Technische Hochschule
Brandenburg
University of
Applied Sciences

Fachbereich Informatik und Medien

Bachelorarbeit

Zum Thema

Konzept und Entwicklung einer Software zur Verwaltung der Infrastruktur eines mittelgroßen Entwicklerteams

von Peter Fredebold

zur

Erlangung des akademischen Grades

Bachelor of Science

(B.Sc.)

Unter Betreuung von:

Prof. Dr.-Ing. Martin Schafföner

Technische Hochschule Brandenburg

und

Dipl.-Ing. Andre Huge, Engineering Manager

NETSCOUT Berlin GmbH & Co. KG

Inhalt

1. Einleitung

Unsere moderne Welt entwickelt sich immer schneller weiter und Entwicklungsteams müssen sich an dieses Tempo anpassen. Dazu gehört nicht nur eine gute Projektplanung und gute Kommunikation, sondern auch ein gutes Infrastrukturmanagement. Wächst die Zahl der benötigten Server und Testumgebungen, so wird es unvermeidlich, ein übersichtliches, zentrales System aufzubauen, um nicht die Übersicht zu verlieren. Sei es ein Web- oder Mailserver, Versionskontrollhost, eine Buildumgebung oder eine virtuelle Maschine zum Testen der neusten Funktionen: immer mehr Maschinen mit eigener IP-Adresse, Namen und Zugangsinformationen sind selbst in den kleinsten Unternehmen anzufinden. Sind es mehr als zwei oder drei, wird es wichtig, sich zu organisieren.

1.1. Problem

Das Unternehmen, in dem mein Praktikum absolviert wurde, hatte etwa 30 physische Server und mehr als 50 virtuelle Maschinen, deren Spezifikationen, Nutzungsdaten u.Ä. alle in einer zentralen Excel-Arbeitsmappe abgelegt waren. Dies funktionierte zuvor hervorragend, da nicht sehr viele Einträge in der Tabelle nötig waren und für jeden Eintrag lediglich sehr wenige Informationen gespeichert wurden. Dies änderte sich jedoch mit der Zeit, so sollten nach und nach mehr Informationen wie der „Besitzer“ einer Maschine, die aktuelle IP-Adresse, Login-Informationen, Inventarisierungsnummern und mehr abgelegt werden. Da nicht alle Informationen sowohl für Server als auch Virtuelle Maschinen vorhanden sind, wurde die Excel-Arbeitsmappe um mehrere Tabellen erweitert und versucht, Ordnung ins Chaos zu bringen.

Mit der wachsenden Datenmenge stellte sich jedoch heraus, dass eine Excel-Arbeitsmappe der Aufgabe der effizienten und einfachen Inventarisierung nicht gewachsen ist. So werden alte Einträge vergessen oder übersehen, neue an der falschen Stelle eingetragen und zum Teil kam es sogar zu Bearbeitungskonflikten, wenn mehrere Entwickler gleichzeitig Daten aktualisieren wollten.

Das Unternehmen setzt verstärkt auf virtuelle Ressourcen bzw. Maschinen, die dynamisch erzeugt, installiert und Entwicklern für die Projektaufgaben zur Verfügung gestellt werden. Diese hohe Änderungsgeschwindigkeit konnte in dem zentralen Excel-Dokument nicht abgebildet werden.

Dieses Projekt soll diese Probleme beheben und eine neue Lösung implementieren, die auch neue Anforderungen ermöglicht. Im nächsten Kapitel sind diese Anforderungen definiert.

2. Anforderungen

Die Anforderungen sind neben der thematischen Einteilung in funktionale und Benutzeroberflächen-Anforderungen priorisiert als „mandatory“, also unabdingbar benötigt, und „nice-to-have“, also nicht unbedingt sofort benötigt, eher interessant zu haben. Dies ist markiert mit „-M“ und „-NTH“ in der Kennung.

2.1. Funktionale Anforderungen

REQ-F000-M: Die Lösung soll die Attribute der Entwicklungsressourcen in einer zentralen Datenbank speichern.

REQ-F010-M: Folgende allgemeine Attribute für Entwicklungsmaschinen sind zu erfassen.

- Name
- Einsatzzweck
- Benutzer
- IP-Adressen
- Anzahl CPUs/Threads
- RAM
- Login-Informationen
- Betriebssystem

REQ-F020-M: Folgende zusätzliche Attribute für physische Server sind zu erfassen.

- Service-Tag
- Kaufdatum
- Festplattenspeicher
- Modell

REQ-F030-M: Folgende zusätzliche Attribute für virtuelle Maschinen sind zu erfassen.

- Host

REQ-F035-M: Die Lösung muss die verwendeten Hypervisoren unterstützen.

- KVM
- Hyper-V
- VirtualBox
- VMWare

REQ-F040-M: Die Lösung soll auf folgenden Betriebssystemen lauffähig sein.

- Windows Server (2008R2 +) (Priorität 1)
- Linux (mindestens Ubuntu 16 und CentOS 7) (Priorität 2)

REQ-F050-M: Die Datenbank-Inhalte müssen täglich in einem externen Backup-System gesichert werden.

REQ-F060-M: Die Lösung muss innerhalb von 8 Stunden auf einen neuen Rechner installiert werden können.

REQ-F070-NTH: Die Lösung soll die gespeicherten Informationen mindestens einmal täglich automatisch aktualisieren.

2.2. Benutzerinterface Anforderungen

REQ-UI000-M: Die Lösung soll über ein Web-Interface bedient werden.

REQ-UI010-M: Folgende Web-Browser müssen unterstützt werden.

- Google Chrome (Version 68+)
- Mozilla Firefox (Version 61+)
- Microsoft Internet Explorer (Version 9+)

REQ-UI020-M: Das Benutzerinterface soll spezifische Übersichten/Dashboards anbieten.

- Ressourcen-Statistiken
- Server-Informationen
- VM-Informationen
- VM-Pool

REQ-UI021-M: Folgende Aktionen sollen Ansicht-Übergreifend verfügbar sein:

- Hinzufügen fehlender Ressourcen

REQ-UI030-M: Die Ressourcen-Statistiken soll folgende Information anbieten:

- Anzahl Server
- Anzahl VMs
- Gesamtanzahl CPUs/Threads
- Belegte/Freie Ressourcen

REQ-UI040-M: Die Ressourcen-Statistiken soll folgende Aktionen bereitstellen:

- Drilldown

REQ-UI050-M: Die VM-Pool-Übersicht soll folgende Informationen anbieten:

- Verfügbare und belegte Pool-Ressourcen

REQ-UI060-M: Die VM-Pool-Übersicht soll folgende Aktionen bereitstellen:

- Reservieren von Pool-Ressourcen
- Freigeben von Pool-Ressourcen

REQ-UI070-M: Die Server-Informations-Übersicht soll folgende Informationen anbieten:

- Details zu allen physikalischen Servern, wie in REQ-F010 und REQ-F020 definiert

REQ-UI080-M: Die Server-Informations-Übersicht soll folgende Aktionen bereitstellen:

- Anzeigen der auf dieser Maschine gehosteten VMs
- Bearbeiten der Ressourcen-Informationen
- Anzeigen der Garantie-Informationen, falls verfügbar
- Anzeigen des LOM-Zugangs, falls verfügbar

REQ-UI090-M: Die VM-Informations-Übersicht soll folgende Informationen anbieten:

- Details zu allen Virtuellen Maschinen, wie in REQ-F010 und REQ-F030 definiert

REQ-UI100-M: Die VM-Informations-Übersicht soll folgende Aktionen bereitstellen:

- Bearbeiten der Ressourcen-Informationen

REQ-UI101-NTH: Die VM-Informations-Übersicht soll folgende Aktionen bereitstellen:

- Neu starten beziehungsweise Zurücksetzen der VM

2.3. Kapazität und Performance Anforderungen

REQ-P000-M: Die Lösung soll bis zu 200 Ressourcen verwalten können.

REQ-P010-M: Das Web-Interface soll die Informationen innerhalb von 2 Sekunden darstellen können.

3. Konzept

Diese Anforderungen könnten auf verschiedene Arten umgesetzt werden, sowie es für jedes Problem nicht nur exakt eine Lösung gibt. Die Auswahl ist groß, aber einige Lösungsansätze wurden in die engere Auswahl gezogen und gegeneinander abgewägt, um eine passende Lösung zu finden.

3.1. Warum selbstprogrammiert?

Viele Probleme der IT wurden bereits mehrfach gelöst und sind als Anwendungen kommerziell oder kostenfrei erhältlich. Dadurch stellt sich sehr schnell die Frage: Warum sollte eine komplett neue Lösung entwickelt werden anstatt eine bereits existierende zu nutzen bzw. zu kaufen?

Diese Frage haben wir uns ebenfalls gestellt und die folgenden Anwendungen als Möglichkeiten in Betracht gezogen: vSphere, Azure Cloud und OpenStack. Alle drei bieten sie die Möglichkeit, virtuelle Server zu verwalten, jedoch fallen sowohl vSphere als auch Azure Cloud sofort aus dem Rennen, da sie entgegen REQ-F035-M lediglich einen Hypervisor unterstützen; vSphere unterstützt lediglich VMWare und Azure Cloud basiert auf Hyper-V von Microsoft. Da wir eine Lösung benötigen, die sowohl Hyper-V, VMWare, Virtual Box, als auch KVM-basierte Hosts unterstützen, fiel unser Blick auf OpenStack. Dies wurde testweise aufgesetzt, jedoch stellte sich sehr schnell heraus, dass OpenStack, wie die anderen beiden Lösungen, auf größere Daten- beziehungsweise Rechen-Zentren ausgelegt ist und für unsere vergleichsweise kleineren Anforderungen ungeeignet ist. Ebenfalls sind diese Lösungen mehr auf die Überwachung von virtuellen Maschinen ausgelegt und weniger auf die Verwaltung aller Maschinen eines Netzes und Auflistung derer Informationen.

Aus diesen Gründen und auf Grund der Tatsache, dass wir eine stärkere Anpassung an unsere Vorstellungen wünschten, haben wir uns dazu entschieden, eine eigene Lösung zu entwickeln.

Warum nun welche Bausteine genutzt wurden, sei in den folgenden Abschnitten erläutert.

3.2. Architektur

Gemäß den Anforderungen, vor allen REQ-F000-M und REQ-UI000-M, wurde sich für eine Web-Applikation mit zugrundeliegender Datenbank entschieden, welche manuell oder automatisiert aktualisiert werden kann. Der Endnutzer kann durch das bereitgestellte Webinterface auf die aufbereiteten Daten zugreifen. Die Datenbank kann sowohl auf der selben Maschine wie der Webapplikation liegen, oder auch an einem beliebigen anderen Ort, der durch einen normalen Dateisystemzugriff ermöglicht wird; Zum Beispiel auch auf einer über das Netzwerk geteilten Festplatte. Dies erlaubt es, die Applikation samt Daten sowohl unkompliziert umzuziehen (REQ-F060-M) als auch durch einfache Dateisystem-Backups zu sichern. (REQ-F050-M)

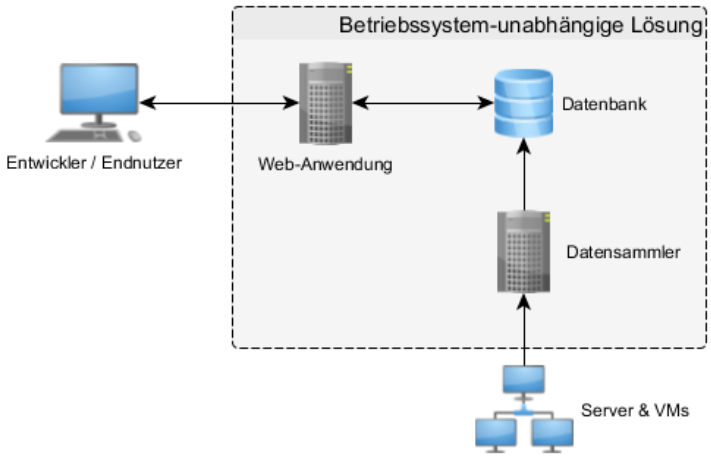

Abbildung 1: Visualisierung der Architektur

Da die Applikation lediglich intern erreichbar sein soll, sind alle Informationen ohne weitere Sicherheitsprüfung lesbar, zum Ändern wird jedoch ein Passwort erfragt, um ungewollten Änderungen vorzubeugen.

Der „Datensammler“ ist in dieser Architektur beabsichtigt als ein einzelner Baustein zu sehen, da dieser Unabhängig der eigentlichen Web-Anwendung ist, die im Zentrum des Projekts steht, und einen eigenen Lebenszyklus hat. So können die gesammelten Informationen periodisch aktualisiert werden, ohne eine Unterbrechung der Web-Applikation hervorzurufen. Ebenfalls kann der „Datensammler“ nicht nur als eigenständiges Programm, sondern auch als menschlicher Prozess angesehen werden, der regelmäßig manuell geschieht. Ebenfalls

ermöglicht die Abtrennung eine gezielte Aktualisierung der „Datensammler“-Komponente, ohne das vollständige Programm aktualisieren zu müssen.

3.3. Web-Anwendungs-Frameworks

Im Zentrum des Projektes steht jedoch die Web-Anwendung, die gemäß REQ-UI000-M genutzt werden soll. Dies dient dem Zwecke, dass die Anwendung sowohl unabhängig vom Betriebssystem des Nutzers ist, als auch ohne größeren Aufwand wie der Installation einer Client-Software genutzt werden kann. Um den Aufwand weiter zu reduzieren, soll die Applikation gemäß REQ-UI010-M mit allen gängigen Webbrowsern nutzbar sein – so muss kein Nutzer eine ungewohnte Umgebung nutzen, nur um auf die Applikation zuzugreifen.

Zur Umsetzung eben dieser Web-Applikation wurden im Laufe der Entwicklung die nachfolgenden Lösungsansätze in Betracht gezogen.

3.3.1. Java Server Faces

Java ist weit verbreitet und hat somit Einzug sowohl in den Studiengang Informatik an der TH Brandenburg als auch das Hause NETSCOUT gefunden; Im ersteren wurde das Konzept von auf Java Enterprise Edition bzw. Java Server Faces basierten Webanwendungen gelehrt, im letzteren ist dies bereits in einer eigenen Applikation genutzt. Aus diesem Grund wurde dieser Ansatz für das Projekt in Betracht gezogen.

Frameworks wie „Wildfly“ (ehemals JBoss) ermöglichen es, Webanwendungen zu gestalten, die auf Objektorientierter Logik basieren und integrieren sich mit vielen gängigen Datenbanken und geben die Möglichkeit, auf einem hohem Abstraktionslevel zu arbeiten. Hierdurch wäre eine spätere Erweiterung durch andere Entwickler stark vereinfacht, jedoch ergibt sich ebenfalls ein erhöhter Anfangsaufwand.

Trotz dieser Vorteile wurde sich gegen die Nutzung von Java-Frameworks entschieden, da diese einen hohen installations- und konfigurations-Overhead generieren, also entgegen REQ-F060-M nicht sonderlich schnell auf einem neuen System installierbar sind und nach erfolgreicher Installation aller benötigten Java-Software und Anwendungs-Server muss stets eine stark System-abhängige Konfiguration erfolgen. Zwar ist es in der vorgegebenen Zeit von 8 Stunden sehr wohl möglich, aber aus Erfahrung im Studium und im Unternehmen heraus ergeben sich oftmals zuvor unvorhergesehene Probleme, welche häufig frustrierend und nur zeitaufwändig zu beheben sind.

Ein abschließender Grund, aus dem sich stark gegen jegliche Java-basierte Lösungen entschieden wurde, ist die ungewisse beziehungsweise nicht mehr kostenlos verfügbare Zukunft von Java, wodurch es zu eventuellen Problemen kommen könnte, die durch eine nicht-Nutzung sehr einfach vermieden werden können. (Oracle 2018)

3.3.2. ASP.NET

Eine (laut Q-Success 2018) noch häufiger genutzte Lösung zur Web-Applikations-Erstellung ist ASP.NET. Im Gegensatz zu Java-basierten Lösungen sind keine Pläne bekannt, die momentan komplett kostenlose Lösungsmöglichkeiten abzuschaffen.

ASP.NET basiert auf dem .NET Framework und ist dank „.NET Core“ seit einiger Zeit auch auf allen nach REQ-F040-M erforderlichen Betriebssystemen lauffähig. In einigen Fällen wäre hier die Bereitstellungsmöglichkeiten dank Microsoft IIS bereits im Betriebssystem verankert, wodurch eine Neuinstallation auf einem Server mit Windows Server definitiv im durch REQ-F060-M angegebenen Zeitraum von 8 Stunden möglich wäre. Auf Linux-basierten Servern wäre der Aufwand vergleichsweise größer, da dort zunächst die besagte „.NET Core“-Installation vorgenommen werden muss und weitere Konfigurationsschritte benötigt werden.

Dank des .NET-Frameworks, welches in C# genutzt werden kann, ist es für diesen Ansatz ebenfalls gut möglich, auf einer sehr hohen Abstraktionsebene zu arbeiten, welches erneut die in 3.3.1 für Java genannten Vor- und Nachteile hat. Ein Nachteil von C# ist jedoch, dass die offiziellen Tools zum schreiben und kompilieren von Applikationen zum kommerziellen Gebrauch nicht für die gewollten Ziele kostenlos verfügbar sind und somit entsprechende Kosten entstehen oder auf andere Entwicklungslösungen zurückgegriffen werden muss. Zusätzlich ist der vorhandene Wissensgehalt in der Firma und persönlich bei ASP.NET deutlich kleiner als bei einer Java-basierten Lösung, weshalb ebenfalls dieser Ansatz letztendlich verworfen wurde.

3.3.3. NodeJS

Im Kontrast dazu sind unter meiner Leitung bereits Projekte in NodeJS entstanden, wodurch es sich ebenfalls als ein möglicher Lösungsansatz anbot. Anstatt einer kompilierten Programmiersprache wie C# oder Java nutzt NodeJS JavaScript, welches zur Laufzeit interpretiert wird, wodurch der teilweise langsame Prozess des Kompilierens während der Entwicklung entfällt.

NodeJS an sich ist JavaScript, welches ohne einen Browser ausgeführt wird. Ähnlich der anderen beiden vorgestellten Ansätze existieren für NodeJS diverse Frameworks, die es erleichtern sollen, Web-Applikationen zu erstellen und eine gewisse Abstraktionsebene zu erstellen. Dies hätte erneut die Vorteile, für spätere Entwickler einfacher erweiterbar zu sein, aber dafür einen höheren Aufwand zu Beginn des Projektes zu haben, weshalb dieser Ansatz ebenfalls nicht genutzt wurde.

Trotzdem war für das Projekt NodeJS die gewählte Antwort, jedoch ohne ein zusätzliches Framework, da, wie bereits erwähnt, mit dieser Art der Web-Anwendung bereits zuvor gearbeitet wurde. Diese Auswahl wurde ebenfalls aus der Begründung heraus getroffen, dass eine Neuinstallation sowohl unter Windows als auch Linux mit den anderen Ansätzen verglichen sehr einfach und schnell möglich ist. Dies ist ermöglicht durch im Projekt enthaltene Skripte, welche nutzbar sind dank des „Node Package Managers“, (kurz npm) welcher in einer standardmäßigen NodeJS-Installation mitgeliefert wird. Ist dieser installiert, so lässt sich eine NodeJS-basierte Lösung mit einem einfachen Befehl starten und besagte zusätzliche Skripte können ausgeführt werden.

3.4. Datenbanken

Die in der Web-Applikation anzuzeigenden Daten müssen in irgendeiner Art abgelegt werden, damit sie ohne größeren Aufwand, wie dem konstanten aktualisieren der Informationen, bereitgestellt werden können. Dies könnte auf viele Arten geschehen, wie dem Ablegen in Text-Dateien oder einem Cloud-basierten Speicher, doch eine konventionelle relationale Datenbank ist für diesen Fall das angemessenste, da diese stets konsistent, schnell verfügbar und durch einen gut dokumentierten Standard, SQL, ansprechbar ist.

So wurden aus diversen Gründen, welche im Folgenden gelistet und erklärt sind, die folgenden Datenbank-Management-Systeme (DBMS) ausprobiert.

3.4.1. PostgreSQL

PostgreSQL wurde auf Grund vorheriger Nutzung zunächst gewählt. PostgreSQL ist ein modernes DBMS, welches anstrebt, den Spezifikationen von SQL sehr genau zu folgen:

„PostgreSQL development aims for conformance with the latest official version of the [SQL] standard where such conformance does not contradict traditional features or common sense.“

(PostgreSQL Global Development Group 2018)

Es ist eine quelloffene Implementierung, die äußerst viele verschiedene Datentypen anbietet. Für das Projekt am interessantesten hierbei ist die native Unterstützung des Datentyps „inet“, welcher IP-Adressen effektiv ablegt und diese indexieren lässt. Da das Projekt für jeden Server und jede VM mehrere IP-Adressen ablegen soll, ist es hilfreich, diesen Datentypen zu haben.

Jedoch existierten bis zum ersten Prototypen des Projektes keinerlei Instanzen von PostgreSQL, die hätten genutzt werden können. Dies würde die Neuinstallation der Applikation auf einem neuen Server ungemein verlängern, da eine solche PostgreSQL-Datenbank ebenfalls von Grund auf installiert und konfiguriert werden müsste. Aus diesem Grund wurde von PostgreSQL abgesehen.

3.4.2. MySQL

Im Gegensatz dazu wurde MySQL vorgeschlagen. Dank der stärkeren Verbreitung von MySQL im Vergleich zu PostgreSQL (solid IT GmbH 2018) existierten bereits mehrere aktive MySQL-Server, welche für die Zwecke des Projektes genutzt werden könnten.

MySQL ist ebenfalls Quelloffen verfügbar, allerdings unter der GPL Lizenz und mit der Möglichkeit, eine Unternehmens-Lizenz mit extra Support und weiteren Vorteilen zu erwerben. Für die Nutzung in diesem Projekt ist dies jedoch unerheblich, da es für MySQL und Probleme, die damit auftreten können, mehr als ausreichend Unterstützung durch die große Nutzerbasis erfolgt. Ebenso wird das Projekt nicht kommerziell verkauft, weshalb keinerlei Gründe für eine Lizenz existieren.

Anders als PostgreSQL unterstützt MySQL jedoch keinen Datentypen, welcher IP-Adressen darstellt. Lediglich existieren einige vordefinierte Funktionen, welche IP-Adressen, die als Zahlen abgelegt wurden in menschen-lesbare Zahlenketten und umgekehrt umwandeln können. Dies erhöht den Programmieraufwand.

Ausschlaggebend gegen sowohl PostgreSQL als auch MySQL spricht jedoch, dass es sich bei beiden DBMS um Client-Server-Applikationen handelt. Dies bedeutet, dass ein zusätzliches Programm auf demselben oder einem anderen Server, wie der, auf dem das Projekt läuft, aufgesetzt werden und laufen muss.

3.4.3. SQLite

Im Kontrast benötigt SQLite kein zusätzliches Programm, das läuft; Ziel von SQLite ist es, „light“, also „leicht“ oder „leichtgewichtig“ zu sein. So besteht eine SQLite-Datenbank lediglich aus einer einzelnen Datei, auf welche von allen Prozessen, die diese lesen können,

zugegriffen werden kann. Dieses simple Prinzip beschränkt die Funktionalitäten, sowie die maximalen Ausmaße der Datenbank jedoch erheblich, aber ist für eine Lösung, die keine größeren Ausmaße hat (siehe REQ-P000-M), angemessen, da die Datenbankdatei entweder mit dem Projekt mitgeliefert oder von neu auf erstellt werden kann – keine weiteren Installationen oder Konfigurationen sind notwendig.

Eine SQLite Datei enthält Anweisungen, wie die von ihr enthaltene Datenbank aufgebaut werden kann, in einfachem SQL in Textform, die eigentlichen Daten und einige weitere Informationen sind binärkodiert. Wird eine SQLite Datei in einem einfachen Text-Programm geöffnet, lässt sich dies sehr einfach sehen.

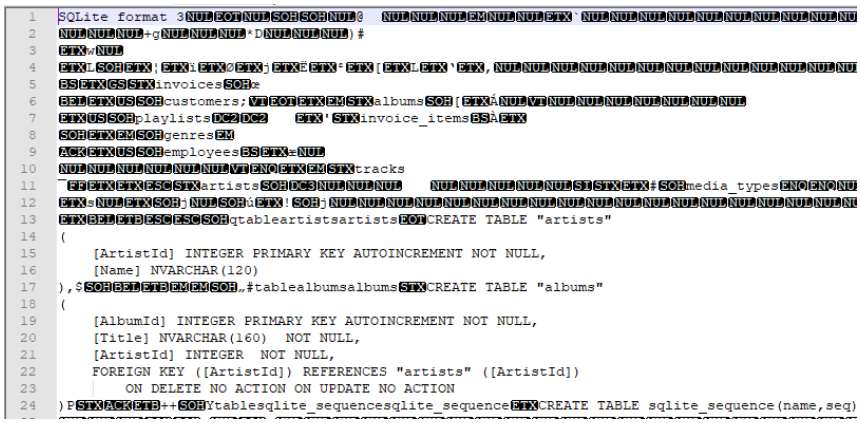

Abbildung 2: Eine SQLite Beispiel-Datenbank in einem Text-Editor

Dies ermöglicht es einem lesenden Prozess, die Datenbank im Hauptspeicher anzulegen bzw. zu speichern und diese dann verändern. Ist eine SQLite-Datei teilweise beschädigt, ist es so ebenfalls eventuell möglich, Teile der Datenbank relativ einfach wiederherzustellen.

Zusätzlich lassen sich SQLite Datenbanken jedoch nur begrenzt in ihrer Struktur verändern, so muss zum Beispiel zum ändern des Datentyps einer Tabellenspalte die komplette Tabelle gelöscht und neu erstellt werden. Ebenfalls erlaubt SQLite lediglich einige wenige Basis-Datentypen, was die Nutzung erheblich einschränkt und eigene Konvertierungsschritte notwendig macht.

Insgesamt ist SQLite jedoch leistungsstark genug, um für das Projekt nahezu perfekt zu sein, da es zwar Programmieraufwand erzeugt, allerdings die spätere Nutzung beziehungsweise Installation und Datensicherung ungemein vereinfacht.

3.5. Darstellung

Mit den zu Grunde liegenden Programmen geklärt, stellt sich die Frage, wie die in der Datenbank abgelegten Daten möglichst passend und leicht für Menschen lesbar gemacht werden können. Da die Applikation über einen Browser aufgerufen wird, ist es klar, dass die Daten über HTTP (HyperText Transfer Protocol) übertragen und mit HTML (HyperText Markup Language) angezeigt werden. Einfache, kleine Datenansammlungen oder Informationen lassen sich mit simpelsten Methoden darstellen, doch ist eine lange Tabelle oder pure, kommagetrennte Auflistung nicht sehr angenehm oder einfach für einen Menschen aufzunehmen. Zwar lassen sich mit einfachem HTML bereits lange Paragraphen visuell gut unterteilen, doch „lebt“ eine moderne Website erst dank visuellen Abtrennungen und stilistisch angepassten Elementen auf. Diese könnten nun alle einzeln in aufwendiger Detailarbeit kreiert werden, doch würde dies sehr schnell den Rahmen des Projektes sprengen. Stattdessen wurde sich dazu entschieden, dass CSS- (Cascading Style Sheets) und JavaScript-Framework „Bootstrap“ zu verwenden.

3.5.1. Bootstrap

Bootstrap zielt darauf ab, responsives Websitedesign stark zu vereinfachen und viele Standardelemente von HTML-Webseiten zu „modernisieren“. Dies erreicht Bootstrap durch viele kleine, aber auch große Anpassungen der Darstellungsinformationen von beinahe allen verfügbaren HTML-Elementen.

Eine der wohl auffälligsten Änderungen ist das Festlegen der Schriftarten auf serifenlose Schriftarten. Auch wenn dies nachweißlich eher unbedeutend ist für die Lesegeschwindigkeit, sprechen viele Personen doch eher auf serifenlose Schriftarten an, (Liebig 2009; Hellbusch 2005) weshalb sogar Applikationen der deutschen Regierung auf eben diese Setzen. (DIN Berlin 2003)

Abbildung 3: Einfache Website, ohne Bootstrap

In diesem Beispiel ist eine simple HTML-Datei in Google Chrome geöffnet zu sehen. Sie enthält keinerlei Darstellungsinformationen und nutzt deshalb eine Serifen-Schrift. Fügt man lediglich Bootstrap hinzu, sieht die ansonsten gleiche Datei wie folgt aus.

Abbildung 4: Einfache Website, mit Bootstrap

Diese grundlegenden Anpassungen zielen darauf ab, das eigentliche „Look and Feel“, das generelle „Aussehen und Gefühl“ der Website unangetastet zu lassen, aber es angenehmer für die Augen zu machen. Im Gegensatz dazu kommt Bootstrap noch mit einigen CSS-Klassen, welche Elementen ein ganz anderes Aussehen geben. So werden Eingabefelder und Knöpfe abgerundet, standardisierte Farben genutzt und viele Elemente einheitlich gestaltet.

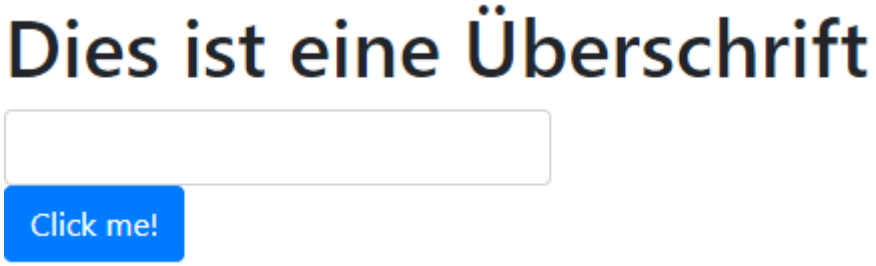

Abbildung 5: Die gleiche Website, mit Bootstrap-Klassen

3.6. Automatisierung

Gemäß REQ-F-070-NTH sollen die gesammelten Daten mindestens einmal täglich automatisch aktualisiert werden. Dies könnte manuell geschehen, würde jedoch sehr viel Zeit jeden Tag kosten und den Aufwand nicht wert sein. Die automatische Aktualisierung wirkt ebenfalls dem Problem entgegen, welches sich in der Originallösung der Excel-Tabelle wiederspiegelte, dass sich selten jemand aktiv kümmerte und Daten damit ihre Korrektheit verloren. Ebenfalls soll es möglich sein, die VMs wenn benötigt neu zu starten. Sowohl das Sammeln der Daten als auch das Steuern der VMs lässt sich auf verschiedene Arten realisieren.

3.6.1. Datensammler

Wie bereits zuvor erwähnt, soll unabhängig von der Web-Applikation ein Programm laufen, welches regelmäßig die bekannten Informationen verifiziert und gegebenenfalls aktualisiert. Die verschiedenen Ansätze, die hierfür in Betracht gezogen wurden, sind im Folgenden erklärt.

3.6.1.1. Via SSH

Alle vorhandenen VMs, Linux-Server und einige wenige der Windows-Server, die das Projekt abdecken soll, unterstützen eine Verbindung über SSH. Da die benötigten Login-Informationen hierfür in der Datenbank abgelegt sind, kann sich ein Datensammlungsprogramm über SSH mit jedem Server verbinden. Ebenfalls kann ein regelmäßiger Scan des bekannten Adressbereiches durchgeführt werden, um eventuell „vergessene“ Server oder VMs zu finden. Einmal eingeloggt, lassen sich unter Linux fast alle gewollten Daten abfragen, indem einfache, auf allen Linux-Distributionen verfügbaren Befehle an den Server gesendet und die Antworten Zeile für Zeile ausgewertet werden.

Folgende, in REQ-F010-M und REQ-F020-M festgelegten Informationen lassen sich wie folgt erfassen:

- IP-Adressen. Auch wenn mindestens eine funktionierende bekannt sein muss, um überhaupt mit SSH auf den Server zugreifen zu können, so ist es vernünftig, alle bekannten abzufragen. Hierfür kann der Befehl `ip addr show up` gut genutzt werden. Er gibt alle verfügbaren Adressen der aktiven Netzwerkkarten zurück. Dadurch können sowohl bereits bekannte Adressen bestätigt, als auch eventuell zuvor unbekannte Adressen in Erfahrung gebracht werden.

```
$ ip addr show up
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 6a:22:22:48:08:cc brd ff:ff:ff:ff:ff:ff
    inet [redacted]/22 brd 188.68.51.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 [redacted]/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.1/24 brd 10.8.0.255 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::2b0e:55f3:3901:8749/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Abbildung 6: Beispielausgabe des Befehls "ip addr show up"

- Festplattenspeicher. Mit zunehmender Nutzungsdauer eines Systems ist es nur natürlich, dass der freie Festplattenspeicher zunehmend gering wird. Um dies präventiv zu bemerken und neue Hardware zu entdecken, kann mit dem Befehl `df -BG --total` der verfügbare und freie Festplattenspeicher ausgelesen werden. Die Option `-BG` dient dazu, die Ergebnisse in Gigabyte anzuzeigen und `--total`, um nicht nur alle verfügbaren Laufwerke anzuzeigen, sondern diese auch zusammenzurechnen.

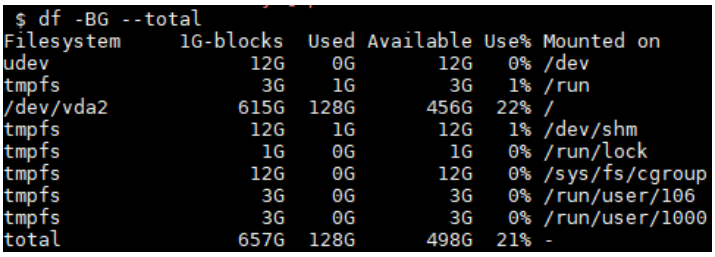

Abbildung 7: Beispielausgabe des Befehls "df -BG --total"

- Verfügbarer Arbeitsspeicher, RAM. Vor allem bei VMs ist es sehr wahrscheinlich, dass sich im Laufe der Lebenszeit der verfügbare Arbeitsspeicher verändert, da sich die Anforderungen an diese ändern können und der Arbeitsspeicher vom Hypervisor ohne größere Umstände neu zugewiesen werden kann. Um diese Änderungen dokumentiert zu halten, ist der Befehl `free` nutzbar, welcher sowohl den verfügbaren, als auch den genutzten Arbeitsspeicher und Details über die Auslagerungsdatei zurückgibt.

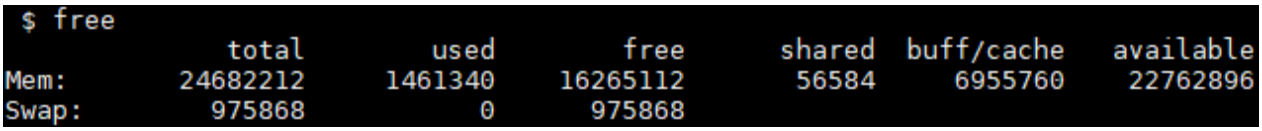

Abbildung 8: Beispielausgabe des Befehls "free"

- Betriebssystem. Da verschiedene Linux-Distributionen genutzt werden, ist es ebenfalls relevant, welche Version und welche Distribution die jeweiligen Server nutzen. Neben dem Distributionsnamen, Linux Version und Hostnamen gibt der Befehl `uname -a` außerdem Auskunft über die Architektur und Datum der Kompilierung, welche jedoch nicht aufgezeichnet werden.

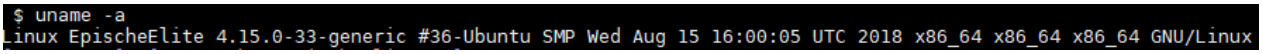

Abbildung 9: Beispielausgabe des Befehls "uname -a"

- Anzahl der CPUs/Threads sowie Hypervisor. Eine Angabe, die sich unter normalen Umständen nicht ändert, aber für neue Systeme zu Leistungszwecken erfasst werden

sollte. Hierbei hilft der Befehl `lscpu`, welcher sehr viele Details zur genutzten CPU bereitstellt, inklusive der verfügbaren Kerne und des Hypervisors.

```
$ lscpu
Architektur:             x86_64
CPU Operationsmodus:     32-bit, 64-bit
Byte-Reihenfolge:        Little Endian
CPU(s):                  8
Liste der Online-CPU(s): 0-7
Thread(s) pro Kern:      1
Kern(e) pro Socket:      1
Sockel:                  8
NUMA-Knoten:             1
Anbieterkennung:         GenuineIntel
Prozessorfamilie:        6
Modell:                  63
Modellname:              Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
Stepping:                2
CPU MHz:                 2297.338
BogoMIPS:                4594.67
Hypervisor-Anbieter:     KVM
Virtualisierungstyp:     voll
L1d Cache:               32K
L1i Cache:               32K
L2 Cache:                4096K
L3 Cache:                16384K
NUMA-Knoten0 CPU(s):     0-7
```

Abbildung 10: Beispielausgabe des Befehls "lscpu"

- Zuletzt ist es ebenfalls möglich, an einem vordefinierten Ort eine einfache Text-Datei anzulegen, welche weitere Informationen wie Nutzungszweck und Besitzer enthalten kann. Beim Scannen kann diese dann ausgelesen werden.

All diese Befehle können vom Datensammler ausgeführt und ausgewertet werden. Um die Auswertung zu vereinfachen, kann die Ausgabe der Befehle durch ein `grep` Skript gefiltert werden. Zum Beispiel kann so die sehr lange Liste von `lscpu` gekürzt werden.

3.6.1.2. Via Redfish

Die meisten Server, die vom Projekt verwaltet werden sollen bieten durch ihr Lights Out Management-Interface eine sogenannte „Redfish API“ zur Verfügung. Diese REST-API bietet diverse Informationen zu den Systemen an, welche in der dazugehörigen Server-Chassis installiert sind. Dies ermöglicht es, mit simplen HTTP(S)-Anfragen anstatt einer vergleichsweise komplizierten SSH-Verbindung zu arbeiten, welches zu deutlich weniger Fehlern führt. Ebenfalls ist die Redfish API über alle Systeme hinweg genormt, weshalb weniger Tests durchgeführt und weniger Anpassungen programmiert werden müssen.

Zusätzlich erlaubt es Redfish nicht nur, im eingeschalteten Zustand Informationen auszulesen, sondern auch, sogar wenn das Host-System ausgeschaltet ist, diverse Steuerungsaktionen auszuführen. Des Weiteren lassen sich neben den über SSH auslesbaren Eigenschaften Details wie die Seriennummer bzw. der Service-Tag und Modell-Nummer des Servers auslesen, welche

innerhalb des Betriebssystems lediglich nach der Installation von Zusatz-Software verfügbar wären.

Redfish ist jedoch ebenfalls nicht universell verfügbar, da diese API erst seit 2015 existiert, der Datensammler aber ebenfalls mit Servern von 2013 kommunizieren soll.

3.6.1.3. Via SNMP

Im Gegensatz zu Redfish ist das „Simple Network Management Protocol“ selbst in der neusten Version von 2002 (RFC3410 2002) älter und weitverbreiteter. Ebenso wie Redfish bieten die Lights Out Management-Interfaces der Server Informationen mithilfe dieses Protokolls an, aber im Vergleich ist es nicht nur in einigen, sondern allen Servern des Projektes aktivierbar. Aus diesem Grund ist eine Abfrage der Daten über SNMP ein guter letzter Ausweg, sollten die in den Anforderungen definierten Informationen nicht über die anderen beiden Wege erhalten worden sein.

3.6.2. Steuerung

Neben dem Sammeln von Daten soll es ebenfalls möglich sein, die VMs zu steuern; Sie mindestens gemäß REQ-UI101-NTH neu zu starten. Dies ist für Server bereits dank Redfish möglich, diese REST-API hat jedoch keinen Zugriff auf die VMs, die auf diesen Systemen laufen, also wird ein weiterer Weg benötigt. Da mit verschiedenen Hypervisoren gearbeitet wird, werden ebenfalls mehrere Wege zum neustarten benötigt. Weil diese Funktionalität allerdings nur als „Nice to have“ eingestuft wurde, wurde sich zunächst lediglich mit Hyper-V und KVM beschäftigt, da ausschließlich wenige VMs unter anderen Hypervisoren laufen.

3.6.2.1. KVM

KVM unter Linux lässt sich über die Kommandozeile steuern. Auf diese kann, wie in 3.6.1.1 bereits erwähnt, mit Hilfe einer SSH-Verbindung zugegriffen werden. In der Datenbank ist der Name der VM abgelegt, welche mit dem Namen, der im Hypervisor genutzt wird übereinstimmen muss. Dies kann dann genutzt werden, um die entsprechende VM zu steuern.

Dafür wird das Kommandozeilen-Tool `virsh` genutzt. Es ermöglicht die komplette Steuerung von vorhandenen VMs auf dem verbundenen Host-System, aber das einzige, was momentan interessant ist, ist die Option `reset` – hiermit lässt sich das drücken des RESET-Schalters der Maschine emulieren, welches das Gastsystem sofort ausschaltet und neu startet. Normalerweise wäre das Fehlen einer normalen Abschalt-Routine unerwünscht, da so nicht gespeicherte Daten verloren gehen, doch dies wird für das Projekt nicht benötigt. Sofern das Gastsystem noch

normal erreichbar ist, kann ein Neustart innerhalb des Gastbetriebssystems initiiert werden; Diese externe Steuerung soll lediglich für Fälle aufkommen, in denen sich das Gastsystem aufgehangen hat.

3.6.2.2. Hyper-V

Das Zurücksetzen einer VM unter Hyper-V ist leider nicht so simpel wie der Fall unter KVM, aber es ist möglich. Hyper-V läuft unter Windows, also muss eine Verbindung zu diesem Windows-Host geschehen, was über SSH normalerweise nicht möglich ist. Ist jedoch sowohl das Zielsystem als auch das System, auf dem das Projekt läuft, korrekt konfiguriert, d.h. sie kennen sich beide als so genannte „Trusted Hosts“, so ist es möglich, über PowerShell eine Verbindung aufzubauen. Ist dies geschehen, so erlaubt es ähnlich zu KVM, eine bekannte VM gezwungen neu zu starten.

Mit Hilfe des PowerShell-Befehls `Invoke-Command` und dem Parameter `-Computername` kann ein beliebiger Befehl auf dem VM-Host ausgeführt werden. Dies, kombiniert mit dem Befehl `Restart-VM` ermöglicht das gewünschte Ergebnis.

4. Umsetzung

Kein Software-Projekt ist perfekt im ersten Anlauf. Ebenso war es mit diesem Projekt. Das grobe Gesamtkonzept stand bereits zu Beginn, aber die alte Lösung der Excel-Arbeitsmappe benötigte einen möglichst schnellen, funktionierenden Ersatz. Aus diesem Grund entstanden die ersten Versionen des Projekts möglichst schnell mit einem verringerten Funktionsumfang. Die eigentlichen Anforderungen, welche unter Punkt 2 aufgelistet sind, entstanden erst im weiteren Verlauf des Projektes. Durch diesen schnellen, iterativen Entwicklungszyklus ergaben sich einige Dinge, welche bereits in das oben beschriebene Konzept eingebaut sind, auch wenn sich diese erst durch Ausprobieren und Fehlschlag ergaben. Diese Punkte sollen im folgenden Abschnitt erläutert werden.

4.1. Datenbankentscheidung

Die wohl größte Änderung gleich zu Beginn der Programmierung war das gewählte DBMS. Wie bereits in 3. erklärt, wurden PostgreSQL, MySQL und schließlich SQLite genutzt. PostgreSQL war die initiale Lösung, da es die Erste war, die in den Sinn kam. Nach mehreren teaminternen Gesprächen war jedoch klar, dass es zu bevorzugen is, MySQL zu nutzen, da dies

bereits auf einigen Servern installiert ist und somit kein zusätzliches DBMS installiert werden musste.

Nach weiteren Besprechungen wurde jedoch stärker verlangt, dass die Applikation gemäß REQ-F060-M schnell und einfach auf einem neuen Server aufgesetzt werden sollte. Da sowohl PostgreSQL und MySQL eine eigenständige Installation benötigen, wurde sich schnell auf SQLite geeinigt.

4.2. Navigation

Die ersten Versionen der Web-Applikation bestanden lediglich aus einer einzigen Seite, die alle Server nach Namen sortiert anzeigte. Sobald ebenfalls die VMs hinzugefügt wurden, wurden diese auf einer zusätzlichen Seite gelistet. Diese musste von der Hauptseite der Serverauflistung aus erreichbar sein, also musste ein Navigationselement kreiert werden. Nach mehreren Anläufen mit simplen Links wurde sich für eine sogenannte „Navbar“, eine Navigationszeile, die am oberen Rand der Webseite bleibt, entschieden. Diese wird von Bootstrap bereitgestellt und ermöglicht es, Navigationslinks und andere Features wie Suchleisten einzubinden.

Eben diese Suchfunktion wurde ebenfalls hinzugefügt, da klar wurde, dass es nicht sehr übersichtlich ist, wenn alle Server oder VMs einfach nur nacheinander aufgelistet sind. Dank eines einfachen JavaScripts war es ab sofort möglich, die aktuelle Seite nach einem eingegebenen Namen zu Filtern. Dies erleichterte es, eine gewünschte Information schneller zu finden.

Zusätzlich zur Navigationsleiste wurde die nach REQ-UI020-M geforderte Hauptübersicht mit Ressourcen-Statistiken hinzugefügt. Dies erlaubt es dem Nutzer, einfacher zu den gewollten Informationen zu finden, anstatt direkt in einer langen Liste von Informationen zu landen. Dieser Punkt wurde vor allem dadurch klar, weil einige Anwender bei der Nutzung der Applikation beobachtet wurden.

4.3. Datennutzung

Ebenfalls durch diese Beobachtungen stellte sich früh heraus, dass der Applikation einige Funktionalitäten fehlten, die es besser als die alte Excel-Arbeitsmappe machen. So nutzen neben den Entwicklern auch einige Entscheidungsträger die Applikation, um Informationen herauszufinden und gegebenenfalls Ersatzteile zu bestellen oder Garantiefälle zu melden.

Für diesen Nutzungsfall wurden der Darstellung einige Links hinzugefügt: Die gelistete LOM-IP ist klickbar, um das standardmäßig vorhandene Webinterface aufzurufen und die Seriennummer beziehungsweise der Service-Tag leitet auf die entsprechende Website des Herstellers weiter. Ebenso wurde ein Darstellungsmodus hinzugefügt, in welchem lediglich der Name, die IP-Adresse, die LOM-IP-Adresse und der Service-Tag aller Server in einer Liste angezeigt werden, um die gewohnte Ansicht einer Tabelle nachzuahmen.

4.4. Autostart

Nach einem Stromausfall und daraus resultierenden Abschalten der Applikation wurde klar, dass die Applikation in irgendeiner Art automatisch gestartet werden muss, da sie ansonsten nach einem unerwarteten Neustart des Hosts nicht mehr zur Verfügung steht. Um dieses Problem zu lösen wurde der NodeJS-Applikation ein Skript hinzugefügt, welches die Anwendung als Windows Service installiert. Dies geschieht nicht automatisch, damit die Applikation weiterhin auf einem Linux-Host funktionieren kann. Ebenfalls existiert ein ähnliches Skript, um die Applikation als einen „daemon“ unter Linux zu installieren, was vergleichbar mit einem Windows Service ist. Noch wurde besagtes Skript jedoch nicht integriert, da die Applikation bisher nur auf einem Windows-Host genutzt wurde.

4.5. Barrierefreiheit

Durch die einsetzende Nutzung der Applikation kamen ebenfalls weitere Beobachtungen auf: Einige der Entwickler hatten Probleme mit der Schriftgröße, die standardmäßig verwendet wird. Dies musste jedoch nicht auf der Software-Seite geändert werden, da dank der Verwendung von Bootstrap bereits die genutzten Komponenten sehr gut auf eine Vergrößerung der Ansicht reagierten und somit die Schriftgröße clientseitig angepasst werden konnte, ohne die Funktionalität zu beeinträchtigen.

Ein anderer Entwickler hat eine leichte Rot-Grün-Schwäche. Da die Applikation auf Farben setzt, um den Status der Server einfach sichtbar zu machen, wurde befürchtet, dass dies eventuell zu Problemen führt und einige Details nicht mehr lesbar sind. Dank der Rückmeldung und der Zusammenarbeit mit besagtem Entwickler konnten die Farben jedoch so angepasst werden, dass alle Elemente trotz dieser Sehschwäche gut lesbar und nutzbar sind.

5. Zusammenfassung und Zukunftsausblick

Zusammenfassend lässt sich sagen, dass das Projekt dadurch lebt, dass ein Problem existierte und dieses aktiv angegangen wurde, mit wenig vorheriger Planung. Einige Entscheidungen wurden ohne die Zukunft zu bedenken gefällt, wodurch mehrere Dinge mehrfach geändert werden mussten, um den eigentlichen Anforderungen gerecht zu werden. Es half ungemein, als diese Anforderungen formuliert wurden und die Applikation initial genutzt wurde, um eigentlich offensichtliche Probleme aufzuzeigen und diese beheben zu können. Zukünftige Projekte sollten aus diesen Fehlern lernen und die Nützlichkeit von fest formulierten Anforderungen auf keinen Fall unterschätzen. Ebenfalls sehr hilfreich ist es, die Zielgruppe im Voraus anzusprechen und aktiv zu fragen, was sie eigentlich wollen und erwarten.

Trotzdem ist es nicht zu vernachlässigen, dass es hilfreich war, schnell eine funktionsfähige Lösung zu haben. Dies ermöglichte es, Feedback zu sammeln und die existierende Situation, aus der das Projekt entstand, zu verbessern. Selbst ein nicht gutaussehendes Programm, was halbwegs funktioniert, hat seinen Nutzen.

Des Weiteren ist anzumerken, dass das Projekt momentan definitiv noch keinen finalen Status erlangt hat. Zwar ist der Datensammler in dieser Arbeit beschrieben, doch ist er noch nicht implementiert. Diesen Teil fertig zu stellen wird jedoch der nächste Abschnitt der Entwicklung, da das Problem des Informationsalters unbestreitbar eines der größten Probleme der Ausgangssituation war. Dies wird weitere Probleme aufdecken, die sich allerdings lösen werden und im Endeffekt zu einem nützlichen Tool führen werden, was den Alltag aller Nutzer ungemein erleichtern wird.

6. Literaturverzeichnis

Hellbusch, Jan Eric: Barrierefreies Webdesign -- Praxishandbuch für Webgestaltung und grafische Programmoberflächen. dpunkt.verlag, 2005. Speziell Seite 92/93.

DIN Berlin: PAS 1020: Designrichtlinien und Formulierungsstandards für E-Government-Applikationen. Beuth Verlag, 2003. Speziell Seite 6.

Web-Quellen:

Oracle: Oracle Java SE Support Roadmap. 2018. Abgerufen am 1.10.2018 von https://www.oracle.com/technetwork/java/eol-135779.html

Q-Success: Usage of server-side programming languages for websites. 2018. Abgerufen am 1.10.2018 von https://w3techs.com/technologies/overview/programming_language/all

solid IT GmbH: DB-Engines Ranking. 2018. Abgerufen am 1.10.2018 von https://db-engines.com/de/ranking

PostgreSQL Global Development Group: Appendix D. SQL Conformance in PostgreSQL 8.3.23 Documentation. 2018. Abgerufen: am 1.10.2018 von https://www.postgresql.org/docs/8.3/static/features.html

Liebig, Martin: Die gefühlte Lesbarkeit. 2009. Abgerufen: am 1.10.2018 von https://www.designtagebuch.de/wp-content/uploads/2009/08/Martin_Liebig_Die_gefuehlte_Lesbarkeit.pdf

[RFC3410] Case, J., SNMP Research, Inc., Mundy, R., Network Associates Laboratories, Partain, D., Ericsson und Stewart, B.: Introduction and Applicability Statements for Internet Standard Management Framework. 2002. Abgerufen: am 1.10.2018 von https://tools.ietf.org/html/rfc3410

Abbildungen:

Alle verwendeten Abbildungen in dieser Arbeit wurden explizit für diese erstellt und stammen aus eigener Hand. Für Abbildung 1 wurde das Programm yEd genutzt, alle anderen sind Bildschirmaufnahmen. Für Abbildung 2 wurde eine kostenlos und frei verfügbare Beispiel-SQLite-Datenbank von http://www.sqlitetutorial.net/sqlite-sample-database/ heruntergeladen und im Texteditor Notepad++ geöffnet.

7. Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema „Konzept und Entwicklung einer Software zur Verwaltung der Infrastruktur eines mittelgroßen Entwicklerteams“ vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg/Havel, den

(Unterschrift Peter Fredebold)