Hochschule Neubrandenburg
University of Applied Sciences

Fachbereich Landschaftsarchitektur, Geoinformatik, Geodäsie und Bauingenieurwesen
Masterstudiengang Geoinformatik und Geodäsie

Konzeption und Entwicklung eines datenbankbasierten Systems zur Verwaltung von GNSS-Daten

Masterarbeit
vorgelegt von: Markus Bradke

Zum Erlangen des akademischen Grades
„Master of Engineering“ (M.Eng.)

Erstprüfer: Prof. Dr.-Ing. Andreas Wehrenpfennig
Zweitprüfer: Dr. Jens Wickert

Angefertigt am Helmholtz-Zentrum Potsdam Deutsches GeoForschungsZentrum GFZ
im Zeitraum vom 02. Januar 2012 - 17. August 2012

urn:nbn:de:gbv:519-thesis2012-0633-8

Kurzfassung

Die Beobachtung des Erdsystems und die Erforschung geodynamischer Prozesse mit Hilfe globaler Navigationssatellitensysteme (GNSS) haben in den letzten Dekaden stark an Bedeutung gewonnen. Die Modellierung dieses Systems basiert auf einer Vielzahl von Daten, die von global verteilten Beobachtungsstationen registriert werden. Für die Analyse und Auswertung dieser Daten müssen die Eigenschaften der Stationsempfänger und -antennen sowie der Satelliten bekannt sein. Die benötigten Informationen werden derzeit dateibasiert, redundant und ohne Möglichkeit zur Datenvalidierung verwaltet.

Das Ziel der vorliegenden Masterarbeit besteht in der Konzeption und Entwicklung eines datenbank- und webbasierten Systems zur Verwaltung von Beobachtungsstations- und Satellitenmetadaten. Die Applikation soll dazu beitragen den Verwaltungsaufwand bei der Pflege der Metainformationen zu minimieren. Die graphische Benutzeroberfläche wurde auf Basis von PHP 5.3, HTML und JavaScript entwickelt. Für die Verwaltung der Metainformationen wurde das objektrelationale Datenbankmanagementsystem PostgreSQL 8.4 genutzt. Durch den modularen und objektorientierten Programmieransatz lässt sich die Applikation leicht um zusätzliche Funktionalitäten erweitern.

Abstract

The monitoring of the earth system and the research of geodynamic processes based upon global navigation satellite systems (GNSS) became more significant in the last decades. The modelling of this system rested on a variety of data that are registered by globally distributed observation stations. The analysis and evaluation of these data require established features of the station's receivers and antennas as well as satellites. This information is currently redundantly stored in ASCII data files without an option for validation.

The present master thesis focuses on the conception and development of a database-based web application for the management of station and satellite metadata. The application shall contribute maintaining metainformation and minimizing administration effort. The implementation of the graphical user interface is based upon PHP 5.3, HTML and JavaScript. The data management restricts on the usage of the object-relational database management system PostgreSQL 8.4. Due to the modular and object-oriented programming approach the application is simply extensible.

Abkürzungsverzeichnis

ACID	Atomicity, Consistency, Isolation, Durability
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DSN	Data Source Name
DTD	Document Type Definition
EPN	EUREF Permanent Network
EUREF	Reference Frame Sub Commission for Europe
GFZ	Helmholtz-Zentrum Potsdam Deutsches GeoForschungsZentrum
GIS	Geoinformationssystem
GLONASS	Globalnaja Nawigazionnaja Sputnikowaja Sistema
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IGS	International GNSS Service
ITRF	International Reference Frame
JS	JavaScript
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
PDF	Portable Document Format
PHP	PHP:Hypertext PreProcessor
RINEX	Receiver Independent Exchange
SEMISYS	Sensor Meta Information System
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/ Internet Protocol
URL	Uniform Resource Locator
WWW	World Wide Web

Inhaltsverzeichnis

1 Einleitung

Die Beobachtung des Erdsystems und die Erforschung geodynamischer Prozesse mit Hilfe globaler Navigationssatellitensysteme (GNSS, engl. *Global Navigation Satellite System*) haben in den letzten Dekaden stark an Bedeutung gewonnen. Die Sektion „1.1 GPS/Galileo Erdbeobachtung“ des Deutschen GeoForschungsZentrums Potsdam (GFZ) beschäftigt sich mit der Nutzbarmachung satellitengestützter Verfahren für geowissenschaftliche Anwendungen. Die Grundlage dieser Anwendungen stellen Daten dar, die von global verteilten Beobachtungsstationen registriert und von Analysezentren ausgewertet werden. Für die Analyse und Auswertung dieser Daten müssen die Charakteristika der Stationsempfänger- und antennen sowie der Satelliten bekannt sein. Für die historische Erfassung und Verwaltung von Stationsmetadaten hat sich eine vom International GNSS Service (IGS) konzipierte ASCII-basierte Datei (als *IGS Sitelog* bezeichnet, IGSCB (2012d)) als De-facto-Standard im geowissenschaftlichen Umfeld etabliert. Die Verwaltung der Satellitenparameter erfolgt in einer vom GFZ geführten Textdatei.

Das grundlegende Ziel dieser Arbeit ist die persistente, datenformatunabhängige und validierte Speicherung aller Stations- und Satellitenmetainformationen an einer zentralen Stelle. Durch die zentrale Verwaltung soll sich insbesondere der Aufwand bei der Pflege und Verwaltung der IGS Sitelogs der vom GFZ verwalteten Stationen[1] minimieren. Darüber hinaus soll die Applikation die Grundlage für die Weiterentwicklung der Routine-Software des operationellen Datenzentrums am GFZ Potsdam darstellen, um für eigene Stationen automatisch spezifische RINEX-Header[2] (engl. *Receiver Independent Exchange*) der empfangenen Beobachtungsdaten zu generieren und die RINEX-Header externer Beobachtungsdaten gegenüber den in der Datenbank hinterlegten Metainformationen zu validieren. Dadurch wird verhindert, dass inkonsistente Beobachtungsdaten archiviert werden, die bei der Analyse der Daten zu Fehlern führen. Im Rahmen der vorliegenden Masterarbeit wurde ein lauffähiger, auf Client/Server-Technologie basierender Prototyp zur projektbezogenen Verwaltung von Stations- und Satellitenmetadaten für das GFZ Potsdam konzipiert und entwickelt. Darüber hinaus wurde ein internetbasiertes Geoinformationssystem (Internet-GIS) mit Grundfunktionalitäten für die Visualisierung und einfache Analyse der Beobachtungsstationen implementiert.

[1] Das GFZ Potsdam betreibt derzeit circa 30 eigene, global verteilte Beobachtungsstationen.

[2] Der Kopf einer RINEX-Datei (siehe Gourtner und Estey (2012)) enthält wichtige stationsspezifische Metainformationen (u.a. Stationsname, Receivertyp, Empfängertyp).

2 Grundlagen

2.1 Global Navigation Satellite Systems (GNSS)

Der Begriff *Global Navigation Satellite Systems* (GNSS) bezeichnet den Zusammenschluss verschiedener satellitengestützter Systeme für die Positionierung und Navigation auf der Erdoberfläche und in der Luft. Zu diesen zählen u.a. die sich im operationellen Betrieb befindlichen Systeme GPS (Global Positioning System) und GLONASS (Globalnaja Nawigazionnaja Sputnikowaja Sistema) sowie die sich im Aufbau befindlichen Systeme Galileo und Beidou/Compass.

2.1.1 Grundprinzip der Positionsbestimmung

Satellitennavigationssysteme wurden entwickelt, um die Position, Geschwindigkeit und Zeit eines Körpers oder Punktes im erdfesten System zu bestimmen. Alle mit GNSS bestimmten Positionen beruhen auf der Entfernungsmessung zwischen dem Satelliten und der Empfangseinheit auf der Erde (vgl. Abbildung 2.1).

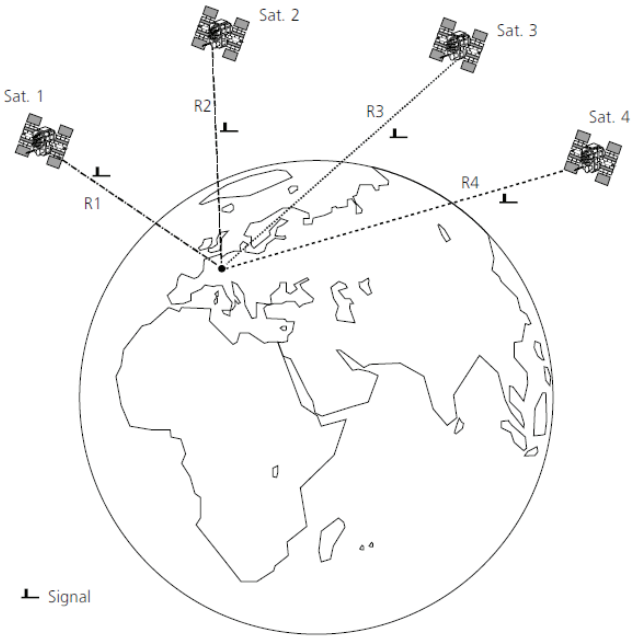

Abbildung 2.1: Positionsbestimmung mittels GNSS (Zogg, 2011, S. 17)

Die Distanz zu jedem der beobachteten Satelliten kann über die Laufzeit des Signals (skaliert mit der Lichtgeschwindigkeit $c = 299792458\frac{m}{s}$) ermittelt werden. Bei allen Satellitennavigationssystemen handelt es sich um Einweg-Entfernungsmessverfahren, d.h. das Sende- und Empfangszeit von unterschiedlichen und nicht synchronisierten Uhren bestimmt werden und somit einen bestimmten Offset aufweisen. Es werden daher nur Pseudoentfernungen (engl. *Pseudorange*) und die Zeit, zu der das Signal im Empfänger einläuft, gemessen. Für die Bestimmung einer dreidimensionalen Empfängerposition sowie des Empfängeruhrfehlers werden daher mindestens vier Satelliten zur mathematischen Lösung der vier unbekannten Größen benötigt. (vgl. Seeber (2003, S. 211), Zogg (2011, S. 11ff))

2.1.2 Aufbau globaler Navigationssatellitensysteme

Alle satellitengestützten Navigationssysteme können strukturell in drei Segmente gegliedert werden: dem Raumsegment, dem Kontrollsegment und dem Nutzersegment.

Raumsegment Das Ziel aller Systeme ist die kontinuierliche, globale Bestimmung von Positionen auf der Erde. Daher muss eine Satellitenkonstellation entwickelt und gewählt werden, bei der an jedem Ort und zu jedem Zeitpunkt (mindestens) vier Satelliten nutzbar sind.

Die Satelliten sind mit Atomuhren, Radio Transceivern[3], Computern und zusätzlichen Equipment (u.a. Sonnenkollektoren zur Stromversorgung, Antriebssystem zur Bahnkorrektur und Stabilität) ausgestattet, um die Satelliten zu betreiben (Hofmann-Wellenhof et al., 2008, S. 6f). Die Satelliten senden Signale auf definierten Frequenzen, die die aktuelle Position des Satelliten sowie die Zeit, zu der dieses Signal ausgesandt wurde, enthalten. Diese Signale werden vom Empfänger registriert und in eine Pseudoentfernung transformiert.

Kontrollsegment Das Kontrollsegment ist für die Steuerung des Gesamtsystems zuständig. Zu den Hauptaufgaben zählen (vgl. Zogg (2011, S. 51), Seeber (2003, S. 217)):

- Überwachung, Kontrolle und Wartung des Systems inklusive Satelliten,
- Berechnung und Übermittlung der Bahndaten (Ephemeriden und Almanach),
- Überwachung und Synchronisation der Satellitenuhren,
- Kommunikation mit den Satelliten (Übermittlung von Navigationsnachrichten, Einleitung von Manövern).

[3] Ein *Transceiver* ist eine zusammengesetzte technische Einrichtung, bestehend aus einem *Transmitter* (Sendeeinheit) und einem *Receiver* (Empfangseinheit).

Das Kontrollsegment besteht im Allgemeinen aus einer Hauptkontrollstation sowie mehreren Monitor- und Bodenkontrollstationen. Die Monitorstationen beobachten permanent alle sichtbaren Satelliten und senden die registrierten Daten an die Hauptkontrollstation, wo die Bahn- und Uhrparameter aus diesen Beobachtungen berechnet werden. Die Ergebnisse werden an die Bodenkontrollstationen übermittelt und von dort an die Satelliten übertragen. (vgl. Hofmann-Wellenhof et al. (1993, S. 18f), Seeber (2003, S. 217))

Nutzersegment Das Nutzersegment umfasst im Wesentlichen alle Empfangseinheiten (engl. *Receiver*), mit denen Satellitensignale empfangen und die enthaltenen Informationen zur Berechnung der eigenen Position genutzt werden können. Nach Hofmann-Wellenhof et al. (2008) kann das Nutzersegment in Nutzerkategorien, Receivertypen und verschiedene Informationsdienste klassifiziert werden.

Die Nutzerkategorien lassen sich sowohl in zivile und militärische als auch in autorisierte und nicht-autorisierte Nutzergruppen unterteilen.

Die Unterteilung in Receivertypen ergibt sich aus der Vielfalt der auf dem kommerziellen Markt angebotenen Geräte[4], die unterschiedliche Eigenschaften aufweisen können. Die Receiver lassen sich u.a. nach den empfangenen GNSS-Typen (z.B. GPS, GLONASS, Galileo), nach der Anzahl der Kanäle sowie nach den empfangenen Beobachtungstypen (z.B. L1, L2, P1, P2) charakterisieren.

Für die Bereitstellung von GNSS Status-Informationen und GNSS-Daten für die Nutzer haben sich verschiedene behördliche Institutionen und private Unternehmen etabliert. Sie veröffentlichen u.a. Bahndaten (Ephemeriden, Almanach), Erdrotationsparameter, atmosphärische Parameter sowie Beobachtungsdaten. Als Beispiel sei an dieser Stelle der International GNSS Service (IGS) zu nennen, der im Nachfolgenden näher vorgestellt werden soll.

2.1.3 International GNSS Service (IGS)

Der International GNSS Service (IGS) ist ein freiwilliger Zusammenschluss von mehr als 200 weltweit operierenden Organisationen. Der IGS bündelt die Ressourcen der mitwirkenden Organisationen und kann dadurch ein globales Netzwerk an GNSS-Stationen erschließen, das aktuell[5] aus 440 Stationen besteht. Die GNSS-Stationen zeichnen permanent Rohdaten auf und senden diese an die zuständigen operationellen Datenzentren. Da die Rohdaten binär und empfängerspezifisch sind, müssen diese in das ASCII-basierte,

[4] Der International GNSS Service (IGS) führt derzeit 282 unterschiedliche, im geodätischen und geowissenschaftlichen Umfeld genutzte Receivertypen (IGSCB, 2012c, Stand: 07.05.2012).

[5] Stand: 11.08.2012

empfängerunabhängige Datenformat RINEX konvertiert und auf Validität geprüft werden. Anschließend werden die formatierten Daten an regionale oder globale Datenzentren übertragen. Auf Grundlage der gewonnenen Daten können präzise GNSS-Produkte erzeugt und bereitgestellt werden, die für wissenschaftliche und technische Anwendung genutzt werden. Dazu zählen:

- GPS und GLONASS Ephemeriden,
- Erdrotationsparameter,
- GPS Satelliten- und Stationsuhreninformationen,
- Koordinaten der Beobachtungsstationen sowie
- ionosphärische und troposphärische Parameter.

Die Aktivitäten der GNSS-Stationen, Datenzentren und Analysezentren werden durch das Zentralbüro (engl. *Central Bureau*, CB) koordiniert. IGS CB veröffentlicht Standards und Richtlinien, an die sich die Beteiligten richten müssen. Darüber hinaus ist das Zentralbüro für die Verwaltung der IGS Sitelogs sowie für die Aktualisierung weiterer Metadaten verantwortlich. Die qualitative Sicherung der Analyseergebnisse und veröffentlichten Produkte ist ebenso Bestandteil der Aufgaben des IGS CB.

Das GFZ Potsdam, als mitwirkende Organisation, unterstützt den IGS mit 24[6] weltweit verteilten Stationen sowie einem operationellen Daten- und Analysezentrum. Das GFZ beteiligt sich an der Generierung und Instandhaltung des International Reference Frame (ITRF) sowie an der Bereitstellung hochgenauer Satellitenorbits, Uhrenparameter und Erdrotationsparameter. Für die Analyse der Beobachtungs- und Navigationsdaten und der damit verbundenen Generierung von GNSS-Produkten müssen die Eigenschaften der Stationsempfänger und -antennen bekannt sein. Hierfür stellt der IGS für jede Station eine ASCII-basierte Datei (IGS Sitelog) bereit, die für die historische Erfassung der Stationsmetadaten genutzt wird. Darüber hinaus werden textbasierte Dateien mit Informationen über Receiver- und Antennentypen zur Verfügung gestellt.

Die dateibasierte Verwaltung von Metadaten ist für eine kleine Anzahl an Beobachtungsstationen unproblematisch. Mit zunehmender Stationsanzahl und der damit verbundenen Menge an Daten und komplexen Zusammenhänge hat sich der Verwaltungsaufwand erhöht. Darüber hinaus gibt es keine Möglichkeit der Bearbeitung nutzerspezifischer Anfragen. Ein datenbankbasierter Ansatz ist somit unumgänglich.

[6] Stand: 13.07.2012

2.2 Datenbanksysteme

Ein Datenbanksystem (DBS) ist ein System, das der elektronischen Speicherung und Verwaltung von Daten dient und aus zwei Komponenten besteht: dem Datenbankmanagementsystem (DBMS) und der eigentlichen Datenbank. Die Datenbank ist eine strukturierte Sammlung einheitlich beschriebener, persistent gespeicherter Daten (Brinkhoff, 2008). Das Datenbankmanagementsystem ist die Software, die als Schnittstelle zwischen der Datenbank und den Applikationen fungiert und einen einheitlichen und effizienten Zugriff auf die Daten ermöglicht, ohne dass deren interne Struktur bekannt ist. In Abbildung 2.2 ist der Zugriff eines Anwenderprogramms über die Programmschnittstellen des DBMS auf die Datenbank graphisch dargestellt.

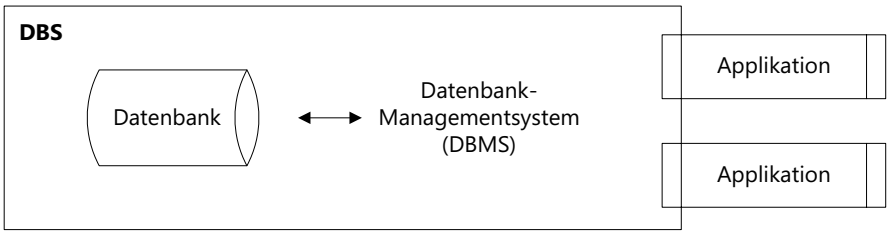

Abbildung 2.2: Aufbau eines Datenbanksystems
(angelehnt an Bartelme (2005) und Brinkhoff (2008))

2.2.1 Eigenschaften eines Datenbanksystems

Die Motivation zum Einsatz von Datenbanksystemen ergibt sich aus den Unzulänglichkeiten beim Nutzen von Dateisystemen (Bradke, 2009, S. 6). Dateisysteme bieten als Strukturierungsmechanismus Dateien und hierarchische Verzeichnisse an. Sie bieten aber keine Funktionalitäten hinsichtlich einer Integritätsprüfung, eines Mehrbenutzerbetriebs oder einer Anfragebearbeitung. Diese Schwächen werden bei der datenbankbasierten Verwaltung von Daten aufgelöst. Nachfolgend sollen die Eigenschaften eines Datenbanksystems beschrieben werden (angelehnt an Codd (1970)).

Datenunabhängigkeit Die Datenmodellierung und -abfrage erfolgt unabhängig von der physischen Organisation der Daten, d.h. dass die enge Verknüpfung zwischen den Daten und den Applikationen aufgelöst wird und sich Änderungen bei der Datenspeicherung nicht auf das Anwenderprogramm auswirken (vgl. Brinkhoff (2008)).

Zentrale Datenhaltung, Mehrbenutzerbetrieb und Datensicherheit Die Daten werden zentral auf einem Server gehalten und stehen den Anwendern somit in gleicher Form zur Verfügung. Das Datenbanksystem muss daher einen Mehrbenutzerbetrieb sicherstellen, sodass mehrere Anwender gleichzeitig sowohl lesend als auch schreibend auf die

Datensätze in der Datenbank zugreifen können (Brinkhoff, 2008). Dies hat zur Folge, dass das DBMS ein Transaktionskonzept (ACID-Prinzip, siehe Tabelle 2.1) unterstützen muss, um die Konsistenz des Datenbestandes zu gewährleisten.

Eigenschaften	Bedeutung	Beschreibung
Atomicity	Atomarität	Eine Transaktion wird entweder als Ganzes oder gar nicht ausgeführt.
Consistency	Konsistenz	Die Datenbank ist vor und nach jeder Transaktion in einem konsistenten Zustand. Alle Datenmanipulationen müssen den Integritätsbedingungen des Datenmodells entsprechen.
Isolation	Isolation	Jede Transaktion läuft isoliert, d.h. unabhängig von anderen Transaktionen.
Durability	Dauerhaftigkeit	Alle transaktionalen Änderungen werden persistent in der Datenbank gespeichert.

Tabelle 2.1: Das Transaktionskonzept nach dem ACID-Prinzip (vgl. Kemper und Eickler (2011, S. 289))

Datenintegrität Sämtliche Daten sollten nur einmal gespeichert werden, um Redundanzen und somit die Gefahr von Dateninkonsistenzen zu vermeiden.

Operationen und Performance Ein Datenbanksystem stellt im Gegensatz zur dateibasierten Verwaltung eine strukturierte Datenbanksprache zur Verfügung, mit der der Datenbestand abgefragt, beschrieben oder manipuliert werden kann. Als Beispiel ist an dieser Stelle SQL (engl. *Structured Query Language*) zu nennen.

Bei Dateisystemen steigt die Zugriffszeit auf die Daten im Allgemeinen proportional mit dem Datenvolumen. Datenbanksysteme verwenden hingegen Indexverfahren, um einen performanten Zugriff auf große Datenvolumina und große Mengen von Datensätzen zu gewährleisten.

Zugriffskontrolle und Datenschutz Datenbanksysteme verfügen über eine integrierte Benutzerverwaltung, die den Zugriff auf die Datenbank und somit die Rechte jedes Nutzers steuert. Dadurch kann der Datenbestand vor nicht-autorisierten Zugriffen geschützt werden.

2.2.2 Datenbankmodelle

Datenbanksysteme können grundsätzlich auf verschiedenen Modellen basieren, die die Struktur einer Datenbank auf Grundlage von Regeln und Konzepten beschreiben und

somit festlegen, wie die Daten gespeichert und bearbeitet werden können. Nachfolgend sollen kurz die gängigen, in der Praxis verwendeten Datenbankmodelle vorgestellt werden. Eine längere Einführung in diese Thematik wurde bereits in Bradke (2009, S. S. 8ff) gegeben. Ausführliche Erläuterungen finden sich unter anderem in Brinkhoff (2008) und de Lange (2002).

2.2.2.1 Relationales Modell

Das relationale Datenmodell basiert auf den theoretischen Grundsätzen von Codd (1970) und hat sich seit Mitte der 1980er Jahre als Standard kommerzieller Datenbanksysteme etabliert. Das Modell beschreibt die Datenbank als eine Sammlung von Tabellen (Relationen) sowie deren Beziehungen zueinander. Die Datensätze werden eindeutig über die Verwendung von Primärschlüsseln identifiziert. Die Beziehungen zwischen den Relationen werden über Fremdschlüssel hergestellt. Darüber hinaus müssen relationale DBMS (RDBMS) über mengen- und relationenorientierte Operatoren verfügen, mit deren Hilfe sich Tabellen kombinieren bzw. Datenbestände hinsichtlich vielfältiger Kriterien selektieren lassen.

2.2.2.2 Objektorientiertes Modell

Das objektorientierte Modell nutzt Objekte sowie deren Eigenschaften und Methoden zur Beschreibung der Struktur der Datenbank. Das Datenbanksystem muss sowohl den in Abschnitt 2.2.1 beschriebenen Datenbankeigenschaften als auch den nachfolgenden Prinzipien des Objektmodells entsprechen (vgl. Atkinson et al. (1989, S. 2ff):

- die Zusammensetzung einfacher Elemente zu komplexen Objekten,
- die eindeutige und systemweite Identifizierung von Objekten,
- die Kapselung von Objekten,
- die hierarchische Organisation von Objekten gleicher Struktur und äquivalentem Verhalten in Form von Klassen,
- die Vererbung von Objekteigenschaften und -methoden,
- der Polymorphismus sowie
- die Vollständigkeit und Erweiterbarkeit.

Das objektorientierte Modell findet v.a. in komplexen Systemen seine Anwendung, um stark strukturierte Objekte zu beschreiben. Auf dem kommerziellen Markt konnten sich objektorientierte DBMS (OODBMS) bisher aber nicht durchsetzen, was nicht zuletzt an der Dominanz und weiten Verbreitung des relationalen Ansatzes liegt (vgl. Brinkhoff

(2008)). Ein weiterer Nachteil zeigt sich vor allem bei der Verarbeitung großer Datenmengen und der damit verbundenen schlechteren Performance gegenüber relationalen Datenbanksystemen.

2.2.2.3 Objektrelationales Modell

Die Alternative zum Einsatz des objektorientierten Modells stellen objektrelationale DBMS (ORDBMS) dar, die das etablierte relationale Modell um die zuvor beschriebenen objektorientierten Konzepte erweitern. Die Speicherung der Objekte erfolgt nach dem relationalen Grundsatz tabellenbasiert. Das DBMS muss hierfür strukturierte Datentypen und entsprechende Speicherungskonzepte zur Verfügung stellen.

Alle großen Datenbankhersteller bieten mittlerweile ORDBMS an. Auf kommerzieller Seite sind die DBMS von Oracle und IBM zu nennen. Im Open-Source-Bereich sind MySQL und PostgreSQL führend. Nachfolgend soll das ORDBMS PostgreSQL vorgestellt werden, da dieses die Grundlage des zu entwickelnden Systems darstellen soll.

2.2.3 PostgreSQL

PostgreSQL bezeichnet ein auf Postgres 4.2[7] basierendes, quelloffenes, objektrelationales Datenbankmanagementsystem. Postgres wurde an der University of California at Berkley unter Leitung von Prof. Michael Stonebraker von 1986 bis 1993 entwickelt. Seit 1996 trägt PostgreSQL den aktuellen Namen und steht unter der anerkannten PostgreSQL Licence (Initiative (2012)), die den Einsatz im kommerziellen Bereich erlaubt.

2.2.3.1 Kommunikation

PostgreSQL wurde auf Grundlage eines Client-Server-Modells entwickelt. Ein Serverprozess ist für die Verwaltung der Datenbankdateien, für den Verbindungsaufbau zwischen Client und Server sowie für die Bearbeitung der Client-Anfragen zuständig. Das Datenbank-Serverprogramm wird als postgres bezeichnet.

Für die Kommunikation mit postgres können unterschiedliche Clients verwendet werden, die die Verbindung zum Datenbankserver aufbauen und Datenbankoperationen durchführen. In der PostgreSQL-Distribution sind u.a. ein terminal-basierter (psql) sowie ein graphischer Client (pgAccess) enthalten.

Die verteilte Architektur hat den Vorteil, dass Client und Server nicht auf demselben Host installiert sein müssen. Die Kommunikation erfolgt über TCP/IP[8]. Durch diese Architektur muss der PostgreSQL Server in der Lage sein, mehrere konkurrierende

[7] Der Quellcode der Software ist verfügbar unter: http://db.cs.berkeley.edu/postgres.html
[8] TCP/IP ist ein Netzwerkprotokoll und steht für Transmission Control Protocol/Internet Protocol

Datenbankverbindungen zu verwalten. Das Serverprogramm POSTGRES startet in diesem Fall einen neuen Prozess, der die Client-Server-Kommunikation übernimmt. (vgl. PostgreSQL (2012a), Scherbaum (2012))

2.2.3.2 Funktionsumfang

Ein Vorteil von PostgreSQL ist der große und ständig wachsende Funktionsumfang. Es unterstützt den Großteil des SQL-Standards (ISO/IEC 9075:2008) und bietet moderne Funktionalitäten wie komplexe Abfragen, Trigger, Sichten (engl. *Views*) und transaktionale Integritätsprüfungen an (PostgreSQL, 2012b). Darüber hinaus kann PostgreSQL auf Grundlage der Objektorientierung um weitere Eigenschaften erweitert werden. Dazu zählen u.a. benutzerdefinierte Datentypen, benutzerdefinierte Funktionen, Operatoren, Indexierungsmethoden und prozedurale Sprachen. Ein Beispiel für den Einsatz dieser objektorientierten Konzepte ist das Zusatzmodul (engl. *Add-on*) PostGIS, mit dem PostgreSQL um räumliche Datentypen, räumliche Funktionen und räumliche Indexierungsmechanismen erweitert werden kann.

2.2.4 Datensicherung

Die Sicherung der in einer Datenbank abgelegten Informationen ist ein wichtiger Aspekt, um Verlusten insbesonderer sensibler und täglich benötigter Daten (z.B. für die Analyse und Prozessierung von Beobachtungsdaten) entgegenzuwirken. Hierfür ist eine sinnvolle Strategie zu entwickeln, die auf der einen Seite größtmögliche Sicherheit[9] und auf der anderen Seite eine optimale Ausnutzung des Speicherplatzes bietet.

Im Allgemeinen wird bei der Sicherung einer Datenbank zwischen Hot- und Cold-Backup unterschieden. Das Cold-Backup stellt die sicherste Variante dar, da die Datenbank im heruntergefahrenen Zustand komplett gesichert wird. Dieses Verfahren wird aus Gründen der Verfügbarkeit aber zumeist nicht verwendet. Die Alternative stellt das Hot-Backup dar, bei der die Datenbank während des laufenden Betriebs gesichert wird. Der Nachteil dieser Sicherungsart besteht in der Gefahr, dass Inkonsistenzen im Datenbestand auftreten können. Das Datensicherungskonzept ist von verschiedenen Einflussfaktoren abhängig (vgl. BSI (2012b) und BSI (2012a)):

Verfügbarkeit Das zu wählende Sicherungsverfahren ist abhängig von den Anforderungen an die Verfügbarkeit. Bei dauerhafter Verfügbarkeit (24/7) kommt in der Regel nur ein Hot-Backup in Frage.

[9] Das impliziert sowohl den größtmöglichen Prozentsatz an sichergestellten Daten als auch die Vermeidung von inkonsistenten Datenbeständen.

Datenvolumen Bei der Art der Sicherung muss das gesamte Datenvolumen der Datenbank berücksichtigt werden. Es wird grundsätzlich zwischen komplettem, differentiellem und inkrementellem Backup unterschieden (siehe Tabelle 2.2).

Typen	Beschreibung
Vollständig	Alle Daten werden auf einem unabhängigen Speichermedium gesichert.
Differentiell	Es werden nur die Änderungen seit der letzten vollständigen Sicherung gespeichert.
Inkrementell	Es werden nur die Änderungen seit der letzten Sicherung gespeichert.

Tabelle 2.2: Typen der Datensicherung (vgl. BSI (2012a))

Bei größer dimensionierten Datenvolumen empfehlen sich aus Zeit- und Kapazitätsgründen Teilsicherungen der Datenbank, bei der nur die Änderungen gesichert werden. Dieses Vorgehen hat allerdings den Nachteil, dass bei der Restaurierung der Daten ein höherer Aufwand und Zeitbedarf besteht als bei der Volldatensicherung, da die wiederherzustellenden Daten aus mehreren Sicherungen extrahiert werden müssen.

Maximal verkraftbarer Datenverlust Die Anzahl und das Zeitintervall durchzuführender Sicherungen ist abhängig von den Anforderungen an die Relevanz, der Verfügbarkeit und der Integrität der Daten. Je höher diese Faktoren einzustufen sind, desto häufiger sollten Sicherungen durchgeführt werden.

Wiederanlaufzeit Die maximal zulässige Wiederanlaufzeit der Datenbank nach einem Absturz muss spezifiziert werden, um die Anforderungen an die Verfügbarkeit zu gewährleisten.

2.3 Internet-GIS

Geographische Informationssysteme (engl. *Geographic Information Systems*, GIS), die auf Webtechnologien[10] basieren, stellen aufgrund ihrer flexiblen Einsatzmöglichkeiten eine gute Alternative zum bewährten Desktop-GIS dar (vgl. Peng und Tsou (2003, S. 1f)). Als GIS werden rechnergestützte Systeme bezeichnet, die aus den Komponenten Hardware, Software, Daten und Anwendungen bestehen und der Erfassung und Verarbeitung, Verwaltung und Organisation, Modellierung und Analyse sowie der alpahanumerischen und graphischen Präsentation raumbezogener Daten dienen (Bill und Fritsch, 1994, S. 40).

2.3.1 Geodaten

Die Qualität der resultierenden Analyse- und Visualisierungsergebnisse hängt primär von der Güte des zugrunde liegenden Datenmaterials ab. Ohne entsprechende Daten wären Analysen sowie die Darstellung der Analyseergebnisse nicht möglich. Unter Geodaten werden Daten mit einem räumlichen Bezug zu Objekten der realen Welt verstanden. Sie können sowohl direkt (Position im Raum über Koordinaten) als auch indirekt (durch Relationen) referenziert und aus verschiedenen Datenarten gebildet werden. Dazu zählen sowohl raumbezogene und attributbezogene Daten, die miteinander in Beziehung gesetzt werden, als auch graphische Daten (z.B. Hintergrunddaten, Symbole, Texte etc.), die der besseren Orientierung dienen.

2.3.2 Technologie

Das Ziel internetbasierter GIS-Anwendungen ist die weltweite, schnelle und kostengünstige (in vielen Fällen kostenlose) Bereitstellung von Geodaten für eine breite Öffentlichkeit. Dementsprechend wird eine technologische Architektur benötigt, die diesen Zielstellungen gerecht wird. Dabei wird zumeist auf das Modell der n-tier Client-Server-Architektur zurückgegriffen (vgl. Peng und Tsou (2003, S. 20)), die im Nachfolgenden näher erläutert werden soll.

2.3.2.1 Komponenten

Im Allgemeinen besteht ein Internet-GIS aus vier Hauptkomponenten: dem Client, dem Webserver, dem Kartenserver und dem Datenserver (siehe Abbildung 2.3). Die Interaktion des Nutzers mit der Applikation und dem zugrunde liegenden Datenmaterial erfolgt einzig und allein mit Hilfe eines Webbrowsers (Client), der die Datenanfragen des Nutzers an den Webserver[11] sendet und die Antwort graphisch präsentiert.

[10] Dazu zählen neben dem Internet auch internetbasierende Systeme (z.B. Intranet, Extranet).

[11] Der Webserver wird aufgrund des verwendeten Protokolls meist auch als HTTP-Server (Hypertext Transfer Protocol) bezeichnet.

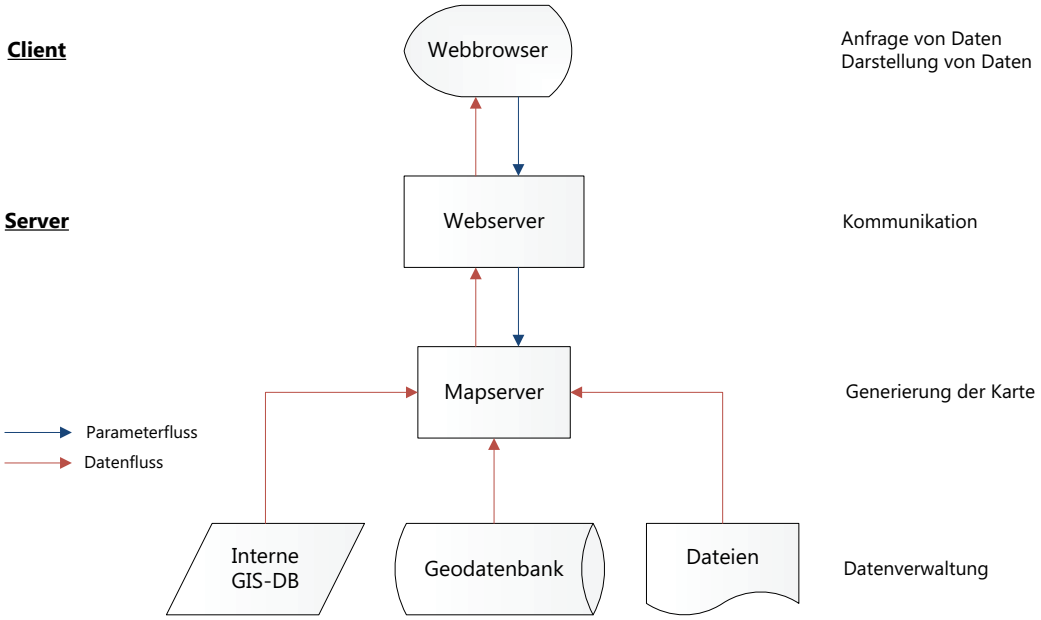

Abbildung 2.3: Komponenten eines Internet-GIS
(angelehnt an Aden et al. (2007, S. 7) und Peng und Tsou (2003, S. 20))

Der Webserver kann auf die gestellte Anfrage unterschiedlich reagieren (vgl. Peng und Tsou (2003, S. 22)):

1. Durch Senden vorgefertigter HTML-Dokumente,
2. Durch Senden von Java Applets oder ActiveX Kontrollelementen,
3. Durch Weiterleitung an andere Programme (z.B. Common Gateway Interface, CGI), die die Anfragen bearbeiten.

Im letzteren Fall werden Dienste von entsprechenden Applikationsservern angefordert, die als Middleware zwischen Webserver und serverseitigen Applikationen (z.B. Mapserver) fungieren.

Der Kartenserver (engl. *Mapserver*) ist für die Generierung der Karten durch die räumliche Abfrage des Datenbestandes zuständig. Die Geodaten werden auf einem Datenserver gehalten, der über verschiedene Datenbankschnittstellen[12] auf Grundlage einer definierten Abfragesprache (z.B. SQL) abgefragt werden kann.

[12] z.B. ODBC (Open Database Connectivity) oder JDBC (Java Database Connectivity)

2.3.2.2 Anwendungen

Je nach Funktionalität lässt sich grundsätzlich zwischen drei Implementierungsarten differenzieren (vgl. Bill (2003)):

- Statisches Internet-GIS,
- Client-dynamisches Internet-GIS,
- Server-dynamisches Internet-GIS.

Das statische Internet-GIS stellt die einfachste technische Variante und Umsetzungsform dar, da lediglich vorgefertigte statische HTML-Seiten mit statischen Rasterkarten (engl. *Static Maps*) verwendet werden, sodass auf einen Kartenserver verzichtet werden kann. Die Karten sind mit sensitiven Flächen (engl. *Area Shape*) versehen, die bei Anklicken auf andere HTML-Seiten verweisen. Diese Form bietet dem Nutzer wenige Möglichkeiten der Interaktion.

Interaktionen werden u.a. durch client-dynamische Anwendungen ermöglicht. Dabei werden Erweiterungen auf Seite des Clients (z.B. Java-Applets, JavaScripts, AJAX) für die Nutzerinteraktion genutzt, ohne das permanent auf den Server zugegriffen werden muss. Eine vollkommene Interaktivität kann dadurch aber nicht gewährleistet werden.

Hierfür werden serverseitige Skripte (z.B. PHP, ASP) benötigt, die die Seiten dynamisch erstellen. Dabei werden den Skripten entsprechende Parameter vom Client übergeben, die für die Abfrage der Datenbank benötigt werden. Aus den zurückgegebenen Daten können die Karten auf dem Server generiert und auf dem Client visualisiert werden.

2.3.2.3 Aufwand und Kosten

Der zeitliche Aufwand und die Kosten bei der Einrichtung, Administration und Wartung eines Kartenservers können sehr hoch ausfallen (vgl. Bill (2003)). Daher sollten im vorab alle Funktionalitäten des Systems definiert werden. Die Anzahl und der Anteil an Analysefunktionalitäten gibt letzten Endes Auskunft über die Notwendigkeit für den Einsatz eines Kartenservers.

Je höher der Anteil an Analysefunktionen ist, desto höher sind die Anforderungen an die einzusetzende Technologie. Bei Applikationen, die lediglich Geodaten visualisieren und nur eingeschränkt analysieren sollen, kann in der Regel auf die Installation eines Kartenservers verzichtet werden. Die Alternative stellen Online-Kartendienste dar, auf die im Nachfolgenden näher eingegangen werden soll.

2.3.3 Online-Kartenanwendungen und Programmierschnittstellen

Seit dem Start des Internet-Kartendienstes Google Maps im Jahre 2005 hat sich eine Vielzahl weiterer Dienste[13] am Markt etabliert. Die Dienste stellen neben der interaktiven Karte typische GIS-Funktionalitäten[14] bereit.

Die Interaktion des Nutzers mit der Applikation basiert auf der AJAX-Technologie (engl. *Asynchronous JavaScript and XML*), d.h. dass die durch den Nutzer ausgelösten Ereignisse clientseitig abgefangen und die Anfragen asynchron zum Server übertragen werden, ohne dass die Internetseite neu geladen werden muss.

Die Dienste stellen eine JavaScript-basierte Programmierschnittstelle (engl. *Application Programming Interface*, API) bereit, die es erlaubt die Internet-GIS-Funktionalitäten in einer eigenen Anwendung zu implementieren. Durch die Nutzung der objektorientierten Bibliotheken lassen sich die Applikationen individuell anpassen. Die Entwicklung erfolgt clientseitig und unabhängig vom zugrunde liegenden Server. Die Installation eines Mapservers ist nicht erforderlich.

[13] An dieser Stelle sind exemplarisch die Kartendienste Microsoft Bing Maps, OpenLayers, Open Street Map und Yahoo Maps zu nennen.

[14] u.a. Zoom, Panning, Sachdatenabfragen über Kartenelemente, Maßstabsanzeige

3 Analyse

Die Phase der Analyse dient dazu, die Anforderungen an das zu entwickelnde System zu spezifizieren und zu beschreiben. Die Analyseergebnisse bilden die Grundlage für die weiteren Arbeiten und haben entscheidenden Einfluss auf die Qualität und Produktivität der Softwarelösung. Die Systemanforderungen sollten daher so genau wie möglich definiert werden, um ein realitätsnahes Modell zu entwickeln.

3.1 Datengrundlage

Das Ziel dieser Arbeit ist die Entwicklung eines Systems zur Verwaltung von GNSS-Sensormetainformationen auf Basis einer zentralen Datenbank. Zu den Beobachtungsstationen werden Metadaten entsprechend der Sitelog-Deklaration des IGS (IGSCB, 2012d) abgelegt. Die Satellitenparameter werden derzeit in einer GFZ-eigenen Tabelle gepflegt.

3.1.1 IGS Site Information Form (Sitelog)

Das IGS Sitelog umfasst 14 informelle Sektionen (siehe Tabelle 3.1), in denen die wichtigsten Metadaten zu den Beobachtungsstationen zusammengefasst sind. Die ASCII-Datei ist in Form von Schlüssel-Wert-Paaren, die jeweils durch einen Doppelpunkt voneinander getrennt sind, aufgebaut. Für die einzelnen Schlüssel sind jeweils unterschiedliche Datentypen (vgl. IGSCB (2012d), IGSCB (2012b)) definiert.

Auf Basis dieser Informationen und der zulässigen Datentypen kann ein präzises und realitätsnahes Datenmodell entwickelt werden.

3.1.2 IGS Receiver/Antennen-Tabelle

Für die Beschreibung der auf der jeweiligen Beobachtungsstation installierten Hardware (Receiver, Antenne, Radom) wurden seitens des IGS standardisierte Gerätenamen eingeführt, die in einer ASCII-basierten Datei (*rcvr_ ant.tab*, IGSCB (2012c)) gepflegt werden und bestimmten Konventionen unterliegen (vgl. IGSCB (2012c)). Listing 3.1 zeigt einen Ausschnitt aus der IGS Receiver/Antennen-Tabelle, in der die standardisierten Gerätenamen mit ihren Eigenschaften aufgeführt sind.

#	Name	Beschreibung
0	Form	Informationen zum IGS Sitelog
1	Site Identification	Allgemeine Informationen und Eigenschaften der GNSS-Station
2	Site Location Information	Informationen zum Standort der Station
3	GNSS Receiver Information	Historische Informationen zu den bisher und aktuell installierten GNSS-Empfängern
4	GNSS Antenna Information	Historische Informationen zu den bisher und aktuell installierten GNSS-Antennen
5	Surveyed Local Ties	Informationen zu lokalen Messungen
6	Frequency Standard	Uhreninformationen
7	Collocation Information	Informationen über permanente oder mobile geodätische Messaufstellungen
8	Meteorological Information	Informationen über installierte meteorologische Instrumente (Temperatur-, Luftfeuchte-, Luftdruck-, Wasserdampfsensor)
9	Local Ongoing Conditions	Informationen über andauernde, das Satellitensignal beeinflussende Faktoren
10	Local Episodic Effects	Informationen über episodisch, das Satellitensignal beeinflussende Faktoren
11	On-Site Agency	Kontaktinformationen über die vor Ort betreuende Organisation
12	Responsible Agency	Kontaktinformationen über die verantwortliche Organisation
13	More Information	Zusätzliche Informationen (u.a. Antennengraphiken)

Tabelle 3.1: Aufbau und Inhalt eines IGS Sitelogs (IGSCB, 2012d)

```
+-------------------------+-----------------------------------------+
| JAVAD Antennae          |                                         |
| IGS Codes-15 columns    |              Description                |
| XXXXXXXXXXXXXXX DOME    |                                         |
+-------------------------+-----------------------------------------+
| JAV_RINGANT_G3T         | GPS L1/L2/L5, GLO L1/L2, GAL E1/E5A     |
+-------------------------+-----------------------------------------+
```

Listing 3.1: Ausschnitt aus der IGS Receiver/Antennen-Tabelle (IGSCB, 2012c)

3.1.3 IGS Antennengraphik-Datei

Die IGS Antennengraphik-Datei (*antenna.gra*) enthält einen Großteil der in der *rcvr_ ant.tab* definierten Antennentypen in Form textbasierter Graphiken (siehe Listing 3.2).

```
JAV_RINGANT_G3T

                  ----------                    <--   0.0614
                 /          \
                /            \
+-----------------------------------------------+ <--   0.0410   TCR
|                                               |
|          +--------------------+               |
|          |    \          /    |               |
|          |     \        /     |               |
|          |      +--x--+       |               | <--   0.0000   BAM=ARP
+----------+--------------------+---------------+ <--  -0.0260   BCR

<--                0.3260                  -->
```

Listing 3.2: Ausschnitt aus der IGS Antennengraphik-Datei (IGSCB, 2012a)

Die Graphiken definieren die Position des Antennenreferenzpunktes (engl. *Antenna Reference Point*, ARP) und beinhalten wichtige physikalische Antennenabmessungen. Die Antennengraphik ist Bestandteil des IGS Sitelogs und muss daher auch im Datenmodell berücksichtigt werden.

3.1.4 GFZ Satellitenparameter-Tabelle

Die Verwaltung der für die Prozessierung der Beobachtungsdaten notwendigen Satellitenparameter erfolgt derzeit in einer textbasierten Datei, die von autorisierten Personen des GFZs gepflegt wird. Die Datei enthält Informationen über die Laufzeit, Masse, Frequenz und Uhren sowie Codes zur Identifikation der Satelliten. Dabei wird zwischen GPS, GLONASS und Galileo Satelliten differenziert.

3.2 Anforderungen

In Kapitel 1 wurden bereits erste Probleme und Zielstellungen dargestellt. Anhand dieser und anhand der erfassten Daten aus Abschnitt 3.1 können die funktionalen und nichtfunktionalen sowie technischen Anforderungen an das zu entwickelnde System spezifiziert werden.

3.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die geforderten Funktionalitäten der Applikation sowie die enthaltenen Daten und Schnittstellen.

Sensor-Metadatenbank Die wichtigste Komponente des **Se**nsor **M**eta **I**nformations**sys**tems (SEMISYS) stellt die zentrale Datenbank dar. Sie dient der Speicherung aller in Abschnitt 3.1 beschriebenen Sensor-Metadaten sowie zur Verwaltung bestimmter Systemkomponenten, die sich aus den weiteren Anforderungen an das System ergeben.

Nutzer- und Sitzungsverwaltung Die Authentifizierung von Nutzern ist ein zentraler und sicherheitsrelevanter Aspekt des gesamten Systems. Lediglich autorisierte Personen dürfen Zugriff auf die Applikation und deren Daten nach erfolgreicher Anmeldung haben. Die Anmeldung kann über den GFZ-internen LDAP-Server[15] oder manuell über den Abgleich der in der Datenbank hinterlegten Anmeldeinformationen erfolgen. Die Nutzer sollen über alle systeminternen Seiten eindeutig mit Hilfe von Sitzungen identifizierbar sein.

Rollen und Rechte Die einzelnen Nutzer nehmen unterschiedliche Rollen im System an. Es sollen grundsätzlich drei Rollen unterschieden werden:

- Standardnutzer: Nur Leserechte auf ausgewählte Bereiche.
- Gruppenmitglied: Lese- und Schreibrechte auf ausgewählte Bereiche, die aber abhängig von der Projektzugehörigkeit der Nutzer sind.
- Administrator: Lese- und Schreibrechte auf alle Bereiche und Daten des Systems. Er ist darüber hinaus zuständig für die Konfiguration der Betriebsparameter des Systems, für die Sicherung der Datenbank sowie für die Nutzerverwaltung und Verteilung projektbezogener Rechte.

Die Rollenzuordnung ermöglicht den Zugriff auf einzelne Menupunkte der Anwendung. Dabei wird lediglich das Recht auf die Sichtbarkeit einer Seite gegeben, die Berechtigung

[15] LDAP (engl. *Lightweight Directory Access Protocol*) ermöglicht den Zugriff auf Nutzerinformationen innerhalb eines Verzeichnisdienstes. SEMISYS verwendet LDAP für die Authentifizierung der Nutzer.

für den Zugriff auf die Daten (Lese- und Schreibrechte) ergibt sich aus der Zugehörigkeit zu definierten Projekten.

Projektverwaltung Die Berechtigungen zur Manipulation von Datenbankeinträgen werden anhand der Zuordnungen der Nutzer zu einzelnen Projekten getroffen. Jede Projektgruppe soll selbstständig für die Verwaltung der Stations- und Satellitenmetadaten verantwortlich sein.

Protokollierung von Änderungen Alle Änderungen am Datenbestand müssen protokolliert werden. Darüber hinaus soll das Datenmodell das Rückgängigmachen von Änderungen zulassen. Die Implementierung letzterer Funktion erfolgt in einer späteren Phase und ist nicht Bestandteil dieser Arbeit.

Sperren von Datensätzen Bei der zu entwickelnden Applikation handelt es sich um ein typisches Mehrbenutzersystem, d.h. dass grundsätzlich die Möglichkeit besteht, dass mehrere Nutzer gleichzeitig ein und denselben Datenbestand bearbeiten. Dies führt in der Regel zu Problemen in der Integrität der Daten. Aufgrund dessen sollen in Bearbeitung stehende Datensätze für einen bestimmten Zeitraum für andere Nutzer explizit gesperrt werden. Ein Nutzer soll immer nur einen Datensatz gleichzeitig bearbeiten dürfen, sodass unnötig gesetzte Sperren entfallen.

Protokollierung skript- und datenbankseitiger Fehler Keine Applikation ist frei von Fehlern. Daher ist es wichtig skript- und datenbankseitige Fehler zu protokollieren, um diese beheben zu können. Die Protokollierung erfolgt sowohl datenbankseitig als auch dateibasiert. Die Administratoren sollen darüber hinaus via E-Mail über schwerwiegende Fehler informiert werden.

Import von Dateien Alle Dateien aus Abschnitt 3.1, die Metadaten für die Beobachtungsstationen enthalten (IGS Sitelogs, IGS rcvr_ant.tab, IGS antenna.gra), sollen über die Applikation transaktional importiert werden können. Die enthaltenen Daten sollen auf syntaktische und semantische Validität geprüft werden.

Es sollen mehrere IGS Sitelogs gleichzeitig importiert werden können. Schwerwiegende Fehler, Warnungen und Informationen über die Validität der IGS Sitelogs sollen sowohl clientseitig dargestellt als auch in Form von Fehlerprotokollen exportiert werden können. Sitelogs mit schwerwiegenden Fehlern dürfen nicht importiert werden.

Export von Dateien Aus den in der Datenbank vorhandenen Metadaten der Beobachtungsstationen sollen IGS Sitelogs generiert werden können. Ein Export aller anderen Metainformationen (z.B. rcvr_ant.tab) im CSV- oder PDF-Format soll ebenso realisiert werden.

Benutzeroberfläche Auf die Erstellung eines nutzerfreundlichen Webinterfaces ist in Absprache mit den Nutzern des Systems zu achten. Die Benutzeroberfläche soll zusätzlich um ein Internet-GIS-Modul mit Grundfunktionalitäten erweitert werden.

Internet-GIS Funktionalität Als Orientierungshilfe sollen die im IGS Sitelog vorhandenen Stationskoordinaten für die Darstellung in einer Online-Kartenanwendung genutzt werden. Das Internet-GIS-Modul soll dabei über Grundfunktionalitäten (u.a. Zoom, Panning, Maßstabsanzeige) verfügen und die globalen Beobachtungsstationen als Punktgeometrie repräsentieren. Darüber hinaus sollen wesentliche Metainformationen der Stationen (z.B. installierte Hardware) interaktiv abgefragt und clientseitig dargestellt werden können. Die Ergebnismenge soll durch die Implementierung geeigneter Filter eingeschränkt werden können.

Sonstiges Die Systemsprache ist aufgrund der Internationalität der Nutzer des Systems Englisch. Weitere Sprachmodule sind derzeit nicht vorgesehen.

3.2.2 Nichtfunktionale Anforderungen

Entscheidend für die Akzeptanz des Systems bei den Nutzern sind neben den funktionalen Anforderungen vor allem die nichtfunktionalen Anforderungen. Sie beschreiben, in welcher Qualität die geforderten Kriterien zu implementieren sind.

Zuverlässigkeit und Verfügbarkeit Die Nutzer des Systems stellen hohe Anforderungen an die Zuverlässigkeit des Systems. Alle gespeicherten Daten müssen konsistent und korrekt sein. Fehlerhaft abgelegte Metadaten führen automatisch zu Fehlern in der Prozessierung der Beobachtungsdaten. Darüber hinaus werden hohe Anforderungen an die Verfügbarkeit der Daten gestellt. Aufgrund dessen ist es ratsam einen Replikationsserver aufzusetzen, der ein oder mehrere Slave-Datenbanken (auf anderen Servern) mit der Master-Datenbank abgleicht, um mögliche Ausfälle zu kompensieren. Dieser Schritt ist aber nicht Teil dieser Arbeit.

Datensicherung Aufgrund der Relevanz der Metainformationen für die Prozessierung sollte die Datenbank täglich gesichert werden.

Bedienbarkeit Es wird besonderen Wert auf eine intuitive Bedienung des Systems ohne lange Einarbeitungszeit gelegt. Die Voraussetzung für eine gute Interaktion zwischen Nutzer und System ist hierfür ein ansprechendes Webdesign, das folgende Merkmale aufzuweisen hat:

- Einfache Darstellung,
- Einfacher, logischer und klarer Aufbau (visuelle Hierarchien),

- Verständlichkeit,
- Lesbarkeit.

Skalierbarkeit und Effizienz Die Datenbank wächst mit zunehmender Anzahl an Stationen stetig. Für die Gewährleistung eines effizienten Zugriffs auf die Daten muss daher ein besonderes Augenmerk auf die Entwicklung des Datenmodells geworfen werden, um Redundanzen und Inkonsistenzen im Datenbestand zu vermeiden.

Betriebssystemunabhängigkeit Eine Untersuchung der verwendeten Betriebssysteme am GFZ hat gezeigt, dass es keine Homogenität in diesem Bereich gibt. Es werden sowohl Microsoft Windows, verschiedene Linux-Distributionen sowie Mac OS X Betriebssysteme verwendet. Aufgrund dessen muss SEMISYS betriebssystemunabhängig entwickelt werden. Eine weitere Anforderung ist, dass die Nutzer keine zusätzliche Software installieren müssen und lediglich den Webbrowser für die Interaktion mit der Datenbank nutzen sollen. Da es auch in diesem Bereich keine Festlegung auf einen Hersteller gibt, muss die Applikation darüber hinaus browserunabhängig sein und für die gängigen Webbrowser funktionieren.

Erweiterbarkeit SEMISYS ist die Grundlage für weitere Entwicklungen, die die Archivierung und Validitätsprüfung der eingehenden Beobachtungsdaten betreffen. Das System soll deshalb leicht um zusätzliche Funktionalitäten erweiterbar sein, wobei möglichst viel Programmcode wiederverwendet werden sollte.

3.2.3 Technische Anforderungen

Neben den funktionalen und qualitativen wurden auch einige technische Anforderungen und Restriktionen an das System gestellt, die berücksichtigt werden mussten.

OpenSource Die Applikation soll keine Mehrkosten verursachen und daher auf Grundlage freier Software entwickelt werden.

Vorhandene Infrastruktur nutzen Die bereits bestehende technische Infrastruktur des GFZs soll genutzt werden und für die Implementierung des Systems ausreichen. Als Basiskomponenten stehen das objektrelationale DBMS (ORDBMS) PostgreSQL, der Webserver Apache und die Skriptsprachen Perl und PHP zur Verfügung.

Beschränkung des Einsatzes Die Applikation soll nur auf GFZ-internen Servern ohne Zugang zum World Wide Web (WWW) eingesetzt werden.

3.3 Stand der Technik

Es gibt derzeit kein freies oder kommerzielles System, das alle zuvor gelisteten, projektspezifischen Anforderungen an die Verwaltung von Stationsmetadaten erfüllt. Daher ist die Entwicklung einer eigenständigen Applikation, die auf die Bedürfnisse der Anwender zugeschnitten ist, unumgänglich.

Es lohnt sich dennoch eine Untersuchung und Analyse bereits bestehender Systeme zur Verwaltung, Aufbereitung und Darstellung von Stationsmetadaten. Daher sollen nachfolgend die Metainformationssysteme des IGS und des europäischen Pendants EPN (EUREF Permanent Network) exemplarisch vorgestellt und kritisch analysiert werden.

3.3.1 IGS Metainformationssystem

Datenverwaltung und Validitätsprüfung Der IGS nutzt für die Verwaltung der Stationsmetadaten keine Datenbank. Die eingehenden Sitelogs werden dateibasiert gehalten. Die Validitätsprüfung der Sitelogs erfolgt auf Grundlage eines PHP-basierten Parsers[16], der die Nutzer auf wichtige fehlende Werte hinweist. Hinweise auf etwaige Formatfehler (z.B. im Datumsformat) oder widersprüchliche Einträge (z.B. Enddatum vor Anfangsdatum) werden nicht gegeben. Ebenso ist es nicht möglich mehrere Sitelogs gleichzeitig zu prüfen. Darüber hinaus gibt es keinen Automatisierungsmechanismus zur Verwaltung eingehender Sitelogs. Aktualisierte Sitelogs müssen via E-Mail an das Zentralbüro des IGS geschickt werden, wo diese manuell geprüft und in das System eingepflegt werden.

Informationssystem Der Zugang zu den Informationsseiten der einzelnen Beobachtungsstationen kann über zwei Wege erfolgen:

- über eine tabellenbasierte Stationsliste oder
- über eine kartenbasierte Darstellung.

Die Stationsliste[17] enthält grundlegende, standortbezogene Informationen (Ort, Koordinaten, verantwortliche Organisation) über die Stationen. Eine stationsbezogene Such- oder kriterienbasierte Filterfunktion ist nicht vorhanden.

Zur globalen Präsentation der Stationen werden vorgefertigte statische Rasterkarten[18] (vgl. Abschnitt 2.3.2) genutzt. Diese Form bietet dem Nutzer wenig Möglichkeiten zur Interaktion. Die Karten dienen lediglich als Orientierung und als Querverweis zu den Informationsseiten der jeweiligen Station. Eine Filterung der Stationen nach bestimmten Kriterien ist dementsprechend nicht möglich.

Die Informationsseiten der einzelnen Stationen enthalten keine aufbereiteten Stationsmetainformationen. Es besteht lediglich die Möglichkeit der Anzeige aktueller wie alter

[16] siehe http://igscb.jpl.nasa.gov/network/sitelog_tester.php (Stand: 11.07.2012)
[17] siehe http://igscb.jpl.nasa.gov/network/list.html (Stand: 11.07.2012)
[18] siehe http://igscb.jpl.nasa.gov/network/complete.html (Stand: 11.07.2012)

Sitelogs. Die Seite enthält vielmehr Informationen über die Qualität der Beobachtungsdaten in Form statischer Graphiken.

3.3.2 EPN Metainformationssystem

Datenverwaltung und Validitätsprüfung Im Gegensatz zum IGS erfolgt die Verwaltung der Stationsmetadaten im EPN datenbankbasiert. Die Prüfung der Korrektheit von Sitelogs[19] erfolgt auf Basis eines PHP-Skripts, das das Sitelog in Schlüssel-Wert-Paare gliedert und auf syntaktische und semantische Validität testet. Die Darstellung der im Sitelog enthaltenen Metainformationen erfolgt formularbasiert, sodass der Nutzer auftretende Fehler direkt korrigieren kann. Nach erfolgreicher Validierung können die Metainformationen direkt in die Datenbank übernommen werden.

Informationssystem Der datenbankbasierte Ansatz bringt eine Vielzahl von Vorteilen mit sich. Zum einen können stationsbezogene Informationsseiten dynamisch aus den Einträgen in der Datenbank erzeugt werden, sodass kein zusätzlicher Aufwand bei der Verwaltung und Pflege der einzelnen Internetseiten entsteht. Zum anderen lassen sich die Metainformationen durch einfache SQL-Abfragen nach bestimmten Kriterien filtern, was wiederum zur Übersichtlichkeit beiträgt. Diese Vorteile spiegeln sich im Informationssystem des EPN wider. Der Zugang zu den stationsbezogenen Informationsseiten erfolgt wie beim IGS nach dem Zwei-Wege-Prinzip:

- über eine tabellenbasierte Stationsliste oder
- über eine kartenbasierte Darstellung.

In der Stationsliste[20] lassen sich vielfältige Informationen über die Stationen ein- und ausblenden. Eine Filterung nach bestimmten Kriterien ist nicht möglich. Diese Restriktion wird im kartenbasierten Ansatz[21] aber aufgelöst. Dort lassen sich die Stationen nach folgenden Kriterien filtern:

- Receiver-Hersteller,
- Antennen-Hersteller,
- GNSS-Typ (GPS, GLONASS, Galileo),
- Datentyp (Tagesdaten, Stundendaten, Real-Time-Daten),
- Status (aktiv, inaktiv, ehemalig).

Die Kartenanwendung wurde auf Basis der Google Maps API implementiert.

[19] siehe http://www.epncb.oma.be/_networkdata/sitelogsubmission (Stand: 11.07.2012)
[20] siehe http://www.epncb.oma.be/_networkdata/stationlist.php (Stand: 11.07.2012)
[21] siehe http://www.epncb.oma.be/_networkdata/stationmaps.php (Stand: 11.07.2012)

4 Konzeption und Entwurf

In Kapitel 3 wurden die funktionalen und nichtfunktionalen Anforderungen und Restriktionen an SEMISYS definiert und Eigenschaften bereits bestehender Metainformationssysteme analysiert. Die gewonnenen Erkenntnisse sollen in diesem Kapitel zur Modellierung eines realitätsnahen Software-Entwurfs beitragen.

4.1 Systemarchitektur

SEMISYS soll auf Grundlage eines webbasierten Ansatzes und unter Berücksichtigung der gegebenen technischen Infrastruktur entwickelt werden. Diese Restriktion und der Ansatz, dass das Internet-GIS-Modul auf Basis einer Online-Kartenanwendung, d.h. ohne zusätzlichen Applikationsserver, implementiert werden soll, vereinfacht die Wahl der Systemarchitektur.

Das System wird auf Basis des Client-Server-Modells entwickelt, sodass die Konsistenz der Informationen durch die zentrale Verwaltung und Verarbeitung immer gewährleistet ist und Änderungen unmittelbar allen Benutzern des Systems zugänglich sind.

Abbildung 4.1 beschreibt die Kommunikation zwischen Client und Server über TCP/IP im System. Der Nutzer stellt via Client (Webbrowser) eine Anfrage an den Webserver, der diese mit Hilfe der übergebenen Parameter verarbeitet. Das System ist modular aufgebaut, d.h. dass die serverseitigen Systemkomponenten in einzelne Module gegliedert sind und nach Bedarf in das Kernsystem integriert werden. Die Ergebnisse aus den Modulaufrufen werden an den Client zurückgesendet und mit Hilfe des Webbrowsers dargestellt. Das Basissystem umfasst Klassen für:

- die Kommunikation mit der Datenbank,
- die Authentifizierung von Nutzern via LDAP,
- die projekt- und nutzerbezogene Rechteverwaltung,
- die datenbankbasierte Verwaltung von Sitzungen (engl. *Sessions*),
- die Erstellung eines einheitlichen und wiederverwendbaren HTML-Grundgerüsts,
- die automatische Formularerzeugung,
- das Parsen von IGS-Dateien und die Validierung von Daten und Datentypen,

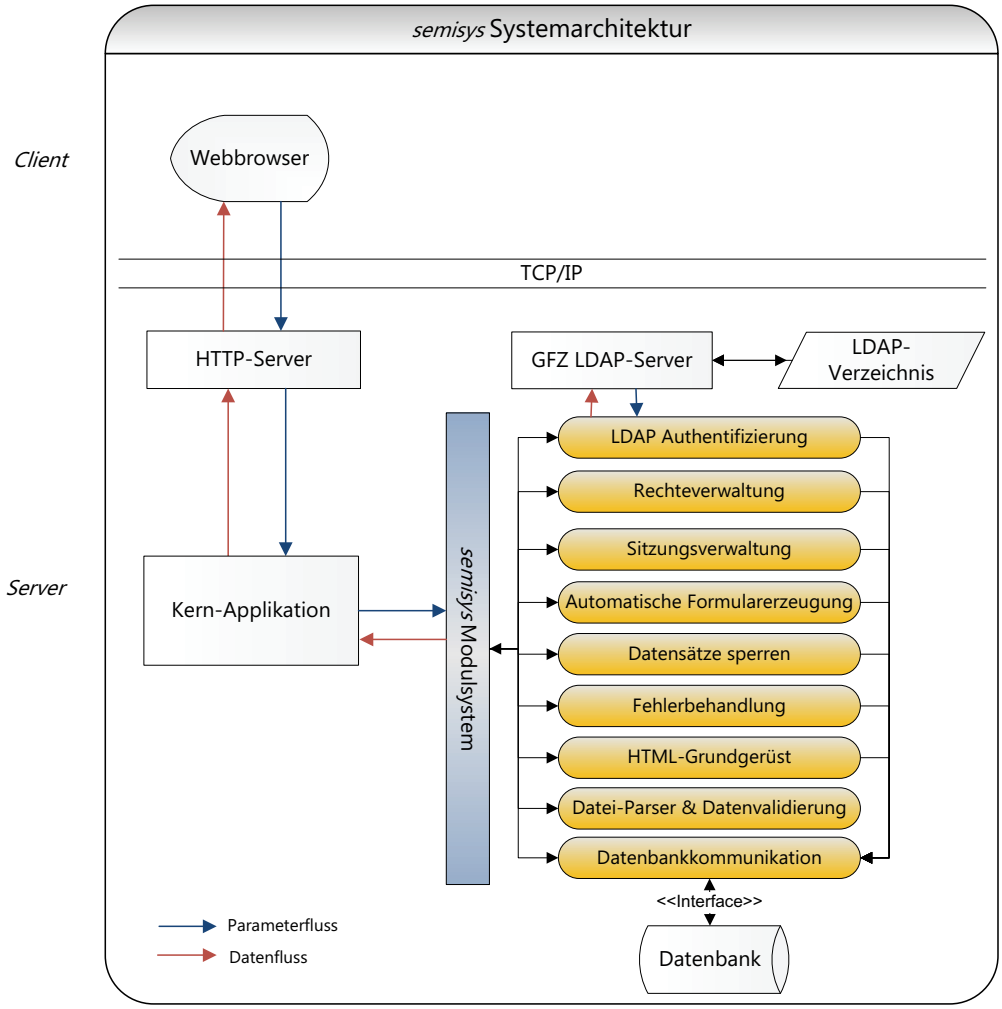

Abbildung 4.1: Systemarchitektur von SEMISYS

- die Behandlung und Protokollierung auftretender Skript- und Datenbankfehler,
- das Sperren von Datensätzen.

Im nächsten Abschnitt soll zunächst das Datenmodell näher betrachtet werden, da eine Vielzahl der Basisklassen mit der Datenbank kommuniziert. Anschließend werden die Basisklassen selbst beschrieben.

4.2 Datenmodellierung

Das Kernstück der gesamten Applikation stellt die zentrale Datenbank dar. Das Datenmodell ist die Grundlage für die weitere Entwicklung der Software. Aufgrund dessen muss auf die Modellierung der Daten ein besonderes Augenmerk gelegt werden, da Änderungen im Datenmodell zwangsläufig zu Änderungen in den einzelnen Programmen führen. Des Weiteren ist bei der Modellierung darauf zu achten, Datenredundanzen und somit Inkonsistenzen im Datenbestand zu vermeiden, um den Speicherbedarf zu minimieren und den Datenzugriff zu beschleunigen. Die Normalisierung der Daten[22] findet an dieser Stelle ihre Anwendung.

4.2.1 Verwaltung von Metainformationen

Abschnitt 3.1 hat bereits einen Einblick in die Quantität und Komplexität der zu verwaltenden Stations- und Satellitenmetadaten gegeben. Daher soll an dieser Stelle nur ein Ausschnitt aus dem Datenmodell vorgestellt werden, das die wichtigsten, für die Prozessierung von Beobachtungsdaten benötigten, Metainformationen enthält.

4.2.1.1 Empfangsstationsmetadaten

Abbildung 4.2 zeigt einen Ausschnitt aus dem Datenmodell für die Verwaltung der Metainformationen für die Beobachtungsstationen. Auffällig ist die sternförmige Anordnung der einzelnen Entitäten um die zentrale Tabelle `sites`, die lediglich den Primärschlüssel als eindeutige Identifikationsnummer (ID) und den vierstelligen Stationsnamen beinhaltet. Diese ID dient als Fremdschlüssel für die referenzierenden Entitäten, die die eigentlichen Metainformationen entsprechend der Sitelog-Deklaration (vgl. Abschnitt 3.1.1) beinhalten. In der Entität `receiver_metadata` werden alle wichtigen Informationen (Empfängertyp, GNSS-Typ, Seriennummer, Firmware, Installations-/ Deinstallationsdatum, etc.) über die auf der jeweiligen Station installierten Empfänger verwaltet. Die Informationen über die Empfängertypen werden in der Tabelle `receivers` gehalten. Die Beziehung zwischen diesen beiden Tabellen wird über den Fremdschlüssel *rec_id* der Tabelle `receiver_metadata` aufgebaut. Die Verbindungen zu den anderen Entitäten, von

[22] Für eine Zusammenfassung der gängigen Normalformen siehe auch Bradke (2009, S. 10f).

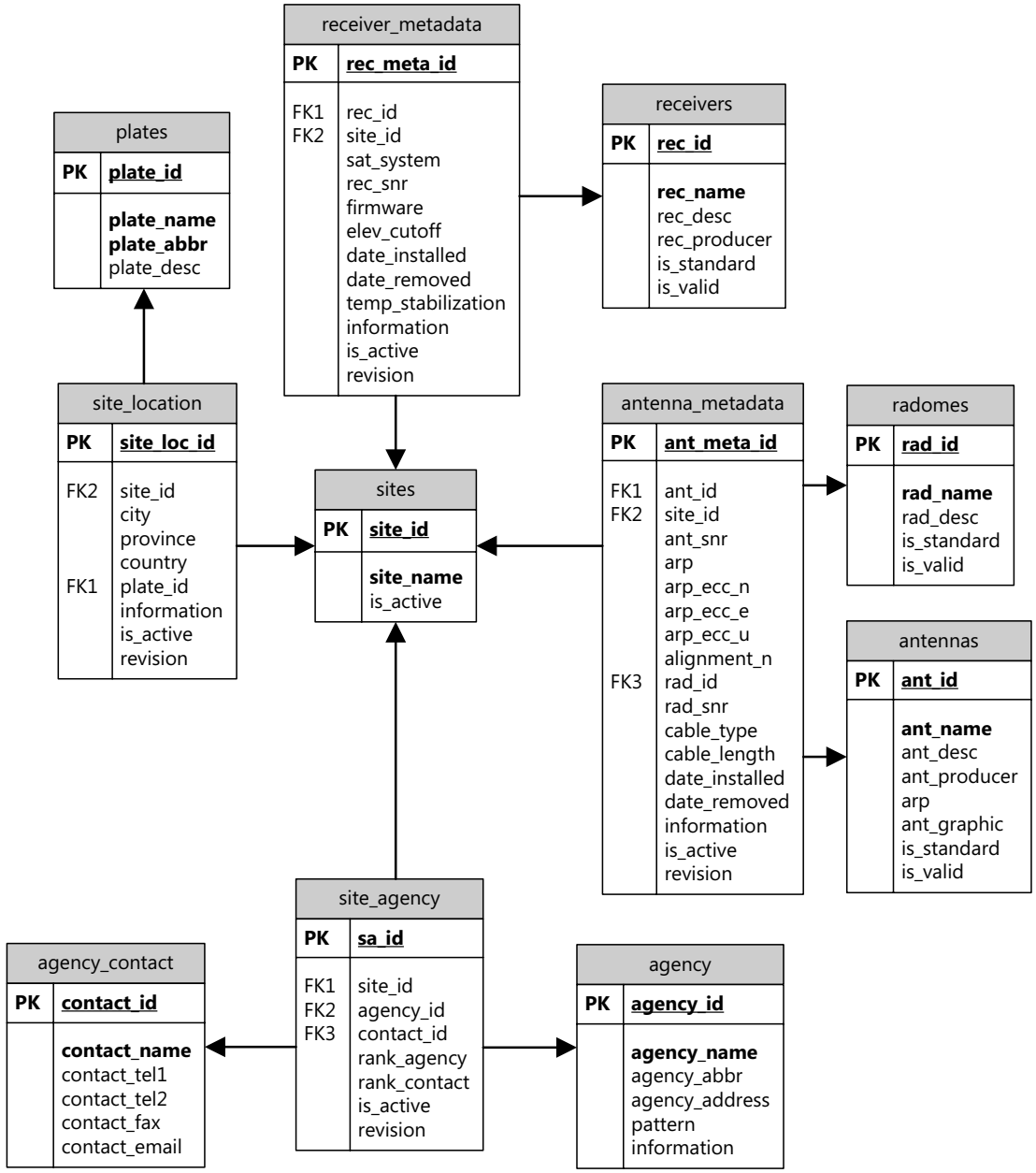

Abbildung 4.2: Ausschnitt aus dem Datenmodell für die Verwaltung der Stationsmetadaten

denen eine Vielzahl aus Gründen der Übersichtlichkeit in Abbildung 4.2 nicht aufgeführt sind, werden nach gleichem Muster aufgebaut.

An dieser Stelle sei auf die beiden Attribute *is_ active* und *revision* der jeweiligen Entitäten hingewiesen, die für die Versionierung der Metainformationen (siehe Abschnitt 4.2.3) benötigt werden.

4.2.1.2 Satellitenmetadaten

Die Modellierung der Satellitenmetadaten erweist sich im Gegensatz zur Verwaltung der Stationsmetadaten als wesentlich einfacher. Die Metainformationen zu den einzelnen Satelliten (vgl. Abschnitt 3.1.4) können in einer einzigen Entität (siehe Abbildung 4.3) gespeichert werden.

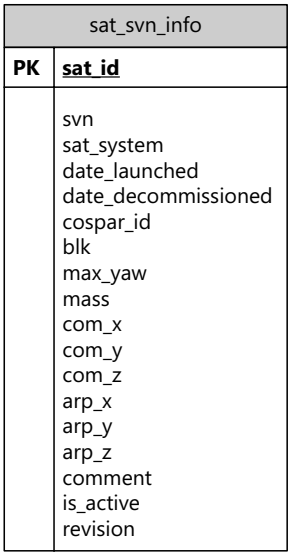

Abbildung 4.3: Datenmodell für die Verwaltung der Satellitenmetadaten

4.2.2 Projektbezogene Nutzer-, Rechte- und Sitzungsverwaltung

Die in Abbildung 4.4 dargestellten Entitäten dienen der Verwaltung von Nutzern, Rechten und Sitzungen innerhalb der Applikation.

Nutzerverwaltung In der Tabelle `users` werden alle Nutzer des Systems mit einer eindeutigen ID, Vor- und Nachname, E-Mail-Adresse sowie der Authentifizierungsart verwaltet. Letztere ist entscheidend, ob ein Passwort für den jeweiligen Nutzer angelegt werden muss. Bei einer Authentifizierung über `ldap` ist dies nicht erforderlich, da das eingegebene Passwort mit dem Passwort, das im LDAP-Verzeichnisdienst hinterlegt ist, verglichen wird. Bei manueller Authentifizierung ist die Angabe eines Passworts zwingend erforderlich.

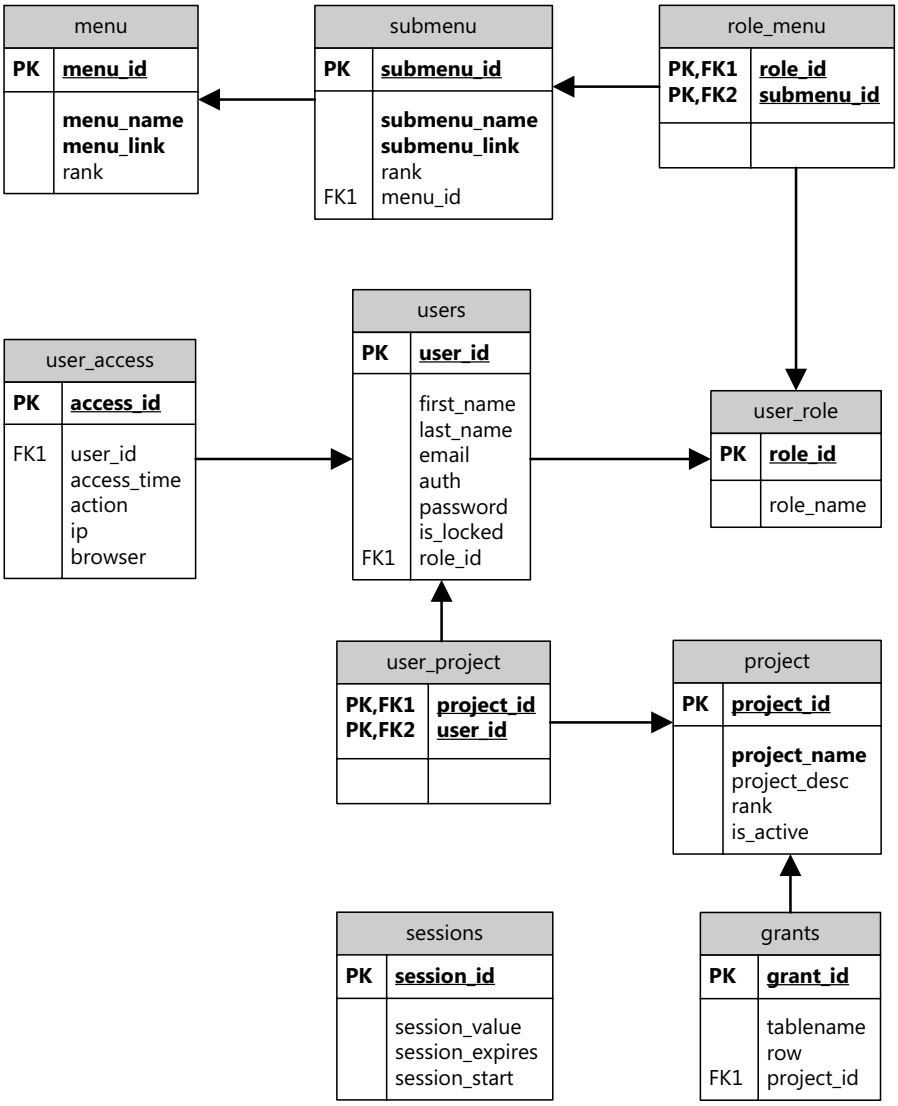

Abbildung 4.4: Datenmodell für die Verwaltung von Nutzern, Projekten und Rechten

Rollen- und Rechteverwaltung Darüber hinaus enthält die Tabelle den Fremdschlüssel der Entität `user_role`, der die Rolle des Nutzers innerhalb der Anwendung definiert. Dieses Rollenkonzept ist Bestandteil des zweistufigen Sicherheits- und Berechtigungssystems in SEMISYS. Über die Tabelle `role_menu` wird der Zugriff auf die einzelnen Menuunterpunkte der Applikation gesteuert. Eine gültige Kombination aus Rollennummer und Menunummer gilt als Zugriffsrecht auf den Menuunterpunkt. Diese Berechtigung ist aber lediglich applikationsgebunden, die Berechtigung für den Zugriff auf die Daten erfolgt über den zweiten Mechanismus.

In der Entität `grants` werden projektbezogene Rechte für den Zugriff auf die Daten erteilt. Jeder Nutzer hat lesenden Zugriff auf alle Daten, sofern die Berechtigung auf den Menupunkt besteht. Der schreibende Zugriff auf die Daten wird innerhalb der Tabelle abgebildet. Dort werden der Entitätsname der Datenbanktabelle, die zugehörige ID sowie das zugehörige Projekt gespeichert. Dies hat den Vorteil, dass die Berechtigungen datensatzbezogen vergeben werden können.

Sitzungsverwaltung Die Sitzungen der einzelnen Nutzer werden in der Tabelle `sessions` verwaltet. Die Session-ID dient dabei als Primärschlüssel. Im Attribut *session_ start* wird der Zeitpunkt des Sitzungsbeginns hinterlegt. Das Attribut *session_ expires* speichert den Zeitstempel der letzten Aktualisierung einer aktiven Sitzung. Über diesen Wert kann später das Ablaufen einer Sitzung gesteuert werden. In *session_ value* werden alle Werte abgelegt, die in der Sitzungsvariablen gespeichert sind.

4.2.3 Protokollierung von Änderungen

Bei einem Mehrbenutzerbetrieb ist es zwingend erforderlich, dass alle Änderungen in den Datensätzen protokolliert werden und ggf. wieder rückgängig gemacht werden können. Aufgrund dessen werden zusätzlich zu den eigentlichen Metainformationen Attribute zur Verwaltung des Zustands (*is_ active*) und der Revision (*revision*) gespeichert. Die Protokollierung der Änderungen erfolgt in der Entität `logs` (siehe Abbildung 4.5), in der der Datensatz durch Angabe von Tabellenname und zugehörigem Primärschlüssel eindeutig identifiziert wird.

Darüber hinaus werden der Nutzername sowie der Zeitpunkt der Datensatzänderung hinterlegt. Das Attribut *source* speichert den Ursprung der Information (z.B. Import aus rcvr_ant.tab), *status* beschreibt, ob es sich um einen neuen oder aktualisierten Eintrag handelt und *information* dient als Freitextinformation.

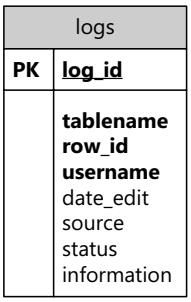

Abbildung 4.5: Datenmodell für die Protokollierung von Änderungen in Datensätzen

4.2.4 Datensätze sperren

Ein weiteres Problem in Mehrbenutzersystemen besteht in der Möglichkeit, dass mehrere Nutzer gleichzeitig denselben Datensatz bearbeiten können. Durch explizites Sperren von Datensätzen kann die Integrität der Daten gewahrt werden. Dazu werden in der Tabelle `locks` (siehe Abbildung 4.6) der Primärschlüsseleintrag des zu sperrenden Datensatzes (*row_ id*) und der Name der dazugehörigen Tabelle (*tablename*) gespeichert, die zusammen den eindeutigen Primärschlüssel dieser Tabelle ergeben. Dadurch kann für jeden Datensatz immer nur eine Sperre existieren. Darüber hinaus wird der Nutzername des sperrenden Nutzers in *locked_ by* sowie der Zeitpunkt, an dem die Sperre abläuft, in *locked_ until* hinterlegt.

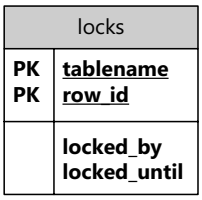

Abbildung 4.6: Datenmodell für die Sperrung von Datensätzen

4.2.5 Fehlerbehandlung

Keine Software ist frei von Fehlern. Daher ist es wichtig, skript- wie datenbankseitige Fehler zu protokollieren, um diese beheben zu können. Sofern eine Verbindung zur Datenbank möglich ist, werden auftretende Fehler in Skripten oder SQL-Statements abgefangen und in die Tabelle `error_log` (siehe Abbildung 4.7) eingetragen. Bei skriptseitigen Fehlern werden zusätzlich zur Fehlermeldung der Pfad zum fehlerhaften Skript (*script*), die fehlerverursachende Zeile (*line*) sowie der Zeitpunkt, an dem der Fehler auftrat (*timestamp*), gespeichert. Bei datenbankseitigen Fehler wird lediglich der Fehler sowie der Zeitstempel gespeichert, da aus der Fehlermeldung ersichtlich wird, bei welcher Abfrage der Fehler auftrat.

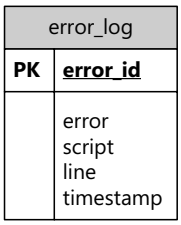

Abbildung 4.7: Datenmodell für die Verwaltung skript- und datenbankseitiger Fehler

4.3 Struktur und Funktionen der Systemkomponenten

In diesem Abschnitt sollen die Systemkomponenten, die bereits in Abbildung 4.1 dargestellt wurden, näher beschrieben werden. Das komplette System ist modular aufgebaut, sodass die einzelnen objektorientierten Klassen je nach Bedarf in die Skripte inkludiert und geladen werden können.

4.3.1 HTML-Grundgerüst

HTML-Dokumente bilden die Grundlage des World Wide Webs (WWW), sodass webbasierte Systeme zu großen Teilen aus diesen Dokumenten aufgebaut sind. In der Entwicklungsphase werden oft Änderungen am Grundgerüst der Webseiten durchgeführt. Um diese Änderungen für alle im Projekt betroffenen Seiten wirksam zu machen, wird eine Klasse `HTML` (siehe Abbildung 4.8) bereitgestellt, die statische Methoden zum Aufbau einer HTML-Seite beinhaltet.

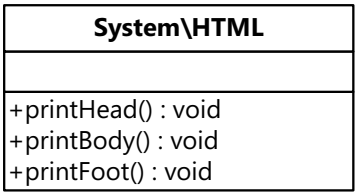

Abbildung 4.8: Die Klasse HTML

Die Methode `printHead()` beinhaltet die Dokumenttypdefinition (DTD) und gibt einen HTML-konformen Dateikopf aus. Dieser enthält Metadaten über die HTML-Seite und Informationen über die eingebundenen JavaScript-Bibliotheken und CSS-Deklarationen. Der HTML-Körper, der die Informationen über die Visualisierung der Internetseite beinhaltet, wird durch die Funktion `printBody()` eingeleitet. An dieser Stelle wird das Layout der Webseite in Abhängigkeit von den im CSS definierten Stilinformationen sowie die Menustruktur geladen. Die Methode `printFoot()` schließt den Rumpf des HTML-Körpers.

4.3.2 Datenbankkommunikation

Die permanente Kommunikation mit der Datenbank ist ein wichtiger Bestandteil des Gesamtsystems, um lesend wie schreibend auf die Daten zugreifen zu können. Eine der technischen Anforderungen war die Verwendung des ORDBMS PostgreSQL. Für die Verbindung zu dieser Datenbank werden in jeder Programmiersprache entsprechende Schnittstellen bereitgestellt. Die Entwicklung der Software ist in diesem Fall aber auf ein einziges DBMS beschränkt, sodass bei einem Wechsel zu einem anderen Datenbankhersteller alle Schnittstellen entsprechend modifiziert werden müssten. Aus diesem Grund empfiehlt sich der Einsatz von Datenbankschnittstellen, die unabhängig vom darunter liegenden Datenbanksystem operieren. Der Konstruktor der abstrakten Klasse `Database` (siehe Abbildung 4.9) bestimmt, zu welcher Datenbank eine Verbindung hergestellt werden soll. Hierfür werden folgende Parameter für den Verbindungsaufbau benötigt:

- Data Source Name (DSN); bestehend aus Datenbanktreiber, Datenbankhost, Datenbankport und Datenbankname,
- Eigentümer der Datenbank,
- Passwort des Eigentümers.

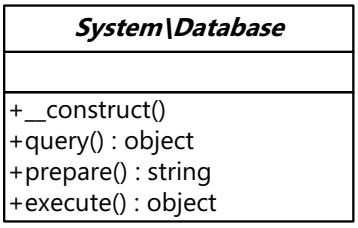

Abbildung 4.9: Die Klasse für die Kommunikation mit der Datenbank

Die Methode `query()` schickt eine SQL-Anfrage an die verbundene Datenbank und erhält als Ergebnis ein Objekt in Form eines assoziativen Arrays zurück. Eine andere Möglichkeit eine Datenbank abzufragen, besteht in der Nutzung so genannter *Prepared Statements*, die im Vergleich zu erster Variante anstelle der Parameterwerte Platzhalter enthalten und lediglich eine Anfrage vorbereiten. Die Vorbereitung dieser Anfrage erfolgt in der Methode `prepare()`, die Ausführung mit den übergebenen Werten in der Funktion `execute()`. Der größte Vorteil dieser Variante besteht darin, dass die Gültigkeit der Parameter vor der Ausführung überprüft wird und sich somit eingeschleuste SQL-Statements (SQL-Injection) verhindern lassen. Darüber hinaus ergibt sich ein Geschwindigkeitsvorteil bei mehrmaligen Ausführen mit unterschiedlichen Parametern, da die Anweisung nur einmal übersetzt werden muss.

4.3.3 Fehlerbehandlung

Alle Programmiersprachen bieten eine Form der Behandlung von Fehlern an. Das Problem ist allerdings, dass viele Fehler nur in bestimmten Situationen, die in entwickelten Testszenarien u.U. nicht bedacht wurden, auftreten und nur für den Nutzer sichtbar sind. Aus diesem Grund sollen auftretende Fehler protokolliert werden. Abbildung 4.10 beschreibt die allgemeine Vorgehensweise bei Eintreten von Fehlern.

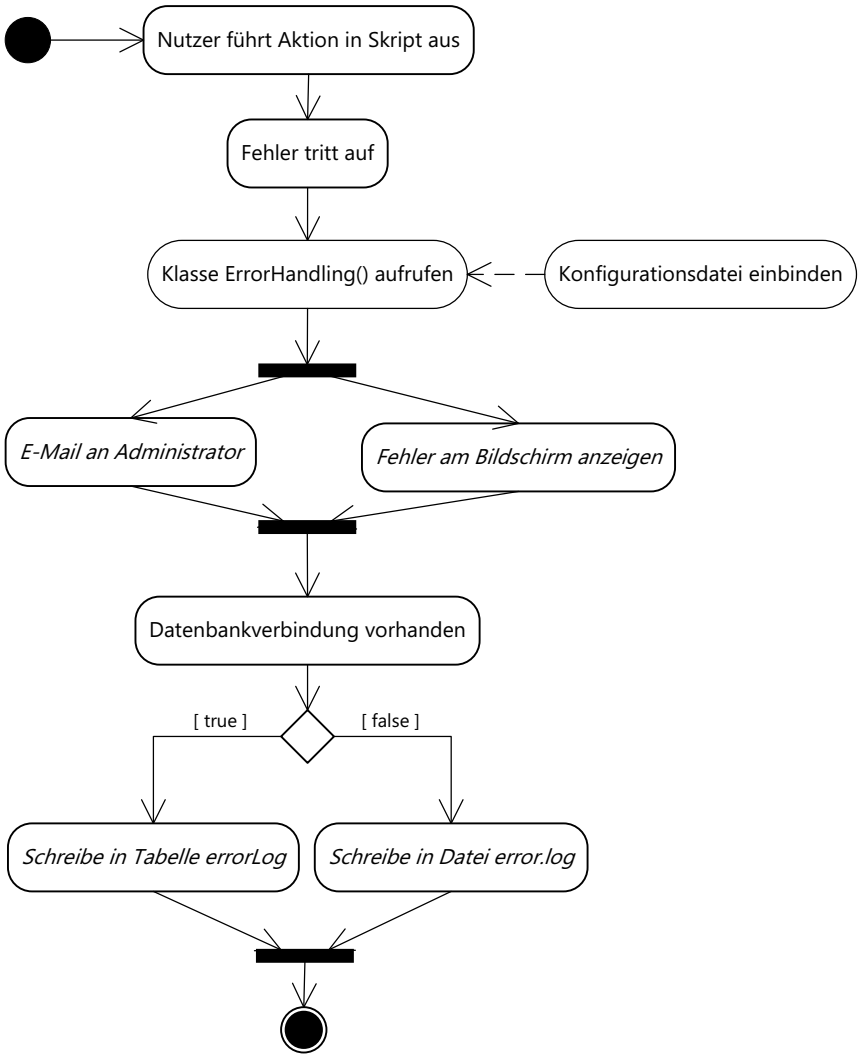

Abbildung 4.10: Aktivitätsdiagramm für die Behandlung auftretender Fehler

Es wird grundsätzlich zwischen skriptseitigen und datenbankseitigen Fehlern differenziert. Bei auftretenden Fehlern wird die Klasse `ErrorHandling` (siehe Abbildung 4.11) für die Fehlerbehandlung aufgerufen, die eine Konfigurationsdatei mit wichtigen Einstellungen[23] für die Protokollierung einbindet.

[23] Die Einstellungen beinhalten, welche Art der Protokollierung bzw. Fehleranzeige gewählt wird und welche Fehlerarten (Informationen, Warnungen, schwerwiegende Fehler) protokolliert werden sollen.

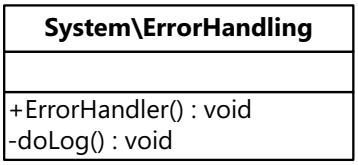

Abbildung 4.11: Die Klasse für die Behandlung skriptseitiger Fehler

Die von außen zugängliche Methode `ErrorHandler()` enthält als Parameter die Fehlernummer, die Fehlermeldung sowie bei skriptseitigen Fehlern zusätzlich den Skriptnamen sowie die Zeile, in der der Fehler auftrat. Diese Parameter werden bei entsprechender Konfiguration auf dem Bildschirm ausgegeben oder per E-Mail an den Administrator versandt. Darüber hinaus wird aus dieser Methode eine private Methode `doLog()`[24] aufgerufen, die diese Parameter bei vorliegender Datenbankverbindung in die Tabelle `errorLog` (vgl. Abbildung 4.7) einträgt. Liegt keine Verbindung vor, erfolgt die Protokollierung der Fehlermeldung in der Datei `error.log`.

4.3.4 Authentifizierung

Die Klasse `Auth` (siehe Abbildung 4.12) dient der Authentifizierung von Nutzern im System.

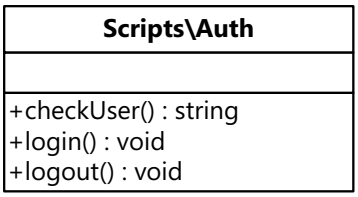

Abbildung 4.12: Die Klasse für die Authentifizierung von Nutzern

Dabei wird von der Anmeldung eines Nutzers durch Angabe eines Nutzernamens und Passworts über ein webbasiertes Formular ausgegangen. Das in Abbildung 4.13 dargestellte Aktivitätsdiagramm beschreibt den elementaren Ablauf bei der Authentifizierung der Nutzer. Nach Absenden des Formulars wird die Methode `checkUser()` aufgerufen, die zunächst prüft, welche Authentifizierungsart für den angegebenen Nutzernamen existiert. Falls der Nutzername nicht in der Datenbank registriert ist, wird der Zugriff verweigert. Bei einer Authentifizierung via LDAP werden die Benutzerangaben mit denen im LDAP-Verzeichnisdienst verglichen. Bei einer manuellen Authentifizierung erfolgt der Test auf eine gültige Nutzer/Passwort-Kombination durch Abgleich der Daten in der Tabelle `users` (vgl. Abbildung 4.4). Bei einer gültigen Kombination wird die Methode `login()` aufgerufen. Diese registriert die Sitzungsdaten des Nutzers in der Datenbank

[24] Die Methode kann nur innerhalb der Klasse aufgerufen werden, um sie nach außen hin zu schützen.

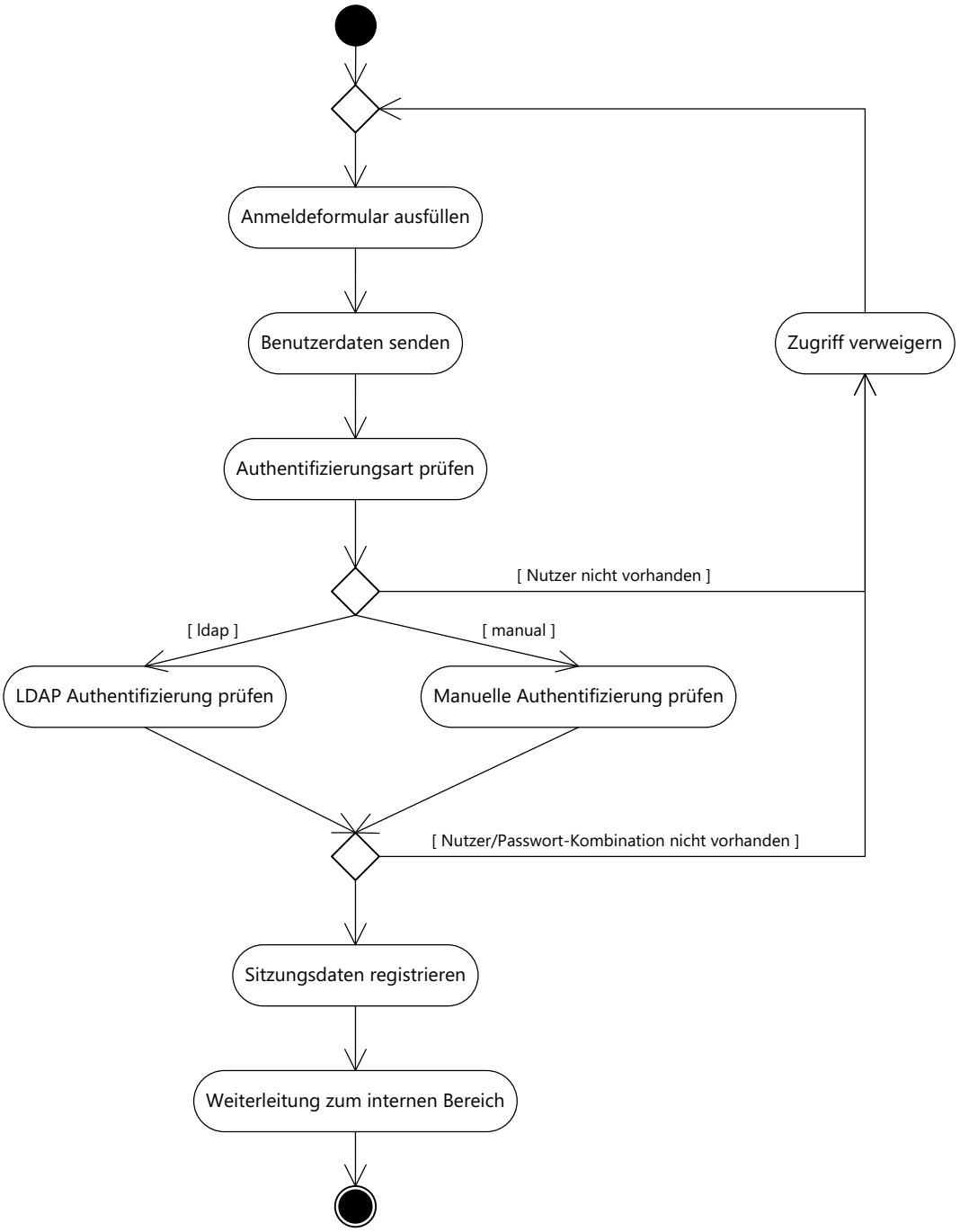

Abbildung 4.13: Aktivitätsdiagramm für die Authentifizierung von Nutzern

und leitet den Nutzer zum internen Bereich des Systems weiter. Im negativen Fall wird der Zugriff für den Nutzer verweigert.

Die Methode `logout()` löscht die Sitzung des Nutzers und leitet diesen zur Anmeldeseite weiter.

4.3.5 Rechteverwaltung

In den funktionalen Anforderungen (vgl. Abschnitt 3.2) wurde bereits spezifiziert, dass die Verwaltung der Rechte zweistufig erfolgen soll:

- Der Zugriff auf die Internetseiten wird über die Rollenzugehörigkeit geregelt.
- Der (schreibende) Zugriff auf Datensätze erfolgt über die Projektzugehörigkeit.

Die Klasse `grant` (siehe Abbildung 4.14) enthält zwei Methoden, die dieses zweistufige Modell abbilden.

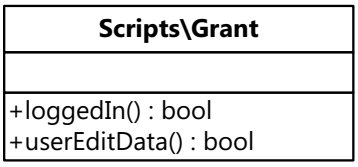

Abbildung 4.14: Die Klasse für die Rechteverwaltung

Die Funktion `loggedIn()` ist für die Steuerung des Zugriffs auf Menuunterpunkte entsprechend der Rollenzugehörigkeit zuständig. Dabei wird zunächst geprüft, ob eine Sitzung überhaupt registriert ist. Im negativen Fall erfolgt eine Weiterleitung zur Anmeldeseite. Im positiven Fall wird als nächstes getestet, ob die Rolle des Nutzers zur Ansicht der aufgerufenen Seite berechtigt. Dafür wird eine Verknüpfung zwischen den Tabellen `users`, `user_role`, `role_menu`, `menu` und `submenu` (vgl. Abbildung 4.4) hergestellt, die im positiven Fall einen entsprechenden Datensatz zurückliefert und somit den Zugriff auf die Seite erlaubt. Bei leerer Ergebnismenge wird der Nutzer auf eine Seite mit dem Inhalt „Zugriff verweigert“ weitergeleitet.

Die Methode `userEditData()` definiert den schreibenden Zugriff auf Datensätze bestimmter Relationen entsprechend der Projektzugehörigkeit. Dabei wird geprüft, ob ein Nutzer einem bestimmten Projekt angehört, welches schreibenden Zugriff auf eine gesamte Relation oder bestimmte Datensätze einer Relation hat.

4.3.6 Projektbezogene Verwaltung von Metainformationen

Nachdem in den vorangegangenen Abschnitten das Grundgerüst der Applikation näher erläutert wurde, soll nachfolgend die eigentliche Verwaltung der Stations- und Satellitenmetadaten vorgestellt werden.

4.3.6.1 Import von IGS Sitelogs

Das in Abbildung 4.15 dargestellte Sequenzdiagramm beschreibt die Kommunikation der einzelnen Systemkomponenten beim Import von IGS Sitelogs auf vereinfachte Weise. Dabei wird von einem erfolgreichen Import der Sitelogs in die Datenbank ohne syntaktische oder semantische Fehler ausgegangen.

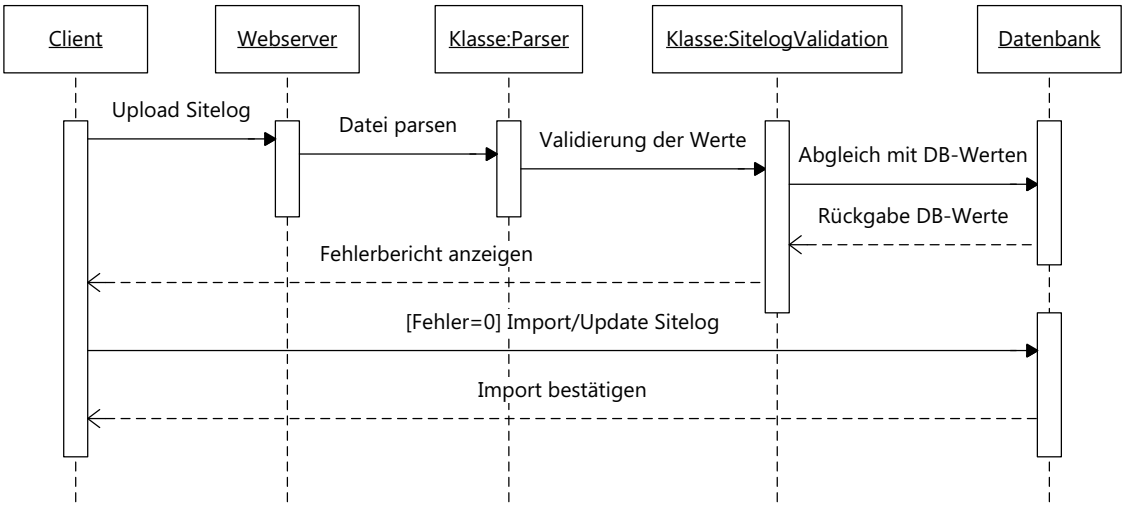

Abbildung 4.15: Vereinfachtes Sequenzdiagramm für den Import von IGS Sitelogs

Zunächst lädt der Nutzer ausgewählte IGS Sitelogs über eine Benutzerschnittstelle auf den Webserver, wo diese sequentiell abgearbeitet werden. Die Abarbeitung umfasst zwei wesentliche Schritte: das *Parsen*, d.h. das Zerlegen des Dateiinhalts in Schlüssel-Wert-Paare, und die *Validierung* der Werte nach erwarteten Dateitypen oder erwartetem Inhalt.

Parsen von IGS Sitelogs Die Klasse `Parser` umfasst vier Methoden (siehe Abbildung 4.16).

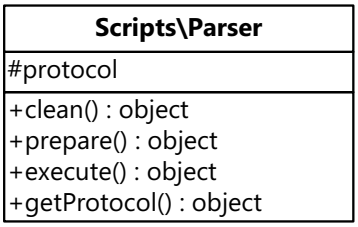

Abbildung 4.16: Die Klasse für das Parsen von IGS Sitelogs

Die Methode `clean()` öffnet die Datei und generiert ein Feld von Zeilen. Nicht benötigte oder leere Zeilen werden genauso wie unnötige Steuerzeichen aus dem Feld entfernt. Die

Methode `prepare()` zerlegt das Feld von Zeilen anhand des Trennzeichens „:“ in eine Schlüssel-Wert-Struktur und ordnet diese den Sektionsnummern zu. Das Resultat ist ein assoziatives Array, das sich aufgrund seiner Struktur eindeutig auf das Datenmodell abbilden lässt. Die Methode `execute()` prüft, ob die Schlüssel im Sitelog mit den erwarteten Schlüsseln in einem vordefinierten assoziativen Array übereinstimmen bzw. ob Schlüssel fehlen. Ist dies der Fall, wird das vordefinierte Feld mit den Werten des Sitelogs gefüllt. Im negativen Fall werden die syntaktischen Fehler in einem Feld (`protocol`) gespeichert, das über die Methode `getProtocol()` abgefragt werden kann.

Validierung von IGS Sitelogs Sofern keine syntaktischen Fehler vorliegen, kann der Inhalt der Sitelogs auf semantische Validität geprüft werden. Hierfür werden die Werte mit einem Parameter, der den zu erwartenden Datentyp enthält, an die Methode `checkValue()` der Klasse `SitelogValidator` (siehe Abbildung 4.17) übergeben. Zunächst

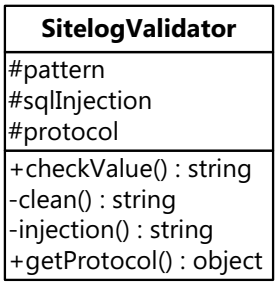

Abbildung 4.17: Die Klasse für die Validierung von IGS Sitelogs

werden die Werte bereinigt und auf eingeschleuste SQL-Statements untersucht. Hierfür werden die Methoden `clean()` und `injection()` eingesetzt. Anschließend werden die Werte auf Kompatibilität zum erwarteten Datentyp oder zum erwarteten Inhalt getestet. Der Vergleich der Datentypen erfolgt mit Hilfe von definierten Mustern (engl. *pattern matching*), die in dem Objekt `pattern` gespeichert sind. In manchen Fällen ist ein Abgleich mit den in der Datenbank hinterlegten, vordefinierten Werten erforderlich, um z.B. standardisierte Receiver- oder Antennentypen eindeutig zu identifizieren.

Bei der Validierung der Werte wird zwischen drei Fehlerarten, die sich hinsichtlich ihrer Schwere unterscheiden, differenziert:

- Informelle Bemerkungen,
- Warnungen und
- schwerwiegende Fehler.

Diese werden in dem Feld `protocol` gespeichert und können über die Methode `getProtocol()` abgefragt werden.

Fehlerklasse	Beschreibung
Fremdschlüssel	Receivertyp nicht in Datenbank vorhanden
	Antennentyp nicht in Datenbank vorhanden
	Radomtyp nicht in Datenbank vorhanden
	Geotektonische Platte nicht in Datenbank vorhanden
	Zuständige Organisation(en) nicht in Datenbank vorhanden
Format	Falsches Stations-ID-Format
	Falsches Koordinatenformat
	Falsches Datumsformat
	Falsches E-Mail-Format
	Wert außerhalb einer definierten Länge
Datum	Fehlendes Datum
	Invalide Zeitraumsangabe
Einheiten	Unerwartete Maßeinheit
SQL-Injection	Eingeschleuste SQL-Abfrage

Tabelle 4.1: Schwerwiegende semantische Fehler in IGS Sitelogs

Import von IGS Sitelogs in die Datenbank Der Import der Sitelogs in die Datenbank ist abhängig von der Schwere der registrierten Fehler. Sofern keine schwerwiegenden Fehler (vgl. Tabelle 4.1) vorliegen, ist der Import prinzipiell möglich, aber von weiteren Faktoren abhängig, die sich in Abbildung 4.18 widerspiegeln. Die Verwaltung der Stationsmetadaten erfolgt projektbezogen, d.h. dass der Nutzer ein für die Station zugehöriges Projekt[25] auswählen muss, um den Import zu initialisieren. Anschließend wird anhand verschiedener Kriterien[26] und Eigenschaften geprüft, ob die Station bereits in der Datenbank registriert ist. Ist dies nicht der Fall, erfolgt der initiale Import der Informationen. Im positiven Fall muss als nächstes die Berechtigung zur Aktualisierung der Informationen geprüft werden. Falls die Station bereits einem übergeordneten Projekt angehört, dürfen die Informationen nicht eingefügt werden. Besteht die Berechtigung zum Import, wird als letztes getestet, ob eine Aktualisierung nötig ist. Dazu wird die MD5-Summe des aktuellen Sitelogs mit der MD5-Summe des letzten importierten Sitelogs verglichen. Bei einer notwendigen Aktualisierung werden alle in der Datenbank vorhandenen Metainformationen auf inaktiv gesetzt und der Import initialisiert.

Entscheidend ist, dass der Import transaktional (vgl. Abschnitt 2.2.1) durchgeführt wird, um die Konsistenz des Datenbestandes bei möglichen Fehlern in den Abfragen zu erhalten. Jede durchgeführte Operation wird zusätzlich in der Tabelle `logs` (vgl. 4.2.3) protokolliert.

[25] Die Auswahlliste enthält nur dem Nutzer zugeordnete Projekte.

[26] Zu diesen Kriterien zählen der vierstellige Stationsname, der Typ und der Zeitpunkt der ersten installierten Hardware sowie die verantwortliche(n) Organisation(en).

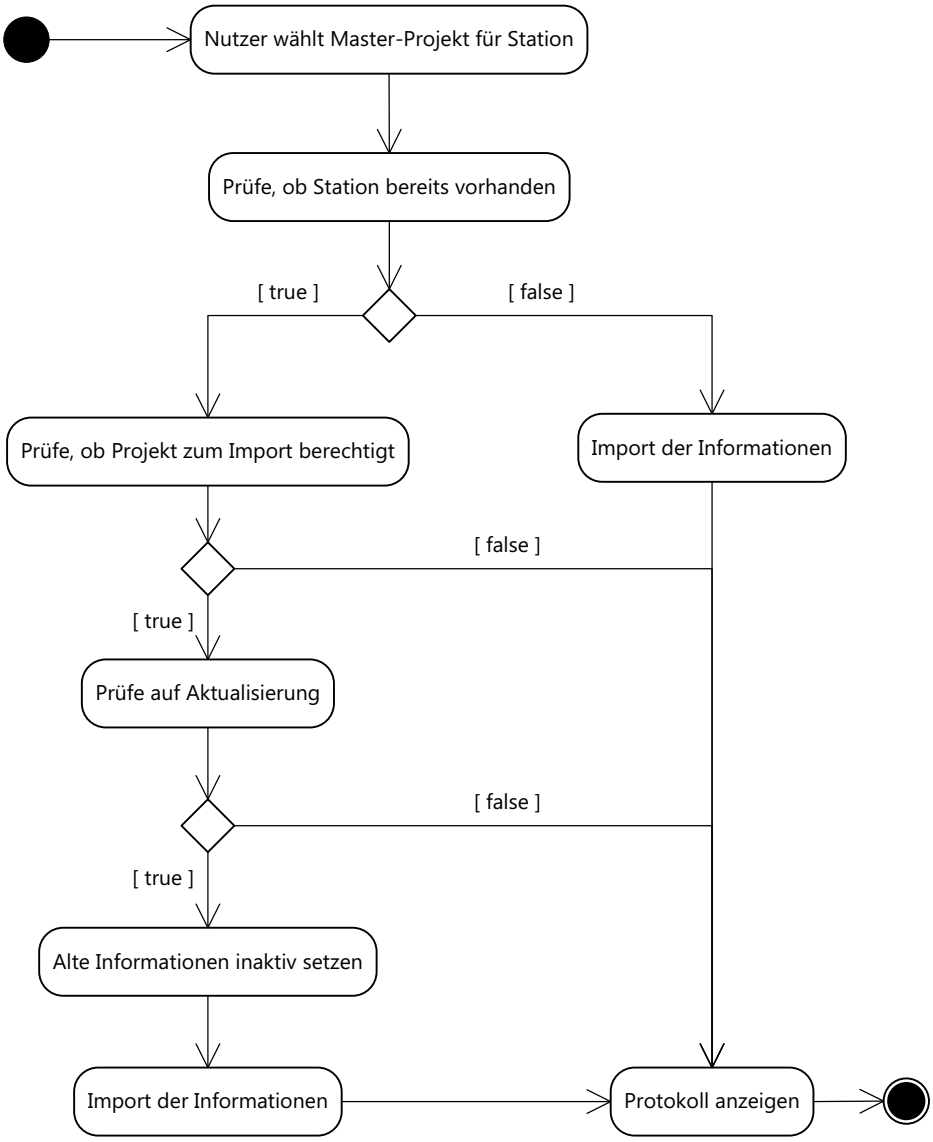

Abbildung 4.18: Aktivitätsdiagramm für den Import von IGS Sitelogs

4.3.6.2 Import von Receiver- und Antenneninformationen

Die Dateien *rcvr_ant.tab* und *antenna.gra* (vgl. Abschnitte 3.1.2 und 3.1.3), die wichtige Informationen über die vom IGS standardisierten Gerätenamen enthalten, werden in unregelmäßigen Abständen aktualisiert. Dementsprechend müssen die enthaltenen Informationen auch in der Datenbank auf den aktuellen Stand gebracht werden, sodass ein entsprechender Parser entwickelt werden muss. Zu beachten ist, dass die Informationen im Gegensatz zu den in den Sitelog enthaltenen Metainformationen nicht auf inaktiv gesetzt werden, sondern wirklich aktualisiert werden, da die Primärschlüssel der Tabellen `receivers`, `antennas` und `radomes` mit den Metainformationen der Stationen referenziert sind. Abbildung 4.19 beschreibt den Ablauf der Aktualisierung von Hardwaremetainformationen.

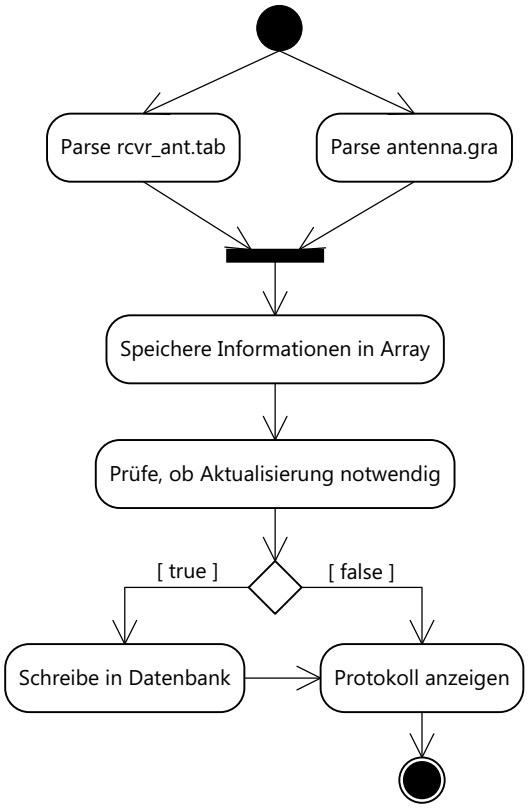

Abbildung 4.19: Aktivitätsdiagramm für den Import von Hardware-Metainformationen

Zunächst müssen die Metainformationen aus den beiden Dateien extrahiert und in ein assoziatives Array der in Tabelle 4.2 dargestellten Struktur überführt werden.
Anschließend wird geprüft, ob eine Aktualisierung der Datenbank notwendig ist, indem die Werte innerhalb der Datei mit den in der Datenbank enthaltenen Informationen verglichen werden. Bei Abweichungen werden die betroffenen Relationen entsprechend aktualisiert bzw. hinzugefügt.

Schlüssel	Werte
Receivertyp	Receiverhersteller, Beschreibung, Status
Antennentyp	Antennenhersteller, Beschreibung, Status, Antennenreferenzpunkt, Antennengraphik
Radomtyp	Beschreibung, Status

Tabelle 4.2: Struktur des Arrays zur Speicherung der Hardwaremetainformationen

4.3.6.3 Formularerzeugung und Datensätze sperren

In vielen Fällen sollen die Metainformationen nicht über einen Import von Dateien aktualisiert werden, sondern mit Hilfe von Formularen. Da Formulare in der Regel nach dem gleichen Schema aufgebaut sind und nur eine überschaubare Menge von elementaren Typen enthalten, soll eine entsprechende Klasse modelliert werden, mit deren Hilfe Formulare automatisch erstellt werden und somit die Datenbearbeitung und -manipulation erleichtern.

Abbildung 4.20 beschreibt den Ablauf von der automatischen Generierung von Formularen bis hin zur dynamischen Erstellung von Tabellen.

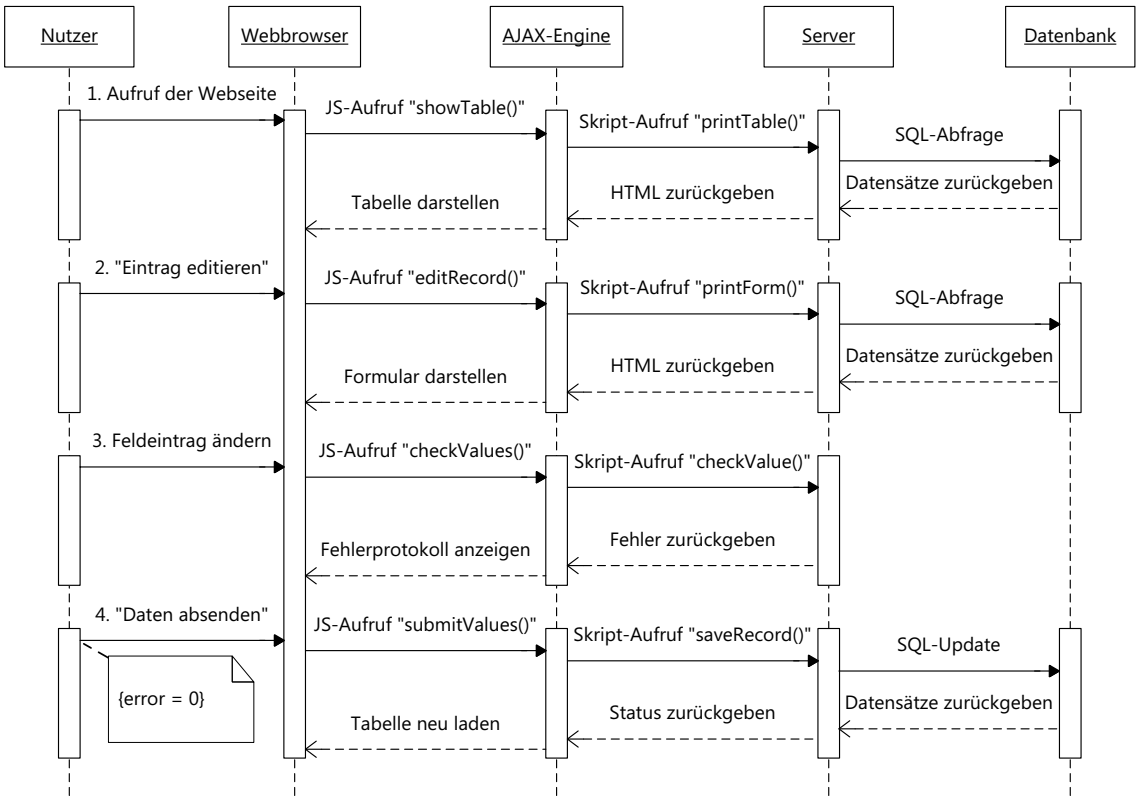

Abbildung 4.20: Sequenzdiagramm für den Ablauf der Generierung automatischer Tabellen und Formulare

Schritt 1: Aufruf der Webseite Mit dem Aufruf einer Webseite durch den Nutzer wird auf Seite des Clients die JavaScript-Funktion `showTable()` aufgerufen, die eine asynchrone HTTP-Anfrage (AJAX-Request) an das serverseitige Skript sendet. Als Parameter wird der Name der Datenbankentität, die dargestellt werden soll, übergeben. Anschließend wird der Konstruktor der Klasse `AutoForm` (siehe Abbildung 4.21) mit diesem Parameter aufgerufen. Dieser wird benötigt, um die Struktur der Datenbankta-

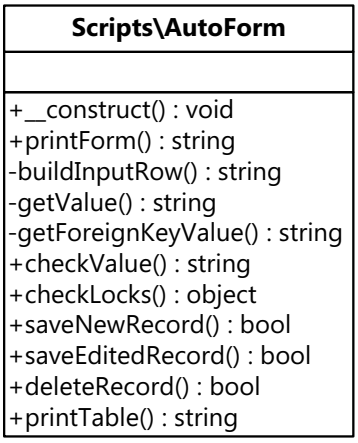

Abbildung 4.21: Die Klasse zur Generierung dynamischer Formulare

belle zu beschreiben und somit die Formulare zu konstruieren. Für die Darstellung der Tabelle wird die Methode `printTable()` genutzt, die die Informationen aus der Datenbank extrahiert und als HTML formatiert. Das Ergebnis wird als Text an die aufrufende JavaScript-Funktion zurückgegeben und ausgegeben. Da nicht immer alle Spalten einer Tabelle ausgegeben werden sollen, ist es nötig die entsprechenden Spalten in einer Konfigurationsdatei zu definieren.

Schritt 2: Eintrag editieren Für das Editieren von Datensätzen muss ein entsprechendes Formular erzeugt werden, das die vorhandenen Datenbankeinträge für den aufgerufenen Datensatz beinhaltet. Die Methode `printForm()` erzeugt unter Angabe der Datenbanktabelle und der zugehörigen Datensatz-ID und unter Berücksichtigung der in der Konfigurationsdatei definierten Spalten ein Formular, das in der Grundkonfiguration[27] aus folgenden HTML-Elementen bestehen kann:

- Einzeiliges Textfeld (`<input type="text"\>`)
- Mehrzeiliges Textfeld (`<textarea\>`)
- Auswahlliste (`<select\>`)
- Schaltfläche zum Absenden (`<input type="submit"\>`)

[27] Die Klasse lässt sich problemlos um andere Formular-Elemente erweitern.

Die Ausgabe richtet sich nach dem Datentyp des jeweiligen Attributs und wird in der Methode `buildInputRow()` gebildet. Die Attributwerte werden durch die Methode `getValue()` zurückgegeben. Fremdschlüsseleinträge, die für die Füllung der Auswahllisten benötigt werden, werden durch die Methode `getForeignKeyValue()` bestimmt.

Schritt 3: Feldeintrag ändern Mit jeder Änderung im Formular wird ein AJAX-Request an die Methode `checkValue()` gesendet, der die Einträge auf Validität überprüft und auftretende Fehler in Echtzeit auf dem Bildschirm anzeigt. Sofern keine Fehler aufgetreten sind, wird die Schaltfläche zum Absenden des Formulars angezeigt.

Schritt 4: Daten absenden Mit Betätigung der Schaltfläche zum Absenden des Formulars wird eine AJAX-Anfrage an die Methode `saveEditedRecord()` gestellt, die die enthaltenen Informationen in der Datenbank aktualisiert. Sofern keine Fehler bei der Datenbankabfrage auftreten, wird die aktualisierte Tabelle direkt angezeigt. Bei Fehlern werden diese auf dem Bildschirm ausgegeben.

Um Datensätze editieren oder löschen zu können, muss zunächst geprüft werden, ob keine Sperre durch einen anderen Nutzer auf diesem Datensatz existiert. Das Setzen, Löschen und Prüfen von Sperren wird durch die Klasse `Locks` (siehe Abbildung 4.22) gesteuert.

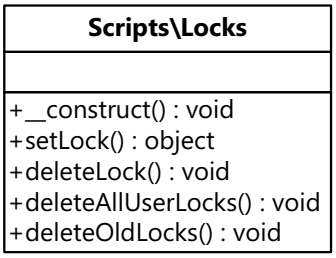

Abbildung 4.22: Die Klasse für das Sperren von Datensätzen

Der Konstruktor der Klasse hat die Aufgabe, die Dauer einer Sperre (in Sekunden) auf einer privaten Variable zu speichern, die innerhalb der Klassenmethoden zugänglich ist. Sperren werden in der Methode `setLock()` angelegt. Dabei wird unter Angabe des Tabellennamens, der Datensatz-ID und des Nutzernamens zunächst geprüft, ob bereits eine Sperre für den Datensatz durch Abfrage der Tabelle `locks` (siehe Abbildung 4.6) existiert. Falls nicht, wird die Sperre in der Tabelle registriert. Im anderen Fall wird ein Datensatz mit den Informationen über den Ablauf der Sperre zurückgegeben.

Die Methode `deleteLock()` löscht einen spezifischen Datensatz einer Tabelle unabhängig vom Nutzer.

Die Methode `deleteAllUserLocks()` löscht alle von einem Nutzer gesetzten Sperren

unabhängig von der betroffenen Tabelle oder dem betroffenen Datensatz. Dadurch wird erreicht, dass ein Nutzer immer nur einen Datensatz in Bearbeitung haben darf und unnötige Sperren entfallen.

Die Methode `deleteOldLocks()` löscht alle abgelaufenen Sperren, sodass die Datensätze wieder zur Bearbeitung frei gegeben werden können.

4.4 Interaktive Kartendarstellung

In Abschnitt 2.3 wurden bereits die gängigen technologischen Möglichkeiten für die Implementierung von Internet-GIS-Funktionalitäten dargestellt. Im Allgemeinen lässt sich demnach zwischen client- und serverseitiger Kartendarstellung differenzieren. Nachfolgend sind die Anforderungen und Restriktionen (vgl. Abschnitt 3.2) an die Anwendung, die über die Wahl der Technologie entscheiden, noch einmal aufgeführt:

- Verzicht auf zusätzlich zu installierende Software (Kartenserver),
- Überschaubarer Funktionsumfang der Anwendung (mehr Visualisierungs- als Analysefunktionen),
- Geodatenbestand besteht einzig aus Punktgeometrien der Beobachtungsstationen.

Die dargestellten Anforderungen führten zur Verwendung einer clientseitigen Kartenanwendung, deren allgemeiner Systemaufbau in Abbildung 4.23 dargestellt ist.

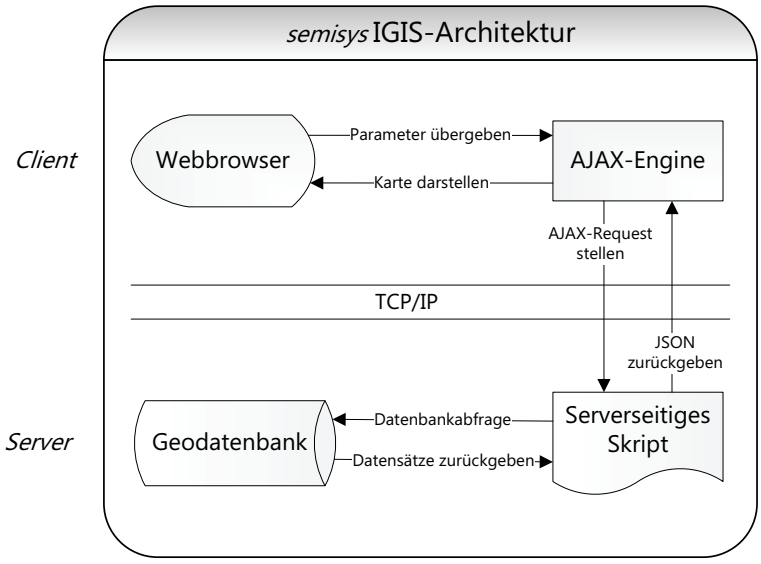

Abbildung 4.23: Systemarchitektur des Internet-GIS-Moduls

Nach dem Aufruf der Webseite, in der die Karte dargestellt werden soll, wird eine JavaScript-Funktion aufgerufen, die eine asynchrone HTTP-Anfrage (AJAX-Request)

an ein serverseitiges Skript stellt. Als Parameter wird lediglich eine Initialisierungsvariable übergeben. Das serverseitige Skript verarbeitet die Anfrage und stellt eine SQL-Anfrage an die Datenbank. Die zurückgegebenen Datensätze, die neben den Geodaten auch Stationsmetainformationen enthalten, werden als JSON[28]-Objekt kodiert und an die AJAX-Engine zurückgegeben. Das zurückgegebene JSON-Objekt wird an die JavaScript-Funktion, die die Kartendarstellung initialisiert, übergeben und dekodiert. Das Resultat ist eine Karte, die die Punktgeometrien der Stationen als Marker beinhalten. Die Marker enthalten zusätzliche Metainformationen.

Ferner sollen sich die Stationen nach verschiedenen Kriterien filtern lassen. Dafür reicht lediglich die Übergabe entsprechender Parameter, die für die SQL-Abfrage benötigt werden.

4.5 Graphische Benutzeroberfläche

Der Entwurf der Benutzeroberfläche folgt den Grundsätzen guten Webdesigns, die bereits in Abschnitt 3.2.2 dargelegt wurden. Hinzu kommt, dass die Anwendung im Zusammenhang mit dem GFZ bzw. GFZ-Applikationen assoziiert, aber trotzdem als eigenständige Anwendung wahrgenommen werden soll. Abbildung 4.24 zeigt den Entwurf der graphischen Benutzeroberfläche (engl. *Graphical User Interface*, GUI).

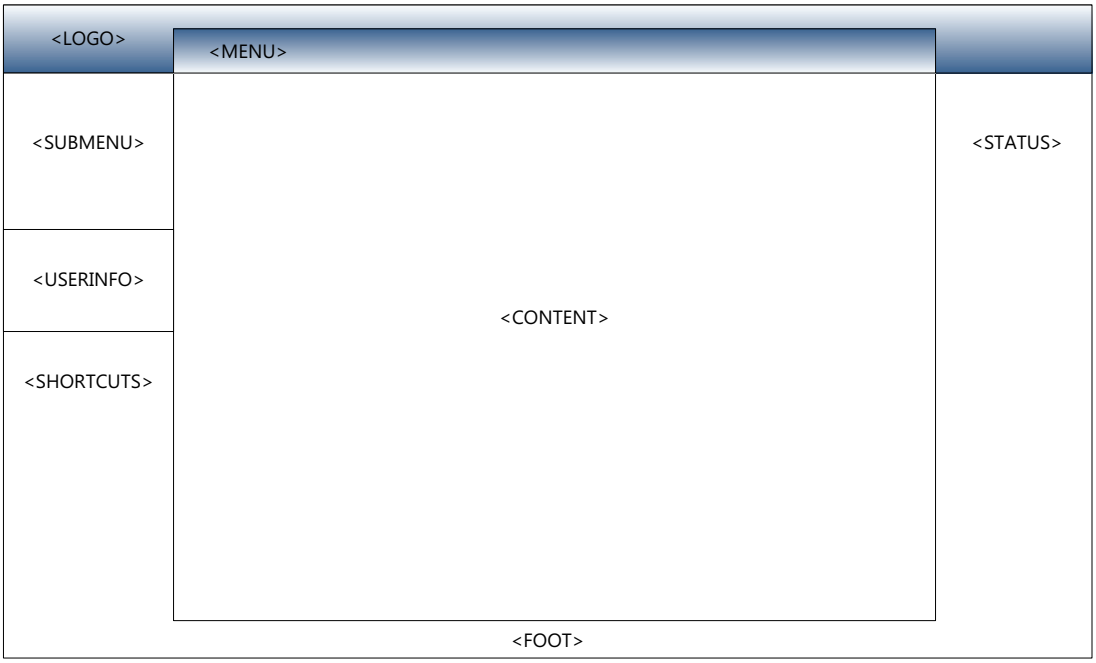

Abbildung 4.24: Entwurf der Benutzeroberfläche

[28] Die JavaScript Object Notation (JSON) ist ein mensch- und maschinenlesbares Datenaustauschformat, das in seiner Grundform einem assoziativen Array entspricht.

Jede Seite der Applikation hat ein einheitliches Layout. Die GUI ist in fünf Sektionen geteilt, die unterschiedliche Informationen enthalten:

- Die Kopfzeile enthält das Logo der Applikation sowie das übergeordnete Menu.
- Der linke Abschnitt enthält das untergeordnete Menu, die Anmeldeinformationen des Nutzers sowie einen Bereich, der Schnellzugriffe auf bestimmte Inhalte ermöglichen soll.
- Im Zentrum steht der eigentliche Inhalt der einzelnen Seiten.
- Der rechte Abschnitt kann zusätzliche Informationen enthalten, z.B. den Online-Status der Nutzer.
- Die Fußzeile enthält das Copyright sowie Verweise zur GFZ-Startseite und zur Dokumentation der Applikation.

Die Farbwahl der Kopfzeile und Rahmen orientierte sich am Corporate Design des GFZs und ist daher in Blau gehalten. Der Rest der Seite ist weiß. Als Schriftart wurde *Arial* gewählt, da sie serifenlos und somit gut lesbar ist. Außerdem ist sie standardmäßig auf allen Betriebssystemen installiert. Die Schriftfarbe ist abhängig von der Lokation und Funktion. Querverweise (Hyperlinks) werden vor dem jeweiligen Hintergrund entsprechend farbig hervorgehoben. Standardtexte werden in schwarz dargestellt.

Das Logo (siehe Abbildung 4.25) enthält den Namenszug der Applikation und soll durch den darunter liegenden Bogen eine Assoziation zum GFZ Logo (vgl. Abbildung 4.26) aufbauen.

Abbildung 4.25: Das Logo von SEMISYS

Abbildung 4.26: Das Logo des GFZ Potsdam

Die Länge des Seiteninhalts ist in vielen Fällen variabel, sodass häufig benötigte Elemente (Menu, Anmeldeinformationen, etc.) bei langen Seiten in der Regel verschwinden. Für eine gute Navigation wurden diese Elemente an einer bestimmten Position verankert, sodass sie permanent sichtbar sind.

5 Implementierung und Testphase

In den nächsten Abschnitten wird die Umsetzung des in Kapitel 4 vorgestellten Systementwurfs beschrieben. Das Ergebnis dieser Entwicklungsphase ist ein lauffähiger Prototyp des Sensormetainformationssystems.

5.1 Systemarchitektur

Nachfolgend sollen die verwendeten Produkte der zu entwickelnden Client/Server-Architektur beschrieben werden. Diese sind im Wesentlichen durch die in Abschnitt 3.2.3 festgelegten technischen Restriktionen seitens des GFZ Potsdam gekennzeichnet.

5.1.1 Basiskomponenten

Als Basiskomponenten standen das Betriebssystem Linux 2.6-64bit mit der Distribution openSUSE 11.2 sowie der Webserver Apache 2.2 (Linux/SUSE) zur Verfügung.

5.1.2 Datenbank

Für die Implementierung des Datenmodells wurde das freie, objektrelationale DBMS PostgreSQL in der Version 8.4 gewählt. Dieses ist Bestandteil jeder Linux-Distribution und standardmäßig auf allen Servern des GFZs installiert und konfiguriert. Aufgrund des großen Funktionsumfangs (vgl. Abschnitt 2.2.3) und der Tatsache, dass PostgreSQL bereits in vielen internen Projekten eingesetzt wurde, soll dieses DBMS auch in diesem Projekt das Kernstück bilden.

5.1.3 Programmier- und Skriptsprachen

Die zu entwickelnde Applikation soll webbasiert sein. Dementsprechend empfiehlt sich der Einsatz einer Skriptsprache, da diese vom Webserver interpretiert und zur Laufzeit ausgeführt werden kann. Eine Kompilierung der Quelltexte ist nicht erforderlich.

5.1.3.1 Serverseitige Skriptsprache

Zur Auswahl standen die quelloffenen und frei verfügbaren Skriptsprachen PHP 5.3 und Perl 5.12. Mit beiden Sprachen lassen sich die definierten Anforderungen und vorgestellten Module entwickeln, sodass sich anhand der Funktionalitäten das Pro und Contra für

oder gegen eine bestimmte Skriptsprache nicht ableiten lässt. Es gab dennoch objektive wie subjektive Gründe für die Verwendung von PHP, obwohl Perl in der Sektion 1.1 des GFZs vorrangig (aber v.a. für die Datenverarbeitung) eingesetzt wird. Die Gründe sind in Tabelle 5.1 zusammengefasst.

Pro PHP	**Contra Perl**
Leistungsfähige, einfache und bereits integrierte Schnittstellen für Netzwerkdienste (LDAP, Datenbanken, E-Mail)	Benötigte Module müssen nachinstalliert werden
Für Server-Side-Scripting entwickelt	Keine für Webapplikationen entwickelte Sprache
Geringere Serverbelastung als mit CGI-Modul, da keine neuen Prozesse erzeugt werden müssen	Höhere Serverbelastung bei Nutzung von Standard-Perl über CGI
Größere Erfahrung des Entwicklers	

Tabelle 5.1: Gründe für den Einsatz von PHP

5.1.3.2 Clientseitige Skriptsprache

Clientseitige Skriptsprachen werden benötigt, um die limitierten Möglichkeiten des statischen HTML zu umgehen und somit dynamische webbasierte Applikation zu erstellen. Es haben sich drei Sprachen in diesem Bereich etabliert: Tool command language (Tcl), Visual Basic Script (VBScript) und JavaScript (JS). Für Tcl wird ein zusätzlicher Interpreter benötigt, der für die gängigen Webbrowser nachinstalliert werden muss. VBScript wird derzeit nur vom Internet Explorer unterstützt. Die Anforderungen, dass keine Zusatzsoftware installiert werden und die Anwendung browserunabhängig sein soll, werden von diesen Skriptsprachen demnach nicht eingehalten. JavaScript wird hingegen von den meisten Browserherstellern unterstützt und bietet durch den Einsatz der AJAX-Technologie die Möglichkeit, die Anwendung für den Nutzer zu vereinfachen.

Das Problem ist allerdings, dass sich nicht alle Browser an die festgelegten Standards halten, sodass die Entwicklung von browserunabhängigen Entwicklung schwer fällt. Aus diesem Grund sind eine Vielzahl freier JS-Frameworks[29] entwickelt worden, die die Funktionalitäten kapseln und eine einheitliche Programmierschnittstelle bereitstellen. Für die Umsetzung dieses Projekts fiel die Entscheidung aus nachfolgenden Gründen auf die JS-Bibliothek jQuery:

[29] z.B. Prototype, MooTools, YUI, Script.aculo.us

- Kompaktheit (Version 1.7: 90kB),
- Vielzahl zusätzlicher Plugins[30],
- Schnelligkeit[31],
- permanente Weiterentwicklung durch wachsende Community,
- Marktführer[32] im Bereich der JS-Bibliotheken.

[30] siehe http://www.jqueryplugins.com/

[31] Die Schnelligkeitstests wurden auf http://www.domassistant.com/slickspeed/ für die Webbrowser Mozilla Firefox und Microsoft Internet Explorer durchgeführt.

[32] jQuery hat einen Marktanteil von 87.7 % (W3Techs, 2012, Stand: 13.07.2012)

5.2 Systemkomponenten

Das Basissystem hat aus Gründen der Übersichtlichkeit eine festgelegte Ordnerstruktur, die in Abbildung 5.1 dargestellt ist.

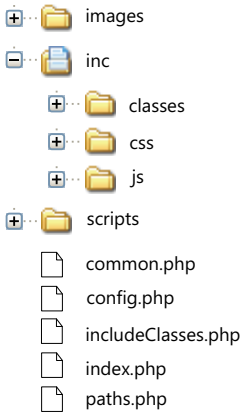

Abbildung 5.1: Verzeichnisstruktur des Basissystems

Die oberste Ebene besteht aus fünf Dateien, deren Funktionen nachfolgend näher beschrieben werden.

index.php Diese Datei enthält die Startseite des Projekts.

config.php In der Konfigurationsdatei werden wichtige und häufig benötigte globale Konstanten definiert. Dazu zählen die Parameter für die Verbindung zu der Datenbank und zum LDAP-Server sowie applikationsgebundene Konstanten (z.B. verwendete Systemzeit, Verweise zu externen Seiten, E-Mail des Administrators).

paths.php Die Datei dient dem automatischen Setzen von Projektpfaden, die für das Einbinden von Ressourcen in den einzelnen Skripten benötigt werden. Es werden zwei Konstanten benötigt:

- `PROJECT_DOCUMENT_ROOT`: Absoluter Verzeichnispfad zum Projekt auf dem Server.
- `PROJECT_HTTP_ROOT`: URL[33]-Adresse des Projekts.

includeClasses.php Die Datei bindet die einzelnen Klassen entsprechend des Systementwurfs (vgl. Abschnitt 4.3) in die Applikation ein.

[33] Uniform Resource Locators (URL) dienen der Identifikation von Ressourcen in Computernetzwerken.

common.php Die Datei enthält die zuvor beschriebenen Konfigurationsdateien. Zunächst muss die Datei `paths.php` eingebunden werden, um mit den Konstanten für die Beschreibung der Projektpfade arbeiten zu können. Anschließend werden die Dateien `config.php` und `includeClasses.php` inkludiert. Des Weiteren wird eine Verbindung zur Datenbank aufgebaut sowie eine Instanz der Klasse `SessionHandler()` erstellt, die für die Verwaltung von Sitzungen benötigt wird.

Im Ordner *images* werden alle Graphiken verwaltet, die für das System benötigt werden. In *inc* werden wiederverwendbare Skripte hinterlegt. Dazu zählen JS-Quelltext und JS-Bibliotheken (Ordner *js*), CSS-Dateien für die Beschreibung des Layouts (Ordner *css*) sowie die objektorientierten PHP-Klassen, die über `includeClasses.php` eingebunden werden (Ordner *classes*). Im Ordner *scripts* werden die einzelnen Seiten der Applikation entsprechend ihrer Kategorie gespeichert.

5.2.1 Authentifizierung

Für die Nutzung der Funktionalitäten von SEMISYS muss sich der Nutzer gegenüber dem System authentifizieren. Dies erfolgt über eine Login-Maske (siehe Abbildung 5.2), in die der Nutzer seinen Namen und das dazugehörige Passwort eintragen muss.

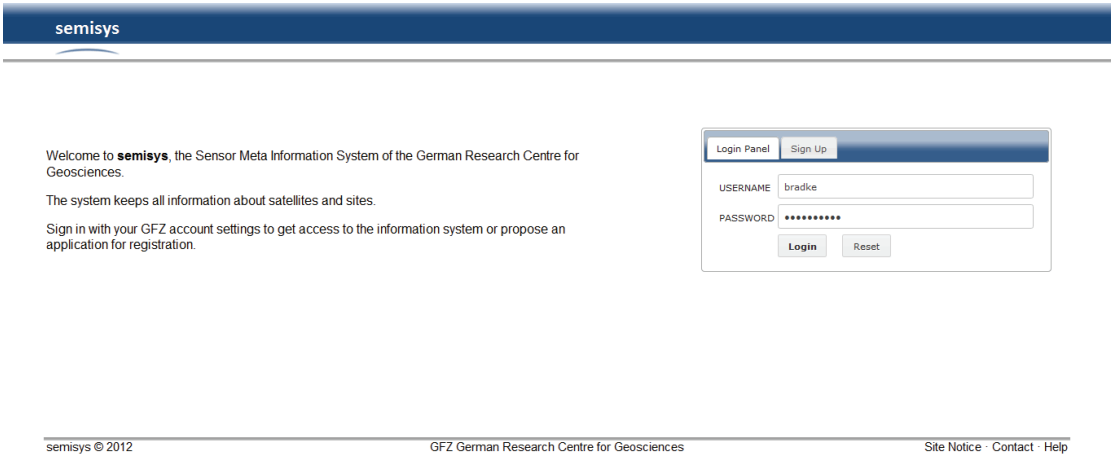

Abbildung 5.2: Maske für die Authentifizierung von Nutzern

Bei fehlerhaften Angaben (Nutzer existiert nicht, invalide Nutzer-Passwort-Kombination) erhält der Nutzer die Fehlermeldung, dass der Zugriff verweigert wird. Bei einer validen Nutzer-Passwort-Kombination wird der Nutzer in den internen Bereich der Applikation weitergeleitet.

5.2.2 Nutzerbereich

5.2.2.1 Verwaltung allgemeiner Metadaten

Unter dem Hauptmenupunkt *General* können allgemeine Metadaten zur Verwaltung von Organisationen und tektonischen Platten (siehe Abbildung 5.3) tabellenbasiert eingesehen und formularbasiert manipuliert werden. Ein Export der Metadaten in Form von CSV- oder PDF-Dateien ist ebenso möglich wie die stichwortbasierte Suche nach bestimmten Daten. Die Berechtigung zur Manipulation dieser Daten ist abhängig von der

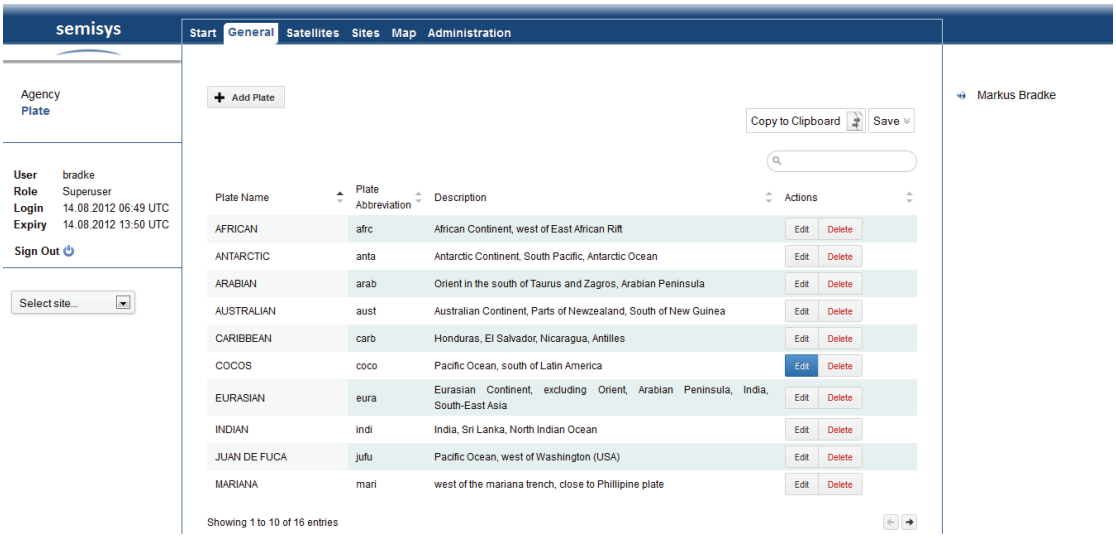

Abbildung 5.3: Verwaltung von tektonischen Platten

Zugehörigkeit des Nutzers zu einem bestimmten Projekt. Ist der Nutzer nicht Mitglied der entsprechenden Gruppe, sind die Aktionsschaltflächen zum Hinzufügen, Ändern und Löschen von Metadaten ausgeblendet.

Die nachfolgenden Ausführungen beschreiben die Implementierung der in Abschnitt 4.3.6.3 entworfenen Klassen und Funktionen. Bei Anklicken der Schaltfläche `Add` wird ein leeres Formular erzeugt (siehe Abbildung 5.4). Sobald der Nutzer in die Formularfelder klickt oder Text eingibt, werden die Werte geprüft und auftretende Fehler rot markiert und erklärt. Sofern keine Fehler vorliegen, erscheint die Schaltfläche zum Absenden der Informationen. Bevor bereits bestehende Datensätze manipuliert (editiert oder gelöscht) werden können, wird zunächst geprüft, ob der betroffene Datenbankeintrag bereits in Bearbeitung ist. Abbildung 5.5 zeigt die Antwort des Systems bei dem Versuch einen in Bearbeitung stehenden Datensatz zu löschen. Falls keine Sperre vorliegt, kann der Datensatz editiert oder gelöscht werden. Bei Anklicken der Schaltfläche `Edit` wird der Datensatz formularbasiert geladen und die verbleibende Zeit, die der Nutzer für die Bearbeitung des Eintrags hat, angezeigt (siehe Abbildung 5.6). Nach Ablauf der definierten Sperrzeit wird die Formularansicht verlassen.

Abbildung 5.4: Formularansicht für den initialen Import eines Datensatzes

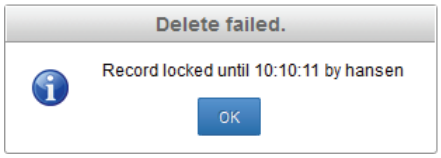

Abbildung 5.5: Systemnachricht bei vorliegender Sperre eines Datensatzes

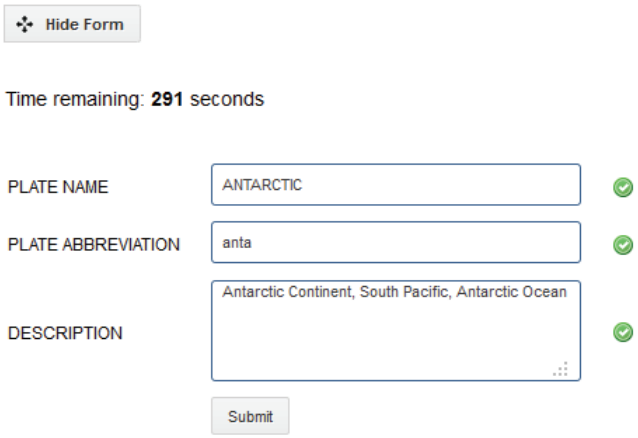

Abbildung 5.6: Formularansicht für die Bearbeitung eines bestehenden Datensatzes

5.2.2.2 Verwaltung von Satellitenmetadaten

Die Verwaltung von Satellitenmetadaten erfolgt analog der in Abschnitt 5.2.2.1 beschriebenen Vorgehensweise. Der Unterschied besteht lediglich in der verwendeten Datenbasis und in der Validierung der einzelnen Datentypen.

5.2.2.3 Verwaltung von Beobachtungsstationsmetadaten

Unter dem Menupunkt *Sites => Overview* befindet sich eine tabellarische Darstellung aller in der Datenbank registrierten Beobachtungsstationen mit grundlegenden Informationen (siehe Abbildung 5.7).

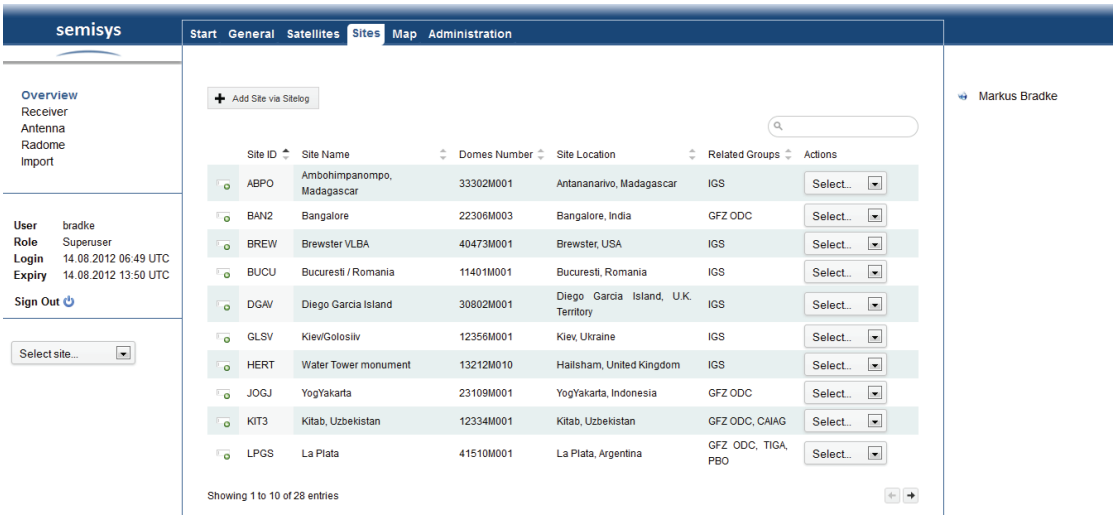

Abbildung 5.7: Tabellarische Darstellung vorhandener Beobachtungsstationen

In den rechten Spalten der Tabelle befinden sich jeweils Auswahllisten, in denen verschiedene Aktionen definiert sind:

- Weiterleitung zu einer stationsspezifischen Informationsseite (siehe Abbildung 5.8),
- Generierung IGS-konformer und valider Sitelogs aus den in der Datenbank hinterlegten Metainformationen,
- Deaktivieren (kein Löschen) der Station.

Neue und bereits vorhandene Stationen lassen sich derzeit nur über den Import der Sitelogs in die Datenbank einpflegen. Eine Verwaltung der Metainformationen über Formulare ist in einer späteren Projektphase vorgesehen.

Import von IGS Sitelogs IGS Sitelogs lassen sich unter dem Menupunkt *Sites => Import* in die Datenbank importieren (siehe Abbildung 5.9).

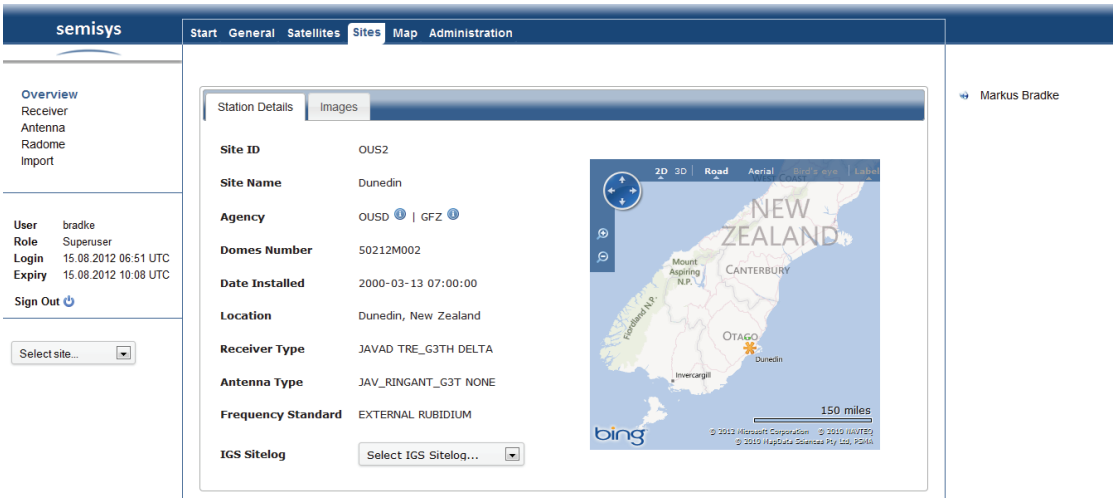

Abbildung 5.8: Stationsspezifische Informationsseite

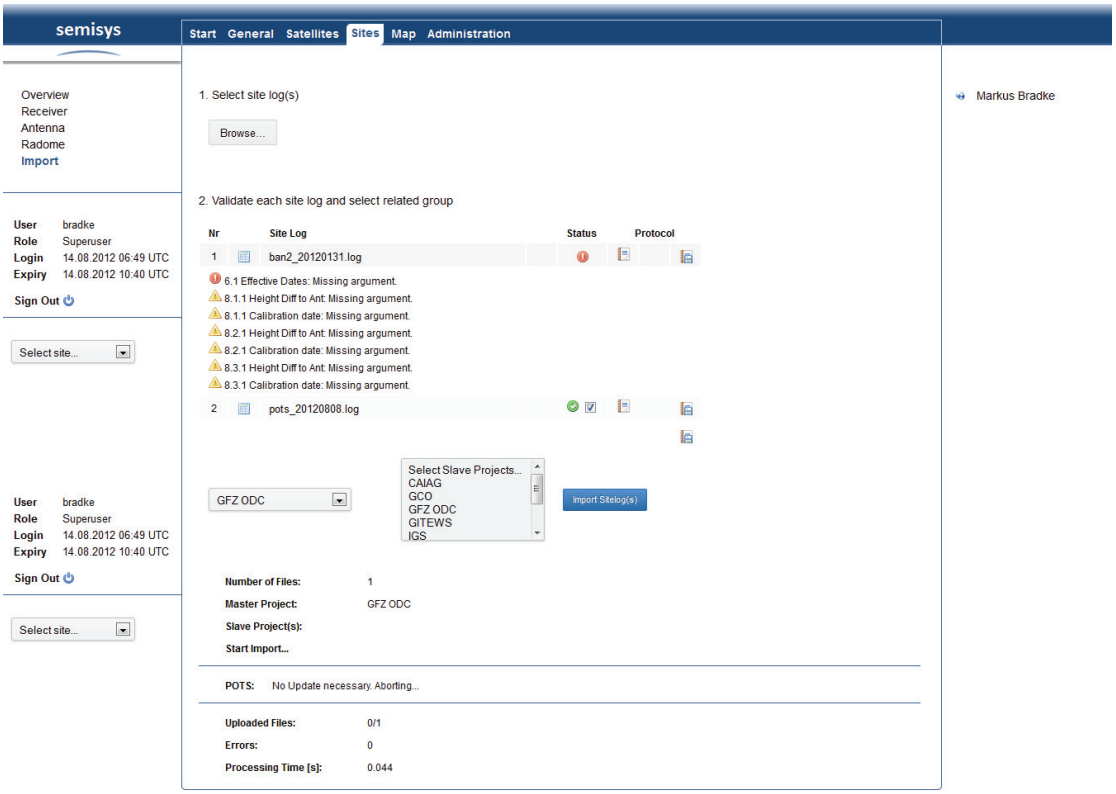

Abbildung 5.9: Maske für den Import von IGS Sitelogs

Hierfür wählt der Nutzer ein oder mehrere Dateien von seiner lokalen Festplatte aus und lädt diese auf den Webserver. Nach dem Upload werden die Dateien geparst und auf Validität geprüft. Das generierte Fehlerprotokoll lässt sich sowohl webbasiert als auch dateibasiert analysieren. Beim Upload mehrerer Dateien lässt sich darüber hinaus ein gemeinsames Fehlerprotokoll exportieren.

Sofern keine syntaktischen oder semantischen Fehler vorliegen, können die IGS Sitelogs nach der Auswahl des dazugehörigen Hauptprojekts in die Datenbank importiert werden. Der erfolgreiche Import hängt aber von weiteren Faktoren ab, die bereits in Abschnitt 4.3.6.1 erläutert wurden. In Tabelle 5.2 sind die möglichen Systemmeldungen und deren Bedeutung zusammengefasst.

Systemmeldung	**Beschreibung**	**Resultat**
Site is still related to another higher weighted project.	Die Beobachtungsstation gehört bereits zu einem höher eingestuften Projekt.	Kein Import
No Update necessary.	Die in der Datenbank hinterlegten Metainformationen sind identisch mit denen im IGS Sitelog.	Kein Import
Try to import. <ERROR MESSAGE>	Die Station ist noch nicht in der Datenbank vorhanden und soll importiert werden. Es trat aber ein unerwarteter Datenbankfehler auf.	Kein Import
Update necessary. Try to import. <ERROR MESSAGE>	Die Station ist bereits in der Datenbank vorhanden und soll importiert werden. Es trat aber ein unerwarteter Datenbankfehler auf.	Kein Import
Try to import. Import successful.	Die Station ist noch nicht in der Datenbank vorhanden und wird importiert.	Import
Update necessary. Try to import. Import successful.	Die Station ist bereits in der Datenbank vorhanden und wird aktualisiert.	Import

Tabelle 5.2: Mögliche Systemmeldungen bei Import von IGS Sitelogs

Import von Receiver- und Antenneninformationen Der Import der in den Dateien *rcvr_ant.tab* und *antenna.gra* enthaltenen Informationen erfolgt über die in Abbildung 5.10 dargestellte Maske.

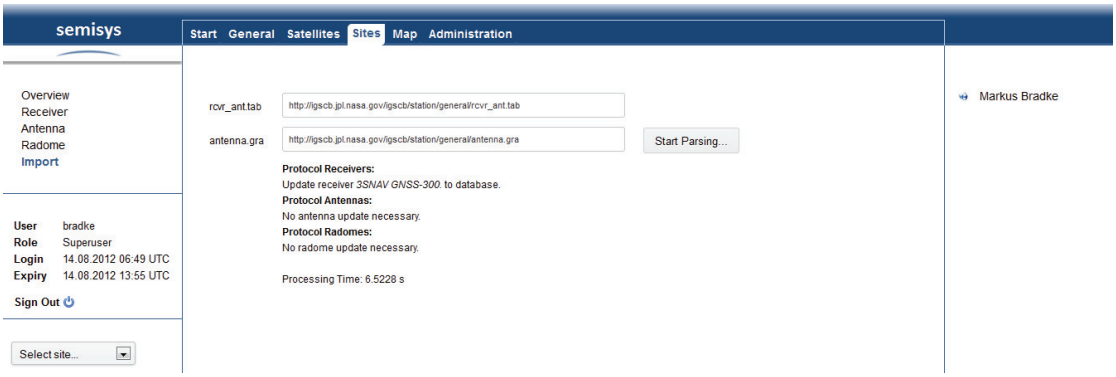

Abbildung 5.10: Maske für den Import von Hardware-Metainformationen

Es genügt lediglich ein Klick auf die Schaltfläche `Start Parsing...`, um das Parsen der Dateien sowie den Abgleich mit der Datenbank und eine (eventuelle) Aktualisierung zu initialisieren. Die Nutzer werden nach Beendigung des Vorgangs über den Status des Imports in Form eines Protokolls informiert. Die importierten Informationen lassen sich nach Receiver, Antenne und Radom unterteilt in den gleichnamigen Menuunterpunkten von *Sites* betrachten. Abbildung 5.11 zeigt einen Ausschnitt aus der importierten Antennentabelle des IGS.

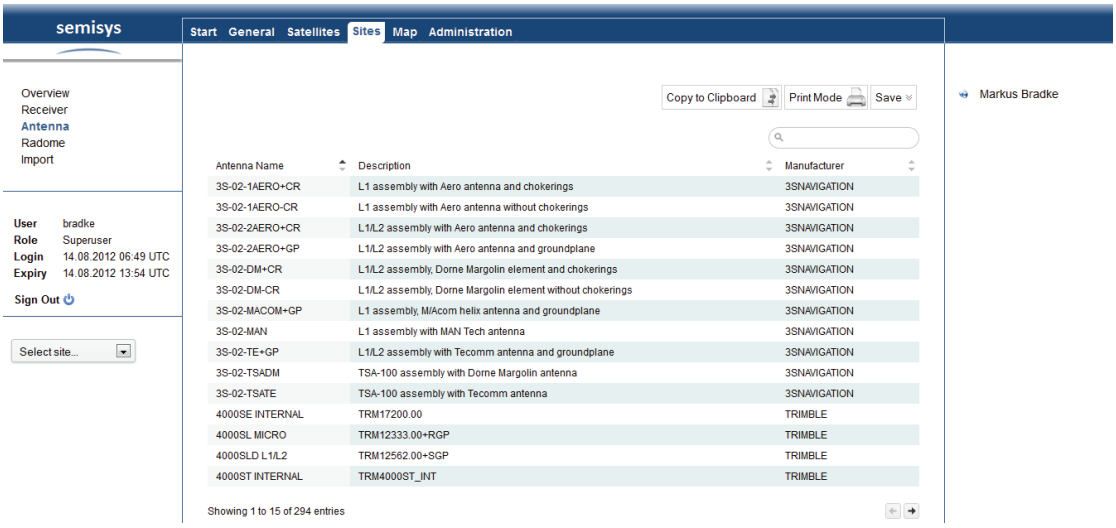

Abbildung 5.11: Tabellarische Darstellung der vom IGS standardisierten Antennentypen

5.2.2.4 Interaktive Kartendarstellung

Das Internet-GIS-Modul dient der visuellen Unterstützung der registrierten Beobachtungsstationen. Die Implementierung erfolgte auf Basis der API Mapstraction[34], die eine allgemeine Schnittstelle für eine Vielzahl von JS-Kartenbibliotheken bereitstellt, sodass der entwickelte Quellcode bei der Umstellung auf eine andere Karten-API wiederverwendet werden kann. In der aktuellen Version 2 werden folgende Kartenschnittstellen unterstützt:

- Google Maps,
- Open Street Map,
- Yahoo Maps,
- OpenLayers,
- Microsoft Bing Maps und
- CloudMade.

Als Kartenbasis wurde exemplarisch die API von Bing Maps genutzt (siehe Abbildung 5.12). Nach Anklicken des Menupunktes *Map* werden wichtige Metainformationen des kompletten Datenbestands via AJAX geladen und auf Grundlage der in der Datenbank hinterlegten Koordinaten graphisch in der Karte dargestellt. Bei wachsender Anzahl von Stationen wird die Übersichtlichkeit der Karte aber stark beeinflusst. Aus diesem Grund lässt sich der Datenbestand hinsichtlich folgender Kriterien filtern, die sich mit Hilfe von Auswahllisten selektieren lassen:

- Nach verantwortlicher Organisation,
- Nach untergeordnetem Projekt,
- Nach tektonischer Platte,
- Nach Receiverhersteller,
- Nach Antennenhersteller.

Innerhalb der Auswahllisten erfolgt die Verknüpfung der Werte durch ODER-Beziehungen, unterschiedliche Kriterien werden durch UND-Beziehungen miteinander in Verbindung gesetzt. Darüber hinaus lassen sich durch Klicken der Marker zusätzliche Informationen (Standort, installierte Hardware) über entsprechende Beobachtungsstationen ermitteln (siehe Abbildung 5.13). Die Marker enthalten zudem Verweise auf die Informationsseiten der Beobachtungsstation und unterstützen die Möglichkeit zum Export der Metainformationen in Form von IGS Sitelogs.

[34](MAPSTRACTION, 2012)

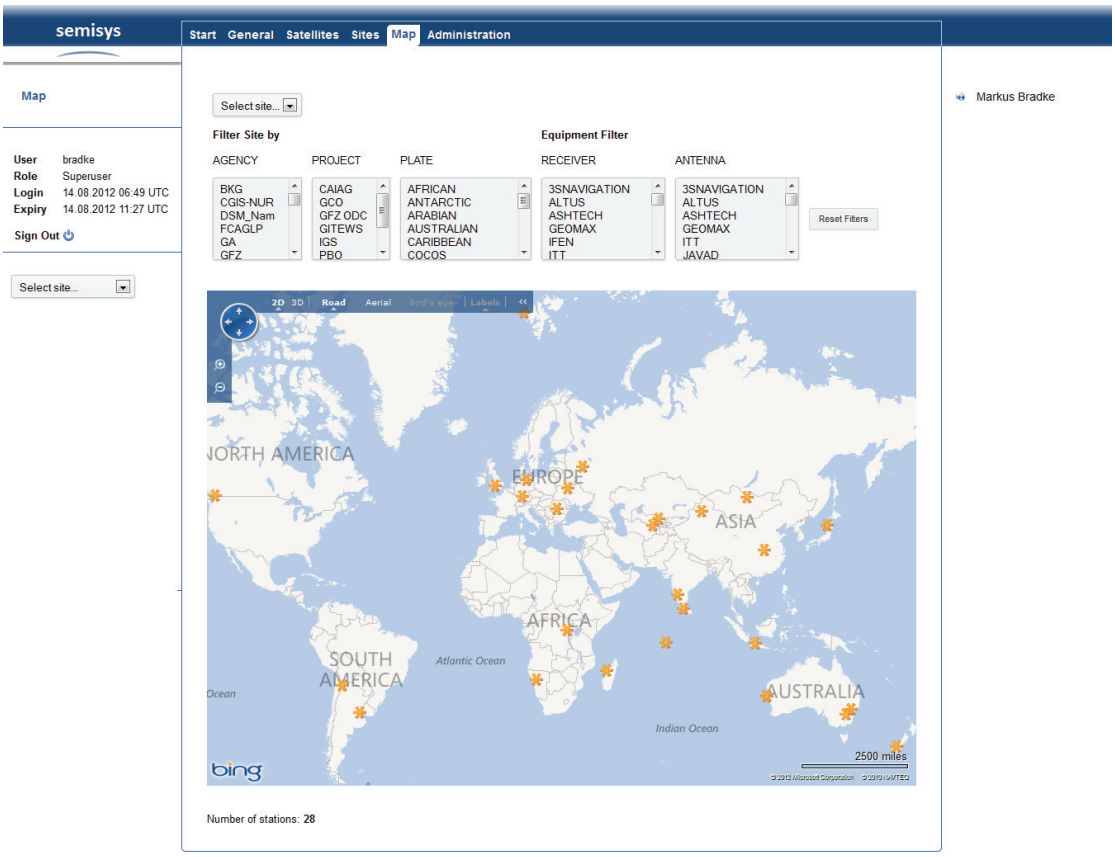

Abbildung 5.12: Interaktive Kartendarstellung der Beobachtungsstationen

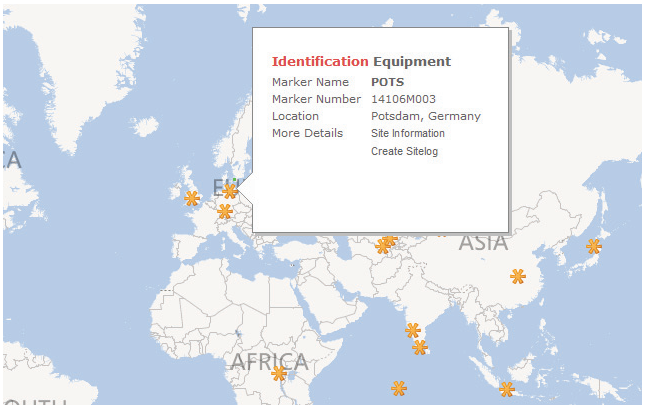

Abbildung 5.13: Informationsfeld in der Kartenanwendung

5.2.3 Administrativer Bereich

Der administrative Bereich ist für die Standardnutzer des Systems nicht sichtbar. Lediglich Administratoren haben Zugriff auf diesen Bereich der Applikation, um applikationsbezogene Einstellungen und Konfigurationen vorzunehmen.

5.2.3.1 Nutzerverwaltung

Über den Menuunterpunkt *User Management* werden die Benutzer des Systems verwaltet. Die einzelnen Benutzer werden mit den wichtigsten Informationen in einer Tabelle (siehe Abbildung 5.14) dargestellt. In der letzten Spalte befinden sich die Schaltflächen `Edit` und `Delete` für die Verwaltung bereits registrierter Nutzer. Neue Nutzer können über ein Formular, das nach Anklicken der Schaltfläche `Add User` dynamisch erzeugt wird, in die Datenbank aufgenommen werden.

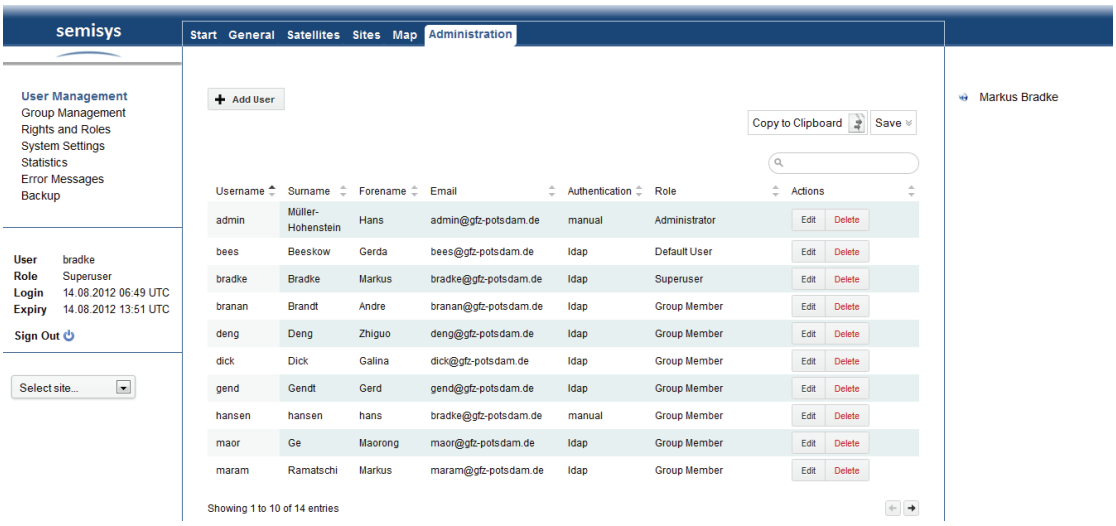

Abbildung 5.14: Benutzerübersicht

Für die Implementierung wurden die bereits in Abschnitt 4.3.6.3 beschriebenen Klassen `AutoForm` und `Locks` genutzt.

5.2.3.2 Rollen- und Rechteverwaltung

Im Menuunterpunkt *Rights and Roles* (siehe Abbildung 5.15) werden die Rechte für den Zugriff auf die einzelnen Seiten der Applikation gesteuert. Durch Setzen der Kontrollkästchen wird das Zugriffsrecht für jeden Menuunterpunkt bestätigt. Die Einstellungen werden nach dem Klick auf die Schaltfläche `Update` übernommen. Die erfolgreiche Aktualisierung wird durch das grüne Häkchen angezeigt. Unerwartete Fehler werden durch ein rotes Ausrufezeichen gekennzeichnet.

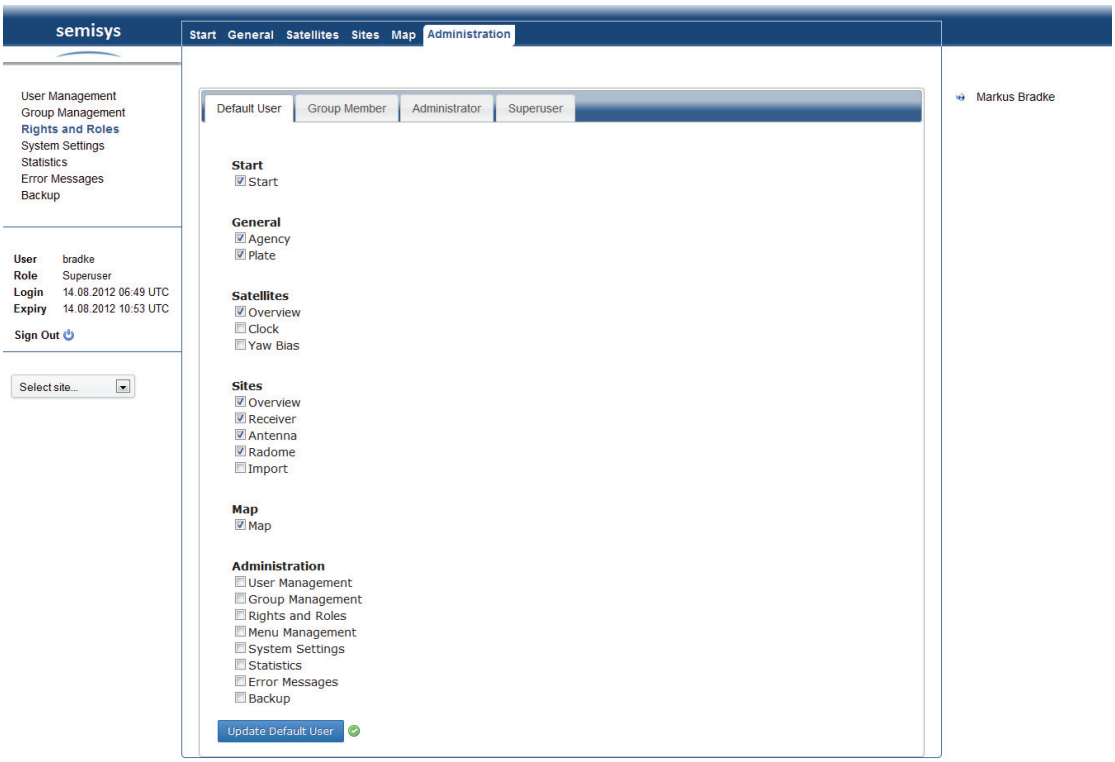

Abbildung 5.15: Rollen- und Rechtevergabe

5.2.3.3 Gruppenverwaltung

Über die zuvor beschriebene Rollenzugehörigkeit wird lediglich der Zugriff auf den Inhalt einer Internetseite erteilt. Die Berechtigung für den schreibenden Zugriff auf Datensätze wird in der in Abbildung 5.16 dargestellten Tabelle über die Projektzugehörigkeit der einzelnen Nutzer erteilt. Für die Bearbeitung dieser Tabelle sind keine zusätzlichen Masken notwendig, da die Einträge direkt in der Tabelle editiert werden können. Ein Doppelklick in die Spalte `Users` genügt, um Nutzer entsprechend hinzuzufügen oder zu entfernen. Die Eingabe wird durch die Eingabetaste bestätigt. Die Werte werden auf Validität geprüft.

5.2.3.4 Systemverwaltung

Unter *System Settings* (siehe Abbildung 5.17) können die globalen Systemparameter der Anwendung angepasst werden. Dazu zählen Einstellungen über die Dauer einer Sitzung, die Dauer einer Datensatzsperre sowie die maximale Dateigröße beim Upload von IGS Sitelogs. Darüber hinaus kann das komplette System deaktiviert werden, falls größere Wartungsarbeiten durchgeführt werden müssen. In diesem Fall erhält der Nutzer auf allen Seiten des Systems eine Statusnachricht, die nach Bedarf angepasst werden kann.

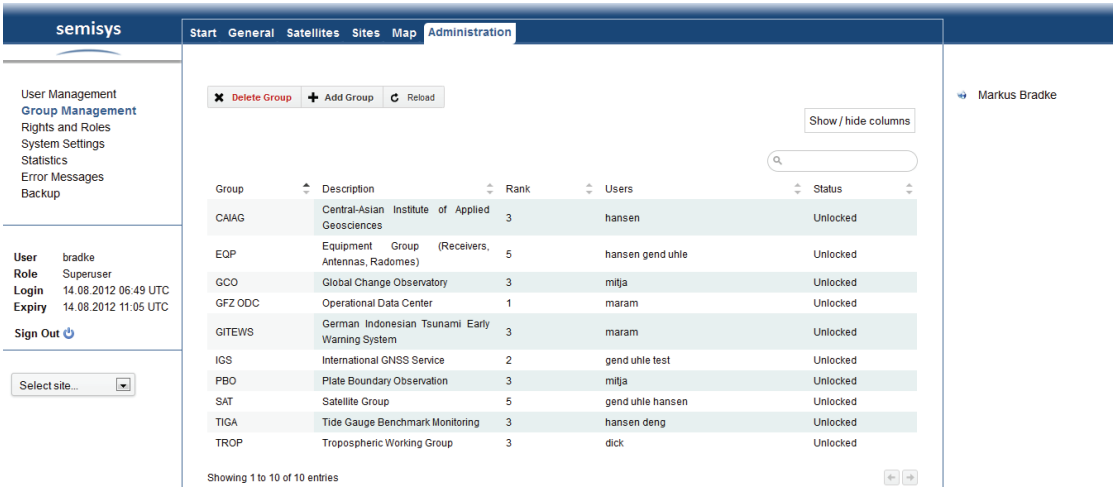

Abbildung 5.16: Gruppenverwaltung

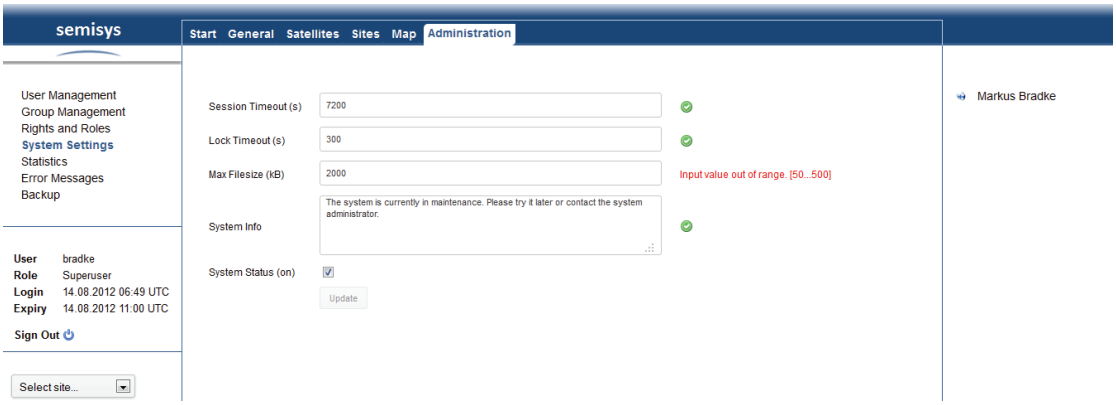

Abbildung 5.17: Systemeinstellungen

5.2.3.5 Fehlermeldungen

Alle auftretenden skript- und datenbankseitigen Fehler werden in der Datenbank protokolliert (vgl. Abschnitt 4.3.3). Unter *Error Messages* werden die registrierten Fehler tabellenbasiert und nach Datum sortiert ausgegeben. Eine Suche nach bestimmten Fehlern ist ebenso möglich.

5.2.3.6 Datensicherung

In Abschnitt 3.2.2 wurde definiert, dass die Datenbank aufgrund der Relevanz der enthaltenen Informationen täglich gesichert werden soll. Dafür wurde ein csh-Skript implementiert, welches im Wesentlichen den PostgreSQL-Befehl zum Sichern der Datenbank (`pg_dump`) enthält. Dieses Skript muss lediglich in die Tabelle zur Verwaltung der Jobs (Crontab, siehe Listing 5.1) eingetragen werden, um durch den Cron-Daemon zu einem definierten Zeitpunkt ausgeführt zu werden. Da sich das Datenmodell in der Regel

```
#M   H   D   M   W    Befehl
 0   3   *   *   *    /srv/www/htdocs/semisys/pg_dump.csh
```

Listing 5.1: Ausschnitt aus der Crontab, Sicherung jeden Tag um 03:00 Uhr

nicht ändert, wird einmalig die Schemadefinition der Datenbank gespeichert. Im weiteren Verlauf werden lediglich die enthaltenen Daten gesichert. Neben der automatischen Sicherung der Datenbank wurde zusätzlich ein webbasierter Ansatz (siehe Abbildung 5.18) implementiert. Durch Angabe eines Dateinamens sowie ausgewählter Parameter für den `pg_dump`-Befehl kann die Datenbank manuell in Form eines SQL-Skripts gesichert werden.

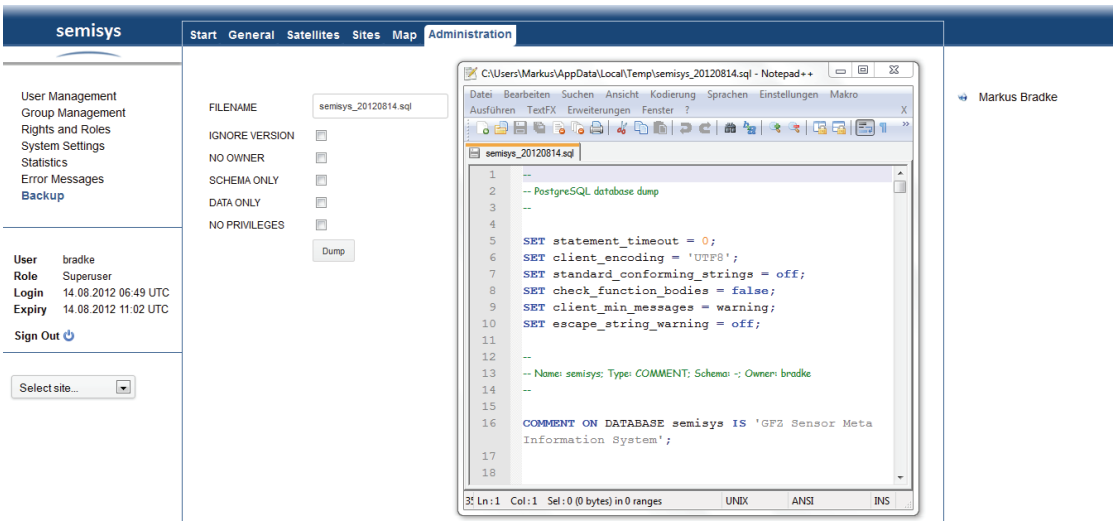

Abbildung 5.18: Webbasierte Sicherung der Datenbank

5.3 Testphase

Spätestens nach Abschluss jeder Software-Entwicklung ist es notwendig, die implementierten Module und Funktionalitäten auf die Umsetzung und Einhaltung der definierten funktionalen und nichtfunktionalen Anforderungen (vgl. Abschnitt 3.2) zu prüfen. Alle entwickelten Systemkomponenten und Module wurden im Verlauf der Implementierung permanenten Tests unterzogen. Durch die Protokollierung skript- und datenbankseitiger Fehler konnten diese bei der Prüfung der implementierten Systemfunktionen on-the-fly erkannt werden, sodass syntaktische Fehler im Quellcode oder in den SQL-Abfragen sofort behoben werden konnten. Semantische Fehler konnten durch verschiedene Testszenarien aufgedeckt werden.

5.3.1 Komponententest

Die gesamte Applikation ist modular aufgebaut, sodass sich die implementierten Module einzeln auf ihre technische Lauffähigkeit testen lassen. Alle entwickelten Module und enthaltenen Methoden funktionieren den Spezifikationen entsprechend und liefern korrekte und erwartete Ergebnisse.

5.3.2 Systemtest

In den Systemtests wird geprüft, ob die funktionalen und nichtfunktionalen Anforderungen an das System qualitativ und quantitativ erfüllt wurden. In den Tabellen 5.3 und 5.4 sind die Umsetzungen der definierten Anforderungen zusammengefasst dargestellt.
Auf Seite der funktionalen Anforderungen sind nahezu alle Kriterien erfüllt worden. Lediglich das Rückgängigmachen von Änderungen ist bisher nicht implementiert. Diese Funktion ist aber bereits im Datenmodell berücksichtigt worden. Die Überprüfung der Einhaltung aller nichtfunktionalen Anforderungen fiel hingegen schwerer. Insbesondere die Verfügbarkeit der Applikation lässt sich schwer überwachen. Es ist ohnehin unmöglich eine Hochverfügbarkeit mit lediglich einem Web- und Datenbankserver zu gewährleisten, sodass der nächste Schritt das Aufsetzen von Replikationsservern umfasst. Die Bedienbarkeit der Software ist ein subjektives Kriterium, das von verschiedenen Nutzern unterschiedlich bewertet werden kann. Die einheitliche und klar strukturierte GUI sowie die intuitive Bedienung der Applikation wurden aber durchweg positiv bewertet. Die Unabhängigkeit von verschiedenen Browserherstellern bei der Nutzung von JavaScript wurde durch die Nutzung der JS-Bibliothek jQuery erreicht. Die Programmfunktionen wurden für nachfolgende Browser überprüft:

- Microsoft Internet Explorer ab Version 8.x,
- Mozilla Firefox ab Version 10.x,
- Konqueror ab Version 4.x.

Anforderungen	Status
Zentrale Speicherung aller Metainformationen in einer Datenbank	⊕⊕
Lokale oder LDAP-basierte Authentifizierung von Nutzern	⊕⊕
Datenbankbasierte Sitzungsverwaltung	⊕⊕
Zweistufige Rechteverwaltung	⊕⊕
Protokollierung von Änderungen	⊕⊕
Rückgängigmachen von Änderungen	⊕⊖
Protokollierung skript- und datenbankseitiger Fehler	⊕⊕
Sperren von Datensätzen	⊕⊕
Import von IGS Sitelogs, rcvr_ant.tab und antenna.gra	⊕⊕
Validierung von Formularfeldern	⊕⊕
Export von Stationsmetadaten als IGS Sitelogs	⊕⊕
Export einzelner Tabellen als CSV oder PDF	⊕⊕
(Gefilterte) Darstellung der Beobachtungsstationen in einer Kartenanwendung	⊕⊕

Tabelle 5.3: Tabellarische Darstellung implementierter funktionaler Anforderungen
⊕⊕ Anforderung komplett erfüllt
⊕⊖ Anforderung teilweise erfüllt

Anforderungen	**Status**
Zuverlässigkeit	⊕⊕
Verfügbarkeit	⊕⊖
Datensicherung	⊕⊕
Bedienbarkeit	⊕⊕
Browserunabhängigkeit	⊕⊕
Effizienz	⊙⊙
Skalierbarkeit	⊕⊕

Tabelle 5.4: Tabellarische Darstellung implementierter nichtfunktionaler Anforderungen
⊕⊕ Anforderung komplett erfüllt
⊕⊖ Anforderung teilweise erfüllt
⊙⊙ Anforderung nicht getestet

Die Effizienz der Software wurde nicht getestet, da aus Zeitgründen nur eine bestimmte Anzahl von Stationen in die Datenbank importiert werden konnte. Die Skalierbarkeit der Datenbank ist hingegen unbegrenzt, da der zur Verfügung stehende Speicher beliebig erweitert werden kann.

6 Zusammenfassung und Ausblick

In der vorliegenden Masterarbeit wurde ein Konzept und Entwurf zur Entwicklung eines web- und datenbankbasierten Systems für die Verwaltung von GNSS-Sensormetadaten aufgezeigt. Das System ist modular aufgebaut und umfasst Basiskomponenten für die Authentifizierung von Nutzern, die Verwaltung von Sitzungen, die Behandlung skript- und datenbankseitiger Fehler, die dynamische Generierung von Tabellen und Formularen, das Sperren von Datensätzen sowie das Parsen von IGS spezifischen Dateien und deren Validierung. Darüber hinaus wurde ein internetbasiertes Auskunftssystem zur graphischen Darstellung der Beobachtungsstationen auf Basis eines Online-Kartendienstes implementiert, das die Suche nach Stationen durch die geographische Orientierung vereinfachen soll. Als Resultat geht ein lauffähiger auf Client/Server-Technologie basierender Prototyp aus diesem Projekt hervor.

Dem GFZ Potsdam steht mit der entwickelten Applikation ein vollwertiges Werkzeug zur Verwaltung von Beobachtungsstations- und Satellitenmetadaten zur Verfügung. Das System kann durch den strukturierten und objektorientierten Ansatz der Basisklassen leicht um zusätzliche Funktionen erweitert werden und bildet die Basis für zukünftige Entwicklungen bezüglich der Archivierung und Validierung der eingehenden Beobachtungsdaten verschiedener Stationen.

Weitere Entwicklungen werden zum einen die Implementierung zusätzlicher Funktionen in SEMISYS und zum anderen die Konsolidierung der bestehenden Routine-Software des Datenzentrums betreffen. SEMISYS wird in einer anstehenden Projektphase um nachfolgende Funktionen erweitert:

- Formularbasierte Änderungen von IGS Sitelogs,
- Rückgängigmachen von Änderungen (Versionierung),
- Automatische Generierung von E-Mails für Interessengruppen bei Änderungen am Datenbestand,
- Generierung historischer IGS Sitelogs auf Grundlage der gespeicherten Revision.

Die in der Datenbank enthaltenen Stationsmetainformationen können für die Weiterentwicklung der Routine-Software des operationellen Datenzentrums am GFZ Potsdam

genutzt werden. Zum einen können die Daten für die automatische Generierung von RINEX-Headern der vom GFZ betriebenen Beobachtungsstationen genutzt werden, um somit Inkonsistenzen zwischen RINEX-Headern und IGS Sitelogs zu vermeiden. Zum anderen kann durch die Validierung eingehender Beobachtungsdaten und dem Abgleich mit den in der Datenbank hinterlegten Metainformationen verhindert werden, dass fehlerhafte Beobachtungsdaten externer Stationen archiviert werden und zu Fehlern in der Analyse dieser Daten führen.

Literaturverzeichnis

[Aden et al. 2007] Aden, Christian ; Schmidt, Gunther ; Schröder, Winfried: Ein web-basiertes Geo-Informationssystem für das Monitoring gentechnisch veränderter Organismen. In: *eZAI, Gesellschaft für Informatik in der Land-, Forst- und Ernährungswirtschaft e.V. (GIL)* (2007)

[Atkinson et al. 1989] Atkinson, Malcolm ; Bancilhon, Francois ; DeWitt, David ; Dittrich, Klaus ; Maier, David ; Zdonik, Stanley: The Object-Oriented Database System Manifesto / University of Glasgow. 1989. – Forschungsbericht

[Bartelme 2005] Bartelme, Norbert: *Geoinformatik - Modelle, Strukturen, Funktionen*. 4. Auflage. Berlin : Springer Verlag, 2005. – ISBN 3-540-20254-4

[Bill 2003] Bill, Ralf: GIS und Internet / Universität Rostock. 2003. – Vorlesung

[Bill und Fritsch 1994] Bill, Ralf ; Fritsch, Dieter: *Grundlagen der Geoinformationssysteme - Hardware, Software und Daten*. Bd. 1. 2. Auflage. Heidelberg : Herbert Wichmann Verlag, 1994. – ISBN 3-87907-265-5

[Bradke 2009] Bradke, Markus: *Speicherung raum-zeitlicher Daten und deren Visualisierung*, Hochschule Neubrandenburg, Bachelorarbeit, 2009

[Brinkhoff 2008] Brinkhoff, Thomas: *Geodatenbanksysteme in Theorie und Praxis - Eine Einführung in objektrelationale Geodatenbanken unter besonderer Berücksichtigung von Oracle Spatial*. 2. Auflage. Heidelberg : Herbert Wichmann Verlag, 2008. – ISBN 978-3-87907-472-3

[BSI 2012a] BSI: *M 6.35 Festlegung der Verfahrensweise für die Datensicherung*. 2012. – URL `https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/m/m06/m06035.htmlp`. – Zugriffsdatum: 29.05.2012. – Website

[BSI 2012b] BSI: *M 6.49 Datensicherung einer Datenbank*. 2012. – URL `https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/m/m06/m06049.html`. – Zugriffsdatum: 29.05.2012. – Website

[Codd 1970] Codd, Edgar F.: A Relational Model of Data for Large Shared Data Banks. In: *Communications of the ACM* 13 (1970), S. 377–387

[Gourtner und Estey 2012] Gourtner, Werner ; Estey, Lou: *RINEX - The Receiver Independent Exchange Format - Version 3.01.* 2012. – URL `http://igscb.jpl.nasa.gov/igscb/data/format/rinex301.pdf`. – Zugriffsdatum: 15.08.2012. – Formatbeschreibung

[Hofmann-Wellenhof et al. 1993] Hofmann-Wellenhof, Bernhard ; Lichtenegger, Herbert ; Collins, James: *GPS - Theory and Practice.* Second Edition. Wien New York : Springer Verlag, 1993. – ISBN 3-211-82477-4

[Hofmann-Wellenhof et al. 2008] Hofmann-Wellenhof, Bernhard ; Lichtenegger, Herbert ; Wasle, Elmar: *GNSS - Global Navigation Satellite Systems: GPS, GLONASS, Galileo and more.* Wien : Springer Verlag, 2008. – ISBN 978-3-211-73012-6

[IGSCB 2012a] IGSCB: *Antenna Graphics File.* 2012. – URL `http://igscb.jpl.nasa.gov/igscb/station/general/antenna.gra`. – Zugriffsdatum: 15.06.2012. – Dokument

[IGSCB 2012b] IGSCB: *Instructions for filling out IGS site logs.* 2012. – URL `http://igscb.jpl.nasa.gov/igscb/station/general/sitelog_instr.txt`. – Zugriffsdatum: 15.06.2012. – Website

[IGSCB 2012c] IGSCB: *Receiver/Antenna Table.* 2012. – URL `http://igscb.jpl.nasa.gov/igscb/station/general/rcvr_ant.tab`. – Zugriffsdatum: 07.05.2012. – Dokument

[IGSCB 2012d] IGSCB: *Site Information Form.* 2012. – URL `http://igscb.jpl.nasa.gov/igscb/station/general/blank.log`. – Zugriffsdatum: 15.06.2012. – Formular

[Initiative 2012] Initiative, Open S.: *The PostgreSQL Licence (PostgreSQL).* 2012. – URL `http://www.opensource.org/licenses/postgresql`. – Zugriffsdatum: 23.05.2012. – Website

[Kemper und Eickler 2011] Kemper, Alfons ; Eickler, André: *Datenbanksysteme - Eine Einführung.* 8. Auflage. München : Oldenbourg Verlag, 2011. – ISBN 978-3-486-59834-6

[de Lange 2002] Lange, Norbert de: *Geoinformatik in Theorie und Praxis.* Heidelberg, Berlin, New York : Springer Verlag, 2002. – ISBN 3-540-43286-8

[MAPSTRACTION 2012] MAPSTRACTION: *Javascript mapping abstraction library.* 2012. – URL `http://mapstraction.com/`. – Zugriffsdatum: 17.07.2012. – Website

[Peng und Tsou 2003] Peng, Zhong-Ren ; Tsou, Ming-Hsiang: *Internet GIS - distributed geographic information services for the internet and wireless networks.* Hoboken (NJ) : John Wiley & Sons, Inc., 2003. – ISBN 0-471-35923-8

[PostgreSQL 2012a] PostgreSQL: *PostgreSQL 8.4 Documentation - 1.2. Architectural Fundamentals.* 2012. – URL `http://www.postgresql.org/docs/8.4/static/tutorial-arch.html`. – Zugriffsdatum: 23.05.2012. – Website

[PostgreSQL 2012b] PostgreSQL: *PostgreSQL 8.4 Documentation - What is PostgreSQL?* 2012. – URL `http://www.postgresql.org/docs/8.4/static/intro-whatis.html`. – Zugriffsdatum: 23.05.2012. – Website

[Scherbaum 2012] Scherbaum, Andreas: *PostgreSQL - das freie objektrelationale Open Source Datenbanksystem.* 2012. – URL `http://www.postgresql.de/`. – Zugriffsdatum: 13.06.2012. – Website

[Seeber 2003] Seeber, Günter: *Satellite Geodesy.* 2nd Edition. Berlin : de Gruyter, 2003. – ISBN 3-11-017549-5

[W3Techs 2012] W3Techs: *Usage of JavaScript libraries for websites.* 2012. – URL `http://w3techs.com/technologies/overview/javascript_library/all`. – Zugriffsdatum: 13.07.2012. – Website

[Zogg 2011] Zogg, Jean-Marie: *GPS und GNSS: Grundlagen der Ortung und Navigation mit Satelliten.* 2011

Abbildungsverzeichnis

Tabellenverzeichnis

Eidesstattliche Erklärung

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

Potsdam, den 21.08.2012

Markus Bradke

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während der Erstellung dieser Masterarbeit unterstützt haben. Herrn Prof. Dr.-Ing. Andreas Wehrenpfennig möchte ich für die Betreuung dieser Arbeit und die kritische Auseinandersetzung mit der Thematik während der Bearbeitungszeit danken. Herrn Dr. Jens Wickert danke ich für die Übernahme des Korreferats.

Weiterhin möchte ich den Kollegen und Mitarbeitern der Sektion 1.1 des GFZ Potsdam für die zahlreichen konstruktiven Diskussionen, Hinweise und Vorschläge danken. Ein besonderer Dank gilt Herrn Thomas Nischan, der das grundlegende Thema dieser Arbeit vorgeschlagen und mich mit vielen beantworteten Fragen zu dieser Thematik unterstützt hat. Darüber hinaus möchte ich Herrn Dr. Markus Ramatschi und Herrn PD Dr.-Ing. Jürgen Klotz danken, die sich für das Korrekturlesen der Arbeit Zeit genommen haben.

Nicht zuletzt möchte ich mich bei meiner Familie für die langjährige moralische und finanzielle Unterstützung bedanken.

Potsdam, im August 2012

Anhang

Der Anhang befindet sich auf der beigefügten CD-R mit folgendem Inhalt:

1. Masterarbeit als PDF
2. Quellcodes der implementierten Applikation
3. Datenbank-Backup des implementierten Datenmodells als SQL-Skript