-------------------------------------------------------------------------------------------------------------------- alles nach dieser Linie, bis zum Anhang gehört in die obere Struktur gepackt. ---------------------------------------------------------------------------------------------------------------------

# Recherche

Die folgenden Recherchen sind notwendiger Teil des Projektes für die Informationsbeschaffung sowie der Prävention von eventuell doppelt gemachter Arbeit.

## ER-Modell

Um ein geeignetes Datenbankmodell zur Verfügung stellen zu können, müssen vorhandene Datenbankmodelle genauer betrachtet werden.

Das Datenmodell der JRZ-DB wird analysiert und auf Erweiterungsnotwendigkeiten untersucht.

## Analyse von Datenbankanforderungen

Es gilt herauszufinden, welches Datenbankmodell (DBM) für die Umsetzung des Projektes ideal ist. Um ein geeignetes DBM zu finden, wurden verschiedene Typen wie RDBMS, NoSQL und Hadoop genauer betrachtet. Weitere Datenbankmodelle ansehen und auflisten. Für die Ablage und den Zugriff auf Messdaten bietet jede dieser Architekturen Vorteile, die gegeneinander abgewogen werden. RDBMS stellen mit SQL eine Programmiersprache der vierten Generation zur Abfrage zur Verfügung, NoSQL Datenbanken, wie zum Beispiel MongoDB[nicht weil die flexibilität auf Kosten der Kompatibilität geht], sind flexibel bei der Erweiterung um zusätzliche Messwertarten und das Hadoop Rahmenwerk verwaltet Daten im Bereich der zu erwartenden Messdatenmengen.

Für die Analyse wird folgendermaßen Vorgegangen:

* Analyse des bestehenden Datenbankmodells (JRZ-DB)
* Performanceanalyse der zu testenden DBM sowie
* Einlesen in relevante Produkte

*Beim Meeting mit der Projektbetreuung am 27.1.2017 wurde der Hinweis gegeben, dass Hadoop für dieses Projekt interessant sein könnte und wir uns das näher ansehen sollten. Zudem wurde erwähnt, dass es von Hortonworks eine Sandbox gibt, auf der ein fertig konfiguriertes Hadoop System mit unterschiedlichsten Tools verfügbar ist.*

*Ein großer Teil dieses Arbeitspaketes bestand darin, sich in Hadoop einzuarbeiten, Tutorials durchzumachen und erste Erfahrungen mit Big Data Systemen zu machen.*

*Bei den Gesprächen mit der Projektbetreuung hat sich allerdings auch herausgestellt, dass ein Weiterverwenden der JRZ-DB wünschenswert ist, da es bereits einiges an Software dafür gibt.*

Des Weiteren spricht dafür, dass es eine BAC1 Gruppe gibt, welche sich mit dem Datenimport von frei verfügbaren Smartmeter Datensätzen beschäftigt hat. Der aktuelle Stand dieser Arbeit ist nach Aussage der Gruppe soweit, dass Daten normiert importiert werden können, die Performance allerdings noch nicht optimal ist. *Das ist zum aktuellen Zeitpunkt noch nicht verifiziert, allerdings gehen wir davon aus, dass dieses Projekt nach Performanceoptimierung einsatzbereit ist. Da die verwendetete Programmiersprache allerdings C# ist, ist auf jeden Fall eine Portierung notwendig, da wir uns für das Projekt auf Java geeinigt haben.*

### Analyse JRZ-DB

* Auf Grund der Analyse des ER-Modells der JRZ-DB können die Tabellen in zwei Gruppen eingeteilt werden: Stammdaten, wie zum Beispiel customer\_data oder meter\_management, deren Anzahl verwalteter Datensätze von einem RDBMS ohne Performanceeinbußen verwaltet werden kann. Zum Beispiel verwaltet der Projektpartner Salzburg AG ca. 500.000 intelligente Zähler [Referenz: Salzburg AG Geschäftsbericht 2015 https://www.google.at/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwioxJnvzK7SAhXiCJoKHfubAM0QFggcMAA&url=https%3A%2F%2Fwww.salzburg-ag.at%2F%3FeID%3Ddownload%26uid%3D1825&usg=AFQjCNHc-oFEdGUSo3qC\_JXquXgB6QzpVg&cad=rja]. Andererseits werden in der Tabelle ‚meter\_data‘ die Bewegungsdaten abgelegt, hier sind durch die viertelstündliche Auslesung von jedem dieser Smart Meter täglich 96 Tupel abzulegen. Hochgerechnet auf die ca. 500.000 Smart Meter, welche durch die Salzburg AG versorgt werden, ergeben sich pro Monat 1,44 Milliarden Sätze. Das Datenmodell der JRZ-DB gilt als zu erfüllende Vorgabe, daher wird dieses in der Grundform eingesetzt, bezüglich der Abbildung der performancekritischen Tabelle meter\_data erfolgen weitere Untersuchungen mit dem Hadoop Rahmenwerk.

### Performanceanalyse

Für erste Tests wurde ein Teil der REDD ‚low\_freq‘ Daten verwendet. Diese wurde in verschiedene Datenbanken importiert und es wurden darauf Abfragen ausgeführt. Ziel dieser Analyse war es ein Gefühl zu bekommen, wie sich die Performance mit Zunahme an Daten verhält und ob eine SQL Datenbank überhaupt in Frage kommen kann.

## Analyse vorhandener Software

* Da das Projekt ein Teilprojekt eines Größeren ist, galt es festzustellen, ob schon bestehende Software für das Projekt verwendet werden kann. Somit wird redundante Arbeit vermieden und der Fokus auf die noch benötigen Dinge gesetzt.

## Rollendefinition

* Mit SmartValAPI wird der Zugriff auf sensible Daten verwaltet, daher ist die Schutz des Zugriffs unerlässlich. Auf Aspekte der Datensicherheit wie physischer Zugang zum Datenbank beziehungsweise Applikationsserver geht der Abschnitt „Installation“ näher ein, dieser Abschnitt beleuchtet den Zugriff über Rollen und legt die Rollendefinition fest.
* Die Kernaufgaben sind wie folgt:
* Rollen identifizieren und definieren.
* Unterschiede in fachlichen und technischen Anforderungen in notwendigen Rollen überleiten.
* Verbindung zum Code, beziehungsweise Funktionen herstellen.
* Rechtliche Umgebung einbeziehen.

# Umsetzung

In dieser Sektion wird beschrieben, wie bisher die Pakete Systemarchitektur, ER-Modell und die Rollendefinition umgesetzt wird.

## Systemarchitektur

* Um die Systemumgebung festzulegen, und vor allem die Software passgenau in die Softwarelandschaft des JRZ einfügen zu können, werden die bestehenden System wie zum Beispiel Smart Viz, die in Entstehung befindlichen Zugriffsysteme (BAC1 Gruppe – OpenTC) und Importprogramme (BAC1 Gruppe Oberluggauer und Co). Was war mit Open-Nes? In die Überlegungen werden weitrs etwaige Kosten für Lizenzen und andererseits Sicherheitsaspekte einbezogen [18]. Funktionen nach Aufruf und Zusammengehörigkeit gruppieren.

## ER-Modell

~~Beim Festschreiben des geeigneten Datenbankmodells wurde wie unterhalb beschrieben vorgegangen:~~

* ~~Recherche nach Datenmodellen, die bereits abseits der JRZ-DB im Einsatz sind, wie zum Beispiel COSEM.~~
* ~~Feststellen der Wertemenge die SmartMeter zur Verfügung stellen und herausarbeiten welche davon gespeichert werden.~~
* ~~Kontaktaufnahme mit österreichischen Energieversorgern bezüglich der Werte von Interesse.~~
* ~~Analyse der Usecases von Österreichs Energie, ergibt vor allem Daten bezüglich der Steuerung des Smart Meter als solches, und nicht der inhaltlichen Bedeutung der übertragenen Messdaten.~~
* ~~Erhebung der rechtlichen Rahmenbedingungen und sich daraus ergebende Einschränkungen.~~
* ~~Prüfung auf Verwendbarkeit des Datenmodells in weiteren Domänen wie zum Beispiel: Gas, Wärme und Wasser.~~
* ~~Festschreiben des Datenmodells.~~

## Analyse der Datenmodelle

### Hadoop Tests

Für die Tests wurde eine virtuelle Maschine mit der ‚Hortonworks Hadoop Sandbox‘ aufgesetzt. Der Maschine wurden alle Cores des Hosts[[1]](#footnote-1) sowie 8GB Arbeitsspeicher zur Verfügung gestellt.

### MySQL Tests

Für die Tests wurde ein MySQL Server auf dem o.g. Testsystem aufgesetzt. Im Gegensatz zu den Hadoop Tests allerdings direkt auf dem Host-Betriebssystem.

### Testdaten

Als Testdaten wurden REDD Daten eines Hauses importiert. Die Abfrage, die darauf abgesetzt wurde, ergibt den Durchschnittsverbrauch pro Tag pro Kanal.

Die Daten liegen in CSV Dateien mit folgendem Format vor.

|  |  |
| --- | --- |
| timestamp | power |
| int | double |

Tabelle 1: Datenformat CSV

Der Timestamp lässt sich mit FROM\_UNIXTIME(timestamp) in ein Datum umwandeln womit gerechnet werden kann.

Folgende Abfrage liefert den Durchschnittsverbrauch gruppiert nach Kanal, Monat und Tag. Die Dauer ist unser Performance Index.



Abbildung 7: Abfrage Durchschnitt pro Tag

Es wurden Schritt für Schritt Kanäle hinzugefügt um einen wachsenden Datensatz zu simulieren.

# Ergebnisse

Bisherige Ergebnisse in Bezug Systemarchitektur, ER- Modell, Analyse vorhandener Datensätze sowie vorhandener Software, Recherche noch benötigter Software beziehungsweise Funktion und momentaner Rollendefinition werden nun vorgestellt.

## Systemarchitektur

* Hardware: für den Betrieb ist keine explizite Hardware vonnöten, vom JRZ wurde eine virtuelle Maschine im Bladecenter zur Verfügung gestellt.
* Betriebssystem: auf Grund der größeren Erfahrung der Entwickler mit der Administration erfolgt die Implementierung auf einem Windows System (Windows Server 2012 R2),
* Programmiersprache: bestehende Software wurde im Umfeld des JRZ in Java implementiert, ebenso werden laufenden Projekte in Java programmiert, die Erfahrung der Umsetzenden reicht aus, um die Anforderungen umzusetzen. Im Sinne einer einfachen Übergabe, Weiterführung und Wartung fällt die Entscheidung, dieses Projekt in Java zu implementieren.
* Datenbank: neben unterschiedlichen RDBMS (Oracle Database Server, MySQL) wurden NoSQL Datenbankensysteme untersucht (MongoDB, Hadoop). Im Sinne der Integration in die bestehende Softwarelandschaft wird MySQL eingesetzt. Ziel des Projektes ist eine Integrationsdatenbank, was dazu führt, dass sehr große Datenmengen verwaltet werden können müssen, und der Zugriff in akzeptabler Zeit erfolgt. MySQL unterstützt lokale (partitioning) und verteilte (sharding) Fragmentierung. Nach dem „Guide to Scaling Web Databases with MySQL Cluster“ [19] 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, über die Administration des MySQL Clusters wird das Partitionierung transparent für die Anwendung durchgeführt. Details dazu sind im Abschnitt „Datenbankzugriff“ erläutert.
* Datenbankdesigntool: untersucht wurden Oracle SQL Developer Data Modeler 4.1.5 und die MySQL Workbench 6.3.9. Für den graphischen Entwurf des Entity-Relation-Modells bieten beide Tools Unterstützung. Modeler wie Workbench generieren aus dem ER-Modell sowohl die graphische Übersicht der Tabellen und Schlüssel, als auch Generation der Skriptdateien zur Anlage der Tabellen, Indizes, Einschränkungen bezüglich referentieller Integrität und die automatische Vergabe eindeutiger Schlüssel. Da als Datenbanksystem MySQL eingesetzt wird, liegt es nahe das Designtool vom gleichen Hersteller einzusetzen und damit Kompatibilitätsprobleme beziehungsweise Nachbearbeitungen zu vermeiden.
* Schnittstellen zu BAC1 (Autorisierung): der bisherige Entwurf des API ist in einer generischen Form implementiert, sodass einerseits ein Austausch des Autorisierungsmoduls möglich ist (zum Beispiel in einem ersten Schritt realisiert als Benutzername/Passwort Zugang). Eine Abhängigkeit zum Datenmodell wurde identifiziert: die Form der Benutzerkennung des externen Rollensystems muss in den Tabellen untergebracht werden könne (Details siehe Abschnitt Datenmodell).
* Schnittstellen zu BAC1 (Importmodule): gemeinsame Basis für die API und Importmodule ist das Datenmodell (Details zu notwendigen Erweiterungen siehe im Abschnitt Datenmodell), programmtechnisch wurden keine Überschneidungen identifiziert.
* Die Anbindung und die Veröffentlichung der Schnittstellen: <wir haben noch nicht festgelegt wie das API aufgerufen werden kann> Webservice? Datenstrukturen?
* Aus den Anforderungen ergeben sich folgende Komponenten, deren Abhängigkeiten untereinander und Verbindungen zueinander, die im UML Diagramm (Abbildung 1) dargestellt werden:
* 

Abbildung 8: Komponentenmodell SmartValAPI

Die Details zu den Komponenten und die exportierten Methoden werden im Abschnitt „API Funktionen“ beschrieben. Der Übersicht halber wird die Komponente „Logger“, die jede der angeführten Komponenten verwendet, nicht in der Übersicht dargestellt.

## ER Modell

Hier werden die Ergebnisse des zu verwendenden Datenbankmodells gezeigt.

### Analyse bestehendes Datenmodell

### Anforderungen von Energieversorgern und Netzbetreibern

* Anfragen an 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 ergeben:
* Verweis auf das ElWOG: die erheb baren Daten sind klar definiert eine darüberhinausgehende Erfassung von Messwerten ist nicht zulässig.
* Verweis auf die laufende Ausschreibung der Smart Meter Geräte, hier werden die Messwerte in Pflicht- und optionale Werte eingeteilt.
* Die im Datenmodell vorhandene Netzfrequenz ist auf Grund der Rückmeldungen nicht auswertungsrelevant auf Basis einzelner Smart Meter nicht interessant.
* Keine Rückmeldung, beziehungsweise nur telefonisch, wobei sich aus letzterer keine weiteren Datenfelder ergeben.
* Hervorzuheben sind die Antworten von Salzburg Netz GmbH und Vorarlberger Energienetze GmbH, bezüglich der Messwerte steht für den Netzbetreiber, neben den bereits vorhandenen Werten stehen der Winkel zwischen Spannung und Strom, und die daraus resultierende Blindleistung im Fokus.

Im Gegenzug dazu kann auf die Frequenz verzichtet werden, dieses Datenfeld verbleibt auf Kompatibilitätsgründen im ERM.

### Analyse der Usecases

### Weitere Datenmodelle

* Fusco et al. [20] schlagen 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 Abschnitt Systemarchitektur).
* Ein dezentraler Ansatz, wie zum Beispiel das COUGAR Sensornetzwerk [Referenz Cougar], als Alternative zu einer zentralen Datenbank bietet zwar den Vorteil, einen zentralen Angriffspunkt zu vermeiden, Messdaten hingegen ausschließlich ad hoc auszulesen widerspricht den Regelungen des ElWOG und scheidet damit aus.

### Rechtliche Rahmenbedingungen und daraus resultierende Einschränkungen der möglichen Messwerte

* Die Erfassung, Übertragung und Speicherung von Smart Meter Messdaten wird in vier Richtlinien geregelt:
* ElWOG
* GDPR
* IMA-VO
* DAVID-VO

Geregelt werden einerseits Mindestanforderungen an Smart Meter, andererseits die Inhalte und die Frequenzen, mit denen die Werte ausgelesen werden dürfen. Im ElWOG werden in §84 dem Verbraucher die Daten bezüglich des „Verbrauchs der über ein intelligentes Messgerät gemessen wird“ zeitnah zur Verfügung zu stellen. Es erfolgt keine genauere Definition, welche Daten das im Detail sind, lediglich die Frequenzen, mit denen ausgelesen wird, werden festgelegt, Details dazu, siehe Abschnitt Rollendefinitionen.

Einer der Hauptgründe für die Reglementierung ist der Schutz der Privatsphäre [21]. Bezüglich der auslesbaren Daten legt die [11] jene Daten fest, die von einem Smartmeter übertragen werden müssen. Die IMA-VO und DAVID-VO befassen sich mit der Einführung der der intelligenten Zähler und der Weitergabe der ausgelesenen Daten und haben keinen Einfluss auf das Datenmodell.

### Das COSEM Modell

[2] versucht hier einen Standard zu etablieren. Nachdem die Salzburg AG als Partner des Projektes in den Ausschreibungen für Smart Meter die Kommunikation über DLSM/COSEM als Muss-Kriterium festlegt, wird in der Folge dieses Protokoll in Verbindung mit den Anforderungen des Lastenheftes von Österreichs Energie [11] als Ausgangspunkt für die Festlegung der Datenbank herangezogen.

### ER-Modell Festlegung

Ausgangsbasis ist die JRZ-DB, bei der Erweiterung wurde auf die Kompatibilität zu bestehenden Programmen geachtet, um diese Applikationen ohne Anpassung auch weiterhin gegen die gleiche Datenbank betreiben zu können. Folgendes Diagramm gibt einen Überblick über die Entitäten:



Abbildung 9: ER-Modell Entitäten

Aus dem existierenden Modell werden die bestehenden Entitäten übernommen, für die zusätzlichen Anforderungen ergeben sich folgende Erweiterungen:

Zusätzliche Entitäten:

* userAdmin: die Benutzerverwaltung wird als eigene Komponente eingebunden, um die Kopplung lose ausführen zu können wird im System lediglich der Schlüssel (der Benutzername, eine LDAP ID, ein etwaiges anders identifizierendes Merkmal) hinterlegt.

Zusätzliche Beziehungen:

* istSystembenutzer (userAdmin – customer): optionale Erweiterung zum customer, ermöglicht die Ablage einer externen Benutzerkennung.
* berät\_oder\_verwaltet (customer – meter\_management): verbindet m customer mit n meter\_management, Zweck ist die Abbildung von Verbinungen wie zu, Beispiel: Netzbetreiber versorgt Meter, Energieberater berät Eigentümer von Meter.
* ist\_Kunde\_von: ermöglicht die hierachische Verbindung von Kunden, zum Beispiel Energeversorger mit Kunden.

Zusätzliche Attribute:

* in meter\_data: reactive\_P1, reactive\_P2, reative\_P3: Blindleistungsanteil aufgeteilt nach Phase, sofern nur gesamt übermittelt in P1, wenn nicht ausgelesen: 0.
* phi\_P1, phi\_P2, phi\_P3: Phasenwinkel von Strom und Spannung, je Phase, wenn ausgelesen,

Aus den Typen der Beziehungen ergeben sich folgende Tabellen:



Abbildung 10: Datenbankbeziehungen

Alternative für meter\_data: da die Tabelle das maximale Set an Daten abbilden kann entstehen eventuell einige Tupel mit Null-Werten, sofern Smart Meter nicht alle Felder auslesen können. Die Alternative besteht in der Definition einer Tabelle die je Tupel einen Messwerttyp (bedingt eine weitere Definitionstabelle für die zugelassenen Werttypen) und einen Messwert.

Der Messwerttyp ist Teil des primären Schlüssels. Die Vorteile der Vermeidung von Nullwerten und der flexiblen Einführung neuer Werttypen, zum Beispiel auch Gas, Wasser oder Wärme, werden durch die Vervielfachung der Einträge einerseits und des Verlustes der Kompatibilität mit den bestehenden Softwarepaketen erkauft.

Formate der Datenfelder:

Aus der Tabellendefinition können die Formate der jeweiligen Felder ermittelt werden, inhaltlich wird festgelegt, dass in **meter\_management:**  meterId entsprechend der OBIS Identifikation befüllt wird (Details siehe Abschnitt „API Funktionen“).

## Analyse möglicher Datensätze

Es haben sich bei den Tests folgende Ergebnisse errechnen lassen:

### MySQL Datenbank

Zuerst wird die Dauer des Datenimports gemessen.

|  |  |  |
| --- | --- | --- |
| **Import Dauer** | | |
| **Vorhanden** | **Hinzugefügt** | **Dauer [s]** |
| - | 1.561.660 | 27 |
| 1.561.660 | 1.561.660 | 28 |
| 3.123.320 | 745.878 | 10 |
| 3.869.198 | 745.878 | 11 |
| 4.615.076 | 745.878 | 10 |
| 5.360.954 | 3.729.390 | 62 |
| 9.090.344 |  |  |

Tabelle 2: Dauer des Datenimports

Abbildung 11: Dauer des Datenimports

Hierbei ist zu sehen, dass die Dauer unabhängig von den bereits vorhandenen Datensätzen linear abhängig zu der Menge der Importierten Datensätze ist.

Als nächsten Schritt wird die Dauer des Query aus *Abbildung 1: Abfrage Durchschnitt pro Tag*gemessen.

|  |  |
| --- | --- |
| **Berechnung des Durchschnitts** | |
| **Anzahl Zeilen** | **Dauer** |
| 1.561.660 | 6 |
| 3.123.320 | 16 |
| 3.869.198 | 19 |
| 4.615.076 | 23 |
| 5.360.954 | 27 |
| 9.090.344 | 42 |

Tabelle 3: Dauer Berechnung des Durchschnittsverbrauchs

Abbildung 12: Dauer Berechnung des Durchschnittsverbrauchs

Abbildung 6 zeigt, dass die auch die Dauer der Abfrage linear abhängig zur Anzahl der vorhandenen Datensätze ist.

Abschließend wurden Tag und Monat aus dem Timestamp fix in die Tabelle geschrieben und mit einem Index versehen. Dadurch konnte die Berechnungsdauer bei ~ 10 Millionen Datensätzen von 42 Sekunden auf 8 reduziert werden.

### Hadoop Datenbank

Erste Tests haben gezeigt, dass die Abfragen mit HIVE annähernd gleich funktionieren wie mit SQL, die Performance allerdings auf dem Testsystem schlechter ist. Aufgrund der zufriedenstellenden Ergebnisse mit SQL wurde auf ebenso detaillierte Tests verzichtet. Die Performance lässt sich durch Hinzufügen von mehreren Nodes nahezu beliebig steigern.

### Zusammenfassung der Datenbankanalyse

* Beim Import der Datensätze fällt auf, dass die Anzahl der bereits vorhandenen Daten keine Auswirkung auf die Dauer des Vorgangs hat, was uns entgegenkommt.
* Die Dauer einer Berechnung des Durchschnitts nimmt proportional mit der Anzahl der Datensätze zu. Der REDD Datensatz mit niedriger Auflösung beinhaltet ungefähr 52 Millionen Datensätze.
* Hier würde die Abfrage hochgerechnet 210 Sekunden dauern. Es hat sich aber gezeigt, dass sich durch die geschickte Verwendung von Indizes, die Performance des getesteten MySQL System sehr stark steigern lässt. Aus diesem Grund, und auch um die Kompatibilität zu den bestehenden Applikationen zu wahren, wird für dieses Projekt bis auf weiteres ein Relationales Datenbanksystem verwendet.
* Falls sich im weiteren Verlauf des Projekts herausstellen sollte, dass eine SQL Datenbank nicht ausreichend ist, kann eine Hybrid Lösung angestrebt werden.
* Hier wäre vorstellbar, dass die ganze Meterverwaltung weiterhin in einer SQL Datenbank verbleibt, die Messdaten allerdings in Hadoop abgespeichert werden.

## Analyse vorhandener Software

Für die Analyse bereits vorhandener Software wurde folgendermaßen vorgegangen:

* Abfragen der einzelnen Gruppen, welche Projekte verfolgt werden und welche Dokumente oder Projekte denen zur Verfügung stehen.
* Einschätzen, ob eines dieser Programme oder Projekte verwendet werden kann.
* Einholung der Dokumentation und des Projektes bei Interesse an einem Projekt.
* Abfragen der einzelnen Gruppen, welche Projekte verfolgt werden und welche Dokumente oder Projekte denen zur Verfügung stehen.
  + Eine BAC Gruppe hat ein vielversprechendes Projekt (OpenTC). Hierbei handelt es sich um eine Software, die Rollenbasiertes Zugreifen auf Dokumente zulässt.
  + Eine weitere Gruppe wurde ausgeforscht, welche interessante Dokumentation hat. (BAC 1 Gruppe Oberluggauer und Co)
* Einschätzen, ob eines dieser Programme oder Projekte verwendet werden kann:
  + Das BAC Projekt OpenTC könnte für unseren rollenbasierten Zugriff auf die Datenbank umgeschrieben werden.
  + Informationen der weiteren Gruppe (BAC1 Gruppe Oberluggauer und Co).
* Einholung der Dokumentation und des Projektes bei Interesse an einem Projekt:
  + Die Dokumentation sowie das ganze Projekt konnte organisiert werden für Recherche.
  + Die Informationen der weiteren Gruppe. (BAC1 Gruppe Oberluggauer und Co)

## Rollendefinition

* Folgende Rollen wurden identifiziert und mit den im Use-Case-Diagramm angeführten Anwendungsfällen verbunden, die in Klammern angeführten Attribute geben die grundlegende Ansiedlung der Personen, die die Rollen bekleiden.
* Administrator (technisch): Stammdatenwartung (Meter Anlage und abhängige Daten),
* Energieversorger (fachlich): jener, der den Verbraucher mit Energie versorgt.
* Energieberater (fachlich): über die ausdrückliche Zustimmung des Eigentümers befugte Person.
* Eigentümer (fachlich): derjenige dessen Verbrauch mit dem Smartmeter gemessen wird.
* Importeur (technisch): Meterdaten in die Datenbank importieren, das Smart Meter muss bereits im System bestehen.
* 

Abbildung 13: Rollendefinition - UseCase Diagramm

# Weitere Schritte

1. Intel Core i5-4690K @ 3.5Ghz [↑](#footnote-ref-1)