Konzept zur automatischen Erstellung von kategorisierten Incidents

Klassifizierung *	Nicht klassifiziert
Dokumentstatus **	Abgeschlossen
Auftraggeber	Patrick Bregy, Porteous Chloe
Projektleiter	Giuseppe Scavetta
Projektname	Konzept zur automatischen Erstellung von kategorisierten Incidents
Projektabkürzung	OP-ALARMING
Projektnummer	-
Abnahme durch	Patrick Bregy
Kurzbeschreibung	Ein Technisches Konzept eines Frameworks zum Erstellen von Incidents und Versenden von E-Mails. Diese werden auf Basis von automatisch durchgeführten Überprüfungen ausgelöst.

* Nicht klassifiziert, Intern, Vertraulich
** In Arbeit, In Prüfung, Abgeschlossen

Inhaltsverzeichnis

Management Summary

Umsetzung eines Frameworks um den Aufwand des Operation-Teams zu vermindern.

Es geht um die Realisierung eines Technischen Konzepts, welches den Aufwand des Operation-Teams verkleinert und um die Freigabe für die Umsetzung dieser Konzeption.

Hintergrund

In unserer Datenbank Applikation gibt es keine zielgerichteten Überprüfungen, die automatisch Incidents im Ticketsystem anlegt. Es gibt eine generische Überprüfung, die bei jedem Job-Unterbruch ein Incident erzeugt. Leider heisst es nicht, dass jeder Job-Unterbuch automatisch ein Fehler ist. Deshalb hat das Operation-Team einen enormen Aufwand um Incidents zu bearbeiten, welche keinen bearbeitungsbedarf hätten. Dies schränkt die Kapazität vom Operation-Team ein und vermindert die Qualität der Ticketbearbeitung sowie auch die Qualität vom Betrieb unserer Datenbank Applikation.

Ziele und Rahmenbedingungen

Der Aufwand des Operation-Teams soll mit einem Überprüfungs-Framework verkleinert werden. Dies sollte die Kapazität des Operation-Teams erhöhen und somit die Qualität vom Betrieb verbessern. Wichtig ist:

- Im Überprüfungs-Framework kann das Operation-Team zielgerichtete Überprüfungen definieren, die kategorisierte Incidents im Ticketsystem anlegen.
- Die definierten Überprüfungen können einfach aktiviert und deaktiviert werden.
- Das Framework soll so konzipiert werden, dass eine Realisierung einer grafischen Benutzeroberfläche in einem späteren Zeitpunkt möglich ist.
- Das Konzept nutzt den PostFinance Standard.

Es wurde ein Technisches Konzept für ein Überprüfungs-Framework erarbeitet:

Alarming-Framework mit Definitionen auf der Datenbank und Nutzung von Splunk

- Vorhandenes Know-how, da alle benötigte technische Komponenten bereits im Einsatz sind.
- Mit Splunk wird der PostFinance Standard zur automatischen Incidenterstellung eingehalten.
- Überprüfungs-Definitionen auf der Datenbank können später mit einer grafischen Benutzeroberfläche hinzugefügt werden.
- Das Operation-Team muss nur noch Zeit aufwenden die Überprüfung zu definieren, der Rest läuft automatisch ab.
- Aufwand: ca. sechs Entwicklungstage *(gesamtes Technisches Konzept vorhanden und wurde durch DWH Architekt abgenommen)*

Empfehlung

Umsetzung des technischen Konzepts, weil dadurch rund 100 Stunden jährlicher Aufwand des Operation-Teams eingespart werden kann.

Glossar

A

ADW
: Active Datawarehouse. So heisst das DWH bei PostFinance. · 1, 10, 16, 23, 24, 27, 28, 29, 35

Alarmierung
: System bzw. Aufgabe anhand von Ereignissen die betroffenen zu Benachrichtigen · 1, 8

B

Backlog
: Im Kanban-Board befinden sich dort die Stories · 39

BMC Remedy
: ITMS Tool bzw. Ticket-System, dass bei PostFinance verwendet wird · 8

C

Confluence
: Wiki-Software zum Wissen dokumentieren · 8, 9, 24

D

Datalake
: Ein "Datensee". Bei uns benutzen wir es um Daten im Rohdatenformat zu speichern oder archivieren. · 23

Dataquality-Framework
: Framework, welches die Datenqualität auf dem DWH überprüft. · 17, 18, 20, 28

DB2
: relationales Datenbankmanagementsystem von IBM · 11, 39

DWH
: DataWareHouse. Ein Lager für strukturierte Daten. · I, 1, 3, 13, 18, 28, 41

I

Incident
: Ein offener Fall im Ticket-System · I, 9, 11, 14, 15, 18, 19, 21, 23, 25, 26, 27, 30, 38, 39

O

OP
: Operations bzw. Betrieb · 6, 8, 12, 20

Order
: So heisst ein Job im SOS Job Scheduler · 38

Outbound
: Alle DB-Prozesse, die Daten zusammenführen und ausliefern · 39

P

PF
: PostFinance AG · 8, 14, 15, 16, 21, 35

S

SOS JobScheduler
: Software von SOS GmbH. Ermöglicht das Erstellen von Jobketten und Schedules. · 21

Splunk
: Software bzw. Plattform um Logs effizient zu lesen und darstellen · I, 1, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 21, 25, 26, 35, 36, 37, 38

T

Ticket-System
: Software um Empfang, Bestätigung, Klassifizierung und Bearbeitung von Fällen zu handhaben · 8

TrueSight-Konsole
: Schnittstelle, die empfangene Events im Ticket-System als Incident anlegt · 9, 27

TS
: TypeScript, Skriptsprache für Webapplikationen · 40

U

UI
: User-Interface bzw. grafische Benutzeroberfläche · 1, 8, 14, 15, 16, 20, 36

Lebenslauf Diplomand

Kontakt

Giuseppe Scavetta
Aarauerstrasse 55, 4600 Olten
079 756 49 47
scavetta@outlook.com

21.05.1997, Olten

Castelmezzano PZ, Italien

ledig

Referenz

Kann nachgereicht werden

Berufserfahrung

01/2021 – heute — **DWH Entwickler, PostFinance AG,** Bern, 80%

- Arbeiten mit Oracle 19c (Toad, SQL, PL/SQL)
- Arbeiten mit Entwicklertools (IntelliJ IDEA, Bitbucket
- Deployment arbeiten (Datenmigration DB2 -> Oracle, Bereinigung der DB, starten der Ladejobs)
- Entwicklung vom Betriebscockpit (HTML, Angular, TypeScript und Java)
- Operative Verantwortung ADW2 (Oracle DWH)

09/2019 – 12/2020 — **DWH Operator, PostFinance AG,** Zofingen, 80%

- Arbeiten mit IBM DB2 6.5 (Toad, SQL)
- Ladeverantwortlichkeit von ETL-Jobs (IBM InfoSphere Datastage and QualityStage Director)
- 1st Level: Service Management (Active-DWH)
- Fehleranalysen und Erfassung von Verbesserungsvorschlägen (Atlassian Jira Software)
- Pflegen der Betriebsdokumentation (Confluence)

08/2018 – 08/2019 — **Informatiker, Glas Trösch AG,** Oensingen, 80 - 100%

- 2nd Level: Service Management (DWH, ERP Infor COM)
- Kundenberatung und Umsetzung div. Kundenbedürfnisse (Infor COM, Infor BI mit MS SQL DWH Umgebung)
- Stellvertretende Produktverantwortung des Glas Trösch MIS (Infor BI)
- Stellvertretende Produktverantwortung des DataWareHouses (Oracle 11g und MS SQL 2016)

08/2014 – 07/2018 — **Ausbildung zum Informatiker, Glas Trösch AG,** Oensingen, 100%

- Kundenberatung und Umsetzung div. Kundenbedürfnisse (Infor COM, Infor BI)
- Prozessautomatisierungen
- Arbeiten mit Oracle 11g r2 (Oracle Developer, SQL, PL/SQL)
- Arbeiten mit MS SQL 2016 (SSMS, SSIS, SSDT, SQL, T-SQL)
- Arbeiten im DWH (ETL, OLAP, Daten- und Schnittstellen bauen und überwachen)
- Scripting-Projekte (JavaScript, VB-Script, PowerShell)
- DWH-Migrationsprojekt (Oracle -> MS SQL)
- 1st Level: Service Management als Stellvertretung

08/2013 – 07/2014 — **Kaufmann, Glas Trösch AG,** Oensingen, 100%

- Sekretariat, vertieftes arbeiten mit Office-Anwendungen
- Lager und Spedition, Sachbearbeiter für die Abläufe im Lager (Arbeiten mit Infor COM)

Kontakt

Giuseppe Scavetta
Aarauerstrasse 55, 4600 Olten
079 756 49 47
scavetta@outlook.com

Referenz

Kann nachgereicht werden

Aus- und Weiterbildungen

2018 - Heute	**Berufsbegleitendes Studium, diplomierter Wirtschaftsinformatiker HF,** Höhere Fachschule TEKO Olten
2014 - 2018	**Gewerblich-Industrielle Berufsfachschule, Informatiker Applikationsentwickler EFZ,** Solothurn Ausbildungs-Betrieb: Glas Trösch AG, Oensingen

Zertifizierungen

07/2020	ITIL® Foundation Certificate in IT Service Management
05/2017	Kommunikation
05/2012	Italienisches Sprachdiplom

Sprachen

Italienisch	Muttersprache
Deutsch	2. Muttersprache
English	Gut in Wort und Schrift
Französisch	Schulkenntnisse

Interessen und Hobbys

Musik, Familie, soziale Kontakte, Natur, Geschichte

1. Initialisierung und Planung

1.1. Pflichtenheft

1.1.1. Vorstellung des Themenbereiches

PostFinance AG gehört zu den führenden Finanzinstituten der Schweiz. Rund 2,7 Millionen Kundinnen und Kunden vertrauen auf PostFinance AG, wenn es um ihr Geld geht. Dabei wird auf zukunftsweisende Tools und Technologien gesetzt, die speziell für den Schweizer Markt entwickelt oder adaptiert werden.

Die meisten PostFinance Applikationen verarbeiten und liefern Daten welche in einem Datawarehouse integriert werden. Das Besondere ist, dass in kürzester Zeit Millionen von Datensätzen konsumiert, verarbeitet und anschliessend weiteren Applikationen bereitgestellt werden müssen. Aus diesem Grund wird unser Warehouse «Active»-DataWarehouse genannt (ADW).

Für den reibungslosen Ablauf des ADWs ist das Operation-Team zuständig. Während dieses Ablaufs können die unterschiedlichsten Fehler entstehen, die vom Operation-Team überprüft werden müssen.

1.1.2. Beschreibung Zwecks

Es soll eine Schnittstelle entwickelt werden, die es dem Operation-Team ermöglicht die Überprüfungen erfassen zu können. Auf Basis dieser Überprüfung sollen dann automatisiert Incidents im Ticketsystem generiert werden.

Durch diese Diplomarbeit wird dem Abnehmer ein Technisches Konzept vorgelegt, welches dann zur Umsetzung in Auftrag gegeben werden kann.

1.1.3. Woraus ist die Idee entstanden?

Ich habe vor einem Jahr die Verantwortung der Alarmierung bei ADW übernommen. Alle Anfragen für neue Überprüfungen die erstellt werden sollten, kommen zu mir.

Um Log-Files zu lesen und daraus Incidents zu erstellen eignet sich die bestehende Umsetzung mittels Spunk sehr gut. Leider eignet es sich nicht um komplizierte Überprüfungen auf der DB zu tätigen. SQL-Abfragen über Splunk, die länger als 30 Sekunden dauern, brechen ab. Die SQL-Syntax in Splunk entspricht nicht der unserer Datenbank und von Splunk aus haben wir nicht auf alle Datenbank-Schemas Zugriff.

Dies war für mich der Auslöser eine strukturierte Komponente auf der Datenbank zu entwickeln um Überprüfungen zu tätigen. Somit kann das gesamte Operation-Team für ihre Aufgaben und Kundenanfragen selbstständig Überprüfungen erfassen. In einem weiteren Schritt kann diese Komponente in das Betriebscockpit integriert werden und somit die Erfassung mittels UI ermöglichen.

1.1.4. Abnehmer

Das Technische Konzept wird von unserem DWH Architekt Patrick Bregy abgenommen und anschliessend von mir umgesetzt. Das gesamte Operation-Team wird diese neue Komponente nutzen.

1.1.5. Ansprüche vonseiten des Abnehmers

Der Abnehmer möchte, dass in regelmässigen Abständen der fortschritt des Konzepts besprochen wird. Somit kann er verifizieren, dass es in die richtige Richtung geht.

1.1.6. Vorgehensweise bei dieser Arbeit

Dieses Projekt wird nach dem Bottom-Up-Ansatz umgesetzt. Ich habe die Freiheit Lösungsvarianten zu erarbeiten und nach Absprache mit dem Fachbetreuer umzusetzen.

1.2. Inhalt und Teilziele

1.2.1. Übergeordnetes Richtziel der Arbeit

Am 25. Oktober 2021 liegt ein Technisches Konzept zur automatischen Erstellung von kategorisierten Incidents für die PostFinance AG vor.

Mit folgenden zu bearbeitenden Punkten und Resultaten (Endergebnisse und Erfolgskriterien) soll der Abnehmer die Umsetzung des Konzepts genehmigen.

1.2.2. Erfolgskriterien zu den Endergebnissen

Tabelle 1: Erfolgskriterien und Endergebnisse

Endergebnisse Was liegt am Schluss vor?	**Erfolgskriterium** Wie messe ich die erfolgreiche Bearbeitung des Endergebnisses?
Analyse des bestehenden Konzepts	Das bestehende Alarmierungs-Konzept wird gründlich analysiert. Durch die Analyse kann ein tieferes Verständnis gewonnen werden und eventuell die Stärken bzw. nutzbaren Elemente in das neue Konzept integriert werden. • Es sind je drei Vor- und Nachteile des bestehenden Konzepts beschrieben. • Die Operation-Teamleitung bestätigt bis zum 03.09.2021 je zwei der vorgelegten Vor- und Nachteile.
Zwei Lösungsvorschläge	Durch das erlangte Wissen können zwei Lösungsvorschläge erstellt werden und anschliessend mit dem Fachbetreuer besprochen werden. • Aus einem Brainstorming werden zwei Lösungsvorschläge abgeleitet. • Bis zum 10.09.2021 bestätigt der Fachbetreuer an einem Meeting, dass einer der Lösungsvorschläge 40% des Funktionsumfangs wiedergibt.

Technisches Konzept	Nachdem sich der Fachbetreuer für eine Variante entschieden hat, wird in einem Technischen Konzept die Variante vollständig ausgearbeitet.
	• Am 01.10.2021 liegt ein technisches Konzept, welches mindestens die folgenden Punkte beinhaltet und zwei von drei evaluierten Nachteile der aktuellen Lösung Eliminiert
	  ○ Lösungsstrategie
	  ○ Abhängigkeiten
	  ○ Laufzeitübersicht mit Aktivitäten Diagramm
	  ○ Deploystrategie
	  ○ Konzept mit ERD
	  ○ Qualitätsanforderungen
	  ○ Risiko/Technische Schulden
	• Der Fachbetreuer bestätigt bis zum 01.10.2021 die Vollständigkeit des Konzepts.

1.3. Fachbetreuer

Der Fachbetreuer ist auch der Abnehmer der Arbeit:

Patrick Bregy, DWH Architekt

Mingerstrasse 12, 3030 Bern

+41 76 568 13 44, patrick.bregy@postfinance.ch

1.4. Projektstrukturplanung

In der Projektstrukturplanung habe ich vor der Aufgabenebene den dazugehörigen Prozessschritt angegeben.

Tabelle 2: Projektstrukturplanung

Phase	Nr.	Aufgabenebene	
Initialisierung und Planung	A	*Pflichtenheft*	Themenvorstellung
			Zweck
			Entstehung der Idee
			Abnehmer
			Ansprüche des Abnehmers
			Zieldefinition
	B	*Projektstrukturplan mit Arbeitspaketbeschreibung*	Erarbeiten
			Abschliessen
	C	*Projektablaufplanung*	Netzplan anhand von Arbeitspaketbeschreibung
			Arbeitspaketbeschreibung in Ablaufplanung integrieren
			Milestones definieren
Realisierung	D	*Analyse des bestehenden Konzepts*	Informationsbeschaffung (Recherche)
			Vor- und Nachteile herausfinden
			Nutzbare Elemente beschreiben
			Mit OP-Teamleitung besprechen und bestätigen
	E	*Lösungsvorschlag*	Brainstorming des Projektproblems
			Variante aus Brainstorming finden
			Gefundene Variante beschreiben
			Mit Fachbetreuer besprechen und bestätigen
	F	*Technisches Konzept*	Benötigte Komponenten/Applikationen beschreiben
			Konfigurationen der Komponenten/Applikationen beschreiben
			Jobsteuerung bzw. Auslöser beschreiben
			Mit Fachbetreuer besprechen
			ERD erstellen für neue Datenbankobjekte
			Struktogramm erstellen für zu programmierende Logik
			Mit Fachbetreuer besprechen und bestätigen
Abschluss	G	*Mehrwert dieser Arbeit*	Fragestellung
			Berechnung der Wirtschaftlichkeit
	H	*Review*	Persönliche Reflexion und Lessons learnt
			Schlusswort und Danksagung
			Eigenständigkeits-Erklärung
	I	*Finalisierung*	Texte ausarbeiten
			Formatierungsarbeiten
			Drucken und binden
			Abgabe

1.5. Netzplan

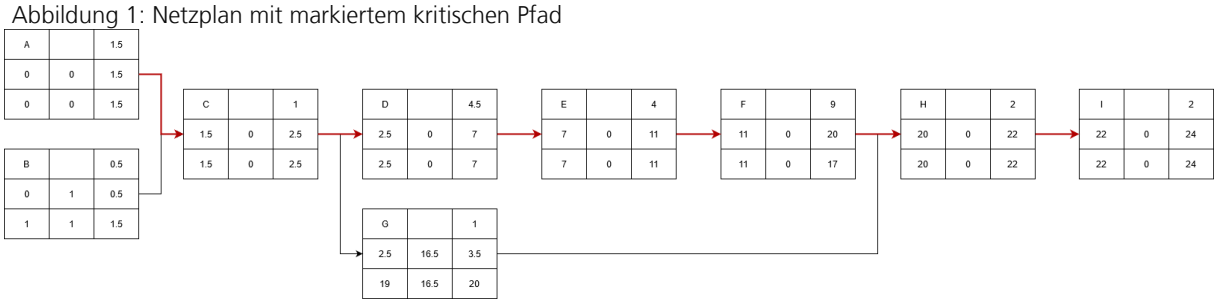
Aus der Projektstrukturplanung habe ich die Prozessschritte entnommen und meinen Aufwand (anhand von den Tätigkeiten) eingeschätzt:

Tabelle 3: Prozessschritte aus Projektstrukturplanung

Prozessschritt	Dauer in Tagen	Vorher zu beenden
A Pflichtenheft	1.5	
B Projektstrukturplan	0.5	
C Projektablaufplan	1	A, B
D IST-Analyse	4.5	C
E Lösungsvorschlag	4	D
F Technisches Konzept	9	E
G Mehrwert dieser Arbeit	1	C
H Review	2	F, G
I Finalisierung	2	H

Dadurch können wir den folgenden Netzplan entnehmen:

Abbildung 1: Netzplan mit markiertem kritischen Pfad



1.6. Ablaufplanung

1.6.1. SOLL

Mit meiner Aufwandschätzung und der gegebenen Zeit komme ich auf den folgenden Projektablaufplan:

Abbildung 2: Projekt SOLL-Ablaufplanung

	August			September				Oktober			
	KW 33	KW 34	KW 35	KW 36	KW 37	KW 38	KW 39	KW 40	KW 41	KW 42	KW 43
	16.08	23.08	30.08	6.09	13.09	20.09	27.09	4.10	11.10	18.10	25.10
Initialisierung und Planung											
Pflichtenheft											
Erarbeiten											
Projektstrukturplan mit Arbeitspaketbeschreibung											
Erarbeiten											
Projektablaufplanung											
Netzplan erstellen											
Ablaufplan erstellen											
Milestones definieren											
Realisierung											
Analyse des bestehenden Konzepts											
Informationsbeschaffung											
Vor- und Nachteile beschreiben											
Nutzbare Elemente beschreiben											
Mit OP-Teamleitung besprechen und bestätigen											
Lösungsvorschlag											
Brainstorming											
Variante beschreiben											
Mit Fachbetreuer besprechen und bestätigen											
Technisches Konzept											
Benötigte Komponenten/Applikationen beschreiben											
Konfiguration der Komponenten/Applikationen beschreiben											
Jobsteuerung bzw. Auslöser beschreiben											
ERD für neue Datenbankobjekte											
Struktogramm für zu programmierende Logik											
Mit Fachbetreuer besprechen und bestätigen											
Abschluss											
Mehrwert dieser Arbeit											
Fragestellung											
Berechnung der Wirtschaftlichkeit											
Review											
Persönliche Reflexion und Lessons learnt											
Schlusswort und Danksagung											
Eigenständigkeits-Erklärung											
Finalisierung											
Texte ausarbeiten											
Formatierungsarbeiten											
Drucken und binden											
Abgabe											

Die Meilensteine bestehen aus den Meetings bzw. Besprechungen mit der OP-Teamleiterin und dem Fachbetreuer. Erst durch ihre Bestätigungen kann ich mit dem Projekt fortfahren.

1.6.2. 1. Milestone

Bis zum 03.09.2021 hat die Operation-Teamleitung meine Voranalyse bestätigt.

1.6.3. 2. Milestone

Bis zum 10.09.2021 wählt der Fachbetreuer eine mögliche Lösungsvariante.

1.6.4. 3. Milestone

Bis zum 01.10.2021 bestätigt der Fachbetreuer die Vollständigkeit des Technischen Konzepts.

1.6.5. IST

Abbildung 3: Projekt IST-Ablaufplanung

	August			September				Oktober				November	
	KW 33	KW 34	KW 35	KW 36	KW 37	KW 38	KW 39	KW 40	KW 41	KW 42	KW 43	KW 44	KW 45
	16.08	23.08	30.08	6.09	13.09	20.09	27.09	4.10	11.10	18.10	25.10	01.11	08.11
Initialisierung und Planung													
Pflichtenheft													
Erarbeiten													
Projektstrukturplan mit Arbeitspaketbeschreibung													
Erarbeiten													
Projektablaufplanung													
Netzplan erstellen													
Ablaufplan erstellen													
Milestones definieren													
Realisierung													
Analyse des bestehenden Konzepts													
Informationsbeschaffung													
Vor- und Nachteile beschreiben													
Nutzbare Elemente beschreiben													
Mit OP-Teamleitung besprechen und bestätigen													
Lösungsvorschlag													
Brainstorming													
Variante beschreiben													
Mit Fachbetreuer besprechen und bestätigen													
Technisches Konzept													
Benötigte Komponenten/Applikationen beschreiben						Krank		Bugfix für Prod					
Konfiguration der Komponenten/Applikationen beschreiben													
Jobsteuerung bzw. Auslöser beschreiben													
ERD für neue Datenbankobjekte													
Struktogramm für zu programmierende Logik													
Mit Fachbetreuer besprechen und bestätigen													
Abschluss													
Mehrwert dieser Arbeit													
Fragestellung													
Berechnung der Wirtschaftlichkeit													
Review													
Persönliche Reflexion und Lessons learnt													
Schlusswort und Danksagung													
Eigenständigkeits-Erklärung													
Finalisierung													
Texte ausarbeiten													
Formatierungsarbeiten													
Drucken und binden													
Abgabe													

Leider war ich an COVID-19 erkrankt und deshalb hat sich meine Abgabe um zwei Wochen verschoben. Die Stakeholder (Telefonisch) und der Diplomlehrer bzw. Teko wurden informiert und sie sind mit der Verschiebung einverstanden.

Abbildung 4: Screenshot Mail von Abwesenheit

Von: giuseppe.scavetta@postfinance.ch <giuseppe.scavetta@postfinance.ch>
Gesendet: Montag, 11. Oktober 2021 19:18
An: Gregor von Flüe <gregor.von_fluee@edu.teko.ch>; Matthias Aregger <matthias.aregger@edu.teko.ch>
Cc: scavetta@outlook.com
Betreff: Diplomarbeit: Verschiebung Abgabetermin

Hallo zusammen

Anbei findet ihr die Verfügung des Kantonsarztes meiner getätigten Isolation vom 20.09.2021 bis und mit 29.09.2021.
Mein Hausarzt hat mich noch für 2 weitere Tage krankgeschrieben, vom 30.09.2021 bis und mit 01.10.2021.

Somit hatte ich eine 100% Arbeitsunfähigkeit für 10 Arbeitstage.
Ich habe dies mit Gregor (Diplomlehrer) besprochen und wir sind beide damit Einverstanden die Abgabe auf den 08.11.2021 zu verschieben.

Meine Online-Publikation werde ich bis zum 12.11.2021 aufschalten können und den Präsentationstermin wie geplant am 19.11.2021 wahrnehmen.

Für weitere Fragen stehe ich euch gerne zur Verfügung.

Viele Grüsse

Giuseppe Scavetta
ADW2 DevOps Engineer

PostFinance AG
Mingerstrasse 12
3030 Bern

☎ +41 76 458 21 48
✉ giuseppe.scavetta@postfinance.ch

2. Realisierung

In diesem Abschnitt folgt die Dokumentation der gesamten Realisierungsphase.

2.1. Analyse des bestehenden Konzepts

Als erstes wurde das bestehende Alarmierungs-Konzept analysiert. Daraus sollten schwächen und stärken erkennbar gemacht werden und das Bedürfnis des Operation-Teams verstanden werden.

2.1.1. Informationsbeschaffung

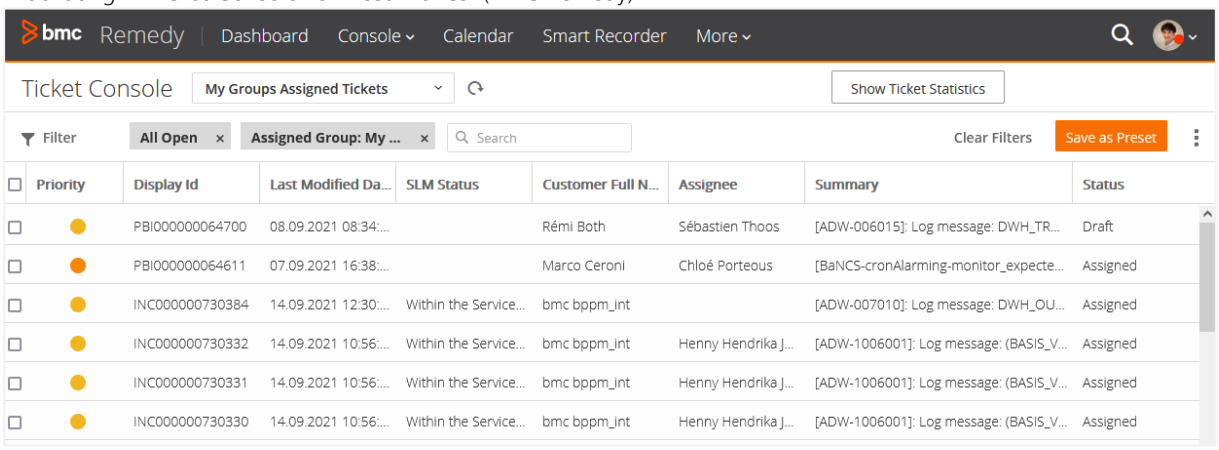
Bei PostFinance haben wir Confluence im Einsatz. Aus einer ausführlichen Confluence-Seite konnte ich die PF-Standards entnehmen, um Incidents im Ticket-System anzulegen.

Neben den Confluenceseiten wurde ich von meinem Vorgänger (Verantwortlicher Alarmierung) vor einem Jahr eingeschult. Das Meeting mit der OP-Teamleitung hat ebenfalls zur Informationsbeschaffung beigetragen.

2.1.2. Übersicht

Wie in Abbildung 5 ersichtlich, nutzen wir als ITSM das BMC Remedy:

Abbildung 5: Ticket Console von PostFinance. (BMC Remedy)



Das Operation-Team kann hier (im UI) manuell Tickets anlegen und beliebig in der gesamten Unternehmung zuweisen.

Jedoch gibt es drei PostFinance-Standards bzw. Schnittstellen, die es ermöglichen Incidents im ITSM anzulegen:

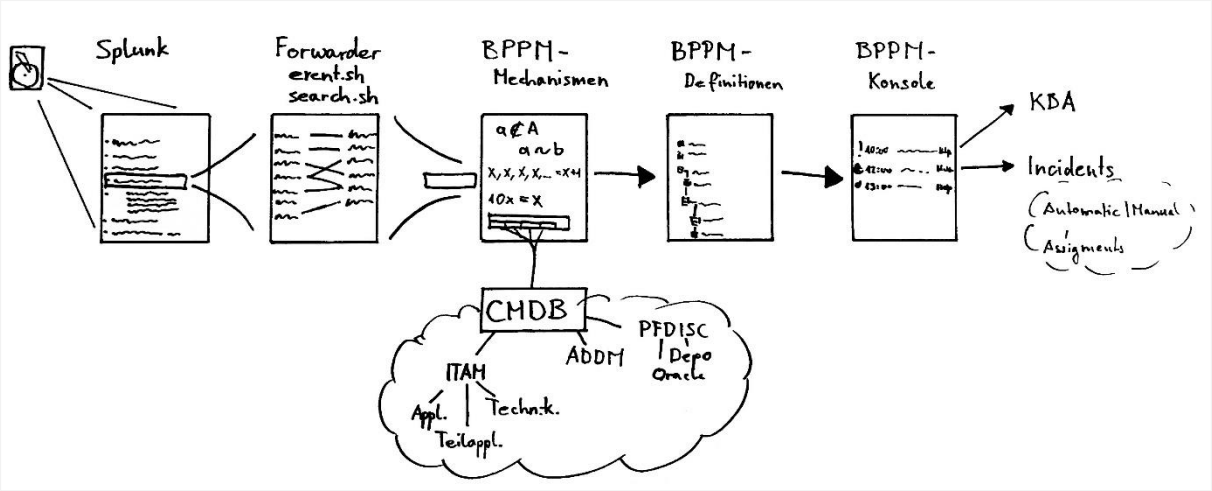
- **Java Library APLAT**
  - Eine Java-Bibliothek, die genutzt werden kann um Incidents anzulegen. Wird nur in PF eigenen Applikationen angewendet.

- **Command line interface PATSNMP und OPCMSG**
  - Auf Solaris- und Linux-Systemen stehen diese Kommandozeilen-Tools zur Verfügung. Diese werden hauptsächlich genutzt um Zustände von Servern zu melden. (z. B. Diskspace full bei Server xy)
  - OPCMSG steht nur noch aus Kompatibilitätsgründen zur Verfügung (Vorgänger von PATSNMP).

- **Log Monitoring Splunk**
  - Diese Methode ist für alle Applikationen geeignet, weil sie viele Möglichkeiten bietet um Events zu lokalisieren.
  - Es kann eine beliebige Splunk-Benachrichtigung/Suche angelegt werden, wichtig ist, dass bei der Ausführung das vorgegebene Skript aufgerufen wird und mit den Pflichtparametern befüllt wird. Anschliessend wird der Incident im ITSM angelegt.

Im bestehenden Alarming-Konzept wird auf das Log Monitoring von Splunk gesetzt.

Abbildung 6: Ablauf Log-Monitoring von PostFinance *(Quelle: Confluence von PostFinance)*



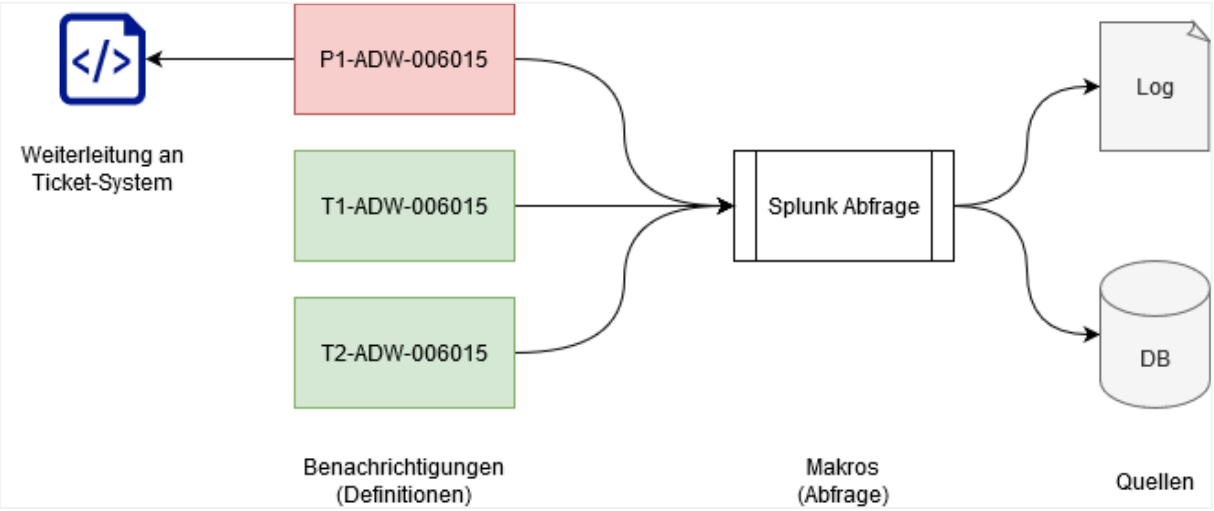
Der Ablauf vom Log Monitoring über Splunk ist in Abbildung 6 visualisiert. Folgende drei wichtige Schritte werden in diesem Ablauf getätigt:

1. Splunk Event wird über Forwarder-Skript weitergeleitet. Es gibt das event.sh und das search.sh.
   a. Sobald es Dynamische-Werte bei einem Splunk-Search gibt, die zur Erstellung eines Incidents benötigt werden, nutzt man das Event.sh-Skript.
   b. Bei einem Statischen Splunk-Search (also keine Dynamischen Werte) nutzt man das search.sh Skript.
2. Eine andere PostFinance Applikation («BPPM» heisst jetzt neu TrueSight-Konsole) sammelt diese Events. Mittels einem Parameter (DEDUP-KEY), können mehrere Events zu einem gruppiert werden.
3. Die gesammelten Events werden im ITSM von der TrueSight-Konsole angelegt.

Da es sich um ein PostFinance-Standard handelt, kann man am oberen Konstrukt nichts verändern. Falls man genügend Argumente für eine Änderung des Standards hat, kann man das bestimmt in die Wege leiten. Wiederum muss man sagen, dass sich dieser Standard bis jetzt bewährt hat und es keinen Grund für eine Änderung gibt im Zusammenhang mit meiner Arbeit.

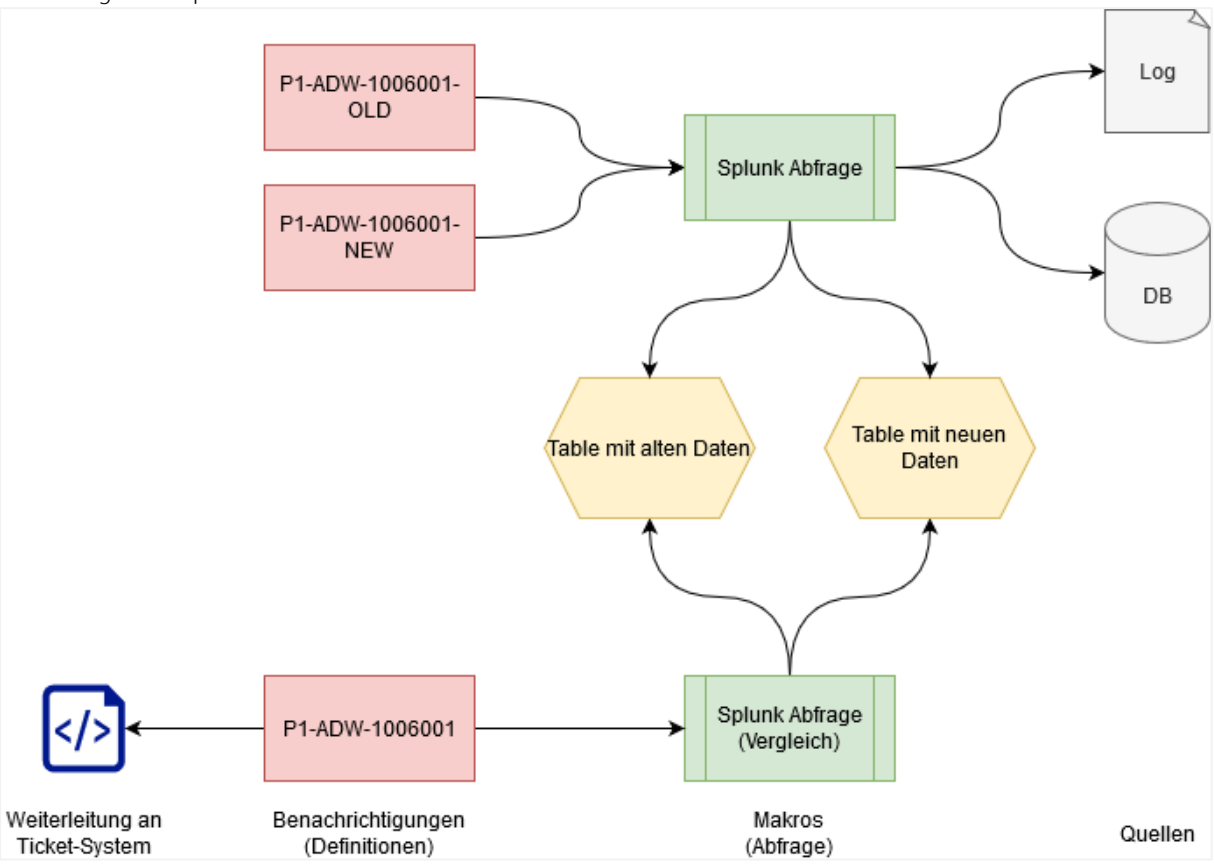
Für die Erfassung der Benachrichtigungen im Splunk haben wir bei unserer Applikation (ADW) einen Standard definiert. In Abbildung 7 ist der definierte Standard visualisiert. Die erfassten Benachrichtigungen/Überprüfungen im Splunk müssen eine Funktion/Makros aufrufen, welche dann die Splunk-Suche auf den Quellen durchführen. Die Funktion/Makros gibt dann die Pflichtfelder für das Weiterleitungs-Skript zurück.

Abbildung 7: ADW-Standard für Alarm-Definition



Leider konnte mein Vorgänger den definierten Standard nicht sehr konsequent durchziehen, deshalb treffen wir grösstenteils sehr komplizierte und nicht dokumentierte Splunk-Benachrichtigungen an wie in Abbildung 8 zu sehen ist.:

Abbildung 8: Komplexe Alarm-Definition



Da wir gerade von DB2 auf Oracle migrieren, kommt eine Bereinigung der rund 200 Splunk-Benachrichtigungen nicht in Frage (da sie alle zu DB2 gehören).

Für die Oracle-Umgebung haben wir noch kein Alarmierungs-Konzept, deshalb haben wir eine unberührte Umgebung auf der wir ein sauberes Alarmierungs-Konstrukt aufbauen können.

2.1.3. Vor- und Nachteile vom aktuellen Konzept

Aus der vorherigen Analyse und der früheren täglichen Arbeit im Operation-Team, habe ich die folgenden Vor- und Nachteile identifizieren können:

Tabelle 4: Vor- und Nachteile vom alten Konzept

Vorteile	Nachteile
Es können sehr schnell Log-Files ausgelesen werden. • Log-Files in Splunk sind indexiert, deshalb kann in einem grossen Log-File sehr schnell eine spezifische Message ausgelesen werden.	Keine komplexen SQL-Abfragen, da die Abfrage nicht länger als 30s sein darf. • Beispielsweise würden Daten-Qualitätschecks von mehreren Millionen Zeilen nicht möglich sein.
In Splunk Lookup-Tables können Resultate aus vorherigen Überprüfungen gespeichert werden. • Auf Basis der bestehenden Resultate können weitere Überprüfungen durchgeführt werden	Einen Alarm auf SQL Basis umzusetzen ist mit grösserem Aufwand verbunden: • Es muss eine Splunk-Funktion mit dem SQL-Statement eingerichtet werden. • Die eigentliche Benachrichtigung bzw. Überprüfung muss dann auf diese Funktion zugreifen.
Der «Forwarder» übergibt Splunk-Events bzw. Ereignisse so weiter, dass Incidents im ITSM angelegt werden.	In diesem Konzept werden keine fachlichen Mails versendet. Es werden Mails in den Splunk-Alarmen definiert, aber dabei handelt es sich eher um ein E-Mail mit einem Technischen Inhalt. Dies ist hilfreich für den Operator oder Entwickler, aber nicht für einen Stakeholder.

2.1.4. Nutzbare Elemente

Für das Auslesen von Events in Log-Files und das Erstellen von Incidents im ITSM, kann Splunk weiterhin als Schnittstelle genutzt werden.

Jedoch sollten keine komplexen SQL-Statements mehr in Splunk implementiert werden. Vor allem werden auf dieser Art und Weise nicht alle Anforderungen des Operation-Teams gedeckt. Z.B. kann das Operation-Team noch keine E-Mails versenden, die von spezifischen Konstellationen auf der Datenbank ausgelöst werden.

2.1.5. Bedürfnis

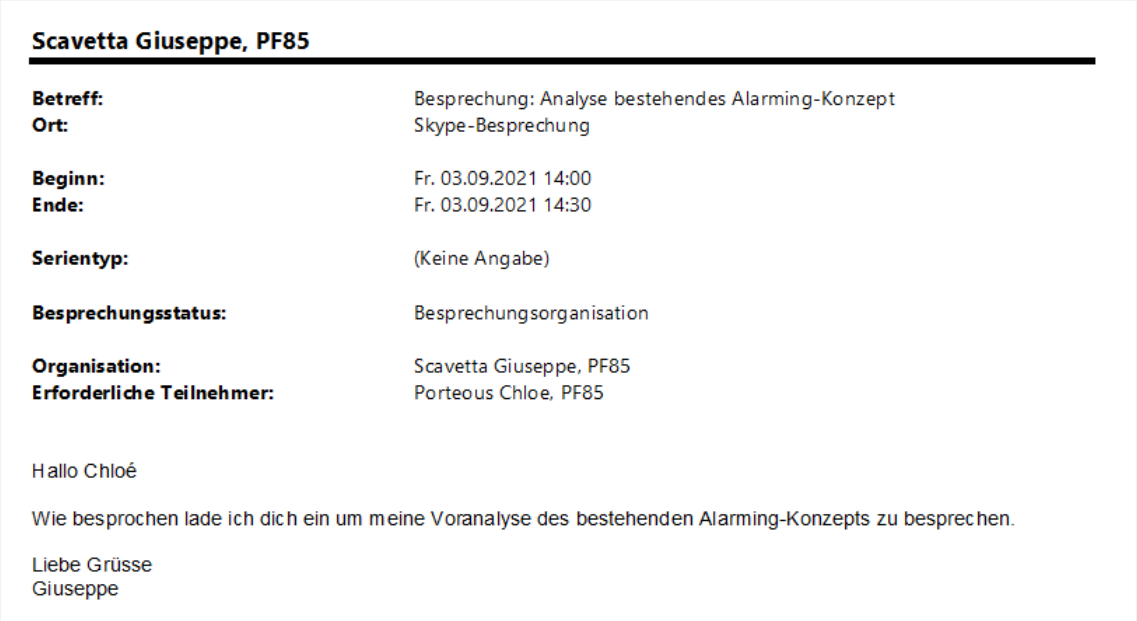
Das Operation-Team möchte möglichst einfach SQL-Überprüfungen definieren, die dann (je nach Definition) ein Incident erstellen oder E-Mail versenden.

Die SQL-Überprüfungen sind überschaubar und können ohne grösseren Aufwand aktiviert oder deaktiviert werden.

2.1.6. Meeting mit OP-Teamleitung

Wie in Abbildung 9 ersichtlich, habe ich die OP-Teamleitung zu einem Termin eingeladen, um meine Analyse sowie auch die Vor- und Nachteile zu besprechen.

Abbildung 9: Screenshot von Termineinladung mit OP-Teamleitung



Scavetta Giuseppe, PF85

Betreff:	Besprechung: Analyse bestehendes Alarming-Konzept
Ort:	Skype-Besprechung
Beginn:	Fr. 03.09.2021 14:00
Ende:	Fr. 03.09.2021 14:30
Serientyp:	(Keine Angabe)
Besprechungsstatus:	Besprechungsorganisation
Organisation:	Scavetta Giuseppe, PF85
Erforderliche Teilnehmer:	Porteous Chloe, PF85

Hallo Chloé

Wie besprochen lade ich dich ein um meine Voranalyse des bestehenden Alarming-Konzepts zu besprechen.

Liebe Grüsse
Giuseppe

Alle Vor- und Nachteile wurden bestätigt. Die OP-Teamleitung (Porteous Chloe) hat zu den Bedürfnissen angemerkt, dass die Wartbarkeit und Nachvollziehbarkeit mit Splunk nicht gegeben ist. Ebenfalls ist es im Splunk umständlich mehrere Benachrichtigungen zu deaktivieren, somit wird der Deployment-Prozess erschwert.

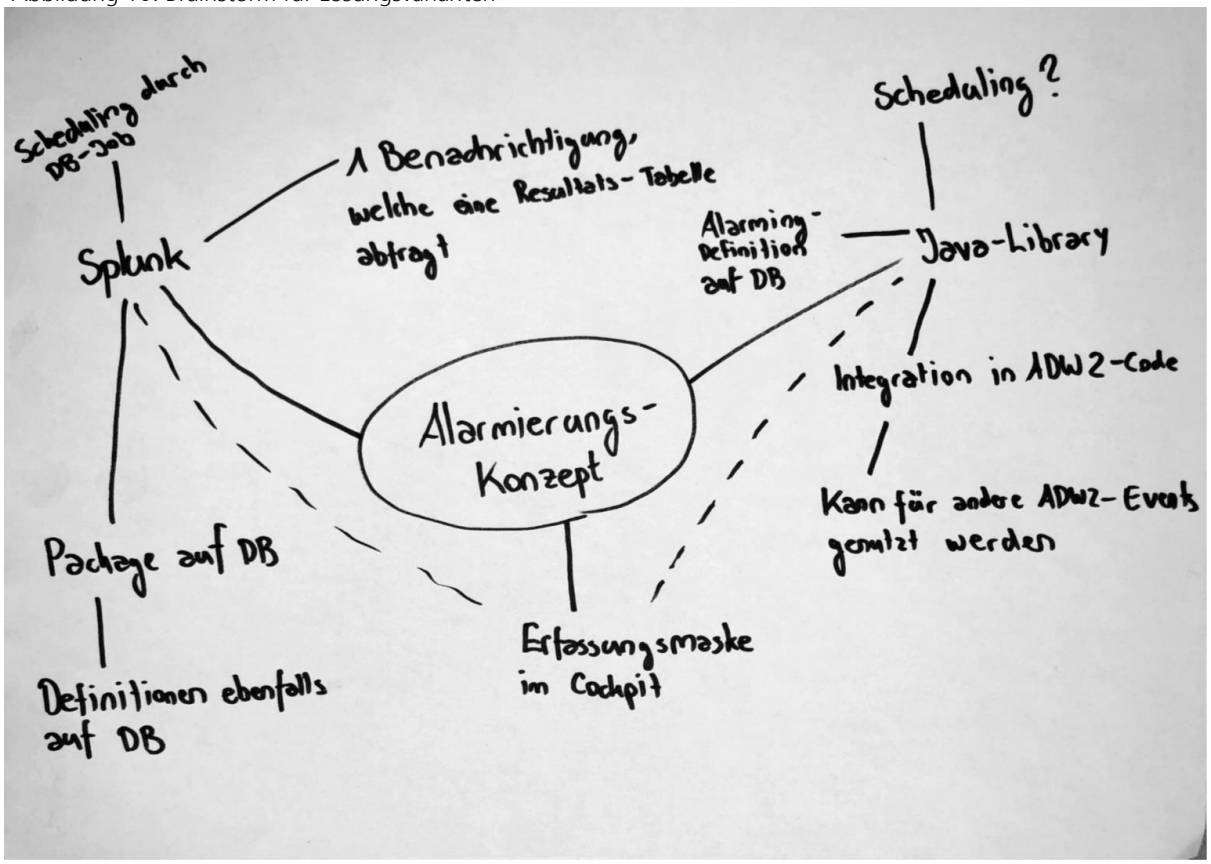
Die OP-Teamleitung hat die Inputs mitgegeben, damit diese in der Konzeptionierung berücksichtigt werden.

2.2. Lösungsvorschlag

Durch die Analyse kennen wir nun die Vor- und Nachteile und habe ein tieferes Verständnis der Bedürfnisse vom Operation-Team. Durch die Vorgaben von PostFinance sind wir auf drei Technologien bzw. Möglichkeiten begrenzt. Eine davon (Command Line Interface) fällt weg, da diese für Server-Events geeignet ist.

2.2.1. Brainstorm

Abbildung 10: Brainstorm für Lösungsvarianten



Wie in Abbildung 10 ersichtlich, konnte ich aus dem Brainstorming zwei Mögliche Varianten ableiten. Eine Variante mit dem Java-Library Standard und eine Variante mit dem Splunk Standard.

Unser gesamtes DWH basiert auf Metadaten, d. h. dass alle Abläufe bzw. Prozesse von Metadaten beschrieben bzw. definiert werden. Für eine automatische Durchführung von Überprüfungen müssen auch irgendwo die Überprüfungen definiert werden. Z.B. muss definiert werden wann eine Überprüfung durchgeführt wird, was geschieht bei einem Ereignis, wie kritisch ist ein Ereignis usw.

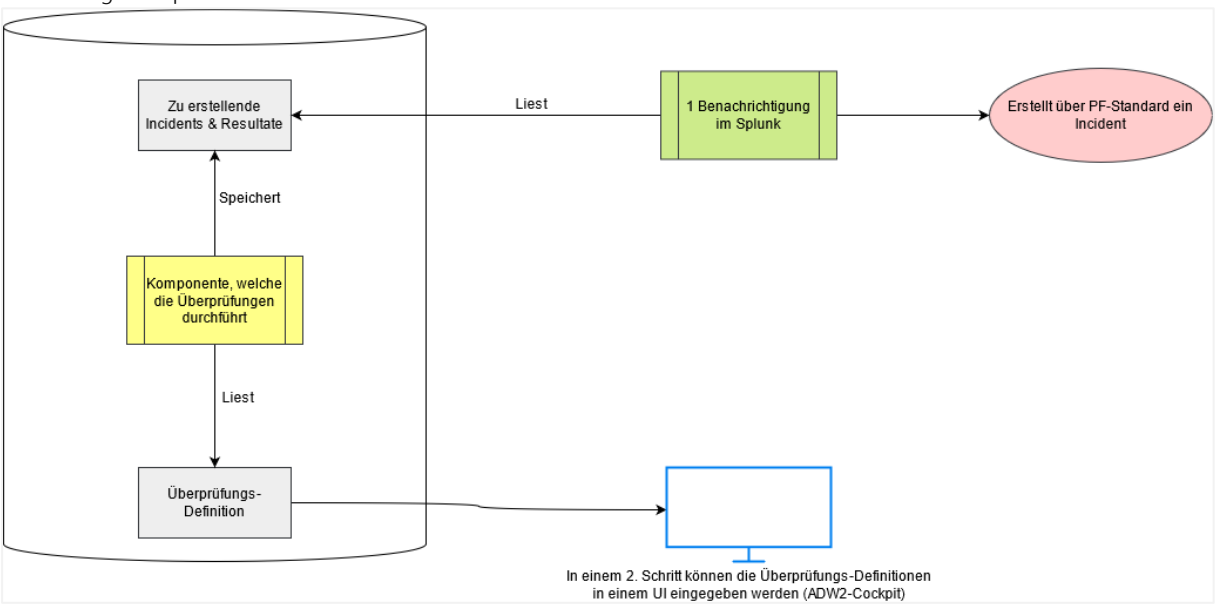
Aus diesem Grund basieren beide Varianten auf Metadaten bzw. SQL-Überprüfungs-Definitionen, die auf der Datenbank angelegt werden.

Im nachfolgenden Abschnitt werde ich den groben Ablauf der zwei Varianten beschreiben. Somit kann ich diese dem Fachbetreuer vorstellen, damit die Entscheidung leichter fällt in welche Richtung wir mit dem Technischen Konzept einlenken werden.

2.2.2. 1. Variante: Splunk mit Komponente auf der Datenbank

Die erste Variante basiert auf den PF-Standard zur Incident Erstellung über Splunk. In Abbildung 11 ist eine Visualisierung, wie ich mir diese Variante vorstelle.

Abbildung 11: Splunk Variante im Überblick



Das Operation-Team legt die SQL-Überprüfungen bzw. die Überprüfungsdefinitionen auf der Datenbank an.

Die gelbe Komponente bzw. ein Package auf der Datenbank enthält dann Funktionen, die auf Basis der Überprüfungsdefinitionen die Überprüfungen durchführt und die Resultate in einer Tabelle speichert.

Anschliessend arbeitet eine Splunk-Benachrichtigung diese Tabelle ab und erstellt die dazugehörenden Incidents über den gegebenen PF-Standard.

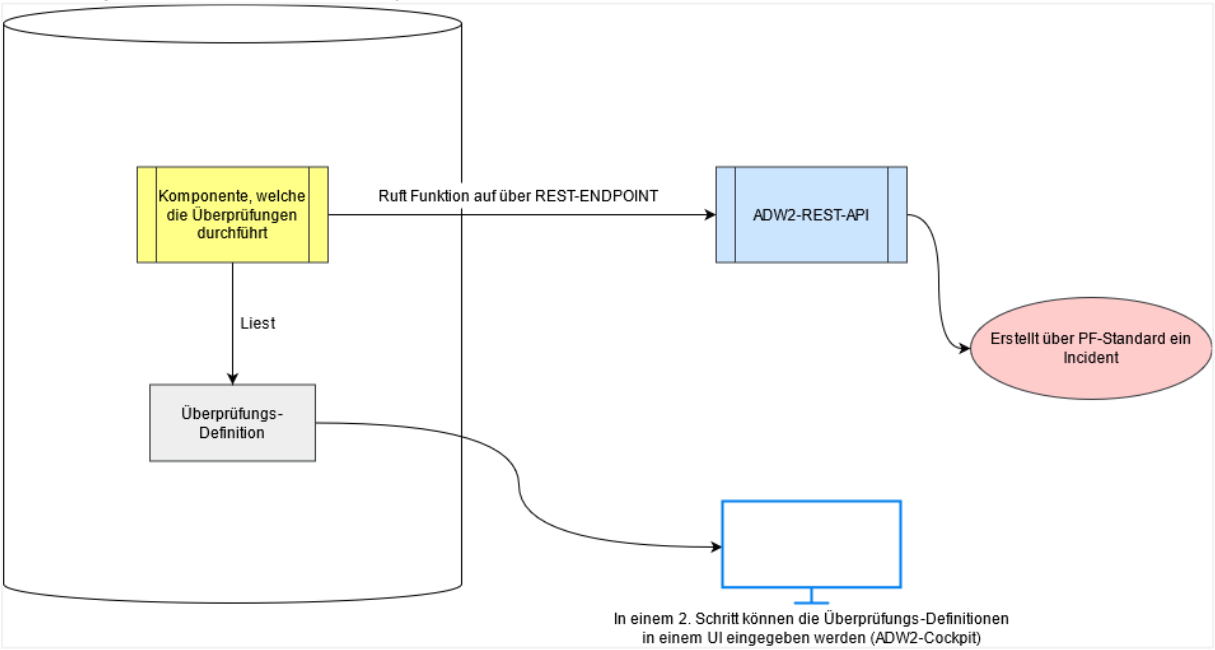
Andere Prozesse können auf die Funktionen vom Package ebenfalls zugreifen und zu erstellende Incidents anlegen.

In einem 2. Schritt könnte man die Erfassung der Überprüfungsdefinitionen in unserem Betriebscockpit integrieren. Somit wäre die Erfassung von den Überprüfungsdefinitionen in einem UI geführt.

2.2.3. 2. Variante: PF Java-Library mit Definitionen auf der Datenbank

In diesem Abschnitt erläutere ich die zweite Variante, die auf den PF-Standard zur Incident Erstellung über die Java-Library abzielt. In Abbildung 12 ist das Zusammenspiel dieser Variante abgebildet.

Abbildung 12: PostFinance Java-Library im Überblick



Wie bei der 1. Variante, werden die Überprüfungsdefinitionen durch das Operation-Team auf der Datenbank angelegt *(graues Kästchen)*.

Die PF-Java-Library zum Incidents erstellen wird in die ADW2-REST-API *(blaues Kästchen)* integriert. Ein neuer Endpunk wird dazu entwickelt, die bei einem Aufruf ein Incident aus den Infos vom gesendeten JSON erstellt.

Ein Package *(gelbe Komponente)* führt die Überprüfungen durch. Bei Bedarf ruft es den Endpunkt auf und gibt die Infos um ein Incident zu erstellen als JSON mit.

Der Endpunkt könnte auch von anderen Prozessen genutzt werden und die Überprüfungsdefinitionen können auch in einem geführten UI erfasst werden.

2.2.4. Vor- und Nachteile der Varianten

Um Vor- und Nachteile aus den zwei Varianten ableiten zu können, musste ich definieren, was ich vergleiche. Die folgenden Eigenschaften bzw. Fragen habe ich benutzt um die Varianten zu vergleichen:

- Haben wir bereits Spezialisten bzw. KnowHow über den Standard?
- Können wir bereits E-Mails versenden mit dem Standard bzw. Applikation?
- Ist der Standard schon im Einsatz oder muss er noch eingerichtet werden?

Tabelle 5: Vor- und Nachteile von Lösungsvorschläge

Splunk Variante	Java Library Variante
+ Splunk und DB KnowHow vorhanden	– Kein KnowHow über JavaLibrary
+ Erstellung von Incidents und Versenden von E-Mails funktioniert bereits über Splunk	– E-Mail können gemäss Dokumentation mit der Library nicht versendet werden
+ Splunk bereits eingerichtet für ADW	– JavaLibrary muss eingerichtet werden

2.2.5. Handlungsempfehlung

Beide Varianten bieten den Vorteil, dass ein geführtes UI dazu entwickelt werden kann und wir uns an den PF-Standard halten. Trotzdem überwiegen die Nachteile der Java Library Variante den Vorteilen der Splunk Variante. Die Variante zu wählen, welche bereits eingerichtet ist und KnowHow besteht ist effizienter in der Umsetzung, als eine neue unbekannte Library in unsere Applikation zu integrieren.

Deshalb empfehle ich die Splunk-Variante zu benutzen.

2.2.6. Meeting mit Fachbetreuer

Mit meinen Erkenntnissen und der Handlungsempfehlung habe ich den Fachbetreuer zu einem Meeting eingeladen. (Abbildung 13)

Abbildung 13: Screenshot von Termineinladung mit Fachbetreuer

Scavetta Giuseppe, PF85

Betreff:	Besprechung: Lösungsvorschläge für neues Alarming-Konzept
Ort:	Skype-Besprechung
Beginn:	Fr. 10.09.2021 10:00
Ende:	Fr. 10.09.2021 10:30
Serientyp:	(Keine Angabe)
Besprechungsstatus:	Besprechungsorganisation
Organisation:	Scavetta Giuseppe, PF85
Erforderliche Teilnehmer:	Bregy Patrick, PF85

Hallo Patrick

Gerne möchte ich mit dir meine Lösungsvorschläge für das neue Alarming-Konzept besprechen.

Liebe Grüsse
Giuseppe

Der Fachbetreuer hat sich für die Splunk Variante entschieden und hat bestätigt, dass es 40% vom Funktionsumfang wiedergibt.

Folgende Inputs hat er mitgegeben für die weitere Ausarbeitung des Konzepts:

- Die SQL-Checks sollten einen Schwellenwert haben für die Dauer der Ausführung (Runtime/Timeout)
- Ein Worker soll eine Tabelle abarbeiten wobei die kritischen Überprüfungen Priorisiert werden. (Sequentieller Ablauf)
- Ein eigener DB-User sollte eingerichtet werden um die Ressourcen zu managen.
- Das Alarming-Framework sollte nie mehr Power haben als der Lade-Job.

Gemäss Fachbetreuer muss ich mich noch mit einem weiteren Entwickler austauschen, der sich aktuell mit dem Dataquality-Framework beschäftigt. Eventuell sollten wir einen Adapter auf das zukünftige Alarming-Framework in Betracht ziehen.

Mein Lösungsvorschlag sollte ich am Framework-Meeting mit den Hauptentwicklern besprechen und auf ihre Inputs achten bevor ich das Technische Konzept vertiefe.

Ebenfalls sollte ich noch einen Vorschlag bringen, wie wir aus der Datenbank Mails versenden können, ohne einen Mailingdienst bzw. Konfiguration auf dem Datenbank-Server einzurichten. Diese Bedingung hat er mir gestellt, weil wir aus Sicherheitsgründen keinen Mailingdienst oder Konfiguration auf dem DB-Server einrichten dürfen.

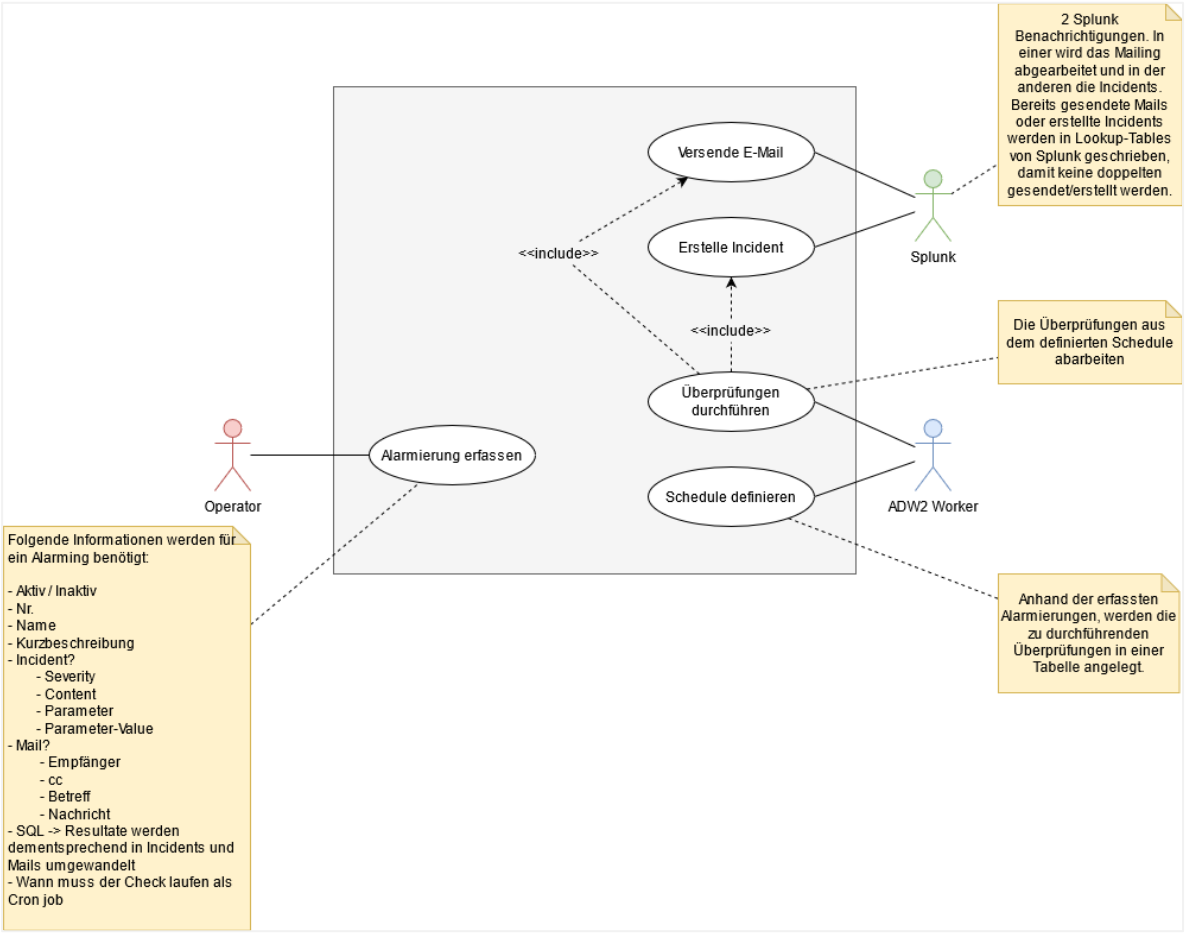
2.3. Technisches Konzept

Bevor ich mich mit dem Dataquality-Framework Entwickler, den Hauptentwicklern und jemand aus dem Operation-Team austausche, habe ich ein Use-Case-Diagramm erstellt um meine Lösungsstrategie zu erläutern.

2.3.1. Lösungsstrategie

In diesem Abschnitt erläutere ich den Ablauf der gewählten Variante zur Incident Erstellung über Splunk. Ebenfalls sind alle besprochenen Punkte aus den unterschiedlichen Meetings dokumentiert. Mit dem Use-Case-Diagramm aus Abbildung 14 habe ich die Variante an den verschiedenen Meetings vorgestellt.

Abbildung 14: Use-Case-Diagramm der Lösungsstrategie



2.3.1.1. Technologien

Folgende drei Technologien werden für meine Lösungsstrategie verwendet.

- Splunk für die Versendung von Mails und die Erstellung von Incidents
- ADW2-Datenbank (Oracle)
  - Tabellen für Metadaten (Überprüfungs-Definitionen)
  - Package mit der Überprüfungs-Funktionalität (Scheduling und Überprüfung)
    - Weitere Funktionen um in die Mailing- und Incident-Tabelle zu schreiben
- ADW2-Worker ist ein wiederkehrender Task im SOS Scheduler, welcher Funktionen aus dem Überprüfungs-Package in der Datenbank aufruft.

Der SOS-Scheduler ist in unserem DWH der Auslöser aller Abläufe, um uns an den Standard zu halten, nutzen wir den SOS-Scheduler auch im Alarming-Framework.

2.3.1.2. Ziele und Anforderungen

Aus dem Meeting mit dem Fachbetreuer habe ich die folgenden Ziele und Anforderungen abgeleitet.

Tabelle 6: Ziele und Anforderungen am Framework

Ziel/Anforderung	Vorgehensweise	Details
Es werden Tabellen abgearbeitet	Im Alarming-Package gibt es die Prozedur «DO_CHECK», welche eine Tabelle abarbeitet die vorher definiert wurde.	2.3.1.3 2.3.3.1 2.3.3.4
DB-Benutzer für Ressourcenmanagement	Es muss ein neuer Connection-Pool in der API definiert werden. Ebenfalls muss von den Datenbankadministratoren ein neuer Benutzer angelegt werden, den wir verwenden können. Das Ressourcenmanagement wird für die DBAs somit ermöglicht.	2.3.4.5
Housekeeping	Im Alarming-Package wird eine «HOUSEKEEPING»-Prozedur integriert, die bei einem Aufruf ältere Daten bereinigt.	2.3.3.2 Tabelle 7 Abbildung 31
Mailing aus DB ohne Konfiguration auf DB-Server	In der Datenbank wird eine Tabelle angelegt mit den zu versendenden E-Mails. Splunk arbeitet diese Tabelle ab.	2.3.3.5

2.3.1.3. Ablauf

In diesem Abschnitt werden der grobe Ablauf und die Zusammenhänge der einzelnen Komponenten bzw. Technologien nähergebracht.

1. Eine DB-Prozedur fügt in eine «To-do»-Überprüfungs-Tabelle die zu durchführenden Überprüfungen ein (anhand der Metadaten). Die Funktion wird vom SOS-Scheduler alle 15 Minuten aufgerufen. Warum alle 15 Minuten wird in Abschnitt 2.3.3.1 erklärt.
2. Eine weitere DB-Prozedur arbeitet die «To-do»-Überprüfungs-Tabelle sequentiell ab und ruft – wenn nötig – die erstelle Incident- oder versende E-Mail-Funktion auf (aus dem DB-Package). Dieser 2. Schritt (Abarbeitung «ToDo»-Tabelle) bzw. Job wird vom SOS-Scheduler aufgerufen, nachdem der vorherige Job (Schritt 1) abgeschlossen wurde.
3. Die erstelle Incident- und versende E-Mail-Funktion schreiben die zu erstellenden Incidents oder zu versendende E-Mails in jeweils eigenen Tabellen. Diese Prozeduren werden jeweils von der vorherigen Prozedur (Schritt 2) aufgerufen.
4. Splunk greift regelmässig auf diese Tabellen zu und arbeitet diese ab. Am Schluss von jeder Ausführung, wird von Splunk aus eine DB-Prozedur aufgerufen die dann das E-Mail als gesendet oder den Incident als erstellt meldet. Die Splunk Benachrichtigung wird alle zehn Minuten ausgeführt, warum wird in Abschnitt 2.3.3.5 erklärt.

2.3.1.4. Austausch Framework Meeting vom 16.09.2021

Am Framework Meeting waren die Hauptentwickler, Projektleiter und der Architekt unserer Applikation anwesend. Das Meeting wurde mit der Vorstellung des Use-Case Diagramms eröffnet.

Nach einigen Verständnisfragen waren alle mit der Strategie einverstanden und haben erwähnt, dass ich mich mit dem DQ-Framework Entwickler austauschen sollte bevor ich mit dem Konzept beginne.

2.3.1.5. Austausch mit Dataquality-Framework Entwickler 16.09.2021

Direkt nach dem Framework Meeting habe ich den DQ-Framework Entwickler über Skype kontaktiert.

Auch er ist mit der Strategie einverstanden und der Meinung, dass wir so nicht zwei Spurig fahren. Ebenfalls hat er angemerkt, dass in einem 2. Schritt Problemlos ein Adapter zwischen unsere Frameworks entwickelt werden kann.

2.3.1.6. Austausch mit Porteous Chloe (OP-Teamleitung) und Hunger Jürgen (Operator)

Am selben Tag habe ich das Operation-Team über den Stand informiert. Sie sind auf die Umsetzung gespannt und finden das E-Mail Feature sehr hilfreich.

Vor allem möchten sie ein UI um die Überprüfungen erfassen und handhaben zu können. Dies wird aber nicht in diesem Konzept behandelt.

2.3.2. Abhängigkeiten

Damit alles funktioniert, sind folgende Applikationen notwendig und voneinander Abhängig:

- JobScheduler von SOS *(ist der Auslöser aller Jobs auf der DB)*

  Abbildung 15: SOS JobScheduler Logo

  Ohne den JobScheduler, wird die Entwickelte Prozedur nicht aufgerufen. Das heisst, dass bei einem Ausfall von SOS Job Scheduler der Prozess stehen bleibt. Ein Ausfall betrifft dann auch alle anderen Abläufe und Prozesse auf der Datenbank.

- Java Spring *(REST-Schnittstelle, enthält alle Befehle, die von SOS ausgeführt werden)*

  Abbildung 16: Java Spring Logo

  Alle Prozeduren bzw. Jobs die vom SOS-Scheduler aufgerufen werden sind als REST-Endpunkt in unserer API definiert (Spring Framework). Dort würde sich der eigentliche Datenbank-Befehl befinden, der dann ausgeführt wird.

- Oracle Datenbank *(Speichert die Überprüfungs-Definitionen und führt die Überprüfungen aus)*

  Abbildung 17: Oracle Database Logo

  Auf der Oracle-Datenbank werden die Metadaten der Überprüfungen gespeichert bzw. definiert. Die gesamte Durchführungs-Logik der Überprüfungen befindet sich in einem PL/SQL-Package. Ohne Oracle-Datenbank, hat der SOS Job Scheduler bzw. API keine Prozedur zum Aufrufen und es wird keine Überprüfung stattfinden.

- Splunk *(liest aus der Datenbank die zu erstellenden Incidents aus und erstellt diese. Dasselbe gilt auch für das E-Mailing)*

  Abbildung 18: Splunk Logo

  Mit Splunk erstellen wir über den PF-Standard unsere Incidents und versenden E-Mails. Ohne Spunk werden die Tabellen (sende E-Mail oder erstelle Incident) auf der Datenbank nicht abgearbeitet und somit keine Aktion durchgeführt.

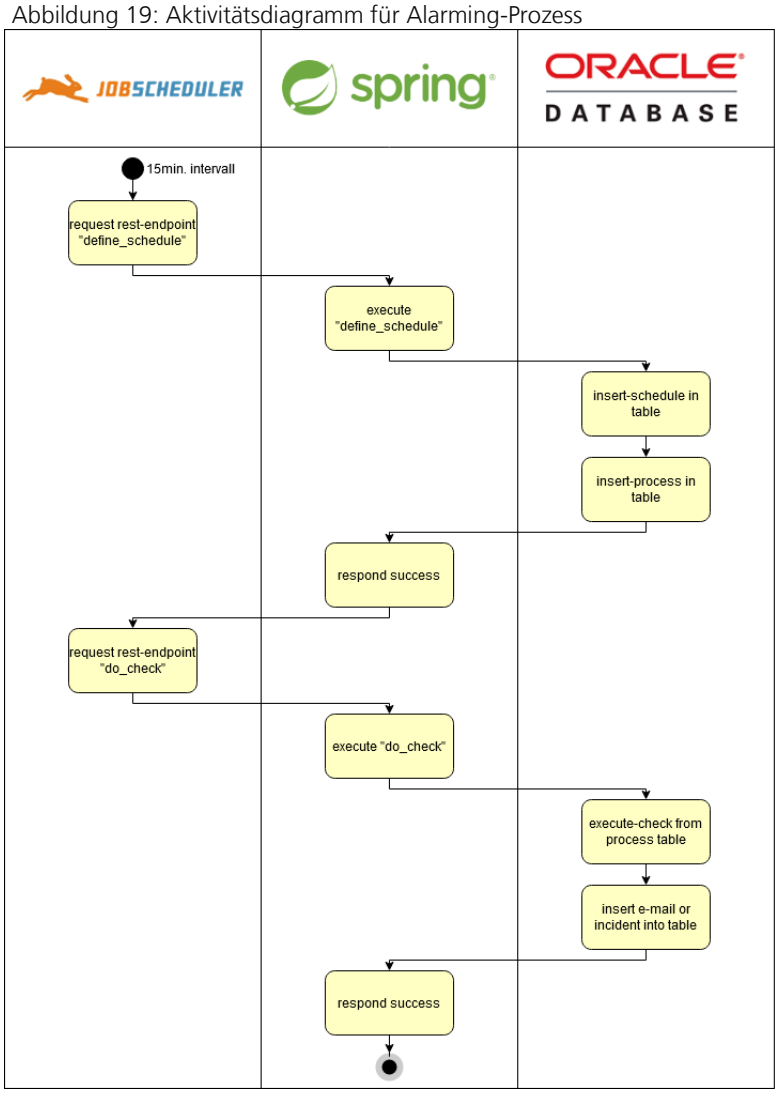
2.3.3. Laufzeitübersicht

Wir kennen die benötigten Technologien für das Framework. Wie läuft das ganze ab? Wie verhält und interagiert das System mit den einzelnen Technologien? Was muss bei den einzelnen Technologien definiert werden?

2.3.3.1. SOS Job Scheduler – Alarming Prozess

Wie in Abbildung 19 ersichtlich, ist der Auslöser bzw. Start im SOS Job Scheduler. Das heisst, dass wir **eine neue Job Kette** für das Alarming-Framework definieren müssen.

Abbildung 19: Aktivitätsdiagramm für Alarming-Prozess



Die Job Kette hat zwei Jobs, die nacheinander gestartet werden. Im ersten Schritt bzw. Job wird der Schedule definiert. Sobald die Ausführung erfolgreich abgeschlossen ist, wird in der Job Kette der 2. Job gestartet. Der 2. Job führt die eigentliche Überprüfung aus und arbeitet die Tabelle ab. Solange der Job läuft, wird die Job Kette nicht nochmal gestartet.

a. **Job Kette** **-> AlarmingProcessing** ***-> täglicher 15min. Intervall***
   1. Job -> DefineSchedule -> ruft REST-Endpunkt «DefineSchedule» auf
   2. Job -> DoCheck -> ruft REST-Endpunkt «DoCheck» auf

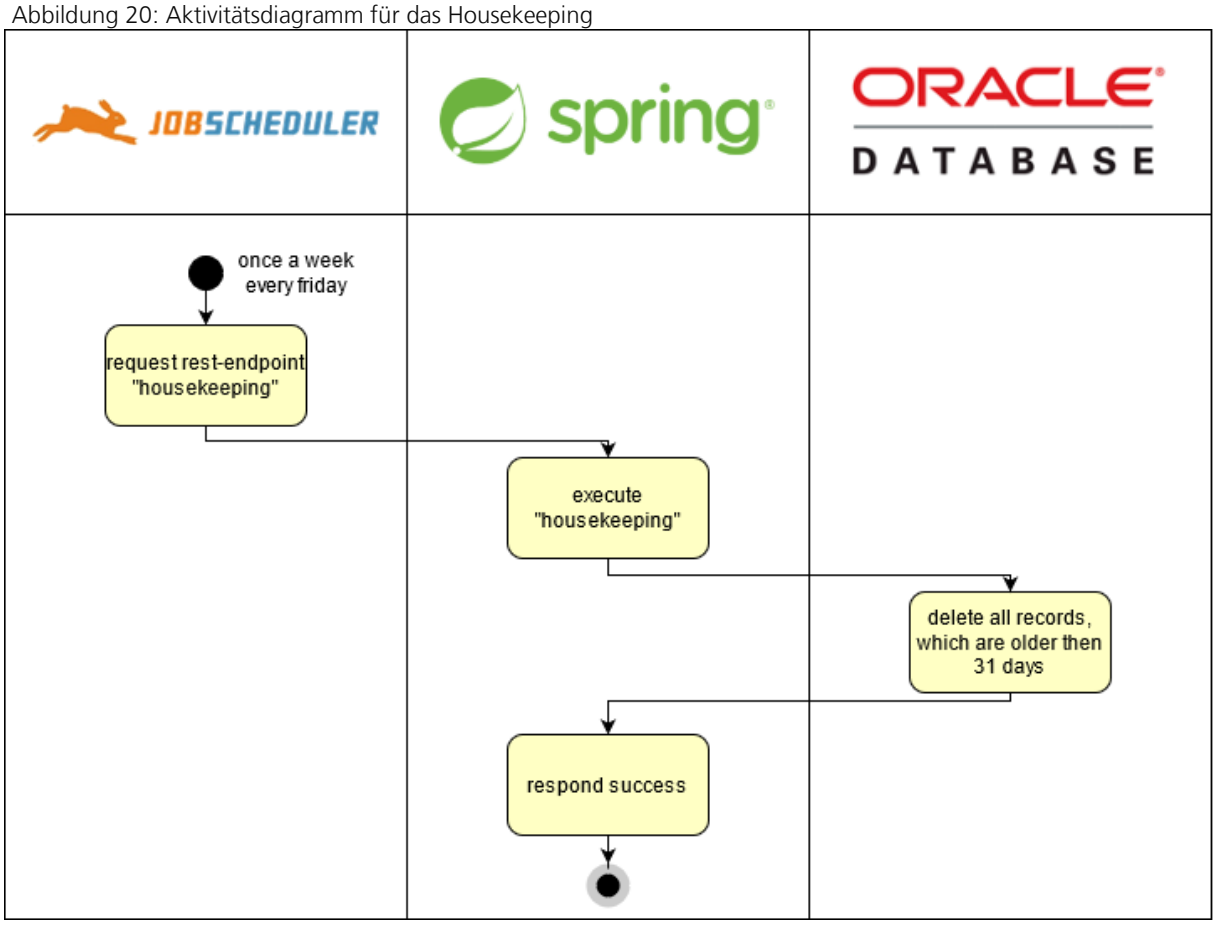
Da verschiedene Überprüfungszeitpunkte definiert werden können, muss die Job Kette in einem regelmässigen Intervall ausgeführt werden. Ich habe einen 15 Minuten Intervall ausgewählt, weil so vier Mal pro Stunde ein Schedule definiert wird und bestimmt ein grosser Teil der Definitionen abgedeckt wird. Das Intervall kann im Nachhinein optimiert werden.

2.3.3.2. SOS Job Scheduler – Housekeeping

Eine weitere Job Kette ist für das Housekeeping (Reinigung) notwendig. Bei ADW haben die Daten eine gewisse Lebenszeit. Es gibt grössere Tabellen (3.6 TB), welche die Daten für 2.5 Jahre in der Datenbank halten und nach der Lebenszeit im Datalake archivieren. Ziel ist es, zu verstehen wie lange man die Daten benötigt und nach der benötigten Zeit diese zu archivieren. Somit spart man unnötig besetzten Speicher und Performance auf der Datenbank.

Das Housekeeping sollte aber nicht nur für grosse Tabellen durchgeführt werden. Auch die Tabellen, die wenig Daten konsumieren, können mit der Zeit sehr viel Speicher und Performance beanspruchen. Aus diesem Grund wird auch im Alarming-Framework ein Housekeeping durchgeführt, siehe dazu die Abbildung 20.

Abbildung 20: Aktivitätsdiagramm für das Housekeeping



Wie lange sind die Ausführungsdaten vom Alarming-Framework für das Operation-Team relevant? Grundsätzlich sind diese für das Operation-Team nicht relevant, da ihnen nur das Endresultat (Mail oder Incident) interessiert. Relevant ist es nur um SQL-Überprüfungen zu identifizieren, die nicht performant genug sind (mehr dazu in Kapitel 2.3.6.5). Deshalb habe ich als Schwellenwert eine Datenhaltung von 31 Tagen definiert. Somit stehen bestimmt genügend Daten zur Verfügung um ineffiziente SQL-Überprüfungen zu identifizieren. Bei Bedarf kann man diesen erhöhen oder reduzieren.

b. Job Kette -> AlarmingHousekeeping -> *jeden Freitag um 19:00 Uhr*
1. Job -> CleanupTables -> ruft REST-Endpunkt «Housekeeping» auf

Nach meiner Operator Erfahrung bei ADW, hat die gesamte DB-Umgebung Freitagabends die meiste Kapazität für jegliche Tätigkeit. Deshalb habe ich den Schedule so definiert. Falls es aus uns unbekannten Gründen sich als Problematisch herausstellen sollte, kann die Startzeit problemlos angepasst werden.

2.3.3.3. REST-Endpoint in Java Spring

Es müssen drei REST-Endpunkte in unserer API definiert werden, die vom SOS Job Scheduler aufgerufen werden. Alle drei REST-Endpunkte führen Befehle auf der Datenbank aus. Sobald die Ausführung auf der DB fertig ist, wird dem SOS Job Scheduler eine Rückmeldung gegeben. Somit weiss die Job Kette wann sie fortfahren kann.

- REST-Endpunkt für «/api/monitoring/alarming/DefineSchedule»
  - Dieser Endpunkt ruft die Prozedur auf der Datenbank auf, um den Schedule zu definieren.

- REST-Endpunkt für «/api/monitoring/alarming/DoCheck»
  - Bei diesem Endpunkt wird die Prozedur auf der Datenbank aufgerufen, welche die Überprüfung durchführt.

- REST-Endpunkt für «/api/monitoring/alarming/Housekeeping»
  - Aufruf von Housekeeping-Prozedur auf der Datenbank. Bereinigt die Alarming-Framework Tabellen.

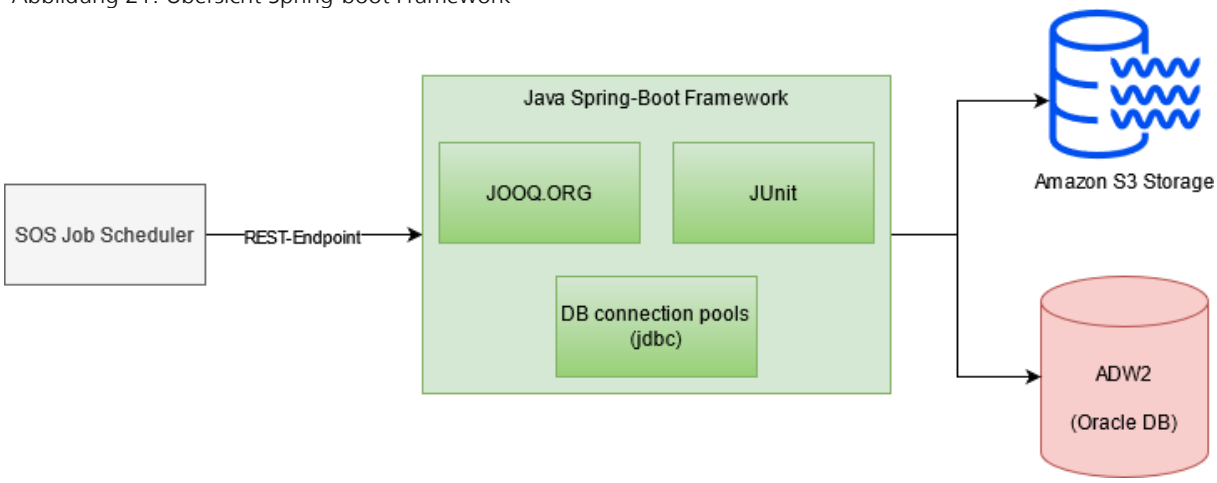
Da ich zum ersten Mal bei ADW mit dem Spring-Boot-Framework in Verbindung komme, ist es für mich sehr neu. Leider gibt es auch noch keine Confluence-Seite, die unser Spring Konzept erklärt, aber ich hatte eine kurze und oberflächliche Einführung mit dem Hauptentwickler.

Bei ADW haben wir Spring Boot im Einsatz und bei jedem Release wird das Framework mit der neusten Version aktualisiert.

Es gibt keine Namenskonvention für die Pfade. Der Klassen-, Packagename und URL-Pfad sollten aber übereinstimmen bzw. Ähnlichkeit haben. Die einzige Regel ist, dass man im Pfad als erstes Verzeichnis «api» angibt und anschliessend die Klasse an einem sinnvollen Ort anlegt.

Da wir keine Mikro-services für externe Stakeholder anbieten, haben wir keine Versionsverwaltung für die URL-Pfade. Sobald es eine Änderung gibt, werden beim Release alle betroffenen Komponenten *(SOS Scheduler, REST-Endpunkt im Spring Framework und DB)* angepasst bzw. deployt.

Abbildung 21: Übersicht Spring-boot Framework



Wie in Abbildung 21 zu sehen ist, haben wir in unserem Java Spring-Boot Framework Klassen (jooq, jdbc usw.) die wir für unsere Prozesse benötigen. Wenn ein REST-Endpunkt aufgerufen wird, können wir von der API aus auf der Datenbank oder sogar im Amazon S3 Storage arbeiten. Das Framework gibt dem SOS Job Scheduler immer ein Feedback zurück.

2.3.3.4. Prozeduren auf der Datenbank

Die Prozeduren «DEFINE_SCHEDULE», «DO_CHECK» und «HOUSEKEEPING» sind für den SOS Job Scheduler bzw. REST-API die wichtigsten, weil diese von ihm aufgerufen werden. Alle anderen Prozeduren werden von «DEFINE_SCHEDULE», «DO_CHECK» oder anderen Prozeduren auf der Datenbank verwendet.

Tabelle 7: Package Header für Alarming-Framework

Name	Parameter	Beschreibung
CREATE_INCIDENT	*PAR_SEVERITY* *PAR_PARAMETER* *PAR_PARAMETER_VALUE* *PAR_CONTENT* *PAR_ALARM_ID* *PAR_DEDUP_KEY* *PAR_PREDECESSOR_RUN_ID*	Die Eingabe vom Benutzer wird validiert und bei gültiger Eingabe in die Incident-Tabelle eingefügt.
INCIDENT_CREATED	*PAR_ALARM_INC_ID* *PAR_SPLUNK_TS*	Der Erstell-Zeitpunkt von einem spezifischen Incident (Splunk Zeitstempel) wird in die Incident-Tabelle eingefügt.
SEND_MAIL	*PAR_RECIPIENT* *PAR_CC* *PAR_SUBJECT* *PAR_MESSAGE* *PAR_PREDECESSOR_RUN_ID*	Die Eingabe vom Benutzer wird validiert und bei gültiger Eingabe in die Mail-Tabelle eingefügt.
MAIL_SENT	*PAR_ALARM_MAIL_ID* *PAR_SEND_TS*	Der Sende-Zeitpunkt von einem spezifischen Mail (Splunk Zeitstempel) wird in die Mail-Tabelle eingefügt.
DEFINE_SCHEDULE		Hier wird der Schedule für das Alarming-Framework definiert. Die Prozedur schaut mit der aktuellen Ausführungs-Uhrzeit auf die Definitionen und fügt in die Prozess-Tabelle Aufgaben hinzu. *(Aufgaben = durchzuführende Überprüfungen)*
DO_CHECK		Mit dieser Prozedur werden die Überprüfungen für das Alarming-Framework durchgeführt. Die Prozedur schaut auf die MD_AL_PROCESS Tabelle und arbeitet diese ab, solange kein anderer Eintrag mit dem Status "RUNNING" vorhanden ist.
HOUSEKEEPING		Löscht alle Einträge aus den Framework-Tabellen, die älter als 31 Tage sind. *(vom Ausführungszeitpunkt aus)*

2.3.3.5. Benachrichtigungen in Splunk

Splunk versendet die E-Mails und erstellt die Incidents, die in den DB-Tabellen angelegt werden. Dazu müssen zwei Benachrichtigungen in Splunk angelegt werden (Abbildung 22 und Abbildung 23). Beide Benachrichtigungen haben so ziemlich denselben Ablauf, sie fragen nur unterschiedliche Tabellen ab und rufen unterschiedliche Prozeduren auf.

Abbildung 22: Aktivitätsdiagramm für Splunk E-Mail

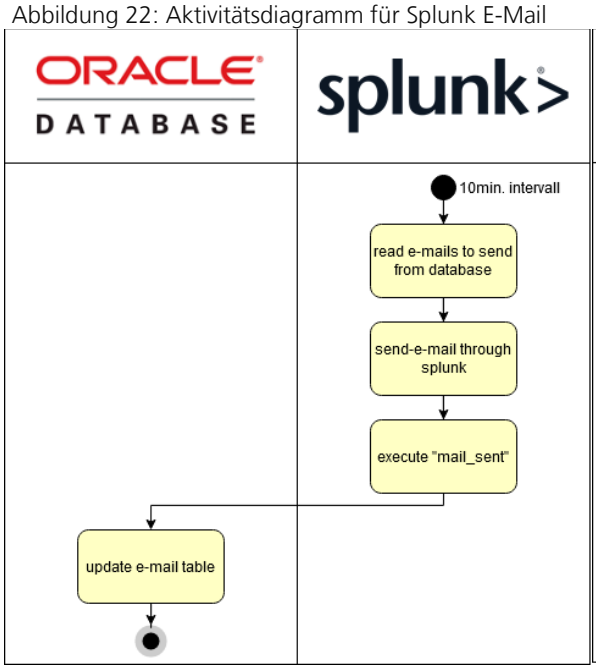
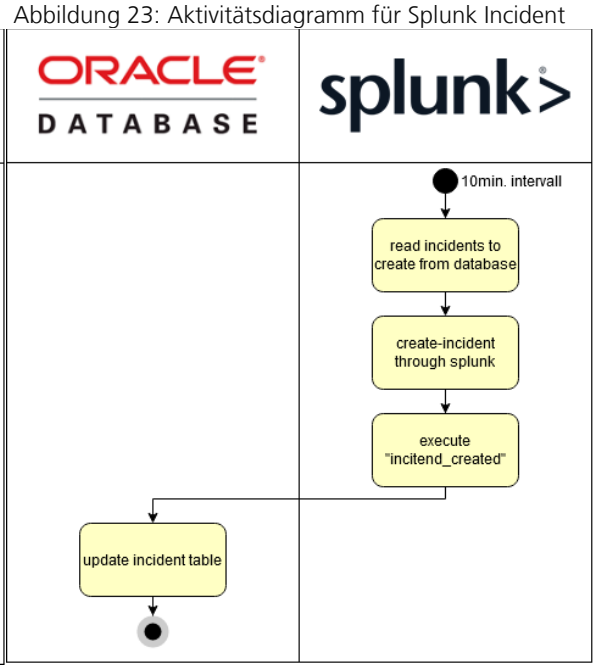


Abbildung 23: Aktivitätsdiagramm für Splunk Incident



1. **E-Mail Splunk Benachrichtigung**
   a. Holt sich die zu versendenden E-Mails aus der Datenbank.
   b. Splunk versendet die E-Mails.
      i. E-Mail-Felder mit dem Event befüllen (dynamisch)
   c. Splunk ruft eine DB-Funktion auf, die das E-Mail als gesendet markiert.

2. **Incident Splunk Benachrichtigung**
   a. Holt sich die zu erstellenden Incidents von der Datenbank.
   b. Splunk erstellt die Incidents indem er das Event.sh-Skript aufruft.
   c. Splunk ruft eine DB-Funktion auf, die das Incident als erstellt markiert.

Das Event.sh erwartet gewisse Informationen, damit der Incident an der TrueSight-Konsole übergeben werden kann:

Tabelle 8: Beschreibung von Pflichtfeldern von Incidenterstellung

Feld	Beschreibung	Beispiel
mon_alarm_id	Unique ID mit welcher der Event identifiziert wird.	ADW-236
host	Der Name des Hosts ohne Domain alias.	p1-app-abcd111
mon_severity	Priorisierung vom Incident	CRITICAL, MAJOR, WARNING oder INFO
mon_object_class	Kategorisierung der Meldung	APPLICATION, OS, FILESYSTEM, CPU, DISK, PROCESS, SERVICE oder DB
mon_object	Teilapplikation (ITAM Abkürzung)	ADW
mon_parameter	Beschreibung (Keywords) des betroffenen Objekts.	Fehlgeschlagener Job im Outboundbereich
mon_parameter_value	Wert oder Name des betroffenen Objekts	JOB_xy ist am xx.xx.xxxx um xx:xx mit Run-ID xy
content_	Frei von Timestamps und eine inhaltlich sinnvolle Beschreibung des Events.	JOB_xy ist mit Run-ID xy Fehlgeschlagen
mon_dedup_key	Der dedupKey wird für die Gruppierung von selben Events verwendet.	$host.$object_class.$object.$alarm_id.$content

Die meisten Informationen werden aus den Überprüfungs-Definitionen (MD_AL_DEFINITION) entnommen.

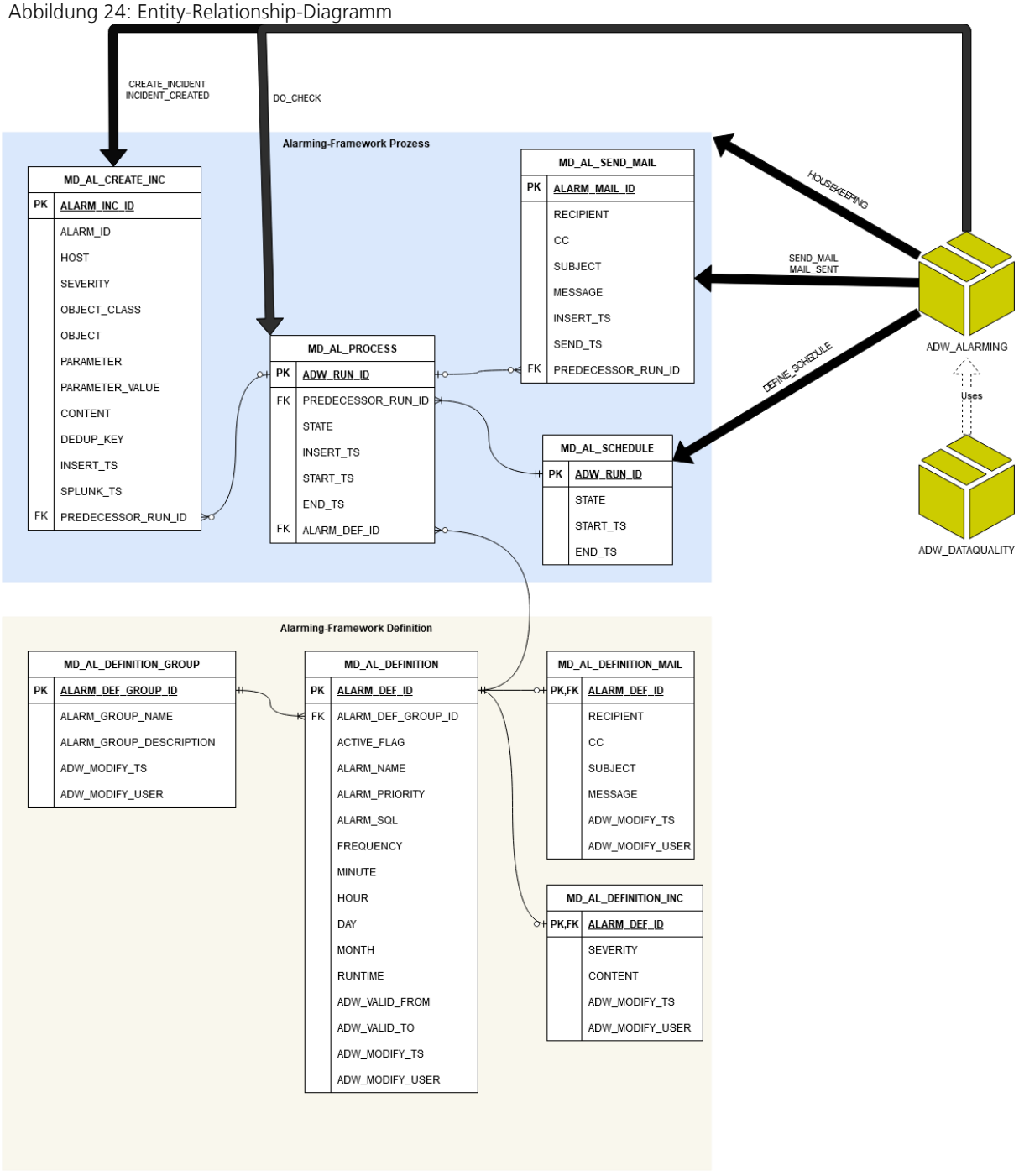
2.3.4. Datenbank

Wir wissen, was es bei allen anderen Abhängigen Technologien zu erstellen gibt. Es ist noch unklar was wir auf der Datenbank genau erstellen müssen. Dies wird in diesem Abschnitt behandelt. Alles wird auf dem ADW_META-Schema angelegt, weil sich auf diesem Schema die Frameworks und Metadaten vom DWH befinden.

2.3.4.1. ERD

Es war noch die Rede vom Dataquality-Framework, welches ebenfalls Mails versenden muss. Im ERD (Abbildung 24) habe ich das Dataquality-Package eingebunden, zum Aufzeigen, dass es ebenfalls die «SEND_MAIL»-Prozedur verwenden kann.

Abbildung 24: Entity-Relationship-Diagramm



2.3.4.2. Datenmodell

Bei der Benennung der Tabellen wurde die Namenskonvention von PostFinance eingehalten. «MD» steht für «Meta Data» und alle Objekte im «ADW_META»-Schema werden so benannt. Das «AL» wurde von mir definiert und weist auf die «Alarming»-Tabellen hin.

Tabelle 9 beinhaltet die Definition der Überprüfung mit dem SQL-Statement. Ebenfalls wird hier der Name, Alarm-ID, Schedule, Gruppe, Priorität und Zustand definiert.

Tabelle 9: Datenmodell für MD_AL_DEFINITION

MD_AL_DEFINITION				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ALARM_DEF_ID	NUMBER	FALSE	PK	SEQ_ALARM_DEF_ID => Sequenz um IDs zu generieren.
ALARM_DEF_GROUP_ID	NUMBER	FALSE	FK	
ACTIVE_FLAG	VARCHAR2 (1 Char)	FALSE		Constraint only 1 or 0
ALARM_NAME	VARCHAR2 (250 Char)	FALSE		
ALARM_PRIORITY	NUMBER	FALSE		Constraint only 1, 2 or 3
ALARM_SQL	CLOB	FALSE		
FREQUENCY	VARCHAR2 (7 Char)	FALSE		Constraint only HOURLY, DAILY, MONTHLY or YEARLY.
MINUTE	NUMBER	FALSE		Constraint only 0 to 59
HOUR	NUMBER	TRUE		Constraint only 0 to 23
DAY	NUMBER	TRUE		Constraint only 1 to 31
MONTH	NUMBER	TRUE		Constraint only 1 to 12
RUNTIME	VARCHAR(100)	TRUE		Constraint pattern "xx,xx,xx…"
ADW_VALID_FROM	TIMESTAMP(6)	FALSE		
ADW_VALID_TO	TIMESTAMP(6)	FALSE		
ADW_MODIFY_USER	VARCHAR2 (128 Char)	FALSE		
ADW_MODIFY_TS	TIMESTAMP(6)	FALSE		

Tabelle 10 beinhaltet die Gruppen bzw. Kategorien der Alarm-Definitionen.

Tabelle 10: Datenmodell für MD_AL_DEFINITION_GROUP

MD_AL_DEFINITION_GROUP				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ALARM_DEF_GROUP_ID	NUMBER	FALSE	PK	SEQ_ALARM_DEF_GROUP_ID => Sequenz um IDs zu generieren.
ALARM_GROUP_NAME	VARCHAR2 (250 CHAR)	FALSE		
ALARM_GROUP_DESCRIPTION	VARCHAR2 (300 CHAR)	TRUE		
ADW_MODIFY_USER	VARCHAR2 (128 Char)	FALSE		
ADW_MODIFY_TS	TIMESTAMP(6)	FALSE		

Tabelle 11 definiert das zu versendende E-Mail. Ein E-Mail kann nur zu einer Alarm-Definition gehören. Im Message-Feld können auch Platzhalter definiert werden, die dann von der SQL-Überprüfung befüllt werden.

Tabelle 11: Datenmodell für MD_AL_DEFINITION_MAIL

MD_AL_DEFINITION_MAIL				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ALARM_DEF_ID	NUMBER	FALSE	PK,FK	
RECIPIENT	VARCHAR2 (2000 Char)	FALSE		
CC	VARCHAR2 (2000 Char)	TRUE		
SUBJECT	VARCHAR2 (70 Char)	FALSE		
MESSAGE	VARCHAR2 (2000 Char)	FALSE		
ADW_MODIFY_USER	VARCHAR2 (128 Char)	FALSE		
ADW_MODIFY_TS	TIMESTAMP(6)	FALSE		

Tabelle 12 definiert das zu erstellende Incident. Ein Incident kann nur zu einer Alarm-Definition gehören und das Content-Feld kann auch mit Platzhaltern gefüllt werden.

Tabelle 12: Datenmodell für MD_AL_DEFINITION_INC

MD_AL_DEFINITION_INC				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ALARM_DEF_ID	NUMBER	FALSE	PK,FK	
SEVERITY	VARCHAR2 (8 Char)	FALSE		Constraint only warning, major or critical
CONTENT	VARCHAR2 (4000 Char)	FALSE		Beschreibung des Events.
ADW_MODIFY_USER	VARCHAR2 (128 Char)	FALSE		
ADW_MODIFY_TS	TIMESTAMP(6)	FALSE		

Tabelle 13 protokolliert alle Ausführungen der Prozedur «DEFINE_SCHEDULE».

Tabelle 13: Datenmodell für MD_AL_SCHEDULE

MD_AL_SCHEDULE				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ADW_RUN_ID	NUMBER (18)	FALSE	PK	
STATE	VARCHAR2 (100 Char)	FALSE		Constraint only RUNNING, OK, FAILED
START_TS	TIMESTAMP(6)	FALSE		
END_TS	TIMESTAMP(6)	TRUE		

Tabelle 14 protokolliert die Ausgeführten Überprüfungen und beinhaltet die zu ausführenden Überprüfungen. Diese Tabelle wird von «DEFINE_SCHEDULE» und «DO_CHECK» verwendet.

Tabelle 14: Datenmodell für MD_AL_PROCESS

MD_AL_PROCESS				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ADW_RUN_ID	NUMBER (18)	FALSE	PK	
PREDECESSOR_RUN_ID	NUMBER (18)	FALSE	FK	
STATE	VARCHAR2 (100 Char)	FALSE		Constraint only INIT, RUNNING, OK, FAILED
INSERT_TS	TIMESTAMP(6)	FALSE		
START_TS	TIMESTAMP(6)	TRUE		
END_TS	TIMESTAMP(6)	TRUE		
ALARM_DEF_ID	NUMBER	FALSE		

Tabelle 15 beinhaltet die zu versendenden und die bereits versendeten E-Mails.

Tabelle 15: Datenmodell für MD_AL_SEND_MAIL

MD_AL_SEND_MAIL				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ALARM_MAIL_ID	NUMBER	FALSE	PK	SEQ_ALARM_MAIL_ID => Sequenz um IDs zu generieren.
RECIPIENT	VARCHAR2 (2000 Char)	FALSE		
CC	VARCHAR2 (2000 Char)	TRUE		
SUBJECT	VARCHAR2 (70 Char)	FALSE		
MESSAGE	VARCHAR2 (2000 Char)	FALSE		
INSERT_TS	TIMESTAMP(6)	FALSE		
SEND_TS	TIMESTAMP(6)	TRUE		
PREDECESSOR_RUN_ID	NUMBER (18)	TRUE	FK	

Tabelle 16 beinhaltet die zu erstellenden und die erstellten Incidents. Die Bedeutungen der Spalten kann in Tabelle 8 nachgeschlagen werden.

Tabelle 16: Datenmodell für MD_AL_CREATE_INC

MD_AL_CREATE_INC				
Name	**Datatype**	**Nullable**	**Key**	**Remark**
ALARM_INC_ID	NUMBER	FALSE	PK	SEQ_ALARM_INC_ID => Sequenz um IDs zu generieren.
ALARM_ID	VARCHAR (7 Char)	FALSE		
HOST	VARCHAR (250 Char)	FALSE		
SEVERITY	VARCHAR2 (8 Char)	FALSE		Constraint only warning, major or critical
OBJECT_CLASS	VARCHAR2 (11 Char)	FALSE		Constraint only APPLICATION
OBJECT	VARCHAR2 (3 Char)	FALSE		Constraint only ADW
PARAMETER	VARCHAR (250 Char)	FALSE		
PARAMETER_VALUE	VARCHAR (250 Char)	FALSE		
CONTENT	VARCHAR2 (4000 Char)	FALSE		
DEDUP_KEY	VARCHAR (250 Char)	FALSE		
INSERT_TS	TIMESTAMP(6)	FALSE		
SPLUNK_TS	TIMESTAMP(6)	TRUE		
PREDECESSOR_RUN_ID	NUMBER (18)	TRUE	FK	

2.3.4.3. Sequenzen

In einigen Tabellen gibt es Primärschlüssel, die von einer Sequenz in der Datenbank generiert werden müssen. Für folgende Tabellen braucht es eine Sequenz um die ID zu führen:

- MD_AL_DEFINITION NUMBER
- MD_AL_DEFINITION_GROUP NUMBER
- MD_AL_SEND_MAIL NUMBER
- MD_AL_CREATE_INC NUMBER

2.3.4.4. Struktogramme für Package Prozeduren

Für alle Prozeduren, welche in Abschnitt «2.3.3.4 Prozeduren auf der Datenbank» definiert und beschrieben worden sind, folgen die Struktogramme:

Abbildung 25: Struktogramm für "CREATE_INCIDENT"

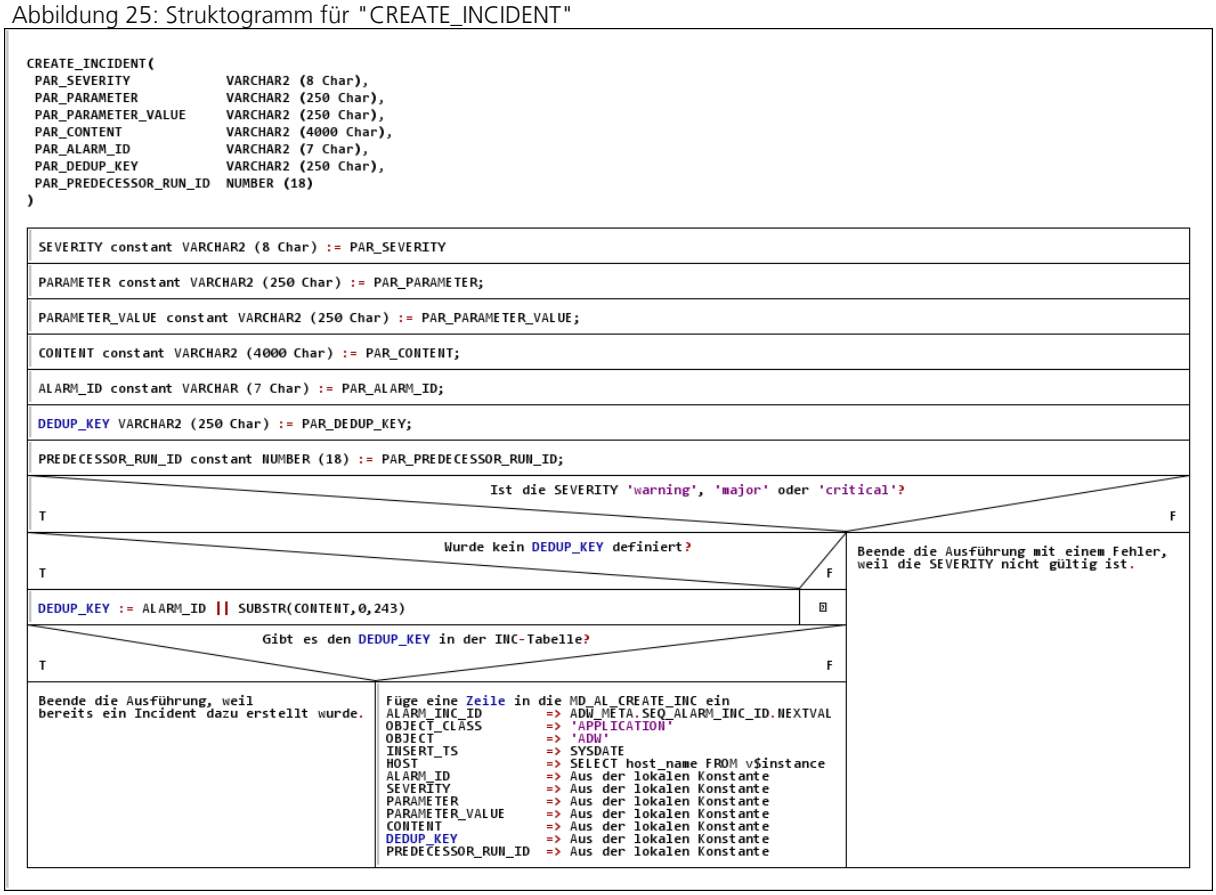


Abbildung 26: Struktogramm für "INCIDENT_CREATED"

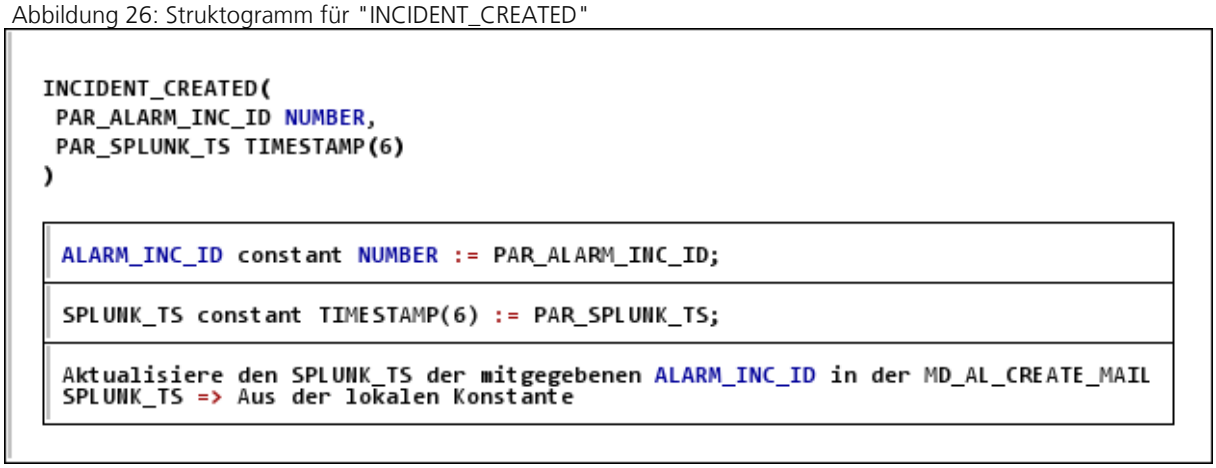


Abbildung 27: Struktogramm für "SEND_MAIL"

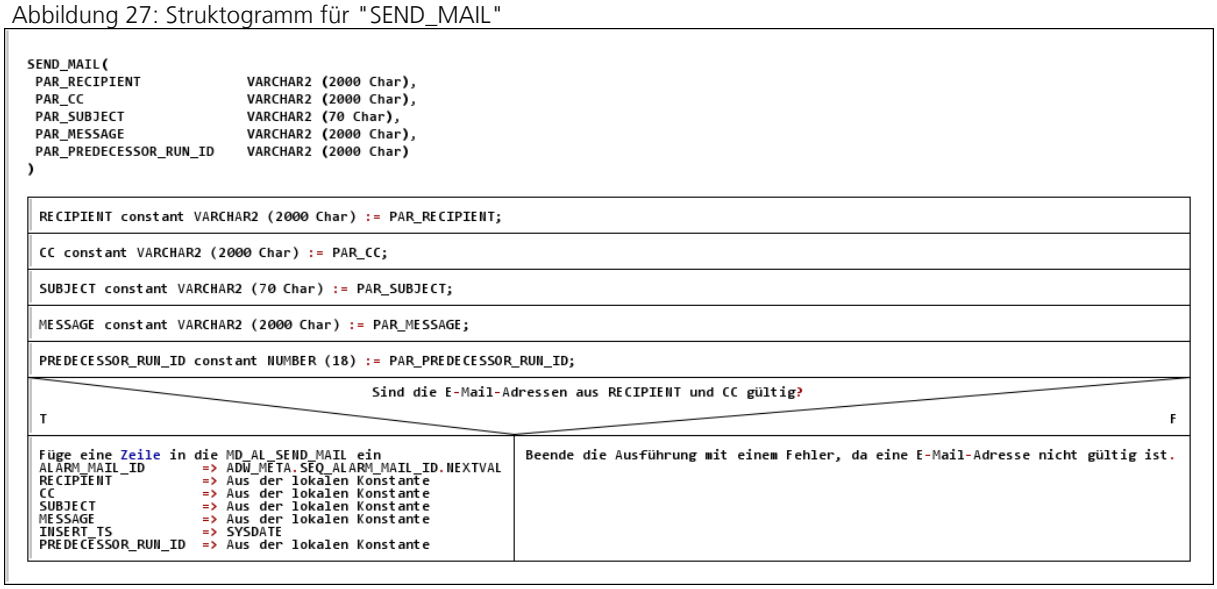


Abbildung 28: Struktogramm für "MAIL_SENT"

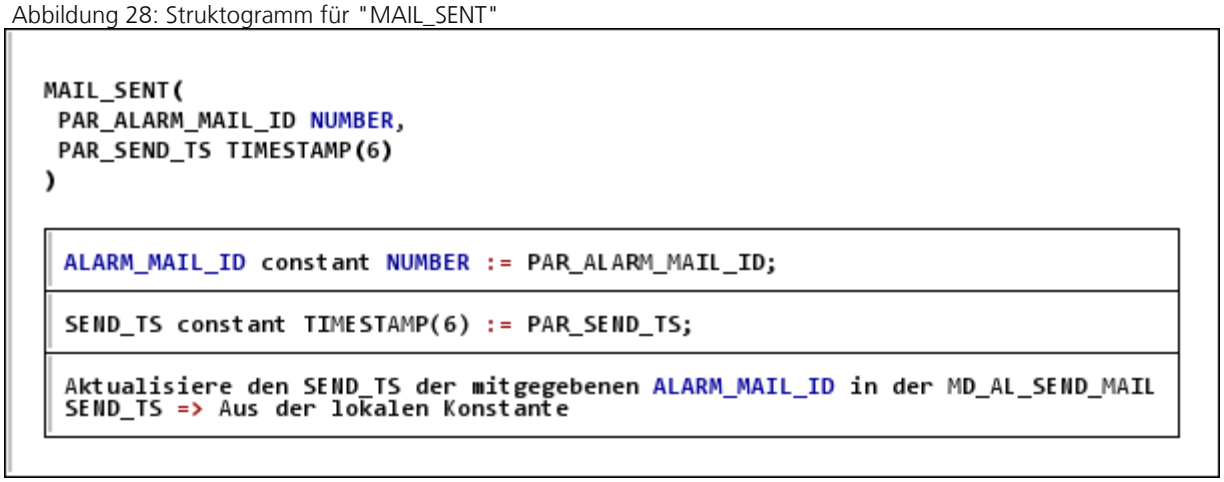


Abbildung 29: Struktogramm für "DEFINE_SCHEDULE"

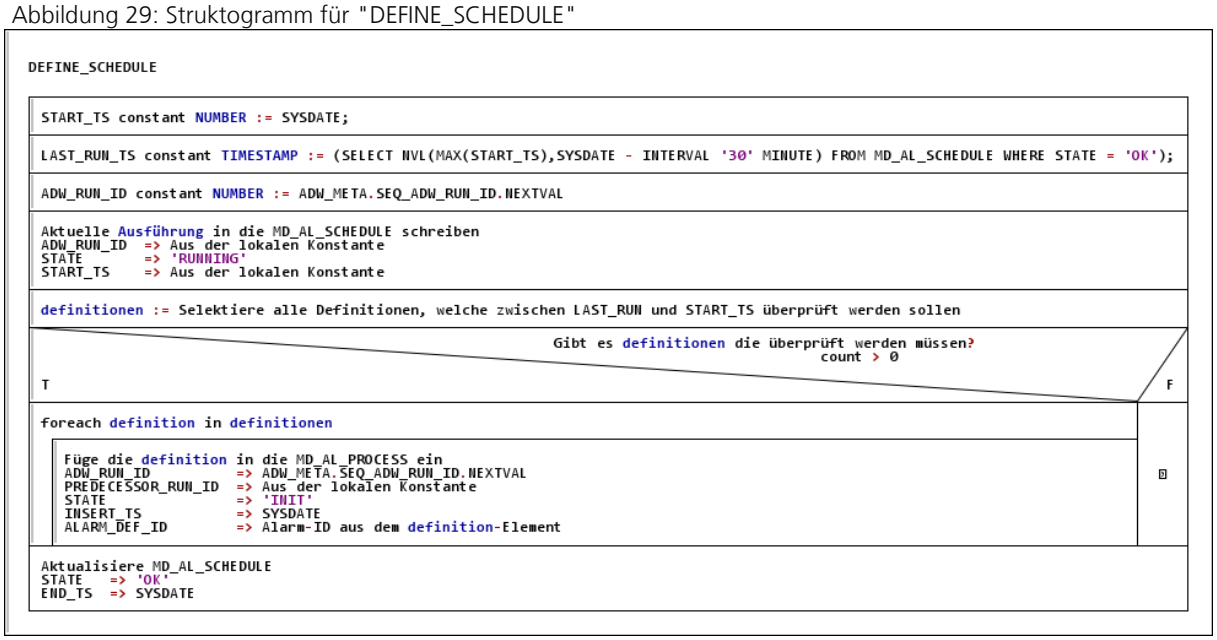


Abbildung 30: Struktogramm für "DO_CHECK"

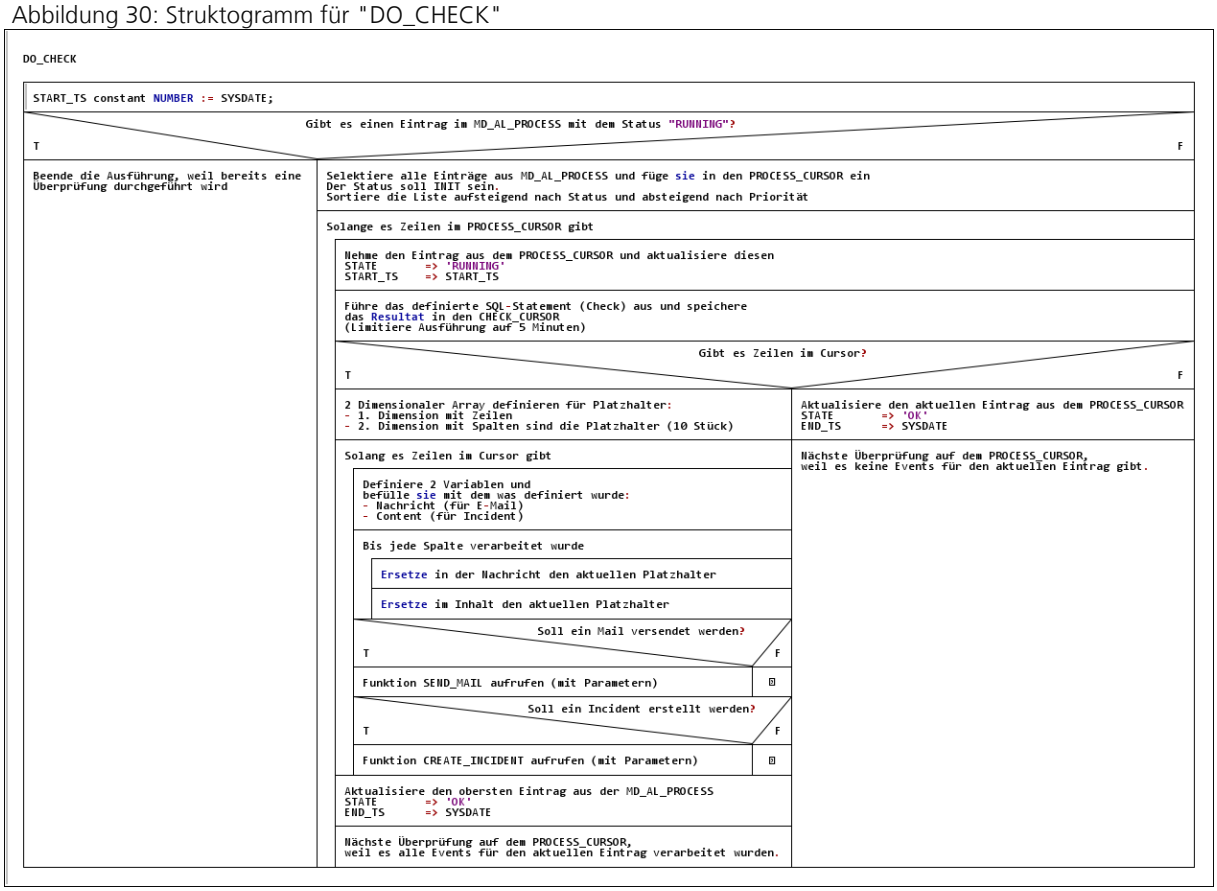
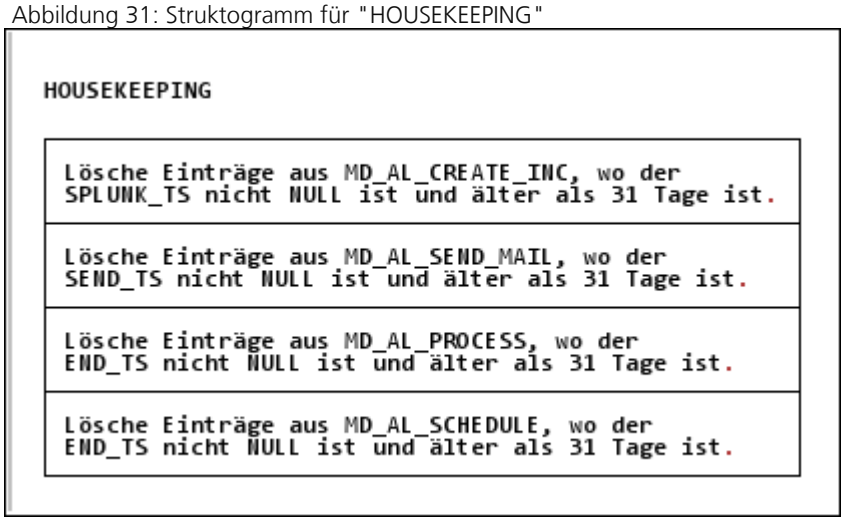


Abbildung 31: Struktogramm für "HOUSEKEEPING"



2.3.4.5. Benutzer

Wir benötigen einen neuen Benutzer auf der Datenbank, der die Prozeduren ausführt. Der Benutzer muss bei den Datenbankadministratoren (DBAs) angefragt werden und anschliessend auch in der API (Spring) definiert werden. Somit können die DBAs die verfügbaren Ressourcen für diesen Benutzer einschränken bzw. managen. Nach Absprache mit dem Hauptentwickler, ist so ein Benutzer sowieso notwendig, weil zurzeit alles mit einem Benutzer durchgeführt wird. Während der Umsetzung sollte man also den DBAs den Auftrag geben diesen Benutzer anzulegen. Mit den Hauptentwicklern muss man in der REST-API einen neuen Connection-Pool für diesen Benutzer erstellen und anschliessend sollten alle REST-Endpunkte von diesem Konzept diesen Connection-Pool verwenden.

Die Datenbank-Abfragen von Splunk, werden vom «PF_LOGMON»-Benutzer durchgeführt. Diesem Benutzer muss man anschliessend Ausführungs-Rechte auf das ADW_ALARMING-Package geben. Ansonsten funktioniert das Alarming-Framework nicht bzw. kann Splunk die E-Mails nicht als gesendet und die Incidents nicht als erstellt markieren.

2.3.5. Deploystrategie

Alle benötigten Technologien sind bereits installiert. Um dieses Technische Konzept umsetzten zu können, braucht man einen Jira-Task mit der Beschreibung von diesem Konzept. Anschliessend kann ein neuer Branch in Git angelegt werden (der zu der Jira Nummer gehört) um dieses Framework zu entwickeln.

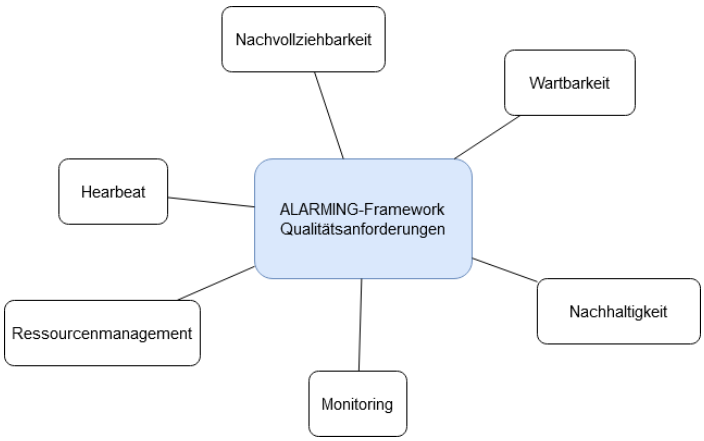
Nachdem alles Entwickelt wurde, wird ein Pull-Request erstellt und die Hauptentwickler überprüfen den Code. Wenn es keine weiteren Anmerkungen gibt, wird es mit dem Branch für das nächste Release zusammengeführt.

Nach der Zusammenführung, wird das Framework auf der Testumgebung verfügbar bzw. deployt und kann getestet werden. Falls es keine weiteren Änderungen gibt, kann man es so belassen und es wird für den nächsten Release eingeführt.

Geplant ist, dass dieses Framework für das Release 22A (12.02.2021) eingeführt wird. Das heisst, dass der Entwickler dieses Framework auf dem 221.000-Branch entwickeln muss.

2.3.6. Qualitätsanforderungen

Abbildung 32: Quality-Tree für Qualitätsanfoderungen



2.3.6.1. Nachvollziehbarkeit

Für das Operation-Team ist es wichtig, dass die Alarm-Definitionen einfach und nachvollziehbar sind. Grundsätzlich wird das bestimmt übersichtlicher als bei Splunk, aber diese Qualitätsanforderung kann mit einem UI besser befriedigt werden. Die Erfassung würde nicht direkt auf der Datenbank stattfinden und man kann eine Übersicht gestalten die den Bedürfnissen des Operation-Teams entspricht.

2.3.6.2. Wartbarkeit

Diese Anforderung kommt ebenfalls vom Operation-Team. Bei einem Release müssen sie oft die Alarmings deaktivieren. Wie bereits erwähnt ist das sehr umständlich mit Splunk. Jetzt können sie die Alarmings bzw. Definitionen auf der Tabelle mit einem Update-Statement deaktivieren. In einem 2. Schritt kann das bestimmt auch in das Cockpit eingebunden werden.

2.3.6.3. Ressourcenmanagement

Der Fachbetreuer bzw. Architekt möchte, dass dieses Framework nie mehr Ressourcen als der ETL-Prozess/-Benutzer hat. Indem wir einen weiteren Benutzer auf der Datenbank anlegen und diesen nutzen, ermöglichen wir den DBAs das Ressourcenmanagement. Somit können wir diese Qualitätsanforderung erfüllen.

2.3.6.4. Nachhaltigkeit

Das Framework erzeugt nicht extrem viele Daten, aber es ist wichtig ein Housekeeping zu betreiben. Mit der Zeit können sich da viele Daten ansammeln und unnötig Speicher/Performance beanspruchen. Diese Qualitätsanforderung wird erfüllt, da eine Prozedur miteingeplant wurde, die wöchentlich durchgeführt wird. Falls weitere Tätigkeiten beim Housekeeping für das Alarming-Framework getätigt werden müssen, kann die Prozedur beliebig erweitert und verändert werden.

2.3.6.5. Monitoring

In der ersten Ausbaustufe von diesem Framework ist keine Parallelisierung geplant. Da die Ausführungszeiten der SQLs nicht eingeschränkt werden können, besteht das Risiko, dass ineffiziente Überprüfungen den Prozess verlangsamen. Deshalb sollten die Ausführungszeiten beobachtet werden, um ineffiziente SQLs zu identifizieren und verbessern.

2.3.6.6. Heartbeat

Mit einem unabhängigen Prozess soll geprüft werden, wann die letzte Überprüfung durchgeführt wurde. Diese Information soll im Betriebscockpit dargestellt werden, um festzustellen, ob das Alarming-Framework noch funktioniert bzw. läuft.

2.3.7. Risiko/Technische Schulden

Nun folgt eine Liste mit Risiken oder technischen Schulden dieses Frameworks.

- Die Eingabe der Definitionen auf der Datenbank wird nicht validiert
  - Das definierte SQL-Statement vom Benutzer gibt nicht die korrekte Anzahl Spalten zurück.
  - Die Definition wird auf der Datenbank unvollständig erfasst.

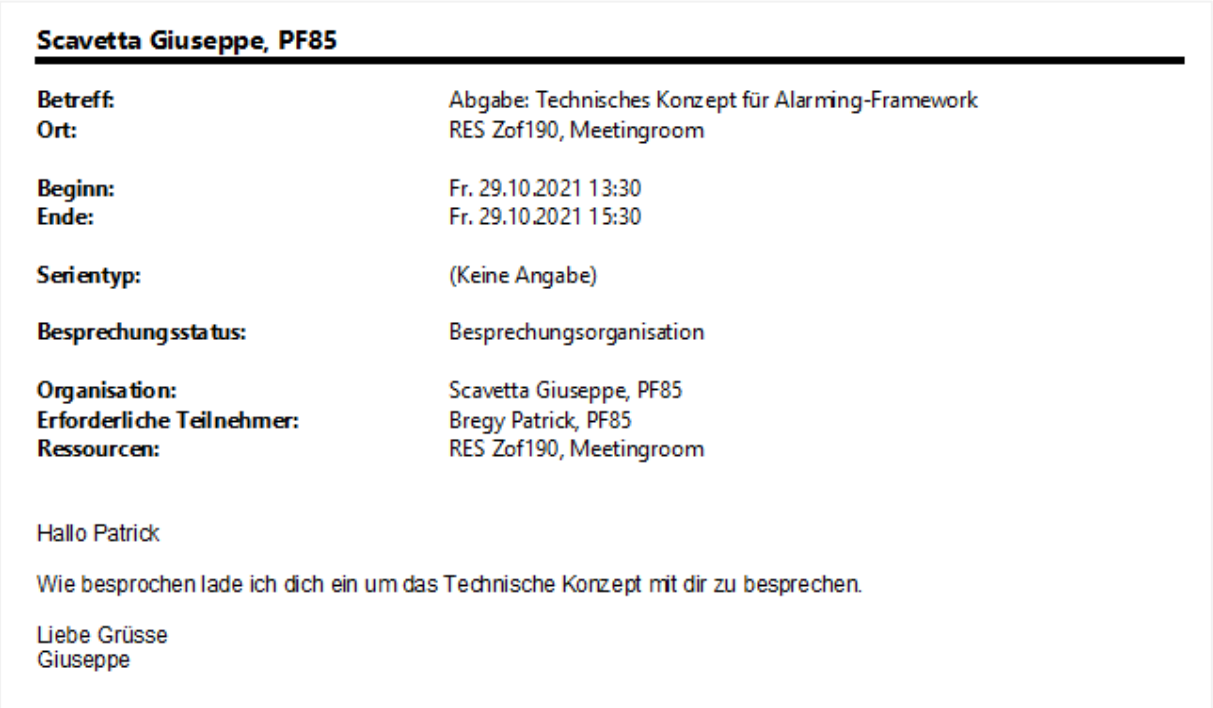
- Datenbank, Job Scheduler, Spring oder Splunk ist offline => Überprüfungen laufen nicht.

- Die einzelnen Durchführungsschritte werden nicht geloggt, erschwert Fehleranalyse.

2.3.8. Meeting mit Fachbetreuer

Das Technische Konzept ist vollständig, deshalb habe ich den Fachbetreuer zu einem Termin eingeladen (Abbildung 33) um das Konzept abzugeben.

Abbildung 33: Screenshot Termineinladung mit Fachbetreuer für Vorstellung von Konzept



Scavetta Giuseppe, PF85

Betreff:	Abgabe: Technisches Konzept für Alarming-Framework
Ort:	RES Zof190, Meetingroom
Beginn:	Fr. 29.10.2021 13:30
Ende:	Fr. 29.10.2021 15:30
Serientyp:	(Keine Angabe)
Besprechungsstatus:	Besprechungsorganisation
Organisation:	Scavetta Giuseppe, PF85
Erforderliche Teilnehmer:	Bregy Patrick, PF85
Ressourcen:	RES Zof190, Meetingroom

Hallo Patrick

Wie besprochen lade ich dich ein um das Technische Konzept mit dir zu besprechen.

Liebe Grüsse
Giuseppe

Der Fachbetreuer war positiv über den Umfang meines Konzepts überrascht. Ich habe nur im Datenmodell die MODIFY_TS und MODIFY_USER Spalten in den Definitions-Tabellen ergänzt. Ansonsten hat er das Konzept so angenommen und findet, dass man es zur Entwicklung freigeben kann.

3. Abschluss

Das Technische Konzept wurde dem Fachbetreuer abgegeben und er ist zufrieden damit. Nun folgt der Abschluss meiner Diplomarbeit.

3.1. Mehrwert dieser Arbeit

In diesem Abschnitt hinterfrage ich selber den Mehrwert dieser Arbeit und begründe diesen.

3.1.1. Fragestellung

Damit ich mir eine passende Frage stellen konnte, musste ich einen wichtigen Punkt ausblenden. Meine Arbeit behandelt auch das Mailen aus der Datenbank und dies ist eine Kernanforderung an der Applikation, der noch nicht umgesetzt wurde. Deshalb wäre die Frage ob wir mit diesem Konzept einen Mehrwehrt haben etwas fehl am Platz.

Trotzdem habe ich – unter Berücksichtigung diesen Punkt auszublenden – mir dazu die folgende Frage gestellt:

« Lohnt sich die Umsetzung dieses Konzepts bis zum RE22A, hinsichtlich der aufzuwendenden Zeit gegenüber der einzusparenden Zeit bei der Incidentbearbeitung? »

3.1.2. Beantwortung der Fragestellung

Um diese Frage zu beantworten muss ich mir die Frage stellen, wie ich das Messen möchte? Auf unserer Oracle Umgebung gibt es nur eine Überprüfung über Splunk, die ein Log-File ausliest und jeden abgebrochenen Order aus SOS Job Scheduler in einen Incident umwandelt. Diese generische Überprüfung erzeugt unnötig viele Incidents und kostet das Operation-Team Zeit. Ein abgebrochener Order heisst nicht immer, dass wir einen Fehler haben.

Jedenfalls habe ich im Zeitraum vom 02.10.2021 bis 03.11.2021 alle erstellten Incidents *(von der erwähnten Überprüfung)* analysiert und als Basis für die Hochrechnung im Jahr 2021 verwendet.

Tabelle 17: Berechnung von verlorener Zeit bei Incidentbearbeitung

	02.10.21 bis 03.11.21	**Hochrechnung 2021**
Anzahl Incidents	**62**	**721**
– Offen	3	35
– Überflüssig, weil Workaround vorhanden	35	407
– Bekannte Fehler (Ticket bereits vorhanden)	18	209
– Brauchen Bugfix (echte Fehler)	6	70
Arbeitstage	22	256
Zeit pro Inc. (Min.)	10	10
Verlorene Zeit durch Überflüssige Incidents und bekannte Fehler (Std.)	**8.83**	**102.67**

Wie in Tabelle 17 zu sehen, haben wir 62 Incidents in 22 Arbeitstagen erhalten. Es ist zu beachten, dass es eine eher ruhigere Phase bzw. Zeitspanne auf der Produktivumgebung ist. Das letzte Release war Mitte August, das heisst, dass die Spitze von den Incidents nach Release Einführung hier nicht als Bemessungsgrundlage verwendet wird.

Von den 62 Incidents sind 35 Überflüssig, weil diese weder bekannte noch neue Fehler sind. Diese 35 Incidents werden vom Operation-Team trotzdem analysiert und anschliessend abgeschlossen. Eine erste Analyse geht im Schnitt *(DB-Verbinden, Log Tabelle selektieren, lesen und verstehen, Incident bearbeiten)* rund zehn Minuten.

Daraus können wir den Entschluss ziehen, dass wir in 22 Arbeitstagen **8 Stunden und 49 Minuten** aufwenden um Fehler zu analysieren, die eigentlich gar keine sind. Auf das Jahr hochgerechnet – mit 256 Arbeitstagen – wären **das 102 Stunden und 40 Minuten**.

Dies ist eigentlich ein Hinweis darauf, dass wir präzisiere Überprüfungen brauchen um uns diesen Aufwand zu sparen oder die Möglichkeit brauchen Überprüfungen zu deaktivieren. Durch das neue Alarming-Framework können wir über die vorhandenen Metadaten auf der Datenbank sehr genaue Überprüfungen definieren, die dann ein Incident oder E-Mail auslösen.

Wiederum haben wir einen Aufwand dieses Alarming-Framework umzusetzen und auch die einzelnen Überprüfungen zu definieren. Da das gesamte Konzept vorhanden ist, rechne ich mit **1 Woche Entwicklungszeit** (5 Arbeitstage) um das Framework umzusetzen. Für die Anpassungen, die Kontrollen des Codes und Fehlerbehebung wird bestimmt **ein weiterer Arbeitstag** benötigt.

Leider kann ich nicht messen wie lange es geht eine Definition im Alarming-Framework anzulegen *(weil es das Framework nicht gibt)*, aber ich schätze mal **1 Stunde** *(SQL-Schreiben, Überprüfung definieren und testen)*. Diese Stunde sollte zukünftig in der Entwicklung von neuen zu überwachenden Schnittstellen miteingerechnet werden, weil jede Schnittstelle eine Überwachung braucht. Ebenfalls ist zu beachten, dass es ein einmaliger Aufwand ist die Überprüfung anzulegen und somit sich den Aufwand spart irrelevante Incidents zu bearbeiten.

Wir können nicht sagen, wie viele Überprüfungen angelegt werden müssen. Zurzeit haben wir drei Überprüfungen im Backlog, die auf das Framework warten. Es werden bestimmt mehr kommen, weil wir bis jetzt nur ca. 20% der Outbound Verarbeitungen von DB2 auf Oracle migriert haben und währenddessen auch neue entwickelt werden. Was auch ein Hinweis darauf ist, dass die verlorene Zeit eher eine steigende Tendenz beibehält, wenn wir jetzt nichts unternehmen.

Die sechs Tage Entwicklungszeit ergeben einen Gesamtaufwand von **51 Stunden**. Gegenüber den hochgerechneten **102 Stunden** jährlichen Aufwand der verloren geht, hätten wir bereits im 1. Jahr eine Einsparung von **51 Stunden**.

« Durch die potenzielle Einsparung, die bereits im ersten Jahr vorhanden sein kann und dem Risiko, dass bei keiner Massnahme sich der Aufwand erhöht. Finde ich, dass sich eine Umsetzung von diesem Technischen Konzept Lohnt. »

3.2. Persönliche Reflexion und Lessons learnt

Diese Diplomarbeit hat die Grenzen meiner Ausdauer gefordert. Neben dem erarbeiten des Technischen Konzepts bzw. der Diplomarbeit, war ich auch im täglichen Betrieb tätig. Eine weitere ganze Woche musste ich investieren um an einem Hotfix für die Produktive Umgebung zu arbeiten und zusätzlich war ich zwei Wochen an COVID-19 erkrankt. Das waren sehr turbulente Monate für mich, aber ich bin froh, dass ich trotzdem mein Ziel erreichen konnte.

Wenn ich meine Arbeit reflektiere, kommen mir folgende gut umgesetzte Punkte in den Sinn:

- Die Kommunikation mit den unterschiedlichen Personen *(Architekt, OP und Hauptentwickler)* habe ich konsequent durchgezogen und die Inputs in das Konzept einfliessen lassen.
- Die beteiligten habe ich regelmässig an Standup-Meetings über den Stand informiert.
- Meine Arbeit habe ich Selbstständig durchgeführt und ich wusste, an wem ich mich wenden musste, wenn etwas zur Abklärung offen war.
- Trotz vielen unterschiedlichen Inputs, Bedürfnissen und Standards, konnte ich eine simple adaptierfähige Lösung designen.
- Ich habe in der SOLL-Planung zum Glück eine Woche vor Abgabe leer gelassen, diese Woche hatte ich für ungeplante Ereignisse definiert. Deshalb konnte ich überhaupt für den Hotfix auf der Produktivumgebung einspringen.

Bisher habe ich nur an unserem Cockpit mit Java, HTML und TS entwickelt *(Angular Framework)*. Deshalb ist ein anderer positiver Nebeneffekt dieser Arbeit, dass ich das DB Spring-Boot Framework kennenlerne durfte.

Trotz all dem gibt es einige Punkte die ich ganz bestimmt jetzt anders umsetzen würde:

- Bei der nächsten Projektplanung plane ich mehr Meetings mit den verschiedenen Betroffenen ein. Die Meetings sind sehr spontan zustande gekommen und haben zum Teil meine Projektplanung beeinflusst oder sogar ausgebremst.
- Um eine Ausbremsung zu verhindern, würde ich bei der nächsten Planung überprüfen, wie die geplanten Abwesenheiten sind der betroffenen Personen. Zum Teil waren Personen die ich brauchte nicht anwesend.
- Beim nächsten Mal würde ich mindestens einen halben Tag investieren um im Web nach ähnlichen Lösungen für dieses Bedürfnis zu suchen. Somit hätte ich meinen Horizont etwas geöffnet und mich von anderen Ansätzen inspirieren lassen.

Durch diese Arbeit habe ich sehr viel Neues über PostFinance und mich selber gelernt. Ich bin sehr froh, dass ich so eine Arbeit bei PostFinance umsetzen und mein Können unter Beweis stellen durfte. Vor allem freue ich mich auf die Realisierung des Technischen Konzepts der ich nach dieser Arbeit nachgehen werde.

3.2.1. Bewertung vom Fachbetreuer

In Abbildung 34 seht ihr die Bewertung meiner Arbeit durch den Fachbetreuer Patrick Bregy (DWH Architect)

Abbildung 34: Screenshot, Bewertung durch Fachbetreuer

Thema, Aspekt	Begründung	Note
Vorbereitung		5.5
• Pflichtenheft *Aufgabenabgrenzung, Zielformulierung, Beschreibung, Vollständigkeit*	Das Pflichtenheft deckt die Bedürfnisse der Postfinance AG zum Thema der Diplomarbeit vollständig ab. Idee dazu kam von Giuseppe Scavetta selbst aus seiner beruflichen Tätigkeit als Verbesserungsvorschlag.	
• Schwierigkeitsgrad *z.B. Anteil von Bekanntem und Unbekanntem*	Schwierigkeitsgrad ist mittel bis hoch. Grund: Das Thema bietet anhand der Tools und möglichen Umsetzungsvarianten einen sehr grossen «Spielraum». Es ist nicht trivial das Thema auf die wesentlichen Aspekte einzugrenzen und zu konsolidieren.	
Planung		5.5
• Projektstruktur- und ablaufplanung, *Detaillierungsgrad, Logik, Übersichtlichkeit, Darstellung, Controlling (Soll-Ist-Vergleich)*	Planung war stets abgesprochen und ausnahmslos eingehalten.	
Umsetzung		5.5
• Analyse, Informationssammlung *Vollständigkeit, Relevanz*	Durch die offene Kommunikation und ausgeprägten Soft Skills von Giuseppe ist er effizient an die nötigen Informationen gekommen. Austausch war immer sehr konstruktiv.	
• Lösungsfindung *Variantenfindung, Beschreibung und Bewertung*	Bei der Lösungsfindung wurden primär die Anforderungen und insbesondere die Qualitätsanforderungen vorbesprochen. Die Lösungsfindung hat Giuseppe selbständig durch regelmässigen Austausch durchgeführt.	
• Nachweise *Berechnungen, Schemas, Pläne, Skizzen, Tests etc.*	Detailliertes Konzept ist erstellt worden. Granularität reicht um in die Entwicklung zu starten	
• Ausführung *Aufbau, Realisierung, Funktionsmuster, Prototyp etc.*	Es wurden funktionale Abklärungen durchgeführt z.B.: prüfen ob die Mailing-Komponente Informationen mit der Datenbank austauschen kann und mit welcher Technologie welches Problem gelöst wird.	
• Nutzen für Betrieb *Bedeutung, Machbarkeit, Akzeptanz*	Nutzen für den Betrieb ist sehr hoch. Die bestehenden Funktionalitäten werden gebündelt und einfach zugänglich gemacht. Es braucht weniger Spezialisten als bei der jetzigen Lösung hinsichtlich Konfiguration und Nutzung.	

3.3. Schlusswort und Danksagung

Die Weiterbildung zum Wirtschaftsinformatiker sind drei sehr Lehrreiche Jahre für mich gewesen. Recht herzlich bedanke ich mich bei jedem Dozenten an der TEKO Olten, der mich auf diesen Weg begleitet hat. Ebenfalls bedanke ich mich bei meinem tollen Arbeitgeber, der mir sehr viel Flexibilität und Möglichkeiten gegeben hat. Mit grosser Erleichterung beende ich meine Diplomarbeit und hoffe, dass auch meine Mitstudenten und Dozenten von mir profitieren konnten.

3.4. Eigenständigkeits-Erklärung

Ich bestätige als Verfasser dieser Arbeit, dass ich die hier vorliegende Arbeit komplett selbstständig verfasst habe und nicht durch eine andere Person erstellt wurde. Diese Arbeit wurde auch nicht weder in gleicher noch in ähnlicher Form an der TEKO oder einer anderen Schule vorgelegt. Sollten sich Hinweise ergeben, dass die hier vorliegende Arbeit nicht selbstständig verfasst, abgeschrieben oder Teile von einer anderen Arbeit übernommen wurden, wird die Arbeit mit der Note 1 bewertet. Ich bin mir bewusst, dass ich dann eine vollständig neue Arbeit erstellen muss.

Literaturverzeichnis

1. Arbeitstagerechner. [Online] https://www.arbeitstage.ch/arbeitstage_2021htm#a20.

2. Splunk Monitoring Events: bewährte Methoden und Einsichten um Events an TrueSight zu senden. [Online] INTRANET.

Abbildungsverzeichnis

Tabellenverzeichnis