# Lösen eines begrenzten Datenhaltungsproblems mit einem nativen XML-Datenbanksystem am Beispiel einer Systemstatusdatenbank im IT-Support

Martin Eigenmann

9. Juni 2015

# 1

# Abstract

Die Seminararbeit des Moduls Datenbanken und Informationssysteme betrachtet das Lösen eines spezifisches Datenhaltungsprolems, mittels eines nativen XML-Datenbanksystems.
Gezeigt wird, dass unter Verwendung eines nativen XML-DMBS, ein WEB-GUI erstellt werden kann, welches den aktuellen Systemzustand der IT-Infrastruktur, sowie vergangene Fehler, übersichtlich darstellt.

# 2

# Inhaltsverzeichnis

# 3
# Einleitung

Die vorliegende Arbeit wird im Rahmen des Seminars XML Datenbanksysteme durchgeführt. In diesem ersten Kapitel wird kurz in das Ziel, sowie Abgrenzung und Vorgehen dieser Arbeit eingeführt.

## 3.1. Thema

Ziel der Arbeit ist es den Zustand der IT-Systeme auf einer Webseite zu publizieren, so dass Mitarbeiter eine übersichtliche Zusammenfassung der Lauffähigkeit der IT-Systeme erhalten.

## 3.2. Ausgangslage

Im Unternehmen existiert ein Überwachungssystem (Neteye) welches das Verhalten von der IT-Infrastruktur misst. Auf Grund dieser regelmässigen Vermessung kann der Zustand der IT-Services abgeleitet und Fehlfunktionen erkannt werden.
Der aktuelle Zustand aller Services wird im Interface von Neteye dargestellt. Diese Darstellung ist jedoch sehr unübersichtlich und nur für geschultes Personal geeignet.
Darüber hinaus ist eine zusammenfassende Auflistung der vergangenen Fehlfunktionen nicht möglich.

## 3.3. Ziele der Arbeit

Ziel der Arbeit ist es, für alle Mitarbeiter Informationen über den aktuellen Zustand der IT-Infrastruktur zugänglich zu machen.

Im Verlaufe der Arbeit soll gezeigt werden, dass unter Verwendung eines XML-DBMS die Zustände der IT-Infrastruktur abgespeichert und verwaltet werden können, so dass, darauf basierend, ein WEB-GUI implementiert werden kann, welche sowohl die Zustände der wichtigsten Services sowie aufgetretene Fehlfunktionen anzeigt.

Neben der Erarbeitung der technischen Grundlagen muss auch eine kurzen Analyse der bereits vorhandenen Möglichkeiten der Mitarbeiter, Informationen über den aktuellen Zustand der IT-Infrastruktur zu erlangen, durchgeführt werden.

## 3.4. Abgrenzung

Die Arbeit konzentriert sich auf die Lösung des Datenhaltungsproblems. Auf eine Anforderungsanalyse an den zu implementierenden Prototypen wird deshalb verzichtet.

Anforderungen an Design und Usability stehen im Hintergrund und werden im Rahmen dieser Arbeit nicht betrachtet.

## 3.5. Vorgehen

Um das Datenhaltungsproblem zu lösen, muss zuerst die Analyse der Problemstellung, sowie eine Erarbeitung der technischen Grundlagen durchgeführt werden.

In einem zweiten Schritt wird die Konzeption des WEB-GUI und der darunter liegenden Logik durchgeführt und umgesetzt.

Im letzten Schritt wird die Erreichung der Ziele überprüft und ein Fazit gezogen.

# 4
# Projektplanung

Das Kapitel Projektplanung führt alle relevanten Informationen zur Durchführungsdauer als auch zum Organisatorischen Rahmen auf.

## 4.1. Projektplan

Die Seminararbeit muss im Zeitraum von 11.03.2015 bis 09.06.2015 durchgeführt werden. Die Arbeit wurde in den Kalenderwochen 21 bis 23 durchgeführt.

## 4.2. Zeitlicher Rahmen

Der offizielle Projektstart erfolgt mit dem Kick-Off am 11.03.2015. Am 16.06.2015 finden die Abschlusspräsentationen statt.

## 4.3. Organisatorischer Rahmen

In der nachstehenden Tabelle sind alle massgeblich involvierten Personen aufgeführt.

Tabelle 4.1.: Involvierte Personen

| Personen | Kontakt |
|---|---|
| **Reto Knaack**<br>(Studeingangs Leiter) | ZHAW Standort Zürich<br>Lagerstrasse 41 / 8004 Zürich<br>Reto.Knaack@zhaw.ch |
| **Klaus Wolfertz**<br>(Lehrperson) | ZHAW Standort Zürich<br>Lagerstrasse 41 / 8004 Zürich<br>xwoe@zhaw.ch |
| **Martin Eigenmann**<br>(Student) | Harfenbergstrasse 5 / 9000 St.Gallen<br>study@eigenmannmartin.ch |

# 5
# Recherche

In diesem Kapitel wird erläutert, welche Funktionalität durch Neteye zur Verfügung gestellt wird, welche Funktionalitäten fehlen und welche Probleme aus diesem Sachverhalt entstehen.

## 5.1. Grundlagen

Neteye ist ein von WürhtPhoenix vertriebenes, auf OpenSource basiertes IT-Management System. Neben Asset, Inventory-, Capacity- und Service Management nach ITIL Standards sind auch ausgeprägte Überwachungsfunktionalitäten vorhanden.

Zur Überwachung von Systemen und Services werden die Tool-Kits Nagios und Alexa verwendet. Während Nagios den Zustand von Systemen über das Auslesen von Logs und Statusabfragen an die Systeme selbst aufzeichnet, führt Alexa sogenannte Real-User Experience Tests durch. Dabei wird der zu testende Service so getestet, als ob ein Benutzer ihn verwendet.

Die aggregierten Zustandsdaten eines Services können in einem Business-Processes zusammengefasst werden. Dabei wird nicht nur der Zustand des zu überwachenden Services mit eingezogen, sondern auch die Abhängigkeiten, wie zum Beispiel Netzwerk, Firewall-Auslastung oder Temperatur im Datacenter.
Ein Business-Process repräsentiert also die Funktionsfähigkeit eines Services. (z.B. SAP oder Email)

Der Detaillierungsgrad der im Neteye angezeigten Informationen kann nicht variiert werden. Es ist auch nicht möglich Informationen auszublenden oder Anzeige-Berechtigungen zu definieren. Beim Zugriff auf das von Neteye zur Verfügung gestellte WEB-GUI sind entweder alle, oder keine Status-Informationen einsehbar.

## 5.2. Problemanalyse

Neteye wurde designed um die Arbeit in einer IT-Abteilung zu erleichtern. So sind sehr viele und sehr detaillierte Informationen im GUI einsehbar. Die verwendete tabellarische Ansicht

macht das Arbeiten und Navigieren für Fachpersonal sehr einfach, für themenfremde Personen jedoch nahezu unmöglich.

Es gibt auch keine Möglichkeit Informationen auszublenden, um so die Verständlichkeit für Benutzer zu erhöhen. Der Betrachter muss also entsprechendes Verständnis und Grundwissen über die gesamte IT-Infrastruktur mitbringen.

Zusammenfassend bietet Neteye viele Informationen, aber keine Möglichkeit diese adressatengerecht für nicht IT-Mitarbeiter aufzubereiten. Der Betrachter wird überflutet mit Informationen, die ihn möglicherweise nicht interessieren oder verwirren.

## 5.3. Zwischenstand

Das WEB-GUI vom Neteye erlaubt eine sehr detaillierte Auswertung und Analyse des Systemzustands. Top-Level Ansichten sind jedoch nicht implementiert und ermöglichen daher keine managementfreundliche Darstellung des Zustandes der IT-Infrastruktur. Aus den Vorgaben des Kapitels Einleitung und den Erkenntnissen aus diesem Kapitel, soll im nachfolgenden Kapitel Konzept ein WEB-GUI entworfen werden, welches allen Mitarbeitern eine einfache und verständliche Übersicht über die Funktionsfähigkeit der IT-Infrastruktur liefert.

# 6
# Konzept

In diesem Kapitel wird ein System entworfen, welchen den Anforderungen aus den Kapiteln Einleitung und Recherche gerecht wird. Dazu wird zuerst ein grobes Design der Software erarbeitet und anschliessend konkretisiert.

## 6.1. Designansätze

Zur Lösung der Aufgabenstellung sind nachfolgend zwei möglich Designansätze aufgezeigt. Diese werden kurz beleuchtet und der am besten passenden ausgewählt.

### 6.1.1. Integrated Application

Die Applikation wird als Plug-In beim Neteye integriert und stellt seine Funktionalitäten als Teil des bestehenden Web-Frontend zur Verfügung. Die zu entwickelnde Software kann direkt auf die Datenbank zugreifen und es entfällt die Entwicklung einer Schnittstelle.

### 6.1.2. Single Tier Application

Die Applikation ist selbständig, verfügt über ein eigenes Web-Gui und greift über die API auf Neteye zu. Die Daten werden periodisch aktualisiert und lokal in einer XML-Datenbank gespeichert.

## 6.2. Entscheid

Die gesetzten Ziele dieser Arbeit sind mit beiden Desingansätzen erreichbar.
Da die Single Tier Application auf einem eigenen Server läuft und nur die API zum Neteye beachtet werden muss, ist dies die sauberere Lösung. Somit muss keine Veränderung am Neteye vorgenommen werden.

## 6.3. Design der Software

Der Prototyp besteht aus den Bausteinen Backend, XML DBMS und Frontend.
Nur die Backend-Komponente greift über eine Schnittstelle auf Neteye zu.

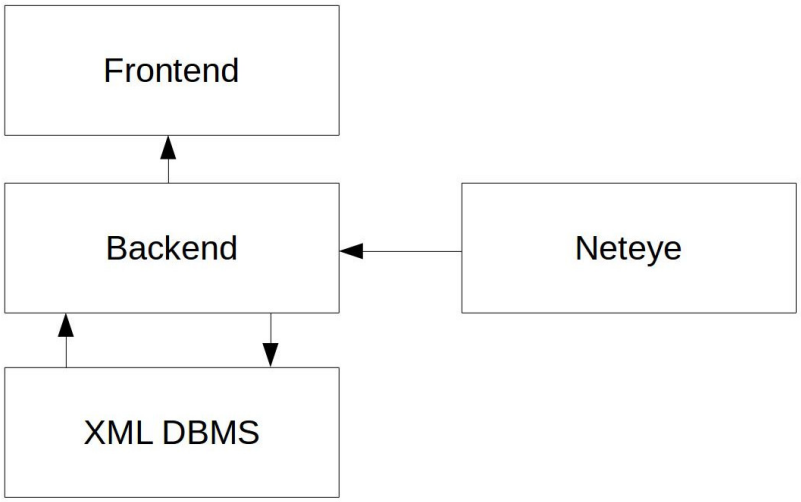


Abbildung 6.1.: Komponenten

### 6.3.1. Backend

Die Daten auf dem Fremdsystem Neteye müssen zur Aufbereitung über die Schnittstelle in die Backend-Datenbank transferiert werden. Dazu wird eine bestehende Schnittstelle des Fremdsystems verwendet.

Nachstehend sind der Aufbau des Backend sowie der Schnittstelle ins Fremdsystem erläutert.

#### Datenfluss

Für jeden anzuzeigenden Service wird eine Schnittstellenabfrage durchgeführt, um den aktuellen Status des entsprechenden Services zu bekommen, welcher in Neteye gespeichert ist. Dieser Vorgang ist im Datenflussdiagramm (Abbildung 6.2) dargestellt.
Die so gesammelten Daten werden durch das Backend gespeichert und können mit Hilfe des Frontends wieder angezeigt werden.

#### Neteye Schnittstelle

Neteye bietet eine Schnittstelle, bei welcher alle aktuell bekannten Informationen über einen beliebigen Service bezogen werden können.

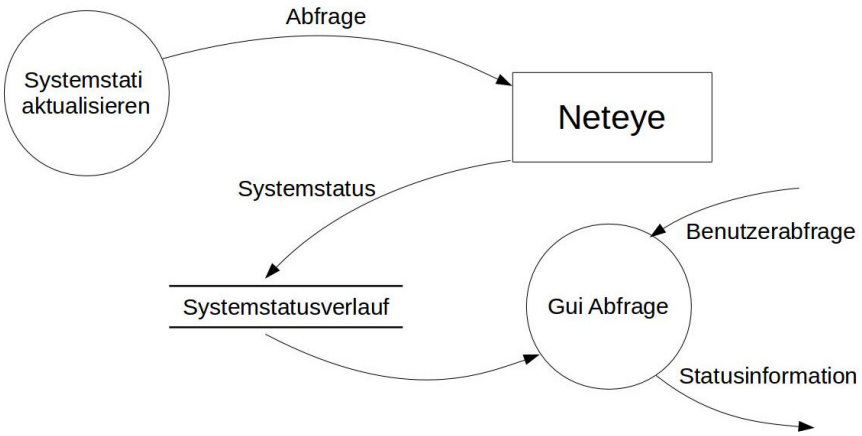


Abbildung 6.2.: Datenflussdiagramm

Die API liefert das Resultat in Form eines JSON-Strings zurück. Die API ist darüber hinaus Passwortgeschützt. Das Kennwort, sowie der Servicename muss also bei jeder Abfrage mitgeschickt werden.

#### Datenbank

Die Servicestati werden in der Datenbank nach Area und danach nach Check aufgeteilt. Area gibt das Gebiet an woher die Testresultate kommen (z.B. ZH).
Der Check selbst beinhaltet alle Zustände der abgefragten Services.

```
<areas>
    <area>
        <areaname></areaname>
        <Check>
            <Date></Date>
            <Service>
                <Servicename></Servicename>
                <Performance></Performance>
                <Infotext></Infotext>
                <State></State>
            </Service>
        </Check>
    </area>
</areas>
```

### 6.3.2. Frontend

Das Frontend zeigt die in der Datenbank abgelegten Daten an.
Die Datenbankabfragen gehören ins Frontend und sind aufgrund der Datenbankstruktur vorgegeben.

Um den aktuellsten Status zu erhalten wird die folgende Query verwendet.

```
doc("DB")//areas/area[areaname="Region"]//Check
    order by $checks/Date descending
```

Um die vergangenen Fehlerfälle auszulesen, wird das Resultat zusätzlich auf den Service/-State>0 geprüft und das Check-Datum limitiert.

```
doc("DB")//areas/area[areaname="Region"]//Check[Date>""]//Service[State>0]
    order by $checks/Date descending
```

## 6.4. Abschluss Konzept

Das konkretisierte Konzept eines Prototypen, welcher Statusinformationen der IT-Infrastruktur vom Neteye ausliest und in einem nativen XML-Datenbankmanagement System abspeichert, konnte erstellt werden, und kann darauf aufbauen im nächsten Kapitel „Implementation“ umgesetzt werden.

# 7 Implementation

In diesem Kapitel werden die wichtigsten Eckpunkte der Umsetzung des Prototypen beleuchtet. Dabei wird neben der verwendeten Technologie auch auf Knackpunkte bei der Umsetzung eingegangen.

## 7.1. Technologie Stack

Um den Prototyp zu implementieren wird auf Software und Framewoks von Dritten zurückgegriffen. In den folgenden Kapiteln werden die Funktion und der Auswahlgrund der Fremdsoftware erläutert.

Tabelle 7.1.: Technologie Stack

| Software | Beschreibung/Auswahlgrund |
|---|---|
| **PHP** | PHP ist eine der am weitesten verbreitetsten Programmiersprachen im Bereich der Web-Entwicklung. Aufgrund der hervorragenden Kompatibilität mit bestehenden Systemen wird PHP verwendet. |
| **eXist** | eXist ist ein sehr bekanntes und einstiegsfreundliches XML DBMS. Darüber hinaus ist es wegen seiner freien Verfügbarkeit für dieses Projekt bestens geeignet.([2], [1]) |
| **BootStrap** | Bootstrap ist das bekannteste CSS Framework der letzten Jahre. Eine sehr grosse Community, viele Plugins und grosse Flexibilität erlauben es schnell ansprechende UIs zu erstellen. |
| **Apache** | Apache ist neben Nginx der am meisten verwendete Webserver, der sich durch seine hohe Konfigurierbarkeit und Stabilität auszeichnet. Aufgrund der Vorkenntnisse des Studenten im Bereich der Konfiguration von Apache, hat er sich für Apache entscheiden. |

## 7.2. Entwicklung

In diesem Abschnitt sind die Knackpunkte der Umsetzung aufgeführt. Neben der Schnittstelle zum Neteye sind auch die einzelnen Layer des Backends beschreiben.

### 7.2.1. Connect to Neteye

Die von Neteye zur Verfügung gestellte API ermöglicht die Abfrage des gesamten Status jedes Services. Im Folgenden sind sowohl Abfrageparameter als auch Rückgabeparameter beschrieben. Die hinterlegte Benutzerkennung wurde eigens für diese Abfragen angelegt.

#### Abfrageparameter

Die Web-API ist unter der URL http://[*ServerName*]/thruk/cgi.bin/status.cgi erreichbar. Sowohl die aufgeführten Get- als auch alle Post-Parameter sind für eine Abfrage anzugeben.

Tabelle 7.2.: Get-Parameters

| Get-Parameter | Beschreibung |
|---|---|
| **dfl_s0_type** | Art des Services<br>„*service*“ oder „*host*“ |
| **dfl_s0_value** | Name des Services<br>„*bp_SAP*“ oder „*bp_DataCenter01*“ etc. |
| **view_mode** | Format der Rückbage<br>„*json*“ |

Tabelle 7.3.: Post-Parameters

| Post-Parameter | Beschreibung |
|---|---|
| **username** | Benutzername der Benutzerkennung<br>„*Neteye-Benutzer*“ |
| **password** | Kennwort der Benutzerkennung<br>z.B. „*Secret$258!*“ |

### 7.2.2. Modulare Implementation

Das Backend ist, wie nachfolgend beschrieben, Layer orientiert implementiert. Dadurch wird eine Separation der Zuständigkeit erreicht und die Wartbarkeit wird deutlich erhöht.

#### DB-Layer

Der DB-Layer verfügt über insgesamt drei öffentliche Funktionen (public functions). Diese drei Funktionen sind in der Tabelle „öffentliche Funktionen“ beschrieben.

Tabelle 7.4.: öffentliche Funktionen

| Funktionsname | Beschreibung |
|---|---|
| **readstate** | Die Funktion gibt ein Array mit allen aktuellen Systemstati zurück.<br>*Parameter: Region* |
| **readpaststate** | Die Funktion gibt ein Array mit den aktuellsten Fehlerfällen pro Service der vergangenen 48 Stunden zurück.<br>*Parameter: Region* |
| **insert** | Die Funktion fügt einen neuen Systemstati der Datensammlung hinzu. |

Die Abfragen auf die Tabelle werden mit einem einfachen Xquery-select durchgeführt. [3]

(for $checks in doc(„*TabellenName*“)//areas/area[areaname=„*Region*“]//Check
order by $checks/Date descending
return $checks)[1]'

#### Logic-Layer

Bei der Ausführung der update.php wird für jeden hinterlegten Service der aktuelle Stati beim Neteye erfragt und der Datensammlung hinzugefügt.

#### Konfigurations-Layer

Die gesamte Konfiguration ist in der Datei *include/settings* hinterlegt.
Im Konfigurations-Layer werden die abzufragenden Services hinterlegt.

## 7.3. Grafische Umsetzung

Die Grafische Oberfläche ist zweigeteilt. (Abbildung 7.1) Im oberen Bereich sind die aktuellen Zustände der Services sichtbar. Im unteren Bereich, mit der Überschrift „Fehlerfälle“ sind die Fehlerfälle der vergangenen 48 Stunden aufgeführt.

Überprüft am Thu 16. Apr 2015, 16:34

| | |
|---|---|
| Datacenter | ok |
| SAP | ok |
| Fileserver | 19 TOTAL 18 OK 1 WARNING<br>WETSRVFILE01[DISKSPACE E:]=WARNING (e:\\ - total: 600.00 Gb - used: 563.12 Gb (94%) - free 36.88 Gb (6%)) |
| Netzwerk | ok |
| SVN-Server | ok |
| Entwicklung-Server | ok |
| SAP-Users | 8 Benutzer eingeloggt |
| Energy Consumption | 4458 W/h |

Fehlerfälle

| | |
|---|---|
| Fileserver Thu 16. Apr 2015, 16:34 | 19 TOTAL 18 OK 1 WARNING<br>WETSRVFILE01[DISKSPACE E:]=WARNING (e:\\ - total: 600.00 Gb - used: 563.12 Gb (94%) - free 36.88 Gb (6%)) |
| SAP Thu 16. Apr 2015, 16:30 | 90 TOTAL 89 OK 1 WARNING<br>WETSRVSAP01[SAP: Used Data Space]=WARNING (Used Data Space=89%) |

Abbildung 7.1.: Übersicht

# 8
# Review und Bewertung

## 8.1. Überprüfung

Die Überprüfung, ob durch den Prototypen ein Mehrwert erzielt wurde, soll anhand der beiden nachstehenden Fragen überprüft werden.

**SAP funktioniert nicht richtig!!!**
Ohne die in dieser Arbeit geschaffene Statusübersicht, musste der Mitarbeiter der IT-Abteilung anrufen und nachfragen, ob das Problem bereits festgestellt wurde. Nun kann er aber selbst erkennen, dass das Problem bereits vom System detektiert wurde und muss nur noch in dringenden Fällen die IT kontaktieren.

**Gestern hat das doch auch schon nicht funktioniert?**
Die Statusübersicht gewährt eine Übersicht der vergangenen 48 Stunden. Jeder Mitarbeiter kann nachvollziehen, wann das letzte mal ein Fehler bei einem System aufgetreten ist. Missverständnisse wie „alles war kaputt“ oder „das hat gestern aber nicht funktioniert“ sind ausgeräumt. Erklärungen und Rechtfertigungen seitens der IT, wie „doch, doch - das hat gestern funktioniert“ entfallen komplett.

Die geforderte Vereinfachung des Zugriffs auf Statusinformationen der IT-Infrastruktur konnte erreicht werden. Informationen sind auch für ungeschultes Personal verständlich und einfach erreichbar.

# 9

# Fazit und Schlusswort

Im Verlaufe dieser Seminararbeit durfte ich mich intensiv mit XML, XML Datenbanksystemen sowie der Abfragesprache xQuery auseinandersetzten und so einen Einblick in eine wenig verbreitete Technologie erlangen. Neben meinem persönlichen Interesse daran kann der Arbeitgeber nun den erstellten Prototypen einsetzten und so echte Probleme lösen.

Ich konnte im Verlaufe des Projekts alle gesetzten Ziele erreichen und alle geforderten Resultate liefern. Ich konnte weiter aufzeigen, dass das Datenhaltungsproblem mit einem XML-DBMS lösbar ist.

Da dies mein erstes Projekt mit einem Schwerpunkt im Bereich der XML Datenbanken ist, bin ich stolz auf den Erfolg dieses Projekts. Es hat mir viel Spass bereitet neue Technologien zu erkunden und kennenzulernen.

# A

# Anhang

## A.1. Quellenverzeichnis


[1] *eXist DB*. `http://www.exist-db.org/exist/apps/homepage/index.html`. [Online, accessed 24-Marc-2015]. 2015.

[2] *eXist PHP*. `http://query-exist.sourceforge.net/`. [Online, accessed 24-Marc-2015]. 2015.

[3] *xQuery*. `http://en.wikibooks.org/wiki/XQuery`. [Online, accessed 24-Marc-2015]. 2015.


## A.2. Tabellenverzeichnis



## A.3. Abbildungsverzeichnis