# SmartValAPI: Systemanforderungen.

SmartValAPI ist in Java umgesetzt und stellt daher keine speziellen Anforderungen an das System. Im Folgenden sind jene Softwarekomponenten, die für die Ausführung notwendig sind, aufgezählt. Die Aufzählung enthält all jene Programme, unter denen die Installation und Ausführung getestet wurde. Alternativen sind möglich, aber nicht getestet.

Betriebssystem: Windows Server 2012 R2, alternativ Windows 10 Home

Datenbank: MySQL Server 5.7

Java Umgebung: Java(TM) SE Runtime Environment (build 1.8.0\_141-b15)

Netzwerkzugang, sofern nicht nur lokal ausgewertet wird

Nur zum Testen: SOAPUI 5.3.0

Datenbank und Applikation können zur Lastverteilung auf 2 Rechner aufgeteilt werden. Netzwerk und Firewall-Konfiguration müssen eine Verbindung ermöglichen beziehungsweise zulassen.

# Einrichten der Datenbank und der Tabellen.

Mit welcher Datenbank SmartValAPI spricht, ist in der Konfigurationsdatei application.properties festgelegt, ausgeliefert wird die Software mit diese Einstellungen:

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

Benutzername: uusseerr, Passwort: 1passwort2 (dem Benutzer sind die Berechtigungen INSERT, UPDATE, SELECT und DELETE einzuräumen, die Anlage ist gesondert durchzuführen).

Datenbankname: smart\_meter

Um diese Datenbank und die Tabellen einzurichten, sind zwei Schritte notwendig:

* 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

Um die bereitgestellten Tests durchführen zu können, sind jene Daten in die Tabellen zu importieren, die mit DB\_und\_Test\_Skripts/importTestData.sql entladen wurden.

# Tests

Jene 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, Tests durchführen. Abbildung 1 zeigt teilweise das Ergebnis der Abfrage dreier meter\_ids über einen gemeinsamen Zeitraum.

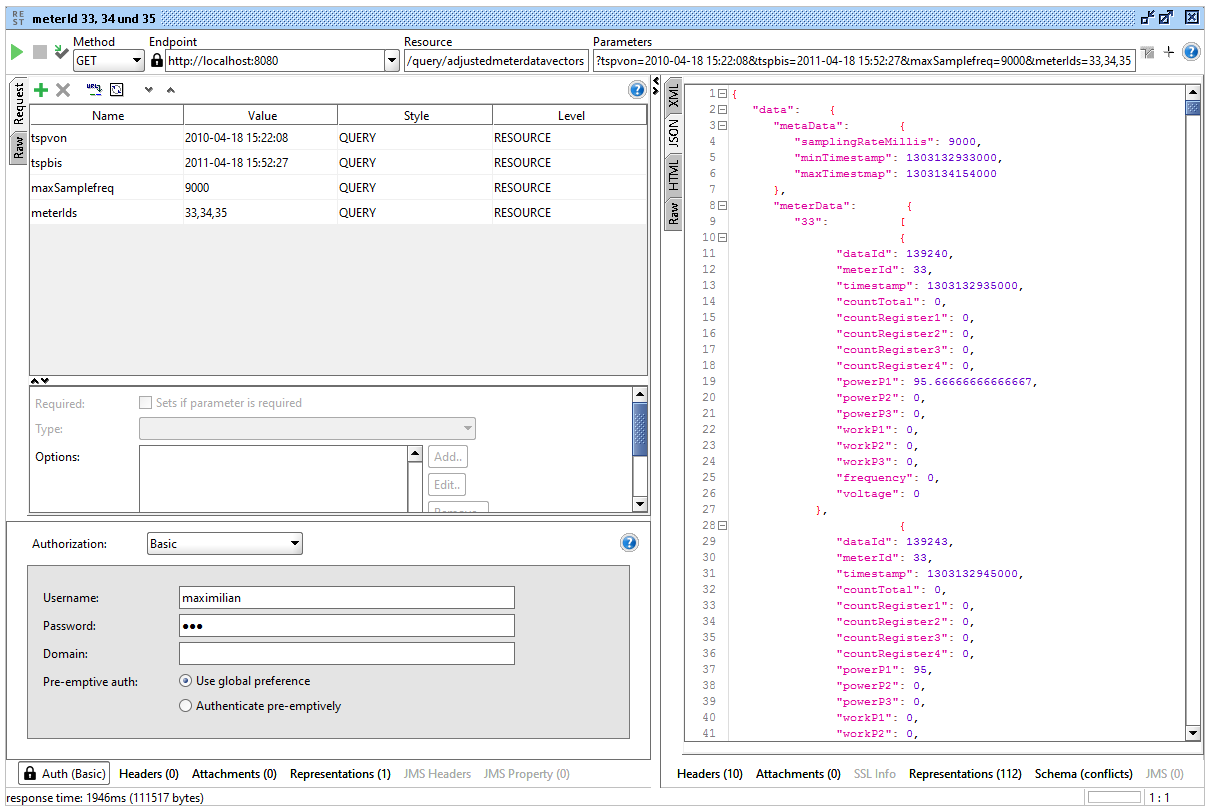


Abbildung : Testfall I-11-A mit drei Messdatenreihen in SOAP UI

Todo: hier noch die Startbarkeit der Tests mit SOAP Ui erwähnen.

## 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.

### 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.

### 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 Zeipunkt“ 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 Zugriffsspfad ist in Abbildung 2, links, dargestellt. Listing 1 zeigt eine Abfrage wie sie zum Beisiel 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 : 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 : Abfrage mehrerer Messdatenreihen

Abhilfe schafft hier die Erstellung eines Index, das Statement ist in Listing 2 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 : Create Index Statement