Fachhochschule Salzburg University of Applied Sciences

R&D PROJEKT

Testplan

Version 1.1

Bezeichnung: Erstellung einer JRZ Demodatenbank (DemoDB)

Projektschlüssel: RD16-03

Betreuer: DI Eduard Hirsch, DI Fabian Knirsch, BSc

Kurzbeschreibung: Konvertierung verschiedener Smartmeter Messdaten und Ablage in einer gemeinsamen Datenbank mit rollenbasiertem Zugriff.

Beteiligte Firma: Salzburg AG

Studenten: Isidor Reimar Klammer, BSc.

Maximilian Unterrainer, BSc.

Christopher Wieland, BSc.

Puch/Salzburg, 29. Mai 2017

Inhalt

Historie 3
Einführung 4
Zweck 4
Umfang 4
Beziehungen zu anderen Dokumenten 4
Testbezeichner 4
Referenzen 5
Systemüberblick 5
Module und deren Funktionen 6
Merkmale 6
Zu testende Merkmale 6
Nicht zu testende Merkmale 6
Abnahme- und Testendekriterien 7
Vorgehen 7
Unit-Tests 7
Komponenten- und Integrationstest 7
Funktionstest 8
Anforderungen an Hardware und Software (Server/Client) 8
Server - Hardware 8
Server - Software 8
Server - Konfiguration 8
Client 8
Testfälle 8
Integrationstest 9
Funktionstests - LDAP 10
Volumentest 10
Fehlerbehebung 11

Historie

Version	Datum	Änderung	Änderer
1.0	15.4.2017	Ersterstellung	M
1.1	29.5.2017	Ergänzung weiterer Testfälle	M

Einführung

Dieses Dokument beschreibt den Testplan für das Research & Development Projekt „SmartValAPI“, welches im Zuge des ITSM Studiums an der FH Salzburg von I. Klammer, M. Unterrainer und C. Wieland umgesetzt wird.

Zweck

Dieser Testplan gibt dem Auftraggeber und den Entwicklern einen Überblick, welche Schritte wie abgeschlossen werden müssen, um das Projekt erfolgreich abzuschließen und die Software produktiv einzusetzen.

Umfang

Dieser Testplan basiert auf den Vorgaben des Standards IEEE 829-2008 [1] und wurde an jenen Stellen, die auf Grund des Projektes nicht anwendbar sind, wie zum Beispiel genaues Ausspezifizieren des Testteams oder Einflüsse parallel umgesetzter Projekte entsprechend gekürzt.

Beziehungen zu anderen Dokumenten

Als Basis für die Umsetzung dienen der Auftrag des JRZ, die Protokolle der Besprechungen und Feedbackrunden und vor allem das Pflichtenheft. Aus diesen Inputs und der Erfahrung der Entwickler ergeben sich die Testfälle. Details, die im Testkonzept nicht enthalten sind, finden sich in der Bedienungs- beziehungsweise der Installationsanleitung.

Testbezeichner

Tests werden mit eindeutigen Bezeichnern in der Form

X-NN-V

identifiziert. Die Bedeutung der Kürzel ist in Tabelle 1 erklärt.

Kürzel	Ausprägungen	Bedeutung
X	F, I, V	F – Funktionstest I – Integrationstest V – Volumentest
NN	00 – 99	Laufende Nummer innerhalb von X
V	A – Z	Variante eines Tests, bezüglich des Ergebnisses und der Umgebungsbedingungen

Tabelle 1 Testbezeichner

Referenzen

IEEE Std. 829-2008: Software Engineering Technical Committee of the IEEE Computer Society. IEEE standard for software test documentation, USA, 2008.

Systemüberblick

Abbildung 1 zeigt die Komponenten von SmartValAPI. In der Folge werden die Funktionen beschrieben.

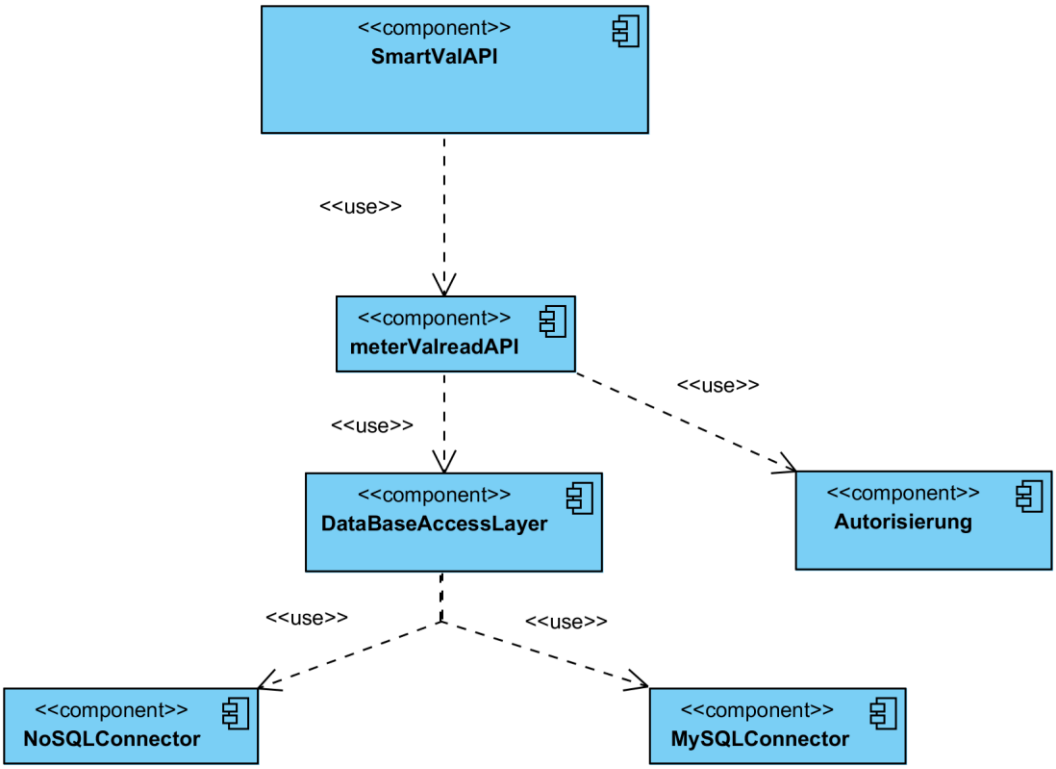

Abbildung 1 Komponenten von SmartValAPI

Module und deren Funktionen

- Administrationsschnittstelle: die Funktionen werden über die Controller Klassen realisiert (Projektverzeichnis: /SmartValAPI/rest/src/main/java/at/ac/fh/salzburg/ smartmeter/data/rest), diese werden durch Annotationen über den Applikationsserver zugänglich gemacht.
- Benutzerschnittstelle: die Musteraufrufe stellen die eigentliche Funktionalität des API zur Verfügung. Klassen, die benutzerspezifische Auswertungen durchführen stehen im selben Verzeichnis wie die Administrationsschnittstellenklassen.
- Datenbankanbindung (MySQL): die Klassen, über die der Datenbankzugriff erfolgt, dient vor allem der Umwandlung der relationalen Struktur der Datenbank in Objekte (Projektverzeichnis: /SmartValAPI/rest/src/main/java/at/ac/fh/salzburg/smartmeter /data/dao und /entities)
- Zugriffsverwaltung: Der Zugriff wurde über LDAP realisiert, die Funktionen stehen im Verzeichnis SmartValAPI/ldap/build/classes/main/at/ac/fh/salzburg/smartmeter/ ldap.

Merkmale

Im Mittelpunkt der Tests steht die korrekte Rückgabe der JSON Objekte, da dies die Hauptaufgabe des API ist.

Zu testende Merkmale

Getestet werden die grundlegenden Funktionalitäten der API, die Testfälle sind weiter unten angeführt. Nichtfunktionale Anforderungen wie zum Beispiel Erweiterbarkeit werden nicht automatisiert getestet, sondern können nur im gemeinsamen Gespräch erörtert werden.

Nicht zu testende Merkmale

Nicht explizit getestet werden die grundsätzlichen Funktionen der eingesetzten Fremdkomponenten, wie zum Beispiel MySQL, die JDBC Anbindung und Hibernate, da davon ausgegangen werden kann, dass diese Systeme vielfach im Einsatz sind, und von den jeweiligen Herstellern bereits ausgiebig getestet wurden. Hingegen werden die Ergebnisse, welche von diesen Komponenten geliefert werden, und vor allem das Zusammenspiel mit SmatValAPI sehr wohl getestet. Performance ist bei den zu erwartenden Datenmengen wichtig, bei der aktuellen Hardwareausstattung können aber keine allgemein gültigen Aussagen getroffen werden, die Tests beschränken sich hier auf Messungen.

Abnahme- und Testendekriterien

Auftretende Fehler werden entsprechend der Einteilung in Tabelle 2 kategorisiert. Das Ende des Teststadiums kann erst erreicht werden, wenn alle Fehler der Kategorien 1 und 2 behoben wurden. Die Abnahme/Übergabe kann erst dann erfolgen, beziehungsweise ist das Ende des Teststadiums erst erreicht, wenn alle Fehler der Priorität 1 und 2 behoben wurden. Die Zuordnung zu einer Klasse unterliegt dem Konsens des Entwicklerteams und des Auftraggebers.

Klasse	**Beschreibung**
Fatal (Priorität 1)	Absturz oder Dienstunterbrechung, es kann nicht weitergearbeitet werden, ein Eingriff am Server ist notwendig und Änderungen in der Software die Folge.
Kritisch (Priorität 2)	Fehler, die die Verwendbarkeit der gelieferten Daten zerstören.
Gering (Priorität 3)	Fehler, welche die Leistung geringfügig einschränken.
Unwesentlich (Priorität 4)	Fehler, die zwar auftreten, aber keine erkennbaren Schäden verursachen, werden mit niedriger Priorität behoben.

Tabelle 2 - Fehlerkategorien

Vorgehen

Unit-Tests

Auf Klassen- und Methodenebene werden die Tests mittels JUnit-Tests getestet. Diese Tests werden während, spätestens aber zum Ende der Implementierung geschrieben und decken neben den Standardfällen („Happy Path“) Situationen in denen Ausnahmen ausgelöst werden und bewusstes Fehlverhalten provoziert wird, ab. Besonderes Augenmerk wird auf Grenzsituationen gelegt, zum Beispiel die Berechnung des Durchschnitts einer leeren Menge.

Komponenten- und Integrationstest

Die Komponenten- und Integrationstests werden von der Basis zur Gesamtkomponente hin durchgeführt („bottom-up“). Das Mapping der Datenbankinhalte auf POJOs ist hier der zuunterst liegende Teil, in der Folge wird durch Schnittstellentests (lokale Installation) die Funktionsfähigkeit getestet. Parallel dazu erfolgen die LDAP-Tests gegen den Deployment-Server (landsteiner.fh-salzburg.ac.at). Anschließend schließen die Tests gegen die REST

Schnittstelle den Integrationstest ab. Hier liegt der Schwerpunkt auf dem Zusammenspiel der einzelnen Komponenten und den Ergebnissen der Schnittstelle.

Funktionstest

Die Musterauswertungen werden, wie sie im Pflichtenheft angeführt sind, getestet. Die Mindestanforderung ist, dass jede der drei Auswertungen für einen konkreten Test korrekte Ergebnisse liefert. Voraussetzung dafür ist die Beschickung der Datenbankinstanz mit jenen Daten, die in den Installationsskripts (creatTestdata.1.sql und smartvalapi_meter_data-REDDGREEND.sql) enthalten sind.

Anforderungen an Hardware und Software (Server/Client)

Server - Hardware

Für die Tests und in der Folge die Ausrollung der Software steht eine virtuelle Maschine am Blade-Server des JRZ zur Verfügung. Ausgestattet ist dieser Rechner mit 2 Intel Xeon E5-2620 Prozessoren mit 2GHz, 4 GB Hauptspeicher und 80 GB Plattenplatz.

Server - Software

Zusätzlich zur eigentlichen Software wird am Server als Betriebssystem Windows Server 2012 R2 installiert. Als RDBMS wird ein MySql Server 5.7.17 eingesetzt, zur einfacheren Überprüfung von Daten am Server wird die MySQL Workbench 6.9 verwendet. Für die optionale NoSQL-Verwaltung der Meterdaten wird MongoDB in der Version 3.4 installiert.

Server - Konfiguration

Dieser Rechner ist über das Internet unter der IP-Adresse 193.170.119.66 oder dem Hostnamen landsteiner.fh-salzburg.ac.at ansprechbar, erreichbar mit dem Remote Desktop Protokoll. Die REST Schnittstelle horcht am Port 8080.

Client

SmartValAPI stellt eine REST-Schnittstelle zur Verfügung, die Anforderung an den Client ist daher gering. Zugriff über das Internet auf den Port des Servers ist zu gewährleisten. Jeder beliebige REST-Client kann eingesetzt werden, getestet wird mit dem in IntelliJ IDEA integrierten REST-Client und dem frei verfügbaren SOAPUI in der Version 5.3.0. Örtlich gibt es keine Einschränkungen.

Testfälle

Die folgende Übersicht führt alle zu erfüllenden Testfälle an.

Integrationstest

Testbezeichner	Testbeschreibung
I-00-A	Abfrage mit Benutzer auf eigene Daten
I-00-B	Abfrage mit Benutzer auf fremde Daten
I-01-A	Aufruf der Administrations-Schnittstelle mit einem Benutzer ohne Administratorberechtigung
I-02-A	Abfrage auf eigene Daten in Zeitraum ohne Messdaten
I-03-A	Abfrage auf Zeitraum mit „Zeitraum-Von“ > „Zeitraum-Bis“
I-04-A	Abfrage ohne Benutzerkontext
I-05-A	Aufruf mit einer weiteren Abfrage die nicht funktioniert
I-06-A	Aufruf mit falschem Pfad im Parameter
I-07-A	Administratorschnittstelle Abfrage bestehender Kundendaten
I-07-B	Administratorschnittstelle Abfrage nicht bestehender Kundendaten
I-08-A	Administratorschnittstelle mit Update Kundendaten
I-08-B	Administratorschnittstelle mit Kundenupdate, fehlender Body
I-08-C	Administratorschnittstelle mit Kundenupdate, unpassende Kundendaten Struktur im Body
I-09-A	Administratorschnittstelle Abfrage bestehender Metermanagementdaten
I-09-B	Administratorschnittstelle Abfrage nicht bestehender Metermanagementdaten
I-10-A	Administratorschnittstelle mit Update Metermanagementdaten
I-10-B	Administratorschnittstelle mit Metermanagementdaten Update, fehlender Body
I-10-C	Administratorschnittstelle mit Update Metermanagementdaten, unpassende Struktur im Body
I-11-A	Abfrage von 2 Smartmetermessdaten in einem gemeinsamen Zeitraum
I-11-B	Abfrage von 2 Smartmetermessdaten in einem gemeinsamen Zeitraum, wobei einmal keine Messdaten zur Verfügung stehen
I-12-A	Abfrage von einem Messdatenvektor über einen Zeitraum
I-12-B	Abfrage von leeren Messdatenvektor über einen Zeitraum

Funktionstests - LDAP

Testbezeichner	Testbeschreibung
F-00-A	Identifizierung mit bestehendem Benutzer und richtigem Passwort
F-00-B	Anmeldung mit Benutzer und falschem Passwort
F-00-C	Anmeldung mit Administrator Benutzer und richtigem Passwort
F-00-D	Anmeldung mit nicht existierendem Benutzer
F-01-A	Abfrage der zugeteilten Smart-Meter zu einer gültigen Benutzerkennung

Funktionstests – DAO

Testbezeichner	Testbeschreibung
F-10-A	CRUD Zugriffe auf Kundendaten
F-11-A	CRUD Zugriffe auf Meterstammdaten
F-12-A	CRUD Zugriffe auf Metermessdaten
F-13-A	CRUD Zugriffe auf Meterherstellerdaten
F-14-A	CRUD Zugriffe auf Metertypendaten
F-15-A	Ermittlung der Zählernummern zur Kundennummer
F-16-A	Meterdatenvektor auf Meter_Id eingeschränkt auf TimestampVon – TimestampBis

Zusätzliche Funktionstest Benutzerschnittstelle

Testbezeichner	Testbeschreibung
F-20-A	Angleichen zweier Messdatenvektoren auf gleiche Abtastfrequenz und minimale gemeinsam aufweisende Messwerte

Volumentest

Die Volumentests werden mit SoapUI mit der Funktion „Load Test“ von mehreren Rechnern aus parallel gegen den Deploymentserver (landsteiner.fh-salzburg.ac.at) durchgeführt.

Testbezeichner	Testbeschreibung
V-00-1	Parallele Anfragen über die Benutzerschnittstelle

Fehlerbehebung

Um Fehler beheben zu können werden diese mit einer eindeutigen ID versehen, neben der Beschreibung ist der Request der zum Fehler führt vonnöten.