

R&D Projekt

Abschlussbericht

**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, 30. August 2017

# Inhaltsverzeichnis

[Inhaltsverzeichnis i](#_Toc491176630)

[Abkürzungsverzeichnis iii](#_Toc491176631)

[Abbildungsverzeichnis iv](#_Toc491176632)

[Tabellenverzeichnis v](#_Toc491176633)

[Listingverzeichnis vi](#_Toc491176634)

[1 Einleitung 1](#_Toc491176635)

[1.1 Problemstellung und Motivation 1](#_Toc491176636)

[1.2 Umgebung 2](#_Toc491176637)

[2 Anforderungsanalyse 1](#_Toc491176638)

[2.1 Datenmodell 1](#_Toc491176639)

[2.1.1 Analyse der JRZ-DB 1](#_Toc491176640)

[2.1.2 Anforderungen von Energieversorgern und Netzdienstleistern 2](#_Toc491176641)

[2.1.3 Anforderungsprofil „Lastenheft Österreichs Energie“ 3](#_Toc491176642)

[2.1.4 Rechtliche Rahmenbedingungen 4](#_Toc491176643)

[2.1.5 Anpassung des Datenmodells 4](#_Toc491176644)

[2.2 Rollenbasierter Zugriff 5](#_Toc491176645)

[2.2.1 Rollenidentifikation 5](#_Toc491176646)

[2.2.1.1 Messdaten aus dem Echtbetrieb 5](#_Toc491176647)

[2.2.1.2 Anonymisierte Messdaten aus dem akademischen oder Forschungsbereich 6](#_Toc491176648)

[2.2.2 Einbeziehung zusätzlicher Domänen 6](#_Toc491176649)

[2.2.3 Rollendefinition 7](#_Toc491176650)

[2.2.4 Verbindung zur Rollenverwaltung 7](#_Toc491176651)

[2.3 Datenbankanforderungen 8](#_Toc491176652)

[2.3.1 Testdaten 9](#_Toc491176653)

[2.3.2 Messung 9](#_Toc491176654)

[2.3.3 Erwartete Datenmengen 11](#_Toc491176655)

[2.4 Systemarchitektur 12](#_Toc491176656)

[2.5 Alternative Datenhaltung 14](#_Toc491176657)

[3 Umsetzung 16](#_Toc491176658)

[3.1 CRUD API 16](#_Toc491176659)

[3.2 Custom API 17](#_Toc491176660)

[3.3 API Dokumentation 17](#_Toc491176661)

[3.4 Erweiterung der API 18](#_Toc491176662)

[3.5 LDAP Zugriffsregelung 21](#_Toc491176663)

[3.5.1 LDAP Server 21](#_Toc491176664)

[3.5.2 OpenLDAP konfiguration 22](#_Toc491176665)

[3.5.3 Schnittstelle zur Authentifizierung in SmartVal API 23](#_Toc491176666)

[3.6 Installation 25](#_Toc491176667)

[3.6.1 Systemvoraussetzungen 26](#_Toc491176668)

[3.6.2 Einrichtung der Datenbank und der Tabellen 26](#_Toc491176669)

[3.7 Start der Applikation 27](#_Toc491176670)

[3.8 Automatisierte Tests 27](#_Toc491176671)

[3.9 Verbesserungsvorschläge am bestehenden Datenmodell, die zu einfacherer Handhabung und Zugriffsbeschleunigung führen 29](#_Toc491176672)

[3.9.1 Benennung der Attribute 29](#_Toc491176673)

[3.9.2 Schlüssel und Indizes auf der Tabelle meter\_data 29](#_Toc491176674)

[Literaturverzeichnis 32](#_Toc491176675)

[Anhang 35](#_Toc491176676)

[SQL Messungen 35](#_Toc491176677)

# 

# Abkürzungsverzeichnis

AMCS Advanced Meter Communication System

API Application Programming Interface

COSEM Companion Specification for Energy Metering

DAVID-VO Datenformat- und Verbrauchsinformationsdarstellungs Verordnung

DBM Datenbankmodell

DLSM Device Language Messaging Specification

ETSI European Telecommunication Standards Institute

ElWOG Elektrizitätswirtschafts- und -organisationsgesetz

GDPR EU General Data Protection Regulation

IMA-VO Intelligente Messgeräte-AnforderungsVO 2011

JRZ Josef Ressel Zentrum für Anwenderorientierte Smart Grid Privacy, Sicherheit und Steuerung

JRZ-DB Datenbank, die im JRZ eingesetzt wird und auf deren Basis die gemeinsame Datenplattform entwickelt wird.

MMS Manufacturing Messaging Specification

OBIS Object identification system, entsprechend der EN 62056-01

OSGP Open Smart Grid Protocol

RBAC Role Based Access Control

REST Representational State Transfer

SmartValAPI Smart Meter Data Value API, Arbeitstitel des Projektes

SML Smart Message Language

SOAP Simple Object Access Protocol

JSON JavaScript Object Notation

XML Extensible Markup Language

# Abbildungsverzeichnis

Abbildung 1: Datenmodell JRZ-DB 2

Abbildung 2: erweiterte Tabelle meter\_data 5

Abbildung 3: Tabelle REDD mit Testdaten 9

Abbildung 4: Dauer Berechnung des Mittelwerts auf der MySQL Datenbank 10

Abbildung 5: Dauer des Einfügens in die MySQL Datenbank 11

Abbildung 6: Komponentenmodell 14

Abbildung 7: Bestandteile der API 16

Abbildung 8: API Dokumentation am Beispiel /admin/customer 18

Abbildung 9: Projektstruktur 19

Abbildung 10: TestSuite, um alle Tests automatisiert durchzuführen 27

Abbildung 11: einzeln durchführbare REST-Testfälle 28

Abbildung 12: Testfall I-11-A mit drei Messdatenreihen in SOAP UI 28

Abbildung 13: Zugriffspfade auf Messdaten (meter\_data) 30

Abbildung 14: Ergebnis von moMeterDataquery 34

# Tabellenverzeichnis

Tabelle 1: Rücklauf der Anfragen bei Energieversorgern 3

Tabelle 2: maximale Auflösung auf Grund einer zugeteilten Rolle 7

Tabelle 3: Zugriff auf anonymisierte Messdaten aus Forschungsquellen 7

Tabelle 4: LDAP-Attribute zu Benutzer-Objekten 7

Tabelle 5: LDAP-Objekt für anonymisierte Messdatenquellen 8

Tabelle 6: Auswahlkriterien Relationale Datenbank 8

Tabelle 7: Erwartete Anzahl an Datensätzen 12

Tabelle 8: Informationen zu Authentifizierungsserver LDAP 22

Tabelle 9: Zugangsdaten zu OpenLDAP 22

# Listingverzeichnis

[Listing 1: CSV-Format der REDD-Daten 9](file:///C:\Users\maximilian\Githubs\R&D\Projektplanung\Endabgabedokumente\Endbericht_August%20-%20work%20in%20progress.docx#_Toc491890889)

[Listing 2: Berechnung des Durchschnittsverbrauchs pro Meter, Tag und Monat 10](file:///C:\Users\maximilian\Githubs\R&D\Projektplanung\Endabgabedokumente\Endbericht_August%20-%20work%20in%20progress.docx#_Toc491890890)

[Listing 3: Definition eines neuen Controllers 19](file:///C:\Users\maximilian\Githubs\R&D\Projektplanung\Endabgabedokumente\Endbericht_August%20-%20work%20in%20progress.docx#_Toc491890891)

[Listing 4: Erweiterung der Klasse Query<T> 20](file:///C:\Users\maximilian\Githubs\R&D\Projektplanung\Endabgabedokumente\Endbericht_August%20-%20work%20in%20progress.docx#_Toc491890892)

[Listing 5: Erweiterung der Klasse QueryResult<T> 21](#_Toc491890893)

[Listing 6: Abfrage mehrerer Messdatenreihen 30](#_Toc491890894)

[Listing 7: Create Index Statement 31](#_Toc491890895)

[Listing 1: JSON Struktur für ein Tupel von Messwerten 32](#_Toc491890896)

[Listing 9: Datenimport - mongoDB 32](#_Toc491890897)

[Listing 10: CSV-Format der Rohdaten 32](#_Toc491890898)

[Listing 11: REST-Anforderung, um auf Daten zuzugreifen 33](#_Toc491890899)

# Einleitung

Durch die Verabschiedung der Richtlinie 2009/72/EC [1] sind die Mitgliedsstaaten der EU aufgefordert, deren Inhalte in nationales Recht umzusetzen. Thema dieser Richtlinie ist es, die vorhandenen analogen Stromzähler durch digitale Smart Meter zu ersetzen. Mit der flächendeckenden Installation stehen sowohl den Netzbetreibern als auch den Energieproduzenten und den Verbrauchern Möglichkeiten das Netz optimal zu nützen, Energie zu günstigen Preisen zu erwerben und Energieverschwendung zu verringern [2]. Um diese Vorteile zu nützen, ist Kommunikation bezüglich des aktuellen Verbrauchs, der Netzbelastung und der im Netz vorhandenen Energie notwendig.

Über Kommunikationsprotokolle tauschen Verteilstationen, Energieeinspeiser und Smart Meter beim Endkunden Daten bezüglich des Verbrauchs aus. Der Preis für diese Vorteile ist die notwendige, zumindest teilweise Offenlegung des Energieverbrauchs des Endkunden.

Im Spannungsfeld von Schutz der Privatsphäre einerseits, und maschineller Messdatenauswertung im Rahmen des Erlaubten andererseits sollen die Ergebnisse dieses Projekts für Komfortverbesserung sorgen.

## Problemstellung und Motivation

Für die Ablage und Verwaltung von Smart Meter Messdaten setzt das JRZ eine hausintern geschriebene Datenbankapplikation (JRZ-DB) ein. Es bestehen bereits einige Importprogramme die Messdaten aus unterschiedlichen Formaten, zum Beispiel REDD [3], UK-DALE [4], ADRES [5] und GREEND [6], in das Datenbankformat konvertieren. Hingegen existiert kein zentrales Zugriffsmodul, jede Anwendung, die auf diese Daten zugreift, benötigt das Wissen über den Tabellenaufbau, die Zugriffspfade sind je nach Anforderung neu zu implementieren und in der Folge zu warten. Nach der Umsetzung dieses Projekts steht existierenden und zukünftigen Applikationen ein vereinheitlichter und damit vereinfachter Zugriff zur Verfügung.

Die Speicherung der Messdaten aus unterschiedlichen Quellen in einer Datenbankinstanz kann in Verbindung mit uneingeschränktem Zugriff datenschutzrechtliche Probleme aufwerfen, andererseits ist der gleichzeitige Zugriff für vergleichende Auswertungen notwendig. Die Einbindung einer externen Komponente zur rollenbasierten Zugriffregelung löst diese Anforderung, der Zugriff auf Messdaten im Allgemeinen und auf bestimmte Auflösungen im Speziellen wird über Berechtigungen des Benutzers gesteuert.

Dieses Projekt verfolgt vier Hauptziele:

* Schaffung einer erweiterbaren Programmierschnittstelle (SmartValAPI), die einen geregelten Zugriff auf Smartmeterdaten ermöglicht
* Einbindung und gegebenenfalls Erweiterung der im JRZ eingesetzten Datenbank (JRZ-DB, Details siehe Abschnitt Anforderungen an das ER-Modell) als einheitliche Datenplattform für bereits existierende Anwendungen
* Evaluierung alternativer Datenbanksysteme zur Ablage der Messdaten
* Einbindung einer rollenbasierten Zugriffsverwaltung

Nach der erfolgreichen Umsetzung des Projektes steht der Zugriff auf alle gespeicherten Messdaten, den Berechtigungen entsprechend, für programmtechnische Auswertungen in vereinheitlichter Form zur Verfügung.

## Umgebung

Die Umsetzung des Projektes erfolgt unter zu Hilfenahme von bereits im Umfeld des Josef Ressel Zentrums für Anwenderorientierte Smart Grid Privacy, Sicherheit und Steuerung an der Fachhochschule Salzburg durchgeführten Projekte. Im Detail sind dies:

* Datenmodell: die JRZ-DB stellt die Ausgangsbasis für mögliche, notwendige Erweiterungen dar und wird auf deren Eignung für die zu erwartenden Messdatenmengen evaluiert.
* Importmodule [7], um Messdaten in der Datenbank abzulegen. Dieses Programmpaket ermöglicht es, Messwerte, die in den Formaten ADRES, GREEND, REDD und UK-DALE vorliegen zu importieren.
* OpenTC [8] stellt eine rollenbasierte Authentifizierung und Autorisierung zur Verfügung, über die der Zugriffsschutz realisiert wird, die Rollenverwaltung erfolgt über ein beliebiges LDAP-Administrationswerkzeug.

Weitere verwendete Softwarepakete werden im Abschnitt Systemarchitektur angeführt.

# Anforderungsanalyse

Aus dem Projektauftrag können Anforderungen abgeleitet werden, die Details dieser Anforderungen und den Weg zur jeweiligen Entscheidungsfindung beleuchtet dieser Abschnitt.

## Datenmodell

Ausgangsbasis für die Anforderungen ist die bereits eingesetzte JRZ-DB, analysiert wird die Eignung für die Datenhaltung. Anforderungen von Energieversorgern und Netzdienstleistern fließen in die Untersuchung mit ein. Nicht zuletzt wird die Kompatibilität zu existierenden Anwendungen erhalten.

## Analyse der JRZ-DB

Die Tabellenwelt der JRZ-DB kann in zwei Gruppen eingeteilt werden: einerseits Stammdaten, wie zum Beispiel meter\_management, meter\_type oder customer\_data und anderseits Bewegungsdaten: meter\_data. Letztere Tabelle ist über einen Fremdschlüssel mit meter\_management verbunden. Die Welt der Stammdaten wird ohne Änderungen übernommen.

Alle Messdaten, die von einem Smart Meter zu einem Zeitpunkt ausgelesen werden, speichert die JRZ-DB als Tupel in der Tabelle meter\_data. Abgelegt. Nicht jedes Smart Meter stellt alle Werte zur Verfügung, in der JRZ-DB werden nicht vorhandene als NULL-Wert gekennzeichnet. Je Messzeitpunkt können folgende Werte abgelegt werden.

* Nutzdaten (Momentanwerte):

je Phase, aktuelle Leistung (power\_p1, power\_p2, power\_p3), aktueller Stromverbrauch (work \_p1, work \_p2, work\_p3), 4 freie Werte (count\_register1 – count\_register4), die abhängig vom Typ des Smart Meter (meter\_type) belegt werden,   
Gesamtwerte für Spannung (voltage) und Frequenz (frequency),  
kumulierter Verbrauch (count\_total).

* Verwaltungsdaten (zur Identifikation):

Identifikationsnummer des Smart Meter (meter\_id), Fremdschlüssel zu meter\_management, eindeutiger Schlüssel des Messdaten-Tupels (data\_id),  
Erstellungszeitpunkt zu dem die Nutzdaten aufgezeichnet werden (timestamp).

Das bestehende Modell der JRZ-DB gibt Abbildung 1 wieder.



Abbildung 1: Datenmodell JRZ-DB

## Anforderungen von Energieversorgern und Netzdienstleistern

Neben den Anforderungen des JRZ, als Auftraggeber, werden die möglichen Bedürfnisse von Energieversorgern und Netzbetreibern ermittelt, dazu wurde ein Fragenkatalog erstellt und an folgende 10 in diesem Feld tätigen Unternehmen übermittelt: Ebner Strom GmbH, Energie AG, Energie Steiermark, EVN AG und Netz Niederösterreich GmbH, Linz AG, Salzburg Netz GmbH, TINETZ – Tiroler Netze GmbH, Vorarlberger Energienetze GmbH und Wien Energie. Diese Fragen wurden gestellt:

* Welche Messwerte, abgesehen von Spannung, Strom und Wirkleistung, jeweils phasengetrennt sind für Sie als Energieversorger/Netzbetreiber von Bedeutung?
* In welcher Granularität (einzelne Smart Meter/Gruppen z. B. Trafostation) sollen diese Messdaten zur Verfügung stehen?
* Mit welcher/n Abtastrate/en soll/en diese Messwerte zur Verfügung stehen?
* Welche Werte, auf Basis der in der JRZ-DB sind verzichtbar?
* Weitere freie Hinweise.

Das Ergebnis zeigt Tabelle 1, auf Grund der Forderung einiger Betriebe, in anonymisierter Form zusammengefasst.

|  |  |
| --- | --- |
| **Zusätzliche Messwerte** | |
| zusätzliche Werte (Phasenlage) | 2 |
| Kein Interesse an Erweiterung bzw. Verweis auf das ElWOG [9] | 2 |
| Keine Auskunft | 6 |
| **Granularität** | |
| Gruppierung | 1 |
| Keine Auskunft | 9 |
| **Auflösung** | |
| Messwerte in der vorliegenden, dem ElWOG [9] entsprechenden, Auflösung sind ausreichend | 4 |
| Keine Auskunft | 6 |
| **Verzichtbare Werte** | |
| Netzfrequenz | 2 |
| Keiner der Werte soll weggelassen werden | 5 |
| Keine Auskunft bzw. keine Antwort | 3 |
| **Weitere Hinweise** | |
| Datenschutzkritische Anwendung | 3 |
| Verweis auf ElWOG, daher gilt „interessant“ nicht | 1 |

Tabelle 1: Rücklauf der Anfragen bei Energieversorgern

## Anforderungsprofil „Lastenheft Österreichs Energie“

Österreichs Energie, als Interessensgemeinschaft der österreichischen Netzbetreiber und einem Großteil der österreichischen Energieversorger, fasst im Zuge des Vergabeverfahrens der Smart Meter Einführung die zu erfüllenden Anwendungsfälle in [10] und der Arbeitsunterlage zur Erstellung eines Lastenheftes in [11] zusammen. Diese Anwendungsfälle beschreiben vor allem die Anforderungen an die Fernsteuerung von bereits beim Kunden installierten Smart Meter Geräten und nur am Rande mit der inhaltlichen Bedeutung der übertragenen Messdaten. Neben den, in der JRZ-DB in *meter\_data* bereits vorhandenen Datenfeldern, bietet das Lastenheft optional die Möglichkeit der Auslesung der Blindleistung.

## Rechtliche Rahmenbedingungen

Die Erfassung, Übertragung und Speicherung von Smart Meter Messdaten wird in vier Richtlinien geregelt:

* ElWOG [9]
* GDPR [12]
* IMA-VO [13]
* DAVID-VO [14]

Geregelt werden einerseits Mindestanforderungen an Smart Meter, andererseits die Inhalte und die Frequenzen, mit denen die Werte ausgelesen werden dürfen. Nach §84 ElWOG müssen dem Verbraucher die Daten bezüglich des „Verbrauchs der über ein intelligentes Messgerät gemessen wird“ zeitnah zur Verfügung zu gestellt werden. Es erfolgt keine genauere Definition, welche Daten dies im Detail sind, lediglich die Frequenzen, mit denen ausgelesen wird, sind festgelegt. Details dazu, finden sich im Abschnitt Rollenidentifikation. Bezüglich der auslesbaren Daten legt das Lastenheft von Österreichs Energie [11] jene Messwerte fest, die von einem Smartmeter übertragen werden müssen. Die IMA-VO befasst sich mit der Einführung der der intelligenten Zähler. die GDPR regelt die kontrollierte Löschung nach der Verarbeitung von Messdaten als Form der personenbezogenen Daten. Beide haben daher keinen Einfluss auf das Datenmodell. Die Weitergabe der erfassten Daten an berechtigte Dritte wird von der DAVID-VO geregelt. Diese Verordnung hat keinen Einfluss auf das Datenmodell, wohl aber auf die maximale Frequenz, mit der Messdaten ausgelesen werden können. Die Details dazu werden daher im Abschnitt Rollenidentifikation angeführt.

## Anpassung des Datenmodells

Auf Grund der Rückmeldungen der Energiedienstleister kommt es, in Übereinstimmung mit den Usecases von Österreichs Energie zur Erweiterung von meter\_data.

Weitere Datenfelder, wie sie zum Beispiel das ETSI in der Definition des OSGP [15] vorschlägt, werden nicht in die Menge der gespeicherten Daten aufgenommen, da diese weder von den Energieversorgern in den Rückmeldungen noch von den Netzbetreibern im Lastenheft gewünscht werden. Abbildung 2 zeigt die erweiterte Tabelle meter\_data.

Um die Vorgabe der Kompatibilität zu erfüllen werden keine Felder aus der Tabelle meter\_data entfernt, auch wenn zum Beispiel auf die Netzfrequenz aus Sicht der Netzbetreiber verzichtet werden könnte.



Abbildung 2: erweiterte Tabelle meter\_data

## Rollenbasierter Zugriff

Werden Messdaten aus unterschiedlichsten Quellen in einer gemeinsamen Datenbank abgelegt, darf der Zugriff darauf nur jenen Benutzern gewährt werden, die dazu berechtigt sind. Neben der Verhinderung von nicht autorisierten Zugriffen auf fremde Messdaten muss eine weitere Anforderung erfüllt werden: die Wartung des Zugriffs über Rollen. Bei einem rollenbasierten Zugriffsmodell werden einzelne Berechtigungen an Rollen zugeteilt und nicht an Benutzer vergeben. Das vereinfacht die Administration und ermöglicht eine flexible Anpassung der Berechtigungen [16].

## Rollenidentifikation

Die Messdatenquellen können auf Grund der Analyse in zwei Gruppen, die unterschiedliche Anforderungen an das Zugriffsystem stellen, eingeteilt werden.

## Messdaten aus dem Echtbetrieb

Für die Erfassung und damit einhergehend dem Zugriff auf Messdaten aus dem Echtbetrieb gelten im Wesentlichen zwei Verordnungen: das ElWOG [9] und die DAVID-VO [14]. Das ElWOG legt in §84 die Rahmenbedingungen für die Erfassung von Messdaten zum Zwecke der Verrechnung, der Kundeninformation, der Energieeffizienz, der Energiestatistik und der Aufrechterhaltung eines sicheren und effizienten Netzbetriebes fest. Abs. (1) legt Periode der Erfassung auf 15 Minuten fest. Die Weitergabe der Viertelstundenwerte ist nur bei ausdrücklicher Zustimmung des Endverbrauchers möglich (Abs. (2)), in begründeten lokalen Einzelfällen, zur Aufrechterhaltung eines sicheren Netzbetriebes, ist die Weitergabe von Viertelstundenwerten ohne ausdrückliche Erlaubnis möglich. Ohne Zustimmung zur Weitergabe steht ein Messwert je 24 Stunden zur Verfügung.

Die DAVID-VO §4 regelt die Übertragung der Messdaten und in der Folge die Bereitstellung für den Kunden über eine Website. Auf Kundenwunsch wird hier die kleinstmögliche Auflösung dargestellt, ohne diese Zustimmung wird ein Wert je 24 Stunden dargestellt. Die Weitergabe erfolgt an den Energielieferanten und optional an einen berechtigten Energieberater.

Messdaten werden in Verbindung mit Kundendaten nach [17] zu einer Ausprägung von personenbezogenen Daten. Es ist daher der Artikel 17 der GDPR [12] anzuwenden. Es wird darin ausgesagt, dass Daten, welche für die Zwecke, für die sie erhoben wurden, nicht mehr benötigt werden, zu löschen sind. Da einerseits der Datenimport keinen Teil dieses Projektes darstellt und andererseits davon ausgegangen wird, dass zur Verfügung gestellte Messdaten anonymisiert wurden, finden die beiden Regelwerke keine Anwendung. Es wird daher im Zuge dieses Projektes in der Folge nicht weiter auf die Löschung von Messdaten eingegangen.

## Anonymisierte Messdaten aus dem akademischen oder Forschungsbereich

Bezüglich der anonymisierten Messwerte gelten diese rechtlichen Einschränkungen nicht. Eine Beschränkung des Zugriffs auf Daten aus Quellen wie zum Beispiel REDD, UK-DALE, ADRES oder GREEND ist aus diesem Blickwinkel nicht notwendig. Manche der anonymisierten Datensätze, beispielsweise REDD, stehen nur mit Benutzername/Passwort-Zugriff zur Verfügung, sie werden daher auch nicht jedem Benutzer des API zugänglich gemacht. Gekennzeichnet werden Messwerte aus anonymisierten Quellen über ein LDAP-Attribut, Details dazu folgen im Abschnitt 2.2.4 .

## Einbeziehung zusätzlicher Domänen

Aktuell wurde die die Richtlinie 2006/3 2/EG in Österreich für die Datenerfassung und Kommunikation von Messgeräten für elektrische Energie umgesetzt, derzeit gibt es in Österreich keine äquivalenten Grundlagen für Gas, Wärme und Wasser. Technisch ist eine Erweiterung um jene Felder, die nach der rechtlichen Festlegung erfasst werden sollen, problemlos möglich, daher wird ein Einsatz in weiteren im Zuge der Umsetzung dieses Projekts nicht weiter in Betracht gezogen.

## Rollendefinition

Aus den in den Abschnitten 2.2.1.1 und 2.2.1.2 angestellten Überlegungen werden folgende Rollen mit den ihnen zugeordneten Auflösungen abgeleitet:

|  |  |
| --- | --- |
| **Rolle** | **maximale Auflösung** |
| Smart Meter Besitzer | 15-Minuten |
| Netzbetreiber, Energieversorger | 24-Stunden |
| Energieberater | 15-Minuten |

Tabelle 2: maximale Auflösung auf Grund einer zugeteilten Rolle

|  |  |
| --- | --- |
| **Rolle** | **Zugriff** |
| Akademischer oder Forschungsbenutzer | Höchste vorhandene Auflösung |

Tabelle 3: Zugriff auf anonymisierte Messdaten aus Forschungsquellen

In Tabelle 2 und Tabelle 3 werden die Zugriffsberechtigungen mit den dafür notwendigen Rollen in Verbindung gebracht und dargestellt. Anzumerken ist, dass für einen Zugriff auf nicht anonymisierte Daten nicht nur die jeweilige Rolle vonnöten ist. Zusätzlich wird eine Zuordnung zwischen Benutzer und Smart Meter hergestellt. Diese grundlegende Berechtigung wird in der Implementierung als Attribut zum Benutzer abgelegt.

## Verbindung zur Rollenverwaltung

Für die Administration der Zugriffsberechtigungen wird die Komponente Open-TC [8] eingesetzt. Alle rollenrelevanten Attribute werden hier abgelegt. Eine Übersicht der dieser Attribute zeigt Tabelle 4.

|  |  |
| --- | --- |
| **Attribut** | **Mögliche Werte** |
| Rolle | Besitzer, Energieversorger, Energieberater, Forscher |
| Kunde | customer\_id |

Tabelle 4: LDAP-Attribute zu Benutzer-Objekten

Um im bestehenden Datenmodell keine Erweiterungen bezüglich des Zugriffes machen zu müssen, werden Smart Meter, die Messdaten aus akademischen oder Forschungsquellen enthalten werden mit der objectClass Computer abgelegt, das identifizierende Attribut verweist, wie in Tabelle 5 zu sehen, auf die meter\_id.

|  |  |
| --- | --- |
| **Attribut** | **Mögliche Werte** |
| cn | meter\_management.meter\_id |
| MeterType | open |

Tabelle 5: LDAP-Objekt für anonymisierte Messdatenquellen

## Datenbankanforderungen

Ziel des Projekts bezüglich der verwendeten Datenbank sind die Weiterverwendung des bestehenden Datenmodells und die Recherche nach möglichen Alternativen.

Für die Weiterverwendung des bestehenden Modells spricht, dass es bereits mehrere Komponenten gibt, die dieses System verwenden [7], [18]. Für die Ablösung des bestehenden Modells spricht eine eventuell bessere Performance alternativer Datenbanken bei großen Datenmengen. Da die Weiterverwendung gewünscht wird, werden zuerst Performancemessungen auf einem relationalen Datenbanksystem durchgeführt. Ist die Performance ausreichend, wird diese Datenbank verwendet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Eigene Vorkenntnisse** | **Popularität** | **Open Source (Kostenlos)** |
| MySQL | 3 | 3 | 1 |
| PostgreSQL | 1 | 2 | 1 |
| MariaDB | 0 | 1 | 1 |

Tabelle 6: Auswahlkriterien Relationale Datenbank

Für die Auswahl der Datenbank wurden die Vorkenntnisse und freie Verfügbarkeit als Kriterien definiert. Da die Erfahrung mit Datenbanken in der Projektgruppe allgemein eher gering ist, ist die erwartete Unterstützung aus der Community ein weiteres Kriterium. Dafür wurde die Anzahl der Fragen innerhalb einer Woche auf Stack Overflow verwendet. Nach den Kriterien aus Tabelle 6 ist die Wahl auf MySQL gefallen.

Zum Zeitpunkt der Messung sind wir von ungefähr 250.000.000 Datensätzen innerhalb eines Jahres ausgegangen. Das ergibt sich aus 500.000 Smartmetern der Salzburg AG (1 Wert pro Tag) und ungefähr 60.000.000 Messwerten aus den REDD-Daten.

Als weitere Argumente für MySQL sprechen die kostenlose Verwendbarkeit, die Verfügbarkeit von einsatzfähigen Paketen unter dem Windows Betriebssystem und die Verteilung eines Datenbankservers auf mehrere Computer, siehe dazu auch Abschnitt 2.5.

## Testdaten

Als Testdaten wurden die ‚low\_freq‘ Daten aus dem REDD-Datensatz verwendet. Die Daten liegen in pro Haus in einem eigenen Ordner und dort pro Kanal in einem eigenen File. Die Files sind ‚channel\_X.dat‘ benannt wobei X eine fortlaufende Nummer ist. Parallel zu den Messwerten liegt eine Datei ‚labels.dat‘ in welche die Zuordnung der einzelnen Kanäle zu den Verbrauchern ermöglicht. Die Messwerte sind, wie in Listing 1 dargestellt, in einem CSV-Format abgespeichert, wobei das Trennzeichen ein Leerzeichen ist.

timestamp\_power

Listing 1: CSV-Format der REDD-Daten

## Messung

Für die Performance Messungen wurde eine C# Applikation geschrieben, welche sämtliche ‚low\_freq‘ REDD-Daten in eine MySQL Datenbank importiert. Dafür wurden die Datensätze pro Kanal geparst und in 100.000er Schritten in die Datenbank importiert. Aus dem Zeitstempel im Unix-Epoch-Format wurde lokal der Tag und der Monat ausgerechnet um später danach gruppieren zu können. Zusätzlich wurde pro Kanal eine fortlaufende Id, in Abbildung 3 als ‚meterId‘ dargestellt, vergeben.



Abbildung 3: Tabelle REDD mit Testdaten

Damit das Einfügen von 100.000 Datensätzen auf einmal funktioniert wurde die maximal erlaubte Paketgröße auf 160 MB erhöht [https://dev.mysql.com/doc/refman/5.7/en/packet-too-large.html]. Die Datenbank läuft auf dem lokalen Testsystem (i5 4690K @ 3.5GHz, 16GB Arbeitsspeicher, Windows 10 Pro) um einen möglichen Delay über das Netzwerk ausschließen zu können.

Nach dem Hochladen eines jeden Datensatzes wurde der Durchschnittsverbrauch pro Id, Monat und Tag abgefragt (Listing 2).

Listing 2: Berechnung des Durchschnittsverbrauchs pro Meter, Tag und Monat

**select** **SQL\_NO\_CACHE** **avg**(**power**) **as** **power**, **day**, **month**, meterId

**from** redd

**group** **by** meterId **asc**, **month** **asc**, **day** **asc**;

Durch die SQL\_NO\_CACHE Anweisung wird verhindert das das Ergebnis der Abfragen aus dem Cache zurückgeliefert werden, was einem realistischen Szenario entspricht. Die Abfrage wurde fünf Mal wiederholt ausgeführt und die jeweilige Zeit mittels der ‚System.Diagnostics.Stopwatch‘ Klasse gemessen.

Die Messungen wurden in einem XML Dokument abgespeichert um dann mit Excel weiterverarbeitet werden zu können. Aus den 5 Messungen wurde der Median und Mittelwert berechnet, welche dieselben Schwankungen zeigen. Daher wurde auf die Darstellung des Mittelwerts verzichtet.

Abbildung 4: Dauer Berechnung des Mittelwerts auf der MySQL Datenbank

Abbildung 4 zeigt, dass die Dauer der Berechnung des Durchschnittsverbrauches in linearem Zusammenhang zur Anzahl der Datensätze in der Tabelle ist. Die teilweise starken Schwankungen des Medianes lassen sich durch Hintergrundprozesse wie Virenscanner und verschiedene Updatedienste erklären. Die Formel der Regressionsgerade liefert im Gegensatz zur y-Achse eine erwartete Dauer, bei der Abfrage auf X Datensätzen, in Millisekunden.

Abbildung 5: Dauer des Einfügens in die MySQL Datenbank

Auch das Einfügen neuer Datensätze steht, wie in Abbildung 5 zu sehen in linearem Zusammenhang mit der Anzahl der neu eingefügten Datensätze. Die Tabelle mit den Messungen ist im Anhang zu finden.

## Erwartete Datenmengen

Die Datenbank soll eine Sammlung von Messdaten aus verschiedenen Quellen sein. Im Moment sind vier verschiedene Datenquellen bekannt (Tabelle 7).

|  |  |  |
| --- | --- | --- |
| **Name** | **Anzahl** | **Kommentar** |
| REDD | 56M | bisher nur low\_freq |
| ADRES | 36M | 30 Haushalte je 1 Sensor, 2 Wochen, 1Hz |
| GREEND | 2.270M | 8 Häuser je 9 Sensoren, 1 Jahr, 1Hz |
| Salzburg AG | 182M / Jahr | 500.000 Haushalte 1 Messung pro Tag |
| **Summe** | 2.544M |  |

Tabelle 7: Erwartete Anzahl an Datensätzen

Diese Datensätze zusammen gerechnet beinhalten ungefähr 2.544 M Einträge. Rechnerisch würde eine Abfrage des Durchschnittswertes auf dem Testsystem ungefähr 40 Minuten dauern. Da moderne Datenbankserver allerdings um ein vielfaches performanter als ein Heimcomputer sind, ist dieser Wert für ein Livesystem nicht aussagekräftig. Es zeigt sich ebenfalls, dass der derzeitig verfügbare Testserver (Xeon E5-2620 @ 2GHz, 4GB Arbeitsspeicher), für einen Betrieb mit allen Datensätzen nicht geeignet ist.

Mögliche Ansätze zur Erhöhung der Performance wären horizontale Skalierung mit einem SQL System oder vertikale Skalierung mit einem NoSQL System. Eine Lösung dieses Problems ist für das nächste Semester geplant.

## Systemarchitektur

Um die Systemumgebung festzulegen, und vor allem die Software passgenau in die Softwarelandschaft des JRZ einfügen zu können, werden die Schnittstellen von Open-TC [7] und das Modell der bestehenden Datenbank, siehe Abschnitt 2.1.1, verwendet. Mit der Ausnahme des Betriebssystems werden Open Source Komponenten eingesetzt. Neben der freien Verfügbarkeit und dem Betrieb ohne Lizenzkosten sprechen weitere Aspekte, wie zum Beispiel Unabhängigkeit, Einsatz offener Standards und oft eine schnellere Behebung von Sicherheitslücken [19]. Die eingesetzten Komponenten sind wie angeführt:

* Hardware: Für den Betrieb ist keine explizite Hardware vonnöten, vom JRZ wurde eine virtuelle Maschine im Bladecenter zur Verfügung gestellt: 2 Prozessoren Intel Xeon E5-2620 2GHz, 4 GB Hauptspeicher und 80 GB Plattenplatz.
* Die Identifikationsdaten des Produktivsystems sind:  
  Hostname: landsteiner.fh-salzburg.ac.at  
  IP-Adresse: 193.170.119.66
* Betriebssystem: Aufgrund der größeren Erfahrung der Entwickler mit der Administration erfolgt der Betrieb auf einem Windows System (Windows Server 2012 R2).
* Java als Programmiersprache ist eine Vorgabe, da bestehende Software im Umfeld des JRZ bereits damit umgesetzt wurde.
* Die Entwicklung der Software erfolgt mit zwei unterschiedlichen Entwicklungsumgebungen: Eclipse Neon oder IntelliJ IDEA. Beide Entwicklungsumgebungen unterstützen die Entwicklung von Java Programmen, der jeweilige Einsatz erfolgt auf Grund der persönlichen Präferenzen der Entwickler.
* Als RDBMS wird MySQL eingesetzt, nach „Guide to Scaling Web Databases with MySQL Cluster“ [20] kann der zu erwartenden Menge von Schreibzugriffen (~500k/d) durch Verteilung auf einzelne Nodes begegnet werden. Sollte ein Cluster mehrerer Nodes zu Performanceeinbußen führen können einem Cluster einfach weitere Nodes hinzugefügt werden, eine Anpassung der Applikation ist in diesem Fall nicht notwendig.
* Als Datenbankdesigntool wird, passend zur Datenbank die MySQL Workbench 6.3.9 eingesetzt
* Der Entwurf des API sieht die Übergabe eines generischen Benutzerkontextes vor, damit ist die Anbindung von Open-TC oder einer anderen Benutzerverwaltung ohne Änderungen am API möglich.
* Die Schnittstelle zu Importmodulen ist einzig und allein die Datenbank, da hier das Schema vorgeschrieben ist, sind die Anforderungen an die Importmodule klar definiert und es bestehen keine programmseitigen Abhängigkeiten.
* Die öffentlichen Schnittstellen sind im Abschnitt 2.4 detailliert dargestellt.
* Aus den Anforderungen ergeben sich folgende Komponenten, deren Abhängigkeiten untereinander und Verbindungen zueinander, und als UML Diagramm in Abbildung 6 dargestellt werden.



Abbildung 6: Komponentenmodell

Der Übersicht halber wird die Komponente „Logger“, die jede der angeführten Komponenten verwendet, nicht in der Übersicht dargestellt.

## Alternative Datenhaltung

Die JRZ-DB ist für aktive Anwendungen wie zum Beispiel Open-TC auf Basis einer MySQL Datenbank im Einsatz. Bestehende Anwendungen, wie zum Beispiel Smart Vis [18] zeigen, dass die Verarbeitungsgeschwindigkeit für Messdatenmengen in der Größe von akademischen und Forschungsquellen ausreichend ist. Die im Abschnitt 2.3.3 angeführten Zahlen führen, insbesondere bei einem zentralen und dauerhaften Betrieb, zu Datensatzmengen, die Überlegungen bezüglich einer alternativen Datenhaltung notwendig machen. Auch wenn es keine einheitliche Definition für den Begriff Big Data gibt [21], ergibt die Anzahl der Messdatensätze eine Datenmenge, die im praktischen Betrieb möglicherweise zu verlängerten Antwortzeiten führen wird.

Die vorhandene Hardware und die bisher durchgeführten Tests lassen noch keine konkreten Aussagen über die Verarbeitungsgeschwindigkeit von Messdatenmengen in der Größenordnung von ca. 500.000 Datensätze pro Tag, wie sie von zum Beispiel der Salzburg AG [22] zu erwarten sind, zu. Abhängig von den Reaktionszeiten werden folgende Vorgehensweisen in Betracht gezogen:

* MySQL unterstützt die Fragmentierung einzelner Tabellen, sowohl lokal („partitioning“) als auch verteilt („sharding“). Nach „Guide to Scaling Web Databases with MySQL Cluster“ [23] kann der zu erwartenden Menge von Schreibzugriffen durch Verteilung auf einzelne Nodes begegnet werden. Sollte ein Cluster mehrerer Nodes zu Performanceeinbußen führen können einem Cluster einfach weitere Nodes hinzugefügt werden, eine Anpassung der Applikation ist in diesem Fall nicht notwendig, über die Administration des MySQL Clusters wird das Partitionierung transparent für die Anwendung durchgeführt.
* Fusco et al. schlagen in [24] einen dualen Betrieb von RDBMS und einer NoSQL Datenverwaltung vor. Die Messwertedatenpakete werden in unterschiedlicher Granularität, zum Beispiel Rohdaten, Messdaten aggregiert nach Smartmeter, Zeitraum und vorverarbeitet, zum Beispiel Durchschnittsverbrauch über einen bestimmten Zeitraum, abgelegt. Ziel dieses Ansatzes ist es auch Vorhersagen über den zukünftigen Verbrauch zu tätigen. Jene Daten die Messwerte betreffen werden in einer separaten Komponente verwaltet, dadurch soll ermöglicht werden die Datenhaltung mittelfristig ersetzen zu können. Details siehe sin im Abschnitt „Systemarchitektur“ angeführt.

Ein dezentraler Ansatz, wie zum Beispiel das COUGAR Sensornetzwerk [25], als Alternative zu einer zentralen Datenbank bietet zwar den Vorteil, den einen, zentralen Angriffspunkt zu vermeiden, Messdaten hingegen ausschließlich ad hoc auszulesen widerspricht den Regelungen des ElWOG und scheidet daher aus den Alternativen aus.

# Umsetzung

Dieses Kapitel beschreibt die konkrete Implementierung der Applikation.

Um sich auf die Domäne konzentrieren zu können wurde das Spring Boot Framework[[1]](#footnote-1) verwendet welches die technischen Details wie Bereitstellung der Applikation mittels eines Tomcat Servers, Abwickeln der HTTP Requests, Einbinden von Hibernate, Dependency Injection, Darstellung der API Dokumentation mittels Swagger UI[[2]](#footnote-2) und die Konfiguration mit einem „Convention over Configuration“ Ansatz kapselt.



Abbildung 7: Bestandteile der API

Abbildung 7 zeigt die Aufteilung der API in drei Teile, welche sich durch die Resource nach der Basis URL auswählen lassen. Resourcen werden in Spring mittels der @Controller Annotation definiert. Eine detaillierte Beschreibung erfolgt im Kapitel 3.4.

## CRUD API

Der erste Teil ist über die Resource /admin/<table-name> erreichbar. Der Benutzer muss in der LDAP Datenbank als Administrator registriert sein. Hier hat der Benutzer Zugriff auf alle Tabellen der Datenbank und kann lesend sowie schreibend darauf zugreifen. Diese Schnittstelle ist dazu gedacht um die Anwendung später erweitern zu können. Beispielsweise können Daten damit automatisiert importieren sowie exportieren werden. Diese Schnittstelle abstrahiert die dahinterliegende Datenbank, sodass auch diese austauschbar ist ohne darauf aufbauende Applikationen verändern zu müssen. Damit ein Austausch der Datenbank auch für die API möglichst wenig Veränderung benötigt wurde hier Hibernate[[3]](#footnote-3) als Data Access Layer verwendet. Die verwendete Datenbanktechnologie kann in der Konfiguration[[4]](#footnote-4) mittels der Eigenschaft „spring.datasource.driver-class-name“ gesetzt werden. Hier sind auch die URL sowie der Benutzername und das Passwort zu setzen.

## Custom API

Der mächtigste Teil der API ist über die Resource /query/<controller-name> zu erreichen. Dieser Teil lässt dem Entwickler völlige Freiheit über die Datenbankabfragen und stellt ein erweiterbares Framework dar. Durch die Implementierung weniger Interfaces kann der Funktionsumfang erweitert werden. Dem Entwickler werden dabei Aufgaben wie die Authentifizierung über LDAP oder der Verbindungsaufbau zur Datenbank vereinfacht. Um die größtmögliche Freiheit zu ermöglichen werden Abfragen hier direkt auf der Datenbank durchgeführt. Falls die Datenbanktechnologie geändert werden soll, müssen diese Abfragen angepasst werden. Es wäre natürlich auch hier möglich Hibernate einzusetzen, um diese Abhängigkeit zu minimieren, allerdings sollten alle Möglichkeiten aufgezeigt werden die dieses Framework bietet.

## API Dokumentation

Um die API weitgehend automatisiert zu dokumentieren und auch testen zu können, wurde SWAGGER UI verwendet. Die Dokumentation ist über die Seite /swagger-ui.html erreichbar. Durch einen automatisierten Scan über alle bereitgestellten Ressourcen wird die Dokumentation bei jedem Applikationsstart automatisiert generiert.

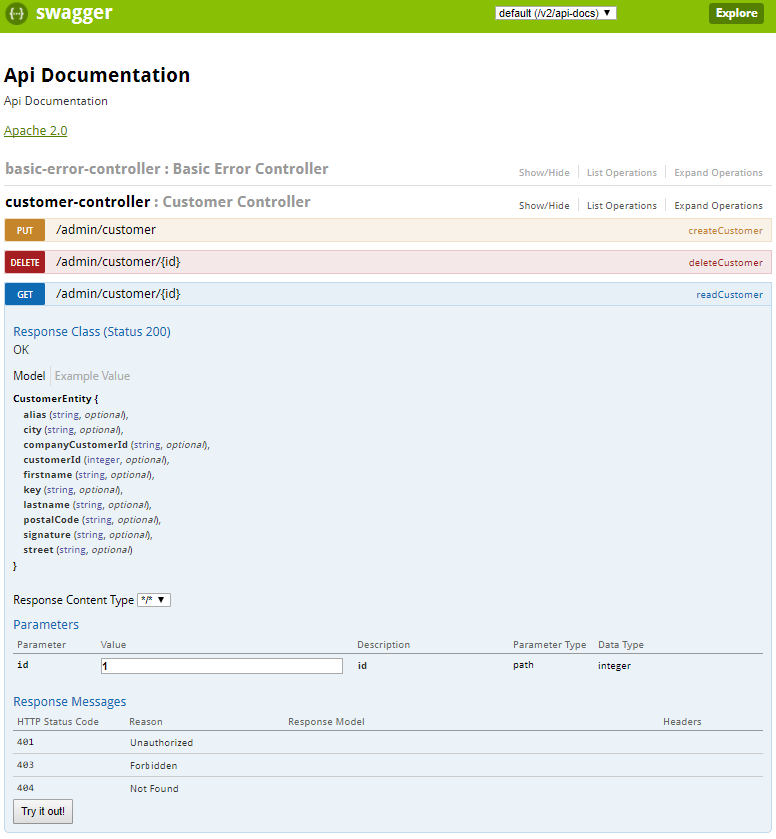


Abbildung 8: API Dokumentation am Beispiel /admin/customer

Abbildung 8 zeigt die möglichen Anfragen an die Ressource /admin/customer. Durch Aufklappen werden Informationen zu den zurückgelieferten Datentypen sowie zu möglichen Parametern der Abfrage gezeigt. Hiermit kann die API auch getestet werden. Spring wurde so konfiguriert, dass für die Darstellung der Dokumentation keine Authentifizierung notwendig ist. Wird eine Abfrage ausgeführt ist dies jedoch notwendig.

## Erweiterung der API

Durch die Erweiterung einiger abstrakter Klassen und Implementierung weniger Interfaces lässt sich das Framework erweitern, was hier an einem einfachen Beispiel gezeigt wird.

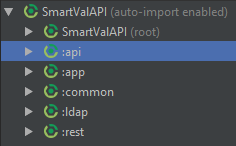


Abbildung 9: Projektstruktur

Die Applikation besteht wie in Abbildung 9 dargestellt aus fünf Projekten und einem Übergeordneten. Für die Erweiterung muss nur das „rest“ Projekt angepasst werden.

Listing 3: Definition eines neuen Controllers

@Controller  
**public class** DemoController **extends** CustomQueryControllerBase {  
  
 @RequestMapping(**"/query/demoquery/{lastName}"**)  
 **public** QueryResult<?> getCustomer(@PathVariable String lastName){  
 **return databaseAccess**.QueryDatabase(**null**);  
 }  
}

Um eine neue Ressource definieren zu können wird eine neue Klasse angelegt (Listing 5). Die Controller Annotation dient dazu, dass Spring die Klasse als Controller erkennt. Durch Ableiten von der Basisklasse CustomQueryControllerBase ist der Datenbankzugriff über Dependency Injection verfügbar. Anschließend wird eine Methode definiert, welche beim Aufruf der API ausgeführt wird. Durch die RequestMapping Annotation wird definiert über welchen Pfad diese Methode über die API bereitgestellt wird. Der Rückgabetyp QueryResult bietet Informationen wie einen Fehlercode oder Fehlertext an.

Im nächsten Schritt wird eine neue Query erzeugt welche der Methode QueryDatabase übergeben wird. Hierfür wird die abstrakte Klasse QueryBase<T> erweitert und die fehlenden Methoden implementiert.

**private class** DemoQuery **extends** QueryBase<List<CustomerEntity>> {  
  
 **private** String **\_lastName**;  
  
 **public** DemoQuery(String lastName) {  
 **super**();  
 **\_lastName** = lastName;  
 }  
  
 @Override  
 **public** IDataSourceContext[] getDataSourceContexts() {  
 **return null**;  
 }  
  
 @Override  
 **public** String getQuery() {  
 **return "select \* from customer where lastname = '"** + **\_lastName**+**"'"**;  
 }  
  
 @Override  
 **public** QueryResult parseDatabaseResultSet(ResultSet resultSet) {  
 List<CustomerEntity> returnList = **new** ArrayList<>();  
 **try**{  
 **while**(resultSet.next()){  
 CustomerEntity e = **new** CustomerEntity();  
 e.setLastname(resultSet.getString(**"lastname"**));  
 e.setFirstname(resultSet.getString(**"firstname"**));  
 e.setCustomerId(resultSet.getInt(**"customer\_id"**));  
 returnList.add(e);  
 }  
 }**catch**(SQLException e){  
 e.printStackTrace();  
 **return null**;  
 }  
 **return null**;  
 }  
}

Listing 4: Erweiterung der Klasse Query<T>

Die Methode getDataSourceContexts liefert ein Array mit Ids von Smart Metern. In diesem Fall werden keine Meter abgefragt, daher kann null zurückgeliefert werden. Die eigentliche SQL Query wird anhand der Parameter des REST Request zusammengesetzt und von der Methode getQuery zurückgeliefert. In diesem Fall wurde auf Schutz vor SQL Injection verhindert. Im Produktiveinsatz sollte das beachtet werden.

Wurde die Query erfolgreich ausgeführt, wird ein ResultSet Objekt an die Methode parseDatabaseResultSet übergeben. Hier wird die Antwort der Datenbank ausgewertet und abschließend über die REST Schnittstelle zurückgeliefert.

Um mehr Metainformationen über die Abfrage zurückgeben zu können wird die Antwort in eine Ableitung von QueryResult<T> gekapselt. Hierfür wird die Klasse QueryResult abgeleitet.

**private class** CustomerListQueryResult **extends** QueryResult<List<CustomerEntity>>{  
  
 **private** List<CustomerEntity> **\_data**;  
  
 **public** CustomerListQueryResult(List<CustomerEntity> data){  
 **super**();  
 **\_data** = data;  
 }  
  
 **public** CustomerListQueryResult(**boolean** isSuccessful, String errorMessage, QueryStatusCode queryStatusCode,

List<CustomerEntity> data){  
 **super**(isSuccessful,errorMessage,queryStatusCode);  
 **\_data** = data;  
 }  
  
 @Override  
 **public** List<CustomerEntity> getData() {  
 **return \_data**;  
 }  
}

Listing 5: Erweiterung der Klasse QueryResult<T>

## LDAP Zugriffsregelung

Um Benutzer in der SmartVal API zu authentifizieren, wurde LDAP auf einem Server installiert. Die SmartVal API stellt bei einer Anfrage eine Verbindung mit dem Server her, um eine Authentifizierung durchzuführen mittels Benutzername und Passwort. Sind die Zugangsdaten richtig und hat der Benutzer die benötigten Rechte, so wird eine entsprechende Response vom Server zurückgegeben.

## LDAP Server

Zur Installation des LDAP wurde OpenLDAP verwendet. Diese Software wurde auf folgendem Server installiert:

|  |  |
| --- | --- |
| Host: | landsteiner.fh-salzburg.ac.at |
| IP: | 193.170.119.66 |
| User: | Administrator |
| Passwort: | 9HxAJ39g8EnudLS7 |

Tabelle 8: Informationen zu Authentifizierungsserver LDAP

## OpenLDAP konfiguration

Um eine Verbindung zu OpenLDAP herstellen zu können, musste in der Software die Verbindung angelegt werden. Hierzu wurden die folgenden Informationen vergeben:

|  |  |  |  |
| --- | --- | --- | --- |
| Host: | 193.170.119.66 | Port: | 389 |
| Base: | dc=maxcrc,dc=com | Version: | 3 |
| Username: | cn=Manager,dc=maxcrc,dc=com | Simple Authentication | |
| Password: | secret |  | |

Tabelle 9: Zugangsdaten zu OpenLDAP

Weiters musste ein Zertifikat „development“ installiert werden. Dieses Zertifikat wurde in der Management Konsole unter „Vertrauenswürdige Zertifikate“ und „persönliche Zertifikate“ im lokalen Computerkonto hinzugefügt.

Um nun eine Hierarchie im OpenLDAP einzurichten, wurden die Kategorien

* Groups,
* People und
* Smartmeter

hinzugefügt. Somit konnte eine erste Unterscheidung zwischen Gruppen, Benutzern und Smartmeter hergestellt werden. In der Kategorie Groups wurden des Weiteren die Gruppen

* Administrator,
* Energieberater,
* Forschungszentrum,
* Kunden und
* Netzbetreiber

hinzugefügt. Somit konnte die Hierarchie zwischen Netzbetreiber über Energieberater bis hin zum einzelnen Smartmeter ermöglicht werden.

Mit diesen Einstellungen war der Server fertig aufgesetzt.

## Schnittstelle zur Authentifizierung in SmartVal API

Die Schnittstelle zur Authentifizierung wurde in zwei Klassen (LDAPManager und LdapContextSourceFactory) sowie in einem Interface (ILDAPManager) unterteilt.

Wie schon erwähnt, wurde SmartVal API mithilfe von Spring erstellt. Mit Spring war es möglich, die Frameworks

* org.springframework.security:spring-security-ldap und
* org.springframework.ldap:spring-ldap-core

zu ergänzen, welche ein erleichtertes Arbeiten mit LDAP ermöglicht.

Im ersten Schritt muss eine Verbindung zum OpenLDAP Server hergestellt werden.

**LdapContextSourceFactory**

In der Klasse LdapContextSourceFactory wurde die Funktion

*public static LdapContextSource getContextSource();*

implementiert. Dieser Funktion wurden die Zugangsdaten ersichtlich aus Tabelle 9 hinzugefügt. Bei Aufruf dieser Funktion wird so eine Verbindung direkt zum OpenLDAP hergestellt.

**LDAPManager**

Um alle Funktionen für die restliche API bereitzustellen, wurde das Interface ILDAPManager erstellt. Dieses Interface enthält alle Funktionen, welche betreffend die Authentifizierung als auch Erstellung oder Löschung von Daten auf dem LDAP Server liefert. Ausprogrammiert wurden alle Funktionen in der Klasse LDAPManager implementiert.

Alle folgenden Funktionen geben ein boolean true, wenn die Funktion problemlos durchgeführt wurde und false, falls es zu einem Fehler gibt zurück. Die Parameter IDataSourceContext als auch IUserContext werden aus der API mitgegeben, welche alle benötigten Daten mitliefern.

Von dieser Schnittstelle werden die folgenden Funktionen bereitgestellt:

* *boolean IsAllowedToAccess(IUserContext userContext, IDataSourceContext dataSourceContext);*

IsAllowedToAccess prüft, ob der mitgegebenen Benutzer die gewollten Berechtigungen besitzt. Wird false zurückgegeben, hat er keine Berechtigung. Hierbei kann direkt der Kunde, oder auch der Energieberater angegeben werden, welcher den Kunden des zugehörigen Smartmeters berät. Ebenso kann mit Mitgabe des Netzbetreibers ein Zugriff geprüft werden.

* *boolean CreateCustomer(IUserContext userContext, IDataSourceContext dataSourceContext);*

Diese Funktion bietet die Möglichkeit, einen neuen Kunden am LDAP Server anzulegen. Bei Erstellung eines Kunden muss ein Smartmeter mitgegeben werden.

* *boolean CreateConsultant(IUserContext source, IUserContext destination);*

Die Funktion CreateConsultant lässt einen Energieberater erstellen.

* *boolean CreateSmartMeter(IDataSourceContext dataSourceContext);*

Mit CreateSmartMeter wird ein neuer Smartmeter am OpenLDAP hinzugefügt.

* *boolean AddUserToGroup(IUserContext userContext, String Group);*

Diese Funktion fügt den mitgegebenen Benutzer einer Gruppe hinzu. Hiermit kann eine Hierarchie aufgebaut werden. Beispielsweise kann mit dieser Funktion ein Kunde X zur Gruppe Kunden hinzugefügt werden oder ein Energieberater Y zur Gruppe Netzbetreiber.

* *boolean AddUserToUser(IUserContext source, IUserContext destination);*

AddUserToUser bietet die Möglichkeit, beispielsweise einen Kunden zu einem Energieberater hinzuzufügen. Somit kann die Hierarchie besser umgesetzt werden.

* *boolean AddMeterToUser(IUserContext userContext, IDataSourceContext dataSourceContext);*

Mit dieser Funktion wird ein Smartmeter einem Kunden hinzugefügt.

* *boolean DeleteUser(IUserContext userContext);*

DeleteUser löscht einen Benutzer aus dem OpenLDAP.

* *boolean DeleteUserFromGroup(IUserContext userContext, String Group);*
* Diese Funktion Löscht einen Benutzer aus einer spezifischen Gruppe.
* *boolean DeleteUserFromAll(IUserContext userContext);*

Diese Funktion ist eine Hilfsfunktion von DeleteUser. Diese Funktion löscht den gelöschten Benutzer aus allen Einträgen und hält die Einträge sauber.

* *boolean DeleteMeterFromUser(IUserContext userContext, IDataSourceContext dataSourceContext);*

DeleteMeterFromUser entfernt den mitgegebenen Smartmeter aus der Mitgliedschaft des mitgegebenen Users.

* *boolean DeleteSmartMeter(IDataSourceContext dataSourceContext);*
* Mit dieser Funktion wird der mitgegebene Smartmeter aus dem OpenLDAP gelöscht.
* *boolean DeleteMeterfromAll(IDataSourceContext dataSourceContext);*

Diese Funktion ist eine Hilfsfunktion von DeleteSmartMeter und löscht den mitgegebenen Smartmeter aus allen Einträgen des OpenLDAP.

Mithilfe dieser Funktionen ist eine saubere Pflege des OpenLDAP möglich als auch eine Authentifizierungsoption, welche von der SmartVal API benötigt wird. Der Quellcode wurde im Anhang beigefügt.

## Installation

Da SmartValAPI einerseits in Java umgesetzt wurde und andererseits eine verteilte Installation der LDAP und Datenbankkomponente unterstützt, ist in dieser Konstellation lediglich eine Java Laufzeitumgebung vonnöten. Wird das gesamte Programmpaket auf einem Computer installiert, wird ein lauffähiges System, für das die in Abschnitt 3.6.1 aufgezählten Komponenten verfügbar sind, vorausgesetzt.

## Systemvoraussetzungen

Im Folgenden sind jene Softwarekomponenten, die für die Ausführung notwendig sind, aufgezählt. Enthalten sind all jene Programme, unter denen die Implementierung erfolgte, die Installation durchgeführt und die Ausführung getestet wurde. Alternativen sind möglich, aber nicht getestet.

* Betriebssystem: Windows Server 2012 R2, alternativ Windows 10 Home, Apple Mac OS X
* Datenbanksystem: MySQL Server 5.7
* Datenbankadministrationsclient: MySQL Workbench 6.3
* Java Umgebung: Java(TM) SE Runtime Environment (build 1.8.0\_141-b15)
* Netzwerkverbindung, sofern nicht nur lokal ausgewertet oder verteilt installiert wird
* Nur zur Durchführung der Tests: SOAPUI 5.3.0
* Nur für eine Erweiterung oder das Neuübersetzen der Applikation: IntelliJ IDEA ULTIMATE 2017.2

Datenbank und Applikation können zur Lastverteilung auf zwei Rechnern verteilt ausgeführt werden. Netzwerk und Firewall-Konfiguration müssen eine Verbindung ermöglichen, beziehungsweise zulassen.

## Einrichtung der Datenbank und der Tabellen

Jene Datenbank, mit der SmartValAPI eine Verbindung aufnimmt, ist in der Konfigurationsdatei /SmartValAPI/app/out/production/resources/application.properties festgelegt, ausgeliefert wird die Software mit diesen Einstellungen:

Datenbankserver: landsteiner.fh-salzburg.ac.at, Standardport für MySQL (3306)

Benutzername: smartvalapi, Passwort: smartval. Dem Benutzer sind die Berechtigungen INSERT, UPDATE, SELECT und DELETE einzuräumen, die Anlage ist gesondert mit einem geeigneten Datenbankclient durchzuführen, beispielsweise mit der MySQL Workbench oder dem MySQL Kommandozeilen Client.

Datenbankname: smart\_meter

Mit der Durchführung von zwei Skripts werden die Datenbank und die Tabellen eingerichtet:

Anlage des Schemas: Durchführung des Skripts: DB\_und\_Test\_Skripts/createSchema.sql

Anlage der Tabellen: Durchführung des Skripts: DB\_und\_Test\_Skripts/db\_meters.sql

## Start der Applikation

Die kompilierte Applikation wird mit der Eingabe von

java -jar SmartValAPI-1.0.0.war

gestartet. Verbindungen zum Datenbank- und zum LDAP-Server sind bereits beim Start notwendig.

## Automatisierte Tests

Um die bereitgestellten Tests erfolgreich durchzuführen, sind jene Daten in die Tabellen zu importieren, die mit DB\_und\_Test\_Skripts/importTestData.sql entladen wurden. Jene REST-Tests, die im Testplan enthalten sind, können mit SOAP UI durchgeführt werden (Workspace DB\_und\_Test\_Skripts/REST-Tests-SmartValAPI.xml öffnen) und Projekt (DB\_und\_Test\_Skripts/REST-Project-1-soapui-projecttestYYYYYY.xml) importieren. Danach können alle Tests gesamt, die Auswahl ist in Abbildung 7 dargestellt, oder wie in Abbildung 8 ersichtlich, einzeln durchgeführt werden. Abbildung 9 zeigt den Beginn des Ergebnisses einer Abfrage dreier meter\_ids über einen gemeinsamen Zeitraum.

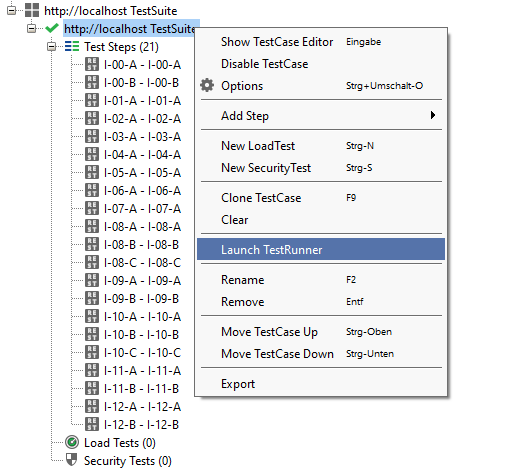


Abbildung 10: TestSuite, um alle Tests automatisiert durchzuführen

Abbildung 11: einzeln durchführbare REST-Testfälle

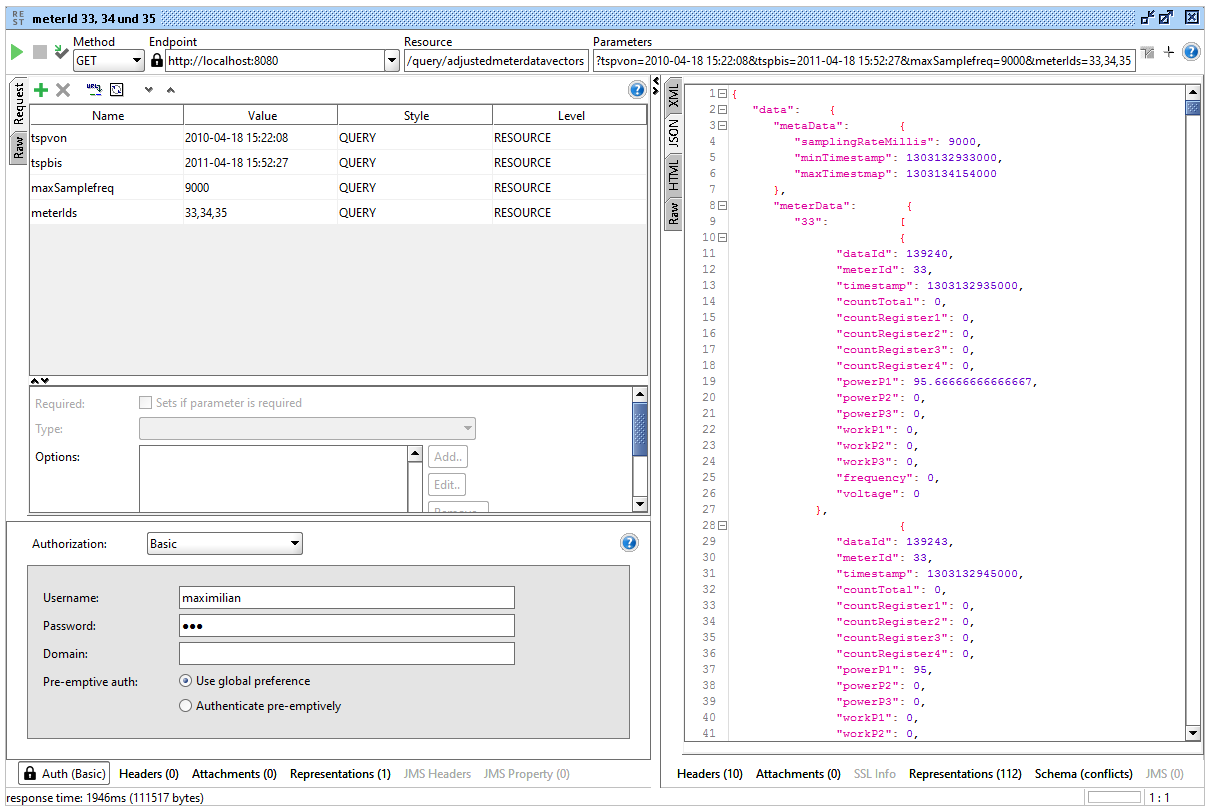


Abbildung 12: Testfall I-11-A mit drei Messdatenreihen in SOAP UI

## Verbesserungsvorschläge am bestehenden Datenmodell, die zu einfacherer Handhabung und Zugriffsbeschleunigung führen

Die im folgenden aufgezählten Vorschläge betreffen Anpassungen am bestehenden Datenmodell und wurden, um die Kompatibilität zu erhalten, im ausgelieferten Softwarepaket nicht berücksichtigt. Eine Ausnahme stellt der zusätzliche Index auf der Tabelle meter\_data dar, welcher im Bezug auf Kompatibilität keine Einschränkung bedingt.

## Benennung der Attribute

Durchgängige Benennung für gleiche Attribute über alle Tabellen hinweg, zum Beispiel manufactor\_id in meter\_manufactor jedoch id\_manufactor in meter\_management; id\_meter in meter\_management jedoch meter\_id in meter\_data; customer\_id in customer jedoch id\_customer in meter\_management. Auch wenn die Abbildung von Fremdschlüsselattributen im Namen möglich ist, erscheint eine unterschiedliche Benennung der Attribute verwirrend, da zumindest die Fremdschlüsselfelder konsistent benannt werden sollten, möglich wäre zum Beispiel „\_id“ am Ende des Namens beim Fremdschlüssel, im Gegensatz dazu beginnt ein Primärschlüsselfeld immer mit „id\_“. Für Auswertungen auf Tabellenbasis birgt die gleiche Benennung ein geringeres Fehlerpotenzial.

Vermeidung der Benennung von Attributen mit Schlüsselwörtern: Timestamp in meter\_data könnte zum Beispiel mit dataTsp benannt werden. Ein weiterer Kandidat für die Umbenennung ist „key“ in der Tabelle meter\_management: dieses Feld ist in SQL nur maskiert (select `key` from meter\_management;) auswertbar. Das Weglassen der Maskierung führt zu einer Fehlermeldung und zu Verzögerung bei der Fehlersuche.

## Schlüssel und Indizes auf der Tabelle meter\_data

Um den Zugriff auf Meterdaten optimieren zu können, werden bezüglich der Auswertungen zwei Annahmen getroffen. Erstens gehen wir davon aus, dass der Zugriff als Aufhänger immer die meter\_id enthält. Auswertungen, wie zum Beispiel „Gesamtsumme des Verbrauchs zu einem Zeitpunkt“ müssen seitens der Applikation verfeinert werden, etwa: alle meter\_Ids eines Versorgers werden ermittelt, Summe des Gesamtverbrauchs dieses Subsets zu einem Zeitpunkt berechnen. Zweitens wird die Mehrheit der Zugriffe nur einen geringen Anteil der Sätze von meter\_data je Zugriff lesen. Je mehr meter\_ids ihre Daten in meter\_data ablegen, desto eher wird diese Annahme gerechtfertigt. Bei einer Anzahl von Sätzen beziehungsweise wenigen abgelegten meter\_ids ist dies nicht erfüllt, es stellt in diesen Fällen ein Full-Table-Scan kein Performance Problem dar, da der Großteil der Einträge gelesen wird.

Auf Grund der Menge der Testdaten und der zur Verfügung stehenden Hardware werden die Untersuchungen auf die Optimierung des Zugriffspfades beschränkt.

Der primäre Schlüssel der Tabelle meter\_data besteht aus data\_id und meter\_id, wobei data\_id lediglich für die Eindeutigkeit des Schlüssels sorgt. Für Auswertungen von Zeitreihen zu meter\_ids ist jedoch der Zeitstempel von Belang was ein Umsortieren nach meter\_id und Timestamp notwendig macht. Eine Anpassung des Primärschlüssels (meter\_id, Timestamp) vermeidet das Umsortieren, greift aber in das Datenmodell ein, weiters wären alle Importprogramme von dieser Änderung betroffen und müssten angepasst werden.

MySQL legt für Fremdschlüssel einen Index an, im Falle von meter\_data ist das der Index fk\_meter\_data\_meter\_management1\_idx, dieser wird beim Zugriff auf Messdaten zu einer meter\_id verwendet. Der Zugriffspfad ist in Abbildung 10, links, dargestellt. Listing 5 zeigt eine Abfrage wie sie zum Beispiel von /query/adjustedmeterdatavectors erzeugt wird. Wird damit auf Messdatenwerte mehrerer Smartmeter zugegriffen, liegen die Daten so verstreut, dass der Zugriff nicht mehr über den Index, sondern über einen Tabellen-Scan erfolgt.



Abbildung 13: Zugriffspfade auf Messdaten (meter\_data)

SELECT \*

FROM METER\_DATA

WHERE METER\_ID IN (9, 10, 33)

AND TIMESTAMP BETWEEN '2011-04-16 07:11:30' AND '2011-04-16 09:19:30'

ORDER BY METER\_ID, TIMESTAMP;

Listing 6: Abfrage mehrerer Messdatenreihen

Abhilfe schafft hier die Erstellung eines Index, das Statement ist in Listing 6 dargestellt und in DB\_und\_Test\_Skripts/importTestData.sql enthalten. Das verbesserte Ergebnis wird in Abbildung 2, rechts gezeigt.

CREATE UNIQUE INDEX xmeteridTimestamp ON METER\_DATA

(METER\_ID, TIMESTAMP);

Listing 7: Create Index Statement

## Alternative Datenhaltung

Um mit den zu erwartenden Datenmengen in der Größe von jährlich 31 Millionen Messwerten pro Smartmeter auf Dauer umgehen zu können, wurde als Alternative zur relationalen Datenbank eine dokumentenorientierte Speicherungsform in SmartValAPI integriert. Als Datenbanksystem wurde MongoDB ausgewählt. Zu diesem Programmpaket steht ein Installationsprogramm für das Windows Betriebssystem zur Verfügung, das eine Out-of-the-Box-Installation ermöglicht, welche sich in die Softwarelandschaft des JRZ einfügt und keine zusätzlichen Konfigurations- oder Übersetzungstätigkeiten erfordert. Weiters kann das Paket als Open Source Software ohne Lizenzkosten eingesetzt werden.

## Datenbank und Collection

Wie in Abschnitt 2.5 erwähnt, werden, wie von Fusco et al. in [24] vorgeschlagen, lediglich die Messdaten in einer NoSQL Datenbank gehalten. Im Datenmodell des JRZ entspricht dies der Entität und in der Folge der Tabelle meter\_data. In der ausgelieferten Version erwartet SmartValAPI, dass der Import von Messdaten in der Datenbank meterData in der Collection meterTable erfolgte.

## Datenstruktur

Ein Tupel von Messwerten wird in der relationalen Speicherungsform als Zeile in einer Tabelle abgelegt (meter\_data). Für die dokumentenorientierte Ablage wurde eine analoge JSON-Struktur festgelegt. Listing 8 stellt die Struktur anhand eines JSON-Objektes dar. Sowohl die Struktur, die Namen der Felder und die Formate der einzelnen Werte sind vom Importprogramm zu gewährleisten. Eine Prüfung seitens der prototypischen Implementierung erfolgt zur Vermeidung von Performanceeinbußen nicht.

**{**

"\_id" **:** ObjectId**(**"59985033fe2010bb02fc1e12"**),**

"dataId" **:** 139243**,**

"meterId" **:** 33**,**

"timestamp" **:** 1303132945000**,**

"countTotal" **:** 0.0**,**

"countRegister1" **:** 0.0**,**

"countRegister2" **:** 0.0**,**

"countRegister3" **:** 0.0**,**

"countRegister4" **:** 0.0**,**

"powerP1" **:** 95.0**,**

"powerP2" **:** 0.0**,**

"powerP3" **:** 0.0**,**

"workP1" **:** 0.0**,**

"workP2" **:** 0.0**,**

"workP3" **:** 0.0**,**

"frequency" **:** 0.0**,**

"voltage" **:** 0.0

**}**

Listing 8: JSON Struktur für ein Tupel von Messwerten

## Messdatenimport

Mit der Anweisung in Listing 9 werden die Daten formatgerecht importiert. Die importierte Datei steht im Projektverzeichnis zur Verfügung, beziehungsweise können weitere Daten mit entsprechenden Importprogrammen in die Collection geladen werden. Die zu importierende Datei muss im CSV-Format, in der ersten Zeile werden die Feldnamen erwartet, ein Beispiel ist in Listing 10 zu sehen.

mongoimport" --db meterData --collection meterTable --drop --file "getAdjusted33hergerichtetfürMongoDouble.csv" --type csv –headerline

Listing 9: Datenimport - mongoDB

dataId,meterId,timestamp,countTotal,countRegister1,countRegister2,countRegister3,countRegister4,powerP1,powerP2,powerP3,workP1,workP2,workP3,frequency,voltage

139240,33,1303132935000,0.0,0.0,0.0,0.0,0.0,95.67,0.0,0.0,0.0,0.0,0.0,0.0,0.0

139243,33,1303132945000,0.0,0.0,0.0,0.0,0.0,95,0.0,0.0,0.0,0.0,0.0,0.0,0.0

139244,33,1303132954000,0.0,0.0,0.0,0.0,0.0,96.33,0.0,0.0,0.0,0.0,0.0,0.0,0.0

Listing 10: CSV-Format der Rohdaten

## Zugriff über SmartValAPI

SmartValAPI stellt den Zugriff transparent über REST zur Verfügung, ein Beispielaufruf, wie in Listing 11 dargestellt ermittelt die vorhandenen Messdaten für die meterId (33) im Zeitraum von 2010-04-18 15:22:08 bis 2011-04-18 15:52:27. Das Ergebnis wird in Abbildung 14 gezeigt.

http://localhost:8080/query/moMeterDataquery?tspvon=2010-04-18%2015:22:08&tspbis=2011-04-18%2015:52:27&maxSamplefreq=18000&meterId=33

Listing 11: REST-Anforderung, um auf Daten zuzugreifen

In der ausgelieferten Version stehen die beide Arten des Zugriffs zur Verfügung, moMeterDataquery greift auf die MongoDB Daten zu, im Gegensatz dazu ermittelt adjustedmeterdatavectors die Messwerte über MySQL. Für einen produktionsähnlichen Betrieb wird man sich, abhängig von der zu verwaltenden Datenmenge für eine der beiden Ablagearten entscheiden, da die aufrufenden Applikationen sich nicht darum kümmern sollen, auf welcher Datenbank die Messwerte abgelegt wurden. Als Ergebnis stehen nach dem Aufruf jeweils Listen der selben Objekte (MeterDataEntity) zur Verfügung. Auf diese Listen können alle Operationen, die von der Klasse ResultSetParser zur Verfügung gestellt werden, angewandt werden. Zum Beispiel stehen Abgleich über gleiche Zeiträume gemacht werden, oder Messdatenreihen, welche in unterschiedlicher Abtastfrequenz vorliegen auf die gemeinsam höchstmögliche umzurechnen.

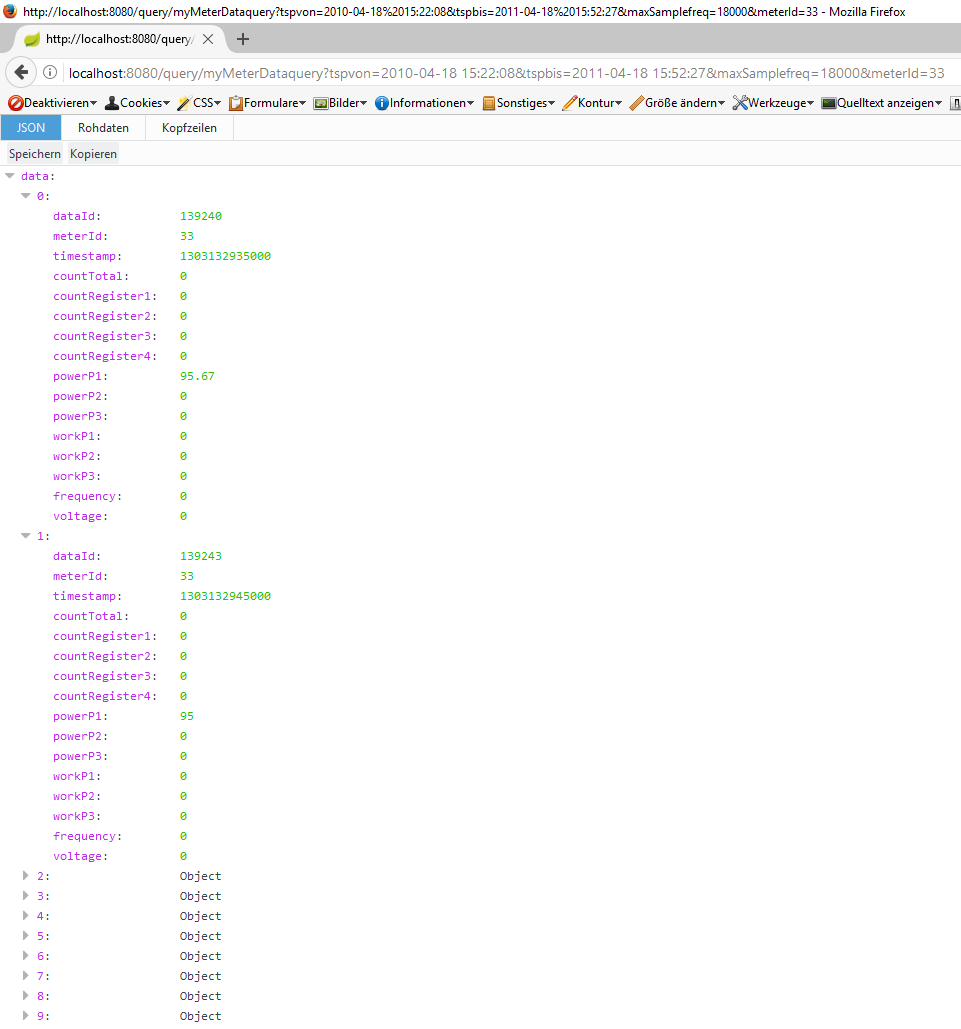
…

Abbildung 14: Ergebnis von moMeterDataquery

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | Das Europäische Parlament und der Rat der Europäischen Union, „Richtlinie 2009/72/EG des Europäischen Parlaments und des Rates vom 13. Juli 2009,“ 13. Juli 2009. [Online]. Available: http: //publications.europa.eu/resource/celex/32009R0713. [Zugriff am 19. Februar 2017]. |
| [2] | K. D. Craemer und G. Deconinck, „Analysis of state-of-the-art smart metering communication standards,“ in *Proceedings of the 5th young researchers symposium*, 2010. [Online]. Available: https://lirias.kuleuven.be/handle/123456789/265822. [Zugriff am 4. September 2017]. |
| [3] | J. Z. Kolter und J. Johnson, „REDD: A public data set for energy disaggregation research,“ in *Workshop on Data Mining Applications in Sustainability (SIGKDD)*, San Diego, CA, 2011. |
| [4] | W. K. Jack Kelly, „UK-DALE: A dataset recording UK Domestic Appliance-Level Electricity demand and whole-house demand,“ 2014. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.750.4515&rep=rep1&type=pdf. [Zugriff am 1. März 2017]. |
| [5] | A. Einfalt und e. al., „ADRES-Concept: Konzeptentwicklung für ADRES-Autonome Dezentrale Regenerative EnergieSysteme,“ TU Wien and Austiran Institute of Technology and Austian Power Grid, Wien, 2012. |
| [6] | Andreas Monacchi; et al., „GREEND: An energy consumption dataset of households in Italy and Austria,“ in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2014. |
| [7] | C. Bellucci, A.-M. Oberluggauer und M. Tschuchnig, „Untersuchung unterschiedlicher Referenzdatensätze im Energiebereich,“ 2017. |
| [8] | M. Egger, W. Ferlitz und T. Hanusch, „Rollenbasierter LDAP Zugriff,“ 2016. |
| [9] | *Elektrizitätswirtschafts- und -organisationsgesetz, Fassung vom 1. März 2017,* Wien, 2017. |
| [10] | Oesterreichs Energie, „Smart Metering Use-Cases,“ 14. Dezember 2015. [Online]. Available: http://oesterreichsenergie.at/branche/stromnetze/smart-meter-use-cases.html?file=files/oesterreichsenergie.at/Downloads%20Netze/Smart%20Meter/Oesterreich%20Use%20Cases%20Smart%20Metering\_14122015\_Version\_1-1.pdf. [Zugriff am 19. Februar 2017]. |
| [11] | Oesterreichs Energie, „Lastenheft Smart Meter,“ 1. Juli 2013. [Online]. Available: http://oesterreichsenergie.at/branche/stromnetze/lastenheft-smart-meter.html?file=files/oesterreichsenergie.at/Downloads%20Netze/Smart%20Meter/Lastenheft\_SmartMeter\_1\_0.pdf. [Zugriff am 19. Februar 2017]. |
| [12] | Das Europäische Parlament und der Rat der Europäischen Union, „Verordnung des Europäischen Parlaments und zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr,“ 11. Juni 2015. [Online]. Available: http://data.consilium.europa.eu/doc/document/ST-9565-2015-INIT/de/pdf. [Zugriff am 28. Februar 2017]. |
| [13] | „Intelligente Messgeräte-AnforderungsVO,“ 25. Oktober 2011. [Online]. Available: https://www.e-control.at/documents/20903/-/-/20a992e6-d11f-48b8-aef9-8e5d66f284c1. [Zugriff am 16. Februar 2017]. |
| [14] | „Datenformat- und VerbrauchsinformationsdarstellungsVO,“ 2012. [Online]. Available: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007999. [Zugriff am 14. Februar 2017]. |
| [15] | ETSI, „Open Smart Grid Protocol (OSGP),“ 1. Januar 2012. [Online]. Available: http://www.etsi.org/deliver/etsi\_gs/OSG/001\_099/001/01.01.01\_60/gs\_osg001v010101p.pdf. [Zugriff am 19. Februar 2017]. |
| [16] | Ferraiolo, David and Cugini, Janet and Kuhn, D Richard, „Role-based access control (RBAC): Features and motivations,“ in *Proceedings of 11th annual computer security application conference*, New Orleans, 1995. |
| [17] | „Gesamte Rechtsvorschrift für Datenschutzgesetz 2000,“ [Online]. Available: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10001597. |
| [18] | P. Unger, B. Moser und M. Wurz, „SmartVis-Dokumentation - Visualisierung von Smart Meter-Daten,“ 2016. |
| [19] | Bundesamt für Sicherheit in der Informationstechnik, „Freie Software (FLOSS: Freie, Libre und Open Source Software) - Strategische Position des BSI zu Freier Software,“ [Online]. Available: https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/FreieSoftware/freiesoftware\_node.html. [Zugriff am 1. März 2017]. |
| [20] | Oracle, „Guide to Scaling Web Databases with MySQL Cluster,“ [Online]. Available: https://www.mysql.de/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/. [Zugriff am 19. Februar 2017]. |
| [21] | J. S. Ward und A. Barker, „Undefined By Data: A Survey of Big Data Definitions,“ 2013. [Online]. Available: https://arxiv.org/pdf/1309.5821.pdf. [Zugriff am 4. September 2017]. |
| [22] | Salzburg AG, [Online]. Available: https://www.google.at/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwilh8z2psDSAhVMJJoKHfRyAJkQFggiMAE&url=https%3A%2F%2Fwww.salzburg-ag.at%2F%3FeID%3Ddownload%26uid%3D1825&usg=AFQjCNHc-oFEdGUSo3qC\_JXquXgB6QzpVg. [Zugriff am 2. März 2017]. |
| [23] | Oracle, „Guide to Scaling Web Databases with MySQL Cluster,“ 18. Oktober 2016. [Online]. Available: https://www.mysql.de/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/. [Zugriff am 19. Februar 2017]. |
| [24] | F. Fusco, U. Fischer, V. Lonij, P. Pompey, J.-B. Fiot, B. Chen, Y. Gkoufas und M. Sinn, „Data Management System for Energy Analytics and its Application to Forecasting,“ in *EDBT/ICDT Workshops*, Bordeaux, 2016. |
| [25] | P. Bonnet, J. Gehrke und P. Seshadri, „Towards sensor database systems,“ in *International Converence on Mobile Data Management*, Berlin, 2001. |

# Anhang

## SQL Messungen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Anzahl Datensätze Gesamt** | **Anzahl Datensätze Neu** | **Median Query** | **Mittelwert Dauer Query** | **Mittelwerte Einfügen** |
| 1.561.660 | 1.561.660 | 3.046 | 2.912 | 29.269 |
| 3.123.320 | 1.561.660 | 2.786 | 4.355 | 30.764 |
| 3.869.198 | 745.878 | 3.449 | 4.797 | 15.450 |
| 4.615.076 | 745.878 | 4.448 | 7.836 | 15.960 |
| 5.360.954 | 745.878 | 5.009 | 6.686 | 17.114 |
| 6.106.832 | 745.878 | 9.382 | 8.209 | 14.166 |
| 6.852.710 | 745.878 | 8.752 | 8.691 | 14.245 |
| 7.598.588 | 745.878 | 7.561 | 8.604 | 14.967 |
| 8.344.466 | 745.878 | 7.788 | 9.909 | 15.903 |
| 9.090.344 | 745.878 | 8.386 | 10.261 | 17.873 |
| 9.836.222 | 745.878 | 11.197 | 10.706 | 14.746 |
| 10.582.100 | 745.878 | 13.738 | 12.065 | 14.338 |
| 11.327.978 | 745.878 | 15.181 | 13.176 | 15.373 |
| 12.073.856 | 745.878 | 15.079 | 13.701 | 15.152 |
| 12.819.734 | 745.878 | 15.215 | 14.287 | 14.555 |
| 13.565.612 | 745.878 | 12.712 | 14.293 | 13.679 |
| 14.311.490 | 745.878 | 14.933 | 14.804 | 13.346 |
| 15.057.368 | 745.878 | 17.127 | 16.247 | 13.601 |
| 15.803.246 | 745.878 | 16.913 | 17.009 | 14.104 |
| 16.549.124 | 745.878 | 17.541 | 17.568 | 13.036 |
| 17.747.658 | 1.198.534 | 18.325 | 18.629 | 21.674 |
| 18.946.192 | 1.198.534 | 20.384 | 21.117 | 22.140 |
| 19.264.951 | 318.759 | 19.933 | 21.712 | 5.870 |
| 19.583.710 | 318.759 | 20.170 | 20.349 | 6.154 |
| 19.902.469 | 318.759 | 20.369 | 21.567 | 5.721 |
| 20.221.228 | 318.759 | 20.404 | 21.537 | 5.413 |
| 20.539.987 | 318.759 | 21.737 | 22.309 | 5.923 |
| 20.858.746 | 318.759 | 21.821 | 21.447 | 5.997 |
| 21.177.505 | 318.759 | 21.820 | 22.041 | 5.338 |
| 21.496.264 | 318.759 | 21.683 | 22.011 | 5.354 |
| 21.815.023 | 318.759 | 22.716 | 23.106 | 5.481 |
| 23.242.307 | 1.427.284 | 22.585 | 22.967 | 26.950 |
| 24.669.591 | 1.427.284 | 24.229 | 24.617 | 25.510 |
| 25.073.698 | 404.107 | 25.250 | 25.136 | 7.577 |
| 25.477.805 | 404.107 | 24.785 | 24.726 | 6.572 |
| 25.881.912 | 404.107 | 25.809 | 25.772 | 6.810 |
| 26.286.019 | 404.107 | 24.898 | 25.111 | 6.633 |
| 26.690.126 | 404.107 | 25.312 | 25.339 | 6.739 |
| 27.094.233 | 404.107 | 27.168 | 27.323 | 7.631 |
| 27.498.340 | 404.107 | 27.355 | 27.535 | 6.823 |
| 27.902.447 | 404.107 | 27.442 | 27.533 | 7.317 |
| 28.306.554 | 404.107 | 28.522 | 28.555 | 6.955 |
| 28.710.661 | 404.107 | 29.089 | 29.080 | 7.985 |
| 29.114.768 | 404.107 | 29.270 | 29.166 | 7.483 |
| 29.518.875 | 404.107 | 29.098 | 29.153 | 7.224 |
| 29.922.982 | 404.107 | 32.988 | 32.276 | 7.922 |
| 30.327.089 | 404.107 | 32.010 | 32.721 | 7.356 |
| 30.731.196 | 404.107 | 28.320 | 28.772 | 7.681 |
| 31.135.303 | 404.107 | 29.645 | 29.863 | 7.436 |
| 31.539.410 | 404.107 | 32.354 | 32.011 | 6.638 |
| 31.943.517 | 404.107 | 29.732 | 30.124 | 7.352 |
| 32.347.624 | 404.107 | 30.250 | 30.300 | 6.979 |
| 32.751.731 | 404.107 | 30.106 | 31.241 | 7.395 |
| 34.431.570 | 1.679.839 | 31.689 | 32.122 | 33.013 |
| 36.111.409 | 1.679.839 | 32.697 | 32.854 | 30.951 |
| 36.681.772 | 570.363 | 32.073 | 32.377 | 10.693 |
| 37.252.135 | 570.363 | 33.639 | 33.956 | 10.918 |
| 37.822.498 | 570.363 | 34.188 | 34.188 | 9.101 |
| 38.392.861 | 570.363 | 35.098 | 35.026 | 10.033 |
| 38.963.224 | 570.363 | 35.123 | 35.631 | 9.734 |
| 39.533.587 | 570.363 | 34.343 | 35.308 | 10.195 |
| 40.103.950 | 570.363 | 35.954 | 36.484 | 10.146 |
| 40.674.313 | 570.363 | 37.084 | 37.248 | 9.601 |
| 41.244.676 | 570.363 | 38.779 | 39.015 | 10.745 |
| 41.815.039 | 570.363 | 39.344 | 39.206 | 10.896 |
| 42.385.402 | 570.363 | 38.134 | 37.900 | 11.496 |
| 42.955.765 | 570.363 | 37.281 | 37.783 | 9.998 |
| 43.526.128 | 570.363 | 38.608 | 39.094 | 10.064 |
| 44.096.491 | 570.363 | 39.215 | 39.504 | 10.215 |
| 44.666.854 | 570.363 | 40.336 | 40.338 | 9.114 |
| 45.237.217 | 570.363 | 40.198 | 40.472 | 10.629 |
| 45.807.580 | 570.363 | 40.761 | 40.711 | 9.836 |
| 46.377.943 | 570.363 | 40.756 | 41.062 | 9.825 |
| 46.680.065 | 302.122 | 41.908 | 42.431 | 5.216 |
| 46.982.187 | 302.122 | 40.767 | 41.030 | 5.221 |
| 47.062.604 | 80.417 | 40.688 | 40.915 | 858 |
| 47.143.021 | 80.417 | 40.119 | 40.690 | 870 |
| 47.223.438 | 80.417 | 41.244 | 41.275 | 1.001 |
| 47.303.855 | 80.417 | 40.245 | 40.402 | 826 |
| 47.384.272 | 80.417 | 42.313 | 42.747 | 904 |
| 47.464.689 | 80.417 | 41.341 | 41.376 | 959 |
| 47.545.106 | 80.417 | 41.157 | 41.489 | 759 |
| 47.625.523 | 80.417 | 40.780 | 40.795 | 879 |
| 47.705.940 | 80.417 | 41.429 | 41.580 | 847 |
| 47.786.357 | 80.417 | 41.859 | 41.926 | 1.098 |
| 47.866.774 | 80.417 | 41.344 | 41.339 | 801 |
| 47.947.191 | 80.417 | 41.318 | 41.392 | 810 |
| 48.027.608 | 80.417 | 41.913 | 41.707 | 880 |
| 48.108.025 | 80.417 | 42.380 | 42.136 | 914 |
| 48.188.442 | 80.417 | 42.311 | 42.161 | 824 |
| 48.268.859 | 80.417 | 42.430 | 42.674 | 787 |
| 48.349.276 | 80.417 | 42.722 | 42.128 | 798 |
| 48.429.693 | 80.417 | 47.812 | 50.891 | 852 |
| 48.510.110 | 80.417 | 47.744 | 47.971 | 1.184 |
| 48.590.527 | 80.417 | 47.434 | 47.925 | 1.149 |
| 48.670.944 | 80.417 | 48.016 | 48.488 | 1.060 |
| 48.751.361 | 80.417 | 48.165 | 50.397 | 994 |
| 48.831.778 | 80.417 | 49.126 | 49.097 | 940 |
| 48.912.195 | 80.417 | 50.028 | 50.125 | 1.018 |
| 49.799.652 | 887.457 | 48.200 | 48.851 | 18.768 |
| 50.687.109 | 887.457 | 48.321 | 48.778 | 18.176 |
| 51.064.077 | 376.968 | 48.709 | 52.024 | 7.211 |
| 51.441.045 | 376.968 | 50.506 | 51.418 | 7.506 |
| 51.818.013 | 376.968 | 54.101 | 55.036 | 8.319 |
| 52.194.981 | 376.968 | 50.304 | 50.361 | 7.627 |
| 52.571.949 | 376.968 | 52.504 | 53.179 | 7.436 |
| 52.948.917 | 376.968 | 52.966 | 53.548 | 7.142 |
| 53.325.885 | 376.968 | 53.259 | 54.507 | 7.273 |
| 53.702.853 | 376.968 | 56.174 | 56.052 | 7.514 |
| 54.079.821 | 376.968 | 56.656 | 56.808 | 7.516 |
| 54.456.789 | 376.968 | 54.351 | 54.845 | 7.066 |
| 54.833.757 | 376.968 | 56.484 | 57.023 | 7.532 |
| 55.210.725 | 376.968 | 53.445 | 55.320 | 6.812 |
| 55.587.693 | 376.968 | 58.440 | 60.198 | 7.319 |
| 55.964.661 | 376.968 | 55.223 | 55.174 | 7.117 |
| 56.341.629 | 376.968 | 56.523 | 56.217 | 7.584 |

1. http://hibernate.org/ [↑](#footnote-ref-1)
2. https://swagger.io/swagger-ui/ [↑](#footnote-ref-2)
3. http://hibernate.org/ [↑](#footnote-ref-3)
4. SmartValAPI\app\src\main\resources\application.properties [↑](#footnote-ref-4)