirtschaft Fuß fassen wollen. Sie benötigen Räume, die ihnen Flächen zum Experimentieren bieten. Bedingt durch die Tatsache, dass sie oft nur über geringe finanzielle Mittel verfügen, sind die Ansprüche Kulturschaffender an bauliche, energetische und ausstattungstechnische Qualität von Gebäuden nicht allzu hoch (vgl. BÜRGI ET AL. 1995, BECKER 2009).

Eine Auswahl von Anforderungen an Liegenschaften, welche BECKER 2009 benennt, ergänzt um eigene, ist in Tab. 1 aufgeführt.
Danach spielen neben den finanziellen Rahmenbedingungen auch organisatorische Kriterien, wie die schnelle Verfügbarkeit, oder funktionale, wie die Raumhöhe oder die Fensterfläche eines Raumes eine Rolle. Ebenso wichtig sind die Umfeldbedingungen, z.B. die Erreichbarkeit durch öffentliche Verkehrsmittel.

Tab. 1 Anforderungen Kulturschaffender an Liegenschaften.

Finanzielle Rahmenbedingungen	Günstige Mietpreise
Organisatorische Rahmenbedingungen	Schnelle Verfügbarkeit der Räume, geringe Auflagen (Vertragsdauer, Kündigungsfristen), Freiheit in der Nutzungsgestaltung
Technische / funktionale Anforderungen	Licht, Schallisolierung, Raumhöhe Vielfältige Nutzungsmöglichkeiten
Umfeldbedingungen	Günstige Lage (Verkehrsanbindung, keine "empfindlichen" Anwohner)

3 Kommunales Immobilienmanagement

Kommunales Immobilienmanagement definiert BEYERSDORFF 2006 wie folgt:

„Kommunales Immobilienmanagement ist eine betriebliche Sekundärfunktion, deren Zweck darin besteht die raumbezogenen Voraussetzungen für den kommunalen Transformationsprozess zu schaffen. Die im Rahmen der Funktionserfüllung zu verrichtenden Aufgabenkomplexe sind die Beschaffung, die Verwaltung und die Veräußerung von Immobilieneigentum und Immobiliennutzungsrechten."

3.1 Funktionen öffentlicher Immobilien

Zur Bewältigung der kommunalen Aufgabengebiete werden Immobilien wie Verwaltungsgebäude, Schulen, Kindergärten, Sportanlagen, Logistikimmobilien, Krankenhäuser, Grünflächen und kulturelle Einrichtungen benötigt. Darüber hinaus haben öffentliche Liegenschaften auch eine stadtentwicklungspolitische Funktion (vgl. SCHULTE & SCHÄFERS 2006).

Die Kommune agiert nicht nur als Eigentümer ihrer Liegenschaften, sondern immer auch im gesamtstädtischen Auftrag. Somit ist im Unterschied zur Privatwirtschaft im öffentlichen Sektor nicht die Gewinnmaximierung als übergeordnetes Ziel anzusehen, sondern die Aufgabe der Daseinsvorsorge (SCHULTE & SCHÄFERS et al. 2006, BECKER 2009).

So übernimmt die Kommune durch die Bereitstellung einer immobilienwirtschaftlichen Infrastruktur, durch Vorschläge und Initiativen zur Mobilisierung von Raumpotenzialen eine wichtige Funktion in der Standortentwicklung und entscheidet über die Ausgestaltung ihrer Planungshoheit auch über eine mögliche Standortwahl von Kulturschaffenden mit (vgl. SCHULTE & SCHÄFERS 2006, BECKER 2009).

3.2 Aufgabenfelder des kommunalen Immobilienmanagements

Als Hauptaufgabenfelder der kommunalen Immobilienbewirtschaftung benennen PORTZ & DÜSTERDIEK 1999:

1. Organisation / Gebäude- und Raummanagement (Gebäudetechnische Optimierung)

- Bedarfsermittlung / -planung
- Bereitstellung von Räumen für die kommunale Nutzung
- Strategische Standortplanung
- An- und Verkauf bzw. Anmietung von Gebäuden und Räumen
- Ausstattung von Gebäuden

2. Bewirtschaftung kommunaler Immobilien (Betriebsoptimierung)

- Energieversorgung
- Abfall- und Abwasserentsorgung
- Gebäudereinigung
- Hausmeisterdienste

3. Liegenschaften und Grundstücke

- An- und Verkauf von Grundstücken
- An- und Vermietung von Grundstücken

4. Finanzierung

- Mittelbeschaffung / Vermögensfinanzierung
- Budgetierung
- Vermögenskontrolle und -übersicht

3.3 Optimierungsbedarf im kommunalen Immobilienmanagement

Im Bereich der Liegenschaften verbirgt sich in Zeiten der angespannten Haushaltslage ein nicht zu unterschätzendes Einsparpotenzial (vgl. SCHULTE & SCHÄFERS 2006). Laut PORTZ & DÜSTERDIEK 1999 stellen die Aufwendungen für die Bewirtschaftung und Instandhaltung kommunaler Immobilien nach den Personalkosten den zweitgrößten Kostenblock im kommunalen Haushalt dar.

Neben einer Neu-Organisation der Verwaltungsstruktur kann eine Effizienzsteigerung in der Immobilienbewirtschaftung in erheblichem Maße zu Einsparungen beitragen (PORTZ & DÜSTERDIEK 1999). BEYERSDORFF 2006 spricht im Zusammenhang mit Immobilienbewirtschaftungsdefiziten von qualitativer bzw. quantitativer Schlechtnutzung.

Es sollte abgewägt werden, ob ein Gebäude mit einer Nutzung an dem Ort benötigt wird, an dem es sich befindet, oder ein anderer Standort für die Nutzung ebenso geeignet wäre. So sind Grundstücke in Innenstadtbereichen weitaus teurer als jene in Randlagen. Ein Verwaltungsgebäude in zentraler Lage ohne Publikumsverkehr bezeichnet BEYERSDORFF 2006 als qualitative Schlechtnutzung.

Leerstand als Problem

Leerstand, nach BEYERSDORFF 2006 Ausdruck quantitaver Schlechtnutzung, bedeutet für den Eigentümer einer Immobilie unnötige Kosten. Ein leerstehendes Gebäude ist nicht befreit von Erhaltungs- und Unterhaltungsmaßnahmen, es bindet Kapital.
So stellt sich die Frage, ob bestimmte Gebäude oder Räume überhaupt benötigt werden. Es gilt, Leerstand zu vermeiden, indem die Bedarfe ermittelt und die Auslastung von Gebäuden optimiert werden. Durch Leerstandsreduzierung bzw. Flächenoptimierung können gebundene finanzielle Mittel freigesetzt werden (SCHULTE & SCHÄFERS 2006).
Kommunales Immobilienmanagement steht vor der Aufgabe neue Nutzungskonzepte für öffentliche Liegenschaften zu entwickeln.

… und Chance

Leerstand bietet aber auch Platz zum Experimentieren. So könnten Zwischen- und Mischnutzungen dazu beitragen, Leerstände zu reduzieren und zu vermeiden.

Zwischennutzung

Zwischennutzungen sind zeitlich begrenzte Nutzungen. Sie eignet sich zum Beispiel für Gebäude, deren Nutzung aufgegeben wurde und noch kein neues Nutzungskonzept vorliegt bzw. noch in Planung ist.

Zwischennutzung kann zu einer Aufwertung einer Immobilie führen, beispielsweise dadurch dass völlig neue Nutzungsmöglichkeiten erahnt werden. Ein Gebäude in genutztem Zustand wird deutlich positiver wahrgenommen, was die Vermarktungschancen steigen lässt. Kulturelle Zwischennutzung von Gebäudeleerstand kann zu einer Aufwertung der entsprechenden Stadtquarttiere führen, zur Imageverbesserung beitragen, Arbeitsräume (Ateliers, Proberäume, Ausstellungsräume) für die Kreativszene produzieren und Initialzündung für Entwicklungsprozesse sein. Zudem bietet sie Schutz vor Vandalismus und Verfall und trägt zu einer Verringerung der Unterhaltungskosten bei (vgl. HMWVL 2008).

Mischnutzung

Viele Räume und Gebäude werden nur an einem Teil des Tages genutzt. Hier könnte eine Mischnutzung zu mehr Effizienz führen (PORTZ & DÜSTERDIEK 1999). Umgesetzt wird dies bereits in der Nutzung von Schulgebäuden durch Volkshochschulkurse in den Abendstunden. Dies könnte weiter ausgebaut werden, indem unter anderem Kulturschaffenden (Vereine, Chöre, Tanzgruppen etc.) Platz zum Ausleben ihrer Kreativität bereitgestellt wird.

CAFM

Um sowohl qualitative, als auch quantitative Schlechtnutzung zu reduzieren oder ganz zu vermeiden, muss die Kommune ihre Nutzungskonzepte validieren und die Auslastung ihrer Räumlichkeiten überprüfen. Dazu bedarf es eines Systems, mit dem Daten über Nutzung und Auslastung auch in Bezug auf ihre räumliche Lage erhoben und ausgewertet werden können, wie es in den meisten Kommunen in Form eines CAFM (Computer Aided Facility Management) bereits existiert.

Geht es darum, Räumlichkeiten für Misch- und Zwischennutzung bereitzustellen, auch hier herauszufinden, inwieweit diese Räume/Gebäude dann tatsächlich genutzt werden, bedarf es eines Systems, auf das auch Externe (Bürger, Kulturschaffende) zu Informationszwecken, Buchungszwecken etc. zugreifen können.

4 Kommunikationsmittel

Stellt man die Interessen, Anforderungen und Bedürfnisse von Kulturschaffenden und Kommunalem Immobilienmanagement, welche in den vorangegangenen Kapiteln erörtert wurden, gegenüber, so fällt auf, dass beide Seiten voneinander profitieren könnten. Nur ist es so, dass erstere sich mit einem Ansuchen eher an das Kulturamt wenden würden, denn an das Liegenschaftsamt. Und letzterem fehlt ein Mittel, um seine Angebote zu kommunizieren.

Das Hessische Ministerium für Wirtschaft, Verkehr und Landesentwicklung (HMWVL) gibt zusammen mit dem Hessischen Ministerium für Wissenschaft und Kunst (HMWK) im 3. Hessischen Kulturwirtschaftsbericht Empfehlungen für die Gestaltung von Instrumenten im Vermittlungsprozess zwischen Eigentümern und Kulturschaffenden als Nutzern.
Die Erfassung der Leerstände wird als „elementare Voraussetzung der Aktivierung von Raumpotenzialen für die Kulturwirtschaft“ angesehen (HMWVL 2008). Konkreter umfasst dies den „Aufbau und die Pflege einer Datenbank leer stehender Flächen, Gebäude […] mit Informationen zu Größe, Lage, Anzahl der Räume, Ausstattung, Mietpreis, Ansprechpartner einschließlich Fotos und / oder Grundrisse“ (HMWVL 2008). Zudem sollte solch eine Datenbank für Eigentümer und potenzielle Nutzer zugänglich gemacht werden (HMWVL 2008). Als geeignetes Kommunkationsmedium bietet sich freilich das Internet an.

4.1 Kommunikationsmedium Internet

Das Internet hat sich in den letzten zehn Jahren zu einem der wichtigsten Kommunikationsmittel entwickelt. Die aktuelle Studie (N)ONLINER Atlas 2010 der Initiative D21 zeigt, dass im Jahr 2010 72 % der Bevölkerung in Deutschland online waren (Onliner). Der Anteil derer, die das Internet noch nicht nutzen (Offliner), ist seit 2001 von 52,5 % auf 24,2 % im Jahr 2010 gesunken (Abb. 2).
Durch den kontinuierlichen Ausbau der Infrastrukturen (Verfügbarkeit schneller Internetverbindungen) und dem technologischen Fortschritt des Internets sind die Möglichkeiten der Interaktion enorm gestiegen. Nachdem das World Wide Web lange Zeit hauptsächlich zu Informationszwecken genutzt wurde, wird im sogenannten Web

2.0 vermehrt auf Interaktivität gesetzt. Begriffe wie Cloud Computing, Soziale Netzwerke, Wikis oder Weblogs gehen damit einher.

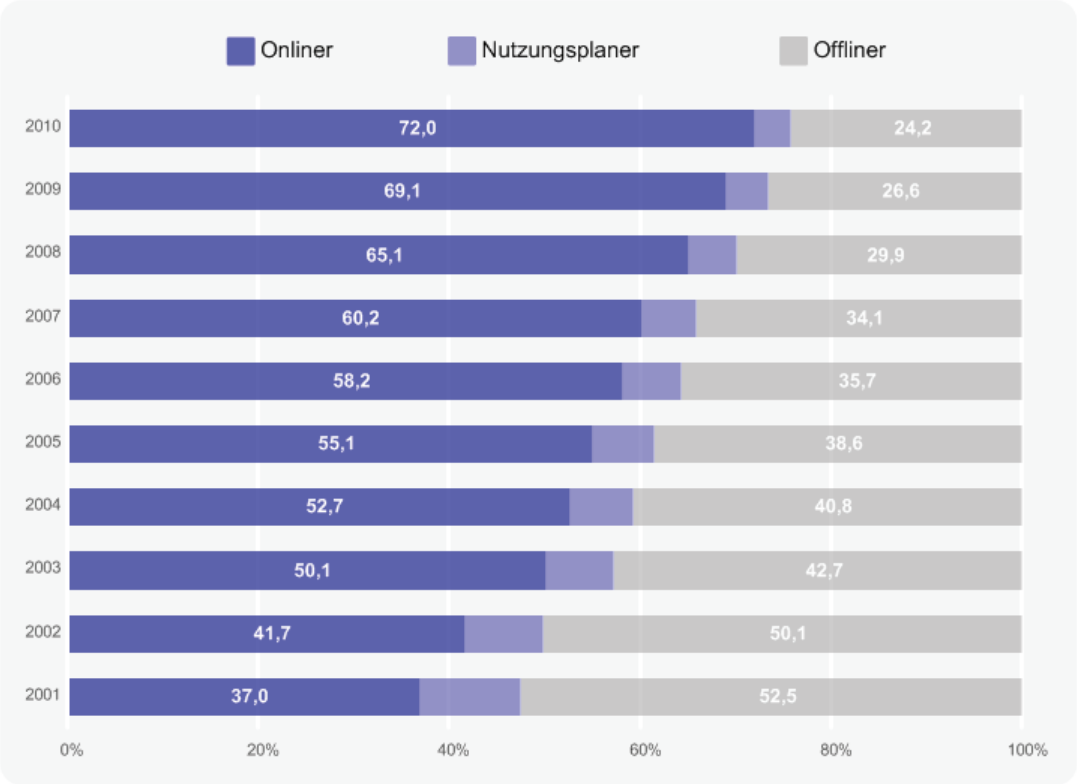


Abb. 2 Entwicklung der Internetnutzung in Deutschland (geändert nach (N)Onliner Atlas 2010).

4.2 WebGIS als Kommunikationsmittel

Geoinformationssysteme (GIS) werden dazu eingesetzt, raumbezogene Daten zu verwalten, zu erfassen, zu visualisieren und zu analysieren. Ein GIS, das sich Internettechnologien bedient, wird als WebGIS oder auch Internet-GIS bezeichnet (vgl. KORDUAN & ZEHNER 2008).

Die bekanntesten Webmapping-Anwendungen sind wohl Google Maps und Open Street Map (OSM). Auch Öffentliche Verwaltungen bieten, vermehrt seit Inkrafttreten der EU-Richtlinie INSPIRE (Infrastructure for Spatial Information in the European Community) 2007 und der damit einhergehenden Verpflichtung zur Bereitstellung von Geodaten, räumliche Daten in Internet-GIS an. Die meisten WebGIS-Anwendungen dienen der Information, die Inhalte sind vordefiniert und durch den Betrachter / Nutzer nicht modifizierbar (vgl. STEINMANN ET AL. 2004). Dynamische Anwendungen, die das Editieren und Analysieren von Geodaten ermöglichen, sind eher selten anzutreffen.

5 WebGIS - Theoretische Grundlagen

5.1 Client-Server-Architektur

Einem WebGIS liegt die Client-Server-Architektur, auf der auch das Internet basiert, zugrunde. Ein Client sendet eine Anfrage (Request) an einen Server, die vom Server beantwortet wird (Response). Dabei kann ein Server mehrere Clients bedienen (Abb. 3).

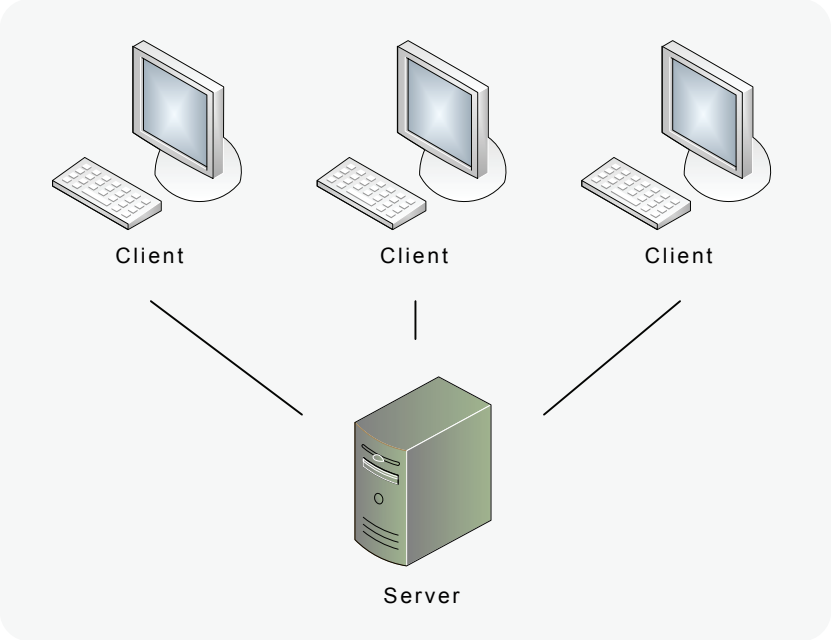


Abb. 3 Client-Server-Architekur.

Die Systemkomponenten einer WebGIS-Architektur umfassen einen Webserver, einen Karten-Server, einen Datenbank-Server und einen Web-Mapping-Client.

Fordert ein Client eine Karte an, so wird der Request vom Webserver an den Kartenserver übergeben. Der Kartenserver, eine serverseitige Software, mit der Geodaten über das Internet bereitgestellt werden, kann nicht direkt mit dem Client kommunizieren. Die Geodaten werden in einer (Geo-)Datenbank verwaltet, die mit dem Kartenserver in Verbindung steht.

5.2 IETF und W3C – Standards

Die IETF (Internet Engineering Task Force) ist eine offene internationale Organisation, die bestrebt ist, die Funktionsweise des Internet stetig zu verbessen. Ins Leben gerufen wurde sie 1986 von einer Gruppe von Forschern in San Diego. Hauptaugenmerk der IETF liegt auf der Standardisierung von Internetprotokollen.

Das W3C (World Wide Web Consortium) ist ein Zusammenschluss verschiedener internationaler Firmen, Institute und Organisationen und wurde 1994 am MIT (Massachusetts Institute of Technology) in Cambridge von Tim Berners-Lee gegründet, der als Erfinder des WWW gilt und auch den Vorsitz inne hat.
Ziel des W3C ist es, Web-Standards zu entwickeln, um dem „World Wide Web dadurch seine vollen Möglichkeiten zu erschließen, dass Protokolle und Richtlinien entwickelt werden, die ein langfristiges Wachstum des Web sichern.“ (http://www.W3C.de)

Zu den Errungenschaften von IETF und W3C zählen die Protokolle IP, TCP und HTTP, URI, HTML, XML und DOM, auf welche im Folgenden näher eingegangen wird.

5.2.1 TCP / IP-Protokollstapel

Das Internet Protocol (IP, RFC 2460) stellt die Internetschicht im TCP/IP-Protokollstapel dar und dient dazu, Computer in einem Netzwerk zu adressieren.
Das TCP (Transmission Control Protocol, RFC 793) gibt an, auf welche Art Daten über das Netzwerk ausgetauscht werden und bildet im TCP/IP-Protokollstapel die Transportschicht.

Das Hypertext Transfer Protocol (HTTP, RFC 2616) wird als Kommunikationsmittel zwischen einem Client und einem Server eingesetzt. Es ist in der Anwendungsschicht des Protokollstapels angesiedelt. Ein Client fordert mit einem HTTP- Request Daten von einem Server, die der Server mit einem Response liefert. Die OGC-Webservices (s. 5.4) nutzen für die Kommunikation die GET- und die POST-Methode.

5.2.2 URI – Uniform Resource Identifier

Der Uniform Resource Identifier ist aktuell im RFC 3986-Standard definiert und dient dazu, eine Web-Resource, etwa eine Internetseite oder einen Web-Service, anhand einer Zeichenfolge zu identifizieren. URIs werden in zwei Gruppen unterteilt: URL (Uniform Resource Locator) und URN (Uniform Resource Name).
Die URL bestimmt eine Ressource über den Ort mittels eines Netzwerkprotokolls (HTTP oder FTP) und wird oft als Synonym für eine Internetadresse gebraucht.
Ein URN kennzeichnet eine Ressource durch einen Namen, unabhängig von ihrem Speicherort. Die URN bleibt auch dann bestehen, wenn sich der Ort der Ressource ändert.

5.2.3 HTML – Hypertext Markup Language

Die Auszeichnungssprache HTML wurde 1989 von einer Gruppe um Tim Berners-Lee am CERN (Europäische Organisation für Kernforschung) in Genf entwickelt und vom W3C als Standard festgelegt (MUSCIANO 2003). HTML beschreibt die Struktur einer Webseite, teilt dem Browser mit, wie der Inhalt einer Seite angezeigt werden soll.
Ursprünglich konnten mit HTML nur Text und Hyperlinks dargestellt werden. Die Sprache wurde seither ständig von den unzähligen Nutzern weitentwickelt. Die aktuelle Version 4.01 unterstützt unter anderem auch das Einbetten von Vektor-Grafiken (SVG) und Skripten (z.B. JavaScript), um Web-Seiten dynamischer zu gestalten

5.2.4 XML – Extensible Markup Language

XML ist eine vom W3C entwickelte Auszeichnungssprache, die eine hierarchische Struktur für den Aufbau von Daten vorgibt und sowohl für die Speicherung, als auch die Modellierung von Daten eingesetzt werden kann. Vor allem im Internet hat sich XML als plattformunabhängiges Austauschformat etabliert (SKULSCHUS 2008). Im Prinzip handelt es sich um ein Text-Format, dessen Syntax derer von HTML auf den ersten Blick sehr ähnlich ist. Während HTML vornehmlich auf die Darstellung abzielt, lässt XML auch eine inhaltliche Strukturierung zu und trennt Inhalt und Darstellung weitgehend voneinander (DEHNHARDT 2001).

Mit der Zeit entstanden XML-Sprachen für die unterschiedlichste Bereiche, wie z.B. SVG (Scalable Vector Graphics) für das Speichern und Darstellen von Vektorgrafiken, MathML für mathematische Formeln oder SMIL (Synchronized Multimedia Integration Language) für zeitsynchronisierte multimediale Inhalte. Für die Verarbeitung von Geodaten wurde unter anderem GML (s. 5.4.335) standardisiert.
Werden mehrere verschiedene XML-Sprachen in einem Dokument verwendet, empfiehlt sich die Nutzung von Namensräumen (engl. Namespace), um einzelne Elemente eindeutig identifizieren zu können. Namensräume werden durch die Angabe einer URI im xmlns-Attribut festgelegt und können jedem Element als Präfix vorangestellt werden.

5.2.5 DOM – Document Object Model

Das DOM ist ein Standard des W3C, der eine sprach- und plattformunabhängige Schnittstelle darstellt, welche den Zugriff auf HTML- / XML-Elemente regelt. Durch die Vergabe einer ID kann jedes Element in einem Webseiten-Dokument eindeutig identifiziert bzw. angesprochen, verändert oder gelöscht werden. Dadurch kommt dem DOM unter anderem in der JavaScript-Programmierung eine zentrale Bedeutung zu (W3C 2004).Abb. 4 zeigt die grafische Darstellung des DOM einer Tabelle.

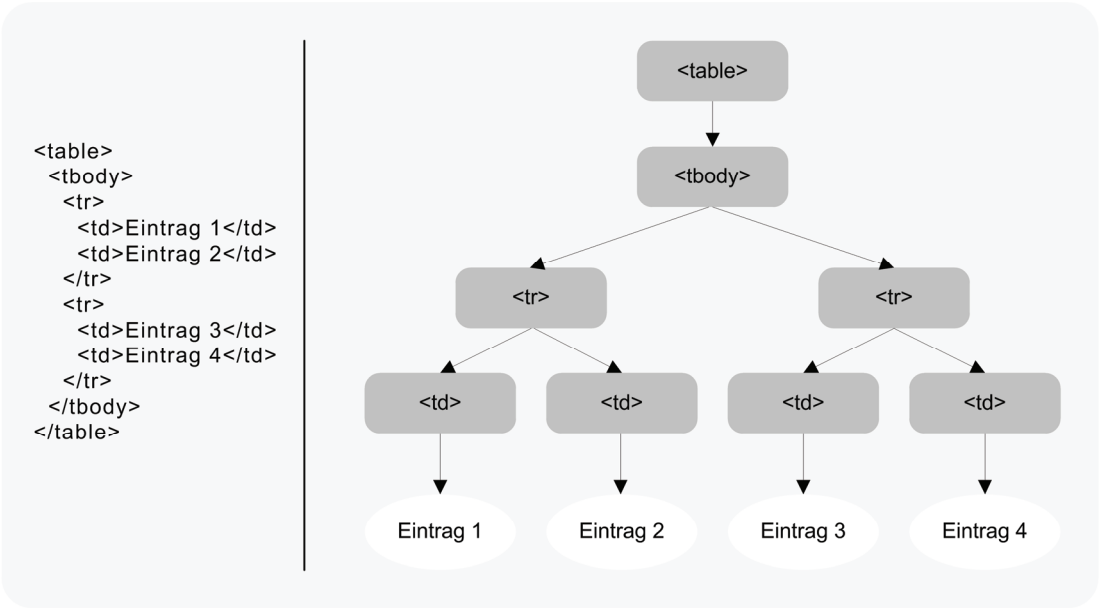


Abb. 4 DOM – Grafische Darstellung des DOM einer Tabelle.

5.3 Weitere Web-Technologien

5.3.1 PHP

Die Skriptsprache PHP (mit der Bedeutung Hypertext Preprocessor oder auch Personal Home Page Tools) findet bei der Programmierung dynamischer Webseiten weite Verbreitung. Kennzeichnend für PHP ist die breite Datenbankunterstützung.
PHP Skripte können in HTML eingebettet werden, werden aber im Gegensatz zu clientseitigen Skripten (s. JavaScript) auf dem Server ausgeführt. Der Quelltext kann nicht vom Client eingesehen werden Ein Interpreter generiert aus dem PHP-Quelltext den Code (HTML, PDF, etc.), der an den Client ausgegeben wird.

5.3.2 JavaScript

JavaScript ist eine clientseitige Skriptsprache, die dazu dient, Webseiten mit dynamischen Inhalten zu versehen (DEHNHARDT 2001). Durch das Erstellen von aktiven Inhalten mit JavaScript können Webseiten, die im Browser angezeigt werden, nachträglich verändert werden, ohne dass sie nochmals neu vom Server geladen werden müssen. Das kann das Anpassen einer Seite an die Größe des Browserfenster sein oder die Verarbeitung von Formularen oder eben die Gestaltung dynamischer Karten sein.

JavaScript kann auf einfache Art und Weise in HTML eingebettet werden, wie folgendes Beispiel für die Berechnung und die Ausgabe des Quadrates der Zahl 10 demonstriert.

```
<html>
 <head>
  <script type="text/javascript">

   function berechnen(){
    var zahl;
    var erg_quadrat;
    zahl = 10;
    erg_quadrat = zahl * zahl;
    alert("Das Quadrat von " + zahl + " ist " + erg_quadrat);
   }

  </script>
 </head>
 <body>
  <h1>JavaScript Beispiel</h1>
  <input type="button" onclick="berechnen()" value="berechnet das
   Quadrat von 10">
 </body>
</html>
```

Mit der ursprünglichen Bezeichnung LiveScript wurde JavaScript von der Firma NETSCAPE für deren Browser Netscape Navigator entwickelt, kurz darauf wurden auch Mozillas Firefox und Microsofts Internet Explorer mit einem Interpreter ausgestattet.
Seit der Festlegung als Standard durch Ecma International (European Computer Manufacturer's Association), einer privaten, internationalen Organisation, lautet der offizielle Name ECMAScript, der indes weniger bekannt ist.
Anfangs tauchten bei der Ausführung von JavaSript in unterschiedlichen Browsern Probleme auf. Mit deren Weiterentwicklung sind diese weitgehend beseitigt worden, gelegentlich kann es noch vorkommen, dass ein Skript nicht korrekt interpretiert wird.
JavaScript kann auf das DOM (s. 5.2.5) zugreifen.

Alle Vorzüge, die aktive Inhalte mit sich bringen, tragen allerdings auch ein gewisses Sicherheitsrisiko. ECKERT 2009 empfiehlt JavaScript im Browser zu deaktivieren, da eine „Vielzahl von Sicherheitsproblemen" entstehen kann. So stehen beispielsweise Attacken durch Cross-Site-Scripting (XSS) ganz oben auf der Liste der häufigsten Sicherheitslücken (ECKERT 2009). Dabei wird bösartiger Code auf einem Server abgelegt, die beim Zugriff vom Server an den Nutzer übertragen und schließlich im Client ausgeführt werden. Ziel solcher Angriffe ist meistens, sensible Daten zu stehlen.

STEYER 2007 hingegen ordnet das Risiko aufgrund der doch beschränkten Möglichkeiten der Skriptsprache als nicht bedrohlich hoch ein. So können mit JavaScript weder auf native Bibliotheken in einem Betriebssystem oder auf die Festplatte eines Anwenders zugegriffen, noch Folgeprogramme direkt gestartet werden.
Ein hundertprozentiger Schutz vor dem Zugriff von außen kann derzeit nicht gewährleistet werden, man sollte trotzdem versuchen, potenzielle Risiken zu minimieren. Wichtige Instrumente dazu sind die Prüfung der übergebenen Datenwerte auf ihre Plausibilität (vgl. WASSERMANN 2007) und die seit Version 1.1 implementierte Same Origin Policy. Letztere erlaubt es JavaScript nur dann auf Eigenschaften einer anderen Webseite zuzugreifen, wenn diese die gleiche Herkunft haben, wie das Dokument, dass Skript beinhaltet (FLANAGAN 2007). Nach der anfänglichen Skepsis hat sich JavaScript mit der Einführung von AJAX (s. 5.3.3) und der Popularität von Web 2.0-Anwendungen letztlich in der Web-Programmierung etabliert.

5.3.3 AJAX

AJAX ist die Abkürzung für Asynchronous Javascript And XML. Dahinter verbirgt sich eine Technologie, die es ermöglicht, eine HTTP-Anfrage an einen Server zu richten, während die HTML-Seite im Browser geöffnet ist. Der Datenaustausch verläuft dabei im Hintergrund. So können Teile einer Seite verändert werden, ohne dass die Webseite komplett neu geladen werden muss.
Einen Grundbestandteil der Ajax Technik bildet der XMLHttpRequest (XHR), welcher aus JavaScript heraus verwendet werden kann. Die vom Server in XML oder PlainText zurück gelieferten Daten werden in das DOM der angezeigten Seite eingebunden.

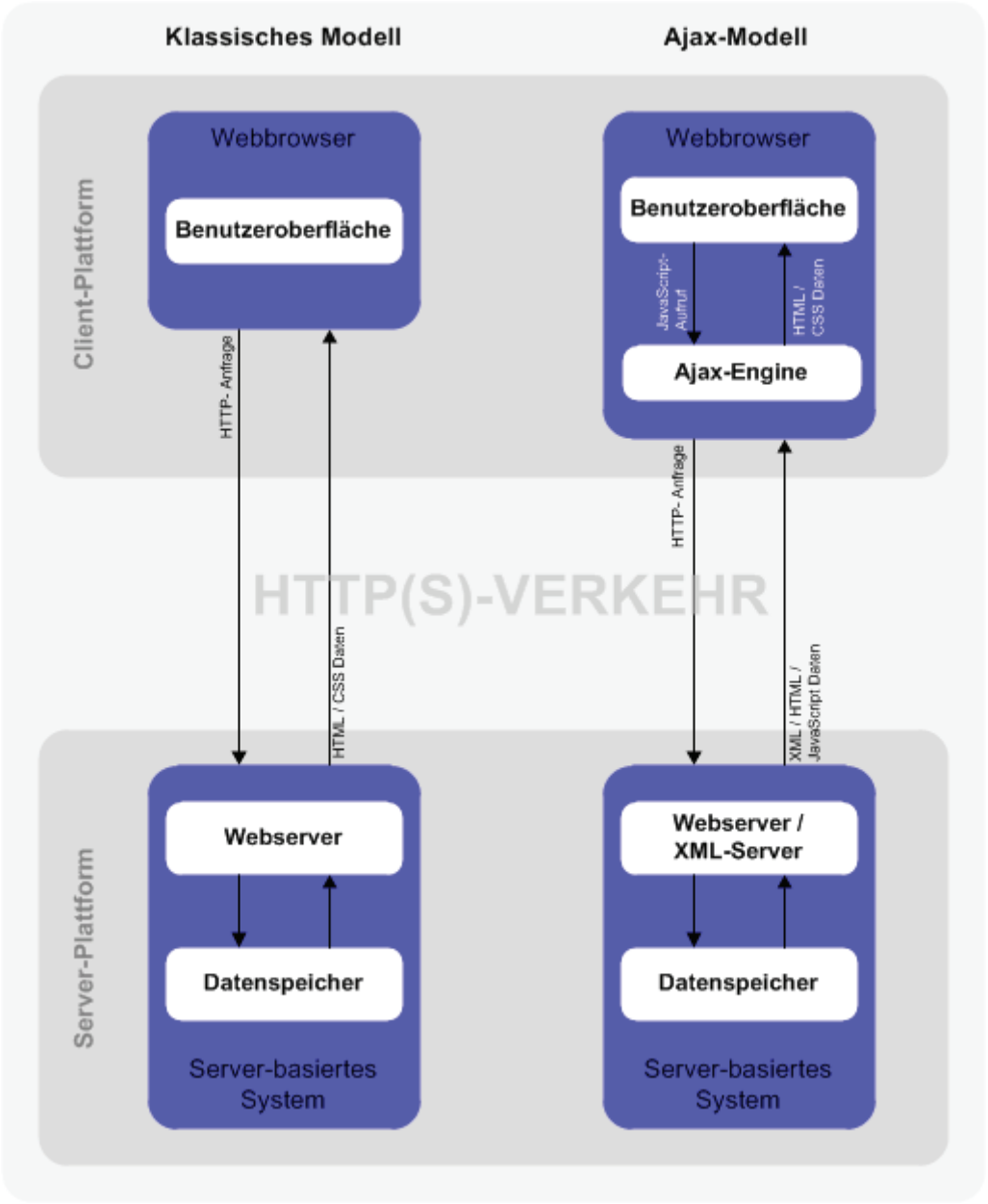


Abb. 5 Vergleich von Klassischem und Ajax-Modell bei Webanwendungen
(geändert nach WIKIPEDIA).

5.4 OGC – Standards

Das Open Geospatial Consortium (OGC) ist ein internationaler Zusammenschluss von Industrie, Unternehmen, Regierungsorganisationen und Universitäten mit dem Ziel der Interoperabilität raumbezogener Daten durch die Festlegung offener Standards für Schnittstellen und Datenformate.

Das Konsortium, derzeit bestehend aus 398 Mitgliedern (Stand 09/2010), entwickelte sich 1994 aus der Open GRASS Foundation (OGF), welche wiederum im Jahre 1992 aus der GRASS User Community hervorgegangen ist. GRASS (Geographic Resources Analysis Support System) kann als eines der ersten globalen Open Source Software Projekte angesehen werden, es wurde in den 1980er Jahren von US-amerikanischen Behörden entwickelt und anfangs vor allem in Universitäten eingesetzt.
(http://www.opengeospatial.org/)

Das OGC versteht sich selbst als globales Forum für Entwickler und Anwender von raumbezogenen Daten und Diensten und treibt so die (Weiter-) Entwicklung internationaler OpenGIS Standards voran. OpenGIS® ist mittlerweile eine eingetragene Marke des OGC und steht für interoperabel nutzbare Geodaten und Services.

Die strategischen Ziele des OGC sind (nach http://www.opengeospatial.org/):

1. kostenfreie und offen zugängliche Standards anzubieten, die es ermöglichen, raumbezogene Inhalte und Services unkompliziert in Geschäftsprozesse, Unternehmens-EDV und Internet zu integrieren und somit einen Mehrwert für Mitglieder und Nutzer zu schaffen
2. bei der Erstellung und Etablierung von Standards weltweit führend zu sein
3. die Übernahme von raumbezogenen Architekturen in die Unternehmens-umgebung zu erleichtern
4. Standards zu fördern, um die Bildung neuer innovativer Märkte und raumbe-zogener Technologien zu unterstützen
5. durch die Zusammenarbeit im Konsortium den Markt im Bereich Interoperabi-lität an die Forschung anzugleichen

5.4.1 WMS – Web Map Service

Der Web Map Service ist ein Dienst, der Karten über das Internet liefert. Eine Karte ist das Ergebnis einer Anfrage an einen Kartenserver. Sie wird dynamisch aus Geodaten erzeugt und in Form eines gerenderten Rasters (JPG, GIF, PNG) ausgegeben (OGC 2006a). Der Standard, der aktuell in Version 1.3.0 verfügbar ist, ist wohl der am häufigsten eingesetzte OGC-Service. Der große Vorteil eines WMS ist, dass viele verschiedene Clients gleichzeitig bedient werden können. So können beispielsweise Luftbilder einem großen Nutzerkreis zugänglich gemacht werden. Die Kommunikation basiert auf dem Hypertext Transfer Protocol (HTTP).

Koordinatensysteme

Der WMS nutzt zwei Klassen von Koordinatensystemen, ein Map CS (Coordinate System) für die Präsentation einer vom WMS erzeugten Karte sowie ein Layer CRS (Coordinate Reference System), welches auf die Quelldaten zurückgreift. Für die Darstellung transformiert der WMS die geographische Lageinformation vom Layer CRS in das Map CS.

Map CS

Unter dem Map CS versteht man ein Koordinatenreferenzsystem für eine Karte, welches vom WMS erzeugt wird. Es wird mittels der ISO 19111 Terminologie definiert, die Identifikation erfolgt über das Label „CRS:1“.

Eine WMS Karte ist ein rechteckiges Pixelraster. Das Map CS weist diesem Raster Koordinatenwerte zu. So verläuft in horizontaler Richtung eine i-Achse, in vertikaler Richtung eine j-Achse. Der Koordinatenursprung liegt in der oberen linken Ecke der Karte, die Werte nehmen nach rechts bzw. unten zu, negative Werte gibt es nicht.

Die Größe der darzustellenden Karte wird mit den Parametern WIDTH (Anzahl der Pixel entlang der i-Achse) und HEIGHT (j-Achse) angegeben.

Layer CRS

Das Layer CRS ist ein horizontales Koordinatenreferenzsystem Es beinhaltet die Information, in welchem Koordinatensystem die Daten vorliegen. Das Layer CRS erscheint als <Bounding Box> Element in den Service Metadaten sowie als CRS-Parameter in der GetMap-Afrage.

Der WMS muss mindestens ein Layer CRS unterstützen, es können aber aufgrund der Vielzahl von Koordinatensystemen auch viele verschiedene sein. Dies ist bei der gleichzeitigen Nutzung von Daten von verschiedenen Servern von großer Bedeutung.

Eindeutig identifizieren lassen sich Layer CRS über einen Character String in Form:

- eines Labels: umfasst ein Namespace-Präfix, einen Doppelpunkt, einen numerischen oder einen Zeichen (string)- Code und z.T. ein Komma gefolgt von zusätzlichen Parametern. Der WMS definiert 3 Namespaces: CRS, EPSG und Auto2.

  CRS Namespace

  Der CRS Namespace bezeichnet Koordinatensysteme, die ISO 19111 konform sind und im Anhang B des internationalen Standards aufgeführt sind.

  EPSG Namespace

  Der EPSG Namespace bezieht sich auf die von der European Petroleum Survey Group definierten Codes für Koordinatenreferenzsysteme. Ihre Verwendung hat sich weltweit etabliert. Die 4-5stelligen Codes gewährleisten eine eindeutige Identifikation eines Koordinatenreferenzsystems. Der EPSG-Namespace für das WGS84 lautet beispielsweise EPSG:4326.

- einer URL, welche auf eine öffentlich zugängliche Datei, die die Definition des Koordinatensystems enthält und zudem ISO 19111 konform ist, verweist.

Das Layer CRS besitzt zwei Achsen, die x- und die y-Achse. In der Definition des CRS ist die x-Achse immer die erste, die y-Achse immer die zweite. Der WMS berücksichtigt die Anordnung der Achsen, den Koordinatenursprung und die Richtung des Layer CRS beim Darstellungsvorgang, bei dem die geografische Information des Layer CRS in das Map CS übertragen wird.

Operationen

Der WMS unterstützt die Operationen GetCapabilities, GetMap und GetFeatureInfo.

GetCapabilities

Der GetCapabilities-Aufruf liefert Metainformationen über den angebotenen WMS in Form einer XML-Datei. Diese umfassen Informationen über unterstützte Koordinatensysteme, verfügbare Layer, Dateiformate für das zu erzeugende Kartenbild, räumliche Ausdehnung der verfügbaren Geodaten, eventuell Angaben über den Betreiber des Dienstes sowie weitere zusätzliche Informationen. Tab. 2 zeigt die Parameter des GetCapabilities-Requests.

Tab. 2 Parameter des GetCapabilities-Requests (geändert nach OGC 2006a)**.**

REQUEST Parameter	obligatorisch	optional
VERSION=version		X
SERVICE=WMS	X	
REQUEST=GetCapabilities	X	
FORMAT=MIME_type		X
UPDATESEQUENCE=string		X

Beispiel für einen GetCapabilities-Request:

```
http://192.168.133.128/geoserver/wms?request=getCapabilities
```

GetMap

Der GetMap-Request liefert das georeferenzierte Rasterbild vom Server an den Client. Innerhalb der Anfrage können verschiedene Parameter (Tab. 3) an den Server übergeben werden, wie etwa das zurückgegebene Dateiformat, die Größe der Karte, der gewünschte Kartenausschnitt, das zugrundeliegende Koordinatensystem, die Layer, die angezeigt werden und ob diese transparent dargestellt werden sollen.

Beispiel für einen GetMap-Request:

```
http://192.168.133.128/geoserver/wms?bbox=-130,24,-66,50&Format=image/
png&request=GetMap&layers=cite:raum&width=550&height=250&srs=EPSG:31467
```

Tab. 3 Parameter des GetMap- Requests (geändert nach OGC 2006a).

REQUEST Parameter	obligatorisch	optional
VERSION=1.3.0	X	
REQUEST=GetMap	X	
LAYERS=layer_list	X	
STYLES=style_list	X	
CRS=namespace:identifier	X	
BBOX=minx,miny,maxx,maxy	X	
WIDTH=output_width	X	
HEIGHT=output_height	X	
FORMAT=output_format	X	
TRANSPARENT=TRUE|FALSE		X
BGCOLOR=color_value		X
EXCEPTIONS=exception_format		X
TIME=time		X
ELEVATION=elevation		X
Other sample dimension(s)		X

GetFeatureInfo (optional)

GetFeatureInfo dient dazu, durch das Klicken auf einen Punkt in der Karte zusätzliche Informationen zu einem Kartenlayer zu erhalten. Die Operation übernimmt dabei die Pixelkoordinaten, also die Koordinaten des Map CS, in einen in den GetFeatureInfo eingebetteten GetMap Request (OGC 2006b).

Ob und für welche Layer GetFeatureInfo verfügbar ist, wird mit dem GetCapabilities-Response beantwortet. Das Ausgabeformat der Zusatzinformation ist durch das OGC nicht festgelegt, zu achten ist auf die Verwendung einen gültigen MIME-Typs (Donaubauer 2004). Tab. 4 beinhaltet die Parameter des GetFeatureInfo Requests.

Tab. 4 Parameter des GetFeatureInfo Requests (geändert nach OGC 2006).

REQUEST Parameter	obligatorisch	optional
VERSION=1.3.0	X	
REQUEST=GetFeatureInfo	X	
map request part	X	
QUERY_LAYERS=layer_list	X	
INFO_FORMAT=output_format	X	
FEATURE_COUNT=number		X
I=pixel_column	X	
J=pixel_row	X	
EXCEPTIONS=exception_format		X

5.4.2 WFS – Web Feature Service

Während der WMS Karten als Rasterbilder überträgt, kann der WFS Geodaten in Form einer GML-Datei übermitteln. GML ist ein auf XML basierendes Austauschformat für Geodaten und wurde vom OGC als OpenGIS® Geography Markup Language (GML) Encoding Standard festgelegt (s.5.4.3).
Die aktuelle Version 2.0 des OGC WFS-Standards wird von der ISO als ISO 19142 geführt. Diese ist in den OGC-konformen Kartenservern derzeit noch nicht implementiert, sie greifen auf Version 1.0 bzw. 1.1.0 zurück. Aus diesem Grund soll hier auf beide Versionen eingegangen werden.
Entscheidend für einen WFS ist, dass der Nutzer auf Vektordaten zugreifen kann, diese abfragen und manipulieren, eigene erzeugen oder löschen kann.
In Version 1.1.0 der Spezifikation werden drei WFS-Klassen nach Unterstützung der durch den Standard vorgegebenen Operationen unterschieden. In Version 2.0 erfolgt die Unterteilung über Conformance-Klassen (Tab. 5).

Operationen

Der Service nutzt zum Ausführen der Operationen KVP (Keyword Value Pair)-Encoding mit HTTP GET Methode bzw. XML-Encoding mit der HTTP Post-Methode, mit Ausnahme der Transaction- und der CreateStoredQuery-Operation, welche nur XML-Encoding unterstützen.

Ein Beispiel-Request mit KVP-Encoding ist "REQUEST=GetCapabilities" mit REQUEST als Keyword und GetCapabilities als Value.

Tab. 5 Einteilung der WFS-Typen in Vers. 1.1.0 und 2.0 nach der Verfügbarkeit der Operationen.

Vers. 1.1.0 **WFS-Class**	**Vers. 2.0** **Conformance-Class**
	Simple WFS - GetCapabilities - DescribeFeatureType - ListStoredQueries - DescribeStoredQueries - GetFeature (nur StoredQuery action) Server sollte außerdem mind. eine der Conformance-Klassen HTTP GET, HTTP POST oder SOAP unterstützen
Basic WFS - GetCapabilities - DescribeFeatureType - GetFeature	**Basic WFS** *- Simple WFS conformance class erweitert um die Operationen:* - GetFeature (mit Query action) - GetPropertyValue
Xlink WFS *- Basic WFS erweitert um die Operation:* - GetGmlObject	
Transaction WFS *- Basic WFS erweitert um die Operationen:* - Transaction - LockFeature (optional) - GetGmlObject (optional)	**Transactional WFS** *- Basic WFS conformance class erweitert um die Operation:* - Transaction
	Locking WFS *- Transactional WFS erweitert um mind. eine der Operationen:* - GetFeatureWithLock - LockFeature
	HTTP GET Server implementiert KVP encoding für die Operationen
	HTTP POST Server implementiert XML encoding für die Operationen
	SOAP Server implementiert XML-Anfragen und Ergebnisse

GetCapabilities

Der GetCapabilities-Aufruf liefert Informationen über die Fähigkeiten des Dienstes, analog zum WMS hier mit einer Auflistung der abfragbaren Features Types.

Der Response des WFS 1.1.0 enthält die Sektionen (OGC 2005b):

- Service Identification section (Information über den WFS)
- Service Provider section (Metadaten über den Betreiber des WFS)
- Operation Metadata section (Metadaten über implementierte Operationen)
- FeatureType list section (Liste der verfügbaren Feature Types)
- ServesGMLObjectType list section
- SupportsGMLObjectType list section
- Filter capabilities section (optional, Liste der verfügbaren Filter des Filter Encoding Standards)

In der Version 2.0 wird auf die OGC Web Services Common Specification (OGC 06-121r3) verwiesen. Darin ist die GetCapabilities Operation, die Bestandteil aller OGC Web Services ist, allgemein beschrieben. Die dort aufgeführten Sektionen des Response-Dokuments sind für alle Services gültig. Zusätzliche für den WFS 2.0 spezifische Sektionen sind im WFS-Standard definiert.

Sektionen aus der OGC Web Services Common Specification (OGC 2006c):

- ServiceIdentification section (Metadaten über den Server)
- ServiceProvider (Metadaten über den Betreiber des Servers)
- OperationsMetadata (Metadaten über die verfügbaren Operationen)
- Contents (Metadaten über die verfügbaren Daten)

Sektionen aus dem OpenGIS Web Feature Service 2.0 Interface Standard (OGC 2010b):

- WSDL section (optional, beschreibt die verfügbaren Operationen im Web Service Description Language-Format)
- FeatureType section (Liste der verfügbaren Feature Types)
- Filter capabilities section (Liste der verfügbaren Filter, gemäß OpenGIS Filter Encoding 2.0 Encoding Standard / ISO 19143)

DescribeFeatureType

Um auf die Daten zugreifen zu können, muss der Nutzer die Datenstruktur der einzelnen Feature Types kennen. Mit der Operation DescribeFeatureType liefert der Server ein GML-Schema des Feature Typs.

Tab. 6 KVP encoding der DescribeFeatureType-Operation
(geändert nach OGC 2010b).

URL Komponente	obligatorisch	optional
REQUEST=DescribeFeatureType	X	
TYPENAME		X
OUTPUTFORMAT		X

GetFeature

Mit der GetFeature-Anfrage wird eine Feature-Auswahl aus einem Datenstore im gml-Format zurückgegeben. Durch die Angabe entsprechender Parameter kann die Feature-Menge eingeschränkt werden.

Tab. 7 listet die möglichen Elemente des GetFeature Requests auf. Die Identifikation einzelner Features erfolgt anhand der ID oder des Typs. Weitere Abfragekriterien, räumliche wie attributive, können im <filter>-Element innerhalb des <query>-Elements formuliert werden.

Aus der ursprünglich für den WFS entwickelten Beschreibung des Filter-Standards ist die Filter Encoding Implementation Specification hervorgegangen, da der Filter-Standard auch für andere Services (z.B. Web Coverage Service) eingesetzt werden kann.

Tab. 7 KVP-encoding der GetFeature-Operation (geändert nach OGC 2010b und OGC 2010c).

URL Komponente	Beschreibung
Common Keywords (REQUEST=GetFeature)	*Zusätzliche Parameter:* NAMESPACES[1], VSPs[1]
Standard Presentation Parameters	STARTINDEX[1], COUNT[1], OUTPUTFORMAT[1], RESULTTYPE[1]
Standard Resolve Parameters	RESOLVE[1], RESOLVEDEPTH[1], RESOLVETIMEOUT[1]
Adhoc Query Keywords (gegenseitig ausschließend mit Stored Query Keywords)	TYPENAMES, ALIASES[1], SRSNAME[1]
	Projection clause: PROPERTYNAME[1]
	Selection clause: FILTER[1], FILTER_LANGUAGE[1], RESOURCEID[1], BBOX[1]
	Sorting clause: SORTBY[1]
Stored Query Keywords (gegenseitig ausschließend mit Adhoc Query Keywords)	STOREDQUERY_ID, storedquery_parameter=value[1]

[1] optional

GetGmlObject

GetGmlObject ist eine optionale WFS-Operation. Wird ein Feature oder ein Element mittels einer ID abgefragt, antwortet der Server mit einer GML-Datei. Diese Operation existiert in Version 2.0 nicht mehr.

Transaction

Beim Transactional WFS (WFS-T) ist zusätzlich die Operation Transaction implementiert. Sie ermöglicht einem Client das Einfügen, Ändern und Löschen von Features.
Die Transaction-Operation wird mit XML-Encoding beschrieben. Tab. 8 zeigt die Elemente des Transaction-Requests. Die Elemente <Insert>, <Update> und <Delete> werden innerhalb des <Transaction>-Elementes verwendet.

Dabei können durchaus mehrere Elemente innerhalb eines <Transaction>-Elementes vorhanden sein. Das Element <LockID> kann optional eingesetzt werden, um zu bearbeitende Feature Instanzen zu sperren.

Tab. 8 Elemente des Transaction Requests.

Element	Beschreibung
<Transaction>	Enthält die Elemente <Insert>, <Update>, <Delete>
<Insert>	Fügt eine neue Feature Instanz eines Feature Types ein
<Update>	Ändert ein Feature
<Delete>	Löscht eine Feature Instanz
<LockID>[1]	Sperrt eine Feature Instanz zur Bearbeitung

[1] optional

LockFeature

Die optionale Operation LockFeature sperrt ein Feature und stellt dadurch sicher, dass es nicht gleichzeitig von einem anderen Client bearbeitet werden kann.

Tab. 9 KVP encoding der LockFeature-Operation (geändert nach OGC 2010b).

URL Component	obligatorisch	optional
REQUEST=LockFeature		
Adhoc Query Keywords (vgl. Tab. 6)		
Stored Query Keywords (vgl. Tab. 6)		
LOCKID		X
EXPIRY		X
LOCKACTION		X

GetFeatureWithLock

Die GetFeatureWithLock-Operation vereint die beiden Operationen GetFeature und LockFeature. Im Request wird festgelgt wie lange ein Feature gesperrt wird (URL Parameter EXPIRY) und wieviele Features eines Feature Types gesperrt werden sollen (LOCKACTION)

Mit Version 2.0 des WFS-Standards vom 02.11.2010 kommen Operationen hinzu, die den Standard um weitere Abfragemöglichkeiten ergänzen.

GetPropertyValue

GetPropertyValue liefert mit einer Abfrage den Wert einer Feature Property eines Feature Sets zurück. Tab. 10 stellt die Parameter der Operation zusammen.

Tab. 10 GetPropertyValue KVP encoding (geändert nach OGC 2010b).

URL Component	obligatorisch	optional
REQUEST=GetPropertyValue		
Adhoc Query Keywords (vgl. Tab. 6)		
Stored Query Keywords (vgl. Tab. 6)		
VALUEREFERENCE	X	
RESOLVEPATH		X

Die folgenden Operationen sind unter dem Begriff Stored query management zusammengefasst.

ListStoredQueries

Die Operation gibt eine Liste der auf dem Server gespeicherten Abfragen aus.

DescribeStoredQueries

DescribeStoredQueries liefert detaillierte Metadaten zu den verfügbaren gespeicherten Abfragen.

CreateStoredQuery

Mit dieser Operation können gespeicherte Abfragen erzeugt werden. Diese Operation muss nicht zwangsläufig verfügbar sein.

DropStoredQuery

Mit DropStoredQuery können zuvor erzeugte Abfragen wieder gelöscht werden. Wie CreateStoredQuery muss diese Operation nicht vorhanden sein.

5.4.3 GML – Geography Markup Language

GML ist eine Auszeichnungssprache, welche basierend auf einer XML-Grammatik für die Modellierung, die Speicherung sowie für den Austausch von Geodaten genutzt werden kann. GML führt die Simple Features Spezifikation des OGC fort, mit Version 3.2.1 wurde die GML-Spezifikation auch ISO-Standard (ISO 19136).
Die Spezifikation definiert (seit Version 3.2.1) unter anderem auch komplexere Geometrien, Koordinatensysteme, Topologie, Maßeinheiten, Zeitbezug und Dynamik.

Durch GML werden „Real world phenomena“, also Objekte aus der realen Welt, abgebildet. Eine Abstraktion eines Realweltobjekts bezeichnet man als „Feature“.
Die Eigenschaften eines Features, geometrische und nicht-geometrische, werden Properties genannt. Features mit gleichen Eigenschaften können zu einem Feature Type, einem Objekttyp, zusammengefasst werden. Ein Beispiel für einen Feature Type wäre ‚Gebäude’. Ist von einem ganz bestimmten Gebäude die Rede, in etwa von ‚Gebäude 1’, so ist dies eine Instanz vom Feature Type ‚Gebäude’.
Mehrere Feature Instanzen können eine sogenannte ‚Feature Collection’ bilden, wobei eine Feature Collection wie ein Feature behandelt wird. Die Beziehung zwischen zwei Features innerhalb einer Feature Collection wird durch die Property „featureMember“ ausgedrückt.

In GML wird analog zu XML zwischen Instanzdokumenten (*.gml), welche die Daten enthalten, und den Struktur beschreibenden Schema-Dateien (*.xsd), unterschieden. Solche Schema-Dateien enthalten unter anderem auch die vordefinierten Geometriedefinitionen:

- geometryBasic0d1d.xsd
- geometryBasic2d.xsd
- geometryComplexes.xsd
- geometryAggregates.xsd

Abb. 6 zeigt das Modell für einfache Geometrie Objekte.

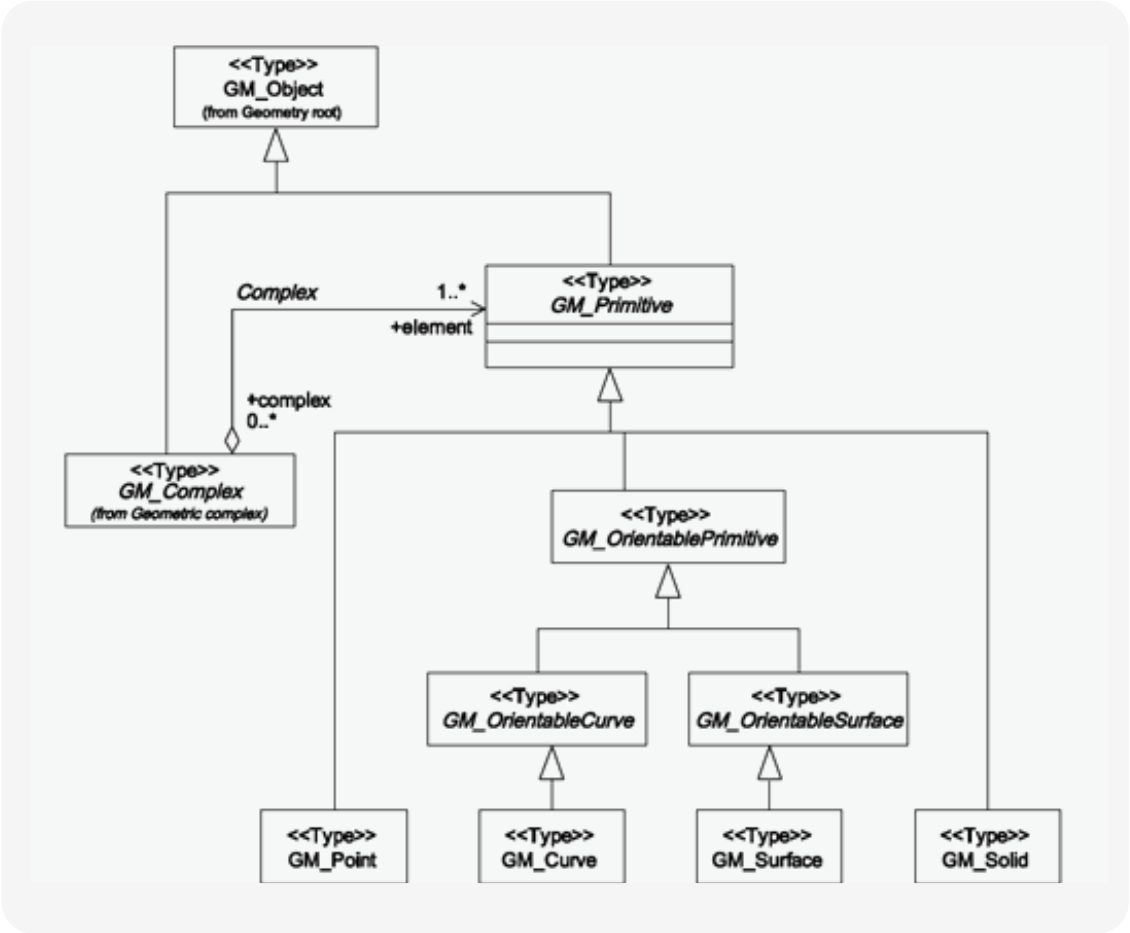


Abb. 6 Geometric primitives (Quelle: OGC 2007b).

In der GML Syntax ist jedes Feature ein Element, der Elementname beginnt mit einem Großbuchstaben. Die dazugehörigen Eigenschaften (Properties) sind Kindelemente eines Features. Der Name eines Kindelementes wird grundsätzlich klein geschrieben. ID und Raumbezugssystem sind XML-Attribute.

Der Raumbezug wird in der Regel in Form von EPSG-Codes angegeben. GML implementiert bezüglich der Definition von Koordinatenreferenzsystemen den ISO 19111 Standard. Das srsName-Attribut spezifiziert das Koordinatensystem, z.B. srsName="EPSG:4326". Der OpenGIS® Geography Markup Language (GML) Encoding Standard Version 3.2.1 enthält folgende Schemata:

- referenceSystems.xsd
- coordinateReferenceSystems.xsd
- datums.xsd
- coordinateSystems.xsd

- coordinateOperations.xsd

Alle Bestandteile des GML-Schemas sind dem Namensraum http://www.opengis.net/gml/3.2 mit dem Präfix *gml* zugeordnet. Für das ebenfalls verwendete xlink-Schema (mit dem Präfix *xlink*) des W3C wird auf den Namensraum http://www.w3.org/1999/xlink zugegriffen.

5.4.4 Simple Feature Access

Die Simple Feature Access-Spezifikation des OGC ist gleichzeitig ISO-Standard (ISO 19125). Sie beschreibt die Speicherung von Geometrien und Methoden, mit denen räumliche Beziehungen zwischen Geometrien analysiert werden können (OGC 2010).
Das Klassenmodell definiert folgende Geometrietypen:

- Point
- MultiPoint
- LineString, Line, LinearRing
- MultiLineString
- Polygon, Triangle
- MultiPolygon
- GeometryCollection

Das geografische Bezugssystem wird durch die Methode SRID festgelegt. Die Beziehungen von Geometrieobjekten zueinander können mit den Methoden Equals, Disjoint, Intersects, Touches, Crosses, Within und Contains beschrieben werden. Methoden, die die räumliche Analyse unterstützen, sind unter anderem Distance, Buffer, Intersection und Union.
Um Vektorgeometrien im Textformat beschreiben zu können, wurde die Auszeichnungssprache WKT (Well Known Text) erdacht, die ebenfalls Bestandteil der Spezifikation ist.

Beispiel für die Darstellung eines Punktes mit WKT:
Point(10 10)

5.4.5 Filter Encoding

Filter Encoding, vormals ein Bestandteil der WFS- Spezifikation, beinhaltet ein XML-Schema, das beschreibt, wie geometrische oder nicht-geometrische Eigenschaften von Objekten gefiltert werden können (OGC 2005). Der Standard, der aktuell in der Version 2.0.0 vorliegt und dem ISO-Standard ISO19143 entspricht. Der Standard beinhaltet neben Vergleichsoperatoren, logische, räumliche und zeitliche (Vers. 2.0) Operatoren.

Vergleichsoperatoren (ComparisonOperator)

Vergleichsoperatoren vergleichen, wie der Name schon sagt, einen Wert mit einem anderen Wert. Tab. 11 listet die im Filter Encoding Standard definierten Operatoren auf.

Tab. 11 Vergleichsoperatoren

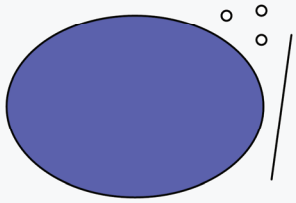
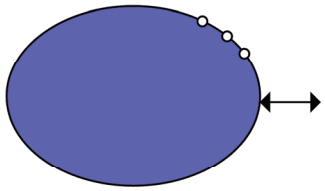
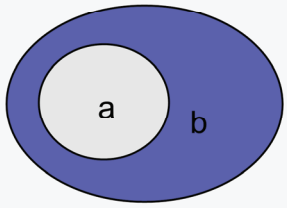
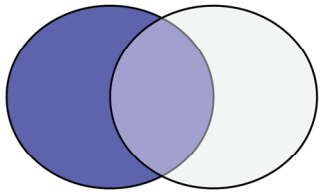
Vergleichsoperator	Beschreibung
Binary comparisons (=,<,>,>=,<=,<>)	Prüft, ob der Wert gleich oder ungleich einem Wert bzw. größer, größer gleich / kleiner, kleiner gleich als ein bestimmter Wert ist
PropertyIsNil[1]	prüft, ob ein Wert NIL ist
PropertyIsNull	testet, ob eine bestimmte Property in der zu prüfenden Resource existiert
PropertyIsLike	überprüft eine Zeichenkette auf Übereinstimmung des Musters
PropertyIsBetween	prüft, ob ein Wert zwischen zwei Werten liegt

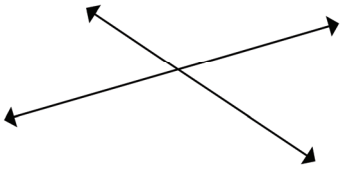
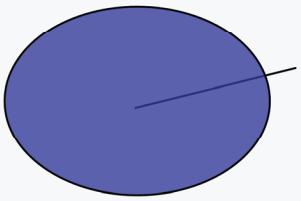
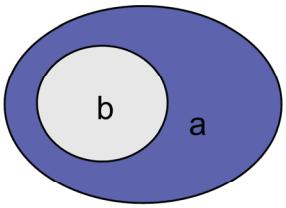
[1] in Version 2.0

Räumliche Operatoren

Mit einem räumlichen Operator kann geprüft werden, ob Geometrien zueinander in Beziehung stehen. Die im FE-Standard enthaltenen Operatoren sind in Tab. 12 aufgeführt. Die Operatoren Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects und Contains entsprechen den Definitionen des OGC Simple Feature Access Standards (ISO 19125-1).

Tab. 12 Räumliche Operatoren in FE Version 1.1 und 2.0.

Räumlicher Operator	Beschreibung
Equals	Geometrien sind identisch
Disjoint	Geometrien sind disjunkt 
Touches	Geometrien berühren sich 
Within	Geometrie a ist in Geometrie b enthalten 
Overlaps	Geometrien überlappen sich 

Räumlicher Operator	Beschreibung
Crosses	Geometrien kreuzen sich 
Intersects	Geometrien schneiden sich 
Contains	Geometrie a enthält Geometrie b 
Dwithin / Beyond	Geometrie a befindet sich innerhalb (Dwithin) oder außerhalb (beyond) einer bestimmten Distanz zu Geometrie b
BBOX	identifiziert alle Geometrien innerhalb einer definierten Box

Zeitliche Operatoren

Eine Neuerung gegenüber der letzten Version 1.1 stellen die zeitlichen Operatoren dar. Der Standard bedient sich dem in ISO 19108 (Geographic information - Temporal schema) definierten Zeitschema und umfasst die Operatoren After, Before, Begins, BegunBy, TContains, During, TEquals, TOverlaps, Meets, OverlappedBY, MetBy, EndedBy, AnyInteracts (OGC 2010c).

Logische Operatoren

Logische Operatoren werden dazu genutzt, mehrere Filterbedingungen zu kombinieren. Der AND-Operator wird verwendet, wenn alle Bedingungen erfüllt sein sollen. Muss nur eine der Bedingungen zutreffen, kommt der OR-Operator zum Einsatz. Der NOT-Operator kehrt den logischen Wert eines Filterausdrucks um (OGC 2010c).

5.4.6 SLD und SE

Werden über einen WMS mehrere Layer in einer Karte angezeigt, ist es, um die einzelnen Layer gut voneinander unterscheiden zu können, sinnvoll, diese verschiedenartig darzustellen. Das OGC hat dafür die Styled Layer Descriptor Implementation Specification (SLD) und die dazu gehörige Symbology Encoding Implementation Specification (SE) vorgelegt.
Die SE ist ein XML-Schema, mit dem Vektor- und Rasterdaten ein Style zugewiesen werden kann. Symbology Encoding kann bei einer Reihe von OGC Services angewandt werden (WMS, WFS, WCS). Wie Symbology Encoding in Verbindung mit einem WMS eingesetzt wird, ist in der SLD Spezifikation beschrieben.

5.4.7 WPS – Web Processing Service

Die OGC-Spezifikationen WMS und WFS beschreiben die Bereitstellung und den Austausch von Geodaten. Geht es aber darum, beispielsweise das Einzugsgebiet von kulturellen Einrichtungen zu ermitteln, Haltestellen im Umkreis ausfindig zu machen oder herauszufinden, ob ein Standort für bestimmte Nutzungen geeignet ist, so wird die Möglichkeit, räumliche Analysen durchführen zu können, benötigt.

Mit der WPS-Spezifikation liegt ein OGC-Standard für das sprach- und plattformunabhängige Ausführen von Prozessen vor, ein erstes Diskussionspapier wurde 2005 publiziert (OGC 2007b, KORDUAN 2008).

Die Spezifikation beschreibt einen allgemeinen Mechanismus, der es ermöglicht jedwede Art von Geoprozessen ausführen zu können, zeigt also, wie Prozesse gestaltet werden können, damit sie interoperabel einsetzbar sind und stellt keine Auflistung einzelner Geoprozesse dar. Von den aktuellen Kartenservern unterstützt vorerst nur deegree 3.0 den WPS, für Geoserver ist mit dem Release Candidate 2.1-RC1 soeben (Januar 2011) eine WPS-Erweiterung erschienen, Bestandteil der Standardkonfiguration ist er aber noch nicht. Eine überarbeitete OGC-Spezifikation in der Version 2.0.0 ist für Anfang 2011 angekündigt.

Über den WPS können sowohl Vektor-, als auch Rasterdaten verarbeitet werden. Die Daten können entweder auf dem Server liegen, auf dem die Prozesse ausgeführt werden, oder über das Netzwerk bereitgestellt werden (OGC 2007b). Der Standard beschreibt drei Operationen.

GetCapabilities

Der GetCapabilities Request liefert Metadaten im XML-Format über die auf dem Server verfügbaren Prozesse.
Ein Beispiel für einen GetCapabilities Request:

```
http://192.168.133.128/geoserver/ows?service=wps&version=1.0.0&request=GetCapabilities
```

Den dazugehörigen XML-Response zeigt Abb. 7.

```
<wps:Capabilities xml:lang="en" service="WPS" version="1.0.0">
   <ows:ServiceIdentification>
      <ows:Title>Prototype GeoServer WPS</ows:Title>
      <ows:Abstract/>
      <ows:ServiceType>WPS</ows:ServiceType>
      <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      <ows:ProviderName>The ancient geographes INC</ows:ProviderName>
      <ows:ProviderSite xlink:href="http://geoserver.org"/>
      <ows:ServiceContact/>
   </ows:ServiceProvider>
   <ows:OperationsMetadata>
      <ows:Operation name="GetCapabilities">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://192.168.133.128:80/geoserver/wps"/>
               <ows:Post xlink:href="http://192.168.133.128:80/geoserver/wps"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="DescribeProcess">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://192.168.133.128:80/geoserver/wps"/>
               <ows:Post xlink:href="http://192.168.133.128:80/geoserver/wps"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
      <ows:Operation name="Execute">
         <ows:DCP>
            <ows:HTTP>
               <ows:Get xlink:href="http://192.168.133.128:80/geoserver/wps"/>
               <ows:Post xlink:href="http://192.168.133.128:80/geoserver/wps"/>
            </ows:HTTP>
         </ows:DCP>
      </ows:Operation>
   </ows:OperationsMetadata>
   <wps:ProcessOfferings>
      <wps:Process wps:processVersion="1.0.0">
         <ows:Identifier>gt:buffer</ows:Identifier>
         <ows:Title>Buffer</ows:Title>
         <ows:Abstract>Buffer a geometry</ows:Abstract>
      </wps:Process>
   </wps:ProcessOfferings>
</wps:Capabilities>
```

Abb. 7 XML-Response des WPS-GetCapabilities-Requests

DescribeProcess

Die DescribeProcess Operation liefert eine detaillierte Beschreibung der zur Verfügung stehenden Prozesse einschließlich der Datenin- und output-Parameter und der zulässigen Datenformate. Tab. 13 zeigt die Parameter des Requests.

Tab. 13 Parameter der DescribeProcess-Operation (geändert nach OGC 2007b).

Name	Wert	obligatorisch	optional
service	WPS	X	
request	DescribeProcess	X	
version	1.0.0	X	
Language	Bsp: en		X
Identifier	Ein oder mehrere Prozesse (durch Komma getrennt) Bsp: intersection, union	X	

Execute

Mit der Execute Operation kann ein WPS-Client einen vom Server implementierten Prozess ausführen. Der Dateninput kann direkt oder über einen Link zu einer Webresource in den Execute Request eingebettet werden. Der Output wird als XML-Datei geliefert, in den Response eingebettet oder als zugängliche Web Resource gespeichert werden.

Die aktuelle Version 1.0.0 wird bisher erst in wenigen Web-GIS-Anwendungen genutzt, da die Anbindung eines WPS-Clients an einen Server sich wesentlich komplexer gestaltet als die eines WMS.

6 WebGIS Konzept & Umsetzung

In diesem Kapitel soll nun ein ein Konzept für ein WebGIS für kulturell nutzbare Immobilien sowie die Umsetzung in eine Beispielanwendung mit Open Source Software vorgestellt werden.

6.1 Anforderungsprofil

6.1.1 Allgemeine Anforderungen

Die Anwendung soll:

- der Information dienen
- die Kommunikation zwischen beiden Seiten erleichtern
- eine Übersicht über verfügbare Räumlichkeiten schaffen
- als Analysewerkzeug für die Auslastung von städtischen Liegenschaften dienen
- die Interaktion fördern
- intuitiv bedienbar sein
- übersichtlich und klar strukturiert sein

6.1.2 Nutzergruppen

Das WebGIS für die Bereitstellung von Daten über leer stehende und kulturell nutzbare Räume und Gebäude soll von verschiedenen Zielgruppen genutzt werden, welche unterschiedliche Anforderungen an ein solches System haben. Aufgrund ihrer Fragestellungen benötigen sie einen bestimmten Funktionsumfang. Anhand der Nutzergruppen können besondere Zugriffsrechte, wie Lese- und Schreibrechte vergeben werden.

Bearbeiter

Die Bearbeiter, Mitarbeiter des Liegenschaftsamtes (Immobilienmanagement), stellen auch die Daten bereit und sind für die Datenbankpflege zuständig. Sie benötigen Zugriff auf den vollen Funktionsumfang und alle Daten.

Interne Nutzer

Interne Nutzer, weitere Mitarbeiter der Kommunalverwaltung (Immobilienmanagement, Kulturförderung, Wirtschaftsförderung, Stadtentwicklung) nutzen das System hauptsächlich zu Informations- und Analysezwecken. Sie benötigen keine Editiermöglichkeit.

Externe Nutzer

Externe Nutzer, Kulturschaffende und Interessierte, benötigen lesenden Zugriff auf die Daten. Da nicht alle Daten des Immobilienmanagements für die Öffentlichkeit bestimmt sind, kann die Datenmenge eingeschränkt werden, z.B. durch die Verwendung von Datenbank-Views.

6.1.3 Funktionen der WebGIS-Anwendung

Grundfunktionen

Grundfunktionen sind Funktionen, die der Orientierung in der Karte dienen sollen. Dazu gehören:

- Zoom (vergrößern/verkleinern)
- Pan (Kartenausschnitt verschieben)
- Messwerkzeug
- Info (Anzeige von Sachdaten)
- Interaktive Layerauswahl

Editierfunktionen

Die Bearbeiter benötigen Funktionen zum Editieren der Daten:

- Objekt erstellen
- Objekt ändern
- Speichern
- Löschen

Abfragefunktionen

Es muss möglich sein, nach bestimmten Attributen zu suchen, ebenso eine Visualisierung der Daten nach bestimmten Filterkriterien vorzunehmen.
Funktionen für die räumliche Analyse, wie die Umkreissuche, sollen ebenfalls verfügbar sein.

6.1.4 Daten

Das WebGIS soll folgende Daten beinhalten:

- Gebäudedaten
- Raumdaten
- Stadtplan
- Luftbilder
- Fotos / Grundrisse
- (Straßennetz)
- (Öffentliche Verkehrsmittel)

6.1.5 Kommunikation

Um die Kommunikation zum Ansprechpartner unkompliziert zu gestalten ist es sinnvoll, wenn die Kulturschaffenden aus dem System heraus eine e-mail an den jeweiligen Ansprechpartner verschicken können.
Die Möglichkeit, einen Raum über einen Belegungskalender online zu buchen, vereinfacht die Verwaltung von Zwischen- und Mischnutzungen.

6.2 PostGIS-Datenbank

PostGIS stellt mit räumlichen Objekten und Funktionen die räumliche Erweiterung für die objektrelationale PostgreSQL-Datenbank dar. Sie kann als Backend für GIS-Anwendungen genutzt werden. Das Open-Source Projekt der OSGeo implementiert die Simple Feature Access Spezifikation des OGC.

Geometriedaten können entweder aus einer Shape-Datei importiert oder aber auch aus dem WKT (Well Known Text)-Format herausgelesen werden. Das Koordinatensystem wird in Form eines EPSG-Codes ablegt.

Abb. 8 zeigt das Datenbankmodell für die Verwaltung der Raum- und Gebäudedaten. Die Liste der Spalten der Tabellen Raum und Gebäude stellt nur eine Auswahl dar. Hier können beliebige weitere Attribute ergänzt werden.

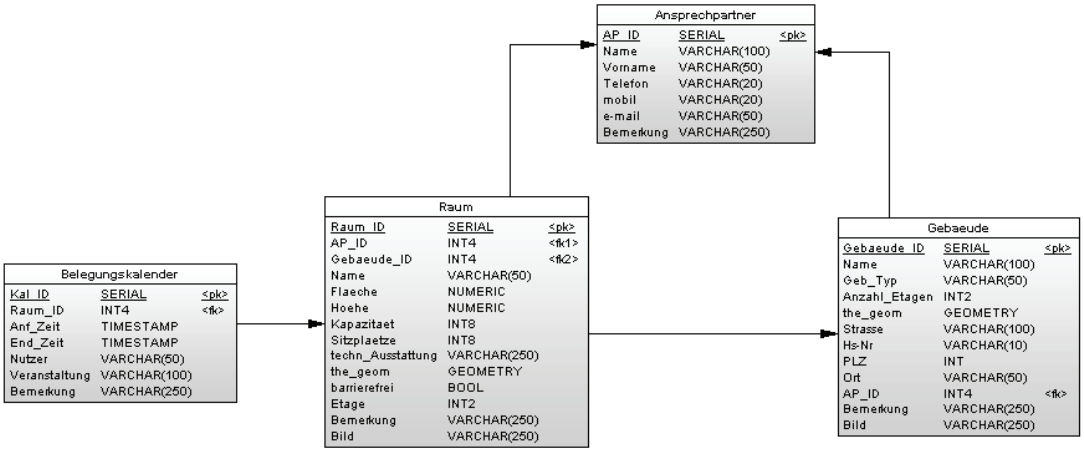


Abb. 8 Datenbankmodell für die Verwaltung der Raum- und Gebäudedaten.

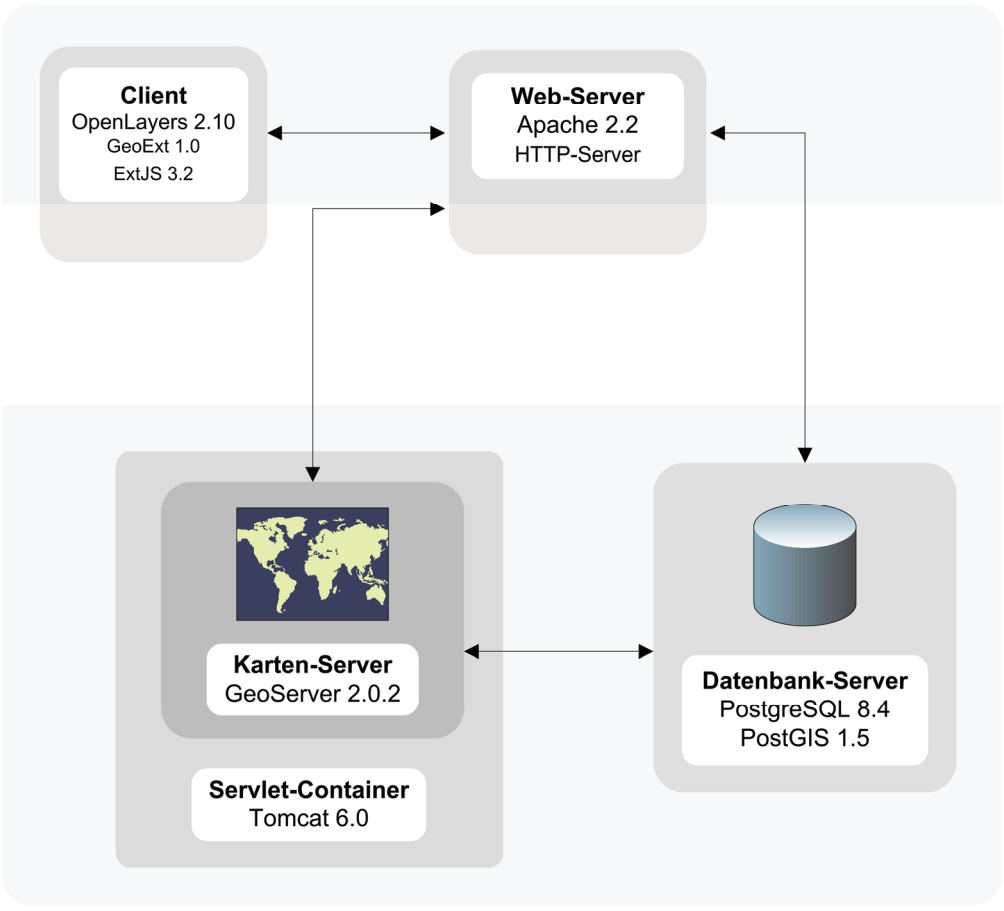


Abb. 9 Systemkomponenten des WebGIS.

6.3 Apache HTTP-Server

Als Webserver kommt der Apache HTTP Server in der Version 2.2 zum Einsatz. Die modular aufgebaute Open Source-Software der Apache Software Foundation entstammt der Unix-Welt und ist der am meisten genutzte Webserver im Internet[1].

[1] http://news.netcraft.com/archives/category/web-server-survey/ - zuletzt geprüft am 25.02.2011

6.4 Apache Tomcat 6.0

Der Apache Tomcat ist ein Servlet-Container, der dazu dient, Java-Code auf einem Webserver auszuführen. Die Kommunikation mit dem Webserver gewährleistet das Modul mod_jk, ein Connector-Plugin, das in den Apache HTTP-Server eingebunden wird.

6.5 GeoServer

Als Kartenserver wird GeoServer, eine Java-basierte Open Source-Software, die auf GeoTools aufbaut, genutzt. GeoServer wird hier als Servlet in den Webapps-Ordner des Apache Tomcat integriert, kann aber auch eigenständig verwendet werden. Die Administration von GeoServer erfolgt über eine Weboberfläche (Abb. 10).

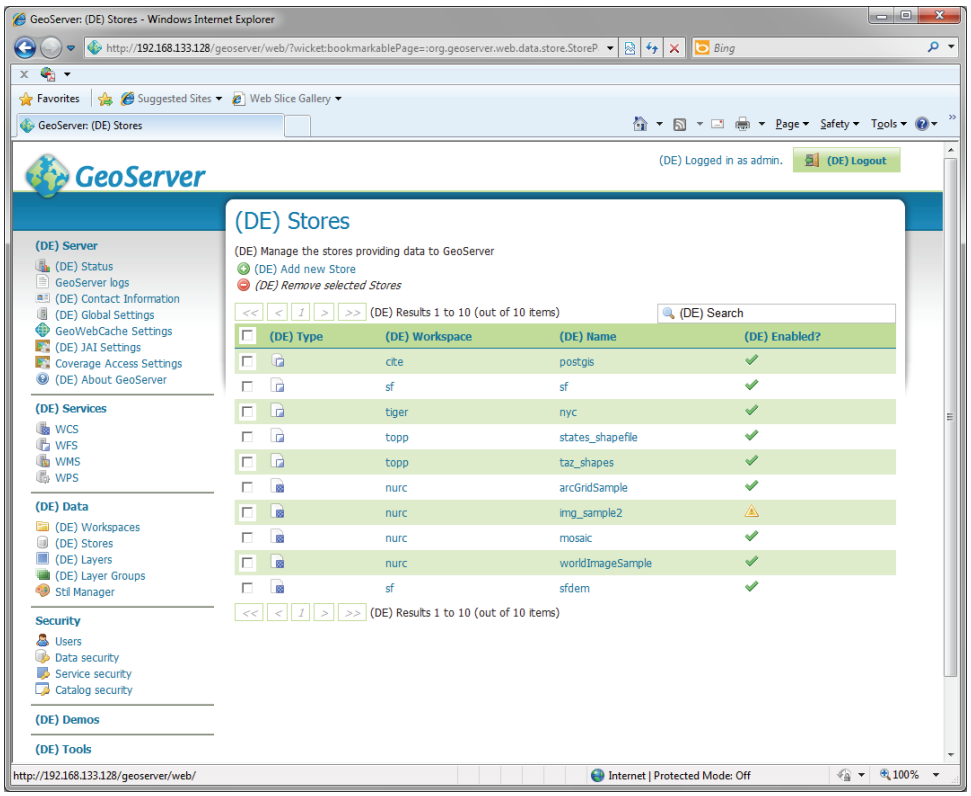


Abb. 10 Administrationsoberfläche von GeoServer.

Das GeoServer-Projekt zeichnet sich durch die Einhaltung offener Standards aus. Die aktuelle GeoServer-Version 2.0.2 unterstützt die OGC-Standards:

- WMS 1.1.1 und 1.3.0
- WFS 1.0 und 1.1.0
- GML 2
- Filter Encoding 1.0 und 1.1
- SLD
- WCS 1.0 und 1.1

Der WPS ist in der Standardkonfiguration nicht verfügbar. Für den gerade erschienenen Release Candidate von GeoServer 2.1 ist eine WPS-Extension erhältlich. Die WPS-Implementierung ist allerdings noch in der Entwicklung. So gibt es derzeit beispielsweise noch keine Möglichkeit, den Output zu speichern oder als WMS zu visualisieren.

6.6 OpenLayers 2.10

Da die hier vorgestellte Lösung für ein breites Publikum gedacht ist, dass über keine besonderen GIS-Kenntnisse verfügt und ohne die Installation von Software auskommen soll, wird der Webmapping-Client OpenLayers verwendet.
Die erste Version der Software wurde 2006 von der Firma MetaCarta veröffentlicht. In den nachfolgenden Jahren entwickelte sich daraus ein Open Source-Projekt, welches 2008 offiziell zu einem OSGeo-Projekt wurde (JANSEN & ADAMS).
OpenLayers ist eine JavaSript-Bibliothek, die auf relativ einfache Art und Weise in jede beliebige Web-Seite integriert werden kann (Abb. 11).

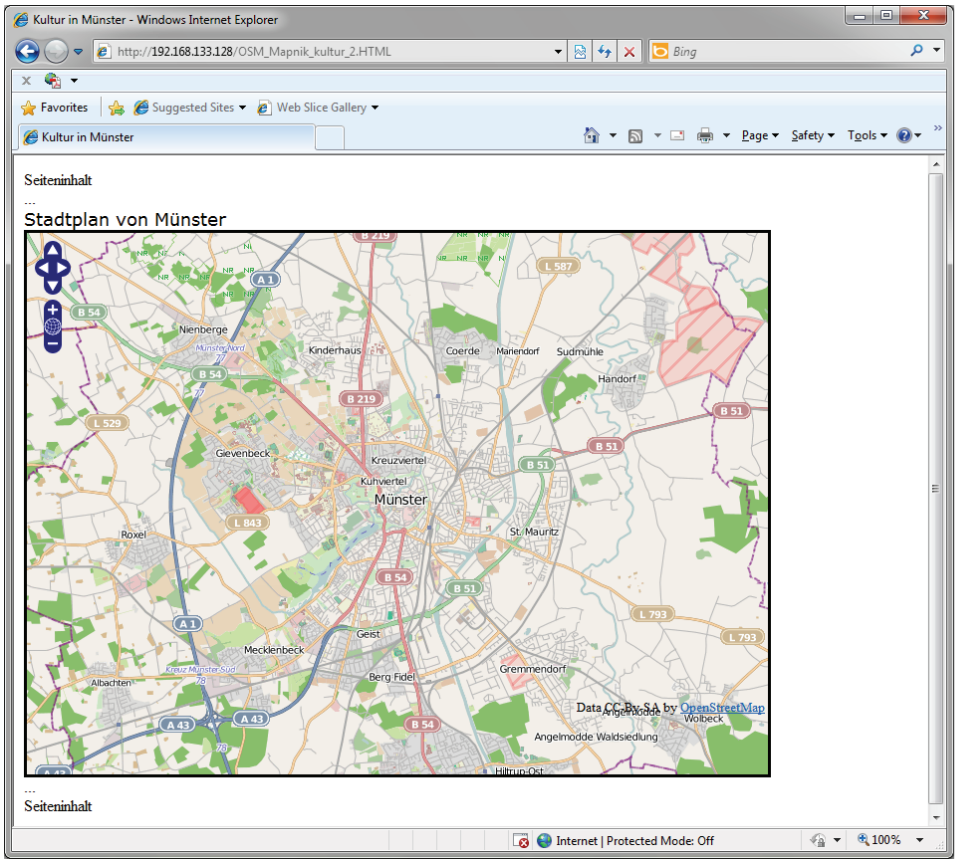


Abb. 11 Beispiel für eine einfache OpenLayers-Karte mit OSM-Daten.

Die Oberfläche ist sehr schlicht gehalten und Standardfunktionen wie Zoom und Pan lassen sich ähnlich wie bei Google Maps intuitiv ausführen. Darüber hinaus lassen sich OpenLayers-Anwendungen beliebig um zusätzliche Funktionen erweitern und somit den Bedürfnissen anpassen.

Um aus der bloßen Darstellung einer Karte eine umfangreiche WebGIS-Anwendung zu machen, soll hier OpenLayers in Kombination mit ExtJS und GeoExt verwendet werden.

ExtJS 3.2

Das clientseitige Framework ExtJS setzt die AJAX-Technologie ein, um Webseiten dynamisch zu gestalten. Die JavaScript-Bibliothek beinhaltet eine Reihe von vordefinierten Elementen, darunter Auswahlboxen, Menüleisten, Checkboxen und Textfelder.

GeoExt 1.0

GeoExt – "JavaScript Toolkit for Rich Web Mapping Applications"[1], verbindet die Eigenschaften der beiden JavaScript-Bibliotheken OpenLayers und ExtJs, um WebGIS-Anwendungen optisch und funktional Desktop-GIS-Anwendungen näher zu kommen.

[1] http://geoext.org/

6.6.1 Die Benutzeroberfläche

Die Benutzeroberfläche gliedert sich in fünf Bereiche: das MapPanel, das GridPanel, die Toolbar, das Suchfenster und den Bereich für die Raumreservierung (Abb. 12).
Anhand der im Anforderungsprofil festgelegten Nutzerkreise empfiehlt es sich, die Funktionen auf die Bedürfnisse anzupassen. Internen Bearbeitern, die den Datenbestand pflegen, kann die Anwendung mit dem gesamten Funktionsumfang im Intranet zur Verfügung gestellt werden. Externe Besucher erhalten über das Internet eine Kartenanwendung ohne Editiermodus. Auch kann der Umfang der bereitgestellten Daten, nicht alle Daten sind für die Öffentlichkeit bestimmt, durch die Verwendung von Views eingeschränkt werden.

MapPanel

Das MapPanel beinhaltet die Karte. Es bedient sich der Klasse GeoExt.MapPanel. In den Optionen des MapPanels werden unter anderem die Größe der Karte, die Inhalte und deren Projektion festgelegt.

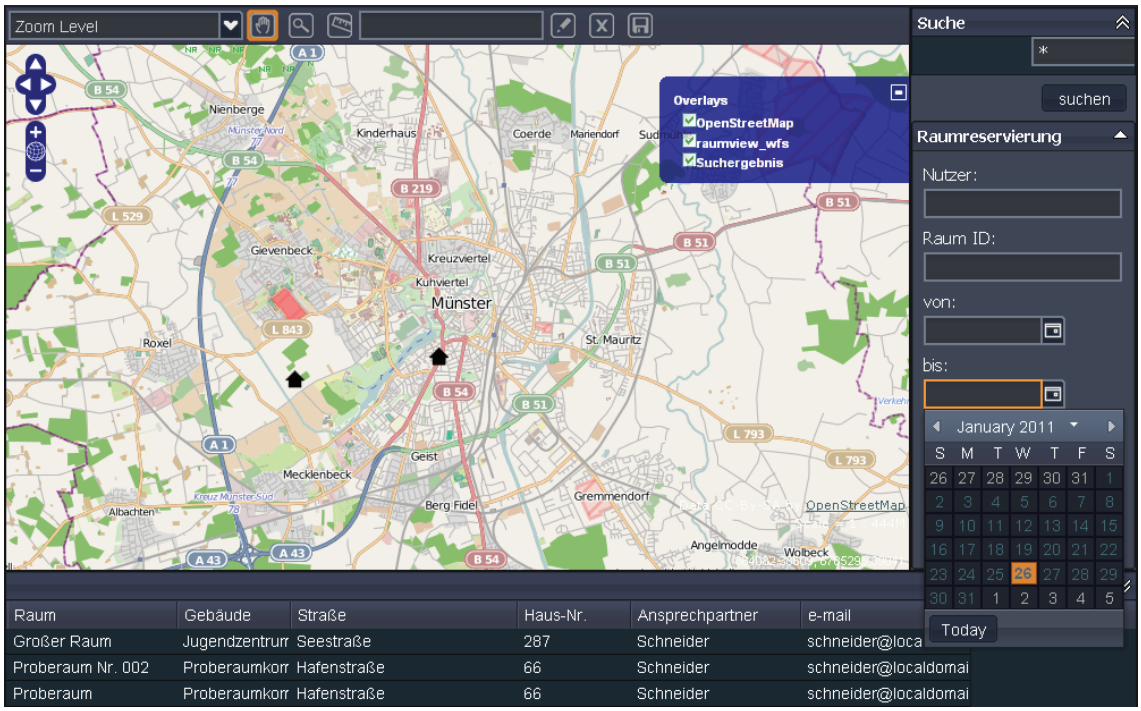


Abb. 12 Benutzeroberfläche der Webanwendung mit OpenLayers, ExtJs und GeoExt.

Mouse Position

Fährt man mit der Maus über die Karte, so werden die jeweiligen Koordinaten angezeigt. Dies wird über die OpenLayers-Control MousePosition vollführt.

```
mapPanel.map.addControl(new
OpenLayers.Control.MousePosition({element: $('location')}));
```

Maßstabsanzeige in der Karte

Der aktuelle Kartenmaßstab wird als Text und als Maßstabsbalken innerhalb der Karte angezeigt. Dafür sorgen die OpenLayers-Controls ‚Scale' und ‚ScaleLine'.

```
mapPanel.map.addControl(new OpenLayers.Control.Scale($('scale')));
```

GridPanel

Das GridPanel basiert auf dem ExtJs Framework und zeigt die Attributtabelle mit Informationen zu den einzelnen Features an. Hier kann festgelegt werden, welche Spalten in der Tabelle angezeigt werden sollen. Unter 6.6.5 soll darauf näher eingegangen werden.

```
gridPanel = new Ext.grid.GridPanel({
                region: "center",
                width: 350,
                store: store,
                columns: [
                   {header: "Name",
                    width: 100,
                    sortable: true,
                    dataIndex: "name"
                   },
                   {header: "beschreibung",
                    width: 250,
                    sortable: true,
                    dataIndex: "beschreibung"
                   }
                ],
                sm: select
});
```

Beispiel 1 GridPanel mit zwei Spalten.

6.6.2 Die Layer

In OpenLayers sind die OGC-Standards WMS, WFS implementiert. Darüberhinaus können proprietäre Dienste wie Google Maps, Yahoo Maps oder Bing Maps sowie die freien OSM-Daten als Layer hinzugefügt werden.

LayerSwitcher

```
mapPanel.map.addControl(new OpenLayers.Control.LayerSwitcher());
```

Die OpenLayers LayerSwitcher-Control dient dazu, einzelne Layer an- oder auszuschalten. In den Map options kann festgelegt werden, ob ein bestimmter Layer als „base layer“ immer angezeigt werden soll oder ob alle Layer an- und ausschaltbar sein sollen.

WMS

Luftbilder, aber auch Karten und Pläne werden meist als WMS bereitgestellt. Beispiel 2 zeigt, wie ein WMS in OpenLayers eingefügt wird. Die angegebenen Parameter (hier: layers und srs) werden als URL-Parameter in den GetMap-Request umgesetzt.

```
var wms = new OpenLayers.Layer.WMS("Raum_wms", {
            "http://192.168.178.51/geoserver/wms",
            {layers: "cite:raum",
             srs: "EPSG:31467"}
});
map.addLayer(wms);
```

Beispiel 2 Einfügen eines WMS in OpenLayers.

OSM

In der Beispielanwendung dient der Stadtplan mit Open Street Map-Daten zur groben Orientierung.

```
var osm = new OpenLayers.Layer.OSM()
map.addLayer(osm);
```

Beispiel 3 Einfügen eines OSM-Layers.

WFS

Die Raum- und Gebäude-Layer werden als WFS eingebunden. Der WFS liefert keine gerenderte Karte, sondern Objekte im GML-Format. Dazu wird ein OpenLayers Vector-Layer erzeugt, der das WFS-Protokoll nutzt.

```
var wfsraum = new OpenLayers.Layer.Vector("Raum_wfs", {
        protocol: new OpenLayers.Protocol.WFS({
        url: "/geoserver/ows",
        version: "1.1.0",
        featureType: "raum",
        featureNS: "http://www.opengeospatial.net/cite",
        typeName: "cite:raum",
        geometryName: "the_geom",
        srsName: "EPSG:31467"
    })
});
```

Beispiel 4 Einfügen des Raum-Layers als WFS.

Editierbarer WFS-Layer

Die Transaction-Operation des WFS-Termöglicht das Erzeugen, Ändern und das Löschen von Objekten (s. 6.6.3 – Editiertoolbar). Die Funktion commit() übergibt die Daten an den WFS. Das Ändern der Daten wird durch die Modify-Control ermöglicht:

```
var modifyControl = new OpenLayers.Control.ModifyFeature(wfsraum);
mapPanel.map.addControl(modifyControl);
modifyControl.activate();
```

Die Sachdaten können in der Attributtabelle im GridPanel bearbeitet werden (s. 6.6.5). Um die Änderungen in der Datenbank zu speichern, bedarf es einer Speicher-Strategie, der OpenLayers.Strategy.Save.

```
var saveStrategy = new OpenLayers.Strategy.Save({
     onCommit: function() {
                 saveStrategy.layer.refresh();
     }
});
```

Die Strategie wird dem WFS-Layer als Property hinzugefügt und dem Speicher-Button (s. 6.6.3 – Editier-Toolbar) zugewiesen.

```
var wfsraum = new OpenLayers.Layer.Vector("Raum_wfs", {
        strategies: [new OpenLayers.Strategy.Fixed(), saveStrategy],
        protocol: new OpenLayers.Protocol.WFS({
        url: "/geoserver/ows",
        version: "1.1.0",
        featureType: "raum",
        featureNS: "http://www.opengeospatial.net/cite",
        typeName: "cite:raum",
        geometryName: "the_geom",
        srsName: "EPSG:31467"
    })
});
```

6.6.3 Toolbar

Die Toolbar enthält alle Werkzeuge, die in der Webanwendung genutzt werden können. Die Bedienung erfolgt aufgrund der selbsterklärenden Icons intuitiv. Die default OpenLayers Icons wurden durch eigene ersetzt und im <style>-Tag verlinkt.

```
<style>
  .zoom {
    background-image:url(icons/zoom-icon.png)!important ;
    height:20px !important;
    width:20px !important;
  }
</style>
```

Damit immer nur ein Werkzeug aktiv sein kann, wird eine Toggle-Group erzeugt. Unter einem Toggle ist ein Umschalter zu verstehen. Der Anwender kann so entweder die ZoomBox benutzen, navigieren oder eine Distanzmessung durchführen, wenn diese Buttons derselben Togglegroup angehören.
Es können auch mehrere solcher Togglegroups existieren. In unserem Fall werden die Buttons für das Zeichnen, Bearbeiten und Löschen eines Features einer eigenen Togglegroup zugeordnet.

Zoom

Das Zoomen kann auf unterschiedliche Weise erfolgen: entweder mit dem Mausrad, der voreingestellten OpenLayers-Control oder mit einer ZoomBox durch das Aufziehen eines Rechtecks.

Pan

Mit der Pan-Funktion kann man die Karte verschieben.

Messwerkzeug

Das Messwerkzeug kann zur Distanzmessung verwendet werden. Grundlage für das Werkzeug ist die OpenLayers Measure-Control.

```
action = new GeoExt.Action(
         {
          iconCls: 'measure',
          control: new OpenLayers.Control.Measure(OpenLayers.Handler.Path,
                 {
                   persist: true,
                   handlerOptions:
                        {
                          layerOptions: {styleMap: new OpenLayers.Style(
                                {
                                  strokeWidth: 3,
                                  strokeColor: "#666666",
                                  strokeDashstyle: "dash"
                                })
                        }
                 },
                   eventListeners:
                        {
                          measure: function(evt) {
                                measureBox.setValue("Distanz : " +
                                evt.measure.toFixed(2)
                                + " " + evt.units);
                               }
                        }
                }),
                tooltip: 'measure distance',
                map:map,
                toggleGroup: 'map'
}) ;
basicTbarItems.push(action);
```

Beispiel 5 Button mit GeoExt.Action für die Distanzmessung

Die Messung wird per Doppelklick abgeschlossen, der Messwert wird auf zwei Nachkommastellen gerundet in eine Textbox in der Toolbar ausgegeben (Abb. 13).

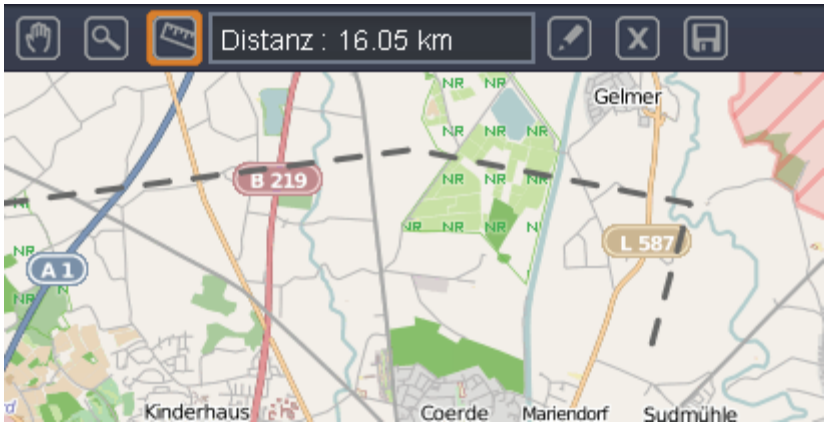


Abb. 13 Distanzmessung.

Vordefinierte Maßstäbe

In der Toolbar können vordefinierte Maßstäbe über eine Auswahlbox eingestellt werden (Abb. 14). Die Funktion bedient sich der Klasse GeoExt.data.ScaleStore. Der ScaleStore beinhaltet die Felder:

- level (Zoom Level)
- scale (Maßstabs-Nenner)
- resolution (map units pro Pixel)

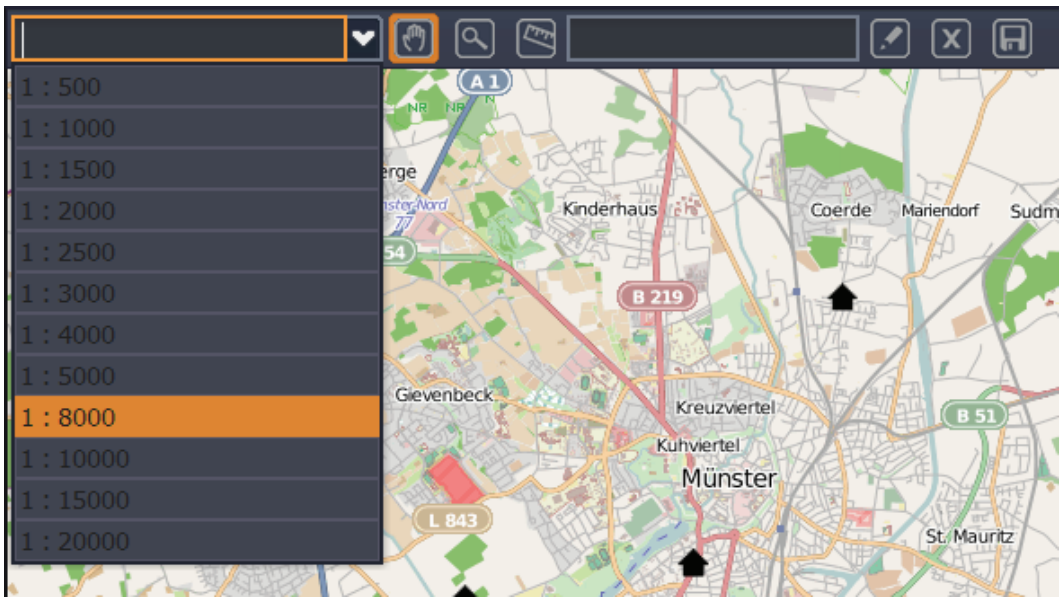


Abb. 14 Maßstab Auswahlbox

Editier-Toolbar

Die Editier-Toolbar enthält die Editierfunktionen zum Erzeugen neuer Features, zum Löschen und zum Speichern, sie sollte nur einem bestimmten internen Nutzerkreis zur Verfügung gestellt werden.

Feature erzeugen

Mit der OpenLayers-Control DrawFeature können neue Objekte gezeichnet werden. Im OpenLayers-Handler kann die Geometrie bestimmt werden.

```
var drawControl = new OpenLayers.Control.DrawFeature(
                    wfsraum,
                    OpenLayers.Handler.Point,
                    {handlerOptions: {multi: true}}

);
```

Beispiel 6 Hinzufügen eines neuen Features des Raum-Layers.

Feature löschen

Ein in der Attributtabelle selektiertes Feature kann durch das Klicken des Delete-Buttons aus der Datenbank gelöscht werden.

```
handler: function() {

        gridPanel.getSelectionModel().each(function(rec) {
                var feature = rec.get("feature");
                modifyControl.unselectFeature(feature);
                store.remove(rec);
                if(feature.state !== OpenLayers.State.INSERT) {
                        feature.state = OpenLayers.State.DELETE;
                        wfsraum.addFeatures([feature]);
                }

        })
}
```

Beispiel 7 Funktion zum Löschen eines Features.

Änderungen speichern

Werden Features hinzugefügt oder geändert, so muss die Bearbeitung mit Speichern abgeschlossen werden. Ein Klick auf den Save-Button übergibt mit der Funktion „commitChanges“ die Änderungen an die Datenbank.

```
handler: function(){
            store.commitChanges();
            saveStrategy.save();
         }
```

6.6.4 Darstellung der WFS-Layer

Über die StyleMap-Klasse kann in OpenLayers einem Layer ein Style zugewiesen werden. OpenLayers implementiert die OGC SLD-Spezifikation.
In der Anwendung werden alle Gebäuden standardmäßig mit einem schwarzen Haussymbol dargestellt. Sobald ein Objekt selektiert wird, wird das Symbol durch ein rotes Haus ersetzt.

Möchte man Objekte aufgrund bestimmter Filterkriterien visualisieren, so kann eine OpenLayers.Rule mit einem Filter erzeugt werden. Im Beispiel werden alle Räumen mit einer Fläche zwischen 100 und 200 qm mit einem besonderen Style versehen (Beispiel 8).

```
new OpenLayers.Rule({
            title: "Raumgröße: 100-200 qm",
            filter: new OpenLayers.Filter.Comparison({
                type: OpenLayers.Filter.Comparison.BETWEEN,
                property: "flaeche",
                upperBoundary: 200,
                lowerBoundary: 100
            }),
            symbolizer: {
                graphicName: "point",
                pointRadius: 20,
                fillColor: "#6699cc",
                strokeColor: "#666666",
                strokeWidth: 1
            }
        }),
```

Beispiel 8 OpenLayers.Rule für die Symbolisierung nach einem Filterkriterium.

6.6.5 Feature Info

Attributtabelle

Das GridPanel beinhaltet die Attributtabelle (Abb. 15). Im GeoExt.data.FeatureStore wird festgelegt, welche Informationen aus der Datenbank angezeigt werden sollen.

```
gridPanel = new Ext.grid.GridPanel({
        region: "south",
        collapsible: true,
        collapseMode: 'mini',
        height: 200,
        store: store,
        columns: [
                        {header: "Raum",
                         width: 150,
                         sortable: true,
                         dataIndex: "raum"
                        },
                        {header: "Gebäude",
                         width: 100,
                         sortable: true,
                         dataIndex: "gebaeude"
                         },

                        . . .

                        {header: "Ansprechpartner",
                         width: 150,
                         sortable: true,
                         dataIndex: "ansprechpartner"
                        },
                        {header: "e-mail",
                         width: 150,
                         sortable: true,
                         dataIndex: "ap_email"
                        }
                    ],
                sm: select
});
```

Beispiel 9 Auzug aus dem GridPanel-Abschnitt.

Soll die Tabelle editierbar sein, wird ein EditorGridPanel erzeugt:

```
var gridPanel = new Ext.grid.EditorGridPanel({})
```

Raum	Gebäude	Straße	Haus-Nr.	Ansprechpartner	e-mail
Proberaum	Proberaumkomplex	Hafenstraße	66	Schneider	schneider@localdomain.net
Proberaum Nr. 002	Proberaumkomplex	Hafenstraße	66	Schneider	schneider@localdomain.net

Abb. 15 Auszug aus der Attributtabelle.

Das FeatureSelectionModel gewährleistet die Synchronisierung zwischen GridPanel und MapPanel. Wird ein Feature in der Tabelle selektiert, so wird dies gleichzeitig in der Karte ausgewählt und umgekehrt.

```
var select = new GeoExt.grid.FeatureSelectionModel();
```

Info-Popup

Fährt man mit der Maus über ein Feature des Layers „Raum“, öffnet sich ein Popup, das die Bezeichnung und eine Kurzinfo sowie ein Foto zu dem betreffenden Objekt beinhaltet. Das Popup nutzt die OpenLayers.Popup.FramedCloud. Die Funktion greift direkt auf die Features des Layers zu. Mit Html wird wird festgelegt, welche Attribute des Features in welcher Form angezeigt werden sollen:

```
this.popup = new OpenLayers.Popup.FramedCloud("ol-info-popup",
window_position, null,'<div class="map-kurzinfo-popup"><span>' +
myFeature.data.name + '</span><br><span>' + myFeature.data.beschreibung
+ '</span><br><img src="'+feature.data.bild+'"></div>', anchor, false);
```

Die Größe des Popups bestimmt sich automatisch durch den Inhalt.
Zum Hinzufügen zu einer Karte wird die OpenLayers.Map.addPopup–Methode genutzt:

```
mapPanel.map.addPopup(this.popup, true);
```

Das Popup wurde so konfiguriert, dass es nach einigen Sekunden wieder erlischt:

```
this.autodestroy = window.setTimeout(function(){
     if (mapPanel.map && mapPanel.map.popups &&
         mapPanel.map.popups.length > 0) {
                var len = mapPanel.map.popups.length;
                for (var i = 0; i < len; i++) {
                    if (mapPanel.map.popups[i] &&
                        mapPanel.map.popups[i].destroy) {
                          mapPanel.map.popups[i].destroy();
                        }
                    }
          }
     }, 3000);
```

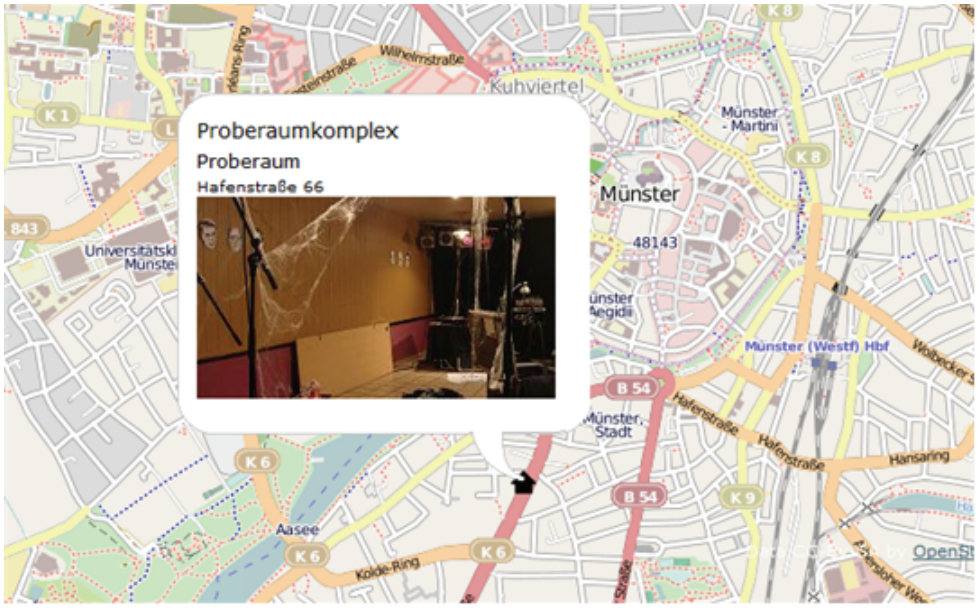


Abb. 16 Info-Popup.

6.6.6 Suchfunktion

Ist der Anwender auf der Suche nach einem Raum mit bestimmten Eigenschaften, so kann im Suchfenster eine attributive Abfrage formuliert werden. Dazu wird ein Vergleichsoperator eingesetzt. OpenLayers implementiert die Vergleichsoperatoren des OGC Filter Encoding Standards. In der GeoExt API sind die Vergleichsoperatoren (Comparison Filter) folgendermaßen deklariert:

- <name>__eq: OpenLayers.Filter.Comparison.EQUAL_TO
- <name>__ne: OpenLayers.Filter.Comparison.NOT_EQUAL_TO
- <name>__lt: OpenLayers.Filter.Comparison.LESS_THAN
- <name>__le: OpenLayers.Filter.Comparison.LESS_THAN_OR_EQUAL_TO
- <name>__gt: OpenLayers.Filter.Comparison.GREATER_THAN
- <name>__ge: OpenLayers.Filter.Comparison.GREATER_THAN_OR_EQUAL_TO
- <name>__like: OpenLayers.Filter.Comparison.LIKE

Im Beispiel soll ein Raum mit dem Namen ‚Proberaum' gesucht werden. Das Suchfenster ist mit einer GeoExt SearchAction verknüpft.

```
items: [{
            xtype: "textfield",
            name: "raum__like",
            value: "*"
        }]
```

Nach Drücken des ‚Suchen'-Buttons oder der Enter-Taste wird der Post-Request, der den Suchbegriff enthält, an den Kartenserver geschickt:

```
<wfs:GetFeature xmlns:wfs="http://www.opengis.net/wfs" service="WFS" version="1.0.0"
xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-
transaction.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <wfs:Query typeName="feature:raumview"
              xmlns:feature="http://www.opengeospatial.net/cite">
      <ogc:Filter  xmlns:ogc="http://www.opengis.net/ogc">
         <ogc:And>
            <ogc:PropertyIsLike wildCard="*" singleChar="."escape="!">
               <ogc:PropertyName>raum</ogc:PropertyName>
                  <ogc:Literal>Proberaum</ogc:Literal>
            </ogc:PropertyIsLike>
         </ogc:And>
      </ogc:Filter>
   </wfs:Query>
</wfs:GetFeature>
```

Das Suchresultat wird von GeoServer in einer XML-Datei ausgegeben:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection xmlns="http://www.opengis.net/wfs" xmlns:wfs="http://
www.opengis.net/wfs" xmlns:gml="http://www.opengis.net/gml" xmlns:cite="http://
www.opengeospatial.net/cite" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/cite http://192.168.133.128:80/
geoserver/
wfs?service=WFS&amp;version=1.0.0&amp;request=DescribeFeatureType&amp;typeName=cite%3Araum
view http://www.opengis.net/wfs http://192.168.133.128:80/geoserver/schemas/wfs/1.0.0/WFS-
basic.xsd">
   <gml:boundedBy>
      <gml:null>unknown</gml:null>
   </gml:boundedBy>
   <gml:featureMember>
      <cite:raumview fid="raumview.fid--4065bf78_12e63c0294b_-7fe5">
         <cite:raum_id>1</cite:raum_id>
         <cite:raum>Proberaum</cite:raum>
         <cite:gebaeude>Proberaumkomplex</cite:gebaeude>
         <cite:strasse>HafenstraÃŸe</cite:strasse>
         <cite:hs_nr>66</cite:hs_nr>
         <cite:plz>48143</cite:plz>
         <cite:ort>Münster</cite:ort>
         <cite:ansprechpartner>Schneider</cite:ansprechpartner>
         <cite:ap_email>schneider@localdomain.net</cite:ap_email>
         <cite:the_geom>
            <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#31467">
               <gml:coordinates xmlns:gml="http://www.opengis.net/gml"
                                decimal="." cs="," ts="  ">3405100,5758500
               </gml:coordinates>
            </gml:Point>
         </cite:the_geom>
         <cite:raum_bild>http://192.168.133.128/fotos/DSC08112_1.JPG</cite:raum_bild>
      </cite:raumview>
   </gml:featureMember>
</wfs:FeatureCollection>
```

Aus dem Abfrageergebnis wird ein neuer temporärer Layer erzeugt, welcher die gefilterten Features in der MapPanel und im GridPanel anzeigt (Abb. 17):

```
listeners: {
        actioncomplete: function(form, action) {
                features = action.response.features;
                store.loadData(features);
                vm=map.getLayersByName("Suchergebnis");
                if(vm.length==0){
                      vecLayer = new OpenLayers.Layer.Vector("Suchergebnis",
                      {styleMap: styles});
                      map.addLayer(vecLayer);
                      store.bind(vecLayer);
                      select.bind(vecLayer);
                }

        }
}
```

Mit den logischen Filter-Operatoren

- OpenLayers.Filter.Locical.AND = “&&”;
- OpenLayers.Filter.Logical.OR = “||”;
- OpenLayers.Filter.Logical.NOT = “!”

können auch mehrere Suchkriterien miteinander verbunden werden. Dazu könnten mehrere Suchfelder in Form von Auswahlboxen bereitgestellt werden.

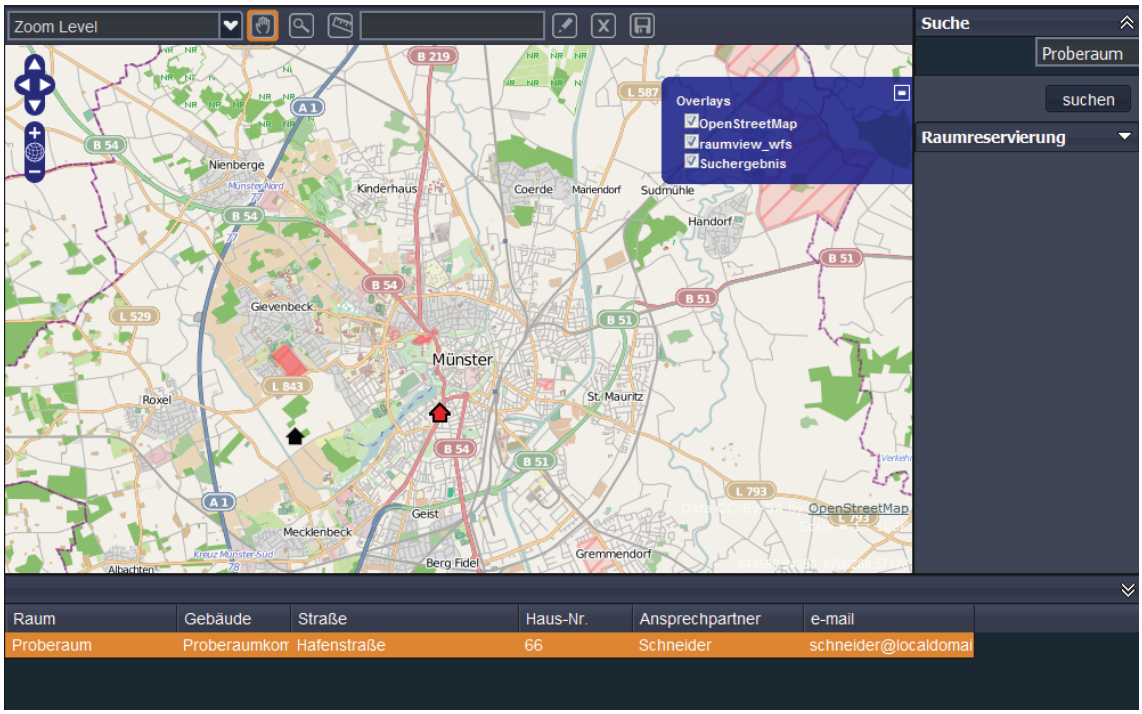


Abb. 17 Darstellung des Suchresultats.

6.6.7 Belegungskalender

Kulturschaffende, die einen Raum für eine bestimmte Zeitspanne belegen möchten, können dies im Bereich „Raumreservierung“ durchführen. Das Formular ist mit der Tabelle ‚Kalender’ der PostGIS-Datenbank verknüpft. Der Nutzer kann den gewünschten Zeitraum über ein ExtJs-DateField auswählen (Abb. 16). Beim Klicken des „Senden“-Buttons, wird mit PHP eine Verbindung zur Datenbank hergestellt und anschließend die Formulardaten in die Datenbank geschrieben (Beispiel 10). Die Übermittlung der Daten an den Server wird mit einem AJAX-Request ausgeführt (Beispiel 11). Abschließend erscheint ein Fenster mit einer Bestätigung, dass die Reservierung erfolgreich durchgeführt wurde (Abb. 19).

```
<?php

//Verbindung mit der Datenbank
$conn = pg_connect("port=5432 dbname=mt_kult_imm user=postgres
        password=postgres");


if (!$conn) {
  echo "An error occured.\n";
  exit;
}

//fügt Benutzer und Reservierungsdatum in die Datenbank ein
$p1 = $_POST['nutzer'];
$p2 = $_POST['raum_id'];
$p3 = encodeDate($_POST['anf_zeit']);
$p4 = encodeDate($_POST['end_zeit']);

  $query = "INSERT INTO belegungskalender (nutzer, raum_id, anf_zeit,
            end_zeit) VALUES ('$p1','$p2','$p3','$p4')";
  $result = pg_query($query);
  echo '1';

//wandelt das Datumsformat um (dd/mm/YYYY) zu (YYYY-mm-dd)
function encodeDate ($date) {
  $tab = explode ("/", $date);
  $r = $tab[2]."-".$tab[1]."-".$tab[0];
  return $r;
}

?>
```

Beispiel 10 Übermittlung der Reservierungsdaten mit PHP.

Der Eigentümer kann auf diese Art Nutzungsdaten sammeln und sie für die Auswertung der Auslastung heranziehen.

```
function saveNewBelegung(){

      Ext.Ajax.request({
        waitMsg: 'Wait a second...',
        url: '/belegungskalender.php',
        params: {
                  nutzer:   Txt_nutzer.getValue(),
                  raum_id:  Txt_raum_id.getValue(),
                  anf_zeit: Txt_anf_zeit.getValue().format('d/m/Y'),
                  end_zeit: Txt_end_zeit.getValue().format('d/m/Y')
        },
        success: function(response){
          var result=eval(response.responseText);
          switch(result){
            case 1:
                  Ext.MessageBox.alert('OK','Ihre Reservierung wurde
                  durchgeführt.');
                  break;
            default:
                  Ext.MessageBox.alert('Error','Bei Ihrer Reservierung
                  ist leider ein Fehler aufgetreten.');
                  break;
          }
        },
        failure: function(response){
          var result=response.responseText;
          Ext.MessageBox.alert('Error','Could not connect to the
          database');
        }
      });
}
```

Beispiel 11 Reservierungsfunktion mit AJAX-Request.

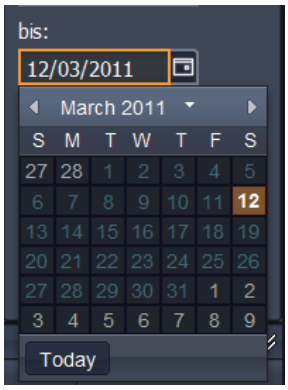


Abb. 18 ExtJs-Datefield.

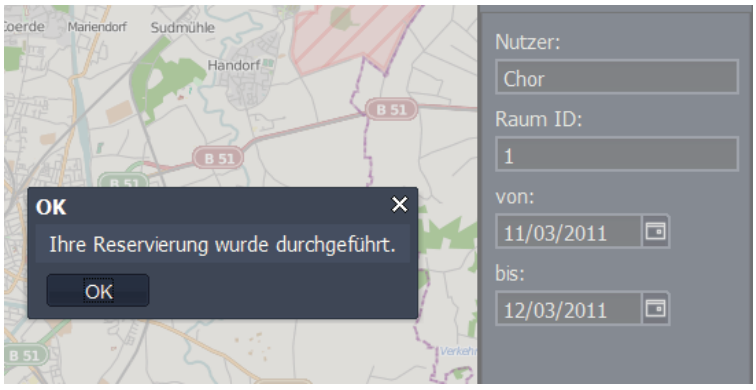


Abb. 19 Reservierungsbestätigung.

Das Beispiel sieht eine Belegung ganzer Tage vor. Die Funktion lässt sich jedoch erweitern, indem auch die gewünschte Uhrzeit angegeben werden kann. So kann sie für die Mischnutzung eingesetzt werden. In der Datenbank ist das Datumsformat als Timestamp definiert. Aus den Buchungsdaten lässt sich ein Belegungskalender generieren, der für jeden Raum beispielsweise in einem Popup oder einem eigenen Fenster angezeigt werden kann.

6.6.8 Räumliche Analyse

Mit OpenLayers können für einfache räumliche Analysen die Spatial Filter verwendet werden:

- OpenLayers.Filter.Spatial.BBOX
- OpenLayers.Filter.Spatial.INTERSECTS
- OpenLayers.Filter.Spatial.DWITHIN

Für umfangreichere Analysen kann in einem WebGIS der unter 5.4.7 beschriebene WPS eingesetzt werden. Damit lassen sich Flächen verschneiden oder Buffer berechnen, um beispielsweise herauszufinden, ob sich in einem bestimmten Umkreis um ein Gebäude Haltestellen für öffentliche Verkehrsmittel befinden.
Standardmäßig wird der WPS von OpenLayers noch nicht unterstützt. Die Firma 52°North hat ein WPS-Plug-in für OpenLayers entwickelt, der den OGC-Standard implementiert und in die Anwendung integriert werden kann. Da aber GeoServer noch keine Visualisierung des Datenoutputs ausgeführter Prozesse unterstützt, ist es im Moment sinnvoller, Geoprocessing in Desktop-GIS durchzuführen. Im Bereich der Desktop-GIS finden sich mit QGIS oder GvSig interessante Open Source-Lösungen.

7 Fazit und Ausblick

7.1 Fazit

Ziel der vorliegenden Arbeit war es, ein Konzept für eine interaktive Open Source WebGIS-Anwendung zu entwickeln, die Daten über leerstehende und kulturell nutzbare Räumlichkeiten bereitstellt und die Kommunikation zwischen Kulturschaffenden und Verwaltern öffentlicher Immobilien erleichtert, und dieses Konzept anschließend beipielhaft umzusetzen.

Wichtig für die Umsetzung war, dass die Benutzeroberfläche klar strukturiert, selbsterklärend, intuitiv bedienbar und übersichtlich ist, da sich die Anwendung an ein Publikum ohne GIS-Vorkenntnisse richtet. Die Wahl eines geeigneten Web-Clients fiel auf OpenLayers, das durch eigene JavaScript-Programmierung unter Zuhilfenahme der Bibiliotheken ExtJs und GeoExt den Anforderungen nachkommend modifiziert wurde.

Durch die Aufteilung der Oberfläche in MapPanel, GridPanel, Toolbar, Suchfeld und Raumbuchungsformular ist die Anwendung in klare, funktionale Bereiche gegliedert. Eine intuitive Bedienung der Anwendung ist durch die selbsterklärende Symbolik der Werkzeuge in der Toolbar gegeben.

Mit den Raum- und Gebäudelayern erhalten die Kulturschaffenden einen Überblick über die zur Verfügung stehenden Raumangebote. Die Daten werden in einer PostGIS-Datenbank gehalten, die durch die Anbindung an den Kartenserver und die damit gewährleistete Verteilung mit Hilfe von OGC-Services auch anderen Clients zugänglich gemacht werden können. Die Mitarbeiter der Immobilienverwaltung haben mit den Editiertools die Möglichkeit die Datenpflege über den WFS-T direkt in der Webanwendung vorzunehmen.

Die Art und der Umfang der Datenbereitstellung hinsichtlich der unterschiedlichen Ansprüche der Nutzer können zum einen durch die Visualisierung der Daten nach bestimmten Kriterien, durch die Festlegung der anzuzeigenden Layer oder durch die Verwendung von Datenbankviews angepasst werden.

Eine Suche nach Räumen mit bestimmten Anforderungen wie Größe oder Raumhöhe kann über das Suchfeld erledigt werden. Die Interaktion von Seiten der Kulturschaffenden wurde mit einem Formular für die Raumreservierung und die Möglichkeit, eine e-mail aus dem System heraus zu versenden umgesetzt.

Die Anwendung stellt ein Kommunikationsangebot dar, das von möglichst vielen Kulturschaffenden genutzt werden soll. Sie kann durch die Nutzung von JavaScript in jede Webseite eingebettet und angepasst werden. So kann ein größerer Nutzerkreis erreicht werden. Datengrundlage bleibt stets die PostGIS-Datenbank.
Dahingehend, dass solch ein Angebot nicht genutzt wird, kommen keine Bedenken auf. Von der vorgestellten WebGIS-Anwendung können beide Seiten profitieren.
Was die Analysemöglichkeiten angeht, vor allem komplexere und räumliche Analysen, kann festgestellt werden, dass hier Desktop-Anwendungen mehr zu empfehlen sind. Im Open Source Bereich bieten QGIS und GvSig eine interessante Alternative.

7.2 Ausblick

Die vorgestellte Lösung bezieht sich nur auf die Kommune als Eigentümer. Um das Spektrum an verfügbaren Immobilien zu erweitern, könnte die Öffnung der Anwendung für private Eigentümer und die Bereitstellung ihrer Immobilien sowohl im Interesse der Kulturschaffenden, als auch der Privateigentümer selbst sein. Hier muss aber aus datenschutzrechlichen Gründen zuvor geklärt werden, wo die Eigentümerdaten gespeichert werden.
Ein weiterer Punkt in zukünftigen Entwicklungen wird sicher die 3D-Visualisierung sein. Hier kommt City-GML ins Spiel. Dazu kann auf die City-GML-Erweiterung für das Facility Management (BLEIFUß 2009) verwiesen werden.
Die Möglichkeiten der Datenanalyse werden stetig weiterentwickelt. Die Nutzung des WPS wird in naher Zukunft WebGIS-Anwendungen um wichtige GIS-Funktionalitäten erweitern. Seit 2009 arbeitet eine Arbeitsgruppe an Version 2.0.0. Anfang 2011 soll der neue Standard bereits in einigen Open Source Anwendungen implementiert sein (http://www.ogf.org/OGF28/materials/1971/OGC-OGF-Session1-4-kiehle.pdf).
Sobald ein gewisser Bestand an Daten über die Nutzung von Räumen existiert, können diese als Grundlage für die Analyse der Auslastung und als Entscheidungshilfe bei der Flächen- und Standortplanung dienen. Analyseergebnisse können im Sinne eines PPGIS (Public Participatory GIS) im Internet von den Bürgern diskutiert werden, wenn es beispielsweise darum geht, bestimmte Einrichtungen aufgrund der fehlenden Auslastung zu schließen. Dies lässt sich auch auf andere Bereiche übertragen, zum Beispiel auf die Auslastung von Schulen und die damit verbundene Schulentwicklungs-planung.

Literaturverzeichnis

BECKER, J. (2009): Standortanalyse Kultur- und Kreativwirtschaft. Books on Demand. Norderstedt

BEYERSDORFF, M. (2007): Effektive Gestaltung des kommunalen Immobilienmanagements. Books on Demand. Norderstedt

Bleifuß, R. (2009): 3D-Innenraummodellierung: Entwicklung und Test einer CityGML-Erweiterung für das Facility Management. Diplomarbeit TU München

BMWi (2009): Endbericht Kultur- und Kreativwirtschaft: Ermittlung der gemeinsamen charakteristischen Definitionselemente der heterogenen Teilbereiche der „Kulturwirtschaft" zur Bestimmung ihrer Perspektiven aus volkswirtschaftlicher Sicht

BÜRGI, M. ET AL. (1995): Zwischennutzung Kulturraum: Soziokulturelle Zentren in zwischengenutzten Industrie- und Gewerbebauten. Diplomarbeit höhere Fachschule für Soziale Arbeit Solothurn

DEHNHARDT, W. (2001): Scriptsprachen für dynamische Webauftritte – JavaScript, VBScript, ASP, Perl, PHP, XML. Carl Hanser Verlag. München

DONAUBAUER, A.J. (2004): Interoperable Nutzung verteilter Geodatenbanken mittels standardisierter Geo Web Services. Dissertation. TU München

ECKERT, C. (2009): IT-Sicherheit. Konzepte – Verfahren – Protokolle. Oldenbourg Wissenschaftsverlag. München

FLANAGAN, D. (2007): JavaScript: Das umfassende Referenzwerk. O'Reilly. Köln

HÄUßERMANN, H. & SIEBEL, W. (1987): Neue Urbanität. Suhrkamp. Frankfurt

HMWVL (Hessisches Ministerium für Wirtschaft, Verkehr und Landesentwicklung) (Hrsg.) (2008): Kulturwirtschaft fördern – Stadt entwickeln. 3. Hessischer Kulturwirtschaftsbericht

HILLMANN, K.H. (1994): Wörterbuch der Soziologie. Kröner. Stuttgart

INITIATIVE D21 (2010): (N)ONLINER Atlas 2010 – Eine Topographie des digitalen Grabens durch Deutschland. Nutzung und Nichtnutzung des Internets, Strukturen und regionale Verteilung – (http://www.initiatived21.de/wp-content/uploads/2010/06/NONLINER2010.pdf – zuletzt geprüft am 24.02.2011)

JANSEN, M. & ADAMS, T. (2010): OpenLayers. Webentwicklung mit dynamischen Karten und Geodaten. Open Source Press. München

KLAUS, P. (2006): Stadt, Kultur, Innovation. Kulturwirtschaft und kreative innovative Kleinstunternehmen in der Stadt Zürich. Seismo Verlag. Zürich

KORDUAN, P. & ZEHNER, M.L. (2008): Geoinformation im Internet. Wichmann. Heidelberg

MITCHELL, T., EMDE, A. & CHRISTL, A. (2008): Web Mapping mit Open Source-GIS-Tools. O'Reilly. Köln

MUSCIANO, C. & KENNEDY, B. (2003): HTML und XHTML. Das umfassende Referenzwerk. O'Reilly. Köln

OGC (2005a): OpenGIS® Filter Encoding Implementation Specification. OGC 04-095. Version: 1.1.0

OGC (2005b): Web Feature Service Implementation Specification. OGC 04-094. Version: 1.1.0

OGC (2006a): OpenGIS® Web Map Server Implementation Specification. OGC® 06-042. Version: 1.3.0

OGC (2006b): Symbology Encoding Implementation Specification. OGC 05-077r4. Version: 1.1.0

OGC (2006c): OGC Web Services Common Specification. OGC 06-121r3. Version: 1.1.0

OGC (2007a): Styled Layer Descriptor profile of the Web Map Service Implementation Specification. OGC 05-078r4. Version: 1.1.0 (revision 4)

OGC (2007b): OpenGIS® Geography Markup Language (GML) Encoding Standard. OGC 07-036. Version: 3.2.1

OGC (2007c): Web Processing Service Implementation Specification. OGC 05-007. Version: 1.0.0

OGC (2010a): OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. OGC 06-103r4. Version: 1.2.1

OGC (2010b): OpenGIS Web Feature Service 2.0 Interface Standard. OGC 09-025r1

OGC (2010c): OpenGIS Filter Encoding 2.0 Encoding Standard. OGC 09-026r1

PORTZ, N. & DÜSTERDIEK, B. (1999): DStGB Dokumentation Nr. 8: Kommunales Immobilienmanagement - Konzepte und Lösungsansätze zur Optimierung der kommunalen Immobilien-wirtschaft. In: Stadt und Gemeinde interaktiv. Jg. 54. Nr. 5/99. Winkler & Stenzel. Burgwedel

RFC 793 (1981): TRANSMISSION CONTROL PROTOCOL

RFC 2460 (1998): Internet Protocol, Version 6 (IPv6) Specification

RFC 2616 (1999): Hypertext Transfer Protocol -- HTTP/1.1

RFC 3986 (2005): Uniform Resource Identifier (URI): Generic Syntax

SAUBERZWEIG, D. (1989): Fragen an eine Kulturpolitik der Stadt. In: Universitas, Juni 1989

SCHÖNEICH, M. (1991): Stadtkultur. In: DifU (Hrsg.): Urbanität in Deutschland. Schriften des DifU Band 83. Verlag W. Kohlhammer GmbH/ Deutscher Gemeindeverlag. Stuttgart / Berlin / Köln

SCHULTE, K.-W., SCHÄFERS, W. et al. (2006): Handbuch Immobilienmanagement der öffentlichen Hand. Rudolf Müller. Köln

SKULSCHUS, M. & WIEDERSTEIN, M. (2008): XML: Standards und Technologien. Comelio Medien. Essen

STEINMANN, R., KREK, A. & BLASCHKE, TH. (2004): Can Online Map-Based Applications Imrove Citizen Participation? In: Lecture Notes in Computer Science, Springer Verlag, TED conference on e-government, Bozen, Italy

STEYER, R. (2007): Das Javascript Codebook. Addison-Wesley. München

W3C (2004): Document Object Model (DOM) Level 3 Core Specification Version: 1.0

WASSERMANN, T. (2007): Sichere Webanwendungen mit PHP: Sicherheit mit PHP, MySQL, Apache, JavaScript, AJAX. mitp. Heidelberg

WEHRLI-SCHINDLER, B. (2002): Kulturelle Einrichtungen als Impulsgeber für Stadtentwicklungen? Beobachtungen am Beispiel Zürich West. DISP 150-3

Internetquellen

WIKIPEDIA – Ajax-Modell
http://de.wikipedia.org/w/index.php?title=Datei:Ajax-vergleich.svg&filetimestamp=20070826155716 - zuletzt geprüft am 22.02.2011

http://news.netcraft.com/archives/category/web-server-survey/ - zuletzt geprüft am 25.02.2011

http://geoext.org/ - zuletzt geprüft am 23.02.2011

http://www.ogf.org/OGF28/materials/1971/OGC-OGF-Session1-4-kiehle.pdf - zuletzt geprüft am 29.12.2010

http://www.opengeospatial.org/ - zuletzt geprüft am 20.02.2011

http://www.W3C.de - zuletzt geprüft am 20.02.2011