Deckblatt

Erklärung

Inhaltsverzeichnis

Inhaltsverzeichnis

[Einleitung 2](#_Toc295903268)

[Abstract 2](#_Toc295903269)

[Projekt Umgebung 2](#_Toc295903270)

# Einleitung

## Abstract

## Projekt Umgebung

Der Tagesspiegel….

Motivation

Die Verwaltung eines hochproduktiven Rechenzentrums ist sehr komplex, da die unterschiedlichsten Daten der verbauten Komponenten erfasst werden müssen. Die Fehlersuche während der laufenden Produktion sind hochzeitkritisch. Um diese Aufgaben bewältigen zu können wurde die in PHP geschriebene Webanwendung speedikon DAMS (Data Center Asset Management Solutions) beschafft. Diese Softwarelösung unterstützt die Verwaltung und Visualisierung von Bestandsdaten in den Rechenzentren. Die Pflege der Daten erfolgt im Moment ausschließlich über eine Weboberfläche, die einen PC-Arbeitsplatz mit entsprechendem Webbrowser voraussetzt. Um die Nutzbarkeit der bereits aufgenommen Daten flexibler zu gestalten, soll nun eine portable Lösung geschaffen werden, die mit kurzen Zugriffszeiten die technischen Daten, Verkabelungswege und Standorte der einzelnen Komponenten bereitstellt.

Aufgabenstellung

In diesem Projekt soll ein Prototyp einer Android-Anwendung entwickelt und implementiert werden, der die vorhandenen Daten von speedikon DAMS nutzt. Die vorhandene Webanwendung stellt keine Schnittstellen für den Abruf der Daten zur Verfügung. Aus diesem Grund soll ein Webservice entworfen und implementiert werden, der diese Aufgabe übernimmt. Die mobile Anwendung soll eine Benutzeroberfläche zur Verfügung stellen, die nach einer erfolgreichen Anmeldung des Nutzers die Suche und Abfrage von Geräteinformationen mit Hilfe diverser Suchkriterien ermöglicht und eine entsprechende Ausgabe generiert.

- BarcodeScanner

-

Abgrenzung/ Beschränkungen

Aufgrund der fehlenden Schnittstellen für die Manipulation der Datensätze und der Komplexität der Webanwendung, soll die mobile Anwendung ausschließlich lesenden Zugriff auf die Datensätze erhalten. Während der Entwicklung der Anwendung wird ausschließlich eine lokale Kopie der vorhandenen Datenbank genutzt. Grund dafür ist die aus Sicherheitsgründen fehlende Anbindung der mobilen Endgeräte an das vorhandene WLAN-Netz des Verlages.

Aufarbeitung des Themas

2.1 Android

2.1.1

2.1.2 Zxing

2.2 Webservice

2.2.1 Einführung

Wie in der Aufgabenstellung beschrieben, stellt die zugrunde liegende Web-Anwendung keine Schnittstellen für den Zugriff auf die vorhandenen Daten zur Verfügung, so dass ein entsprechender Webservice implementiert werden soll. In diesem Abschnitt sollen die Grundlagen zum Thema Web Service erläutert werden.

Web Services spielen im Zeitalter des Internets eine wichtige Rolle. Die Idee bei der Realisierung der Web Services ist die Bereitstellung vorhandener Anwendungen zur Verwendung im Internet bzw. über ein internes Netzwerk. Web Anwendungen wie Amazon, Ebay und Google bieten Schnittstellen zur Nutzung der eigenen Dienste zur Einbettung in anderen Anwendungen an.

Für den Austausch und die Manipulation von Daten über definierte Schnittstellen hat sich die Architektur REST (Representational State Transfer) etabliert [HEI10]. Die Grundlagen hat Roy Fielding im Jahr 2000 in seiner Doktorarbeit gelegt. [FIE00]. Nachfolgend sollen die Eigenschaften von REST aufgezeigt werden.

Die zentrale Einheit bei der REST Architektur, stellen die so genannten Ressourcen dar. Ressourcen sind in diesem Fall Inhalte die über das Netzwerk aufgerufen werden können. Diese Inhalte können in Form von Dokumenten, Bildern oder auch Ergebnisse von Datenbankabfragen vorliegen, die in unterschiedlichen Formaten wie XML[[1]](#endnote-1)3 oder JSON4 repräsentiert zur Verfügung gestellt werden können. Für den Aufruf wird jede der Ressourcen mit einem eindeutigen Uniform Resource Identifier (URI) adressiert. Die Idee bei REST liegt in der einfache Manipulation der angesprochen Ressourcen. Dabei sollen einfache Schnittstellen definiert werden, um die so genannten CRUD-Methoden auf die gewünschten Ressourcen anwenden zu können. CRUD steht für folgendes:

* Create
  + Erzeugen einer neuen Ressource
* Read
  + Lesen/ Aufruf einer Ressource
* Update
  + Änderungen an einer Ressource
* Delete
  + Löschen einer Ressource

Als Transportprotokoll nutzt REST2 das im Internet weit verbreitete Hyper Text Transfer Prototocoll (HTTP). Die Nutzung von HTTP macht die Web Anwendung somit plattform unabhägig, da für die Nutzung der Schnittstelle lediglich ein HTTP-Client benötigt wird.[Ker09] HTTP beschränkt sich dabei auf eine geringe Anzahl von Methoden, die hier kurz aufgezeigt werden sollen:

* GET
  + Die Methode GET stellt den lesenden Zugriff auf Ressourcen zur Verfügung. Die Methode verhält sich dabei idempotent und ist sicher bzw. gefahrlos. Idempotent bedeutet in diesem Fall, dass ein lesender Zugriff die gewünschte Ressource unverändert lässt und somit jeder weitere Zugriff auf die Ressource das gleiche Ergebnis liefert. Sicher im Zusammenhang mit der Methode GET bedeutet, dass der Aufruf einer Ressource durch einen Benutzer keinen Seiteneffekt hervorruft, d.h. dass die Anfrage keinen vom Benutzer erwarteten Einfluss auf andere Ressourcen oder den Betrieb des Servers haben darf.[W3C][Bur10]
* PUT
  + Die Methode PUT ist eine schreibende Methode. Bei der Anfrage an den Server, wird die in der http-Nachricht übertragen Ressource an der eindeutigen Adresse (URI) gespeichert, die in den Meta-Daten des Paketes angegeben ist. Sollte die angegebene URI nicht vorhanden sein, wird sie durch den Server angelegt. Die Methode PUT ist idempotent, d.h. dass die mehrmalige speichernde Anfrage der gleichen Ressource dasselbe Ergebnis herbeiführt. [Bur10]
* DELETE
  + Die Methode DELETE wird benutzt um Ressourcen auf dem Server zu löschen. Die Methode ist ebenfalls idempotent, da das mehrmalige Löschen einer Ressource das gleiche Ergebnis liefert. [Bur10]
* POST
  + Die Methode POST ist sehr flexibel einsetzbar. So ist es möglich in den entsprechenden Anfragen bzw. Antworten Daten zu senden oder zu empfangen. Weiterhin können mit der Methode auch die Funktionalitäten von anderen Methoden wie GET und PUT realisiert werden. Allerdings ist PUT weder idempotent noch sicher, weil die in den Anfragen übertragenen Meta-Daten bzw. Parameter ständig veränderbar sind. Zudem bietet PUT weitere Funktionen an, wie das Annotieren von bekannten Ressourcen. [Bur10] [W3C]
* HEAD
  + Die Methode HEAD stellt die gleiche Funktionalität wie die Methode GET zur Verfügung. Der Unterschied besteht darin, dass der Server bei einer HEAD-Anfrage nicht zwingende den Inhalt der aufgerufenen Ressource zurücksendet, sondern nur die Meta-Daten die die Ressource betreffen.[W3C]
* OPTIONS
  + Die Methode OPTIONS ermöglicht den Aufruf von Optionen für die Kommunikation zwischen Server und Nutzer und deren Auswertung, ohne die Ressource selbst anzufordern. [Bur10]
* TRACE
  + Die Methode TRACE stellt Funktionalitäten zur Verfügung mit denen der Nutzer zu Testzwecken Anfragen an den Server schickt und der Server die verarbeitet Anfrage zurücksendet. Mit dieser Methode kann der Nutzer die Rückantwort auswerten und die gewonnenen Informationen zur Fehlerbehebung oder Diagnose nutzen. [W3C]
* CONNECT
  + Die letzte zu betrachtende Methode ist CONNECT. Diese Methode steht für die Nutzung von Proxy-Eigenschaften zur Verfügung, um dynamisch auf gesicherten bzw. getunnelten Netzwerkverkehr umzuschalten. [W3C]

Neben dem hohen Bekanntheitsgrad bietet HTTP noch weitere Vorteile. In vielen Programmiersprachen stehen Klassen für die Programmierung von Clients zur Verfügung. Die Methode GET als lesende Operation kann durch den Client zwischengespeichert werden, so dass bei einem erneuten Aufruf der Ressource die Wartezeit und die Netzwerklast verringert werden kann.

Eine weitere grundlege Eigenschaft der REST-Architektur ist die zustandslose Kommunikation. Zustandslos bedeutet in diesem Zusammenhang, dass der Server keine Sitzungsdaten der Nutzeranfragen vorhält. Wenn die Verwaltung der Sitzungsdaten für eine Anwendung erforderlich ist wird dies durch das dienstanfragende Gerät realisiert. Die Sitzungsdaten werden dann in jeder HTTP-Anfrage mitgesendet und entsprechend durch den Server ausgewertet.

In der Programmiersprache Java steht seit 2008 das Framework JAX-RS zur Verfügung. JAX-RS stellt Funktionalitäten bereit um „RESTful“-Webservices zu implementieren. Unteranderem bietet es die Möglichkeit Annotations zu verwenden, um Metadaten die für den Webservice benötigt werden einbinden zu können. Im Folgenden sollen kurz die wichtigsten Annotations aufgezeigt werden.

@Path

Definiert den Pfad mit der die zur Verfügung gestellte Klasse aufgerufen werden kann und bildet so den Knotenpunkt („root ressource“) für die mit @Path annotierten Methoden in der Klasse.

@Path(”/test”)

public class TestKlasse{

@Path(/methode1)

public String testMethode1(){

…

}

}

Die im Webservice aufzurufende URI wäre dann

<http://server:port/webprojekt/test/method>

Für die Annotationen der aufgerufenen Methoden gibt es weitere Annotationen:

@GET

Bindet die GET-Methode an die in der HTTP-Anfrage definierte Methode. Diese Annotation ist auch für die HTTP-Methoden PUT, POST, DELETE und HEAD möglich.

@Produces

Über diese Annotation wird der erwartete Rückgabetyp der aufgerufenen Methode festgelegt.

@PathParam

Die Annotation @PathParam dient dazu, die Eingabeparameter einer Methode mit den Werten aus der HTTP-Anfrage zu initialisieren. Der Parameter wird dann in der @Path Annotation der Methode mit geschweiften Klammer angegeben.

Beispiel:

@Path(“/methode2/{parameter}”)

public String testMethode2(@PathParam(“parameter”) String eingabe){

…

}

@FormParam

Die Annotation @FormParam bietet die Möglichkeit Parameter in Form von Name-Werte-Paaren in einer HTTP-Anfrage zu übertragen. Das Einfügen der möglichen Parameter in die HTTP-Anfrage liegt in Verantwortung des Clients.

@Path(“/methode3/”)

public String testMethode3(@FormParam(“parameter”) String eingabe){

…

}

2.3 Testen

2.3.1 Einführung

Das Testen von Software ist ein wichtiger Bestandteil in der Softwareentwicklung, um eine hohe Qualität des erstellten Produktes zu gewährleisten. Die Entwicklung von Testszenarien und den entsprechenden Tests sollte so früh wie möglich erfolgen, so dass auftretende Fehler möglichst wenig Auswirkungen auf spätere Entwicklungen haben können. Nachfolgend sollen kurz mögliche Typen von Softwaretests aufgeführt werden:

* Unittest

Die Unittests sind Testklassen, in denen Methoden implementiert werden, die einzelne Module des Programmes isoliert, d.h. ohne Einwirkung von anderen Modulen, testen. Unittests stellen somit die Grundlage für weitere Tests dar.

* Integrationstest

Diese Tests stellen die nächste Stufe der Softwaretests nach den Unittests dar. Im Gegensatz zu den Unittests, werden die Abhängigkeiten und Funktionsweisen von Modulen untereinander betrachtet.

* Validierung Und Verifikation

Bei diesen Tests ist es wichtig herauszufinden, ob die erstellte Anwendung den gestellten Ansprüchen des Auftraggebers entspricht. Zudem muss geklärt werden, ob die Anforderung des Auftraggebers die Wünsche der Anwender erfüllen.

* Ressourcenverbrauch, Fehlersituation und Wiederherstellung

Neben der korrekten Funktionsweise des erstellten Quelltextes ist auch ist auch die Lauffähigkeit auf dem Zielsystemen zu überprüfen. Die verfügbaren Ressourcen des Zielsystems sind zudem begrenzt. Dazu zählen unteranderem der Speicher, Festplattenplatz und die Leistungsfähigkeit der CPU5. Zudem kann auch die Darstellung zwischen einzelnen Endgeräten variieren.

* Performance-Tests

Diese Tests sollen die Leistungsfähigkeit und die Leistungsgrenzen einer Anwendung aufzeigen. Dabei muss hinterfragt werden, ob die gewünschte Anzahl der Benutzer und Verbindungen unterstützt werden kann. Diese Betrachtung ist vor allem für die Skalierbarkeit der Anwendung wichtig.

* Usability-Tests

Aus Sicht der Anwender stellen Usability-Tests einen sehr wichtigen Aspekt dar. Bei diesen Tests stellt sich heraus, ob die erstellte Anwendung den Anforderung der Benutzer entspricht und ob die gewünschten Funktionen abgedeckt sind. Zudem soll durch die Endanwender die Benutzerbarkeit getestet und eingeschätzt werden. Somit liefern Usability-Tests den Entwicklern Erkenntnisse zur Verbesserung der Benutzerfreundlichkeit.

Nachfolgend soll die Betrachtung der Unittests mit der Umsetzung für Java mit JUnit erfolgen.

2.3.2 JUnit-Framework

Das JUnit-Framework ist aus dem von Kent Beck entwickelten SUnit-Framework für die Programmiersprache Smalltalk hervorgegangen. [RAI05] Das JUnit-Framework stellt Methoden zur Verfügung um Unittests durchzuführen. Unteranderem zählen dazu Methoden mit denen die Testumgebung vorbereitet werden kann, die die Ergebnisse visualisieren und die getesteten Objekte freizugeben. Die zu implementierenden Test-Klassen stellen eine Unterklasse der Klasse TestCase aus dem JUnit-Framework dar, die die benötigten Methoden bereitstellt. Nachfolgend sollen einige Methoden aus der Klasse TestCase aufgeführt werden.

* setUp()

Die Methode setUp() schafft die Umgebungsbedingungen für die aufrufenden Test-Methoden. Dabei werden die für den Unittest benötigten Objekte initialisiert. Die setUp() Methode wird vor jedem Aufruf einer Methode aus der erstellten Test-Klasse aufgerufen, um die Objekte für den nächsten Test erneut zu initialisieren, so dass für jede Test-Methode die gleichen Ausgangswerte vorliegen.

* tearDown()

Die Methode tearDown() wird nach jedem Aufruf einer Test-Methode aufgerufen, um die zuvor für den Test erzeugten Objekte und gebundenen Ressourcen wieder freizugeben.

Neben den Klassen zur Vor- und Nachbereitung der Testumgebung stehen im JUnit-Framework weitere Methoden zur Verfügung um Behauptungen aufzustellen und zu überprüfen ob diese erfüllt werden. Die Methoden stehen in der Klasse Assert zur Verfügung, die durch die Klasse TestCase erweitert wird. Im Folgenden sollen die entsprechenden Methoden kurz beschrieben werden.

* assertEquals([String nachricht], erwartet, derzeitig)

Diese Methode stellt eine Möglichkeit zur Verfügung um einen erwarteten Wert mit dem derzeitgen Wert zu vergleichen, den der zu testende Quellcode bereitstellt. Die Nachricht kann optional angegeben werden und wird bei einem negativen Testergebnis ausgegeben.

* assertNull([String nachricht], objekt)

Die Methode liefert einen Fehler, wenn das zu testende Objekt nicht null ist. Zusätzlich gibt es auch die Methode assertNotNull, die einen Fehler generiert, wenn das Objekt nicht null ist.

* assertSame([String nachricht], erwartet, derzeitig)

Die Methode assertSame() bietet die Möglichkeit, um zu testen ob es sich um das gleiche Objekt handelt. Der Test schlägt fehl, wenn das erwartete und das derzeitige Objekt nicht das gleiche Objekt darstellen. Die Umkehrung für diese Methode ist die assertNotSame()-Methode.

* assertTrue([String nachricht], wahrheitswert)

Diese Methode erwartet, dass der gegebene Wahrheitswert wahr ist. Die Umkehrung dieser Methode ist die assertFalse()-Methode, die einen falschen Wahrheitswert als Behauptung erwartet.

* fail([String nachricht])

Mit Hilfe der Methode fail() kann ein Test abgebrochen werden. Als Einsatzszenario für diese Methode kann unter anderem das Testen von Exceptions angesehen werden. Das folgende Code-Beispiel soll die Funktionweise kurz verdeutlichen:

public void testException() {

try{

methode4(null);

fail(“Die Methode sollte eine Exception werfen!”);

} catch (Exception e){

assertTrue(true);

}

}

Es wird davon ausgegangen, dass die aufgerufene Methode eine Exception wirft, wenn der Eingabeparameter null ist. Wird die Exception geworfen, wird die Methode assertTrue(true) im catch-Block aufgerufen und der Test somit als erfolgreich ausgegeben. Wird die Exception nicht geworfen, wird die Methode fail() mit der entsprechenden Nachricht aufgerufen und somit erzwungen, dass der Test nicht erfolgreich ist.

Clientseitig

* Android

Test

* JUnit
* Testen in Android

(Netzanbindung

* Sichere Anbindung von mobilen Geräten in Unternehmensnetz)

(Barcode

* 1/2D Barcode
* QR-Barcode)

Umsetzung

Analyse der vorhandenen Anwendung

* Anwendung
* Datenbank

Lösungsansatz definieren

* Anwendungsfälle erarbeiten
* Webservice konzipieren
* App konzipieren

GUI

Logik

Externe Bibliotheken (z.B. Barcodescanner)

* Testfälle erarbeiten

Beschreibung der erstellten Klassen

Auswertung der Tests

Abschluss

Fazit/ Problembeschreibung

Erweiterungsmöglichkeiten

Burke, B. (2010). *RESTful Java with JAX-RS.* Sebastopol: O'Reilly.

1. 3 Extensible Markup Languag [↑](#endnote-ref-1)