4. RAHMENPAPIER

GDI – SÜDHESSEN

Technische Voraussetzungen zum Aufbau von Geodateninfrastrukturen

VERSION 1.0

25.03.2008

Auftraggeber

HESSEN

Hessische Verwaltung für Bodenmanagement und Geoinformation

Ansprechpartner (Projektleitung)

Dr.-Ing. Thomas Rossmanith

Amt für Bodenmanagement Heppenheim
Europaplatz 5
64293 Darmstadt

Email: thomas.rossmanith@hvbg.hessen.de
Tel.: 06151 / 50 04 - 303

Dipl.-Ing. (FH) Anja Schupp

Hessisches Landesamt für
Bodenmanagement und Geoinformation
Schaperstraße 16
65195 Wiesbaden

Email: anja.schupp@hvbg.hessen.de
Tel.: 0611 / 535 - 54 86

Weitere Informationen zur GDI-Südhessen unter **http://www.gdi-suedhessen.de**.

INHALTSVERZEICHNIS

1. Technische Voraussetzungen zum Aufbau von GDI’en 4

1.1 Definition einer verteilten Anwendung 4

1.2 Komponenten von verteilten Anwendungen 4

1.3 Argumente für eine verteilte Anwendung 5

1.4 Architekturen verteilter Anwendungen 6

1.4.1 2-Tier-Architektur 7

1.4.2 3-Tier-Architektur 8

1.4.3 4-Tier-Architektur 9

2. GDI als verteilte Anwendung 10

3. Praktische Beispiele 12

4. Weiterführende Literatur 15

1. Technische Voraussetzungen zum Aufbau von GDI’en

Die technische Voraussetzung zum Aufbau und zur effektiven Nutzung einer Geodateninfrastruktur bildet eine **verteilte Anwendung**. Nach der Definition der verteilten Anwendung werden zunächst anhand eines Beispiels die einzelnen Komponenten vorgestellt und im Anschluss daran mögliche Architekturen aufgezeigt.

1.1 Definition einer verteilten Anwendung

Um den Aufbau einer verteilten Anwendung nachvollziehen zu können, muss zunächst der Begriff eines verteilten Systems erläutert werden. Bei einem verteilten System befinden sich die verschiedenen Hardware- und Softwarekomponenten auf unterschiedlichen, unabhängigen Rechnersystemen, die miteinander vernetzt sind. Durch den Austausch von Nachrichten können die Komponenten kommunizieren und somit ihre Aktionen untereinander koordinieren. Das bekannteste Beispiel für ein verteiltes System ist das Internet. Durch Verwendung des Internets können die Nutzer unabhängig von ihrem momentanen Standort verschiedene Dienste und Anwendungen, die sich auf unterschiedlichen Rechnern/Servern befinden, nutzen.
Eine verteilte Anwendung nutzt die Technologie eines verteilten Systems, d.h. auch hier befinden sich die verschiedenen Komponenten (Hard- und Software) auf unterschiedlichen Computer. Über ein Rechnernetzwerk, wie beispielsweise Intranet oder Internet, können die Komponenten sowohl untereinander als auch mit einem Anwender kommunizieren. Beispiele für verteilte Anwendungen sind Internetshops und Flugbuchungssysteme.

1.2 Komponenten von verteilten Anwendungen

Am Beispiel des Internetversandhandels *Amazon* lassen sich die Komponenten einer verteilten Anwendung einfach ausfindig machen. *Amazon* ist nach eigenen Angaben der Markführer unter den Internetshops mit der weltweit größten Auswahl an Bücher, CDs und Videos. Um beispielsweise ein Buch bei *Amazon* bestellen zu können, muss der Nutzer über seinen Web-Browser zunächst die Internetseite von *Amazon* aufrufen. Unter Verwendung von Stichworten kann nach einem passenden Buch recherchiert werden. *Amazon* hat eine sehr große Auswahl an Produkten, so dass es zweckmäßig ist, die Informationen zu den einzelnen Produkten, wie z.B. Titel, Autor, ISBN-Nummer, mit Hilfe einer Datenbank zu verwalten. Um die Funktionalität von *Amazon* zu vervollständigen, werden weitere Anwendungen beispielsweise für die Produktsuche sowie den Bestell- und Abrechnungsvorgang benötigt. Es lassen sich demnach bei *Amazon* als verteilte Anwendung folgende vier grundlegende Komponenten identifizieren:

- Client,
- Webserver,
- Applikationsserver,
- Datenbankserver.

Diese vier Komponenten entsprechen den grundlegenden Komponenten einer verteilten Anwendung. Weiterhin können die Komponenten in die Gruppe der Clients und Server unterteilt werden.

Im Allgemeinen handelt es sich bei einem Client um ein Programm, das auf die Ressourcen eines Servers zugreifen kann und mit diesem über ein Rechnernetzwerk kommuniziert. Ein Client kann in zwei unterschiedlichen Ausprägungen zum Einsatz kommen. Es wird hierbei zwischen einem Thin Client und einem Fat Client unterschieden. Bei einem Fat Client handelt es sich um eine spezielle Software, welche die Daten, die auf einem Server abgelegt sind, selbstständig verarbeitet. Im Gegensatz hierzu enthält ein Thin Client keine Anwendungslogik, sondern stellt eine Präsentationsschnittstelle oder grafische Benutzeroberfläche zur Präsentation der Daten für den Nutzer bereit. Die heute gängigste Variante eines Thin Clients ist ein einfacher Web-Browser wie beispielsweise *Microsoft Internet Explorer*, *Firefox* von *Mozilla* oder *Netscape Navigator*.

Für gewöhnlich ist ein Server ein Programm, welches auf eine Anfrage von einem Client wartet, diese bearbeitet und schließlich beantwortet. Generell können unterschiedliche Kategorien von Servern unterschieden werden. Im Folgenden sollen nur Webserver, Applikationsserver und Datenbankserver näher betrachtet werden.

Ein **Webserver** stellt einem Client Informationen, z.B. Internetseiten, über das HTTP-Protokoll zur Verfügung. Aus diesem Grund werden Webserver oft als HTTP-Server bezeichnet. Die beiden heute bedeutendsten Webserver sind der *Internet Information Server* (IIS) von *Microsoft* und der A*pache HTTP-Server* von der *Apache Software Foundation*, wobei es sich bei dem zweiten Produkt um eine Open-Source Lösung handelt.

Ein **Applikations- oder Anwendungsserver** ist ein Server in einem Computernetzwerk auf dem spezielle Software-Applikationen ausgeführt werden. Des Weiteren stellt ein Applikationsserver für gängige Problemstellungen bereits integrierte Lösungen, wie Sicherheitslösungen oder Entwicklungswerkzeuge zur Verfügung. Auf dem Markt erhältliche Produkte sind beispielsweise *BEA WebLogic*, *IBM WebSphere*, *Oracle* und *JBoss Application Server*. Mit dem JBoss Application Server steht auch in dieser Kategorie ein Open-Source Produkt zur Verfügung.

Die letzte vorzustellende Gruppe der Server stellen die **Datenbankserver** dar. Ein Datenbankserver verwaltet die notwendigen Daten in einer verteilten Anwendung und stellt diese einem Client oder einer weiteren Anwendung zur Verfügung. Gängige Datenbanken sind *Oracle*, *MySQL*, *Microsoft SQLServer* und *PostgreSQL*. MySQL und PostgreSQL sind zwei im Bereich der Datenhaltungskomponenten weit verbreitete Open-Source Produkte.

1.3 Argumente für eine verteilte Anwendung

Das Gegenteil einer verteilten Anwendung ist eine zentrale Anwendung. Im Allgemeinen sind zentrale Anwendungen meist sicherer und performanter als verteilte Anwendungen. Dennoch gibt es Argumente, die für eine verteilte Anwendungen sprechen. Das Ziel einer verteilten Anwendung ist die gemeinsame Nutzung von Ressourcen, wobei unter Ressourcen Hardware, Daten und Funktionalität zu verstehen sind.

Eine gemeinsame Nutzung von Hardware kann beispielsweise die Einrichtung eines gemeinsamen Druckers innerhalb einer Abteilung eines Unternehmens sein. Demzufolge muss nicht an jedem Arbeitsplatz ein eigener Drucker zur Verfügung gestellt werden, so dass auf diese Weise Kosten eingespart werden können. Eine gemeinsame Nutzung von Daten wird durch die Verwendung einer zentralen Datenbank möglich. Auf diese Weise werden Datenredundanzen vermieden und sämtliche Maßnahmen, um Daten konsistent zu halten, können entfallen. Letztendlich besteht die Möglichkeit, Funktionen oder Anwendungen gemeinsam zu nutzen. Hierdurch wird eine Wiederverwendung der Komponenten ermöglicht, wodurch Fehler vermieden werden können.
Wie an den oben genannten Beispielen zu erkennen ist, führt die Nutzung gemeinsamer Ressourcen zur Kosteneinsparungen. Durch Anlegen redundanter Komponenten innerhalb einer verteilten Anwendung kann zusätzlich die Ausfallsicherheit der gesamten Anwendung erhöht werden. Fällt beispielsweise der Datenbankserver innerhalb einer verteilten Anwendung aus, so kann auf einfache Art und Weise eine Ersatzdatenbank angeschlossen werden. Dadurch sind verteilte Anwendungen sehr gut skalierbar, da einzelne Komponenten einfach ausgetauscht oder neue Komponenten hinzugefügt werden können. Durch Aufspaltung der Anwendung in einzelne Komponenten wird die Gesamtlast der Anwendung ebenfalls auf diese Komponenten verteilt.
Auf diese Weise wird die Performance erhöht und es kommt zu schnelleren Antwortzeiten.

1.4 Architekturen verteilter Anwendungen

Die vier Komponenten (Client, Webserver, Applikationsserver, Datenbankserver) einer verteilten Anwendung können auf unterschiedliche Art und Weise miteinander kombiniert und angeordnet werden. Für die möglichen Architekturen hat sich der Oberbegriff der *N-Tier-Architekturen* gebildet („Tier“ engl. für Schicht oder Ebene). Im Allgemeinen wird bei der N-Tier-Architektur zwischen den drei folgenden Schichten unterschieden:

- Präsentationsschicht,
- Anwendungslogik,
- Datenhaltung.

Die Präsentationsschicht stellt die Schnittstelle zum Anwender dar. Sie entspricht im Grunde einer grafischen Benutzeroberfläche für einen Nutzer. Die Ebene mit der Anwendungslogik dient zur Bearbeitung und Beantwortung der Anfragen des Clients. Auf der dritten Schicht sind die eigentlichen Daten der Anwendung in einer Datenbank abgelegt.
Die Ebenen einer N-Tier-Architektur und die Komponenten einer verteilten Anwendung können unterschiedlich kombiniert werden. Hieraus ergeben sich die drei folgenden Architekturen:

- 2-Tier-Architektur,
- 3-Tier-Architektur,
- 4-Tier-Architektur.

In den nachfolgenden Abschnitten sollen diese drei Architekturen vorgestellt und auf die wesentlichen Vor- und Nachteile eingegangen werden.

1.4.1 2-Tier-Architektur

Die 2-Tier-Architektur ist die einfachste Variante der N-Tier-Architektur. Die vier Komponenten einer verteilten Anwendung werden lediglich auf zwei Ebenen aufgeteilt. Die Abbildung 1-1 zeigt den Aufbau einer 2-Tier-Architektur.

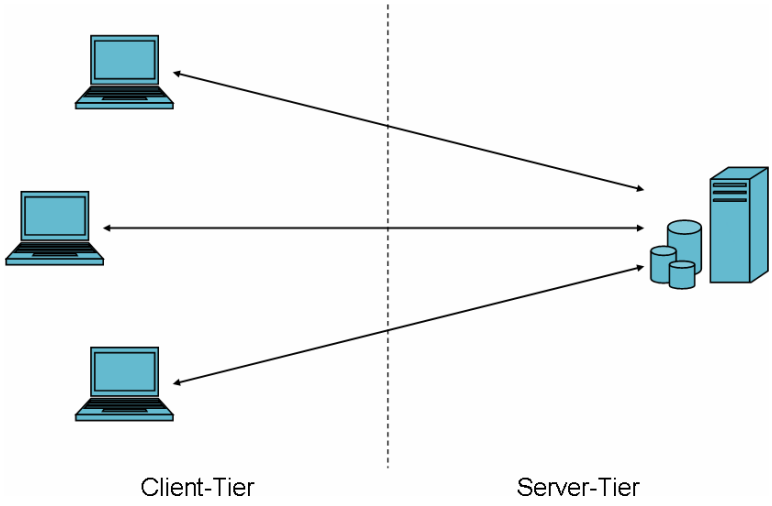


ABBILDUNG 1-1: 2-TIER-ARCHITEKTUR

Die Präsentationsschicht befindet sich bei der 2-Tier-Architektur auf dem so genannten Client-Tier und die Datenhaltungsschicht auf dem Server-Tier. Problematisch bei der 2-Tier-Architektur ist, dass der Anwendungslogik keine eigene Ebene zur Verfügung steht. Für die Verteilung der Anwendungsschicht bestehen drei Möglichkeiten. Sie kann gemeinsam mit der Präsentationsschicht auf dem Client-Tier angesiedelt werden, so dass häufig Fat Clients zum Einsatz kommen. Dies bedeutet, dass auf jedem Client-Rechner oder bei jedem Nutzer der Anwendung eine spezielle Software installiert werden muss. Bei der zweiten Variante wird die Anwendungslogik gemeinsam mit den Daten auf dem Server-Tier untergebracht, wodurch der Einsatz von Thin Clients möglich wird. In der letzten Variante wird die Anwendungslogik zwischen dem Client- und dem Server-Tier aufgeteilt.
Die 2-Tier-Architektur ist einfach programmierbar und kann dadurch sehr schnell umgesetzt werden. Die Nachteile dieser Architektur überwiegen jedoch. Der wohl größte Nachteil ist, dass der Anwendungslogik keine eigene Schicht zugeordnet werden kann, so dass häufiger Fat Clients eingesetzt werden. Diese Client Software benötigt in der Regel einen Direktzugriff auf die Datenbank und ist somit sehr stark an diese gebunden, so dass ein Austausch der einzelnen Komponenten zu Problemen führen kann. Des Weiteren wird die 2-Tier-Architektur sehr schnell unübersichtlich und unstrukturiert und wird damit schwer skalierbar. In der Praxis kommt die 2-Tier-Architektur wegen der oben genannten Nachteile nur sehr selten zum Einsatz.

1.4.2 3-Tier-Architektur

Die 3-Tier-Architektur erweitert die 2-Tier-Architektur um das so genannte Middle-Tier. Die Anordnung der einzelnen Ebenen ist in Abbildung 1-2 dargestellt.

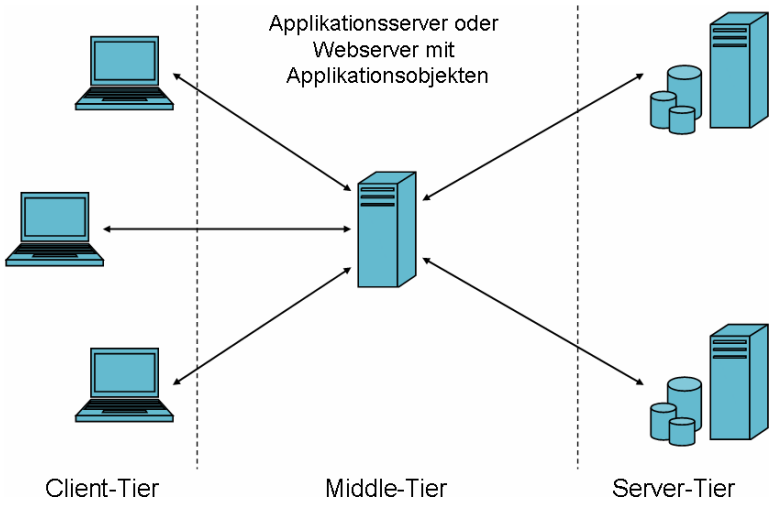


ABBILDUNG 1-2: 3-TIER-ARCHITEKTUR

Auf dem Client-Tier befindet sich lediglich die Präsentationsschicht, so dass Thin Clients, d.h. Web-Browser eingesetzt werden können. Das Middle-Tier enthält die Anwendungslogik und besteht entweder aus einem Applikationsserver oder einem Webserver, der mit Applikationsobjekten umgehen kann. Das Server-Tier enthält letztlich nur noch die eigentlichen Daten. Die 3-Tier-Architektur ist heute die gängigste Variante der N-Tier-Architektur. Eine einfache Internet-Anwendung mit Browser, Webserver und Datenbank ist ein Beispiel für die 3-Tier-Architektur.

Die 3-Tier-Architektur ist sehr gut skalierbar, so dass neue Komponenten einfach hinzugefügt oder Komponenten ausgetauscht werden können. Die Gesamtlast der Anwendung wird auf weitere Komponenten verteilt, so dass die Antwortzeiten beschleunigt werden können. Die Verwendung weiterer Schichten hat nicht nur Vorteile, sondern macht die Gesamtanwendung komplexer. Ein weiterer Nachteil ist, dass die Hersteller mittlerweile komplette Produkte für 3-Tier-Architekturen anbieten, so dass die eigenen Anwendungen häufig nicht mehr auf die Plattform eines anderen Herstellers übertragen werden können. Die Lösung für diese Problematik liefert die 4-Tier-Architektur.

1.4.3 4-Tier-Architektur

Die 4-Tier-Architektur stellt eine Verfeinerung der 3-Tier-Architekur dar. Hierbei wird das Middle-Tier nochmals unterteilt in einen Webserver und einen Applikationsserver (Abbildung 1-3). Ziel dieser Unterteilung ist es, die Architektur herstellerunabhängiger zu gestalten. Auf dem heutigen Markt existieren bisher nur zwei Technologieplattformen zur Realisierung von 4-Tier-Architekturen. Dies sind *Microsoft .NET-Plattform* und *J2EE-Plattform* von *Sun Microsystems*. Die 4-Tier-Architektur besitzt alle Vorteile einer 3-Tier-Architektur und ist zudem noch herstellerunabhängig. In der Praxis sind 4-Tier-Architekturen bisher nur selten anzutreffen.

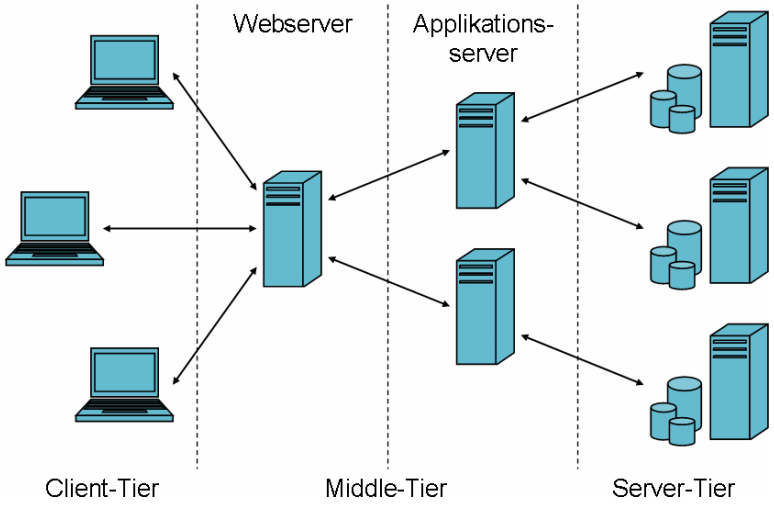


ABBILDUNG 1-3: 4-TIER-ARCHITEKTUR

2. GDI als verteilte Anwendung

Bei einer GDI handelt es sich um eine verteilte Anwendung, die der 3-Tier-Architektur zugeordnet werden kann. Im Folgenden sollen die einzelnen Komponenten der GDI den verschiedenen Schichten der Architektur zugeordnet und erläutert werden. Die Abbildung 2-1 liefert einen Überblick über die wichtigsten Komponenten und deren Einordnung in das Client-, Middle- und Server-Tier.

Das Client-Tier oder die Präsentationsschicht enthält die Nutzer der GDI. Das Middle-Tier umfasst die Anwendungslogik einer verteilten Anwendung. Bezogen auf die GDI sind dies die Komponenten des Geoportals, d.h. die verschiedenen Geoanwendungen und Dienste. Das Server-Tier beinhaltet die Datenbanken, welche sämtliche Daten der GDI, wie Geodaten, Geofachdaten und Metadaten, beinhalten. Im Folgenden sollen die einzelnen Schichten detaillierter betrachtet werden.

Im Allgemeinen kann bei der Client-Komponente zwischen einem Thin und einem Fat Client unterschieden werden. Im Rahmen der GDI kommen beide Clientvarianten zum Einsatz. Bei einem Thin Client handelt es sich meist um einen Web-Browser, mit dem Daten präsentiert werden. Innerhalb der GDI kann ein Thin Client beispielsweise zur Recherche in Metadaten oder zum Visualisieren eines Datendienstes in einem Kartenviewer (z.B. Hessen-Viewer) verwendet werden. Bei einem Fat Client handelt es sich in der Regel um eine spezielle Software, welche auf die Daten des Datenbankservers zugreifen kann. Durch die Verwendung von Datendiensten, wie beispielsweise dem Web Map Service (WMS) oder dem Web Feature Service (WFS), wird ein Direktzugriff auf die Datenbank verzichtbar. Des Weiteren sind viele Komponenten innerhalb einer GDI standardisiert bzw. OGC konform, so auch die zuvor genannten Datendienste. Aus diesen Gründen können im Rahmen der GDI Fat Clients zum Einsatz kommen, ohne eine spezielle Software verwenden zu müssen. Ein Beispiel hierfür wäre der Einsatz eines herkömmlichen Desktop-GIS als Fat Client zum Visualisieren eines WMS oder WFS gemeinsam mit den eigenen Daten.

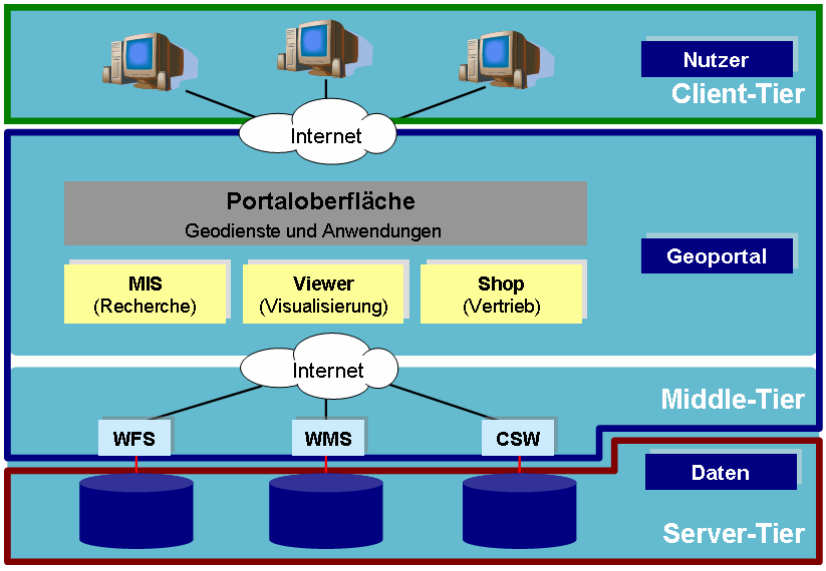


ABBILDUNG 2-1: KOMPONENTEN DER GDI IN EINER 3-TIER ARCHITEKTUR

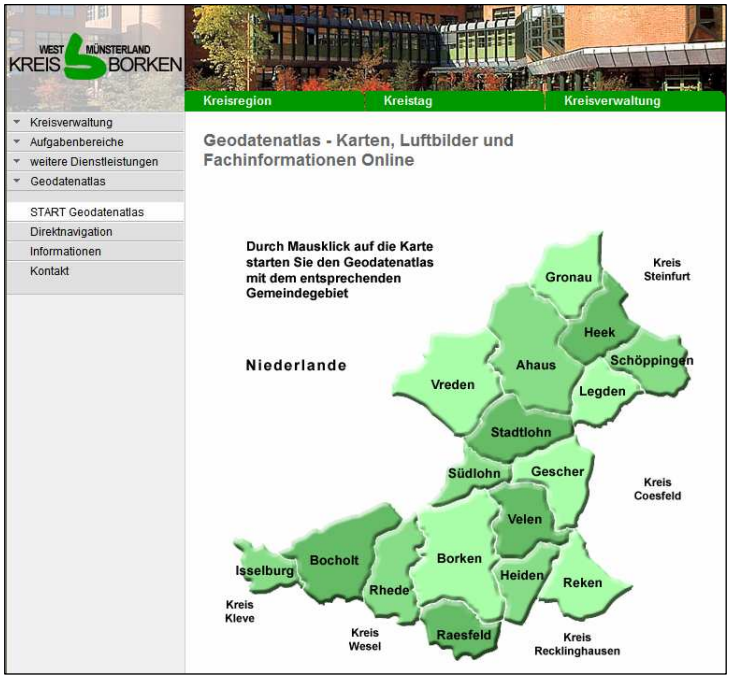
In einer verteilten Anwendung enthält das Middle-Tier die Anwendungslogik. Übertragen auf die GDI sind dies die Komponenten des Geoportals, d.h. die verschiedenen Geodienste und Anwendungen. Die wichtigsten Anwendungen sind ein Metadateninformationssystem (MIS) zur Recherche in den Metadaten, ein Viewer zur Visualisierung von verschiedenen Datendiensten (z.B. WMS oder WFS) und ein Shopsystem zum Vertrieb und zur Abrechnung der verschiedenen GDI-Produkte. Im Rahmen der GDI-Südhessen zählt der geplante *horizontale* und *vertikale Prozess* ebenfalls zu den Anwendungen. Bei den unterschiedlichen Diensten handelt es sich hauptsächlich um OGC Web Services. Die bedeutendsten Dienste umfassen den WMS, WFS und CSW (Catalogue Service for the Web). Der WMS dient der Visualisierung verteilter Geodaten, während der WFS zusätzlich einen Zugriff auf diese erlaubt. Mit Hilfe des CSW können die Metadaten zu Geodaten, Geodiensten und Anwendungen publiziert und bereitgestellt werden.

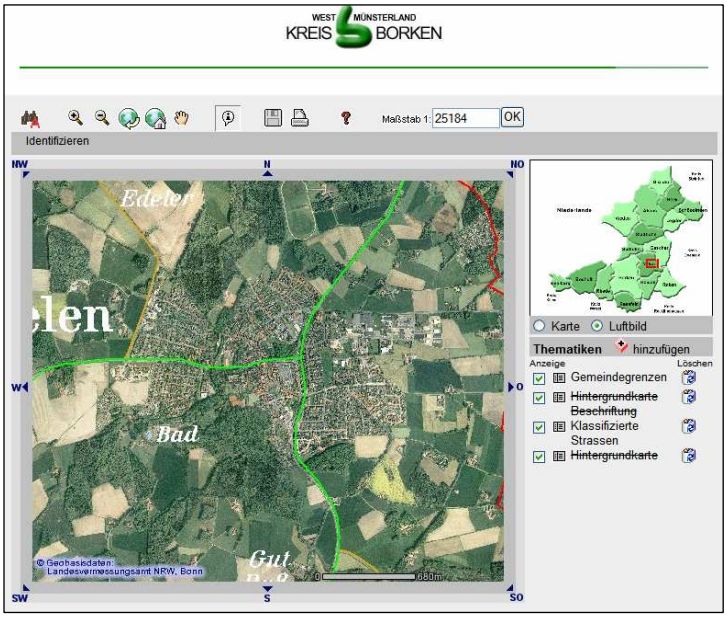
Die Datenhaltungsschicht befindet sich in der 3-Tier-Architektur auf dem Server-Tier. Auch innerhalb der GDI sind hier die notwendigen Daten in den entsprechenden Datenbanken aufzufinden. Die Daten der GDI können in die Bereiche der Geobasisdaten (z.B. ALK, ALB, ATKIS und Luftbilder), der Geofachdaten (z.B. Bauleitpläne, Umweltpläne und statistische Daten) und den zugehörigen Metadaten unterteilt werden. Der Standort der Datenbankserver befindet sich jeweils bei der für die Daten zuständigen Stelle.

3. Praktische Beispiele

Kreis Borken (Nordrhein-Westfalen)

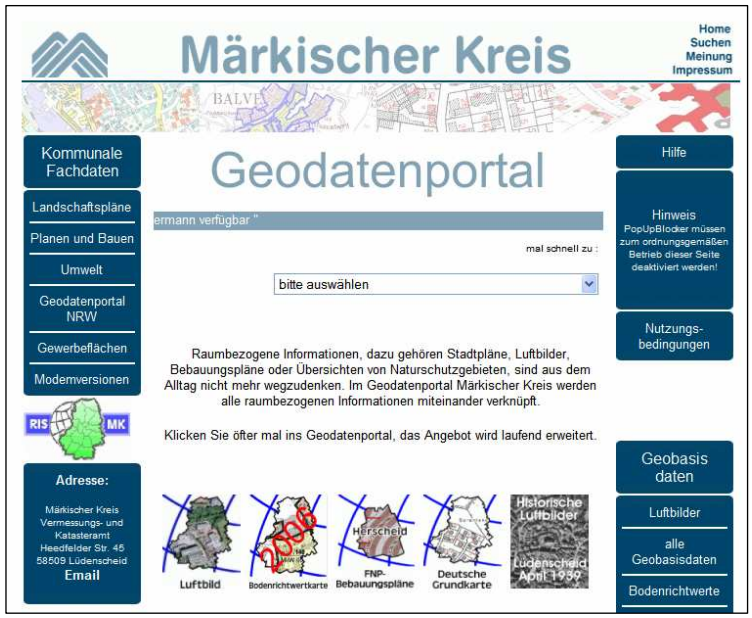
URL: http://www.kreis-borken.de; Link zum *Geodatenatlas*

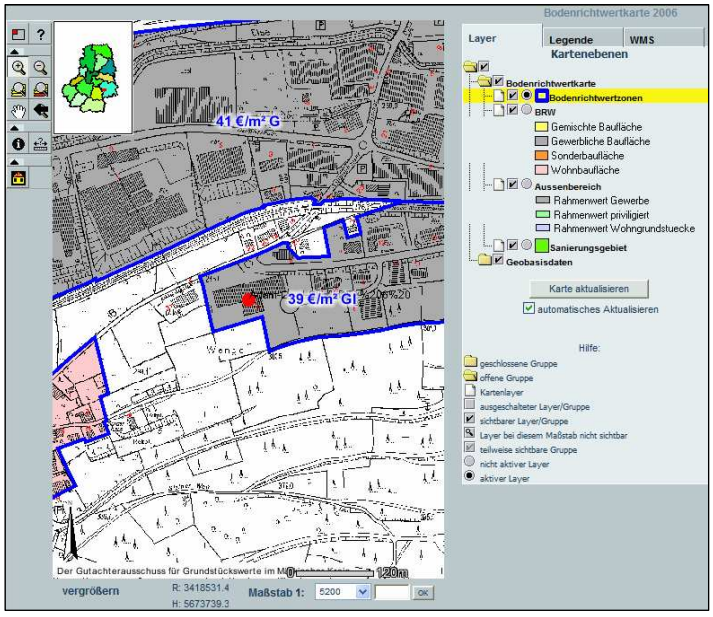


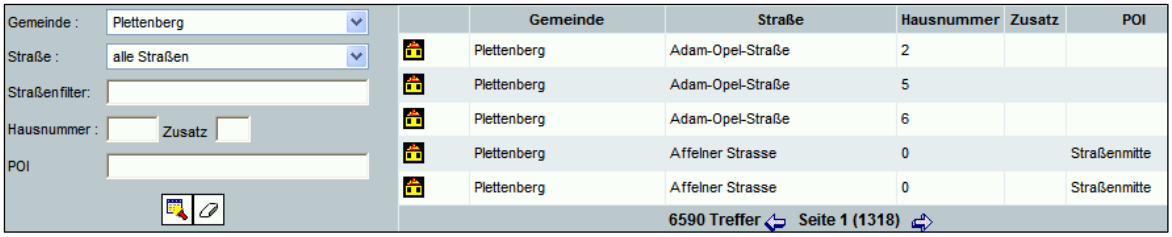


Märkischer Kreis (Nordrhein-Westfalen)

URL: http://ris.maerkischer-kreis.de

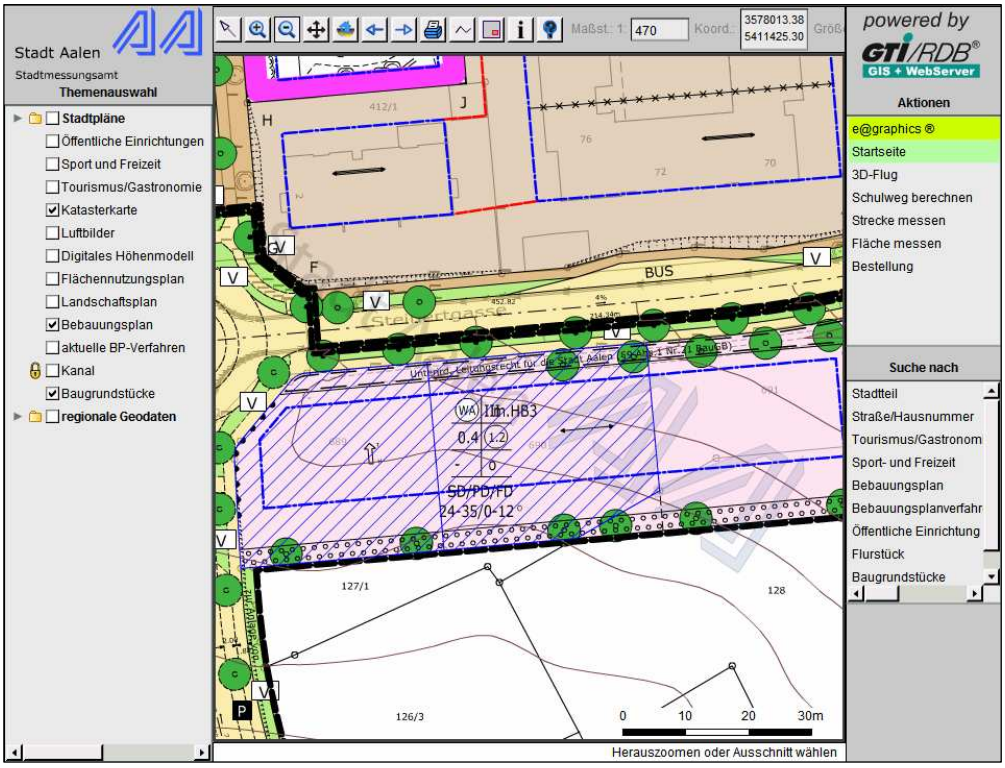






Stadt Aalen (Baden-Württemberg)

URL: http://www.gisserver.de/aalen/start.html



4. Weiterführende Literatur

Stand: 21.Juni 2006

Allgemeine Informationsquellen:

- http://de.wikipedia.org/wiki/Hauptseite

Verteilte Anwendungen:

- Eberhart, Andreas / Fischer, Stefan: Web Services - Grundlagen und praktische Umsetzung mit J2EE und .NET; München / Wien 2003; ISBN 3-446-22530-7
- Langner, Thorsten / Reiberg, Daniel: J2EE und JBoss – Verteilte Enterprise Applikationen auf der Basis von J2EE, JBoss & Eclipse; München / Wien 2006; ISBN 3-446-40508-9