HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Luware
SWISS AND SIMPLE

Bachelorarbeit, Abteilung Informatik

Analytics im Kundendienstumfeld

Hochschule für Technik Rapperswil

Frühlingssemester 2016
17. Juni 2016

Autoren:	Sandro Muggli & Fabio Laib
Matrikelnummern:	12-154-365, 12-154-472
Betreuer:	Prof. Dr. Beat Stettler
Projektpartner:	Michael Jakob, Luware AG
Arbeitsperiode:	22.02.2016 - 17.06.2012
Arbeitsumfang:	360 Stunden, 12 ECTS pro Student

Inhaltsverzeichnis

1 Management Summary **6**
- 1.1 Luware AG . . . 6
- 1.2 Ausgangslage . . . 6
- 1.3 Problemstellung . . . 6
- 1.4 Erwartetes Resultat . . . 7
- 1.5 Methode und Vorgehen . . . 7

2 Analyse **8**
- 2.1 Produktevaluation . . . 8
  - 2.1.1 Bewertungsraster . . . 8
  - 2.1.2 Solr vs. Elasticsearch . . . 10
  - 2.1.3 Bewertung von Solr . . . 12
  - 2.1.4 Bewertung von Elasticsearch . . . 13
  - 2.1.5 Auswertung und Vergleich . . . 14
- 2.2 Vertikales Prototyping (Durchstich) . . . 14
  - 2.2.1 Datenimport . . . 14
  - 2.2.2 Datenspeicherung . . . 15
  - 2.2.3 Datenvisualisierung . . . 15
  - 2.2.4 Benutzerinteraktion und Anzeige . . . 16
  - 2.2.5 Erkenntnisse . . . 16
- 2.3 Anforderungsanalyse . . . 16
  - 2.3.1 Funktionale Anforderungen . . . 17
  - 2.3.2 Use Cases . . . 17
  - 2.3.3 Aktuelle Reports . . . 19
  - 2.3.4 Nichtfunktionale Anforderungen . . . 34

3 Design **35**
- 3.1 Elasticsearch Datenbank . . . 35
  - 3.1.1 Reporting Schema Luware . . . 35
  - 3.1.2 Abbildung in Elasticsearch . . . 37
  - 3.1.3 Import Datenbestand via . . . 38
- 3.2 Application Architecture . . . 38
  - 3.2.1 Architektur Layer . . . 39

4 Realisierung **40**
4.1 Reporting heute und in naher Zukunft 40
4.1.1 Reporting SSRS (Heute) 40
4.1.2 Reporting Portal (Zukunft) 41
4.2 Technische Umsetzung 42
4.2.1 Anforderungen 42
4.2.2 Herausforderungen 42
4.2.3 Umsetzung Reports 42
4.3 Ordnerstruktur Applikation 43
4.4 Umsetzung eines Reports 43
4.4.1 Analyse des SSRS Reports 43
4.4.2 Template für die neuen Reports 44
4.4.3 Erstellen der nötigen Files 46
4.4.4 Anpassungen an bestehenden Files 47
4.4.5 Service Performance per hour 50
4.4.6 UTC Problematik 53
4.5 Performance Messung 54
4.6 Elasticsearch 54
4.7 Logstash 55

5 Testing **56**
5.1 Vorgehen 56
5.2 Testergebnisse 56
5.3 Defects 57
5.4 Fazit 57

6 Projektplanung **58**
6.1 Sprintplanung 58
6.2 Gantt Diagramm 59
6.3 Rollen und Meetings 59
6.3.1 Rollen 59
6.3.2 Meetings 59
6.4 Tools 60
6.4.1 Projektverwaltung 60
6.5 Risikoanalyse 61

7 Zukunftsausblick und Optimierungen **63**
7.1 Fertigstellung Agent Reports 63
7.2 Security 64
7.3 Konfiguration Elastic Produkte 64
7.4 Skalierung Elasticsearch 64
7.5 Logstash Import 65
7.6 Coderefactoring 65
7.7 Defectfixing 65

8 Schlussfolgerung **66**
8.1 Ergebnisse 66
8.1.1 Primäres Ziel 66
8.1.2 Sekundäres Ziel 66
8.2 Bewertung 67
8.3 Empfehlung 67
8.4 Ausblicke 67

Erklärung zur Urheberschaft **68**

Abbildungsverzeichnis **69**

Tabellenverzeichnis **69**

Code-Snippets **71**

Glossar **72**

Literaturverzeichnis **74**

Appendix **75**

Abstract

Heute ist die Datensammlung allgegenwärtig. In nahezu allen Bereichen werden enorm viele Daten generiert und gespeichert. Aus dem reinen Vorhandensein dieser Daten resultiert allerdings noch kein Vorteil für eine Unternehmung. Um einen Nutzen daraus ziehen zu können, muss der Zugriff auf Daten schnell, flexibel, unkompliziert und jederzeit erfolgen können. Für diese Zwecke gibt es eine Vielzahl von Produkten, welche sich auf diesen Bereich spezialisiert haben. Ein Beispiel dafür ist Elasticsearch.

Die Luware AG hat sich auf die Entwicklung von Microsoft Skype for Business basierten Lösungen spezialisiert. Ihr Hauptprodukt ist ein multimodales Contact Center, welches die Kontaktaufnahme über alle von Skype for Business möglichen Kanäle erlaubt. Das Reporting wird zurzeit über Microsoft SQL Reporting Services (SSRS) angeboten. Aufgrund der Entwicklungen im Markt und des Produktes muss in der Zukunft aber ein weitaus flexibleres Reporting zur Verfügung stehen, welches auch mit einer massiv grösseren Menge an Daten umgehen kann.

In einer ersten Phase soll eine Produktevaluation von Elasticsearch und einem anderen Open Source Produkt durchgeführt werden. Anschliessend wird eines dieser Produkte verwendet, um daraus einen Prototypen für das neue Web Reporting Portal zu entwickeln. Das Reporting Portal soll die Daten des Microsoft SQL Reporting Schemas des Contactcenters verwenden und eine bekannte Menge an definierten Reports anbieten. Dabei können bereits vorhandene Drittanbieter-Libraries wie Kendo UI verwendet werden.

Das Ergebnis dieser Arbeit ist ein neues Reporting Portal im Status eines Prototypen. Diese Client-Server Webapplikation nutzt für die Datenhaltung Elasticsearch. Dabei werden die Daten des bestehenden Reporting Schemas der Luware AG mittels Logstash importiert. Zur Erstellung der Reports kommt clientseitig Javascript (AngularJS), unter Verwendung von Kendo UI, zum Einsatz. Das neue Reporting führt zu einer massiven Performanceverbesserung in der Anzeige, entspricht optisch neuesten Anforderungen der Luware AG, ist einfach erweiter- und skalierbar und ist auch zukünftig für grösste Datenmengen vorbereitet.

Kapitel 1

Management Summary

1.1 Luware AG

Die Luware AG ist eine Schweizer Firma mit Sitz in Zürich. Sie ist führender Anbieter für kundenspezifische Service Plattformen auf der Basis von Microsoft Unified Communications (UC) Technologien.

1.2 Ausgangslage

In allen Bereichen wird eine unvorstellbare Menge an Daten gesammelt. Ein sehr wichtiger Bestandteil davon ist der Ort, an dem der Kunde mit dem Unternehmen in Kontakt tritt. In grösseren Unternehmen wird in der Regel ein Contact Center eingesetzt, welches für eine optimale Verteilung und Behandlung der Kundenanfragen über alle Kanäle zuständig ist. Dabei müssen diese Anfragen lückenlos rapportiert werden.

1.3 Problemstellung

Die Luware AG entstand als Startup aus einer erfolgreichen Bachelorarbeit der HSR und beschäftigt rund sechs Jahre nach der Gründung 18 Entwickler. Ein Produkt der Luware ist ein mulitmodales Contact Center, welches die Kontaktaufnahme über alle von Skype for Business möglichen Kanälen erlaubt. Das Reporting wird aktuell über das Web Frontend, SQL Server Reporting Services und MS Excel Powerpivot angeboten und war bis jetzt für die Menge der Daten völlig ausreichend. Aufgrund Marktentwicklungen muss zukünftig ein anpassungsfähigeres Reporting Portal zur Verfügung stehen, welches auch mit grössten Datenmengen umgehen kann.

1.4 Erwartetes Resultat

Primär: Ziel dieser Arbeit ist es, zunächst eine Evaluation von Elasticsearch und einem anderen verfügbaren Non Commercial/Open Source Produkt für die Durchsuchung, Analyse und Visualisierung von Daten durchzuführen. Die Kriterien für die Evaluation werden vom Projektteam mit Hilfe des Industriepartners eruiert. Anschliessend soll das Projektteam eines der beiden Produkte verwenden, um einen Prototypen des neuen Web Reporting Portals zu entwickeln. Das Portal soll Daten aus dem MS SQL Reporting Schema des Contact Centers verwenden und eine zu Beginn bekannte Menge an definierten Reports anbieten. Es können weitere, bereits bei der Luware AG im Einsatz stehende Drittanbieter-Libraries wie Kendo UI verwendet werden.

Sekundär (optional): Konnten die Reports erfolgreich abgebildet werden, soll das Portal um eine flexible Datenanalyse und Visualisierungsplatform erweitert werden.

1.5 Methode und Vorgehen

Das Projektvorgehen ist Scrum. Ein Sprint dauert jeweils zwei Wochen. Die zum Scrumprozess dazu gehörenden Meetings werden vor Ort bei der Luware AG oder via Skype durchgeführt (Daily, Planning, Demo, Retro). Es findet eine rege Zusammenarbeit mit dem Industriepartner statt, welcher zugleich als Product Owner (PO) agiert. Für weitere Details siehe Kapitel 6.

Kapitel 2

Analyse

2.1 Produktevaluation

Teil der Aufgabenstellung ist die Evaluation von zwei möglichen Produkten, welche die Erwartungen des Auftraggebers erfüllen können. Diese Produkte werden anhand verschiedener Kriterien bewertet und darauf basierend wird eine Produktempfehlung abgegeben. Anhand dieser Empfehlung entscheidet der Auftraggeber, mit welchem Produkt das Projekt weitergeführt werden soll.

2.1.1 Bewertungsraster

Folgendes Bewertungsraster dient der Beurteilung der beiden Produkte. Jedes Kriterium wird zwischen einem und fünf Sternen gewichtet.

Kriterium	**Erläuterung**	**Gewichtung**
K1: Lizenz	Braucht es für die Verwendung des Produkts spezielle Lizenzen? Welche Einschränkungen hat die entsprechende Lizenz? Hohe Bewertung = Wenig Einschränkungen	☆☆
K2: Kosten	Welche Kosten fallen für den Erwerb des Produktes und durch allfällige Lizenzgebühren an? Hohe Bewertung = Tiefe Kosten	☆☆☆☆
K3: Einbindung	Lässt sich das Produkt in die bestehende Infrastruktur/Umgebung einbinden? Passt das Produkt in die aktuelle Infrastruktur? Hohe Bewertung = Gute Einbindung	☆☆☆

K4: Integration	Lassen sich die bestehenden Daten (RDBMS) einfach integrieren? Hohe Bewertung = Einfache Datenintergration	☆☆☆
K5: Speicher	Wie speichert das Produkt Daten? Wie viel Datenspeicher ist dafür notwendig? Hohe Bewertung = Wenig Speicherverbrauch	☆☆
K6: Erweiterbarkeit	Ist das Produkt einfach um weitere Funktionen erweiterbar? Gibt es eine umfangreiche Palette an bestehenden Erweiterungen? Hohe Bewertung = Einfache Erweiterbarkeit	☆☆
K7: Performanz	Wie performant ist das Laden von Daten? Wie performant ist das Absetzen von Queries? Hohe Bewertung = Hohe Performanz	☆☆☆☆
K8: Skalierbarkeit	Lässt sich das Produkt skalieren? Wie einfach lässt sich das im laufenden Betrieb durchführen? Hohe Bewertung = Gute, einfache Skalierbarkeit	☆☆☆
K9: Produktreife	Wie gut hat sich das Produkt bis jetzt in der Realität beweisen können? Wie lange existiert es schon? Gibt es eine aktive Community und wird es aktiv weiterentwickelt? Ist das Produkt gut dokumentiert? Hohe Bewertung = Hohe Produktreife	☆☆☆
K10: Nutzung	Wie angenehm ist das initiale Einrichten, die Konfiguration und die Nutzung des Produktes? Hohe Bewertung = Einfache Nutzung	☆☆☆☆☆
K11: Visualisierung	Wie leicht lassen sich die gespeicherten Daten visualisieren? Hohe Bewertung = Einfache Visualisierung	☆☆☆☆
K12: Sicherheit	Gibt es Möglichkeiten zur Sicherheitsüberprüfung? Können diese konfiguriert werden? Hohe Bewertung = Gute Sicherheit	☆☆☆

Tabelle 2.1: Bewertungsraster

2.1.2 Solr vs. Elasticsearch

Nachfolgend werden die beiden Search-Engines Solr und Elasticsearch bewertet. Beide bauen auf Lucene auf. Die Evaluation wird zwischen diesen beiden Produkten vorgenommen, da beide marktführend sind. (siehe Abbildung 2.1) Die Bewertungen wurden nach dem Durcharbeiten von Tutorials erarbeitet.

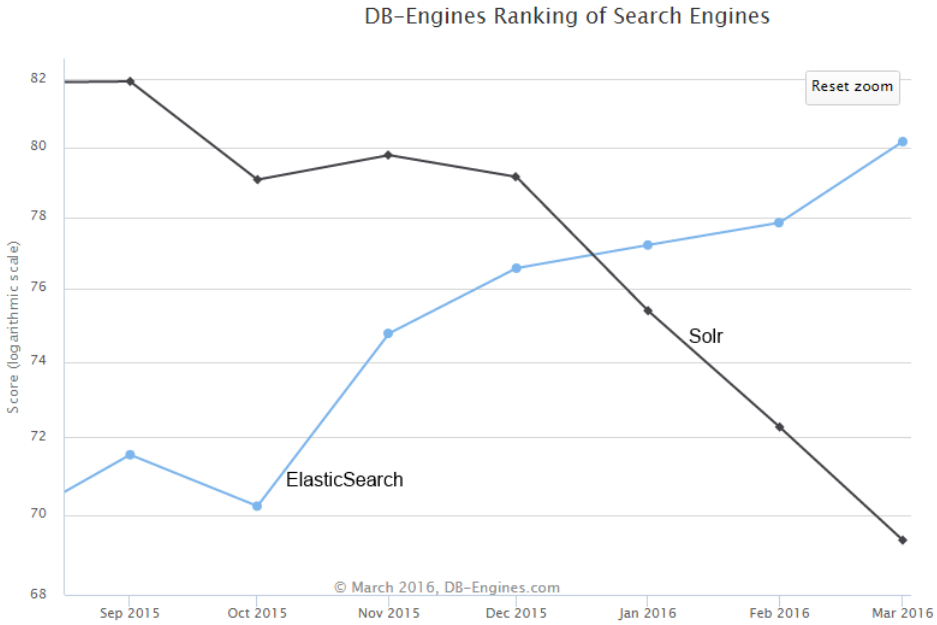


Abbildung 2.1: Trendanalyse: Solr vs. Elastic

Lucene

Apache Lucene ist eine Programmbibliothek zur Volltextsuche und ein Projekt der Apache Software Foundation. Lucene ist eine freie Software.

Solr

Solr ist ein in Lucene enthaltenes Servlet für entsprechende Container wie Apache Tomcat. Solr wurde ursprünglich von CNET entwickelt und Solar genannt. Der Name war eine Abkürzung für Search on Lucene and Resin. Der Download von Solr umfasst als Beispiel eine Konfiguration mit Jetty. Solr kommuniziert über das Hypertext Transfer Protocol. Mittels HTTP POST können verschiedenste Dateiformate von XML über JSON bis PDF erfasst und auch Dokumente erstellt werden. Abfragen erfolgen mittels HTTP GET. Solr ist die populärste Enterprise-Suchmaschine.[Wik16d]

Elasticsearch

Elasticsearch ist eine Suchmaschine auf Basis von Lucene. Das in Java geschriebene Programm speichert die Suchergebnisse in einem NoSQL-Format (JSON) und gibt sie über ein RESTful-Webinterface aus. Er ermöglicht auf einfache Weise den Betrieb im Rechnerverbund zur Umsetzung von Hochverfügbarkeit und Lastverteilung.[Wik16e]

2.1.3 Bewertung von Solr

Untenstehend folgt die Bewertung für Solr.

Kriterium	Begründung	Bewertung
K1	Solr verwendet die Apache Lizenz und ist open source.	☆☆☆☆☆
K2	Solr und auch die Plugins sind kostenlos.	☆☆☆☆☆
K3	Installation erfolgt über ein jar-File, ist problemlos möglich.	☆☆☆☆☆
K4	Ist über das DIH (DataImportHandler Plugin) möglich, dies muss jedoch explizit mit JDBC konfiguriert werden. Erster Versuch war leider nicht erfolgreich. Es muss ein manuelles Mapping auf die Entity Fields erfolgen.	☆☆
K5	Speicherung als JSON, CSV und XML sind out of the box möglich.	☆☆☆☆
K6	Solr bietet viele Plugins, die einfach integriert werden können.	☆☆☆☆
K7	Solr ist auf hohe Performanz ausgelegt.	☆☆☆☆
K8	Skalierung ist sehr einfach möglich. Cluster, Nodes und Indices können zur Laufzeit hinzugefügt werden. Shards müssen initial definiert werden.	☆☆☆☆
K9	Seit 2006 bei der Apache Software Foundation. Grosse Community. Regelmässige Releases, aktueller Release: 5.5.0, 22 Februar 16. Gut dokumentiert, jedoch könnte es mehr Videotutorials geben. Hat den Spitzenplatz per Ende 2015 an Elasticsearch abgegeben.	☆☆☆☆
K10	Einstieg ist einfach, mit dem Daten Import wird es kompliziert (RDBMS).	☆☆☆☆
K11	Banana unterstützt die Visualisierung, jedoch handelt es sich um einen Kibana Port, welcher einen eingeschränkten Funktionsumfang anbietet.	☆☆☆
K12	Solr bietet ein Security Plugin für die Authentifizierung und Autorisierung.	☆☆☆☆

Tabelle 2.2: Bewertungsraster Solr

2.1.4 Bewertung von Elasticsearch

Untenstehend folgt die Bewertung für Elasticsearch.

Kriterium	Begründung	Bewertung
K1	Elasticsearch verwendet die Apache Lizenz und ist open source.	☆☆☆☆☆
K2	Elasticsearch und benötigte Plugins (Logstash, Sense und Kibana) kostenlos. Zusätzliche, wie Shield (Security) oder Marvel (Monitoring) kostenpflichtig.	☆☆☆☆
K3	Installation erfolgt über ein jar-File, ist problemlos möglich.	☆☆☆☆☆
K4	Einfach möglich, erster Versuch erfolgreich. JDBC Schnittstelle für Logstash vorhanden.	☆☆☆☆☆
K5	Speicherung als JSON-Documents.	☆☆☆
K6	Elasticsearch bietet viele Plugins (teilweise kostenpflichtig), die einfach integriert werden können. Bestehende Datenablagen können einfach erweitert werden (neue Indices, Types).	☆☆☆☆
K7	Elasticsearch ist auf hohe Performance und sehr grosse Datenmengen ausgelegt.	☆☆☆☆
K8	Skalierung ist sehr einfach möglich. Cluster, Nodes und Indices können zur Laufzeit hinzugefügt werden. Shards müssen zum Definitionszeitpunkt des Indices definiert werden (nicht mehr veränderbar).	☆☆☆☆
K9	Das Produkt ist seit 2010 auf dem Markt. Es ist mittlerweile globaler Marktführer bei Searchengines, somit ist die Community gross. Regelmässige Releases, aktueller Release: 2.2.0, 2. Februar 16. Sehr gut dokumentiert (Doku, Videotutorials etc.).	☆☆☆☆☆
K10	Schritt für Schritt Einrichtung. Einfacher Einstieg und erste Nutzung.	☆☆☆☆☆
K11	Kibana unterstützt umfangreiche Visualisierungsmöglichkeiten.	☆☆☆☆☆
K12	Elasticsearch bietet das Security Plugin Shield für die Authentifizierung und Autorisierung, es bietet sehr viele Möglichkeiten, ist allerdings kostenpflichtig.	☆☆☆☆

Tabelle 2.3: Bewertungsraster Elaticsearch

2.1.5 Auswertung und Vergleich

Die Auswertung erfolgt gemäss der Formel: $\frac{\sum_{n=1}^{12} Bewertung_{K_n} * Gewichtung_{K_n}}{\sum_{n=1}^{12} Gewichtung_{K_n}}$

Untenstehend folgt der Vergleich der beiden ausgewerteten Produkte.

Suchmaschine	**Anzahl Sterne**
Elasticsearch	4.5
Solr	3.9

Tabelle 2.4: Auswertung Elaticsearch vs. Solr

Aus Tabelle 2.4 kann entnommen werden, dass Elasticsearch klar zu favorisieren ist. Der weitere Verlauf der Arbeit stützt sich auf diesen Entscheid, welcher durch das Team und den Product Owner gefällt worden ist.

2.2 Vertikales Prototyping (Durchstich)

Im Projekt wird möglichst früh ein sogenannter Durchstich angestrebt, das heisst, ein bestimmter Teil des Systems wird durch alle Ebenen hindurch implementiert. In Absprache mit dem Industriepartner ist dies bei uns der SQL Server Reporting Services-Report "Service_ServiceInfo_PerDay". Ziel ist es, diesen Report möglichst früh im neuen System nachzubauen. Wir erhoffen uns damit Erkenntnisse für die nächsten Schritte, vorallem für das Design der Applikation und für die Auswahl der zu verwendenden Technologien. Zudem lässt sich auch schnell erkennen, ob das im Kapitel 2.1 gewählte Produkt "Elasticsearch" sich für die Umsetzung eignet. Es folgt eine Auflistung der für den Durchstich notwendigen Arbeitsschritte und der Tools mit welchen diese umgesetzt werden. Für die Dokumentation, wie diese genau eingesetzt werden (Installation und Konfiguration), siehe Anhang "Installations- und Konfigurationsanleitung für Elastic Produkte".

2.2.1 Datenimport

Der bestehende Report *Service_ServiceInfo_PerDay* joined heute die folgenden vier Tabellen des Luware Datenbankschemas *Reporting*:

- UnifiedSession
- Session
- ContactObject
- OrganizationUnit

Die Daten dieser Tabellen werden mittels Logstash ins Elasticsearch importiert.

Fazit

Datenimport ist relativ einfach möglich. Mittels dem JDBC-Treiber für Logstash werden SQL-Queries abgesetzt, sodass die Daten ab der bestehenden SQL-Datenbank selektiert werden können. Logstash wird weiter verwendet.

2.2.2 Datenspeicherung

Die Daten werden im Elasticsearch gespeichert. Gewisse Tabellen müssen denormalisiert werden (siehe Kapitel 3.1.2), damit die in der bestehenden SQL-Datenbank vorhandenen Beziehungen zwischen den Tabellen auch in den Elasticsearch-Queries abgebildet werden können.

Fazit

Datenstruktur lässt sich beim Datenimport bereits festlegen und erfolgt automatisch. Mittels JDBC-Treiber können Joins abgesetzt werden, sodass wo nötig denormalisierte Daten im Elasticsearch gespeichert werden. Elasticsearch wird weiter verwendet.

2.2.3 Datenvisualisierung

Die gespeicherten Daten werden visualisiert. Diese sollen im neuen System weitgehend den bestehenden Visualisierungen entsprechen oder zumindest den selben Informationsgehalt abdecken.

Fazit

Kibana ermöglicht viele Visualisierungen. Aggregationen (z.B. Prozentberechnungen) sind aber nur eingeschränkt möglich und verlangen viel Kibana-Knowhow und Spezialabfragen. Es war auch nach einem mehrstündigem Versuch nicht möglich, die notwendige Information in einer, dem alten Report entsprechenden Visualisierung, abzubilden. Nach Rücksprache mit dem Industriepartner wird Kibana für die Erstellung der Reports nicht weiter verwendet. Kibana bleibt aber installiert und wird allenfalls für die Umsetzung der dynamischen Reports verwendet (sekundäres Ziel der Arbeit). Als Ersatz für Kibana wird auf Empfehlung des Industriepartners Kendo UI verwendet. Kendo UI ist bereits bei der Luware AG im Einsatz und bietet viele Visualisierungsmöglichkeiten.

2.2.4 Benutzerinteraktion und Anzeige

Die Reports müssen via GUI aufgerufen und angezeigt werden. Dazu wird AngularJS verwendet.

Fazit

Die Queries zur Ermittlung der zu visualisierenden Daten lassen sich via RESTful API von Elasticsearch gut integrieren. Komponenten von Kendo UI lassen sich ebenfalls einfach einbinden. AngluarJS wird weiter verwendet.

2.2.5 Erkenntnisse

Der erste Durchstich war erfolgreich. Die Produkte von Elastic (Logstash und Elasticsearch) in Kombination mit Kendo UI und AngularJS sind hervorragend für die Umsetzung der Aufgabenstellung geeignet. Weitere wichtige Erkenntnisse sind:

- Gewisse Tabellen müssen beim Datenimport mit Logstash denormalisiert werden, damit Informationen über Verbindungen unter den SQL-Tabellen auch im Elasticsearch nachvollzogen werden können.
- Logstash erkennt beim Import veränderte und neue Rows und übernimmt diese automatisch ins Elasticsearch. Gelöschte Rows werden nicht nachvollzogen (bleiben also im Elasticsearch bestehen). Dies ist aber nach Absprache mit dem Industriepartner auch nicht notwendig, da nie nur einzelne Rows gelöscht werden.
- Alle in den Reports benötigten Daten können im Elasticsearch mittels Absetzen von Queries über dessen RESTful API ermittelt werden.
- Aggregierte Daten lassen sich nicht in geeigneter Form im Kibana abbilden. Als Alternative kommt Kendo UI zum Einsatz.

Genauere Erkenntnisse und deren Einfluss auf den Design der Applikation sind im Kapitel 3 festgehalten.

2.3 Anforderungsanalyse

Im folgenden Abschnitt werden die Anforderungen an das neue Reporting Portal festgehalten. Diese entsprechen funktional weitgehend den Anforderungen an das bisherige Reporting Portal. Alle im bestehenden Portal vorhandenen SQL Server Reporting Services-Reports sollen auch im neuen Portal abgebildet werden. Nichtfunktional soll vor allem auf die Leistung und Effizienz Wert gelegt werden, sodass das neue Portal bei Erstellung von Reports bessere Antwortzeiten hat als das bestehende.

2.3.1 Funktionale Anforderungen

Die funktionalen Anforderungen sind in zwei Epics aufgeteilt (siehe Tabelle 2.5).

ID	Name	Beschreibung
E1	Neues Reporting Portal	Entwicklung eines Reporting Portals mit Basis Reports (primäres Ziel)
E2	Dynamische Reports	Erweiterung des Portals um dynamische Reports (sekundäres, optionales Ziel)

Tabelle 2.5: Funktionale Anforderungen

2.3.2 Use Cases

Das in Abbildung 2.2 aufgeführte Diagramm zeigt die für das neue Reporting Portal relevanten Use Cases.

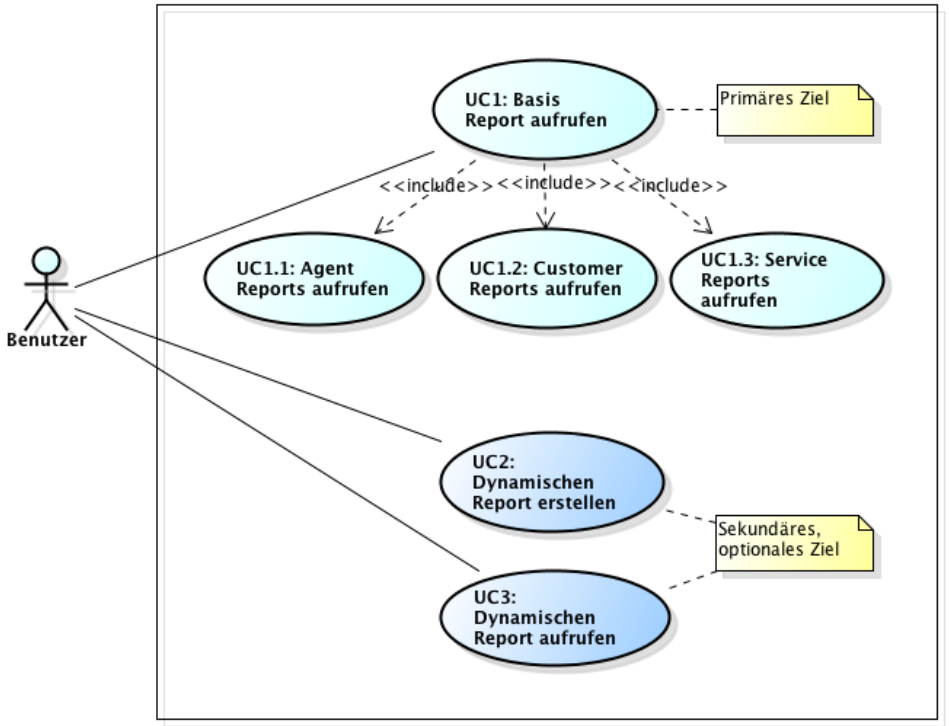


Abbildung 2.2: Use Case Diagramm

UC1: Basis Report aufrufen

Mapped Epic	E1
Actor	Benutzer
Pre Condition	Benutzer hat Zugriff auf Reporting Portal
Post Condition	Alle Reports der drei Bereiche Service, Agent und Customer werden zur Auswahl angezeigt
Main Scenario	Benutzer steigt via Internetbrowser in das Reporting Portal ein

Tabelle 2.6: UC1

UC1.1: Agent Report aufrufen

Mapped Epic	E1
Actor	Benutzer
Pre Condition	Benutzer befindet sich im Reporting Portal
Post Condition	Ausgewählter Agent Report wird angezeigt
Main Scenario	Benutzer selektiert den gewünschten Agent Report

Tabelle 2.7: UC1.1

UC1.2: Customer Report aufrufen

Mapped Epic	E1
Actor	Benutzer
Pre Condition	Benutzer befindet sich im Reporting Portal
Post Condition	Ausgewählter Customer Report wird angezeigt
Main Scenario	Benutzer selektiert den gewünschten Customer Report

Tabelle 2.8: UC1.2

UC1.3: Service Report aufrufen

Mapped Epic	E1
Actor	Benutzer
Pre Condition	Benutzer befindet sich im Reporting Portal
Post Condition	Ausgewählter Service Report wird angezeigt
Main Scenario	Benutzer selektiert den gewünschten Service Report

Tabelle 2.9: UC1.3

UC2: Dynamischen Report erstellen

Mapped Epic	E2
Actor	Benutzer
Pre Condition	Benutzer befindet sich aktuell im Reporting Portal
Post Condition	Dynamischer Report wurde erstellt und gespeichert
Main Scenario	Benutzer wählt die Funktion zur Erstellung eines dynamischen Reports aus und wird dadurch ins Kibana weitergeleitet. Benutzer klickt sich den gewünschten Report im Kibana zusammen. Benutzer speichert den gewünschten Report

Tabelle 2.10: UC2

UC3: Dynamischen Report aufrufen

Mapped Epic	E2
Actor	Benutzer
Pre Condition	Benutzer befindet sich aktuell im Reporting Portal. Der gewünschte dynamische Report wurde erstellt
Post Condition	Der dynamische Report wird angezeigt
Main Scenario	Benutzer selektiert den gewünschten dynamischen Report

Tabelle 2.11: UC2

2.3.3 Aktuelle Reports

Es folgt eine Auflistung aller bisherigen SQL Server Reporting Services-Reports, deren involvierten Tabellen (ab welchen die Daten gelesen werden) und möglichen Parametern, mit denen die Reports aufgerufen werden können. Da alle Reports auch im neuen Reporting Portal abgebildet werden müssen, dient diese Auflistung als grundlegende Anforderung an die neu zu bildenden Reports. Die Reports sind in die drei Bereiche "Service", "Agent" und "Customer" aufgeteilt. Alle bestehenden Reports sind Stored Procedures und beginnen mit dem Namen: "`usp_Report_SSRS_`"

Service Reports

Name	Service: Overview Daily / Monthly / Weekly

Stored Procedures	`Service_ServiceDetails`
	`Service_ServiceInfo_PerDay`
	`Service_ServiceInfo_PerWeek`
	`Service_ServiceInfo_PerMonth`

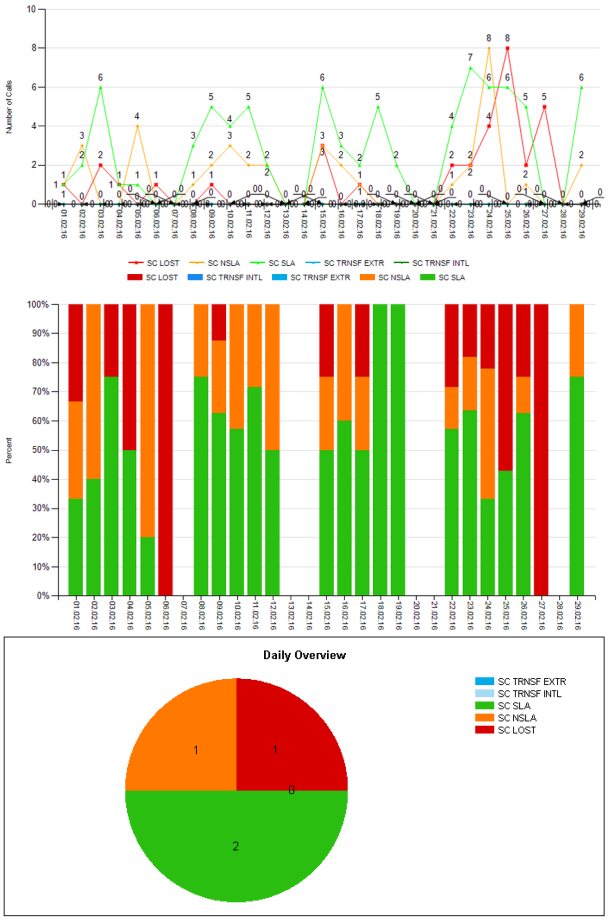
Beschreibung	Zeigt eine aufsummierte Übersicht der Sessions und wie diese verarbeitet wurden (pro Tag, Woche oder Monat)
Tabellen	`OrganizationUnit, ContactObject, Dates, UnifiedSession, Session`
Parameter	`Offset, UseDaylightSavings, OrganizationUnit, DateFrom, DateTo, ContactObjectList, ShowEarlyLosts`
Visualisierung	

Tabelle 2.12: Service: Overview

Name	Service: Performance 15min / Hour / Daily / PerWeekDay / Monthly

Stored Procedures	`Service_ServicePerformanceDetails`
	`Service_ServicePerformance_15min`
	`Service_ServicePerformance_Hour`
	`Service_ServicePerformance_Day`
	`Service_ServicePerformance_WeekDay`
	`Service_ServicePerformance_Month`

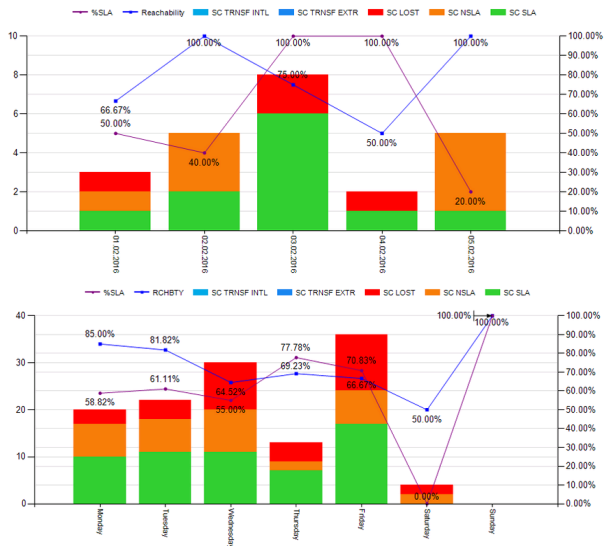
Beschreibung	Zeigt die Performance der Sessions und wie diese verarbeitet wurden (pro 15 Minuten, Tag, Woche oder Monat)
Tabellen	`OrganizationUnit, ContactObject, Dates, Times, UnifiedSession, Session`
Parameter	`Offset, UseDaylightSavings, OrganizationUnit, DateFrom, DateTo, ContactObjectList`
Visualisierung	

Tabelle 2.13: Service: Performance

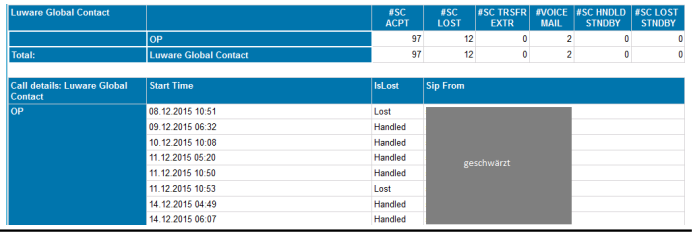
Name	Service: Performance per Opening Hour Type
Stored Procedures	`Service_PerformancePerOpeningHourType`
Beschreibung	Listet die eingegangenen SIP-Anrufe für einen bestimmten Zeitintervall auf und gibt an, ob diese "handled" oder "lost" sind
Tabellen	`ContactObject, UnifiedSession, Session`
Parameter	`Offset, UseDaylightSavings, DateFrom, DateTo, ContactObjectList`
Visualisierung	

Tabelle 2.14: Service: Performance per Opening Hour Type

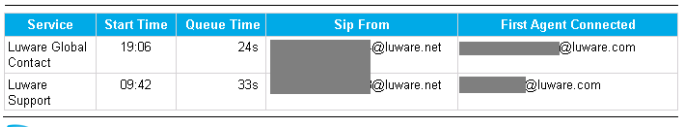
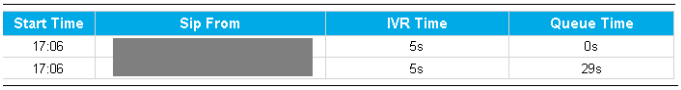
Name	Service: Not in SLA daily
Stored Procedures	`Service_ServiceCallsNotInSla`
Beschreibung	Listet die eingegangenen SIP-Anrufe, deren Anrufzeitpunkt und Queue-Time für ein bestimmtes Datum auf und gibt an, welcher Agent den Anruf als erstes entgegengenommen hat. Optional ist der Parameter "`SipFrom`" mit welchem nur Adressen die mit diesem übereinstimmen, berücksichtigt werden
Tabellen	`ContactObject, UnifiedSession, Session, Agent`
Parameter	`Offset, UseDaylightSavings, Day, ContactObjectList, OrganizationUnit (optional), SipFrom (optional)`
Visualisierung	

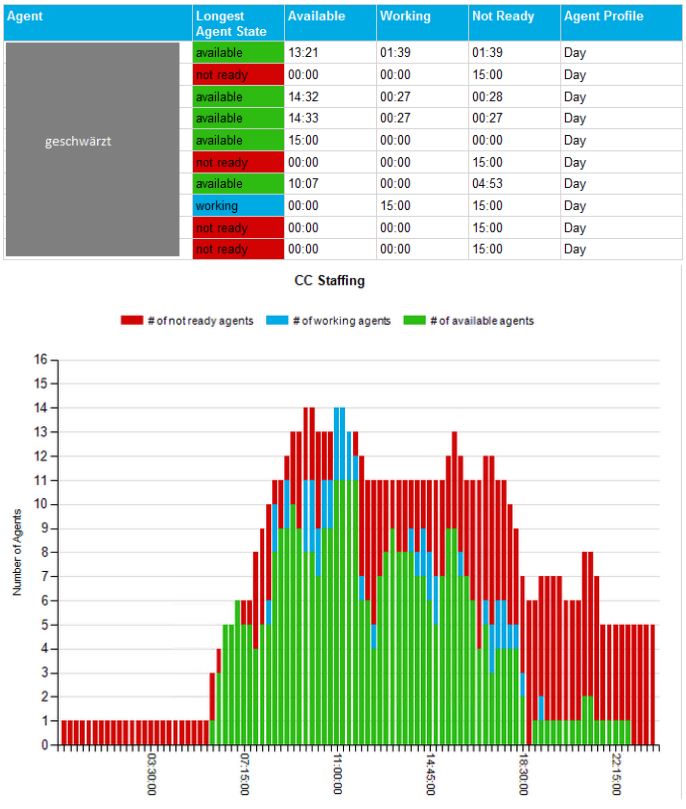
Tabelle 2.15: Service: Not in SLA daily

Name	Service: Lost Call Details
Stored Procedures	`Service_ServiceLostCalls`
Beschreibung	Listet die verlorengegangenen (lost) SIP-Anrufe für ein bestimmtes Datum auf, deren IVR- und Queue-Time
Tabellen	`UnifiedSession, Session`
Parameter	`Offset, UseDaylightSavings, Date, ContactObjectList`
Visualisierung	

Start Time	Sip From	IVR Time	Queue Time
17:06	[redacted]	5s	0s
17:06		5s	29s

Tabelle 2.16: Service: Lost Call Details

Agent Reports

Name	Agent: CC Staffing Daily, CC Staffing Detail
Stored Procedures	`Agent_Staffing_PerDay` `Agent_Staffing_Per15MinSlot`
Beschreibung	Zeigt die Agent State (available, working, not ready) der einzelnen Agents an. Wahlweise täglich oder alle 15min.
Tabellen	`AgentActivity, Agent, AgentProfile, OrganizationUnit`
Parameter	`Offset, UseDayLightSavings, OrganizationUnit, Day, TimeFrom`
Visualisierung	

Agent	Longest Agent State	Available	Working	Not Ready	Agent Profile
geschwärzt	available	13:21	01:39	01:39	Day
	not ready	00:00	00:00	15:00	Day
	available	14:32	00:27	00:28	Day
	available	14:33	00:27	00:27	Day
	available	15:00	00:00	00:00	Day
	not ready	00:00	00:00	15:00	Day
	available	10:07	00:00	04:53	Day
	working	00:00	15:00	15:00	Day
	not ready	00:00	00:00	15:00	Day
	not ready	00:00	00:00	15:00	Day

Tabelle 2.17: Agent: Staffing

Name	Agent: Concurrent Agents per Day/per Month

Stored Procedures	`Agent_ConcurrentAgentsDetails`
	`Agent_ConcurrentAgent_PerDay`
	`Agent_ConcurrentAgent_PerMonth`

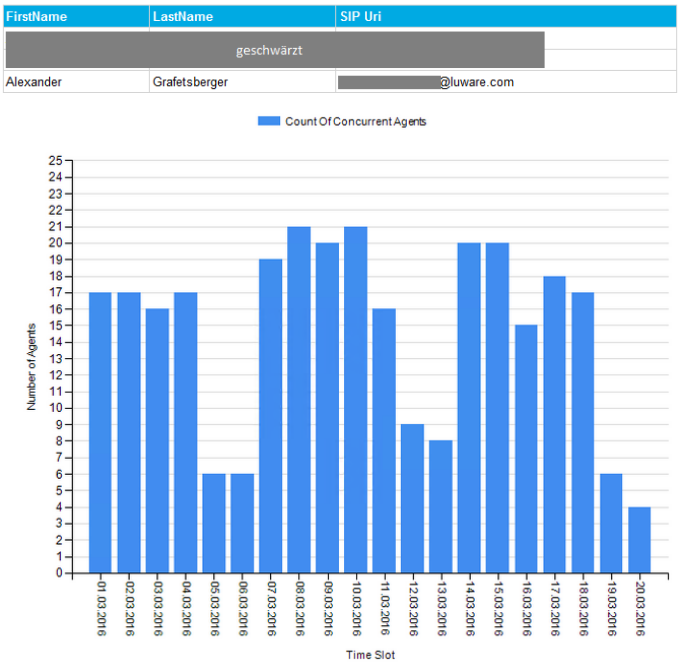
Beschreibung	Zeigt die gleichzeitig arbeitenden Personen pro Tag oder Monat.
Tabellen	`Dates, Agent, AgentActivity`
Parameter	`Offset, UseDayLightSavings, OrganizationUnit, DateFrom, DateTo, Day, Month, MaxDateOnly`
Visualisierung	

Tabelle 2.18: Agent: Concurrent Agents

Name	Agent: NotReady Reasons per Week
Stored Procedures	`Agent_NotReadyReasons_PerWeek` `Agent_ConcurrentAgent_PerMonth`
Beschreibung	Zeigt die Personen und ihre Gründe auf, wieso sie nicht bereit waren.
Tabellen	`Dates, Agent, AgentStatePeriod, NotReadyReason, AgentNotReadyReason, OrganizationUnit`
Parameter	`Offset, UseDayLightSavings, OrganizationUnit, WeekFrom, WeekTo, AgentIdList`
Visualisierung	

Tabelle 2.19: Agent: Not Ready Reasons

Name	Agent: Customer Performance 15Min

Stored Procedures	`Agent_NotReadyReasons_PerWeek`
	`Agent_ConcurrentAgent_PerMonth`
	`Service_ServiceCallsNotInSla`
	`Service_ServiceLostCalls`
	`RunningUserInfoLine`
	`Customer_ServicePerformance_15Min`

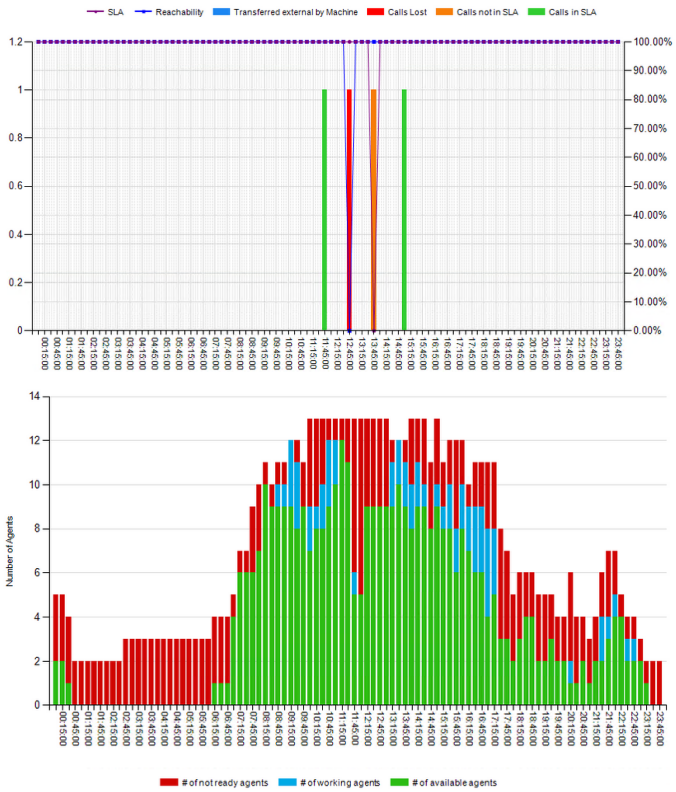
Beschreibung	Zeigt die Performance der Services im Vergleich zu den Agent States
Tabellen	`AgentActivity, Agent, AgentProfile, OrganizationUnit, Dates, AgentStatePeriod, NotReadyReason, AgentNotReadyReason, UnifiedSession, Session, ContactObject, _ReportList`
Parameter	`Offset, UseDayLightSavings, OrganizationUnit, Date, FirstQueuedContactObjectList, UseFilterOnFirstQueuedContactObject, IncomingContactObjectList, ReportName`
Visualisierung	

Tabelle 2.20: Agent: Customer Performance 15Min

Name	Agent: Service per Month

Stored Procedures	`Dimension_AllMonths`
	`Dimension_AllTimeZones`
	`Dimension_OrganizationUnitsFlattenTree`
	`Dimension_ContactObjects`
	`Dimension_Agents`
	`Agent_Service_PerMonth`
	`RunningUserInfoLine`

Beschreibung	Zeigt die Anzahl Services und die durchschnittliche Queuetime pro Monat
Tabellen	`Agent, ContactObject`
Parameter	`Offset, ContactObjectList, UseDayLightSavings, OrganizationUnit, Month, AgentIdList, ReportName,`
Visualisierung	

per service

Service	Service phone	Total #SC	Total #SC OH	Total %SC NA	Total %SC NA OH	Total AvgQT SC	External #SC	External #SC OH	External %SC NA	External %SC NA OH	External AvgQT SC	Internal #SC	Internal #SC OH	Internal %SC NA	Internal %SC NA OH
geschwärzt		60	60	62%	62%	44s	60	60	62%	62%	44s	0	0	0%	0%
		3	3	75%	75%	2min, 17s	3	3	75%	75%	2min, 17s	0	0	0%	0%
		57	57	63%	63%	59s	57	57	63%	63%	59s	0	0	0%	0%
		8	8	75%	75%	41s	7	7	58%	58%	38s	1	1	17%	17%

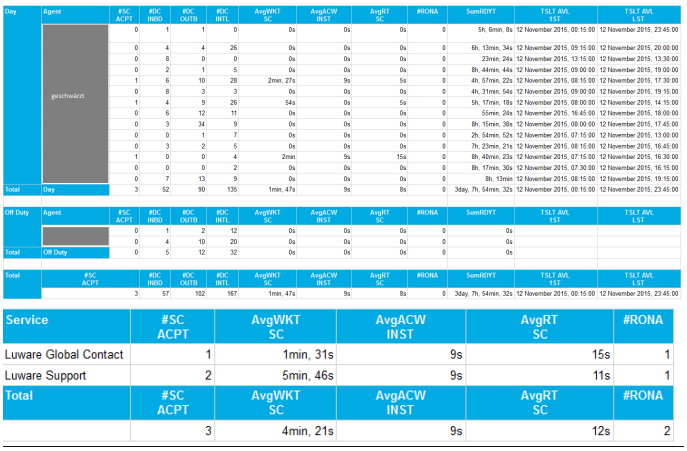
per agent

Agent	SIP uri	Service phone	Total #SC	Total #SC OH	Total %SC NA	Total %SC NA OH	Total AvgQT SC	External #SC	External #SC OH	External %SC NA	External %SC NA OH	External AvgQT SC	Internal #SC	Internal #SC OH	Internal %SC NA	Internal %SC NA OH
geschwärzt			3	3	100%	100%	1min, 29s	3	3	100%	100%	1min, 29s	0	0	0%	0%
			12	12	100%	100%	1min, 9s	12	12	100%	100%	1min, 9s	0	0	0%	0%
			2	2	100%	100%	50s	2	2	100%	100%	50s	0	0	0%	0%
			4	4	75%	75%	26s	4	4	75%	75%	26s	0	0	0%	0%
			1	1	100%	100%	3min	1	1	100%	100%	3min	0	0	0%	0%
			2	2	50%	50%	51s	2	2	50%	50%	51s	0	0	0%	0%
			1	1	0%	0%	30s	1	1	0%	0%	30s	0	0	0%	0%
			2	2	50%	50%	40s	2	2	50%	50%	40s	0	0	0%	0%
			13	13	46%	46%	23s	13	13	46%	46%	23s	0	0	0%	0%
			16	16	6%	6%	1min, 3s	16	16	6%	6%	1min, 3s	0	0	0%	0%
			1	1	100%	100%	1min, 3s	1	1	100%	100%	1min, 3s	0	0	0%	0%
			15	15	40%	40%	20s	15	15	40%	40%	20s	0	0	0%	0%
			2	2	50%	50%	1min, 33s	2	2	50%	50%	1min, 33s	0	0	0%	0%
			7	7	42%	42%	27s	7	7	42%	42%	27s	0	0	0%	0%

Tabelle 2.21: Agent: Service Per Month

Name	Agent: Performance per Day/per Profile/per Service

Stored Procedures	`Dimension_AllTimeZones`
	`Dimension_Agents`
	`Agent_Detail_PerProfile_And_Day`
	`Agent_Detail_PerService_And_Day`
	`Dimension_OrganizationUnitsFlattenTree`
	`RunningUserInfoLine`

Beschreibung	Zeigt die Performance pro Tag, pro Service oder pro Profil
Tabellen	`Session, UnifiedSession, SessionAgent, Agent, AgentProfile, ContactObject, AgentActivity, _ReportList`
Parameter	`Offset, UseDayLightSavings, OrganizationUnit, Day, AgentIdList, ReportName`
Visualisierung	

Day	Agent	#SC ACPT	#DC INBD	#DC OUTB	#DC INTL	AvgWKT SC	AvgACW INST	AvgRT SC	#RONA	SumRDYT	TSLT AVL 1ST	TSLT AVL LST
	gaschwarzt	0	1	1	0	0s	0s	0s	0	5h, 6min, 8s	12 November 2015, 00:15:00	12 November 2015, 23:45:00
		0	4	4	26	0s	0s	0s	0	6h, 13min, 34s	12 November 2015, 09:15:00	12 November 2015, 20:00:00
		0	8	0	0	0s	0s	0s	0	23min, 24s	12 November 2015, 13:15:00	12 November 2015, 13:30:00
		0	2	1	5	0s	0s	0s	0	8h, 44min, 44s	12 November 2015, 09:00:00	12 November 2015, 19:00:00
		1	6	10	28	2min, 27s	9s	5s	0	4h, 57min, 22s	12 November 2015, 08:15:00	12 November 2015, 17:30:00
		0	8	3	3	0s	0s	0s	0	4h, 31min, 54s	12 November 2015, 09:00:00	12 November 2015, 19:15:00
		1	4	9	26	54s	0s	5s	0	5h, 17min, 18s	12 November 2015, 08:00:00	12 November 2015, 14:15:00
		0	6	12	11	0s	0s	0s	0	55min, 24s	12 November 2015, 16:45:00	12 November 2015, 18:00:00
		0	3	34	9	0s	0s	0s	0	8h, 15min, 38s	12 November 2015, 08:00:00	12 November 2015, 17:45:00
		0	0	1	7	0s	0s	0s	0	2h, 54min, 52s	12 November 2015, 07:15:00	12 November 2015, 13:00:00
		0	3	2	5	0s	0s	0s	0	7h, 23min, 21s	12 November 2015, 08:15:00	12 November 2015, 16:45:00
		1	0	0	4	2min	9s	15s	0	8h, 40min, 23s	12 November 2015, 07:15:00	12 November 2015, 16:30:00
		0	0	0	2	0s	0s	0s	0	8h, 17min, 30s	12 November 2015, 07:30:00	12 November 2015, 16:15:00
		0	7	13	9	0s	0s	0s	0	8h, 13min	12 November 2015, 08:15:00	12 November 2015, 19:15:00
Total	Day	3	52	90	135	1min, 47s	9s	8s	0	3day, 7h, 54min, 32s	12 November 2015, 00:15:00	12 November 2015, 23:45:00

Off Duty	Agent	#SC ACPT	#DC INBD	#DC OUTB	#DC INTL	AvgWKT SC	AvgACW INST	AvgRT SC	#RONA	SumRDYT	TSLT AVL 1ST	TSLT AVL LST
		0	1	2	12	0s	0s	0s	0	0s		
		0	4	10	20	0s	0s	0s	0	0s		
Total	Off Duty	0	5	12	32	0s	0s	0s	0	0s		

Total	#SC ACPT	#DC INBD	#DC OUTB	#DC INTL	AvgWKT SC	AvgACW INST	AvgRT SC	#RONA	SumRDYT	TSLT AVL 1ST	TSLT AVL LST
	3	57	102	167	1min, 47s	9s	8s	0	3day, 7h, 54min, 32s	12 November 2015, 00:15:00	12 November 2015, 23:45:00

Service	#SC ACPT	AvgWKT SC	AvgACW INST	AvgRT SC	#RONA
Luware Global Contact	1	1min, 31s	9s	15s	1
Luware Support	2	5min, 46s	9s	11s	1
Total	**#SC ACPT**	**AvgWKT SC**	**AvgACW INST**	**AvgRT SC**	**#RONA**
	3	4min, 21s	9s	12s	2

Tabelle 2.22: Agent: Performance per Day/Profile/Service

Name	Agent: Performance Details/per Month/per Week

Stored Procedures	`Dimension_AllWeeks`
	`Dimension_AllMonths`
	`Dimension_AllAgentProfiles`
	`Agent_Performance_PerMonth`
	`Dimension_OrganizationUnitsFlattenTree`
	`RunningUserInfoLine`
	`Dimension_AllTimeZones`

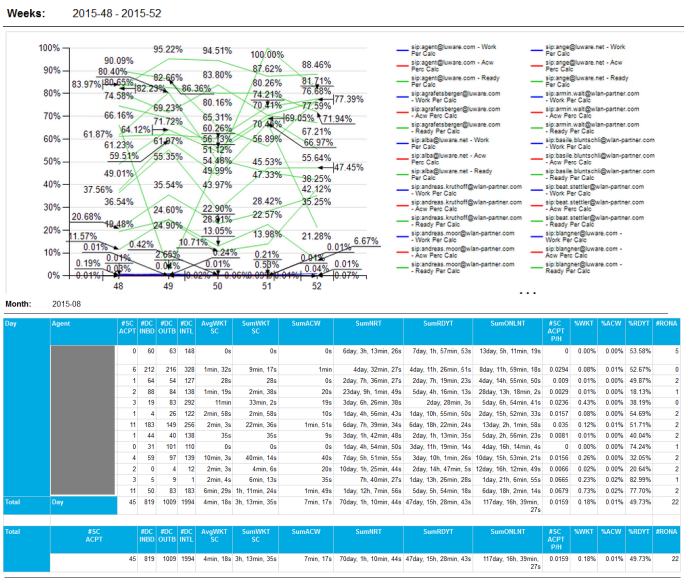
Beschreibung	Zeigt die Performance pro Monat oder Woche
Tabellen	`User, _ReportList, Dates, Agent, _SysSetup, UtcOffsetSetting, SessionAgent, Session, ContactObject, UnifiedSession, AgentStatePerHour, AgentActivity`
Parameter	`Offset, UseDayLightSavings, OrganizationUnit, Month, WeekFrom, WeekTo AgentProfileList, AgentIdList, ReportName`
Visualisierung	

Tabelle 2.23: Agent: Performance per month/Week

Customer Reports

Alle Customer Reports sind sehr ähnlich wie die Service Reports aufgebaut, berücksichtigen aber die Tabelle `UnifiedSession` und nicht die Tabelle `Session`. Ausserdem enthalten sie zusätzliche Parameter.

Name	Customer: Overview Daily / Monthly / Weekly

Stored Procedures	`Customer_ServiceDetails`
	`Customer_ServiceInfo_PerDay`
	`Customer_ServiceInfo_PerWeek`
	`Customer_ServiceInfo_PerMonth`

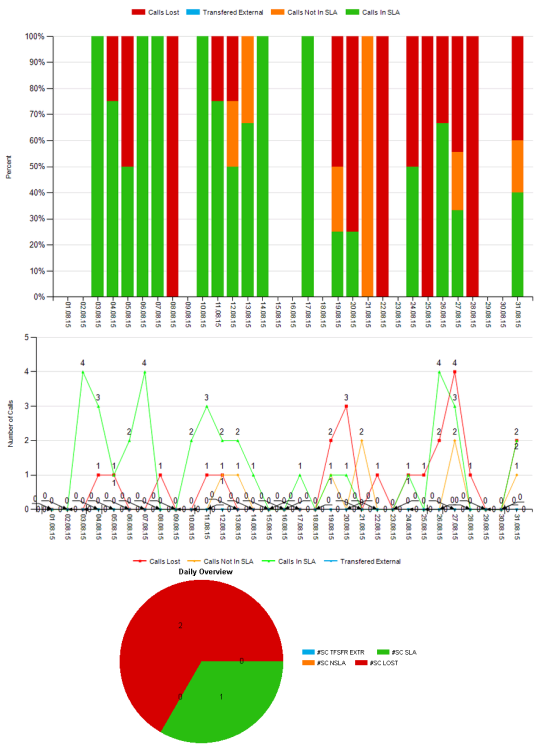
Beschreibung	Zeigt eine aufsummierte Übersicht der Sessions und wie diese verarbeitet wurden (pro Tag, Woche oder Monat). Ist ähnlich wie Report "Service: Overview"
Tabellen	`OrganizationUnit, ContactObject, Dates, UnifiedSession`
Parameter	`Offset, UseDaylightSavings, DateTo, DateFrom, OrganizationUnit, IncomingContactObjectList, ShowEarlyLosts, FirstQueuedContactObjectList, UseFilterOnFirstQueuedContactObject`
Visualisierung	

Tabelle 2.24: Customer: Overview

Name	Customer: Performance 15min / Hour / Daily / PerWeekDay / Monthly

Stored Procedures	`Customer_ServicePerformanceDetails`
	`Customer_ServicePerformanceGeneral`
	`Customer_ServicePerformance_15min`
	`Customer_ServicePerformance_Hour`
	`Customer_ServicePerformance_Day`
	`Customer_ServicePerformance_WeekDay`
	`Customer_ServicePerformance_Month`

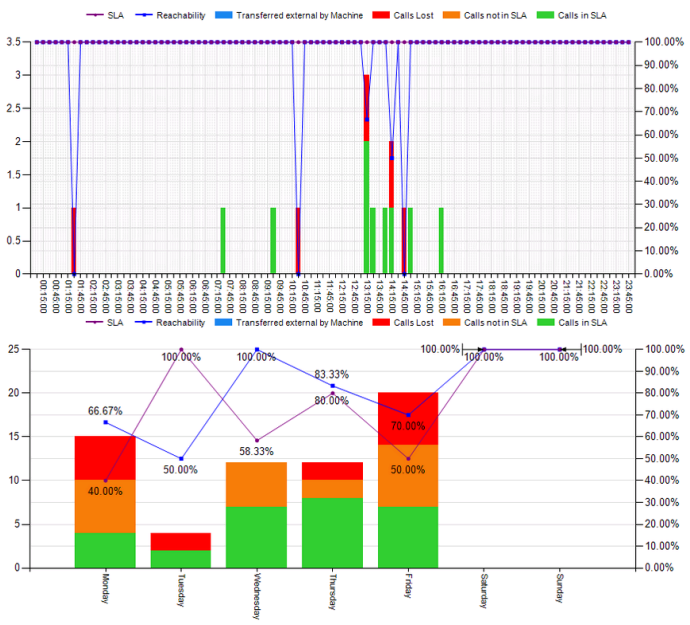
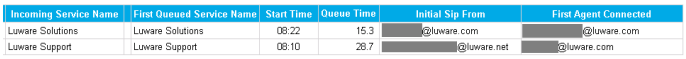
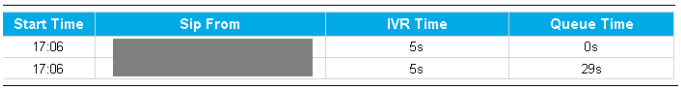
Beschreibung	Zeigt die Performance der UnifiedSessions und wie diese verarbeitet wurden (pro 15 Minuten, Tag, Woche oder Monat). Der monthly Report nutzt das Stored Procedure `Customer_ServicePerformanceGeneral`, alle anderen das `Customer_ServicePerformanceDetails`. Ist ähnlich wie Report "Service: Performance".
Tabellen	`OrganizationUnit, ContactObject, Dates, Times, UnifiedSession, Session`
Parameter	`Offset, UseDaylightSavings, OrganizationUnit, DateFrom, DateTo, IncomingContactObjectList, FirstQueuedContactObjectList, UseFilterOnFirstQueue`
Visualisierung	

Tabelle 2.25: Customer: Performance

Name	Customer: Not in SLA daily
Stored Procedures	`Customer_ServiceCallsNotInSla`
Beschreibung	Listet die eingegangenen SIP-Anrufe, deren Anrufzeitpunkt und Queue-Time für ein bestimmtes Datum auf und gibt an, welcher Agent den Anruf als erstes entgegengenommen hat. Optional ist der Parameter "`SipFrom`" mit welchem nur Adressen die mit diesem übereinstimmen, berücksichtigt werden. Entspricht weitgehend dem Report "Customer: Not in SLA daily".
Tabellen	`ContactObject, UnifiedSession, Agent`
Parameter	`Offset, UseDaylightSavings, Day, IncomingContactObjectList, FirstQueuedContactObjectList, UseFilterOnFirstQueuedContactObject, OrganizationUnit (optional), SipFrom (optional)`
Visualisierung	

Incoming Service Name	First Queued Service Name	Start Time	Queue Time	Initial Sip From	First Agent Connected
Luware Solutions	Luware Solutions	08:22	15.3	[redacted]@luware.com	[redacted]@luware.com
Luware Support	Luware Support	08:10	28.7	[redacted]@luware.net	[redacted]@luware.com

Tabelle 2.26: Customer: Not in SLA daily

Name	Customer: Lost Call Details
Stored Procedures	`Customer_ServiceLostCalls`
Beschreibung	Listet die verlorengegangenen (lost) SIP-Anrufe für ein bestimmtes Datum auf, deren IVR- und Queue-Time. Entspricht weitgehend dem Report "Service: Lost Call Details". Es gibt aber eine leicht andere where-Bedingung.
Tabellen	`UnifiedSession`
Parameter	`Offset, UseDaylightSavings, Date, IncomingContactObject, FirstQueuedContactObject`
Visualisierung	

Start Time	Sip From	IVR Time	Queue Time
17:06	[redacted]	5s	0s
17:06		5s	29s

Tabelle 2.27: Customer: Lost Call Details

2.3.4 Nichtfunktionale Anforderungen

ID	Name	Beschreibung
NF1	Antwortzeit	Die Anzeige eines Reports im Reporting Portal soll weniger lange dauern als die Anzeige desselben Reports in den SQL Server Reporting Services.
NF2	Kompatibilität	Das Reporting Portal unterstützt alle gängigen Browser (Chrome, Firefox, Safari, Internet Explorer) in ihren aktuellen Versionen.
NF3	Zuverlässigkeit	Das Reporting Portal hat eine Uptime von mindestens 95%.
NF4	Korrektheit	Die Reports im Reporting Portal müssen korrekt sein, d.h. die exakt gleichen Resultatmengen liefern wie die bestehenden SQL Server Reporting Services-Reports.
NF5	Skalierbarkeit	Bei Bedarf muss das neue Reporting Portal einfach skaliert werden können. Mit Einsatz von Elasticsearch ist dies gegeben, da sehr einfach neue Clusters und Indexe angelegt werden können.
NF6	Erweiterbarkeit	Das Reporting Portal soll innerhalb eines Personentages (nur Implementation) um einen neuen Report mit Basisfunktionalität erweitert werden können.
NF7	Benutzbarkeit	Das Reporting Portal soll für den User einfach zu benutzen sein. Dazu sollen Diagramme und deren Beschriftungen möglichst gleich wie in den bestehenden SQL Server Reporting Services-Reports daherkommen, wobei gewisse Verbesserungsvorschläge willkommen sind.
NF8	Sicherheit	Das Reporting Portal kontrolliert den Zugriff auf geschützte Ressourcen.

Tabelle 2.28: Nichtfunktionale Anforderungen

Kapitel 3

Design

3.1 Elasticsearch Datenbank

3.1.1 Reporting Schema Luware

Die Daten der folgenden Tabellen des bestehenden Reporting Schemas müssen in die Elasticsearch Datenbank übernommen werden, damit alle SQL Server Reporting Services-Reports auch im neuen System abgebildet werden können (in Klammern die Anzahl zum Zeitpunkt der Erstellung dieses Dokumentes vorhandenen Rows):

- OrganizationUnit (13)
- ContactObject (36)
- UnifiedSession (145'734)
- Session (145'821)
- Agent (61)
- SessionAgent (182'072)
- AgentProfile (8)
- AgentActivity (3'863'616)
- AgentNotReadyReason (576'452)
- NotReadyReason (5)
- AgentStatePeriod (672'305)
- AgentStatePerHour (824'864)

Es folgt das erstellte ER-Diagramm der oben aufgeführten Tabellen (siehe Abbildung 3.1). Darin sind die Beziehungen unter den verschiedenen Tabellen ersichtlich, welche für die Abbildung der Reports im neuen Reporting Portal ebenfalls nachvollzogen werden müssen.

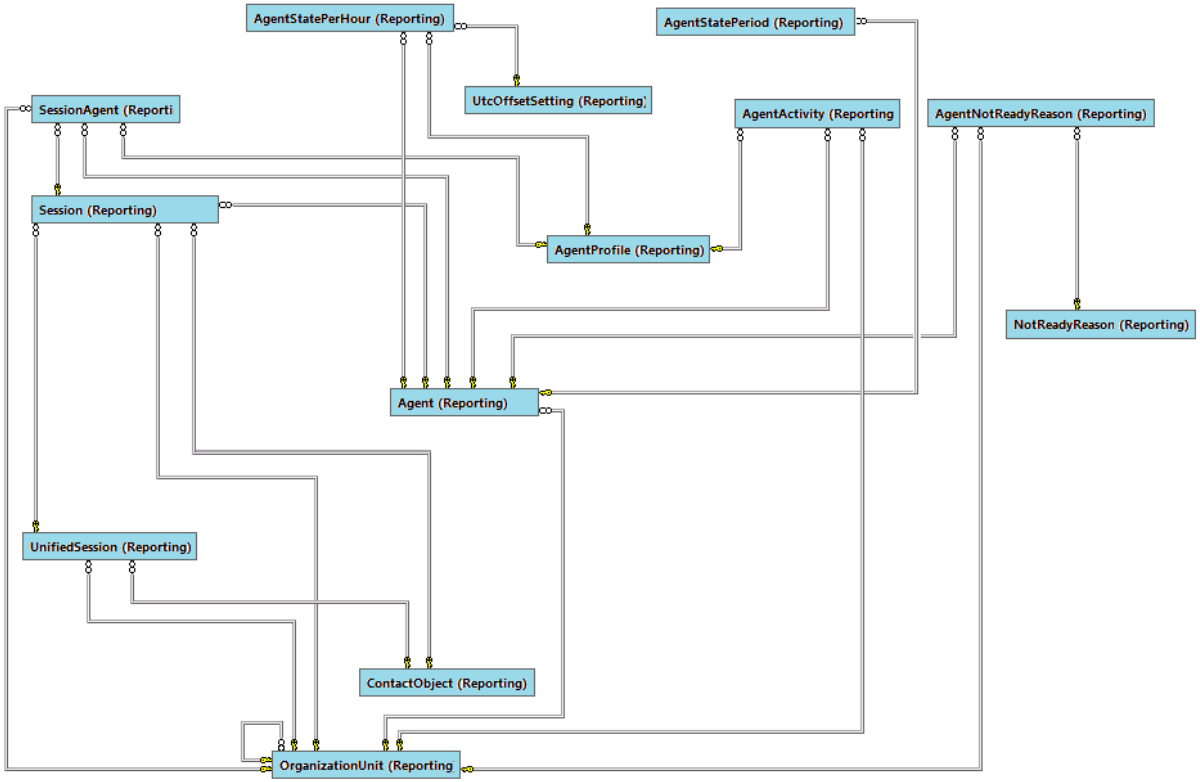


Abbildung 3.1: ER-Diagramm Reporting Schema

Ziel ist es, die Datenbestände dieser Tabellen ins Elasticsearch zu übernehmen und in geeigneter Form abzulegen (siehe Kapitel 3.1.2).

3.1.2 Abbildung in Elasticsearch

Elasticsearch speichert die Daten im NoSQL-Format. Da dieses Format keine Relationen abbildet, müssen diese anders nachvollzogen werden können. Wir haben dazu zwei Varianten kombiniert:

1. Relationen werden in der Applikation abgebildet.
2. Die Daten werden denormalisiert abgespeichert.

Abbildung Relationen in der Applikation

Die notwendigen Relationen (Beziehungen) der Tabellen untereinander werden in der Applikationslogik wo nötig nachvollzogen. In einem relationalen Datenbanksystem sind Relationen mittels Fremdschlüssel-Beziehungen abgebildet. Diese Fremdschlüssel auf den entsprechenden Child-Tabellen werden ins Elasticsearch importiert (siehe Kapitel 3.1.3) und sind somit auch dort vorhanden. Über diese können die Relationen applikatorisch hergestellt werden. Dazu sind allerdings mehrere Aufrufe über die RESTful-API von Elasticsearch notwendig: GET-Request auf die gewünschte Parent-Tabelle und mit der Resultatmenge (je in der Resultatmenge enthaltenem Primärschlüssel) ein weiterer GET-Request auf die entsprechende(n) Child-Tabelle(n). Bei verschachtelten Parent-Child-Beziehungen sind so entsprechend viele GET-Requests notwendig.

Denormalisierung

Die Tabellen `"UnifiedSession"`, `"Session"` und `"SessionAgent"` kommen in den meisten SQL Server Reporting Services-Reports vor und werden mittels Joins gelesen. Um unnötig viele separate und komplexe GET-Requests zu verhindern, werden diese drei Tabellen deshalb denormalisiert im Elasticsearch abgebildet. Dies wird beim Import (siehe Kapitel 3.1.3) mittels SQL-Join der Tabellen realisiert. Es sind somit im Elasticsearch die zwei denormalisierten document_types `"Session_UnifiedSession"` und `"SessionAgent_Session_UnifiedSession"` entstanden. Sie beinhalten alle denormalisierten Attribute der zwei, respektive drei Tabellen, die sie abbilden. Für die Abbildung einiger Reports im Bereich Agent ist zudem eine Denormalisierung der Tabellen `"Agent"`, `"AgentStatePerHour"` und `"AgentStatePeriod"` notwendig. Die entsprechend denormalisierten document_types im Elasticsearch heissen `"AgentStatePerHour_Agent"` und `"Agent-StatePeriod_Agent"`.

3.1.3 Import Datenbestand via

Es gibt drei verschiedene Konfigurationen wie die Daten der bestehenden Luware SQL-Datenbank mittels Logstash ins Elasticsearch übernommen werden:

1. "`select * from TableName`", einfachste Variante, mit der alle Attribute 1:1 ins Elasticsearch übernommen werden (Normalfall).

2. "`select Column1, Column2, ...,`
   `ColumnN from TableName`", ist bei manchen Tabellen von Nöten, da bestimmte Attribute vom SQL-Datentyp "`time`" im Elasticsearch nicht sauber abgebildet werden können und diese somit nicht importiert werden. Die fehlende Information lässt sich aber aus ebenfalls importierten Attributen vom Datentyp "`datetime2`" auslesen.

3. "`select Tab1.Column1, Tab1.Column2, Tab2.Column1, ..., TabN.ColumnN`
   `from TableName1 join TableName2 join ... join TableNameN `", wird bei den zu denormaliserenden Tabellen verwendet (siehe Kapitel 3.1.2)

Bei den Konfigurationen 1 und 2 entspricht der Name der Tabelle (doc_type) im Elasticsearch demjenigen aus dem Reporting Schema der Luware AG. Bei der Konfiguration 3 werden die gejointen Tabellen mittels Unterstrich als zusammengesetzter Name abgebildet.

Kennzahlen zum Import

Die Kennzahlen sind ungefähre Angaben und beruhen auf den Daten des Luware-Reporting-Schemas und deren für die SQL Server Reporting Services-Reports benötigten Tabellen. Zudem läuft der SQL-Server (Reporting-Schema) und Elasticsearch auf dem gleichen HSR-Server.

- Importierte Tabellen: **12**
- Importierte Records: **6.5 Millionen** (siehe auch Kapitel 3.1.1)
- Importierte Datenmenge: **7 GB**
- Dauer Komplett-Import: **85 Minuten** . Für mögliche Performanceverbesserungen siehe Kapitel 7

3.2 Application Architecture

In diesem Abschnitt werden die verschiedenen Ebenen und Komponenten der Architektur beschrieben. Die zu verwendende Technologie (AngularJS) wurde in Rücksprache mit dem Auftraggeber gewählt. Das Endprodukt wird dann als Reporting Portal in die bestehende Umgebung integriert.

3.2.1 Architektur Layer

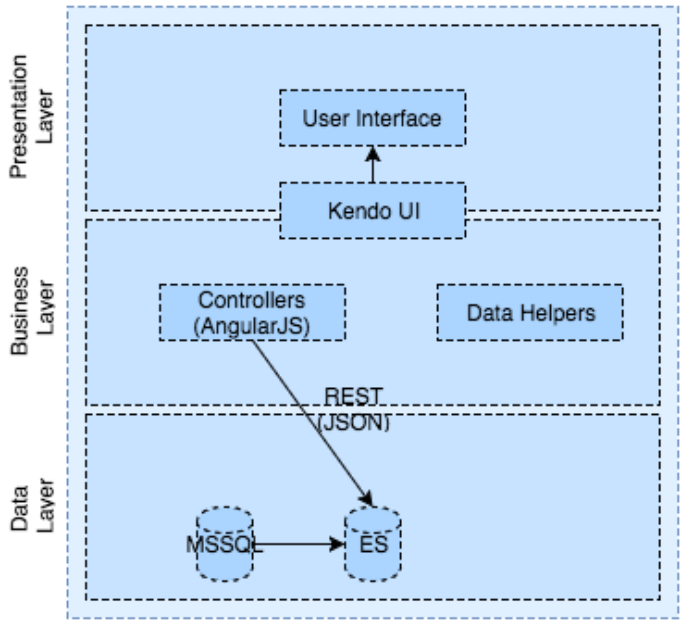


Abbildung 3.2: Layer der Architektur

Presentation Layer

Der Presentation Layer beinhaltet die benutzerorientierte Funktionalität, welche verantwortlich ist für die Interaktion mit der Applikation. Mit Hilfe von Kendo UI wird die Brücke zum Business Layer geschlagen. Die Darstellung von Reports wird von Kendo UI übernommen und dadurch wird die cross-browser Kompatibilität gewährleistet.

Business Layer

Der Business Layer beinhaltet die Hauptlogik der Applikation und kapselt diese. Die Daten werden über REST vom Data Layer bezogen und an die einzelnen Controller als JSON weitergegeben. Das erhaltene JSON muss analysiert und je nach Report anders aggregiert werden. Nach dem Bearbeiten der Server Antwort kann diese ans Kendo UI weitergereicht werden.

Data Layer

Im Data Layer findet der Import von MSSQL-Server ins Elasticsearch statt, welches wiederum die Daten als JSON für den Business Layer bereitstellt. Die Speicherung der Daten im Elasticsearch erfolgt im NoSQL-Format (JSON).

Kapitel 4

Realisierung

4.1 Reporting heute und in naher Zukunft

In den folgenden Absatz wird beschrieben, wie die Luware AG zur Zeit die Reports generiert und welche Komponenten involviert sind. Des weiteren wird auch beschrieben, wie die Umsetzung in Zukunft aussehen soll.

4.1.1 Reporting SSRS (Heute)

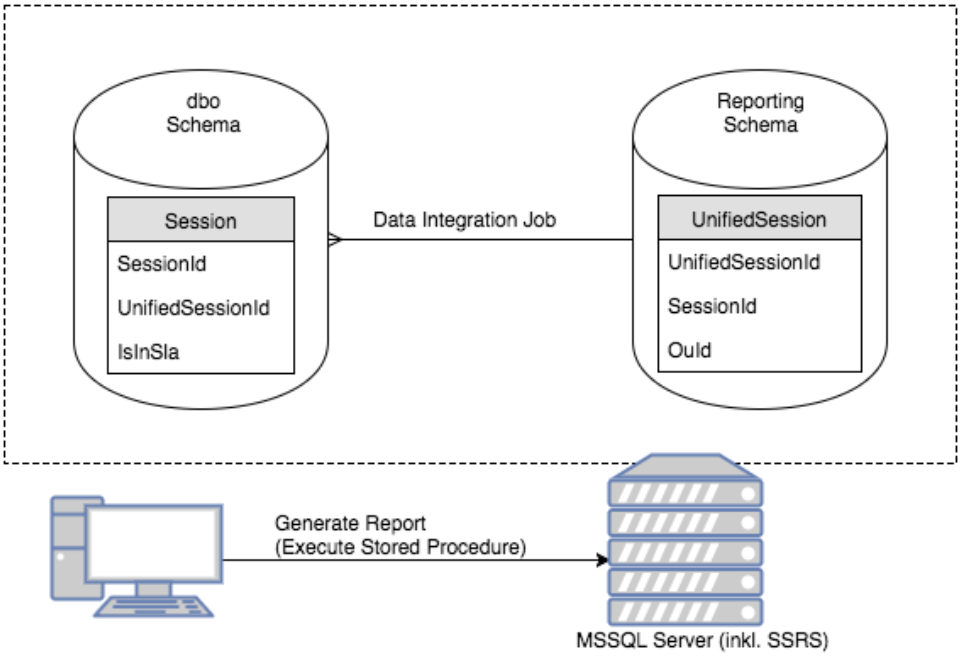


Abbildung 4.1: Integrations Job

Um mit den Daten arbeiten zu können, müssen diese erst bereinigt werden. Zum Beispiel werden einzelne Sessions zu einer UnifiedSession zusammengeführt. Auch werden allfällige Serverunterbrüche oder sonstige Errors richtig markiert. Der Integrationsjob überführt die Daten aus dem dbo-Schema ins Reporting-Schema und aggregiert Daten. Im Reporting-Schema sind die neuen aggregierten Daten vorhanden, jedoch nach wie vor auch die alten Tabellen aus dem dbo-Schema. Diese werden z.B. benötigt um AgentIds in einen Namen aufzulösen. Durch den Integrationsjob wird gewährleistet, dass aus den Daten fehlerfreie Reports generiert werden können.

SSRS Views

Im Moment werden die Reports zur Laufzeit generiert. Der Benutzer wählt einen gewünschten Report und die benötigen Parameter aus und lässt sich dann diesen Report generieren. Im Hintergrund wird ein Stored Procedure aufgerufen und auf der SQL-Datenbank aus dem Reporting-Schema werden die benötigen Daten aggregiert. Je nach Datenzeitraum und Datenumfang kann dies einige Sekunden dauern. Es wird zwischen drei verschiedenen Views unterschieden.

Service View

Die Service View basiert auf der "Reporting.Session"-Tabelle und ist ideal um den Status eines spezifischen Services zu analysieren. Das ermöglicht dem Benutzer Probleme zu identifizieren und Fehler innerhalb eines Services zu erkennen.

Customer View

Die Customer View zeigt die Sicht von Seiten des Kunden und basiert auf der Tabelle "Reporting.UnifiedSession". Diese Tabelle beinhaltet die aggregierten Daten der "dbo.Session"- und "dbo.SessionAgent"-Tabelle, welche einem Kunden zugewiesen werden können.

Agent View

Die Agent View fokussiert sich auf die Hauptaufgaben von jedem Agent in LUCS.

4.1.2 Reporting Portal (Zukunft)

Das Ziel der Arbeit ist es, einen Prototypen für ein neues Reporting Portal zu schaffen. Diese neue Applikation soll Elasticsearch als Datenhaltung nutzen. Die Rohdaten werden aus der SQL-Datenbank entnommen, integriert und danach ins Elasticsearch eingespiesen.

4.2 Technische Umsetzung

Im nachfolgenden Abschnitt wird näher auf die technische Umsetzung eingegangen. Anhand eines Beispiels wird aufgezeigt was involviert ist und wo Probleme entstehen können.

4.2.1 Anforderungen

Einige der Anforderungskriterien sind, dass die angezeigten Daten korrekt sein müssen. Die Benutzbarkeit muss einfach und intuitiv sein und vor allem muss die Antwortzeit kürzer als bei der momentanen Lösung sein. Die gesamte Anforderungsliste an den Prototyp kann dem Kapitel 2.3.4 entnommen werden.

4.2.2 Herausforderungen

Die grösste Herausforderung bei der Entwicklung des Prototyps war, der alten Lösung gerecht zu werden, aber auch die neuen Anforderungen einfliessen zu lassen. Die Darstellung der Daten als Bar- oder Pie-Chart war am Anfang ein grosses Problem im Kibana und konnte durch Kendo UI einfacher umgesetzt werden.

4.2.3 Umsetzung Reports

Wie im Kapitel 2.2 beschrieben, fiel der Entscheid für das Visualisierungs-Toolkit zugunsten von Kendo UI aus. Die Einarbeitung gestaltete sich jedoch schwieriger als geplant. Da die API-Dokumentation recht umfangreich und die Adaption etwas komplexer war als ursprünglich gedacht, war der Zeitaufwand höher als bemessen.

Die Daten werden clientseitig durch ein generiertes Query angefragt und von Elasticsearch geliefert. Danach werden die Daten an ein Kendo Chart übergeben und dementsprechend visualisiert. Der erstmalige Durchstich durch alle Ebenen, also Daten vom MSSQL-Server ins Elasticsearch importieren, Query korrekt erstellen und die Visualisierung, hat einiges an Zeit gekostet, konnte dann aber für die noch anstehenden Reports weiterverwendet werden.

4.3 Ordnerstruktur Applikation

Die Ordnerstruktur ist wie folgt aufgebaut:

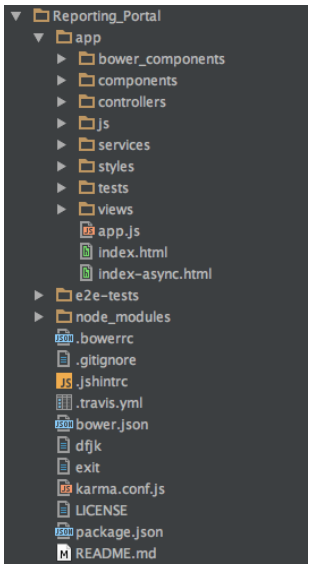
Der Unterordner **app** beinhaltet die wichtigsten Komponenten der Applikation, unter anderem die einzelnen Controller für die Reports im **controllers** Ordner, aber auch die nötigen stylesheets und HTML-files.	

Tabelle 4.1: Ordnerstruktur

4.4 Umsetzung eines Reports

Nachfolgend wird genauer auf den Report "Service Performance per hour" eingegangen und anhand von Code-Snippets erklärt, wie die Umsetzung für das neue Reporting Portal stattgefunden hat.

4.4.1 Analyse des SSRS Reports

Um ein grundlegendes Verständnis für den Report zu erhalten, wurde der bestehende SQL Server Reporting Services Report genauer untersucht. Folgende Fragen wurden gestellt: Wie sieht das gewünschte Diagramm aus? Welche Parameter werden benötigt? Wie lange dauert die Generierung? Die Abbildung 4.2 zeigt den Report "Service Performance per hour" und seine nötigen Parameter.

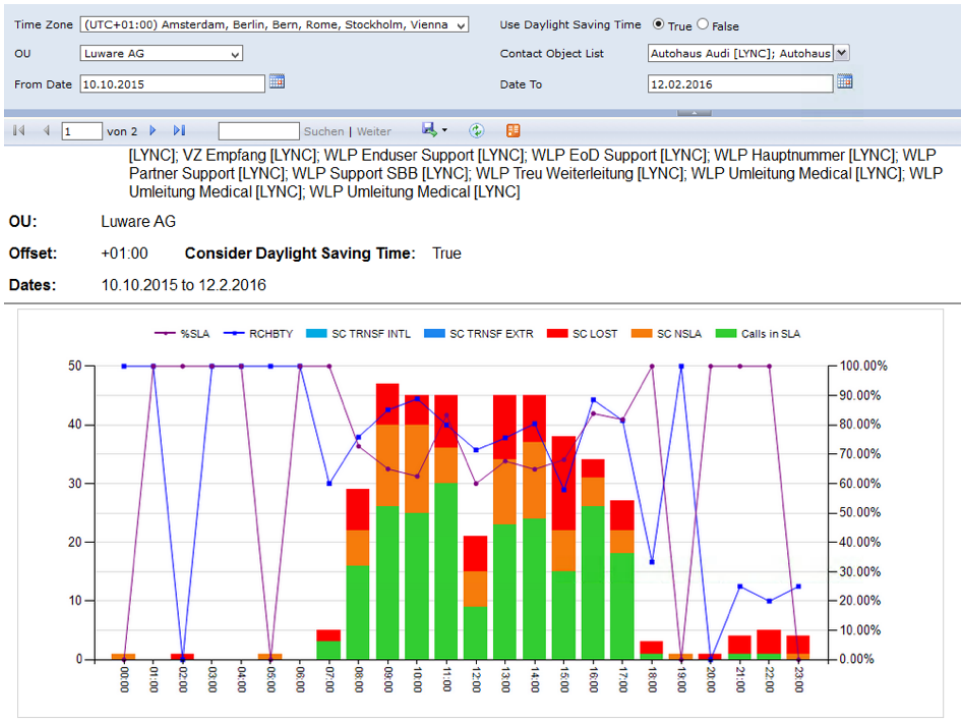


Abbildung 4.2: SSRS Report: Service Performance per hour

4.4.2 Template für die neuen Reports

Aus der Abbildung 4.2 lässt sich gut erkennen, dass eine Trennung zwischen Parameter Menu und Visualisierung gewählt wurde. Dies wurde übernommen und in AngularJS abgebildet.

Das Grundlayout bezüglich Unterteilung von Headerbar und Content wurde uns von dem Industriepartner vorgegeben, da dies bereits so in Verwendung ist.

customer_performance_weekday.html
mainPage.html
menubar.html
propertiesTemplate.html
service_lost_call_details.html
service_not_in_sla_daily.html
service_overview_daily.html

```
<div ng-controller="PropCtrl">
    <div style=" width: 100%;
                 padding-bottom:5px;
                 display:block;">
        <div>
            <h1 id="title">{{title}}</h1>
        </div>
        <div>
            <button id="runReport" ng-click="generateReport()">Run Report</button>
        </div>

    </div>
    <div>
        <div style="width: 24%; display:none" id="dateDayPickerDiv">
            <h4>Date</h4>
            <input id="dateDayPicker" style="width: 80%;" />
        </div>
        <div style="width: 24%; display:none" id="time15MinPickerDiv">
            <h4>Time From</h4>
            <input id="time15MinPicker" style="width: 80%;" />
        </div>
```

Tabelle 4.2: Auszug aus propertiesTemplate.html

Innerhalb dieses Dokuments (siehe Tabelle 4.2) wurden alle nötigen Properties für alle Reports erstellt und entsprechend konfiguriert. Durch Ein- und Ausblenden lassen sich diese individuell pro Report anpassen.

Auch gehört ein entsprechender Controller dazu, welcher im **controllers** Ordner zu finden ist. Dieser wird benötigt um die Business-Logik (siehe Code-Snippet 4.1) hinzuzufügen. Dadurch werden die einzelnen Felder mit Daten befüllt (z.B. die Organization Unit oder die verschiedenen Contact Objects).

```
$scope.organizationUnitList =  new kendo.data.DataSource({
    serverFiltering: true,
    transport: {
        read: function(options) {
            $.post({
                url: "http://152.96.56.34:40002/luware-index-01/
                    _search",
                dataType: "json",
                data: JSON.stringify(organizationUnitquery),
                success: function(result) {
                    options.success(result);
                },
                error: function(result) {
                    options.error(result);
                }
             });
        }
    },
    sort: { field: "name", dir: "desc" },
    schema: {
        data: function(response) {
            var hits = response.hits.hits;
            var orgUnitLst = [];
            for (var i = 0; i < hits.length; i++) {
                var innerSource = hits[i]._source;
                var orgUnit = {
                    ouid: innerSource.ouid,
                    parentouid: innerSource.parentouid,
                    name: innerSource.name,
                    isroot: innerSource.isroot
                };
                orgUnitLst.push(orgUnit);
            }
            return orgUnitLst;
        },
        total: function (data){
            return data.length;
        }
    }
});
```

Code-Snippet 4.1: Auszug aus properties.js

4.4.3 Erstellen der nötigen Files

Damit ein neuer Report hinzugefügt werden kann, muss jeweils ein neuer Controller und eine neue View erstellt werden. Beim "Service Performance per hour" sieht das wie folgt aus:

```
<div>
  <div ng-include="'views/propertiesTemplate.html'"></div>
  <div id="chartingArea"></div>
</div>
```

Code-Snippet 4.2: Auszug aus service_performance_hour.html

```
angular.module('Reporter.services_performance_hour', ['kendo.directives'])
    .controller('ServicePerformanceHourCtrl', ['$scope',
        function($scope) {
        $scope.title = "Service Performance per hour";
        $scope.generateReport = function() {
            var setMonthCorrect = function(monthAsString){...};
            var getDateFromString = function(dateString, start){...};
            var startString = $("#dateDayFromPicker").val();
            var endString = $("#dateDayToPicker").val();
            var startDateTimeUTC = getDateFromString(startString, true);
            var endDateTimeUTC = getDateFromString(endString, false);
            var contactObjectIds = $("#conObjMultiSelect").data("kendoMultiSelect").value();
            var isEarlyLost = $("#trueEarlyLost").prop('checked');
            var concatContactObjectIds = function (arr) {...};
            var reportQuery = {
                "_source": ["unifiedsessionid", "sessionid", "isinsla", "ishandled", "islost", "istransferredexternalbymachine", "isearlylost", "isdirtysession", "sessiontransfertype", "startdatetime", "sessionid"],
                "size": 9999,
                "query": {
                    "query_string": {
                        "query": "_type:Session_UnifiedSession AND (" + concatContactObjectIds(contactObjectIds) + ") AND isdirtysession:0 sessiontype:SERVICE AND startdatetime:[" + startDateTimeUTC + " TO " + endDateTimeUTC + "]"
                    }
                }
            };
```

Code-Snippet 4.3: Auszug aus service_overview_performance_hour.js

4.4.4 Anpassungen an bestehenden Files

Die Erstellung neuer Files reicht so noch nicht, es müssen auch noch bestehende Files angepasst werden: *index.html*, *app.js*, *menubar.html* und *properties.js*.

Im *index.html* muss das neu erstellte File durch einen <script>-Tag hinzugefügt werden, dadurch wird sichergestellt, dass beim Starten der Applikation das nötige Javascript File auch geladen wird.

```
<script src="app.js"></script>
<script src="services/utils.js"></script>
<script src="controllers/properties.js"></script>
<script src="controllers/mainPage.js"></script>
<script src="controllers/menubar.js"></script>
<script src="controllers/service_lost_call_details.js">
</script>
<script src="controllers/service_overview_daily.js"></script>
<script src="controllers/service_overview_weekly.js"></script>
<script src="controllers/service_overview_monthly.js"></script>
<script
src="controllers/service_overview_performance_daily.js"></
    script>
<script
src="controllers/service_overview_performance_15min.js"></
    script>
<script src="controllers/service_overview_performance_hour.js">
</script>
<script
src="controllers/service_overview_performance_month.js"></
    script>
```

Code-Snippet 4.4: Auszug aus index.html

Im *app.js* (siehe Code-Snippet 4.5) muss das File den Dependencies hinzugefügt werden. Auch muss eine weitere Route für die Applikation hinzugefügt werden.

```
'use strict';
angular.module('Reporter', [
    'ngRoute',
    'Reporter.properties',
    'Reporter.mainPage',
    'Reporter.menubar',
    'Reporter.utils',
    'Reporter.services_lost_call_details',
    'Reporter.services_overview_daily',
    'Reporter.services_overview_weekly',
    'Reporter.services_performance_daily',
    'Reporter.services_performance_15min',
    'Reporter.services_performance_hour',
    'Reporter.services_performance_month',
    'Reporter.services_performance_weekday',
```

Code-Snippet 4.5: Auszug aus app.js

Die Routen müssen im *$routeProvider* von AngularJS definiert werden. Dort kann der gewünschte URL-Pfad angegeben werden und mit dem nötigen Template-URL sowie dem Controller ausgestattet werden.

```
.config(['$routeProvider', function($routeProvider) {
  $routeProvider.when('/mainPage', {
      templateUrl: 'views/mainPage.html',
      controller: 'MainPageCtrl'
    });
    $routeProvider.when('/serviceOverviewDaily', {
      templateUrl: 'views/service_overview_daily.html',
      controller: 'ServiceOverviewDailyCtrl'
    });
    $routeProvider.when('/servicePerformance15min', {
      templateUrl: 'views/service_performance_15min.html',
      controller: 'ServicePerformance15minCtrl'
    });
    $routeProvider.when('/servicePerformanceHour', {
      templateUrl: 'views/service_performance_hour.html',
      controller: 'ServicePerformanceHourCtrl'
    });
```

Code-Snippet 4.6: Auszug aus app.js

Der URL-Pfad zum neu erstellten Report muss nun noch im Menu (siehe Code-Snippet 4.7 *menubar.html*) nachgetragen werden.

```
<div id="pnlMenuSlide" class="menuSlide" ng-controller="
    menuCtrl" >
    <header class="clearfix">
        <ul class="left">
            <li id="btnMenu" class="menuBtn mainMenu">
                <a k-ng-click="$scope.toggleMenuBar()">
                    <span class="lwicon-menu fs16"></span><span
                        class="mobileHidden">Reporting</span>
                </a>
                <div class="menuPopup left">
                    <div id="pnlMenuPopup">
                        <ul>
                            <span style="color: #000;">Agent</
                                span>
                            <li>
                              <a id="agentCCStaffingDaily" href
                                  ="#/agentCCStaffingDaily">CC
                                  Staffing Daily</a>
                            </li>
                            <li>
                              <a id="servicePerformanceHour"
                                  href="#/
                                  servicePerformanceHour">
                                  Performance Hour</a>
                            </li>
```

Code-Snippet 4.7: Auszug aus menubar.html

Damit beim Report nicht immer alle Properties die View überladen, werden die nötigen eingeblendet und alle anderen sind durch das "CSS-Visibility-Attribut: hidden" verborgen. Durch einen *switch-case* im *properties.js* wird anhand des Titels des Reports unterschieden, was eingeblendet werden muss.

```
1  switch($scope.title){
2      case 'Service Overview daily':
3          getOUDiv().css("display", "inline-block");
4          getDateDayPickerDiv().css("display", "inline-block");
5          getContactObjectMultiListDiv().css("display", "inline-
               block");
6          getShowEarlyLostDiv().css("display", "inline-block");
7          break;
8      case 'Service Overview weekly':
9          getOUDiv().css("display", "inline-block");
10         getDateDayPickerDiv().css("display", "inline-block");
11         getContactObjectMultiListDiv().css("display", "inline-
               block");
12         getShowEarlyLostDiv().css("display", "inline-block");
13         break;
14     case 'Service Performance per hour':
15       getOUDiv().css("display", "inline-block");
16         getDateDayFromPickerDiv().css("display", "inline-block"
               );
17         getDateDayToPickerDiv().css("display", "inline-block");
18         getContactObjectMultiListDiv().css("display", "inline-
               block");
19         break;
```

Code-Snippet 4.8: Auszug aus properties.js

4.4.5 Service Performance per hour

Der Report "Service Performance per hour" beruht hauptsächlich auf den Tabellen "Reporting.Session" und "Reporting.UnifiedSession". Diese wurden während dem Datenimport ins Elasticsearch miteinander verknüpft. Dies vereinfacht den Zugriff auf die nötigen Daten clientseitig. Untenstehend ist ersichtlich wie das Query anhand der ausgewählten Parameter clientseitig erstellt wird.

```
1 var reportQuery = {
2     "_source": ["unifiedsessionid", "sessionid", "isinsla", "
          ishandled", "islost", "istransferredexternalbymachine",
           "isearlylost", "isdirtysession", "sessiontransfertype"
          , "startdatetime", "sessionid"],
3     "size": 9999,
4     "query": {
5         "query_string": {
6             "query": "_type:Session_UnifiedSession AND (" +
                  concatContactObjectIds(contactObjectIds) + ")
                  AND isdirtysession:0 sessiontype:SERVICE AND
                  startdatetime:[" + startDateTimeUTC + " TO " +
                  endDateTimeUTC + "]"
7         }
8     }
9 };
```

Code-Snippet 4.9: Auszug aus service_overview_performance_hour.js

Der Benutzer wählt die gewünschten Parameter aus (siehe Code-Snippet 4.3) und diese werden dann ins Report-Query eingefügt. Nach Auswahl der Organization Unit (OU) werden die Contact Objects eingegrenzt, da eine OU aus mehreren Contact Objects bestehen kann. Diese werden durch eine interne Funktion konkateniert und dem Query übergeben. Auch nötig für diesen Report ist eine Angabe eines Start- und Enddatums.

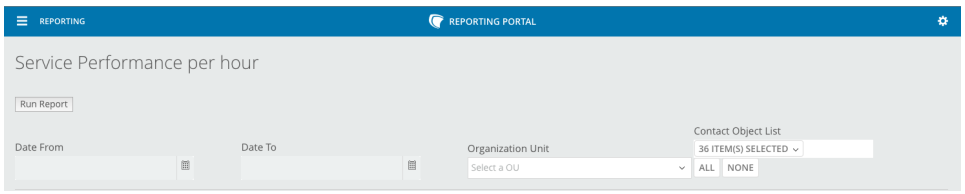


Abbildung 4.3: Auszug aus Reporting Portal

Das zusammengesetzte Query wird danach an eine Kendo DataSource übergeben, welches im Hintergrund jQuery.ajax() verwendet um mit dem Server zu kommunizieren. Die DataSource ist in dem Code-Snippet 4.10 ersichtlich. Nach Absetzen des Queries werden die Daten im JSON Format zurückgeliefert. Um das Weiterarbeiten zu vereinfachen, wird die Serverantwort noch in ein lesbareres Format geparst.

```
var dataSource = new kendo.data.DataSource({
    transport: {
        read: {
            url: "http://152.96.56.34:40002/luware-index-01/
                _search",
            dataType: "json",
            type: "POST"
        },
        parameterMap: function (options, operation) {
            return JSON.stringify(reportQuery);
        }
    },
    sort: { field: "slot", dir: "asc" },
    //group: {field: "slot"},
    aggregate: [{field: "sessionid", aggregate: "count"}],
    schema: {
        parse: function (response) {
            var hits = response.hits.hits;
            var lst = [];
            for (var i = 0; i < hits.length; i++) {
                var innerSource = hits[i]._source;
                var resultObj = {
                    slot: calculateHourSlot(innerSource.
                        startdatetime),
                    unifiedsessionid: innerSource.
                        unifiedsessionid,
                    sessionid: innerSource.sessionid,
                    isinslatrue: (innerSource.isinsla &&
                        innerSource.ishandled ? 1 : 0),
                    isinslafalse: (!innerSource.isinsla &&
                        innerSource.ishandled ? 1 : 0),
                    /* ... */
                    isearlylostfalse: (innerSource.isearlylost
                        ? 0 : 1),
                    istransferredinternalbymachine: ((
                        innerSource.sessiontransfertype === '
                        ICR_AGT' || innerSource.
                        sessiontransfertype === 'ICR_SRV') ? 1
                        : 0),
                    isearlylost: innerSource.isearlylost &&
                        isEarlyLost ? 1 : 0
                };
                lst.push(resultObj);
            }
            return lst;
        },
        total: function (data) {
            return data.length;
        }
    }
});
```

Code-Snippet 4.10: Auszug aus service_overview_performance_hour.js

Nach dem Parsing können die Daten an die Kendo Chart Funktion weitergegeben werden. Als Parameter werden u.A. der Titel, die *dataSource*, die *seriesDefaults* und die *series* mit gegeben. Innerhalb der einzelnen Parameter kann nochmals spezifiziert werden, wie die Darstellung erfolgen soll. So kann die Farbe und die gewünschte Aggregation innerhalb der *series* angegeben werden (siehe Code-Snippet 4.11).

```
$("#chartingArea").kendoChart({
    title: {
            text: $scope.title
    },
    dataSource: dataSource,
    seriesDefaults: {
            type: "column",
            stack: true
    },
    series: [
        {
            name: "SC SLA",
            color: "#32CC32",
            aggregate: "sum",
            field: "isinslatrue",
            categoryField: "slot",
            axis: "left"
        },
        {
            name: "SC NSLA",
            color: "#F57E0E",
            aggregate: "sum",
            field: "isinslafalse",
            categoryField: "slot",
            axis: "left"
        },
```

Code-Snippet 4.11: Auszug aus service_overview_performance_hour.js

Das Endresultat sieht nach erfolgreicher Eingabe der Parameter wie folgt aus:

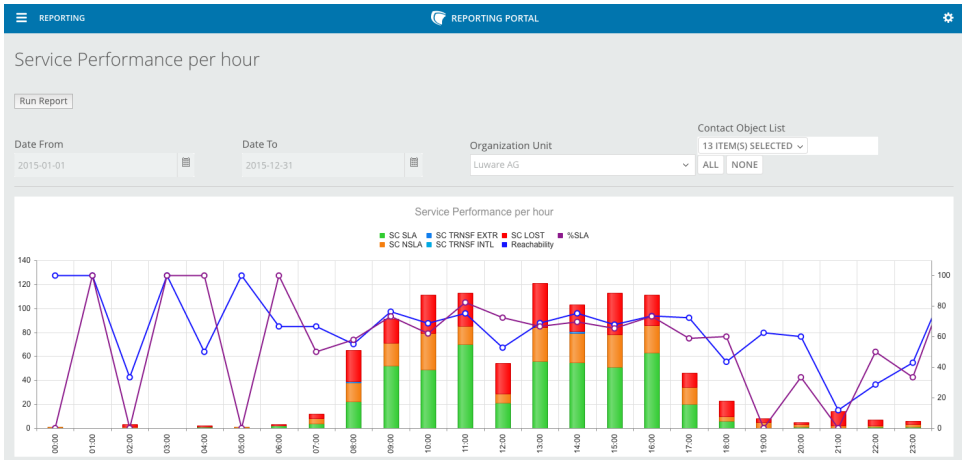


Abbildung 4.4: Reporting Portal Ansicht

4.4.6 UTC Problematik

Wie zu erkennen ist, wurde die Einstellungsmöglichkeit der *Time Zone* und der *Use Daylight Saving Time* nicht ins neue Reporting Portal übernommen. Nach Rücksprache mit dem Industriepartner wurde entschieden, dass die Zeitzonen-Einstellung des Browser als Berechnungsgrundlage genommen werden kann. Die Timestamps auf den einzelnen Tabellen auf dem Server werden in UTC Zeit als ISO-String abgelegt.

Die Daten-Parameter werden durch die .toISOString() Methode [Net16] auf die UTC Zeit zurück gerechnet. Dadurch wird sicher gestellt, dass die gewünschten Resultate erreicht werden, ohne dass die Zeitzone manuell eingestellt werden muss. Ein weiterer Vorteil dieser Methode ist auch die automatische Berücksichtigung der Sommer- und Winterzeit.

4.5 Performance Messung

Nachfolgend werden einige Messungen im SQL Server Reporting Services und im neuen Reporting Portal durchgeführt. Dies soll die Performance Steigerung aufzeigen. Es wird ein bestimmter Zeitraum eingegeben und eine OU (Organization Unit) ausgewählt. Die Messung erfolgt per Stoppuhr nach dem Drücken des "Bericht anzeigen" Buttons. Es handelt sich nicht um eine wissenschaftliche Messung, sondern ist rein empirisch.

Report	**Zeitraum**	**OU**	**Dauer SSRS**	**Dauer Reporting Portal**	**Steigerung**
Agent Service per month	Jan 2016	Luware AG	3 sec	1 sec	66%
Service Performance per hour	Jan 2015 bis Dez 2015	Luware AG	36 sec	2 sec	94%
Customer Performance monthly	Jan 2014 bis Dez 2015	Luware AG	73 sec	2 sec	97%
Service Performance per 15min	Jan 2015 bis Dez 2015	Luware AG	18 sec	1 sec	94%

Tabelle 4.3: Performance Messung

Aus dieser Tabelle ist gut ersichtlich, dass je grösser der Zeitraum gewählt wird, desto schneller arbeitet das neue Portal. Auch gut erkennbar ist, dass der Unterschied bei kleinen Zeitspannen nicht enorm ist.

4.6 Elasticsearch

Wie die Daten im Elasticsearch abgebildet werden, ist im Kapitel 3.1 aufgeführt. Die Installation und Konfiguration von Elasticsearch, ist in der "Installations- und Konfigurationsanleitung" beschrieben.

4.7 Logstash

Due Funktion des Datenimportes via Logstash ist im Kapitel 3.1.3 erläutert. Für die Selektion der Daten ab der SQL-Server Datenbank via JDBC-Treiber, wurde für jeden "doc_type" eine `.conf`-Datei erstellt, welche das SQL-Query und gewisse für Elasticsearch notwendige Parameter enthält. Ausserdem gibt es für jede dieser `.conf`-Datei eine entsprechende `.cmd`-Datei mit gleichem Namen. Um einen One-Click-Import des gesamten Datenbestandes durchführen zu können, gibt es die `.bat`-Datei "run_all". Bei Ausführung dieser werden alle oben erwähnten `.cmd`-Dateien gestartet, welche wiederum alle `.conf`-Dateien anstossen. Dadurch wird ein Komplettimport durchgeführt (siehe Abbildung 4.5). Die Installation und Konfiguration von Logstash ist in der "Installations- und Konfigurationsanleitung" ersichtlich.

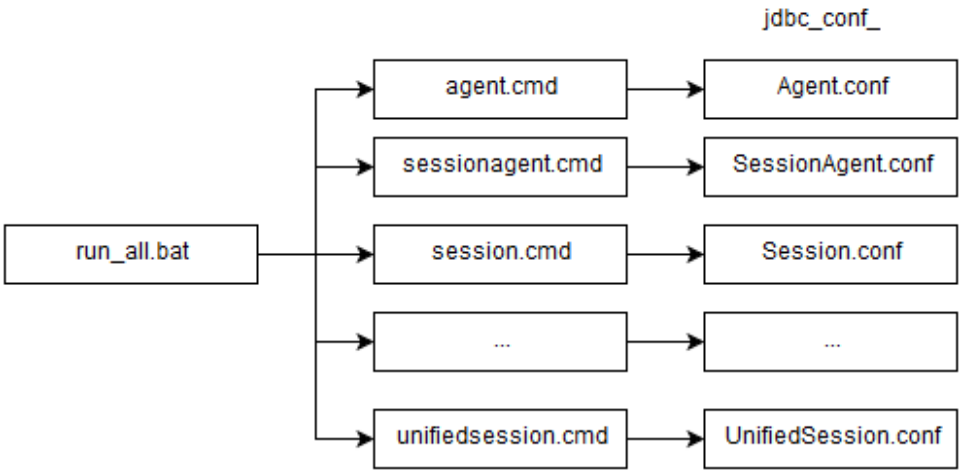


Abbildung 4.5: Konfiguration Komplettimport

Kapitel 5

Testing

Das neue Reporting Portal wird mittels Alt-Neu-Vergleich getestet. Alle Details zu den durchgeführten Tests und deren Ergebnisse sind im Anhang "Testprotokoll für Reporting Portal" aufgeführt. Hier folgt nur eine kurze Zusammenfassung der Testergebnisse und ein dadurch resultierendes Fazit.

5.1 Vorgehen

Beinahe alle neuen Reports (deren 27) konnten mit dem entsprechenden Pendant im alten Reporting Portal verglichen werden. Lediglich vier Reports wurden auf Wunsch des Industriepartners abgeändert, sodass dort kein Alt-Neu-Vergleich durchgeführt werden konnte.

5.2 Testergebnisse

Die Testergebnisse waren sehr erfreulich. Kein Testfall schlug fehl. Vier Testfälle waren nur teilweise erfolgreich (hatten also kleinere Abweichungen), die restlichen 23 Testfälle waren erfolgreich.

5.3 Defects

Die folgenden Defects wurden gefunden und sollten bei einer Weiterführung des Projektes behoben werden (Gewichtung: ein Stern = Low, zwei Sterne = Medium, drei Sterne = high):

Id	Beschreibung	Gewichtung
DL01	Sortierung anpassen	☆
DL02	Zeitformatierung anpassen	☆
DL03	Legende rechts entfernen	☆
DL04	Aggregierung überprüfen und korrigieren	☆
DM01	Letzter Tag auch einblenden	☆☆
DM02	Letzter Tag vom Monat bei 31 Tagen auch einblenden	☆☆
DM03	Die Performanz muss verbessert werden. Allerdings wird dieser Report wohl sowieso wieder entfernt, da er nicht benötigt wird	☆☆☆

Tabelle 5.1: Defects

5.4 Fazit

Das neue Reporting Portal hat eine hohe Qualität. Alle Reports konnten so nachgebildet werden, sodass sie den Resultaten des alten Reporting Portals entsprechen. Teilweise gibt es noch kleinere Fehler bei der Anzeige, welche aber für einen Prototypen nicht weiter ins Gewicht fallen.

Kapitel 6

Projektplanung

6.1 Sprintplanung

Das Vorgehensmodell orientiert sich an Scrum. Es erfolgt eine Aufteilung in acht Sprints. Zu Beginn des Projektes steht das *Setup*, welches noch keinem Sprint zugeordnet wird.

Sprint	**Dauer**	**Beschreibung**
Setup	1 Woche	Grobe Projektplanung, Aufbau der Projektinfrastruktur, diverse Vorarbeiten
Sprint I	2 Wochen	Produktevaluation anhand Bewertungskriterien, Entscheid für ein Produkt
Sprint II	2 Wochen	Installation des ausgewählten Produktes, Analyse Ist Zustand anhand Service-Ansicht, Ausarbeitung Prototyp (Durchstich) für Reports der Service-Ansicht
Sprint III	2 Wochen	Weiterführende Anforderungsanalyse, Festhalten der Anforderungen, Ausarbeitung Design
Sprint IV	2 Wochen	Beginn Umsetzung Reports des Bereichs "Service"
Sprint V	2 Wochen	Fertigstellung "Service"-Reports und Beginn Umsetzung Reports des Bereichs "Agent"
Sprint VI	2 Wochen	Fertigstellung "Agent"-Reports
Sprint VII	2 Wochen	Beginn Umsetzung und Fertigstellung Reports des Bereichs "Customer"
Sprint VIII	2 Wochen	Letzte Abschlussarbeiten, Finalisierung der Dokumentation

Tabelle 6.1: Projektablauf

Das in Abbildung 6.1 aufgeführte Gantt Diagramm zeigt den geplanten Projektverlauf mit dessen Sprints und Meilensteinen. Es gibt acht Sprints und an deren Ende je einen Meilenstein. Jeder Sprint dauert zwei Wochen, beginnt mit der Sprintplanung und endet mit der Retrospektive.

6.2 Gantt Diagramm

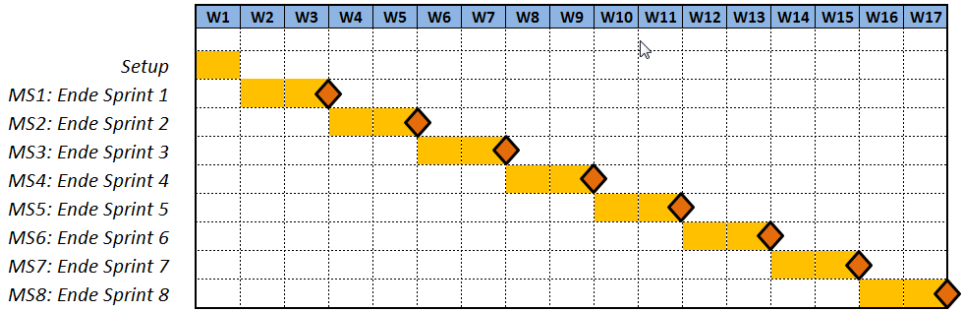


Abbildung 6.1: Gantt Diagramm

6.3 Rollen und Meetings

6.3.1 Rollen

Teammitglieder

Damit sind die beiden Studierenden und Verfasser dieser Arbeit gemeint.

Productowner (PO)

Die Rolle des Product Owner nimmt der Industriepartner ein.

Scrummaster

Im Team gibt es keinen designierten Scrummaster, der Industriepartner übernimmt diesen Teil ebenfalls.

6.3.2 Meetings

Alle Meetings finden in den Räumlichkeiten der Luware AG oder via Skype-Session statt.

Sprintplanung

Zu Beginn jedes Sprints wird die Planung für die nächsten zwei Wochen vorgenommen. Die Erstellung und Planung von User Stories ist Sache der Teammitglieder. Der PO gibt hilfreichen Input und leitet das Meeting.

Sprintdemo

Gegen Ende jedes Sprints werden die im Sprint umgesetzten Userstories dem PO vorgestellt. Der PO entscheidet, ob die Userstories die Akzeptanzkriterien erfüllen.

Sprintretro

Die Sprintretro schliesst den aktuellen Sprint ab. Dabei wird besprochen was gut war, was in Zukunft verbessert werden kann und welche Massnahmen dafür notwendig sind.

Standup

An den drei Tagen, an welchen die Teammitglieder für das Projekt arbeiten, findet jeweils ein maximal 15-minütiges Statusmeeting statt.

6.4 Tools

6.4.1 Projektverwaltung

Für die Projektabwicklung wird Team Foundation Services genutzt. Das Tool wird vom Industriepartner zur Verfügung gestellt. Sämtliche Userstories, Tasks und Arbeitsfortschritte werden darin getrackt. Eine Auflistung aller Sprints mit den dazugehörigen Userstories und Tasks ist im Anhang "Auszug aus TFS BA Reporting Portal" ersichtlich.

6.5 Risikoanalyse

Nr.	Titel	Beschreibung	max. Schaden[h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Ausfall eines Teammitglieds	Ausfälle aufgrund von Krankheit, Unfall etc.	50	10%	5.0	Knowhow ist auf beide Teammitglieder verteilt, beide können im Notfall einspringen	Teampartner erhöhen temporär ihr Pensum
R2	Ausfall des Servers	Elasticsearch läuft auf dem Server. Bei Ausfall kann dieses nicht mehr verwendet werden	10	10%	1.0	Installationsanleitung wird erstellt, somit einfaches Wiederaufsetzen möglich	Erneute Installation von Elasticsearch und neuer Datenimport von SQL-Server
R3	Fehlerhafte Architektur	Fehlerhafte Softwarearchitektur welche bedingt, dass sie noch einmal überarbeitet werden muss. Das Schadenspotenzial erhöht sich stetig mit Fortschritt des Projektes.	40	10%	4.0	Architektur wird sorgfältig ausgearbeitet, projektintern und vom Auftraggeber reviewed	Teammitglieder erhöhen temporär ihr Pensum um die Architektur zu korrigieren
R4	Volatiler Scope	Änderungen des Scopes während des Projektes	20	15%	3.0	Demo alle zwei Wochen um Feedback des Auftraggebers einzuholen	Verzicht auf andere Features müssen diskutiert werden
R5	Falsche Produktwahl	Das in der Produktevaluation gewählte Produkt kann Anforderungen des Auftraggebers nicht oder nur teilweise erfüllen	60	10%	5.0	Saubere und ausführliche Produktevaluation, Abnahme durch Auftraggeber	Wechsel auf Alternativprodukt, Teampartner erhöhen temporär ihr Pensum

Tabelle 6.2: Risikoanalyse

Farbe	Legende
	Nicht akzeptierbar, benötigt Reduktion des Risikos
	Akzeptierbar, müssen weiter beobachtet werden
	Akzeptierbar, keine weiteren Massnahmen nötig

Tabelle 6.3: Risikomatrix Farblegende

Kapitel 7

Zukunftsausblick und Optimierungen

Beim Design und der Umsetzung des Prototypen wurde grossen Wert auf eine zukünftige Ausbaufähigkeit gelegt. Dies war auch explizit der Wunsch des Auftraggebers. Das Kapitel soll bei einer Weiterführung des Projektes einen guten Einstiegspunkt bieten.

Untenstehend wird ein Ausblick in die Zukunft gegeben und es werden mögliche Optimierungen aufgelistet. Funktionen, welche im Umfang der Arbeit nicht umgesetzt werden konnten, werden auch aufgelistet und sollten bei allfälligen Weiterführung des Projektes unbedingt beachtet werden. Auch werden Defects des Prototypen angegeben, welche während der Laufzeit des Projektes nicht behoben werden konnten.

7.1 Fertigstellung Agent Reports

Es sind noch die folgenden vier Agents Reports ausstehend:

- AgentPerformancePerDay
- AgentPerformancePerDayByProfile
- AgentPerformancePerDayByService
- AgentPerformancePerMonth

Die für diese Reports verwendeten Daten sind alle im Elasticsearch vorhanden. Speziell für diese sehr komplexen Reports wurden die zwei denormaliserten doc_types AgentStatePerHour_Agent und AgentStatePeriod_Agent angelegt. Diese können als Datenquelle für die fehlenden Reports herangezogen werden.

Der Report AgentPerformancePerDayByProfile ist zudem schon weitgehend im Client implementiert. Lediglich zwei Spalten sind noch fehlerhaft. Diese sind mittels @todo im Code gekennzeichnet. Dieser Report kann als Vorlage für die anderen drei fehlenden Reports herangezogen werden.

7.2 Security

Zukünftig muss der Zugriff auf die im Elastiscearch gespeicherten Daten gewissen Sicherheitsstandards der Luware AG beziehungsweise deren Kunden entsprechen. Ob sie dabei auf eine eigens erstelle Lösung (z.B. via Proxyzugriff) oder das von Elastic zur Verfügung gestellte Plugin für Security ("Shield") einsetzt, liegt im Ermessensspielraum des Industriepartners.

7.3 Konfiguration Elastic Produkte

Das in der "Installations- und Konfigurationsaleitung" aufgeführte Setup von Elasticsearch, Logstash und Kibana ist für den Umfang dieser Arbeit ausreichend. Die Elasticsearch-Instanz und Kibana laufen dabei zurzeit auf einem von der HSR zur Verfügung gestelltem Server. Zukünftig müssen diese logischerweise auf einem anderen Server betrieben werden. Alle vorzunehmenden Einstellungen finden sich in der besagten Anleitung.

7.4 Skalierung Elasticsearch

Elasticsearch ist zurzeit so eingerichtet, dass es dem aktuellen Datenbestand des Luware Reporting Schemas ohne Probleme genügt. Allerdings muss vor einem produktiven Betrieb geprüft werden, ob allenfalls weitere Nodes benötigt werden. Aktuell läuft auf dem Cluster (`"luware-cluster"`) ein Node (`"luware-node-1"`). Für den produktiven Betrieb empfehlen sich hingegen drei Nodes, um eine Ausfallsicherheit zu haben (ein Master- und zwei Slave-Nodes). Alle `doc_types` befinden sich auf einem Index (`"luware-index-1"`), was auch für den produktiven Betrieb so bleiben sollte. Momentan ist der Index auf fünf Shards und eine Replica aufgeteilt (Standardeinstellung von Elasticsearch). Sollte ein Kunde massiv mehr Daten haben als die Luware AG, empfehlen sich allenfalls mehr Shards und Replicas pro Node. Hier ist es schwierig konkrete Zahlen zu definieren.

7.5 Logstash Import

Das aktuelle Reporting Schema der Luware AG führt nicht auf allen Tabellen einen Updatetimestamp. Aufgrund dessen ist es beim Import via Logstash nicht möglich nur diejenigen Rows ab dem Reporting Schema zu lesen, die auch aktualisiert oder neu hinzugefügt wurden. Somit müssten mit der aktuellen Lösung immer alle Rows der zu importierenden Tabellen mittels Logstash gelesen werden und erst Elasticsearch erkennt dann, dass eine entsprechende Row bereits unverändert im Index vorhanden ist. Das ist ein massiver Overhead. Um zukünftig nur mutierte und neue Rows zu lesen, muss das bestehende SQL-Reporting Schema der Luware AG auf jeder zu importierenden Tabelle um einen Updatetimestamp erweitert werden. Die Queries zum Import via Logstash können dann nur die Rows, die eine Bedingung der folgenden Art aufweisen, lesen: `where updateTimestamp > lastImportTimestamp`. Dadurch kann die Zeitdauer für einen Import (siehe Kapitel 3.1.3) auf einen Bruchteil reduziert werden.

7.6 Coderefactoring

Einige Javascript-Funktionen sind in verschiedenen Reports vorhanden und unterscheiden sich kaum oder gar nicht in ihrer Logik. Sie sollten in eine Util-Klasse ausgelagert werden um nicht das DRY-Principle (don't repeat yourself) zu verletzen. Die entsprechende Klasse ist bereits vorhanden und enthält als erstes Beispiel die Funktion "concatElements", welche ebenfalls in vielen Reports zum Einsatz kommt. Die folgenden Funktionen sollten in einem Refactoring ebenfalls in diese Util-Klasse aufgenommen werden:

- getDateFromString
- setMonthCorrect
- getMonday

7.7 Defectfixing

Die im Kapitel 5 aufgeführten Defects sind nach Abschluss dieser Arbeit bekannt und noch offen. Es wird empfohlen diese bei einer Weiterführung des Reporting Portals zu beheben.

Kapitel 8

Schlussfolgerung

8.1 Ergebnisse

8.1.1 Primäres Ziel

Das primäre Ziel wurde zu 90 Prozent erreicht. Von den 31 abzubildenden Reports konnten deren 27 umgesetzt werden. Es bleiben allerdings vier Reports des Bereiches “Agent” offen. Auch nach einem Extra-Effort war es den Teammitgliedern nicht möglich innert Abgabefrist dieser Arbeit die Reports abzubilden. Die Komplexität der besagten Reports ist weit höher als diejenige der anderen umgesetzten Reports. Die Situation wurde allerdings erkannt und sofort dem Industriepartner kommuniziert. Nach Rücksprache mit ihm wurde der Scope entsprechend leicht reduziert und der Fokus auf die Richtigkeit der restlichen Reports gelegt. Es wurde bereits viel Vorarbeit für die entsprechenden Reports geleistet, so dass die Grundlage für eine erfolgreiche Fertigstellung der fehlenden Reports nach Abschluss dieser Arbeit geschaffen ist. Die getroffenen Vorbereitungen und weiteren notwendigen Schritte für die Fertigstellung der fehlenden vier Reports sind im Kapitel 7 aufgeführt.

8.1.2 Sekundäres Ziel

Für das sekundäre und somit optionale Ziel konnten nur gewisse Vorarbeiten geleistet werden. So ist mit der Installation von Kibana bereits die Grundlage geschafft um zu einem späteren Zeitpunkt dynamische, vom Benutzer selbst zusammenstellbare Reports, via Kibana-Schnittstelle anzubieten. Wie Kibana zu installieren und konfigurieren ist, ist in der "Installations- und Konfigurationsanleitung" aufgeführt.

8.2 Bewertung

Das Ziel dieser Arbeit, einen Prototypen des neuen Reporting Portals zu entwickeln, ist gelungen. Der Industriepartner war sehr zufrieden mit dem Resultat. Diese Arbeit hat gezeigt, dass eine sehr performante und ausbaufähige Lösung, welche auch künftigen Herausforderungen gewachsen ist, erstellt wurde.

8.3 Empfehlung

Wie im Kapitel 7 erwähnt, muss das Reporting Portal noch um die fehlenden Reports erweitert werden. Auch im Bereich Refactoring von Code empfiehlt es sich noch einige Zeit zu investieren, bevor das neue Reporting Portal seinen Weg in Richtung Release bestreitet. Gerade im Bereich Sicherheit muss ebenfalls noch eine Lösung erarbeitet werden.

8.4 Ausblicke

Werden die beschriebenen offenen Punkte noch geschlossen, wird die Luware AG in Zukunft ein hervorragendes neues Reporting Portal im Einsatz haben, das höchsten Anforderungen an Datenmenge, Performance und optischen Standards gerecht wird. Damit sind die veraltet wirkenden und eher langsamen SQL Server Reporting Services-Reports Geschichte.

Erklärung zur Urheberschaft

Wir erklären hiermit, dass wir die vorliegende Arbeit ohne Hilfe Dritter angefertigt haben. Wir haben nur die Hilfsmittel benutzt, die wir angegeben haben. Gedanken, die wir aus fremden Quellen direkt oder indirekt übernommen haben, sind kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort/Datum: Rapperswil, 17. Juni 2016

Unterschrift:

Sandro Muggli Fabio Laib

Abbildungsverzeichnis

2.1 Trendanalyse: Solr vs. Elastic 10
2.2 Use Case Diagramm 17

3.1 ER-Diagramm Reporting Schema 36
3.2 Layer der Architektur 39

4.1 Integrations Job 40
4.2 SSRS Report: Service Performance per hour 44
4.3 Auszug aus Reporting Portal 50
4.4 Reporting Portal Ansicht 53
4.5 Konfiguration Komplettimport 55

6.1 Gantt Diagramm 59

Tabellenverzeichnis

2.1 Bewertungsraster 9
2.2 Bewertungsraster Solr 12
2.3 Bewertungsraster Elaticsearch 13
2.4 Auswertung Elaticsearch vs. Solr 14
2.5 Funktionale Anforderungen 17
2.6 UC1 18
2.7 UC1.1 18
2.8 UC1.2 18
2.9 UC1.3 18
2.10 UC2 19
2.11 UC2 19
2.12 Service: Overview 20
2.13 Service: Performance 21

2.14 Service: Performance per Opening Hour Type 22
2.15 Service: Not in SLA daily 22
2.16 Service: Lost Call Details 23
2.17 Agent: Staffing 24
2.18 Agent: Concurrent Agents 25
2.19 Agent: Not Ready Reasons 26
2.20 Agent: Customer Performance 15Min 27
2.21 Agent: Service Per Month 28
2.22 Agent: Performance per Day/Profile/Service 29
2.23 Agent: Performance per month/Week 30
2.24 Customer: Overview 31
2.25 Customer: Performance 32
2.26 Customer: Not in SLA daily 33
2.27 Customer: Lost Call Details 33
2.28 Nichtfunktionale Anforderungen 34

4.1 Ordnerstruktur 43
4.2 Auszug aus propertiesTemplate.html 44
4.3 Performance Messung 54

5.1 Defects 57

6.1 Projektablauf 58
6.2 Risikoanalyse 61
6.3 Risikomatrix Farblegende 62

Code-Snippets

4.1 Auszug aus properties.js 45
4.2 Auszug aus service_performance_hour.html 46
4.3 Auszug aus service_overview_performance_hour.js 46
4.4 Auszug aus index.html 47
4.5 Auszug aus app.js 47
4.6 Auszug aus app.js 48
4.7 Auszug aus menubar.html 48
4.8 Auszug aus properties.js 49
4.9 Auszug aus service_overview_performance_hour.js 50
4.10 Auszug aus service_overview_performance_hour.js 51
4.11 Auszug aus service_overview_performance_hour.js 52

Glossar

AngularJS

AngularJS ist ein clienseitiges Javascript-Webframework welches genutzt werden kann um Single-Page-Applikationen zu entwickeln.. 16, 38, 44, 48

Apache Lizenz

Die Apache-Lizenz ist eine durch die Free Software Foundation anerkannte Freie-Software-Lizenz der Apache Software Foundation, jedoch keine Copyleft-Lizenz. Prinzipiell beinhaltet sie: Man darf Software unter dieser Lizenz frei in jedem Umfeld verwenden, modifizieren und verteilen. Wenn man sie verteilt, muss eindeutig darauf hingewiesen werden, welche Software unter der Apache-Lizenz verwendet wurde und dass diese vom Lizenzgeber (name of copyright owner) stammt. Eine Kopie der Lizenz muss dem Paket beiliegen. Änderungen am Quellcode der unter der Apache-Lizenz stehenden Software müssen nicht zum Lizenzgeber zurückgeschickt werden. Eigene Software, die unter Apache-Lizenz stehende Software verwendet, muss nicht unter der Apache-Lizenz stehen. Die eigene Software darf nur dann Apache heißen, wenn eine schriftliche Genehmigung der Apache Foundation vorliegt[Wik16b].. 12, 13

Elasticsearch

Elasticsearch ist eine Suchmaschine und das Hauptprodukt von Elastic. Es speichert die Daten im JSON-Format und ermöglicht eine performante Suche im Umfeld von Big Data.. 5, 7, 10–16, 34, 35, 37–39, 41, 42, 50, 54, 55, 63–65

JSON

Die JavaScript Object Notation, ist ein kompaktes Datenformat in einer einfachen lesbaren Form zum Zweck des Datenaustausches zwischen Anwendungen. 10–13, 39, 50

Kendo UI

Kendo UI ist ein HTML Framework der Firma Telerik. Dieses ermöglicht HTML5 und Javascript Apps für alle Platformen zu entwickeln.. 5, 7, 15, 16, 39, 42

Kibana

Ein Plugin für Elasticsearch. Es ermöglicht die einfach Analyse und Visualisierung der im Elasticsearch gespeicherten Daten anhand verschiedener, frei konfigurierbarer Reports.. 12, 13, 15, 16, 19, 42, 64, 66

Logstash

Ein Plugin für Elasticsearch. Mit ihm können Daten von diversen Datenquellen (z.B. Files, RDMS, Logs etc) in Elasticsearch importiert werden.. 13–16, 38, 55, 64, 65

Lucene

Apache Lucene ist eine leistungsstarke, voll funktionsfähige Bibliothek für die Textsuche, welche komplett in Java geschrieben ist. Die Technologie ist nahezu für jede Anwendung geeignet, die Volltextsuche erfordert.. 10, 11, 73

LUCS

LUCS steht für Lean Unified Customer Services und ist ein Produkt der Firma Luware AG.. 41

RDBMS

Eine relationale Datenbank dient zur elektronischen Datenverwaltung in Computersystemen und beruht auf einem tabellenbasierten relationalen Datenbankmodell. Dieses wurde 1970 von Edgar F. Codd erstmals vorgeschlagen und ist bis heute trotz einiger Kritikpunkte ein etablierter Standard für Datenbanken. [Wik16c].. 9, 12

Shield

Ein Plugin für Elasticsearch. Ermöglicht den Schutz der im Elasticsearch gespeicherten Daten mittels Benutzername und Passwort.. 13, 64

Solr

Solr ist ein in Lucene enthaltenes Servlet für entsprechende Container.. 10, 12, 14

SQL Server Reporting Services

SQL Server Reporting Services (SSRS) ist ein Server-basiertes Berichtgenerierungssystem von Microsoft. Es kann zur Vorbereitung und produktiven Erzeugung verschiedener Arten interaktiver oder vorgefertigter Berichte verwendet werden. Die Administration erfolgt über ein Webinterface. Reporting Services bietet eine Webservice-basierte Schnittstelle an, die die Entwicklung eigener Berichtsanwendungen unterstützt.[Wik16a].. 5, 6, 14, 16, 19, 34, 35, 37, 38, 43, 54, 67

Literaturverzeichnis

[Net16] Mozilla Developer Network. *Date.prototype.toISOString()*, 2016 (accessed June 10, 2016). https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Date/toISOString.

[Wik16a] Wikipedia. *SQL Server Reporting Services*, 2015 (accessed March 28, 2016). https://de.wikipedia.org/wiki/SQL_Server_Reporting_Services.

[Wik16b] Wikipedia. *Apache Lizenz*, 2015 (accessed March 9, 2016). https://de.wikipedia.org/wiki/Apache-Lizenz.

[Wik16c] Wikipedia. *RDBMS*, 2016 (accessed June 13, 2016). https://de.wikipedia.org/wiki/Relationale_Datenbank.

[Wik16d] Wikipedia. *Apache Solr*, 2016 (accessed March 7, 2016). https://de.wikipedia.org/wiki/Apache_Lucene#Solr.

[Wik16e] Wikipedia. *Elasticsearch*, 2016 (accessed March 8, 2016). https://de.wikipedia.org/wiki/Elasticsearch.

Appendix

Analyse

a. Installations- und Konfigurationsanleitung für Elastic Produkte

Realisierung

a. Source Code

b. Logstash Config Files

c. Elasticsearch Config Files

d. Kibana Config Files

Testing

a. Testprotokoll für Reporting Portal

Projektplanung

a. Auszug aus TFS BA Reporting Portal

b. Protokolle

c. Zeiterfassung

d. Persönliche Berichte