Projektbericht

iDisplays Webservice

Final Version

Dokumentname	iDisplays Webservice – Projektbericht	Erstellt am	20.08.2010
Autoren des Dokuments	Arne de Wall, Christian Autermann, Raimund Schnürer, Tobias Tresselt		
Seitenanzahl	21		

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

An der Universität Münster sieht man seit einiger Zeit an mehreren Standorten Infobildschirme, die sogenannten iDisplays. Bei einem Studienprojekt entstand die Idee, auf diesen Bildschirmen relevante Informationen für die Studierenden anzeigen zu lassen. Dazu gehören zum Beispiel Neuigkeiten aus dem Institutsumfeld, der Busfahrplan der nächstgelegenen Haltestelle, der Speiseplan der nächsten Mensa, das aktuelle Wetter oder ein Newsticker mit Nachrichten aus aller Welt.

Da jedoch jedes Display über die auf dem angeschlossenen Rechner aufgespielte Software einzeln die notwendigen Daten abruft, sind Änderungen an der Funktionsweise nur mühsam zu bewerkstelligen, weil jedes Display einzeln aktualisiert werden muss. Deshalb soll nun die Softwarearchitektur der iDisplays auf ein Webservice-basiertes Modell umgestellt werden. Dabei sollen die Infobildschirme nur noch als Clients die Daten von einem zentralen Verteilerserver anfordern und in einem einheitlich strukturierten Format erhalten. Dadurch bleibt nur noch die Visualisierung der erhaltenen Informationen auf der Clientseite bestehen und die Hauptarbeit wird auf den Server ausgelagert. Änderungen in den Datenakquisitionsmethoden, beispielsweise bei veränderten Quellstrukturen, können so zentral durchgeführt werden und die Software auf den iDisplays muss nicht angepasst werden.

Zu diesem Zweck haben Oliver Paczkowski und Markus Buzeck, die Begründer der Firma infowerk, ein Studienprojekt zur Erstellung dieses Webservices zum Austausch von Informationen zwischen iDisplays-Server und iDisplays-Clients ausgeschrieben. Die Aufgabe umfasst die Planung des Projekts, die Recherche möglicher Konzepte und Technologien, die Modellierung der einzelnen Teile des Webservice, die Implementierung eines prototypischen Systems sowie die Dokumentation der erstellten Komponenten. Dabei sollen bestehende Standards, wie die der Geo-Webservices des OGC, berücksichtigt werden.

Für die Zukunft verspricht sich die Firma Infowerk jedoch noch weitere Zentralisierungen in Form einer Umgestaltung des Webservices vom reinen Informationsaustausch hin zu einer Video-Streaming Lösung.

1.2 Auftraggeber und Projektmitarbeiter

1.2.1 Auftraggeber

Markus Buzeck & Oliver Paczkowski
melezo
Weseler Straße 253a 48151 Münster

1.2.2 Projektmitarbeiter

Name	Studiengang
Christian Autermann	B.Sc. Geoinformatik
Raimund Schnürer	B.Sc. Geoinformatik
Tobias Tresselt	B.Sc. Geoinformatik
Arne de Wall	B.Sc. Geoinformatik

1.3 Zeitplan

1.3.1 Arbeitszeiteinteilung

12. April – 25. Juli: 15 Wochen à durchschnittlich 5 Stunden = 75 Stunden
26. Juli – 15. August: 3 Wochen à durchschnittlich 25 Stunden = 75 Stunden

Insgesamt 150 Stunden (= 5CP)

1.3.2 Meilensteine

13.06.2010: Ende Modellierung
25.07.2010: Ende Spezifikation der Architektur und Vorbereitung der Implementierung
15.08.2010: Ende Durchführung der Implementierung und des Projekts
19.08.2010: Abschlusspräsentation und Übergabe des Projekts

1.3.3 Meetings

15.04.2010: Kick-Off Meeting
22.04.2010: zweites Treffen
06.05.2010: drittes Treffen
15.06.2010: viertes Treffen
15.07.2010: fünftes Treffen
19.08.2010: Abschlusspräsentation

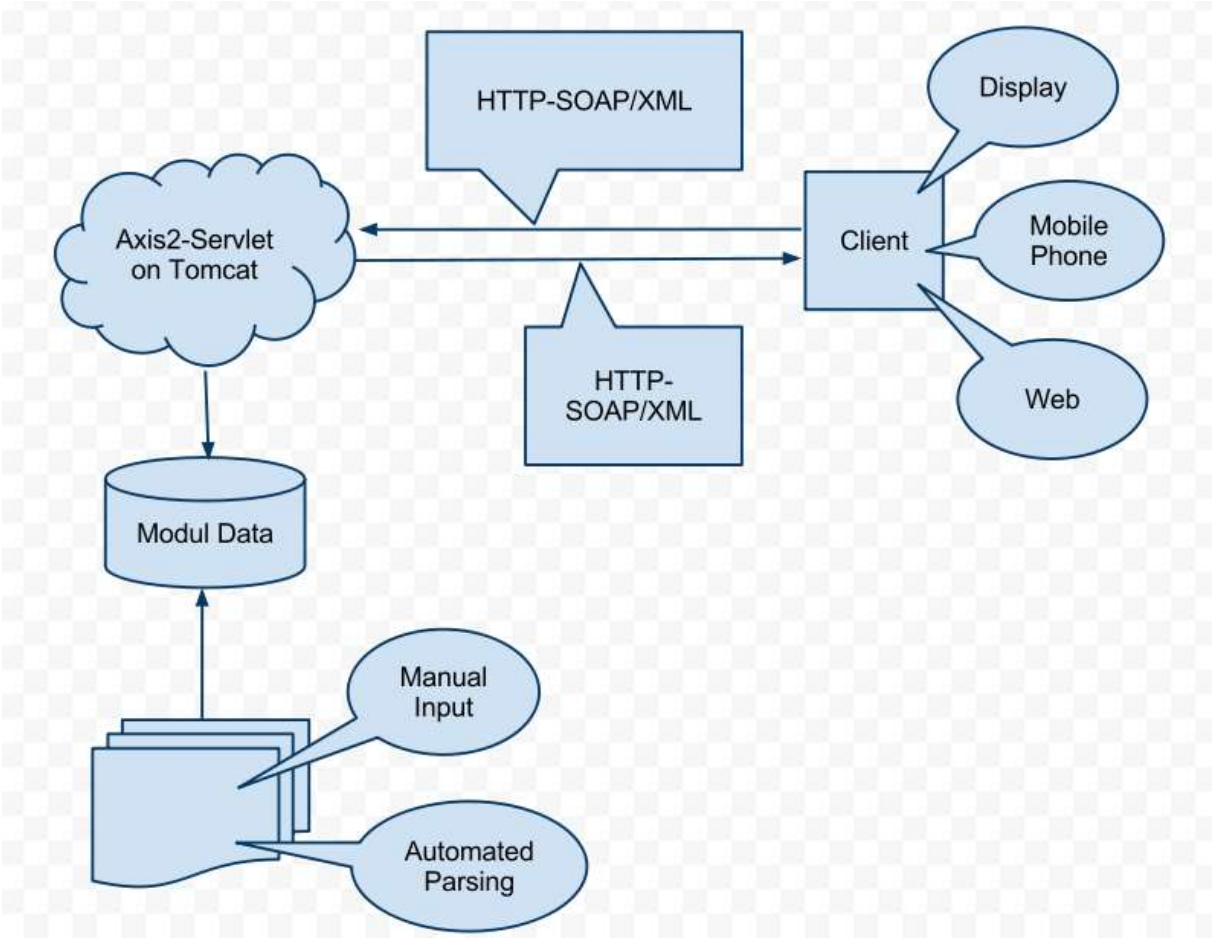
2 Konzept und Rahmenbedingungen

2.1 Benutzer

Die Zielgruppe des Produktes sind vornehmlich interne Entwickler von Infowerk, die die Server-Client-Architektur zur vereinfachten Wartung und Implementierung von Erweiterungen nutzen möchten. Des Weiteren sollen auch externen Entwicklern (beispielsweise von Applikationen für Mobiltelefone) der Zugriff auf die Datenbestände bzw. die einzelnen Module erleichtert werden.

2.2 Systemaufbau

Die folgende Skizze gewährt einen schematischen Überblick über die Kommunikation zwischen Client und Server des implementierten Webservices:



3 Projektplan

Das Studienprojekt untergliedert sich in mehrere Teilbereiche. Zum einen gibt es einen Bereich, welcher projektbegleitende Aufgaben enthält, die fortlaufend ausgeführt werden müssen. Ein anderer Bereich - der Kernbereich des Projekts - beschreibt einzelne Arbeitsschritte, welche nach und nach umgesetzt werden. Wenn nach vollständiger Realisierung des Kernbereichs noch Zeit vorhanden ist, werden die Vorschläge des Erweiterungsteils bearbeitet. Alle drei Bereiche werden im Folgenden kurz beschrieben.

3.1 Projektbegleitende Aufgaben

3.1.1 Projektmanagement

Aufbauend auf dem gleichnamigen Kurs sollen die dort erlernten Techniken praktisch angewendet werden. Hervorzuheben seien hier die Projektplanung, die selbstständige Arbeitsorganisation und Zeiteinteilung als auch die Teamarbeit und Kommunikation mit den Kunden.

3.1.2 Literatur-Recherche

Um sich in das Thema „Webservices" einzuarbeiten, werden hierfür Medien wie Bücher und das Internet zu Rate gezogen. Da die Recherche, um relevante Informationen zu finden oder aufgetretene Probleme zu lösen, ziemlich zeitintensiv sein kann - besonders, wie in unserem Falle, wenn man noch nicht allzu sehr mit der Materie vertraut ist -, sollte dieser Punkt nicht vernachlässigt werden.

3.1.3 Dokumentation

Damit der Auftraggeber und eventuelle Nachfolgeprojekte die Ergebnisse des Projekts verwerten können, müssen diese gut dokumentiert werden. Zu nennen seien hier der an ein Pflichtenheft angelehnte Projektbericht, ergänzende Bemerkungen zur Architektur des Webservices, Kommentare bei der Implementierung und gegebenenfalls eine Anleitung zur Benutzung.

3.2 Obligatorische Kernbereiche

Nachfolgend sind einzelne Arbeitspakete aufgeführt, die spätestens bis zum angegebenen Termin fertig gestellt sein sollten. Bei allen Aufgaben sind alle Teammitglieder zu gleichen Maßen beteiligt. Ein Verantwortlicher kümmert sich dabei um die Koordination des jeweiligen Abschnitts und um die ausreichende Dokumentation im Projektbericht.

3.2.1 Architektur

3.2.1.1 Modellierung

Ziele: Komponenten- und Anwendungsfalldiagramm zur Kommunikation zwischen Server und Client, Klassendiagramm zu den einzelnen Modulen

Verantwortlicher: Tobias

Termin: 13.06.2010

3.2.1.2 Spezifikation

Ziele: Anwendung von Webservice-Standards (wie XML, XSD, SOAP und WSDL) auf die in 3.2.1.1 modellierten Diagramme

Verantwortlicher: Raimund

Termin: 25.07.2010

3.2.2 Implementierung

3.2.2.1 Vorbereitung

Ziele: Einrichtung und Konfiguration der Entwicklungsumgebung (z.B. Server und Datenbank), erste Tests zur Webservice-Programmierung

Verantwortlicher: Christian

Termin: 25.07.2010

3.2.2.2 Durchführung

Ziele: prototypische Implementierung des Webservice anbietenden Servers und anzubindender Clients

Verantwortlicher: Arne

Termin: 15.08.2010

3.3 Optionale Erweiterungen

Falls die Hauptaufgaben des Projekts frühzeitig erfolgreich abgeschlossen werden oder einzelne Teammitglieder motiviert sind, über den zeitlichen Rahmen des Projekts hinaus zu arbeiten, können ein oder mehrere eigene Module erstellt werden, die dann von den iDisplays angezeigt werden.

4 Modellierung

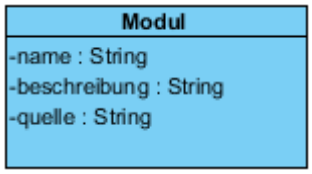
4.1 Daten

Im Folgenden werden die Klassendiagramme und zugehörigen Objektdiagramme einzelner iDisplays-Module aufgeführt, um einen Überblick über die zu übertragenden Daten zu bekommen. Die Objektdiagramme sollen hierbei als anschauliche Beispielinstanzen der Klassen dienen.

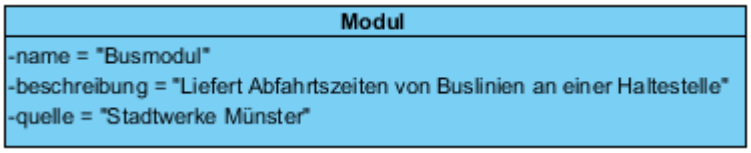
4.1.1 Modul

Diese Klasse stellt eine Oberklasse für die verschiedenen Module dar. Sie enthält Meta-Daten, wie eine Modulbeschreibung, zum besseren Verständnis für den Benutzer.

4.1.1.1 Klassendiagramm



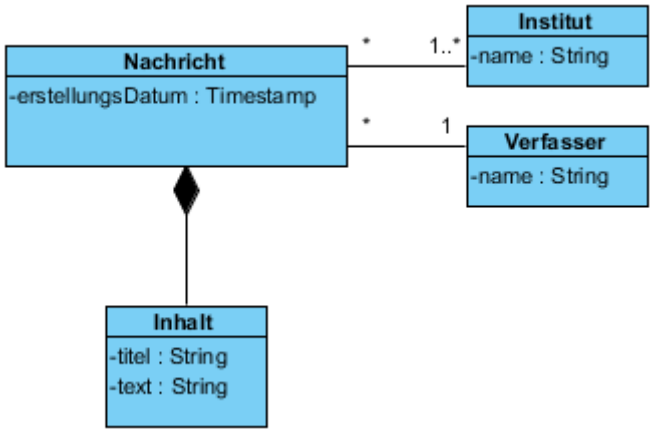
4.1.1.2 Objektdiagramm



4.1.2 Nachrichtenmodul

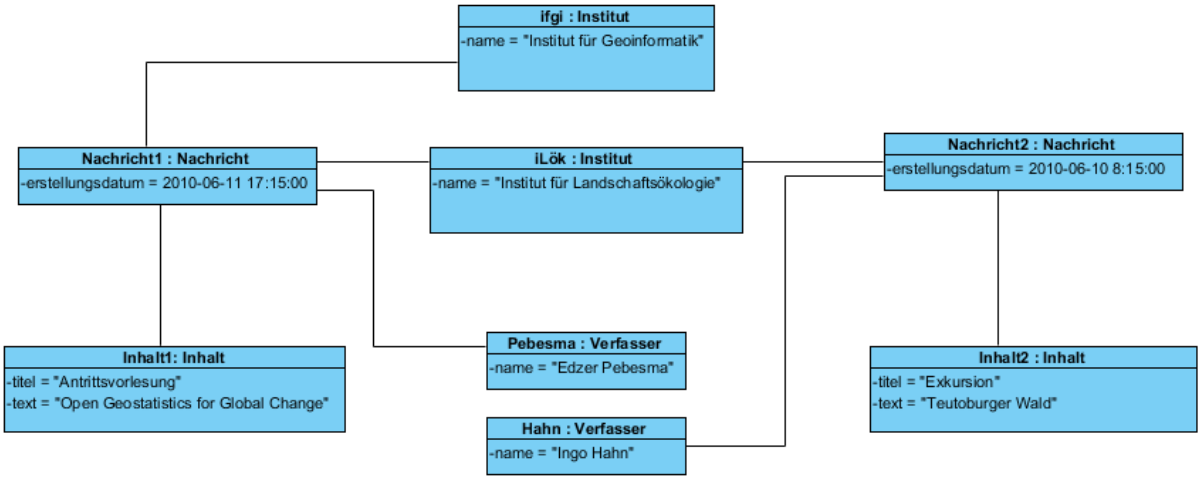
Das Nachrichtenmodul informiert den Benutzer über die neuesten institutsinternen Neuigkeiten und Ereignisse. Diese können den einzelnen Instituten zugeordnet werden, um den Clients sinnvolle Auswahlmöglichkeiten zu bieten.

4.1.2.1 Klassendiagramm



Zusätzlich wären folgende Attribute bei der Klasse „Inhalt" optional denkbar:
„raum: String", „zeit: Timestamp", „link: URL" und „ bild: URL".

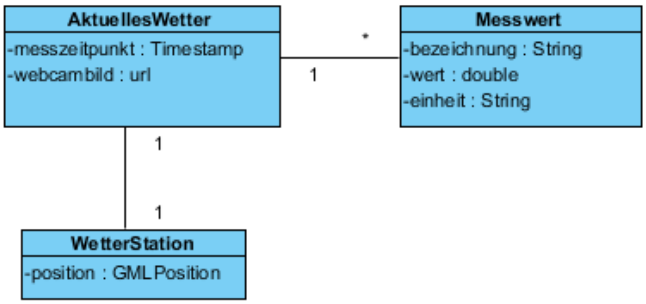
4.1.2.2 Objektdiagramm



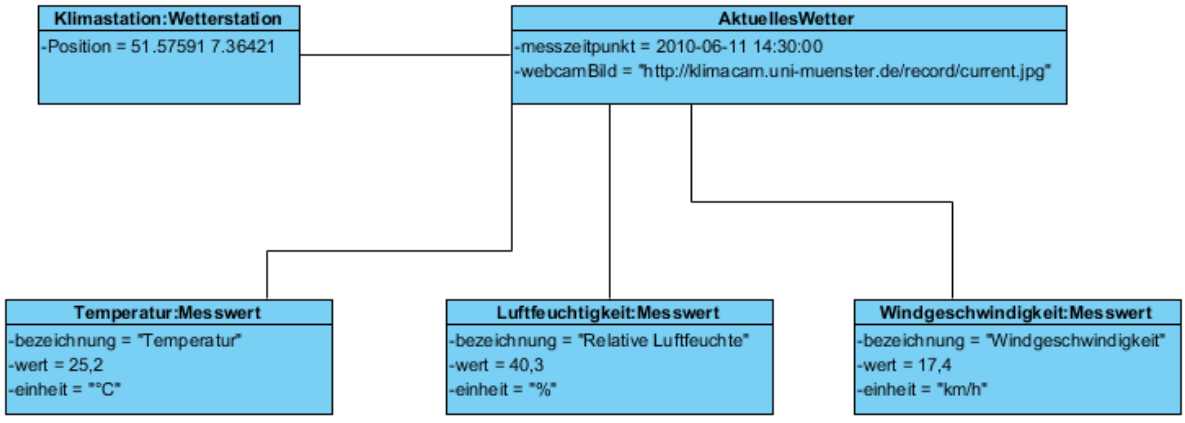
4.1.3 Wettermodul

Das Wettermodul informiert die Benutzer des Webservices über das aktuelle Wetter in ihrer Region. Es stellt dem User beispielsweise Informationen über die aktuelle Temperatur, Windgeschwindigkeit und Luftfeuchtigkeit in der gewünschten Region bereit.

4.1.3.1 Klassendiagramm



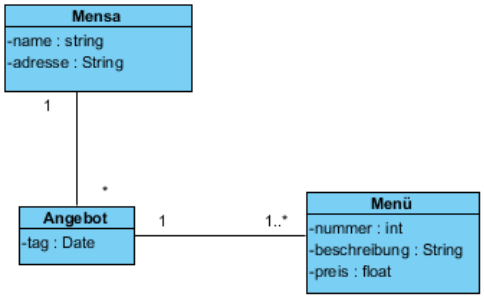
4.1.3.2 Objektdiagramm



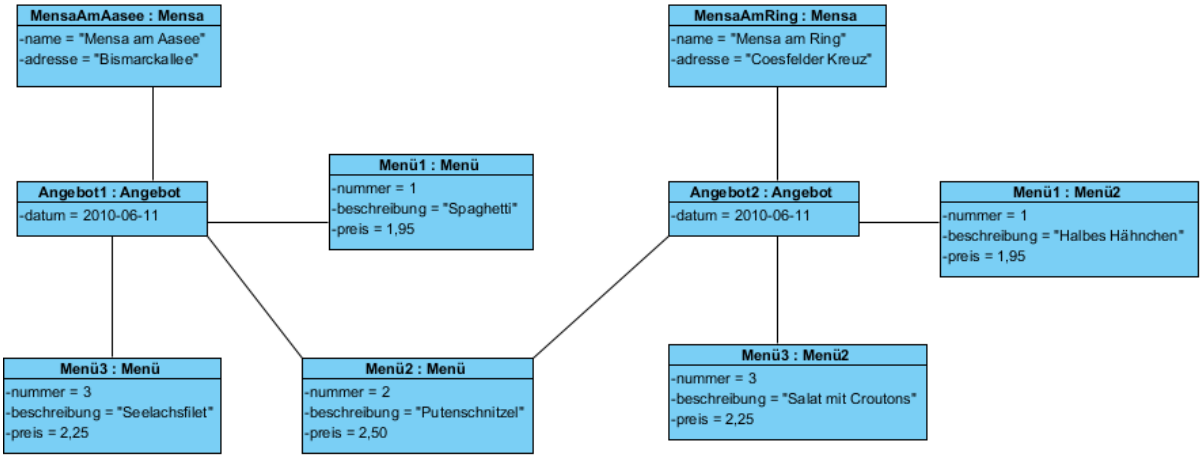
4.1.4 Mensamodul

Im Mensamodul werden die Speisepläne von verschiedenen Mensen angezeigt, die ihr Angebot täglich wechseln. Die Speisepläne werden aus den jeweils aktuellen Onlineinformationen des Studentenwerks bezogen.

4.1.4.1 Klassendiagramm



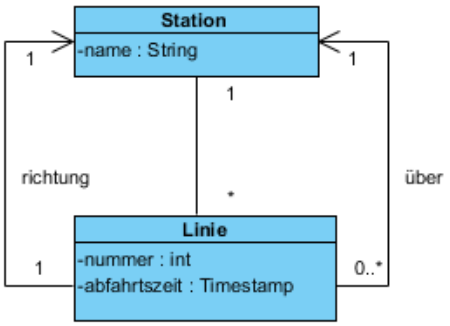
4.1.4.2 Objektdiagramm



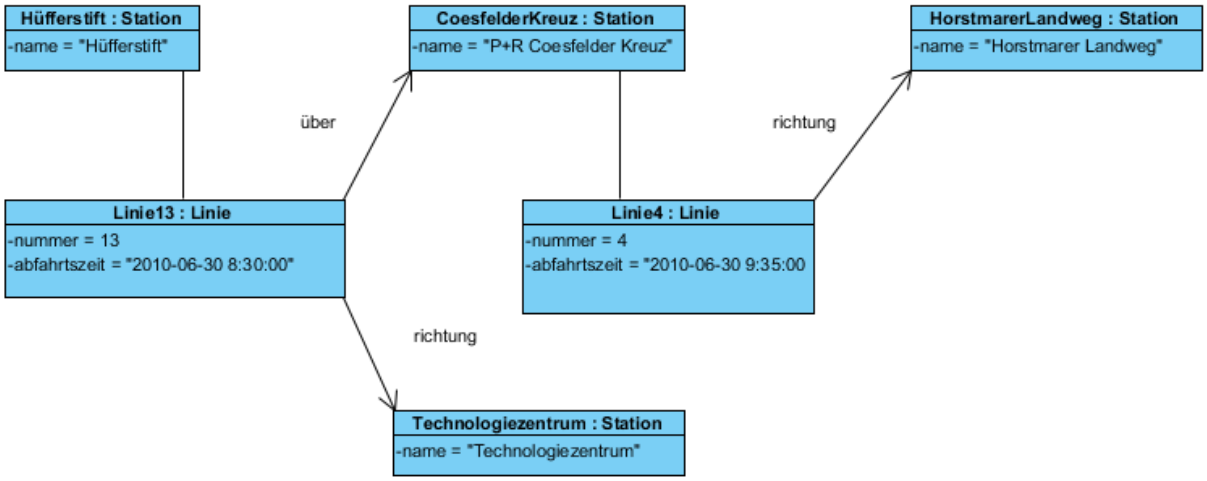
4.1.5 Busmodul

Das Busmodul beinhaltet den Abfahrtszeitplan einer bestimmten Haltestelle. Hierbei wird für die entsprechende Haltestelle eine Liste der nächsten dort abfahrenden Busse mit Informationen über Fahrtrichtung und Abfahrtszeit angezeigt.

4.1.5.1 Klassendiagramm



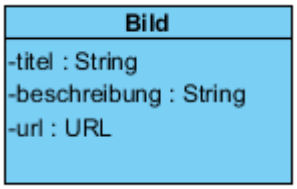
4.1.5.2 Objektdiagramm



4.1.6 Bildmodul

Über das Bildmodul können Bilder mittels einer URL abgerufen werden.

4.1.6.1 Klassendiagramm



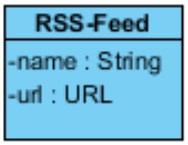
Zusätzlich wären folgende Attribute optional denkbar:
„kategorie: Set<String>", „quelle: String" und „erstellungsdatum: Date".

4.1.6.2 Objektdiagramm

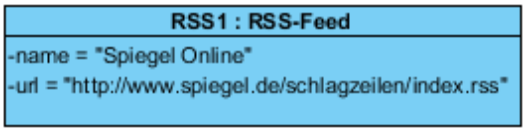
4.1.7 RSS-Feed Modul

RSS Feeds ermöglichen dem User, Inhalte einer Webseite - oder Teile davon - zu „abonnieren". So können neu veröffentlichte Inhalte dank RSS in regelmäßigen Abständen auf die iDisplays, Computer oder portablen Geräte geladen werden. Dadurch bekommt der Abonnent die jeweils neuesten Informationen automatisch und bequem geliefert.

4.1.7.1 Klassendiagramm



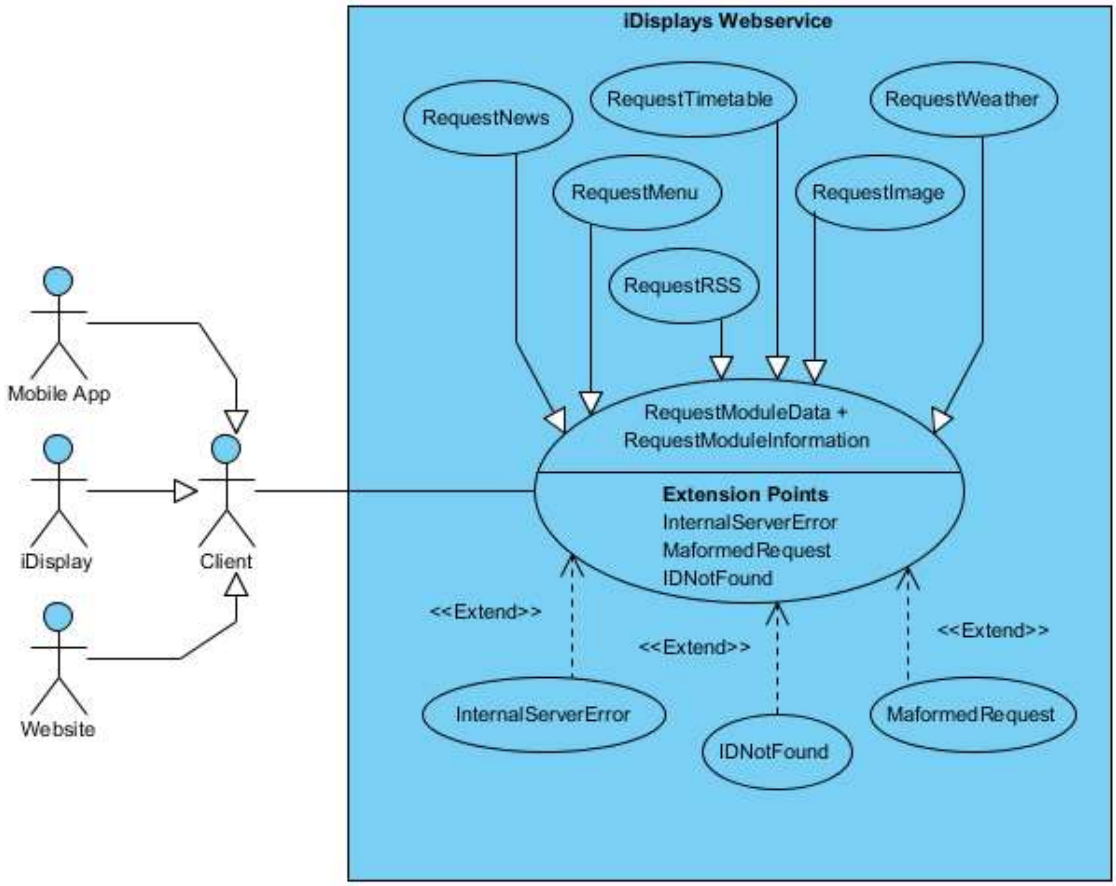
4.1.7.2 Objektdiagramm



4.2 Funktionen

Die Funktionalität unseres Webservices soll anhand eines Anwendungsfalldiagramms sowie Beschreibungen und Beispielen zu den einzelnen Anwendungsfällen veranschaulicht werden.

4.2.1 Anwendungsfalldiagramm



4.2.2 Anwendungsfallbeschreibungen

Name des Anwendungsfalls	RequestModule Information	**Übersetzung**	Modulinformationen abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client schickt seinen Schlüssel an den Server. 2. Der Server schickt Informationen der einzelnen Module (z.B. Identifier) an den Client.		

Name des Anwendungsfalls	RequestModuleData	**Übersetzung**	Moduldaten abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client fragt den Server nach bestimmten Daten von Modulen. 2. Der Server schickt die gewünschten Daten an den Client.		

Name des Anwendungsfalls	RequestNews	**Übersetzung**	Nachrichten abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client wählt eine Anzahl von Nachrichten und eine Menge von Instituten, dessen Nachrichten er erhalten möchte. 2. Der Server schickt die gewünschten Nachrichten an den Client.		

Name des Anwendungsfalls	RequestMenu	**Übersetzung**	Menu abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client wählt eine Menge von Daten (= Plural von Datum) und eine Menge von Mensen, zu denen er alle angebotenen Menüs erhalten möchte. 2. Der Server schickt die gewünschten Menus an den Client.		

Name des Anwendungsfalls	RequestWeather	**Übersetzung**	Wetter abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client schickt eine Position (z.B. in WGS84 Koordinaten), zu der er das aktuelle Wetter erhalten möchte. 2. Der Server schickt Daten der nächstgelegenen Wetterstation an den Client.		

Name des Anwendungsfalls	RequestTimetable	**Übersetzung**	Fahrplan abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client wählt eine Menge von Stationen und die Anzahl der nächsten Verbindungen, die er erhalten möchte. 2. Der Server schickt die gewünschten Verbindungen an den Client.		

Name des Anwendungsfalls	RequestImage	**Übersetzung**	Bild abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client wählt den Titel des Bildes, das er erhalten möchte. 2. Der Server schickt das gewünschte Bild an den Client.		

Name des Anwendungsfalls	RequestRSS	**Übersetzung**	RSS-Feed abfragen
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client wählt den Namen des RSS-Feeds, das er erhalten möchte. 2. Der Server schickt das gewünschte Feed an den Client.		

Name des Anwendungsfalls	MalformedRequest	**Übersetzung**	Syntaxfehler
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client schickt eine syntaktisch nicht korrekte Anfrage (z.B. ein Buchstabe anstatt einer Zahl). 2. Der Server schickt eine entsprechende Fehlermeldung an den Client.		

Name des Anwendungsfalls	IDNotFound	**Übersetzung**	ID nicht gefunden
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client schickt eine Anfrage, dessen Identifier nicht gefunden werden kann (z.B. einen nicht vorhandener Stationsname). 2. Der Server schickt eine entsprechende Fehlermeldung an den Client.		

Name des Anwendungsfalls	InternalServerError	**Übersetzung**	Interner Serverfehler
Beteiligte Akteure	Client		
Ereignisfluss	1. Der Client schickt eine Anfrage an den Server. 2. Der Server kann auf die angefragten Daten intern nicht zugreifen. 3. Der Server schickt eine entsprechende Fehlermeldung an den Client.		

5 Spezifikation

5.1 Grundlagen

Zu den grundlegenden Bestandteilen eines Webservices gehören XML, XML Schema, WSDL, HTTP oder SOAP sowie UDDI. XML erlaubt es, Daten in strukturierter Form und plattformunabhängig zu speichern und zu übertragen. Ein XML Schema definiert dabei den Aufbau eines XML-Dokuments. WSDL beschreibt die Funktionalität des Webservices und wie man auf diesen zugreifen kann. HTTP und SOAP sind Protokolle zur Übertragung von Informationen. Über UDDI kann man seinen Webservice registrieren und andere Webservices suchen.

In unserem Studienprojekt verwenden wir die ersten der vier genannten Bestandteile eines Webservices. Für ein besseres Verständnis der einzelnen Komponenten empfehlen wir, die dazugehörigen Tutorials auf der Seite http://www.w3schools.com/sitemap/sitemap_tutorials.asp durchzulesen. Um die Dateitypen der Komponenten mit hervorgehobener Syntax zu betrachten und zu editieren, empfehlen wir die „Eclipse XML Editor und Tools" und „Eclipse Web Developer Tools" der Eclipse Web Tools Platform. Diese Plugins lassen sich in Eclipse über die Downloadseite http://download.eclipse.org/webtools/updates/ installieren.

5.2 Anwendungsbeispiel

Wie kann nun ein Nutzer mit unserem Webservice kommunizieren?

Zunächst informiert er sich in der WSDL-Datei (siehe Anhang) über die Schnittstellen und Schemata unseres Webservices. Mithilfe von geeigneten Tools (siehe Implementierung) kann sich der Benutzer aus der WDSL-Datei automatisch Java-Klassen generieren lassen, mit denen er den Webservice ansprechen kann. Dadurch ist es ihm möglich, Anfragen an den Webservice zu stellen und Antworten vom Webservice zu bekommen. Im Folgenden wird beschrieben, welche Daten in XML übertragen werden könnten.

Um Informationen (z.B. Identifier und Parameter) über die einzelnen Module zu erhalten, schickt der Benutzer einen ModuleCapabilitiesRequest an den Webservice. Hierbei muss er lediglich einen Schlüssel zur Authentifizierung angeben, den er vorher bei den Administratoren des Webservices beantragt hat:

```
<ModuleCapabilitiesRequest>
  <key>einGueltigerSchluessel</key>
</ModuleCapabilitiesRequest>
```

Der Server schickt ihm daraufhin die ModuleCapabilitiesResponse, bei der alle Module detailliert beschrieben werden:

```
<ModuleCapabilitiesResponse>
  <newsModule>
    <description>Liefert universitätsinterne Nachrichten der angegebenen
Instituten</description>
    <source>Westfälische Wilhelms-Universität Münster</source>
    <instituteID>ifgi</instituteID>
    <instituteID>iLök</instituteID>
  </newsModule>
  <weatherModule>
```

```
   <description>Liefert aktuelle Wetterdaten zu der angegebenen Position
nächstgelegenen Wetterstation</description>
   <source>Verschiedene Quellen</source>
   <weatherStationPosition srsName="EPSG:4326">51.57591
7.36421</weatherStationPosition>
  </weatherModule>
  <cafeteriaModule>
   <description>Liefert Menüs zu den angegebenen Mensen in dem angegebenen
Zeitraum</description>
   <source>Studentenwerk Münster</source>
   <cafeteria>
    <name>Mensa am Aasee</name>
    <offeringDates>
     <min>2010-06-11</min>
     <max>2010-06-11</max>
    </offeringDates>
   </cafeteria>
   <cafeteria>
    <name>Mensa am Ring</name>
    <offeringDates>
     <min>2010-06-11</min>
     <max>2010-06-11</max>
    </offeringDates>
   </cafeteria>
  </cafeteriaModule>
  <busModule>
   <description>Liefert Abfahrtszeiten von Buslinien zu der angegebenen
Haltestelle</description>
   <source>Stadtwerke Münster</source>
   <busStationName>Hüfferstift</busStationName>
   <busStationName>P+R Coesfelder Kreuz</busStationName>
  </busModule>
  <imageModule>
   <description>Liefert die URLs der angegebenen Bilder</description>
   <source>Verschiedene Quellen</source>
   <imageTitle>Comic</imageTitle>
   <imageTitle>Raumplan</imageTitle>
  </imageModule>
  <rssFeedModule>
   <description>Liefert die URLs der angegebenen RSS-Feeds</description>
   <source>Verschiedene Quellen</source>
   <rssFeedName>Spiegel Online</rssFeedName>
   <rssFeedName>Raimunds Blog</rssFeedName>
  </rssFeedModule>
</ModuleCapabilitiesResponse>
```

Schließlich kann der Benutzer die Angaben aus dem ModuleCapabilitiesRequest für den ModuleDataRequest verwenden, bei dem er spezifische Daten der Module anfragt (siehe Anwendungsfälle):

```
<ModuleDataRequest>
  <newsModule>
   <instituteId>ifgi</instituteId>
   <instituteId>iLoek</instituteId>
   <numberOfNewsItems>10</numberOfNewsItems>
```

```
 </newsModule>
 <weatherModule>
  <weatherStationPosition srsName="EPSG:4326">52 7</weatherStationPosition>
 </weatherModule>
 <busModule>
  <busStation>
   <name>P+R Coesfelder Kreuz</name>
   <numberOfDepartures>5</numberOfDepartures>
  </busStation>
  <busStation>
   <name>Hüfferstift</name>
   <numberOfDepartures>5</numberOfDepartures>
  </busStation>
 </busModule>
 <imageModule>
  <imageTitle>Dom</imageTitle>
 </imageModule>
 <rssFeedModule>
  <rssFeedName>Spiegel Online</rssFeedName>
 </rssFeedModule>
</ModuleDataRequest>
```

Als Antwort liefert der Webservice die ModuleDataResponse. Diese enthält die gewünschten Daten der einzelnen Modulen (siehe Klassendiagramme).

```
<ModuleDataResponse>
 <newsModule>
  <newsItem>
   <author>
    <name>Edzer Pebesma</name>
   </author>
   <institute>
    <name>Institut für Geoinformatik</name>
   </institute>
   <institute>
    <name>Institut für Landschaftsökologie</name>
   </institute>
   <creationDate>2010-06-11T17:15:00</creationDate>
   <content>
    <title>Antrittsvorlesung</title>
    <text>Open Geostatistics for Global Change</text>
   </content>
  </newsItem>
  <newsItem>
   <author>
    <name>Ingo Hahn</name>
   </author>
   <institute>
    <name>Institut für Landschaftsökologie</name>
   </institute>
   <creationDate>2010-06-10T08:15:00</creationDate>
   <content>
    <title>Exkursion</title>
    <text>Teutoburger Wald</text>
   </content>
```

```
   </newsItem>
  </newsModule>
  <weatherModule>
   <weatherStation>
    <position srsName="EPSG:4326">51.57591 7.36421</position>
    <measuringTime>2010-06-11T14:30:00</measuringTime>
    <webcamPicture>http://klimacam.uni-
muenster.de/record/current.jpg</webcamPicture>
    <measuringValue>
     <name>Temperatur</name>
     <value>25.2</value>
     <unit>°C</unit>
    </measuringValue>
    <measuringValue>
     <name>Relative Luftfeuchte</name>
     <value>40.3</value>
     <unit>%</unit>
    </measuringValue>
    <measuringValue>
     <name>Windgeschwindigkeit</name>
     <value>17.4</value>
     <unit>km/h</unit>
    </measuringValue>
   </weatherStation>
  </weatherModule>
  <busModule>
   <busStation>
    <name>P+R Coesfelder Kreuz</name>
    <line>
     <number>4</number>
     <destination>Horstmarer Landweg</destination>
     <departureTime>2010-06-30T09:35:00</departureTime>
    </line>
   </busStation>
   <busStation>
    <name>Hüfferstift</name>
    <line>
     <number>13</number>
     <via>P+R Coesfelder Kreuz</via>
     <destination>Technologiezentrum</destination>
     <departureTime>2010-06-30T08:30:00</departureTime>
    </line>
   </busStation>
  </busModule>
  <rssFeedModule>
   <rssFeed>
    <name>Spiegel Online</name>
    <url>http://www.spiegel.de/schlagzeilen/index.rss</url>
   </rssFeed>
  </rssFeedModule>
</ModuleDataResponse>
```

Die XML Schemata (siehe Anhang) helfen bei der Erzeugung und Validierung der XML-Dateien.

6 Implementierung

6.1 Systemvoraussetzungen

Neben der aktuellsten Version des Java SDK wird zum Ausführen des Projekts NetBeans benötigt. Genau wie Eclipse ist NetBeans eine Open-Source Entwicklungsumgebung, u.a. für Java-Programme. Für unser Projekt wird die Java- oder All-Version der NetBeans IDE benötigt, die man sich unter http://netbeans.org/downloads/ herunterladen kann. Bei beiden ist für unser Projekt notwendige Software - den Java Web Tools, Maven und Apache Tomcat - enthalten. Maven ermöglicht hierbei eine einfache Integration von Java-Dateiarchiven und bei Apache Tomcat handelt es sich um einen Open Source Server für Java-Codes. Optional kann man sich deren neueste Versionen unter http://maven.apache.org/download.html und http://tomcat.apache.org/ herunterladen und in NetBeans einbinden. Bei Apache Tomcat ist zu beachten, ein Verzeichnis mit Schreibrechten zu wählen (z.B. dem Benutzer-Ordner unter Windows), damit später Log-Dateien dort gespeichert werden können.

6.2 Systemeinrichtung

Als erstes öffnet man das Projekt „id-ws" in NetBeans. Nun sollte das Wurzelverzeichnis „iDisplays Webservice" bei den Projekten vorhanden sein. Im Unterordner „Modules" findet man folgende fünf Komponenten des Webservices:

- „XML Bindings": In diesem Verzeichnis befinden sich die XML Schemata, aus denen mithilfe von XMLBeans automatisch Java-Klassen generiert werden.
- „Server": Hier befinden sich generische Klassen, die zur Verarbeitung von Anfragen an den Server notwendig sind.
- „Base Modules": Um Anfragen der einzelnen Module zu bearbeiten, ist dieses Unterverzeichnis vorgesehen.
- „Webapp": In diesem Ordner werden alle benötigten Dateien für den Webservice gesammelt, um sie anschließend als *.war-Datei für den Tomcat-Server zu packen.
- „Client": In diesem Verzeichnis ist ein beispielhafter Client implementiert.

Nun führt man mit einem Rechtsklick auf das Wurzelverzeichnis „iDisplays Webservice" den Befehl „Build" aus, um automatisch die Java-Klassen zu kompilieren. Dieser Schritt kann einige Minuten in Anspruch nehmen.

Danach fügt man im Services Tab (wenn noch nicht vorhanden) den Apache Tomcat bei den Servern hinzu. Hier gibt man einen beliebigen Namen für den Server ein, wählt das Wurzelverzeichnis des Tomcat-Servers aus und legt einen Benutzer als Manager an.

Im Projects Tab öffnet man im Anschluss das Unterverzeichnis „Webapp" und führt mit einem Rechtsklick darauf den Befehl „Run" aus, um den soeben eingerichteten Server zu starten. Nun sollte sich unser Webservice ein Browser unter der Adresse http://localhost:8080/id/ öffnen. Die WSDL-Datei zur Beschreibung der Funktionalität lässt sich unter http://localhost:8080/id/services/ws?wsdl abrufen.

In NetBeans kann man anschließend das Unterverzeichnis „Client" öffnen und mit einem Rechtsklick darauf den Befehl „Run" ausführen. Nachdem man die Main-Klasse ausgewählt hat, wird ein Client gestartet, welcher beispielhafte Anfragen an den Webservice stellt und beantwortet bekommt.

6.3 Hinzufügen eines neuen Moduls

6.3.1 XML Schema

Das XML Schema des neuen Moduls muss sowohl in "XML Bindings" unter „Other Sources > xsd > id > 1.0.0 > modules" als auch in „Webapp " unter „WEB-INF > services > ws > META-INF > modules" angelegt werden. Das Schema muss dabei die drei Datentypen „AbstractModuleCapabilities", „AbstractGetModuleDataRequest" und „AbstractModuleData" überschreiben. AbstractModuleCapabilities liefert dem Client Informationen zu dem neuen Modul. Im AbstractGetModuleDataRequest gibt der Client an, welche Moduldaten er vom Server erhalten möchte. Die Antwort erhält er vom Server durch AbstractModuleData. Eine Hilfestellung zur Spezifikation bieten die bereits vorhandenen Schemata.

Anschließend muss das neue Schema in „id.xsd" per include hinzugefügt werden. In dieser Datei sind auch alle abstrakten Datentypen und Fehlertypen enthalten. Die WSDL-Datei, die sich im Verzeichnis „Webapp" unter „Webpages > WEB-INF > services > ws > META-INF > ws.wsdl" befindet, braucht nicht modifiziert zu werden.

Zuletzt führt man vom gesamten Projekt „iDisplays Webservice" den Befehl „Build" aus, um Java-Klassen aus dem angelegten Schema zu erzeugen. Diese kann man nun für den Server- und Client-Code verwenden.

6.3.2 Server-Code

Zur Implementierung des neuen Moduls beim Server legt man zunächst im Verzeichnis "Base Modules" unter Source Packages ein neues Package mit dem neuen Modulnamen an. In diesem Package erstellt man eine neue Klasse, die die Klasse „IModuleHandler" erweitert. Durch Überschreiben der beiden „handle" Methoden können nun Anfragen von Clients ausgewertet und bearbeitet werden. Auch hier geben die bereits implementierten Module einen guten Überblick. Einige Beispiel-Daten werden unter „Other Sources > /src/main/resources/" in xml-Form gespeichert und geparst.

Zuletzt registriert man den neuen ModuleHandler in „Webapp" unter „Other Sources > /src/main/resources/ > config.xml". Im Verzeichnis „Server" braucht hingegen nichts verändert zu werden. Die Webapp kann wieder mit dem Befehl „Run" gestartet werden.

6.3.3 Client-Code

Zur Implementierung des neuen Moduls beim Client legt man zunächst im Verzeichnis "Client" unter Source Packages ein neues Package mit dem neuen Modulnamen an. In diesem Package erstellt man eine neue Klasse, die die Klasse „AbstractModuleUpdater" erweitert. Dadurch können nun in bestimmten Zeitabständen Anfragen an den Webservice gesendet werden. Die bereits implementierten Module helfen bei der Implementierung.

Zuletzt registriert man den neuen ModuleUpdater in „Client" unter „Other Sources > /src/main/resources/ > config.xml". Der Client kann wieder mit dem Befehl „Run" gestartet werden.

7 Quellen

- Frotscher, T. et al. (2007): Java Web Services mit Apache Axis2
- W3 School Tutorials zu XML, XML Schema, WSDL, Web Services und SOAP: http://www.w3schools.com/sitemap/sitemap_tutorials.asp