Bachelor Automatisierungs-/Informationstechnik

Studiengang: BAIT15/16

Bachelorarbeit

Die automatisierungstechnisch unterstützte Integration von Labordaten in das Produktionsleitsystem einer verfahrenstechnischen Anlage

eingereicht von: Hendrik Dorn

geboren am: 01.05.1992

Matrikel-Nr.: 21936

Hochschulbetreuer: Prof. Dr.-Ing. Andreas Ortwein

Betrieblicher Betreuer: Dipl. Ing. Holger Treichel

Abgabetermin: 12.06.2019

Inhaltsverzeichnis

Selbstständigkeitserklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

_____ _____

Ort, Datum Unterschrift Hendrik Dorn

Symbolverzeichnis

A	*Fläche*
c	*Geschwindigkeit*
D	*Geschwindigkeitsgefälle*
dc	*Differenzgeschwindigkeit*
dy	*Filmdicke*
F	*Kraft*
F_S	*Scherkraft*
VI	*Viskositätsindex*
y	*Abstand*
η	*dynamische Viskosität*
ν	*kinematische Viskosität*
ρ	*Dichte*
τ	*Schubspannung*

Abkürzungsverzeichnis

ASTM	*American Society for Testing and Materials*
BDE	*Betriebsdatenerfassung*
CCS	*Cold-Cranking Simulator*
CRUD	*Create Read Update Delete*
CSS	*Cascading Style Sheets*
DBMS	*Datenbankmanagementsystem*
DHCP	*Dynamic Host Configuration Protocol*
EMSR	*Elektro-,Mess-,Steuer und Regelungstechnik*
ERP	*Enterprise Resource Planning*
GmbH	*Gemeinschaft mit beschränkter Haftung*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
I/O	*Ein- und Ausgabe*
IP	*Internet Protocol*
IT	*Informationstechnik*
JSON	*JavaScript Object Notation*
KPI	*Key Performance Indicator*
LAN	*Local Area Network*
LIMS	*Labor-Informations- und Management-System*
MES	*manufacturing execution system*
OPC	*Open Platform Communications*
PFX	*Eigenname Messgerät*
PKS	*Process Knowledge System*
PLS	*Prozessleitsystem*
RS	*Recommended Standard*
SCADA	*Supervisory Control and Data Acquisition*
SN	*Sample Number*
SPS	*speicherprogrammierbare Steuerung*
SQL	*Eigenname Datenbanksprache*
SUB-D	*D-Subminiature*
SVM	*Stabinger Viskosimeter*
UOP	*Universal Oil Production*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
VQT	*Value Quality Timestamp*

Abbildungsverzeichnis

Tabellenverzeichnis

Kurzfassung

In der vorliegenden Arbeit wurde das Thema „Die automatisierungstechnisch unterstützte Integration von Labordaten in das Produktionsleitsystem einer verfahrenstechnischen Anlage“ behandelt. Gegenstand der Betrachtung waren die verschiedenen Unternehmensebenen der PURAGLOBE Germany GmbH. Es wurde der Prozess, das Prozessleitsystem und die Datengrundlage eines Produktionsleitsystems beschrieben. Der Prozess ist das Recycling von Altöl in Basisöl. Als Prozessleitsystem wird Experion PKS von Honeywell verwendet. Die Datengrundlage des Produktionsleitsystems ist inmation.

Da in den Online-Analysesystemen die Analysewerte des Labors nicht aufliefen, sollten im Produktionsleitsystem Laborwerte mit Prozesswerten verglichen werden können. Dazu war es erforderlich, die Labormesswerte in die Datengrundlage des Produktionsleitsystems zu überführen. Dieser Arbeitsablauf sollte mit Hilfe eines Workflowmanagementsystems verbessert werden. Das dabei zu verwendende Workflowmanagementsystem musste ausgewählt werden. Joget Workflow hat alle Entscheidungskriterien erfüllt. Messwerte, die von einem Messgerät auflaufen, sollten automatisch in den Arbeitsablauf einfließen.

Ziel war es, den Zeitpunkt der Messung mit dem bestimmten Messwert zu verbinden. Dafür wurde mit Hilfe von Joget Workflow eine App erstellt, die verschiedene Prozesse enthält. Während der Prozesse werden BeanShell-*Tools* ausgeführt, die eingegebene Daten verarbeiten und an inmation weitergeben. Dazu sind SQL-Befehle und HTTP-Anfragen verwendet worden. Schließlich war es möglich, ein Messgerät an das bestehende Netzwerk anzuschließen und die Messwerte des Messgeräts in die App einfließen zu lassen. Außerdem wird beim Starten der App eine Eingabemaske angezeigt, in die Laboranten die Messdaten eingeben. Die Funktion der App konnte bereits in einer Nutzerpräsentation vorgestellt werden.

Abstract

The topic of this thesis is „the automation-aided integration of laboratory data into the manufacturing execution system of a process plant“. The different enterprise-levels of the Puraglobe Germany GmbH are described. The objects of contemplation are the process, the distributed control system and the database of the manufacturing execution system. The process is the re-refining of used oil into re-refined oil. The distributed control system is an Experion PKS system by Honeywell. The database of the manufacturing execution system is inmation.

Because some laboratory data was not transferred to the online-analysis systems, process data and laboratory data should be compared in the manufacturing execution system. Therefore, it was necessary to transfer the laboratory data into the database of the manufacturing execution system. This workflow had to be implemented into a workflow management system. The special system of choice had to be determined. Joget Workflow met all the chosen criteria. Data from a laboratory device should be directly transferred to the workflow.

The primary goal of the workflow was the combination of the sample data with the sampling time. An App was programmed that includes different processes. During the processes BeanShell Tools, that manage the data input by the laboratory assistants, are executed. This data is then transferred to inmation. SQL- and HTTP-requests are used for this. Finally, it was possible to introduce a laboratory device to the network and input this data into the app through a form, which is being displayed at the start of the app. The function of the app has already been presented.

1 Einleitung

Die vorliegende Bachelorarbeit wurde in enger Zusammenarbeit mit den Firmen Stadler + Schaaf Mess- und Regelungstechnik GmbH (im Folgenden Stadler + Schaaf) sowie der PURAGLOBE Germany GmbH (im Folgenden Puraglobe) erstellt.

Stadler + Schaaf bietet in Deutschland, Europa und weltweit Automatisierungssysteme sowie EMSR-Ausrüstungen an, um die Anlagen der Kunden sicher, zuverlässig und wirtschaftlich zu betreiben. Es werden sowohl Schaltschränke gefertigt, Messgeräte spezifiziert, Roboter programmiert, Energieversorgung geplant als auch Kabel, Sensoren und Maschinen montiert. Über die gesamte Laufzeit erfolgen Wartungen an den technischen Anlagen. Stadler + Schaaf deckt unter anderem die Felder Elektrotechnik, Automation, Industrial IT sowie Mechanik und Nukleartechnik ab und bietet dafür das Konzept, das Engineering, die Montage, die Kalibrierung, die Inbetriebnahme und den Service. [1]

Eine Säule von Stadler + Schaaf ist es, chemische Anlagen zu betreiben. Ein Bestandteil sind petrochemische Raffinerien [2]. Eine solche Anlage ist die Aufbereitungsanlage von Puraglobe, ein wichtiger Kunde von Stadler + Schaaf in Zeitz, mit der unter Verwendung des UOP-HyLube-Verfahrens aus Gebrauchtöl hochqualitative Basisöle hergestellt werden [3]. Für diesen Recyclingprozess wird Experion PKS von Honeywell als Prozessleitsystem verwendet. Mit dem Ziel, den Prozess weiter zu verbessern, arbeiten Puraglobe und Stadler + Schaaf daran, ein Produktionsleitsystem einzuführen. Um die Daten vom Prozessleitsystem in das Produktionsleitsystem zu überführen, wird inmation genutzt.

Essentiell für die Qualitätssicherung bei Puraglobe ist die periodische Gewinnung von Analysedaten an verschiedensten Stellen des technologischen Prozesses. Da viele der zu analysierenden physikalischen bzw. chemischen Größen momentan von Online-Analysesystemen nicht unterstützt werden, wurden im Unternehmen Arbeitsabläufe definiert, die manuelle Entnahmen von Produktproben, sowie deren Untersuchungen im hausinternen sowie externen Laboren zum Inhalt haben. Um diese Arbeitsprozesse zu systematisieren und in eine geeignete IT-Umgebung einzubetten, wurde seitens Puraglobe sowie Stadler + Schaaf das Projekt „Laborintegration inmation“ ins Leben gerufen. Dieses Projekt besteht im Wesentlichen aus den zwei Teilprojekten:

- „Entwicklung eines Workflowmanagementsystems für die automatisierungstechnisch unterstützte Probenentnahme“,
- „Die automatisierungstechnisch unterstützte Integration von Labordaten in das Produktionsleitsystem einer verfahrenstechnischen Anlage“.

Das erste Teilprojekt wird parallel von Dominik Strobel in einer Bachelorarbeit bearbeitet. Die vorliegende Arbeit beschäftigt sich mit dem zweiten Themenkomplex. Im Rahmen dessen soll eine Machbarkeitsanalyse zu diesem Thema durchgeführt und grundlegende Lösungskonzepte angeboten werden. An einem Beispiel ist die Übertragung von Daten aus netzwerkfähigen Laboranalysegeräten in das Produktionsleitsystem zu demonstrieren. Weiterhin soll der Teil des Gesamtprozesses, bei dem Labordaten in das Produktionsleitsystem überführt werden in einem Workflowmanagementsystem abgebildet werden. Die Auswahl einer geeigneten Softwareplattform sowie die Definition der Schnittstellen zwischen den einzelnen Teilprojekten sind mit Dominik Strobel entsprechend zu koordinieren.

2 Unternehmensebenen und deren Abbildung in verschiedenen Systemen

Wie sich die in Abbildung 2-1 gezeigten Unternehmensebenen bei Puraglobe widerspiegeln, wird im Folgenden erläutert.

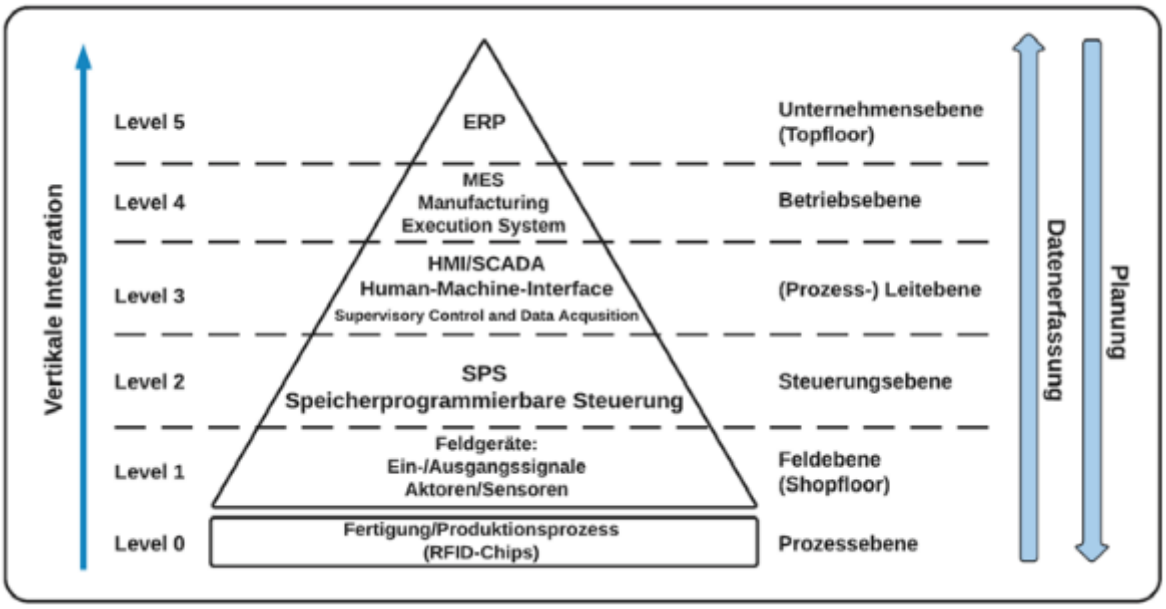


Abbildung 2-1: Unternehmensebenen in der Automatisierungspyramide [4]

Der Produktionsprozess ist bei Puraglobe das Recycling von Altöl in Basisöl mit Hilfe des UOP-HyLube-Verfahrens. Die Prozessleitebene wird bei Puraglobe im Prozessleitsystem mit Hilfe von Experion PKS realisiert. Ziel dieser Arbeit ist es, die Verbindung von Prozessleitebene und Betriebsebene herzustellen. Diese Verbindung soll mit Hilfe von inmation hergestellt werden. So wird eine Datengrundlage für die Betriebsebene bzw. das Produktionsleitsystem geschaffen.

2.1 Prozessebene

Ausgangspunkt der Automatisierungspyramide ist die Prozessebene. Der Prozess zum Recycling von Gebrauchtöl in Basisöle geschieht bei Puraglobe über das UOP-HyLube-Verfahren. Dieses Verfahren wurde von der Firma Universal Oil Production entwickelt und wurde exklusiv für Puraglobe lizensiert [5]. In Zeitz gibt es dafür zwei Anlagen [3]. In dieser Arbeit werden der Prozess und die Probenentnahme von Anlage 1 betrachtet.

In Abbildung 2-2 ist das UOP-HyLube-Verfahren schematisch dargestellt.

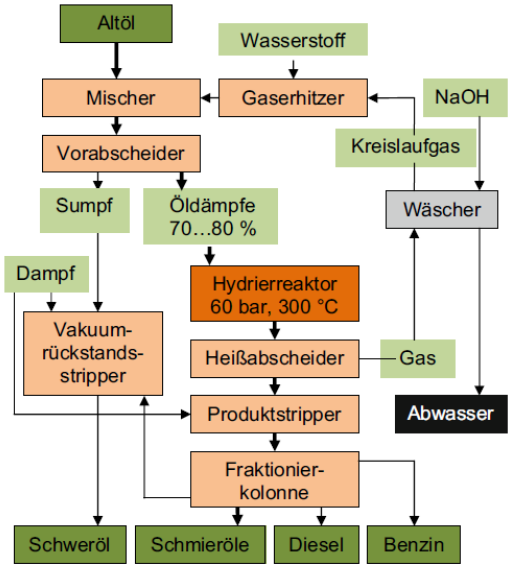


Abbildung 2-2: UOP-HyLube-Verfahren [6]

Mit dem UOP-HyLube-Verfahren erfolgt eine Direktkontakthydrierung des Altöls. Das Altöl wird dabei im Mischer mit dem erhitzten Wasserstoff aus dem Gaserhitzer vermischt. Dabei verdampfen zwischen 50 und 70 % des Öles. Im Vorabscheider wird der Sumpf vom Öldampf getrennt. Die Öldämpfe werden dann in den Hydrierreaktor geleitet. Der Sumpf wird im Vakuumrückstandsstripper mithilfe von Wasserdampf in Schweröl und weitere Öldämpfe separiert. Das Schweröl findet als Brennstoff oder in der Bitumenindustrie Verwendung. Die im Vakuumrückstandsstripper entstandenen Öldämpfe werden ebenfalls dem Hydrierprozess zugänglich gemacht.

Der Hydrierreaktor arbeitet unter Einsatz von speziellen Katalysatoren. Dabei herrscht ein Druck von 60 bis 80 bar, die Temperaturen liegen bei 300 bis 350 °C. Im Hydrierreaktor werden durch physikalische und chemische Umsetzungen Stickstoff-, Chlor- und Schwefelverbindungen der Additive und die Metallverbindungen entfernt sowie aromatische Verbindungen abgesättigt. Im Heißabscheider erfolgt die Trennung der Ölkomponenten vom Kreislaufgas. Mit Hilfe von Natronlauge wird im Wäscher Schwefelwasserstoff und Chlorwasserstoff aus dem Gas gewaschen.

Das noch vorhandene Kreislaufgas kann weiterverwendet werden. Der durch den Prozess verbrauchte Wasserstoff muss durch Frischwasserstoff ersetzt werden. Im Produktstripper werden unerwünschte Reaktionsprodukte, wie z.B. Schwefelwasserstoff und Chlorwasserstoff, aus den Ölkomponenten entfernt. Die so erhaltene Ölmischung wird in die Fraktionierkolonne geleitet. Endprodukte sind Schmierölfraktionen unterschiedlicher Viskositäten und Diesel und Benzin als Nebenprodukte. [6], [7]

Zur Qualitätssicherung werden an unterschiedlichen Stellen der Anlage Proben entnommen. Diese Arbeit konzentriert sich dabei auf die aus der Fraktionierkolonne entnommenen Proben. Die Probeentnahmestellen an den unterschiedlichen Fraktionen bzw. Kolonnenböden werden von Puraglobe SN12 bis SN16 genannt. Von den an den Messstellen SN12 bis SN16 entnommenen Proben werden verschiedene Messgrößen bestimmt.

Die bei Puraglobe verwendeten Messgeräte für die Bestimmung verschiedener Messgrößen können der Tabelle 2-1 entnommen werden. Außerdem wird betrachtet, ob ein Datenexport möglich ist, ob die direkte Anbindung an ein Netzwerk möglich ist und ob ein Computer als Bindeglied zum Netzwerk verwendet werden kann. Der Abbildung 2-3 kann der schematische Aufbau des Labornetzwerkes entnommen werden.

Die verschiedenen Messgrößen und ihre Bedeutung werden im Folgenden erläutert.

Tabelle 2-1: Messgrößen und deren Messgeräte

Messgröße	Messgerät	Daten-export möglich	Direkte Anbindung an Computer möglich	Direkte Anbindung an Netzwerk möglich
Dynamische Viskosität	CCS-2100 Automated Cold-Cranking Simulator	Ja	Ja	Nein
Kinematische Viskosität	SVM 4001 Viskosimeter	Ja	Ja	Ja
Viskositäts-index	SVM 4001 Viskosimeter	Ja	Ja	Ja
Pourpoint	OptiMPP: Mini Cloud & Pour Point	Ja	Ja	Nein
Cloudpoint	OptiMPP: Mini Cloud & Pour Point	Ja	Ja	Nein
Farbe ASTM	Lovibond PFX 195 Tintometer	Ja	Ja	Ja
Farbe Hazen	Lovibond PFX 195 Tintometer	Ja	Ja	Ja
Wasser-gehalt	Karl Fischer Titration Aqua 40	Ja	Ja	Nein

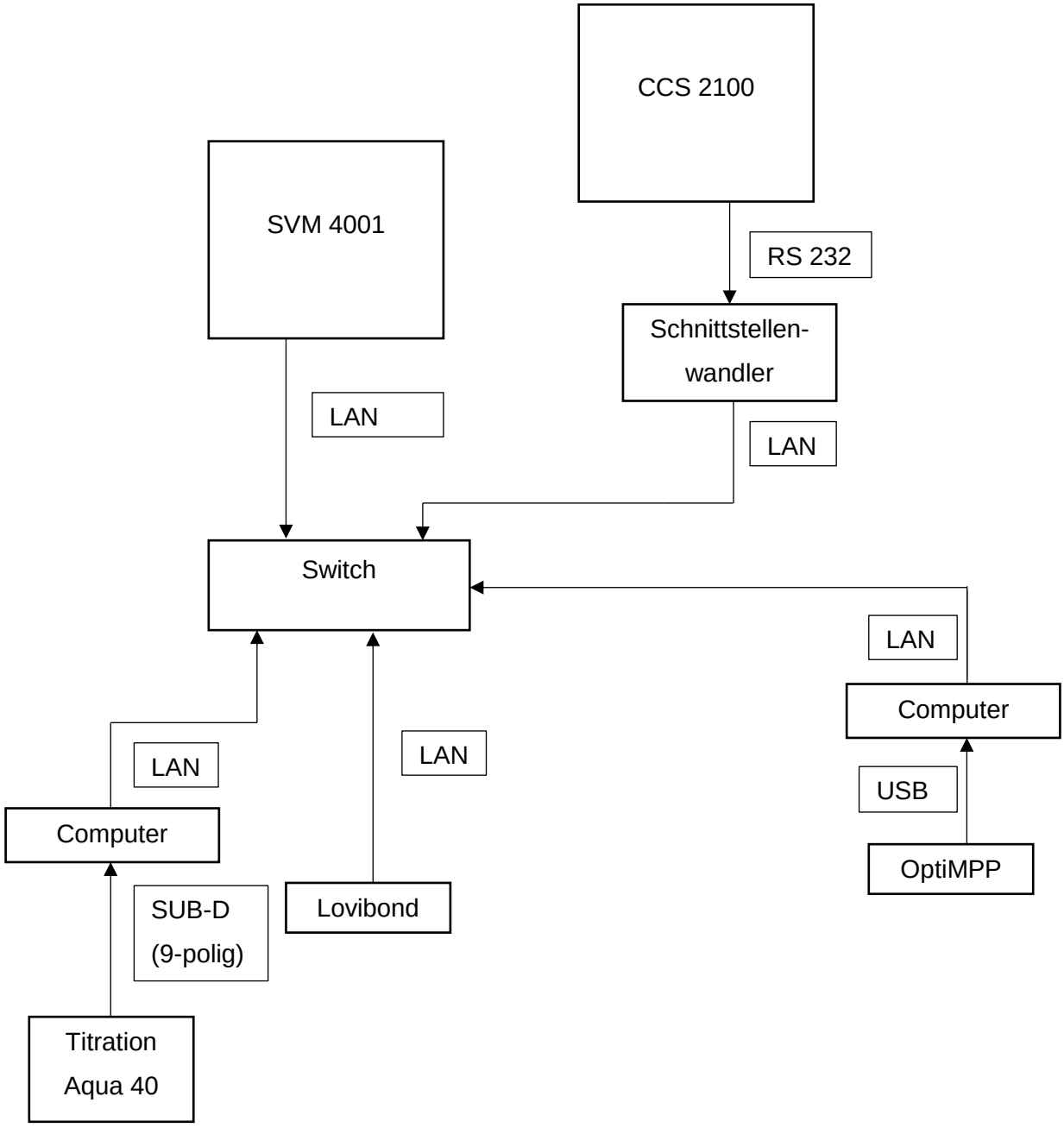


Abbildung 2-3: Labornetzwerk

2.1.1 Pourpoint/Cloudpoint

Wenn ein flüssiges Schmiermittel abgekühlt wird, geht es nicht direkt vom flüssigen in den festen Aggregatzustand über. Die Aggregatszustandsänderung erfolgt schrittweise in zwei Phasen. Die Temperatur, bei der zuerst einzelne Festkörper ausfallen, wird Cloudpoint genannt. Die Temperatur, bei der kein Fließen der Flüssigkeit mehr möglich ist, wird als Pourpoint bezeichnet. [8]

2.1.2 Farbe ASTM/Farbe Hazen

Die Farbe von Öl und Petroliumprodukten kann auf verschiedene Arten bestimmt und beschrieben werden. Eine Möglichkeit ist die Beschreibung über die American Society for Testing and Materials (im Folgenden ASTM) Farbskala. Diese reicht von 0 ASTM bis 8 ASTM. Hierbei bedeutet 0 ASTM keine Einfärbung und 8 ASTM eine dunkle bis zu fast schwarzer Einfärbung. Die Skala beruht auf einem Vergleich mit standardisierten Farbgläsern. [9]

Eine weitere Möglichkeit der Farbbeschreibung wurde von der American Public Health Association herausgegeben. Diese Farbzahl wird in Hazen angegeben. Als Kalibrierflüssigkeit wird eine Kobalt-Platinat-Lösung verwendet. [10]

2.1.3 Wassergehalt

Wassergehalt in Öl ist unerwünscht, da Wasser eine beschleunigte Ölalterung bewirkt, eine Verringerung der Schmierfähigkeit des Öles verursacht, Korrosion zur Folge hat und zu Betriebsschwierigkeiten führen kann. [11]

2.1.4 Viskosität/Viskositätsindex

Die Viskosität beschreibt den Fließwiderstand, die Zähigkeit bzw. die Fließfähigkeit. Sie gibt Aufschluss über die innere Reibung einer Flüssigkeit. Aus Abbildung 2-4 können die zur Berechnung der Viskosität wichtigen Größen abgelesen werden.

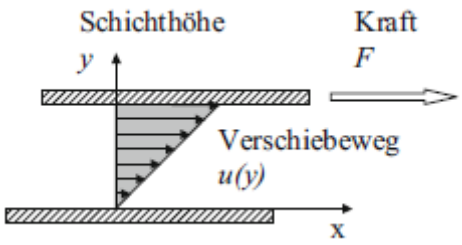


Abbildung 2-4: Skizze Viskosität [12]

Es wird angenommen, dass sich zwischen zwei zueinander parallelen Platten ein homogenes Fluid befindet. Die Platten besitzen die Fläche A und haben den Abstand y. Wirkt nun eine Kraft F auf die obere Platte mit der Geschwindigkeit c, ist die Scherkraft F_S proportional zu c und indirekt proportional zu y. Die Platten zueinander werden dabei mit der Differenzgeschwindigkeit $\mathrm{d}c$ bewegt.

$$F \sim c \sim \frac{1}{y} \qquad (1)$$

Das Geschwindigkeitsgefälle D kann dann aus dem Quotienten aus $\mathrm{d}c$ und der Filmdicke $\mathrm{d}y$ berechnet werden.

$$D = \frac{\mathrm{d}c}{\mathrm{d}y} \qquad (2)$$

Die Schubspannung τ lässt sich dann aus dem Quotienten aus Scherkraft F_S und der Fläche A berechnen.

$$\tau = \frac{F_S}{A} \qquad (3)$$

Die dynamische Viskosität η ist als Quotient aus τ und D definiert.

$$\eta = \frac{F_S \mathrm{d}y}{A \mathrm{d}c} \qquad (4)$$

Die kinematische Viskosität ν lässt sich aus dynamischer Viskosität pro Dichte der Flüssigkeit ρ berechnen.

$$\nu = \frac{\eta}{\rho} \qquad (5)$$

Die Viskosität von Ölen ändert sich abhängig von der Temperatur bei unterschiedlichen Ölsorten in unterschiedlicher Weise. Dazu wird der Viskositätsindex VI als Möglichkeit der Beschreibung der Abhängigkeit von Viskosität und Temperatur von Ölen definiert. Dafür wurden 1929 zwei Referenzöle herangezogen. Ein Referenzöl besaß die niedrigste bis dahin bekannte Abhängigkeit von Viskosität und Temperatur und ein anderes besaß die höchste bis dahin bekannte Abhängigkeit von Viskosität und Temperatur. So wurde der $VI = 0$ und der $VI = 100$ definiert. Der Wert für VI konnte nun mithilfe der kinematischen Viskosität bei 40 °C und 100 °C berechnet werden. Die Viskosität wird als die wichtigste Stoffeigenschaft von Schmiermitteln betrachtet. [12], [13], [14]

Das Messgerät SVM 4001 Viskosimeter (im Folgenden SVM 4001) misst die Viskosität. Es ist möglich, dieses Messgerät direkt an das Netzwerk anzuschließen. Deshalb wird das SVM 4001 ausgewählt, um Messdaten automatisch in die Datengrundlage des Produktionsleitsystems zu übertragen.

2.1.4.1 SVM 4001 Stabinger Viskosimeter

Die Messungen der kinematischen Viskosität bei 40 °C und bei 100 °C sowie die Bestimmung des Viskositätsindexes werden von Laboranten bei Puraglobe mit dem SVM 4001 durchgeführt. Das SVM 4001, wie in Abbildung 2-5 zu sehen, wurde entwickelt, um zwei verschiedene Temperaturen gleichzeitig messen zu können. Optimiert ist es für die Ermittlung des Viskositätsindexes. Das SVM 4001 enthält in einem Gehäuse zwei Dichte- und zwei Viskositätsmesszellen. So können von einer Probe gleichzeitig zwei Messungen bei unterschiedlichen Temperaturen vorgenommen werden. Der Bereich der Messtemperaturen des SVM 4001 liegt zwischen 15 °C bis 100 °C.

Abbildung 2-5: SVM 4001 Viskosimeter [15]

Eine Messung der kinematischen Viskosität bei 40 °C, der kinematischen Viskosität bei 100 °C und dem Viskositätsindex einer Probe mit dem SVM 4001 läuft wie folgt ab: Zuerst muss die gewünschte Messmethode aus den vorhandenen gewählt werden. Nach der Eingabe des Probennamens wird aus der Probenflasche eine Spritze mit Probenflüssigkeit befüllt. Diese Spritze wird am Messgerät angebracht. Wenn die Messung durch Betätigen der Startschaltfläche gestartet wurde, kann der Messfortschritt vom Bildschirm abgelesen werden. Ist die Messung abgeschlossen, ertönt ein akustisches Signal. Die Messergebnisse werden nun abgespeichert. [15]

Um diese Messergebnisse zu verarbeiten, bietet die Anton Paar GmbH die Software LIMS Bridge an.

2.1.4.2 LIMS Bridge

Die vom SVM 4001 auflaufenden Daten sollen an inmation weitergegeben werden. Dafür müssen die Messdaten des Messgeräts verarbeitet werden. Mit der Software LIMS Bridge können externe Messbefehle an Messgeräte gesendet und Ergebnisdateien vom Messgerät zum angeschlossenen System übermittelt werden. Nachdem das Messgerät die Messwerte der Proben bestimmt hat, speichert LIMS Bridge die Ergebnisse in einem festgelegten Ordner. Es wird für jede Messung eine Ergebnisdatei angelegt. Die Ergebnisse werden auch im Datenspeicher des Messgeräts zwischengespeichert. Um die Messdateien nutzen zu können, ist es erforderlich, einige Vorkehrungen zu treffen.

Zuerst wird das Messgerät in der LIMS Bridge hinzugefügt. Dafür muss das Messgerät an ein Netzwerk angeschlossen sein. Wenn das Messgerät mit dem Netzwerk verbunden ist, können Messdaten mithilfe eines Computers, der sich im selben Netzwerk befindet, abgerufen werden. Die Messgeräte der M Reihe können automatisch eine IP Adresse über DHCP zugewiesen bekommen. Es muss darauf geachtet werden, dass die Firewall des Netzwerks die Kommunikation von LIMS Bridge nicht blockiert. Die Kommunikation findet über den Port 8080 statt. Dann kann mithilfe LIMS Bridge das gewünschte Messgerät hinzugefügt werden. Hierfür müssen der Benutzername und das Passwort des Administratoraccounts des Messgeräts bekannt sein.

Um das Exportieren von Ergebnisdateien zu ermöglichen, muss der Export aktiviert werden. Dann kann der Ordner ausgewählt werden, in dem die Dateien abgespeichert werden sollen. Nun kann der Anfang des Dateinamens festgelegt werden, um später unterschiedliche Messergebnisse verschiedener Messgeräte zu unterscheiden. Es können folgende Dateiendungen ausgewählt werden: *.lims, *.csv, *.pdf, *.xls, *.txt. Außerdem kann festgelegt werden, ob die Daten auf dem Messgerät zwischengespeichert oder nur auf dem angeschlossenen System gespeichert werden. Um einen einzigartigen Dateinamen zu garantieren, muss eine der folgenden Optionen ausgewählt werden:

- entweder werden das Datum und die Uhrzeit an den Dateinamenanfang angehängt
- oder es werden Zahlen angehängt, die immer weiter erhöht werden
- oder es wird der Probenname angehängt und sobald ein Probenname doppelt verwendet wird, wird eine Zahl angehängt, um eine Verwechslung auszuschließen.

Des Weiteren kann ausgewählt werden, ob mehrere Messungen einzeln abgespeichert werden sollen oder eine zusammenfassende Datei ausgegeben werden soll. Schließlich kann ausgewählt werden, ob nur aktuelle Messungen abgespeichert werden oder auch solche, die im Archiv des Messgeräts abgespeichert waren, bevor LIMS Bridge für das Messgerät eingerichtet wurde.

Sobald der automatische LIMS Export aktiviert ist, erhält LIMS Bridge das Messergebnis und erzeugt für jede Messung eine Datei. Diese Datei wird im vorher eingestellten Verzeichnis gespeichert. Diese müssen dann weiterverarbeitet werden. In Abbildung 2-6 ist der Aufbau einer solchen Datei exemplarisch dargestellt.

Line 1	#V100
Line 2	#DATAHEADER:Unique Number;Date Time;Sample Name;Datafield1;Datafield2;Datafield3; Magazine Position;Method Name;User Name;Cell Temperature;Density;…<CR><LF>
Line 3	#DATAUNITS:;;;;;;;;;°C;g/cm^-3;…<CR><LF>
Line 4	17575;20080821;Aqua Bidest;Batch 234;info 1;;1;Density;Administrator;20.0;0.925;… <CR><LF>

Abbildung 2-6: Aufbau Datei [16]

In der ersten Zeile steht die LIMS Bridge Version. Die zweite Zeile kann als Kopfzeile verstanden werden, mit der den Werten in der vierten Zeile eine Bedeutung zugeordnet wird. Diese sind: der einzigartige Zahlenwert einer Messung, der Zeitpunkt der Messung im Datum/Uhrzeitformat, der Probenname und die mithilfe einer bestimmten Messmethode bestimmten Messgrößen. In der dritten Zeile stehen die Einheiten dieser Messgrößen. Die vierte Zeile enthält dann die Daten der Messung. [16], [17]

Bis jetzt wurde vorgestellt, welcher Prozess von Puraglobe genutzt wird, um Altöl zu recyceln, wie Proben aus dem Prozess entnommen werden, wie die Messwerte einer Probe bestimmt werden und wie diese Messwerte digitalisiert werden könnten. Im Folgenden wird beschrieben, wie der Zusammenhang von Prozessdaten im Prozessleitsystem und Labordaten im Produktionsleitsystem hergestellt werden soll.

2.2 Prozessleitebene

Auf der Prozessleitebene sind die Komponenten des Gesamtsystems zur Anzeige und Bedienung angeordnet. Die zentrale Einheit des Gesamtsystems ist das Prozessleitsystem [18]. Die Abbildung 2-7, aus firmeninterner Quelle (2019), zeigt die schematische Darstellung des Prozessleitsystems, wie sie von Puraglobe für den Recyclingprozess genutzt wird.

Darin sind die Anlagen 1 und 2 dargestellt, auf eine in Planung befindliche Erweiterung wird hingewiesen. Die Prozessdaten der Feldgeräte laufen in Servern auf. Diese sollen im Produktionsleitsystem mit Labordaten verglichen werden können.

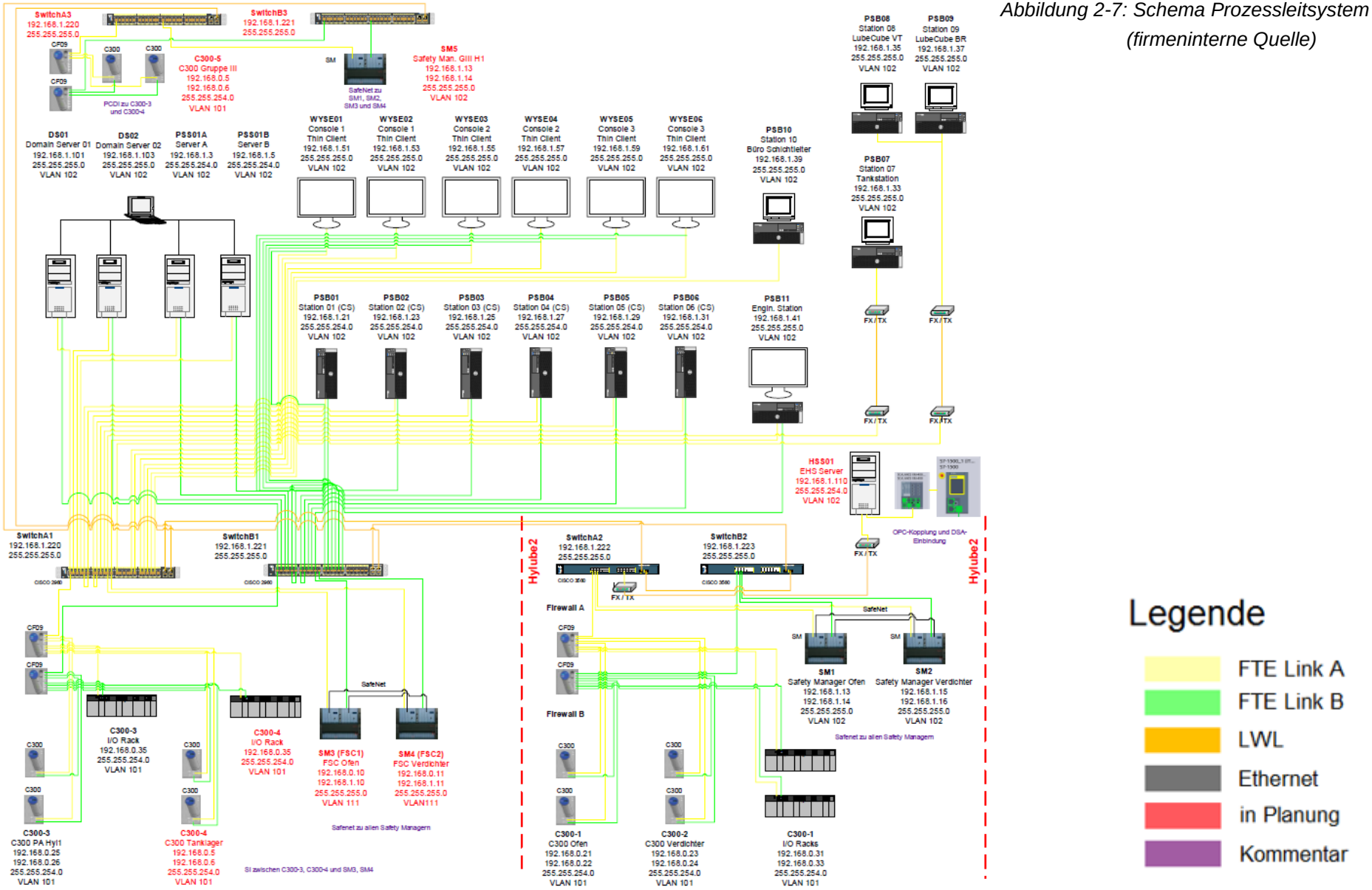


Abbildung 2-7: Schema Prozessleitsystem (firmeninterne Quelle)

2.3 Betriebsebene

Auf der Betriebsebene wird das Produktionsleitsystem angesiedelt [19]. Produktionsleitsystem ist das deutsche Synonym für den englischen Begriff „manufacturing execution system“ (im Folgenden MES). Es handelt sich um ein Produktionsmanagementsystem, das Echtzeitdaten dazu verwendet, Produktionsprozesse von Anlagen zu überwachen, zu dokumentieren und zu steuern. [20]

Die Aufgaben des Produktionsleitsystems sind in verschiedene Gebiete aufgeteilt. Diese Gebiete sind: Produktion, Instandhaltung, Qualität und Inventar. Für jedes Gebiet existieren Aufgaben:

- Definition,
- Vorhersage,
- Managementkapazitäten,
- Leistungsbewertung.

Diese Aufgaben entsprechen Aktionen, die vor, während und nach Produktionsschritten durchgeführt werden und zu denen zusätzlich Referenzdaten aufgenommen werden müssen. [21]

Bei Puraglobe wird inmation dazu benutzt, eine Datengrundlage für das Produktionsleitsystem zu schaffen. Nach erfolgter Einführung der Datenplattform soll das Produktionsleitsystem dafür genutzt werden, um Prozessdaten mit Labordaten zu korrelieren, um aus den gewonnenen Erkenntnissen neue Prozessparameter abzuleiten, die zu einer Erhöhung der Effizienz der Anlage bzw. der Qualität der Endprodukte führen. Weiterhin sollen dort Informationen zur Zuverlässigkeit der Anlagenteile und zur Produktivität der Mitarbeiter hinterlegt werden.

2.4 Inmation

Inmation ist eine Softwarelösung, mit der komponenten- und systemübergreifend Produktions- und Prozessdaten in Echtzeit aufgenommen und verschiedenen Nutzergruppen zur Verfügung gestellt werden können. Mit inmation lassen sich nahezu alle Systeme, die eine digitale Schnittstelle besitzen in ein zentrales Echtzeit-Backbone integrieren. Diese Systeme sind u. a.: ERP, BDE, MES, PLS, SCADA, und SPS. So ergibt sich eine einheitliche Datenwelt, mit der Informationen skaliert und web-basiert nutzbar gemacht werden. [22]

So können die in Abbildung 2-1 gezeigten Unternehmensebenen direkt miteinander verbunden werden.

Die Grundfunktionen von inmation sind:

- die Datensammlung von verschiedensten Datenquellen, u.a. OPC,
- das Zusammenfassen von Daten in einer zentralen no-SQL Datenbank,
- das Archivieren und Zusammenfassen von Meldungen, Alarmen, Trends und mehr,
- die Ermöglichung des Zugriffs auf diese Daten.

Die aktuelle Verbindung von Prozessleitsystem, inmation und Office PCs wird in Abbildung 2-8 dargestellt. DataStudio ist die Client Anwendung von inmation und erlaubt sicheren und schnellen Zugriff auf ein Netzwerk von Datenquellen. DataStudio liefert die komplette Übersicht über I/O Objekte, KPIs, angeschlossene OPC Server und Sicherheitseinstellungen in einer Arbeitsoberfläche. Die Nutzerschnittstelle kann Echtzeit- und archivierte Daten sowie Alarme und Events mit einer Vielzahl von Einstellungsoptionen anzeigen. [23]

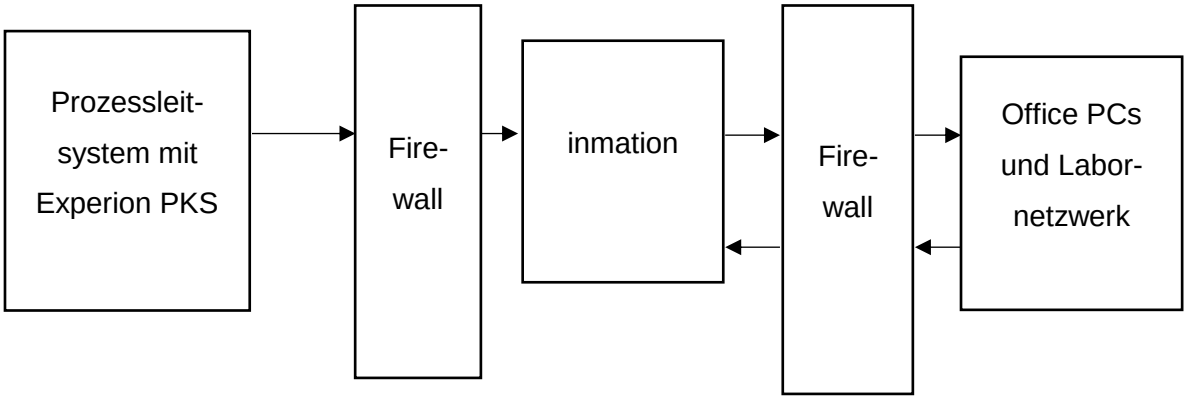


Abbildung 2-8: Verbindung PLS, inmation und PCs

Inmation speichert alle Daten im VQT-Format ab. V steht für *Value* also Wert. Q für *Quality* also Qualität und T für *Timestamp* also Zeitstempel. Diese Abspeicherung findet in der Raw History Database statt. [24]

Das I/O Modell zeigt alle zu ihr gehörenden Objekte an und repräsentiert so die physikalische Infrastruktur von inmation [25]. Folgende Ausgangssituation liegt derzeit bei Puraglobe vor: Es gibt einen Computer, auf dem der Server von inmation läuft. Die Probenauftraggeber, Laboranten und Datenanalysatoren haben Clientrechner, mit denen auf den Server zugegriffen werden kann. In inmation gibt es, wie in Abbildung 2-9 zu sehen, zu jeder Messstelle und Messgröße ein Objekt, in das die Messwerte per Hand eingetragen werden.

Es gibt verschiedene Schnittstellen, mit denen auf inmation zugegriffen werden kann. Die Einrichtung einer Datenquelle als Dropzone stellt eine solche Schnittstelle dar. Wenn in inmation eine Datenquelle als Dropzone eingerichtet wurde, stellt es eine Schnittstelle zu Daten in Dateiform dar. Eine Dropzone muss dem Typ und dem Format der Dateien angepasst werden. [26]

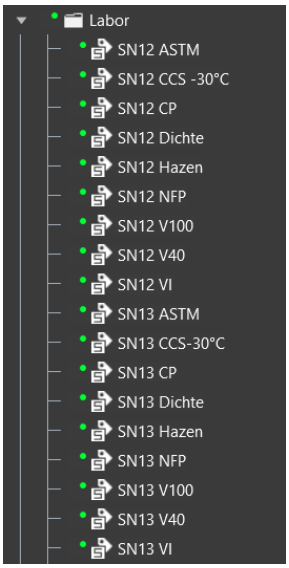


Abbildung 2-9: Objekte der Laborwerte

Eine andere Schnittstelle zu inmation ist die inmation Web API. Diese kann von einer externen Software als Schnittstelle genutzt werden, um auf inmation mit HTTP-Anfragen zuzugreifen. Mithilfe der Web API können folgende Aktionen durchgeführt werden:

- von einem Objekt lesen,
- auf ein Objekt schreiben,
- das Archiv eines Objekts auslesen,
- eine Lua-Funktion ausführen,
- ein JSON Dokument in einen Base64 String umschreiben. [27]

In dieser Arbeit sind „von einem Objekt lesen“ und „auf ein Objekt schreiben“ zur Bearbeitung der Aufgabenstellung erforderlich. Detaillierte Informationen dazu können der Swagger Dokumentation, siehe Abbildung 2-10, entnommen werden.

Zur Integration der Messwerte in inmation wurde die WebAPI als Schnittstelle ausgewählt, da so zusätzlich zur Eingabe von Daten auch Daten aus inmation gelesen werden können. Außerdem benötigt der Nutzer im Falle der WebAPI keinen direkten Zugriff auf Netzwerkordner sondern nur einen direkten Zugriff auf das Netzwerk.

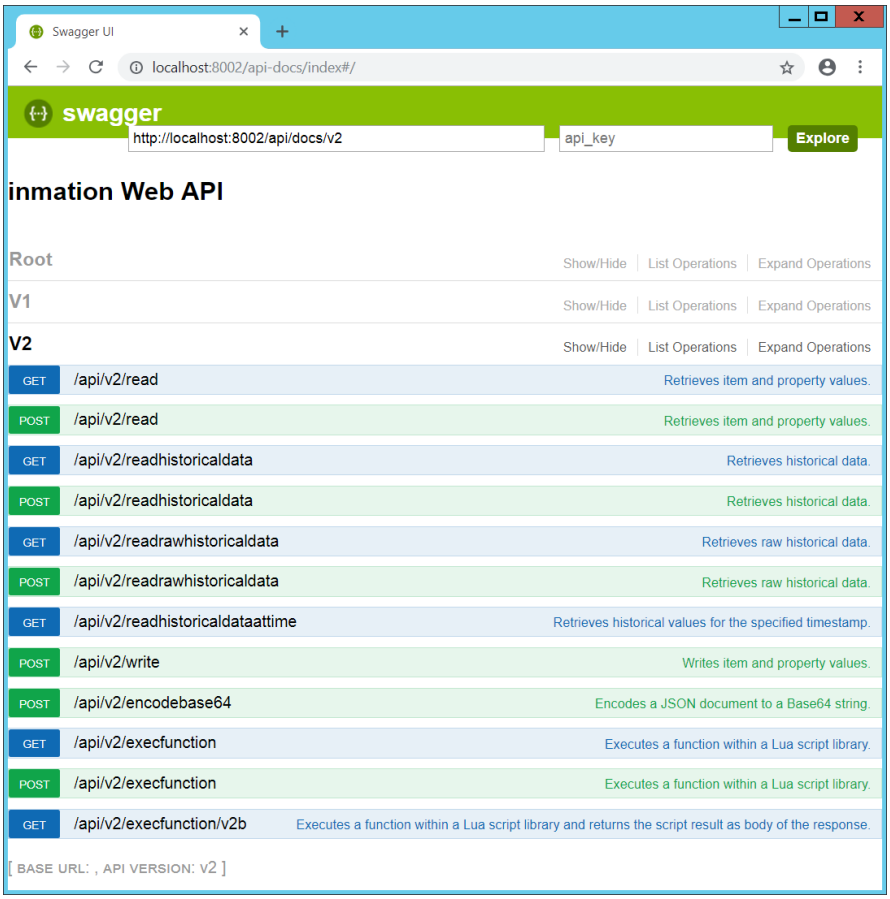


Abbildung 2-10: Swagger Dokumentation

2.4.1 HTTP

Um von externen Schnittstellen auf inmation zuzugreifen, können HTTP-Anfragen verwendet werden. HTTP ist ein Client-Server-Protokoll, das definiert, wie Nachrichten formatiert und übertragen werden. Es wird beschrieben, wie Web Server und Browser auf verschiedene Befehle reagieren. Jede HTTP–Nachricht ist entweder eine Anfrage (request) oder eine Antwort (response), die immer miteinander verbunden sind. Der Client bekommt dabei Informationen vom Server. Jeder Anfrage-/Antwortblock wird einzeln bearbeitet. Jede Art von Daten kann abgearbeitet werden, solange der Client und der Server damit arbeiten können. [28]

Es gibt unterschiedliche Arten von Anfragen, die Methoden genannt werden. Für diese Arbeit sind die *GET*- und *POST*-Anfragen von Bedeutung. Anfragen werden durch request-header spezialisiert. Hierbei werden Informationen zum Kontext der Anfrage und zum Format der Antwort geliefert. Die *GET*-Anfrage kann für die Abfrage von Daten genutzt werden. Die *POST*-Anfrage fordert an, dass die in der Anfrage gesendeten Daten vom Server verarbeitet werden. Hierbei wird immer mit einem Code geantwortet. Dieser Antwortcode beschreibt, inwieweit die Daten verarbeitet werden konnten. [29]

Inmation stellt in der Swagger Dokumentation verschiedene Befehle zur Verfügung. Für diese Arbeit sind dabei die *GET*-Anfrage mit „*Read*“ und die POST- Anfrage mit „*Write*“ von Interesse. Die Auszüge aus der Swagger Dokumentation zu diesen Anfragen können aus dem Anhang B inmation Swagger entnommen werden.

Die request header sind dabei gleich und lauten:

```
Accept: application/json, username: so, password: inmation
```

Die „*Read“*-Anfrage besteht dabei aus einer URL, die den inmation Objektpfad des jeweiligen Objekts enthält, von dem gelesen werden soll.

So lautet die URL für das Objekt SN12 VI das den Ordnern System, Training und Labor untergeordnet ist - z.B.:

```
http://localhost:8002/api/v2/read?identifier=/System/Training/Labor/SN12 VI
```

und die Antwort lautet:

```
{"data":[{"i":281475413901312,"p":"/System/Training/Labor/2SN12 VI","q":0,
"t":"2019-05-24T11:34:29.573Z","v": "Platzhalter"}]}
```

Wobei "t": auf den Zeitstempel hinweist und "v": der Wert ist.

Soll ein Wert in inmation geschrieben werden, kann die *POST*-Anfrage „*Write“* verwendet werden. Hierbei muss zusätzlich zur URL http://localhost:8002/api/v2/write noch ein Datenblock übertragen werden. Dieser lautet, um z.B. den Wert 10 für das Objekt mit dem Objektpfad „/System/Training/Labor/SN12 VI“ um 11:28 Uhr am 25.04.2019 zu schreiben:

```
{"items": [{ "p": "/System/Training/Labor/SN12 VI", "v": 10,"t": "2019-05-24T11:28:00.000Z"},]}
```

Es gilt, diese Anfragen programmtechnisch zu steuern und die Antworten zu verarbeiten. Dazu muss eine geeignete Software ausgewählt werden.

3 Implementierung

Die Integration der Labordaten kann als Arbeitsablauf verstanden werden, bei dem unterschiedliche Mitarbeiter unterschiedliche Arbeitsschritte zu verschiedenen Zeiten durchführen sollen. Dafür wird ein Workflowmanagementsystem verwendet.

3.1 Workflowmanagementsysteme

Workflowmanagementsysteme ermöglichen es, Arbeitsabläufe zu spezifizieren, auszuführen, davon Bericht zu erstatten und dynamisch Arbeitsabläufe zu kontrollieren. Die Arbeitsablaufmodelle, die die meisten Workflowmanagementsysteme unterstützen sind aktivitätsgestützt und bestehen aus folgenden Elementen:

- Arbeitsabläufe,
- Aufgaben,
- manipulierbare Objekte,
- Rollen,
- Einheiten.

Arbeitsabläufe beinhalten verschiedene Aufgaben. Aufgaben beinhalten verschiedene Operationen, die Tätigkeiten unterschiedlicher Nutzer beschreiben. Manipulierbare Objekte können z.B. Dokumente, erhobene Daten, Bilder, Telefone oder Drucker sein. Rollen sind Platzhalter für eine menschliche Fertigkeit oder die eines Informationssystems, die notwendig sind, um eine spezifische Aufgabe durchzuführen. Eine Einheit ist ein Mensch oder ein Informationssystem, die jeweils verschiedene Rollen ausfüllen. Außerdem können verschiedene Aufgaben ineinander verschachtelt werden. Ein Workflowmanagementsystem gibt dem Entwickler die Möglichkeit, eine grafische Nutzeroberfläche zu entwerfen und wenn nötig eine höhere Programmiersprache zu verwenden. [30]

In Zusammenarbeit mit Dominik Strobel musste eine geeignete Software für ein Workflowmanagementsystem ausgewählt werden. Benötigt wird ein Tool, das von verschiedenen Nutzern an verschiedenen Orten regelmäßig ausgeführt werden kann. Dabei muss es möglich sein, dass bei der Erstellung eine Programmiersprache verwendet werden kann, die HTTP unterstützt, zum Beispiel Java. Ein Zwischenspeicher von Daten, eine Datenbank, ist dienlich, um die zusätzlichen Daten, die das Messgerät SVM 4001 liefert, zu verarbeiten. Diese und die von Dominik Strobel erarbeiteten Anforderungen erfüllt die Software „Joget Workflow“.

3.2 Joget Workflow

Joget ist eine Softwarefirma, die die Open Source Software Joget Workflow entwickelt und managt. Joget Workflow vereinfacht die Entwicklung von webbasierten Unternehmensanwendungen und das Automatisieren von Prozessen. [31]

Das *App Center*, in Abbildung 3-1, ist der Ausgangspunkt beim Arbeiten mit Joget Workflow.

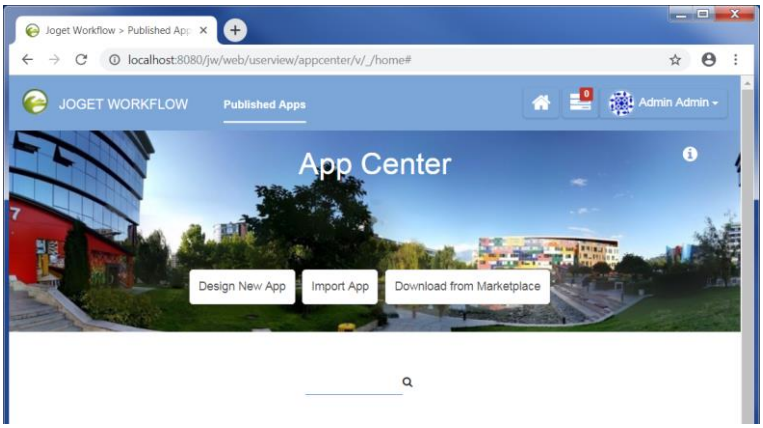


Abbildung 3-1: Joget Workflow App Center

Mit Hilfe eines Browsers kann auf das *App Center* zugegriffen werden [32]. Dies wird durch den Joget Server ermöglicht. Beim Erstellen einer App können, wie in Abbildung 3-2 zu sehen, *Forms*, *Datalists* und *Userviews* erstellt werden.

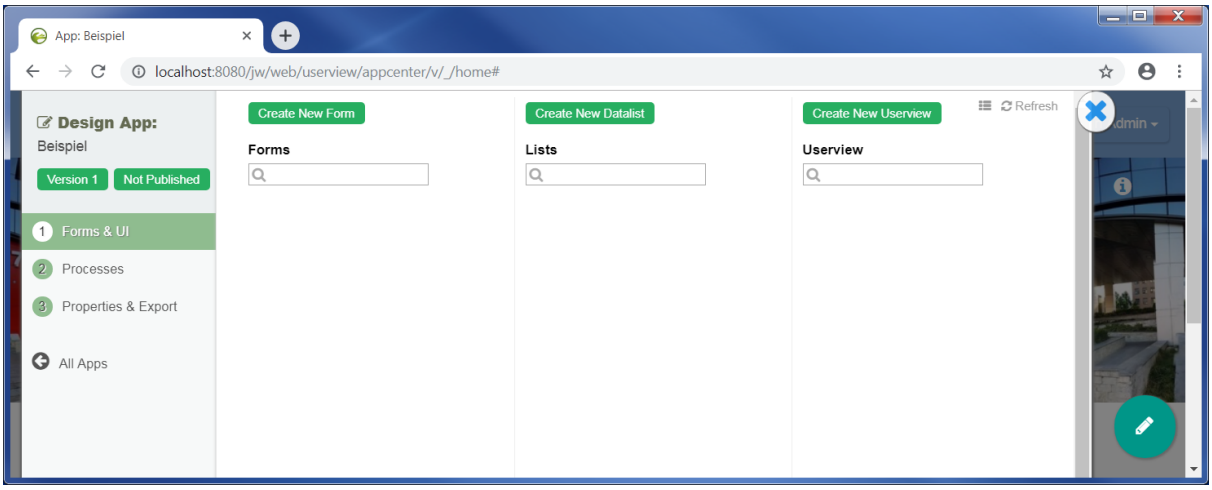


Abbildung 3-2: Erstellen einer App

Dabei muss mit dem *Userview* begonnen werden. Es ist die Nutzerschnittstelle einer Joget Workflow App. Mit einem *Userview* navigiert der Nutzer durch die App. Eine App kann ein oder mehrere *Userviews* besitzen. Ein *Userview* besteht aus einem Menü, das *Forms*, *Datalists*, und Unterfunktionen davon enthalten kann. Ein *Userview* wird mit dem *Userview Builder*, siehe Abbildung 3-3, erstellt. [33]

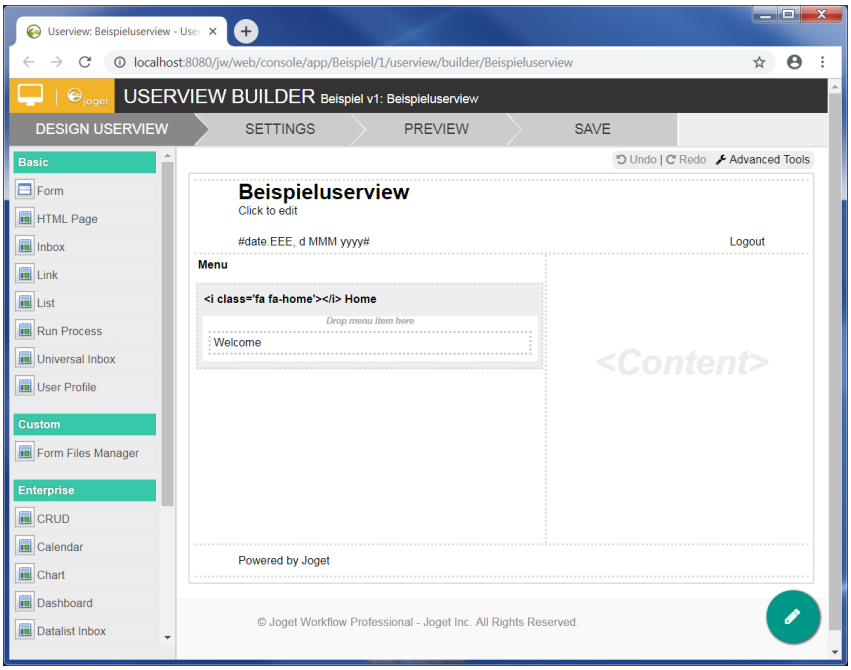


Abbildung 3-3: Userview Builder

In dieser Arbeit werden die Elemente *Form*, *Run Process* und *CRUD (Create, Read, Update, Delete)* des *Userview Builders* benutzt. Die gewünschten Elemente können aus der Liste am linken Rand des *Userview Builders* an die gewünschte Stelle des Menüs per Drag und Drop bewegt werden. Das *Form*-Element erlaubt es, Daten einer *Form* direkt einzugeben und anzusehen [34]. Das *Run Process*-Element erlaubt es, eine Prozessinstanz von einem *Userview* aus zu starten [35]. Das *CRUD*-Element nimmt es dem Nutzer ab, alle Teile eines *CRUDs* selbst zu erstellen. Ein *CRUD*-Element befähigt den Nutzer dazu, verschiedenste Einstellungen an *Form*-Elementen vorzunehmen. *Forms* werden dazu benutzt, Daten ein- und auszugeben. Es gibt unterschiedliche *Form*-Elemente, die von einem Textfeld bis hin zu Tabellen reichen können. [33]

Forms werden mithilfe des *Form Builders*, Abbildung 3-4, erstellt. Hierbei sind das *Text Field*- und *Custom HTML*-Element von Interesse. Das *Text Field*-Element ist ein Standard HTML-Eingabeelement, um die Nutzereingabe in eine *Form* zu ermöglichen [36]. Das *Custom HTML*-Element kann dazu genutzt werden, fortgeschrittene Designelemente in einer *Form* zu erstellen. Dafür kann jede Art von HTML, JavaScript und CSS Syntax verwendet werden. [37]

Datalists sind Berichte in Tabellenform, mit denen man eingegebene Daten bearbeiten, sortieren, filtern und exportieren kann [33]. In Abbildung 3-5 ist ein *Datalist Builder* dargestellt, mit dem *Datalists* in Joget erstellt werden. Mit dem *Datalist Builder* lassen sich Listen anfertigen, die Daten aller *Forms* der einzelnen Workflow-Prozessinstanzen enthalten. Jedes Mal, wenn

ein Nutzer eine *Form* ausfüllt, wird in der zur *Form* zugeordneten Tabelle in der Joget-Datenbank die dazugehörige Zeile in der Tabelle hinzugefügt bzw. ein Update in der Zeile durchgeführt. Diese Zeile kann dann in einer *Datalist* angezeigt werden. [38]

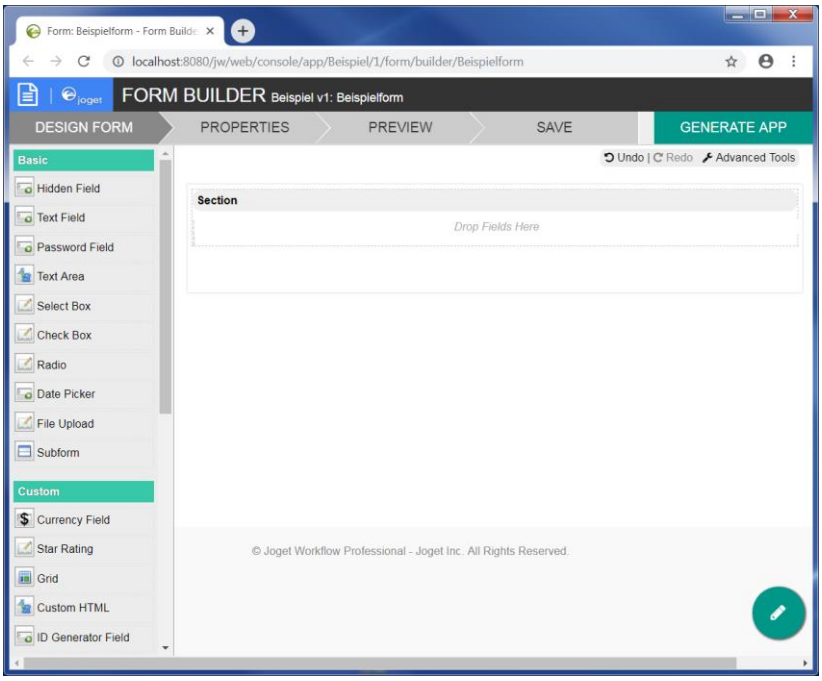


Abbildung 3-4: Form Builder

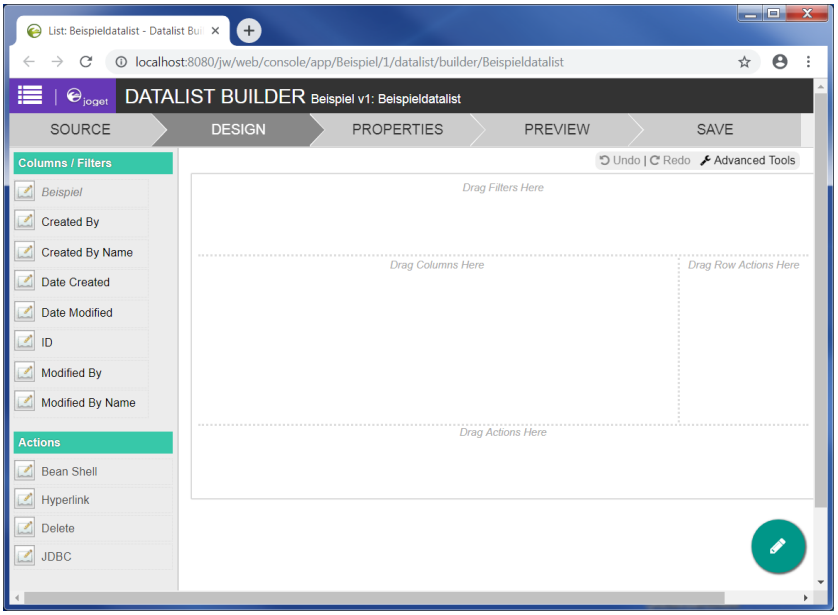


Abbildung 3-5: Datalist Builder

In einer App können Prozesse erstellt werden, mit denen *Activities* der zugehörigen Person zugeordnet werden. Mit Prozessen können externe Systeme mit Hilfe von *Tools* eingebettet werden. Somit ist es möglich, zum Beispiel E-mails zu versenden oder externe Datenbanken zu aktualisieren. [33]

Prozesse werden mithilfe des *Process Builders*, siehe Abbildung 3-6, erstellt.

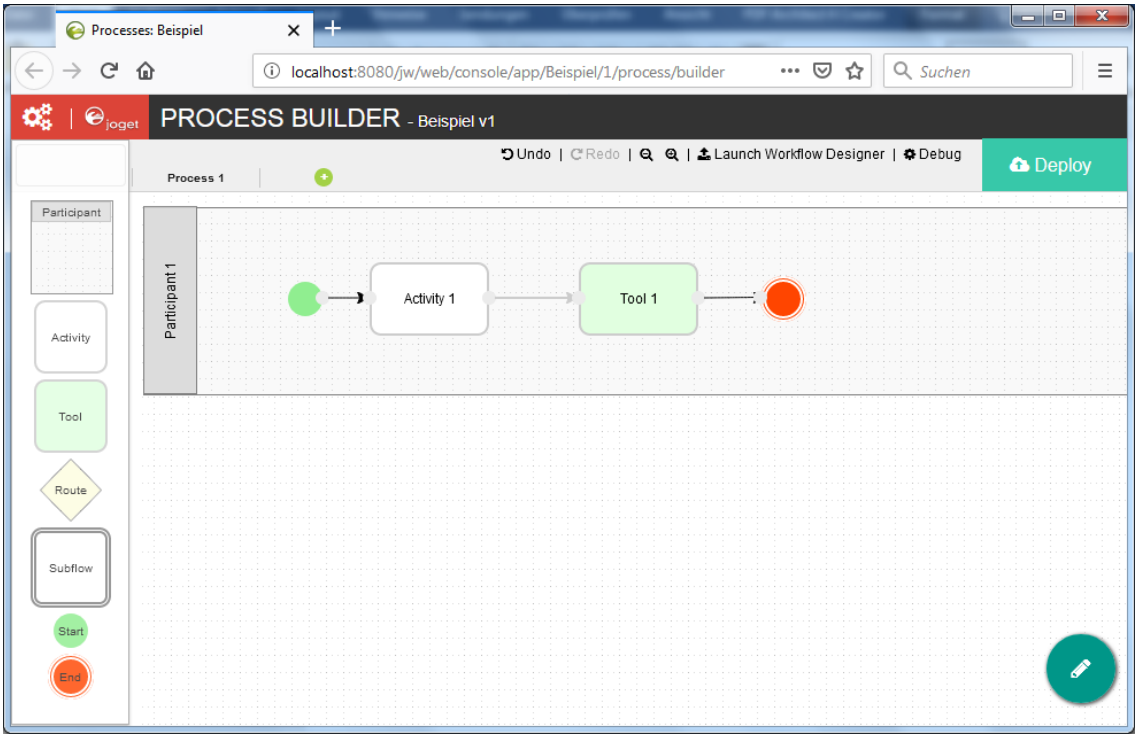


Abbildung 3-6: Process Builder

Mithilfe des *Process Builders* werden per Drag und Drop die einzelnen Elemente des Prozesses zusammengestellt. Diese Elemente sind:

- *Participant,*
- *Activity,*
- *Tool,*
- *Route,*
- *Subflow,*
- *Start*,
- *End.*

Das *Participant*-Element wird genutzt, um eine neue *Swimlane* für einen Teilnehmer festzulegen. Eine *Swimlane* ist, analog einer Schwimmbahn, ein Verantwortungsbereich für einen Akteur in einem Prozessabschnitt [39]. Eine Teilnehmer-*Swimlane* steht für eine Rolle (z.B. Auftragnehmer, Auftraggeber), eine Person (z.B. ein bestimmter Mitarbeiter einer Firma) oder eine Einheit (z.B. eine bestimmte Abteilung) in einem Prozess. Ein *Activity*-Element kann als Teil des Prozesses dazu genutzt werden, eine *Form* zu verlinken, in die der Nutzer Daten eingeben kann. Mit dem *Route*-Element werden die einzelnen Elemente eines Prozesses verbunden. Mit dem *Start*-Element beginnt ein Prozess und das *End*-Element beendet einen Prozess. [40]

Deadlines können als Timer genutzt werden, um eine Transition zu einer anderen *Activity* auszulösen, wenn eine bestimmte Zeit vergangen ist. Für jede *Activity* kann eine Deadline festgelegt werden. Durch das Ablaufen der voreingestellten Zeit wird eine *Exception* ausgelöst. Wenn eine Transition existiert, die diese *Exception* als Namen hat, wird diese Transition ausgeführt. Die Ausführung einer Deadline kann asynchron oder synchron geschehen. Bei synchroner Ausführung wird die *Activity* deaktiviert, wenn die Deadline ausgelöst wird. Bei asynchroner Ausführung können mehrere *Activities* gleichzeitig aktiv sein. Eine *Activity* kann mehrere Deadlines haben. [41]

Das *Tool*-Element wird dazu genutzt, einzelne Programmfunktionen während des Prozesses zu triggern. Hierbei können verschiedene Plugins verwendet werden. Eines dieser Plugins ist das BeanShell *Tool*. BeanShell ist ein kleiner einbettbarer Java Interpreter mit objektorientierten Elementen. BeanShell führt Standard Syntax dynamisch während der Laufzeit einer App aus. Mit Hilfe des BeanShell-Plugins wird geschriebener Java Code beim Aufrufen während der Laufzeit ausgeführt. Bei der Programmierung mit BeanShell kann auf Joget-Bibliotheken zugegriffen werden. [42]

Mit BeanShell ist es möglich, Java Programme zu erstellen. Für die Bearbeitung der Aufgabenstellung dieser Arbeit können die beschriebenen HTTP-Anfragen ausgeführt werden, die Eigenschaften dieser Anfragen sind dabei Bestandteil der gespeicherten Daten von Joget *Forms*. Die essentiellen Informationen der *Forms* können mit Hilfe von SQL-Abfragen während des Programmablaufs aus den von Joget angelegten Tabellen in der Joget-Datenbank ausgelesen werden.

3.2.1 Datenbank

Eine Datenbank ist eine Datensammlung, deren Elemente logisch zueinander in Beziehung stehen. Eine Datenbank kann über ein eigenes Datenbankverwaltungssystem (Database Management System, DBMS) gemanagt werden. [43]

Man spricht von relationalen Datenbanken. Jedes DBMS setzt sich aus einer Verwaltungs- und einer Speicherungskomponente zusammen. In der Speicherungskomponente sind alle Daten zusammengefasst. Alle Daten und deren Beschreibung werden in organisierter Form gespeichert. In der Verwaltungskomponente ist eine Abfragesprache sowie eine Manipulationssprache enthalten, womit die Auswertung und Veränderung von Daten und Informationen erfolgen kann. [44]

3.2.2 SQL

Die Programmiersprache SQL hat sich zur wichtigsten Abfrage- und Manipulationssprache entwickelt, mit der auf relationale Datenbanken zugegriffen wird [44]. In SQL werden alle Daten

in Form einer Tabelle gespeichert. Die Zeilen dieser Tabellen werden als Satz oder Tupel bezeichnet, eine Spalte wird Feld oder Attribut genannt.

Abfragen erfolgen in SQL mit *Select*-Befehlen. Ein Beispiel eines *Select*-Befehls lautet:

```
SELECT Spalten FROM Tabellen WHERE Bedingungen;
```

Hierbei dienen die Informationen zu Spalten, Tabellen und Bedingungen als Unterschiede in der Datenbank. Die Ergebnisse von *Select*-Befehlen sind Ausschnitte aus der Datenbank. Die SQL-Befehle für Mutationen heißen:

- *Delete,*
- *Update,*
- *Insert into.*

Mithilfe von Mutationen werden Ausschnitte aus Datenbanken ausgewählt und abgegrenzt. Die Inhalte dieser Ausschnitte werden gelöscht, verändert oder neu hinzugefügt. Dadurch wird der Datenbankinhalt geändert. Die Syntax des *Delete*-Befehls lautet:

```
DELETE FROM Tabellenname [[AS] Aliasname] [WHERE Bedingung]
```

Durch den *Delete*-Befehl werden alle Tupel der angegebenen Tabelle gelöscht, bei denen die Bedingung der *Where*-Klausel erfüllt sind. Liegt keine *Where*-Bedingung vor, werden alle Einträge der Tabelle gelöscht.

Die Syntax des *Update*-Befehls lautet:

```
UPDATE Tabellenname [[AS]Aliasname] SET Spalte = Spaltenausdruck [, ...] [ WHERE Bedingung]
```

Die Tabelle, die durch den *Update*-Befehl bearbeitet werden soll, ist nach „*Update*“ anzugeben. Die zu ändernden Spaltenelemente werden nach „*Set*“ gemeinsam mit den neuen Informationen benannt. Die „*Where*“ Klausel kann genutzt werden, um eine Restriktion zu erstellen. Eine vereinfachte Syntax des *Insert into*-Befehls lautet:

```
INSERT INTO Tabellenname [(Spaltenliste)] {VALUES (Auswahlliste) [, ... ] | Select-Befehl} [43]
```

Für jeden *Insert into*-Befehl wird ein neues Tupel in eine Tabelle eingefügt. Die *Values*-Klausel ermöglicht das Einfügen der Informationen anhand der Reihenfolge der Spalten der Datenbanktabellen. [45]

Die Befehle *Select*, *Insert into*, *Update* und *Delete* werden im folgenden Beispiel an der Tabelle Messdaten veranschaulicht:

Tabelle 3-1: Messdaten

Messstelle	Messgröße	Messwert
SN12	V40	10
SN12	V100	30
SN12	VI	98
SN13	V40	11
SN13	V100	29
SN13	VI	99

Dazu folgender *Select*-Befehl:

```
Select Messwert FROM Messdaten WHERE Messstelle = SN12
```

Das Ergebnis am Beispiel wäre dann:

Tabelle 3-2: Ergebnis Select

Messwert
10
30
98

Mit dem *Update*-Befehl:

```
Update Messdaten Set Messwert=100 WHERE Messgröße = VI
```

wird die Tabelle Messdaten wie folgt geändert:

Tabelle 3-3: Ergebnis Updatebefehl

Messstelle	Messgröße	Messwert
SN12	V40	10
SN12	V100	30
SN12	VI	100
SN13	V40	11
SN13	V100	29
SN13	VI	100

Mit dem *Delete*-Befehl:

```
Delete FROM Messdaten
```

wird die Tabelle Messdaten wie folgt geändert:

Tabelle 3-4: Ergebnis Delete Befehl

Messstelle	Messgröße	Messwert

Mit dem *Insert into*-Befehl:

```
Insert into Messdaten (Messstelle, Messgröße, Messwert) Values (SN12, VI, 99)
```

Wird die Tabelle Messdaten wie folgt geändert:

Tabelle 3-5: Ergebnis Insert into Befehl

Messstelle	Messgröße	Messwert
SN12	VI	99

Analog dieser Beispiele können mit Hilfe von SQL-Befehlen in BeanShell *Tools* in Joget Workflow Apps Daten aus der Joget-Datenbank ausgelesen oder in der Joget-Datenbank geändert werden.

3.3 Programmierung

Es folgt die Umsetzung der Aufgabenstellung, Labordaten in das Produktionsleitsystem einer verfahrenstechnischen Anlage zu integrieren, mit Joget Workflow, inmation, SQL-Befehlen und HTTP-Anfragen. Es wird eine Zielstellung mit Anforderungen formuliert. Die Bearbeitung erfolgt mit den theoretisch dargelegten Mitteln.

3.3.1 Anforderungen

Wie beschrieben, sollen die Messergebnisse verschiedener Laborproben an inmation weitergegeben werden, um so eine Datengrundlage für das Produktionsleitsystem zu schaffen. Um die Messergebnisse den richtigen Messstellen und Messgrößen in inmation zuzuordnen, soll den Laboranten zur Messwerteingabe eine Eingabemaske vorbereitet werden.

Die Informationen, aus welchen Messstellen und Messgrößen zu einem bestimmten Zeitpunkt Proben entnommen wurden, kann über HTTP-*GET*-„*Read*“-Anfragen an inmation ermittelt werden. Diese Informationen werden über den SQL-Befehl „*Insert into*“ in die Eingabemaske eingefügt. Die vom Messgerät SVM 4001 gesammelten Werte können dann über den SQL-Befehl „*Update*“ in die Eingabemaske eingepflegt werden. Die von Laboranten in die Eingabemaske eingegebenen Messwerte müssen dann von einem Programm verarbeitet werden. Das

Programm gibt die verarbeiteten Werte an inmation weiter. Dies ist über die HTTP-POST-„Write“-Anfrage möglich.

Das Programm, das diese Funktionen enthält, kann in Form einer App mit Joget Workflow erstellt werden. Im Verlauf des Projekts „Laborintegration inmation“ hat Dominik Strobel es möglich gemacht, dass sobald eine Probe aus der Anlage entnommen wurde, zum Entnahmezeitpunkt der Wert des zugehörigen Objekts in inmation auf „Platzhalter“ gesetzt wird. Diesem Entnahmezeitpunkt sollen jetzt die bestimmten Messwerte zugeordnet werden. In Abbildung 3-7 ist die geforderte Umsetzung der Integration von Labordaten in inmation dargestellt. Es soll möglich sein, zu jeder Messstelle, aus denen Proben entnommen wurden, sowie zu jeder Messgröße einen Messwert zu vergeben. Dies soll mit Joget Workflow möglich gemacht werden. Zur Umsetzung dieser Zielstellung gilt es, verschiedene Anforderungen zu erfüllen.

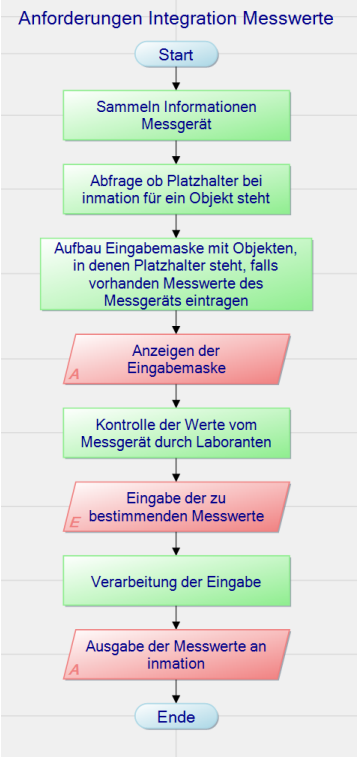


Abbildung 3-7: Programmablaufplan Messwertübertrag

Den Laboranten, die Messwerte im Workflowmanagementsystem eingeben, sollen nur die Kombinationen aus Messstelle und Messgröße angezeigt werden, bei denen bei der betroffenen Messstelle eine Probe entnommen wurde. Dies kann über eine Abfrage auf „Platzhalter“ an inmation überprüft werden. Deshalb muss eine Verbindung von Joget Workflow zu inmation bestehen.

Nach der Abfrage soll eine Eingabemaske erscheinen, in die Laboranten die gemessenen Werte eintragen können. Dafür muss von einem Programm zuerst eine leere Eingabemaske erstellt werden, um dann die relevanten Einträge zur Eingabemaske hinzuzufügen.

Wurde das Messgerät SVM 4001 mit Proben bestückt, die Messwerte bestimmt und diese an einen Computer exportiert, sollen diese vom Programm zur Eingabemaske hinzugefügt werden. Sind alle Daten gesammelt, sollen den Laboranten die mit Informationen aufgefüllte Eingabemaske angezeigt werden. Die Laboranten sollen nun kontrollieren können, ob die gemessenen Werte des Messgeräts in Ordnung sind. Sollte dies nicht der Fall sein, muss die Möglichkeit bestehen, Korrekturen vorzunehmen. In die dann noch offenen Felder der Eingabemaske würden die Laboranten die gemessenen Werte eintragen. Ist dieser Schritt abgeschlossen, soll die Eingabe mit einem Knopfdruck auf eine Schaltfläche beendet werden können. Im nächsten Schritt ist es das Ziel, dass das Programm die Eingabe verarbeitet und die Messwerte an inmation weitergibt.

3.3.2 Anbindung Messgerät

Um die Messergebnisse des Messgeräts SVM 4001 mit einer Joget Workflow App verarbeiten zu können, müssen Vorkehrungen getroffen werden. Zur Archivierung der Messdaten des Messgeräts SVM 4001 kann über die Software LIMS Bridge der Messdatenexport eingestellt werden. Um diese Einstellungen vorzunehmen, muss das Messgerät an das firmeninterne Netzwerk angeschlossen werden.

Zuerst wird über LIMS Bridge das Messgerät im Netzwerk gesucht. Sobald es gefunden wurde, wird ein Name für das Messgerät vergeben, der Benutzername und das Passwort des Administratoraccounts des Messgeräts eingegeben, um es der Software LIMS Bridge hinzuzufügen. Jetzt ist es möglich, den Datenexport von Messdaten zu aktivieren und die Zusammensetzung des Dateinamens zu definieren.

Als Dateityp wird .csv ausgewählt, da so auf die einzelnen, durch Kommata getrennten Elemente schrittweise zugegriffen werden kann. Der Ordner, in dem die Messdaten abgespeichert werden, ist auszuwählen. Hierbei ist darauf zu achten, dass es sich um einen Netzwerkordner handelt, auf den vom gesamten Netzwerk aus zugegriffen werden kann. Der Dateiname beginnt immer mit dem Namen des Messgeräts, in diesem Fall „SVM 4001“. Dazu wird die aktuelle Uhrzeit und das Datum hinzugefügt, um so überprüfen zu können, ob in einem bestimmten Zeitabschnitt eine Messung durchgeführt wurde, von der eine Messdatei vorhanden ist. Wurden diese Einstellungen vorgenommen, entsteht eine Datei, wie in Abbildung 3-8 gezeigt.

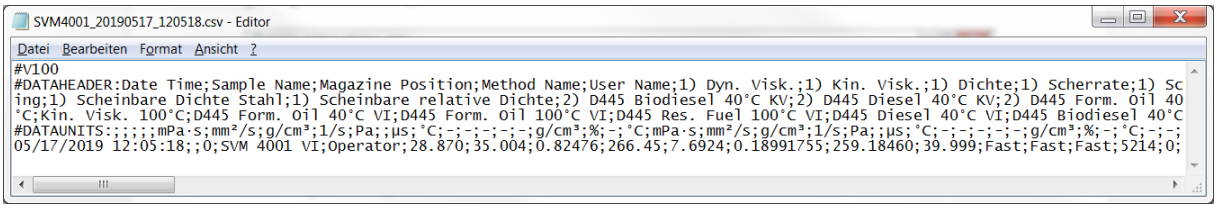


Abbildung 3-8: Ausgabedatei

3.3.3 Programmierung Joget Workflow

Für die Implementierung in Joget Workflow muss eine App erstellt werden, die die nötigen *Userviews*, *Forms*, *Datalists*, *Prozesse* und BeanShell *Tools* enthält. Diese müssen dann geplant und implementiert werden. Um die beschriebene Eingabemaske zu erstellen, müssen die nötigen Informationen gesammelt werden. Es ist erforderlich, die für die Ausgabe nötigen Daten während des Programmablaufs zu speichern und zusammenzustellen. Außerdem müssen die vom Messgerät SVM 4001 auflaufenden Daten verarbeitet werden. So entstehen zwei Programmablaufpläne. Einerseits die Erstellung der Eingabemaske und die Verarbeitung und Ausgabe der eingegebenen Daten, dieser ist in Abbildung 3-9 dargestellt und andererseits die Verarbeitung der Messdaten vom SVM 4001, dieser ist in Abbildung 3-10 dargestellt.

Um diese Programmablaufpläne umzusetzen, muss eine Joget Workflow App erstellt werden, die zwei Prozesse enthält. Ein Prozess „Eingabemaske“, der die eingegebenen Daten aufnimmt und verarbeitet und ein Prozess „Messgerät“, der die vom Messgerät auflaufenden Daten verarbeitet. Um diese Prozesse zu starten, muss mit Joget Workflow ein *Userview* erstellt werden, dass die zwei *Run Process* Elemente enthält, die die beiden Prozesse starten. Außerdem können benötigte Informationen über *Forms* eingegeben werden, um verschiedene Einstellungen vorzunehmen.

3.3.3.1 Prozess Messgerät

Bevor der Prozess programmiert werden kann, muss eine Tabelle in der Jogetdatenbank erstellt werden, in der die Messergebnisse vom Prozess gespeichert werden. Der Prozess wird, wie in Abbildung 3-11 zu sehen, als Kombination aus *Tool* und *Activity* erstellt. Mit dem *Tool* wird der Programmcode eines BeanShell *Tools* ausgeführt. Die *Activity* dient als Werkzeug zum wiederholten Aufrufen mithilfe einer Deadline. Das BeanShell *Tool* setzt den Dateinamen aus SVM 4001, der aktuellen Datum-Uhrzeit-Kombination und .csv zusammen. Mithilfe der Deadline wird dies dann immer wieder aufgerufen.

Dann kann die Datei mit den Messergebnissen schrittweise durchgearbeitet werden. Eine Deadline kann in den Optionen einer *Activity* eingerichtet werden. Hierbei muss zwischen synchron und asynchron ausgewählt werden. Da der Prozess im Hintergrund läuft und nicht immer wieder eine neue *Activity* angelegt werden soll, kann asynchron ausgewählt werden.

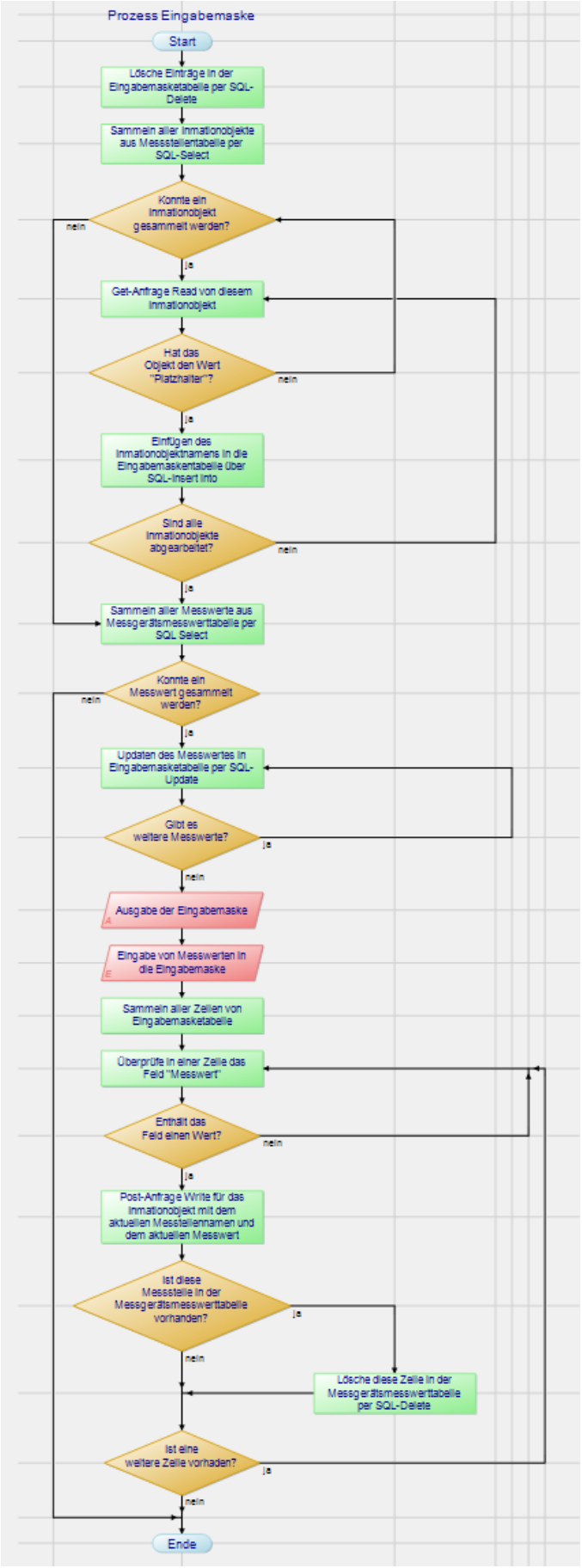


Abbildung 3-9: Programmablaufplan Prozess Eingabemaske

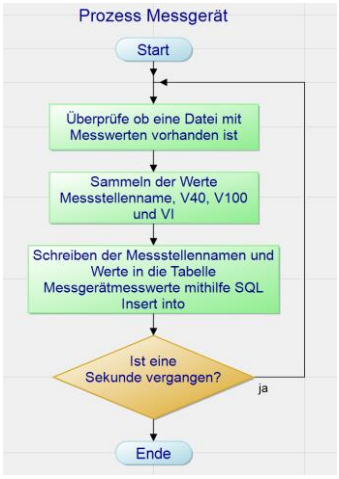


Abbildung 3-10: Programmablaufplan Prozess Messgerät

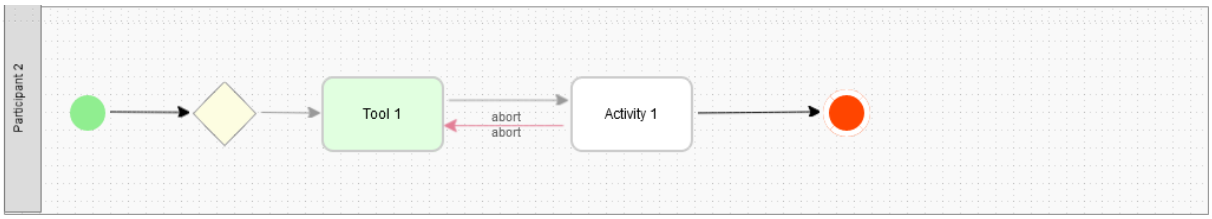


Abbildung 3-11: Prozess Messgerät

Die Deadline kann hier auf eine Sekunde gesetzt werden, da so jede Sekunde überprüft wird, ob eine neue Datei vom Messgerät abgespeichert wurde. Außerdem muss in den Systemeigenschaften von Joget Workflow ausgewählt werden, in welchen Zeitabständen das Ablaufen einer Deadline überprüft werden soll. Diese Einstellung wurde auch auf eine Sekunde festgelegt.

Ein solcher *Insert into*-Befehl lautet z.B.:

```
INSERT INTO app_fc_neu (IDtest,c_messstellenname,c_messwert) VALUES ('"+ID+"','"+
messstellenname+"V40"+"','"+v40+"')
```

IDtest ist der einzigarte Zeilenname in der Tabelle und in der Variable v40 steht der Wert, der in der Textdatei in Zeile 4 an der Stelle 7 stand.

Der komplette Quelltext kann dem „Anhang A Quelltexte “ entnommen werden.

3.3.3.2 Prozess Eingabemaske

Die Eingabemaske kann ein *CRUD* sein, das über ein *Custom*-HTML-Element in eine *Form* eingebunden wird. Die Syntax für das *Custom-HTML*-Element konnte der Joget Workflow Dokumentation entnommen werden [46]. Die *Datalist,* die mit diesem *CRUD* verbunden ist, ist zentrales Element der Programmierung. In diese müssen alle Informationen einfließen und das ist letztendlich die *Datalist*, aus der die inmation HTTP-Anfragen abgeleitet werden.

Im ersten Schritt muss also eine leere Eingabemaske erstellt werden. Die Eingabemaske kann aus mehreren Zeilen bestehen, wobei die Spalten jeweils Informationen zu Messstelle, Messgröße und Messwert enthalten.

Ein Beispiel einer solchen Tabelle kann der Tabelle 3-1 entnommen werden. Da jedoch die inmation-Objekte (siehe Abbildung 2-9) die Kombinationen aus Messstelle und Messgröße enthalten, sollte ein Feld auch die Kombination aus Messstelle und Messgröße enthalten, um die Programmierung zu erleichtern. Um ein wiederholtes Einfügen in die Tabelle zu verhindern, wird am Anfang des Prozesses die Tabelle geleert. Dies ist über einen *Delete*-Befehl mit SQL möglich. Dann kann mithilfe einer HTTP-*GET*-„*Read*"-Anfrage von inmation abgefragt werden, ob in einem Element „Platzhalter" steht. Dabei kann geprüft werden, ob eine Verbindung zum Server besteht. Um die Abfragezusammenstellung zu vereinfachen, wird eine *Form* angelegt, die alle Laborobjekte von inmation enthält. Diese *Form* erhält den Namen „messstellen". Mit Hilfe dieser *Form* kann dann durch alle Namen der Objekte in inmation iteriert werden.

Außerdem müssen für die *GET*-Anfrage Informationen zur IP-Adresse und Ordnerstruktur des inmation-Servers bekannt sein. Diese können extra in einer Form „einstellungen" abgespeichert werden. Das „Wert" Feld einer Spalte kann dabei leer bleiben, da die Laboranten dort schließlich den Messwert eingeben können. Es muss nun überprüft werden, ob vom Prozess „Messgerät" ein Messwert für die einzelnen v40-, v100- und VI Werte abgelegt wurden. Diese können dann in die Spalte Messwert übertragen werden. Jetzt wird vom Prozess die *Activity* zum Messwerteintrag den Nutzern, in diesem Fall den Laboranten angezeigt. In Abbildung 3-12 ist so eine *Activity* zu sehen.

Haben die Laboranten die Eingabe beendet, können sie den *Complete*-Knopf drücken. Es wird von Joget Workflow automatisch für jede *Activity* in einem Prozess ein *Complete*-Knopf angelegt. Wurde der *Complete*-Knopf gedrückt, müssen die eingegebenen Daten verarbeitet werden. Um die Messwerte an inmation zu übergeben, können *POST*-Anfragen verwendet werden. Die Entnahmezeit kann über *GET*-Anfragen abgefragt werden. Die Abbildung 3-13 zeigt, den in Joget Workflow angelegten Prozessablauf.

Der komplette Aufbau der App mit *Userview*, *Forms* und *Datalists* kann dem Anhang C Joget App Screenshots entnommen werden.

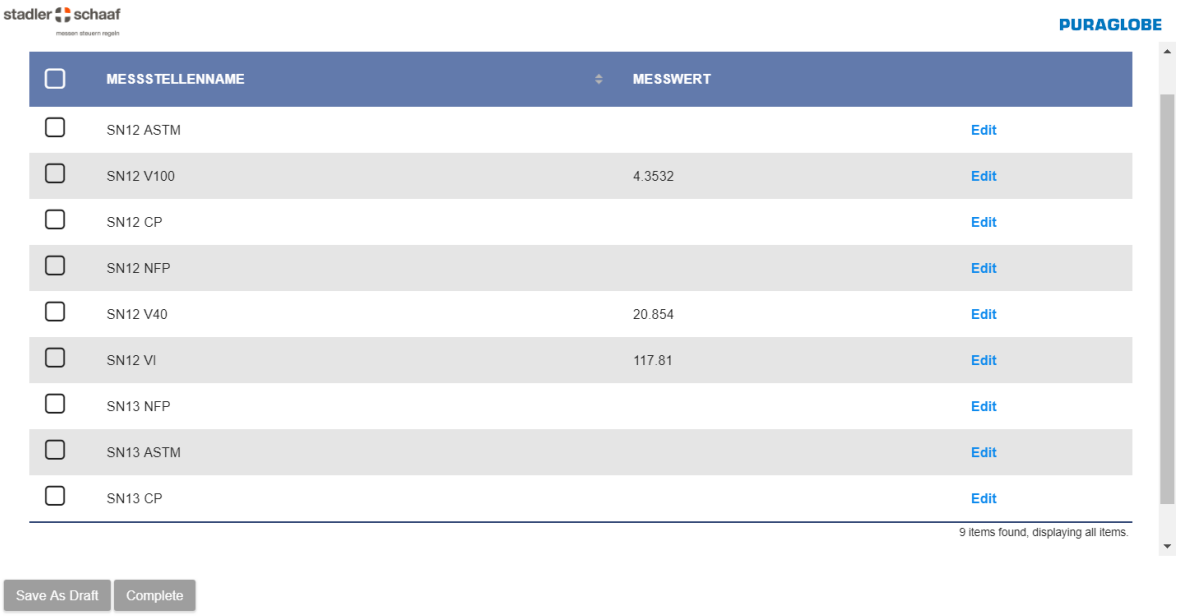


Abbildung 3-12: Activity Eingabemaske

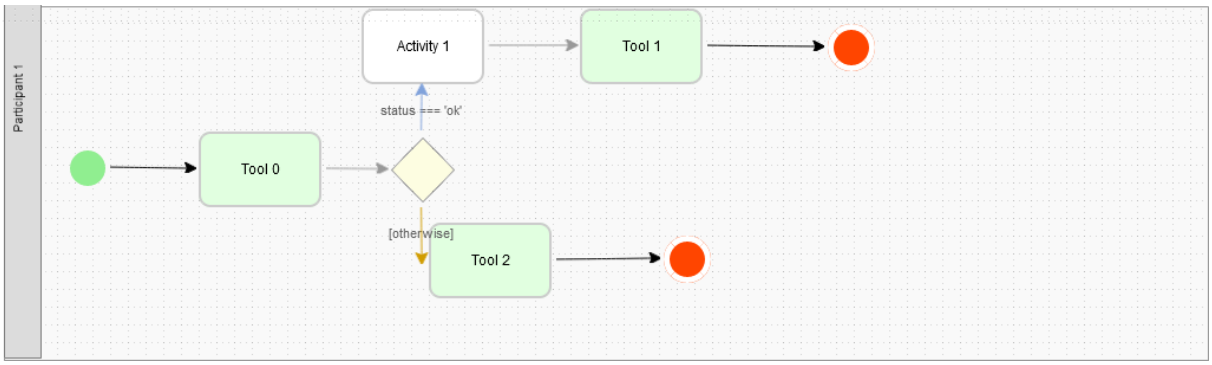


Abbildung 3-13: Prozess Eingabemaske

3.3.4 Zusammenfassung der Implementierung

Ziel der Bewertung ist es, zu beurteilen, inwieweit die Anforderungen mit der implementierten Joget Workflow App erfüllt wurden. Die App konnte im Rahmen einer Kundenpräsentation erfolgreich vorgestellt werden. Die Zusammenfassung der Beurteilung kann der Tabelle 3-6 entnommen werden.

Tabelle 3-6: Zusammenfassung

Anforderung	Erfüllt/ nicht erfüllt?	Lösung
Erstellen einer Eingabemaske, in der die Messstellen und Messgrößen angezeigt werden, aus denen Proben entnommen wurden	Erfüllt	- Abfrage des inmation Werts per *GET*-Anfrage auf „Platzhalter“ - Einfügen der Objekte in Eingabemaske per SQL *Insert into* Befehlen
Einbinden der Daten eines Messgeräts in die Eingabemaske	Erfüllt	- Mithilfe des Prozesses „Messgerät“ - Dieser Prozess verarbeitet die vom Messgerät auflaufenden Messdateien und fügt diese der Eingabemaske hinzu
Übermittlung der eingegebenen Daten an inmation	Erfüllt	- Sammeln der Daten per SQL *Select* Befehl - Übermitteln der Daten an inmation per *POST*-Anfrage
Messwert und Zeitstempel einer Messung zusammenfassen	Erfüllt	- Verwendung des Zeitstempels bei *POST*-Anfrage, bei dem „Platzhalter“ in das inmation-Objekt geschrieben wurde

4 Fazit und Ausblick

Im vorhergehenden Kapitel konnte gezeigt werden, dass die Aufgabenstellung der automatisierungstechnisch unterstützten Integration von Labordaten in das Produktionsleitsystem einer verfahrenstechnischen Anlage vollständig erfüllt wurde. Es wurde die Ausgangslage analysiert, mögliche software- und programmtechnische Lösungsmöglichkeiten erarbeitet und die Implementierung in Form einer Joget Workflow App ausgeführt. Es wurden *Forms* erstellt, in denen die während des Programmablaufs nötigen Informationen eingegeben werden können. Es wurden Prozesse erstellt, die auf diese *Forms* verweisen bzw. BeanShell Programmcode ausführen. Somit konnte ein Arbeitsablauf zum Überführen von Labordaten in die Datengrundlage eines Produktionsleitsystem erstellt werden.

In Zukunft gilt es, das Feedback von Laboranten und Prozessingenieuren einzuholen um Verbesserungen an der Joget Workflow App, bzw. dem Arbeitsablauf vorzunehmen. Verbesserungsmöglichkeiten sind:

- der bestehenden App weitere Messgeräte hinzuzufügen, um das Labordatenmanagement weiter zu automatisieren. Dazu müssten die von den einzelnen Messgeräten ausgegeben Dateien analysiert und der Prozess „Messgerät“ erweitert werden,
- den Fortschritt der Messdatenaufnahme und -eingabe zu protokollieren,
- den Messdaten weitere Informationen hinzuzufügen, z.B.:
    - wer die Messdaten aufgenommen hat,
    - Bemerkungen zu den Messwerten.

Literaturverzeichnis

[1] Stadler + Schaaf Mess- und Regelungstechnik GmbH, „STADLER + SCHAAF," [Online]. Available: https://www.stadler-schaaf.de/. [Zugriff am 31.05.2019].

[2] Stadler + Schaaf Mess- und Regelungstechnik GmbH, „CHEMIE & PETROCHEMIE," [Online]. Available: https://www.stadler-schaaf.de/branche/chemie-petrochemie/. [Zugriff am 31.05.2019].

[3] Chemie- und Industriepark Zeitz, „PURAGLOBE GMBH," Infra-Zeitz Servicegesellschaft mbH, [Online]. Available: https://www.industriepark-zeitz.de/unternehmen/industriebetriebe/puralube-gmbh-und-puralube-raffinerie-3-gmbh/. [Zugriff am 11.04.2019].

[4] A. Roth, Einführung und Umsetzung von Industrie 4.0, Berlin, Heidelberg: Springer-Verlag, 2016.

[5] PURAGLOBE Inc., „SUSTAINABLE BASE OILS," [Online]. Available: https://puraglobe.com/sustainable-base-oils/. [Zugriff am 11.04.2019].

[6] H. M. D. Goldmann, Recylingtechnik Fachbuch für Lehre und Praxis, Wiesbaden: Springer Vieweg, 2016.

[7] „Puralube investiert weiter in die Altölaufbereitung," CHEManager, [Online]. Available: https://www.chemanager-online.com/news-opinions/nachrichten/puralube-investiert-weiter-die-altoelaufbereitung. [Zugriff am 11.04.2019].

[8] V. V. Vaclav Stepina, Lubricants and Special Fluids, Bratislava: Elsevier, 1992.

[9] Chemtronic Waltemode GmbH, „ASTM D-1500 Farbzahl / Saybolt Farbzahl," [Online]. Available: http://www.chemtronic-gmbh.de/images/chemtronic/Apps_d_pdf/ASTM%20D1500%20Farbzahl%20_d_.pdf. [Zugriff am 03.06.2019].

[10] Chemtronic Waltemode GmbH, [Online]. Available: http://www.chemtronic-gmbh.de/images/chemtronic/Apps_d_pdf/APHA%20Farbzahl%20_d_.pdf. [Zugriff am 03.06.2019].

[11] U. J. M. Wifried J. Bartz, Expert Praxislexikon Tribologie Plus, Renningen: expert Verlag, 2000.

[12] H. Watter, Hydraulik und Pneumatik Grundlagen und Übungen - Anwendungen und Simulation, Wiesbaden: Springer Vieweg, 2017.

[13] B. S. Waldemar Steinhilper, Konstruktionselemente des Maschienenbaus 2, Berlin, Heidelberg: Springer Vieweg, 2012.

[14] G. D. E. W. Dean, „Viscosity Variations of Oils with Temperature," *Chemical and Metallurgical Engineering ,* Bd. 36, Nr. 10, pp. 618-619, 1929.

[15] Anton Paar GmbH, *Betriebsanleitung SVM 2001/3001/4001 Stabinger Viskosimeter,* Graz: Anton Paar GmbH, 2017.

[16] Anton Paar GmbH, *Reference Guide LIMS Bridge,* Graz, 2018.

[17] Anton Paar GmbH, *Reference Guide General Software Functions M Series Instruments,* Graz, 2018.

[18] O. Leps, Hybride Testumgebungen für Kritische Infrastrukturen, Berlin: Springer Vieweg, 2018.

[19] M. Kropik, Produktionsleitsysteme in der Automobilfertigung, Berlin, Heidelberg: Springer-Verlag, 2009.

[20] L. Schleupner, Fachlexikon MES & Industrie 4.0, Berlin: VDE Verlag GmbH, 2018.

[21] I. D. P. C. Pascal Blanc, „A holonic approach for manufacturing execution design: An industrial application," Elsevier, Nantes, Marseille, 2008.

[22] inmation, „Pressemitteilung: inmation Software schließt Partnerschaft mit Automatisierungsspezialist Delta Logic," inmation, [Online]. Available: http://www.inmation.com/company/localized-news/item/pressemitteilung-inmation-software-schliesst-partnerschaft-mit-automatisierungsspezialist-delta-logic-2. [Zugriff am 06.05.2019].

[23] inmation, „System Documentation Chapter 1. Product Overview," inmation, [Online]. Available: https://inmation.com/wiki/Sysdoc/Product_Overview. [Zugriff am 06.05.2019].

[24] inmation, „System Documentation Chapter 12. The Repository," [Online]. Available: https://inmation.com/wiki/Sysdoc/The_Repository. [Zugriff am 07.05.2019].

[25] inmation, „I/O Model,“ [Online]. Available: https://inmation.com/docs/datastudio/1.50/model-panels/io-model.html. [Zugriff am 03.06.2019].

[26] inmation, „Dropzone Basics,“ [Online]. Available: https://inmation.com/docs/jumpstarts/1.50/using-the-dropzone-data-source/dropzone-basics.html. [Zugriff am 02.06.2019].

[27] inmation, „System Documentation Chapter 17. Web API Service,“ inmation. [Online]. Available: https://inmation.com/wiki/Sysdoc/Web_API_Service. [Zugriff am 06.05.2019].

[28] A. Kumar, Web Technology Theory and Practice, Boca Raton: Taylor & Francis Group, LLC, 2019.

[29] J. R. R. Fielding, „Hypertext Transfer Protocol (HTTP/1.1) : Semantics and Content,“ Internet Engineering Task Force, San Jose, Münster, 2014.

[30] M. H. A. S. Diimitrios Georgakopoulus, „An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure,“ *Distributed and Parallel Databases,* Bd. 3, Nr. 2, pp. 119-153, 1995.

[31] Joget, Inc., „About Joget,“ [Online]. Available: https://www.joget.org/about-us/. [Zugriff am 06.05.2019].

[32] Joget, Inc., „Apps and the App Center,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Apps+and+the+App+Center. [Zugriff am 06.05.2019].

[33] Joget, Inc., „Userviews, Forms, Lists and Processes,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Userviews%2C+Forms%2C+Lists%2C+and+Processes. [Zugriff am 06.05.2019].

[34] Joget, Inc., „Form,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Form. [Zugriff am 07.05.2019].

[35] Joget, Inc., „Run Process Menu,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Run+Process+Menu. [Zugriff am 07.05.2019].

[36] Joget, Inc., „Text Field,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Text+Field. [Zugriff am 08.05.2019].

[37] Joget, Inc., „Custom HTML,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Custom+HTML. [Zugriff am 08.05.2019].

[38] Joget, Inc., „Datalist Builder,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Datalist+Builder. [Zugriff am 13.05.2019].

[39] A. Gadatsch, Grundkurs Geschäftsprozess-Management, Wiesbaden: Vieweg+Teubner, 2010.

[40] Joget, Inc., „Process Builder,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Process+Builder. [Zugriff am 13.05.2019].

[41] Joget, Inc., „Deadlines and Escalations,“ [Online]. Available: https://dev.joget.org/community/display/KBv5/Deadlines+and+Escalations. [Zugriff am 13.05.2019].

[42] Joget, Inc., „Bean Shell Programming Guide,“ [Online]. Available: https://dev.joget.org/community/display/KBv6/Bean+Shell+Programming+Guide. [Zugriff am 05.06.2019].

[43] E. Schicker, Datenbanken und SQL, Wiesbaden: Springer Vieweg, 2017.

[44] A. Meier und M. Kaufmann, SQL- & NoSQL-Datenbanken, Berlin Heidelberg: Springer Vieweg, 2016.

[45] A. G. Norbert Gronau, Einführung in die Wirtschaftsinformatik Band 2, Berlin: GITO mbh Verlag, 2012.

[46] Joget, Inc., „Embed Joget Datalist inside Joget Form,“ [Online]. Available: https://dev.joget.org/community/display/FORUM/Embed+Joget+Datalist+inside+Joget+Form. [Zugriff am 22.05.2019].

Anhang

Anhangsverzeichnis

Anhang A Quelltexte

Anhang A1 Quelltext BeanShell Tool Prozess Messgerät Tool 1

```
import java.net.*;
import java.io.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.sql.DataSource;
import org.joget.apps.app.service.AppUtil;
import org.joget.apps.form.model.Element;
import org.joget.apps.form.model.FormData;
import org.joget.apps.form.model.FormRow;
import org.joget.workflow.model.service.*;
import org.joget.apps.form.model.FormRowSet;
import org.joget.commons.util.LogUtil;
import java.util.*;
import java.lang.*;
import java.text.*;
import java.time.*;

// erstellen eines neuen Datumobjekts
Date now=new Date();
Date neuesDate=new Date();
// erstellen der Messstellenvariable
String messstellelesen="";
// checkschleife --> Überüprüfungsvariable
int checkschleife=1;
// mache 25 mal
for (int i=0;i<25;i++){// gehe immer eine Sekunde zurück
        now.setSeconds(now.getSeconds()-1);
        // Datumsformatierung
        SimpleDateFormat sdf= new SimpleDateFormat("yyyyMMdd_HHmmss");
        String zeitneu=sdf.format(now.getTime());
        // Dateinamenzusammensetzung
        String dateiname ="test_" +zeitneu+".csv";
        File f =null;
        //Erstelle Datei
        f = new File("C:\\Users\\Administrator\\Desktop\\textdateienmessgeraet\\"+dateiname);
        // Falls Datei existiert
        if (f.isFile()){
                // setze nötige Variablen zurück
                int idint=0;
                String ID="";
                String IDtest="";
                String messstellenname="";
                String v40="";
                String v100="";
                String VI="";
                String messwert="";
                records = new ArrayList();
                // Schreibe die Datei File zeile für zeile in Array records mit den einzelnen
                durch ; getrennten Werten
                FileReader fre = new FileReader("C:\\Users\\Administrator\\Desktop\\”+
                ”textdateienmessgeraet\\"+dateiname);
                BufferedReader br = new BufferedReader(fre) ;
```

```
String line;
while ((line = br.readLine()) != null) {
	String[] values = line.split(";");
	records.add(Arrays.asList(values));
}
br.close();
fre.close();
// Hole die einzelnen wichtige Elemente
String test= records.get(3).get(0);
String neu = "";
messstellenname= records.get(3).get(1);
v40=records.get(3).get(6);
v100=records.get(3).get(23);
VI=records.get(3).get(94);
// Stelle Verbindung zur Datenbank her
DataSource ds = (DataSource)AppUtil.getApplicationContext().getBean("setupDataSource");
con = ds.getConnection();
// Falls die Verbindung besteht
if(!con.isClosed()) {
	// Hole die in der Tabelle vorhandenen Einträge
	PreparedStatement statements = con.prepareStatement(
	"SELECT * from app_fc_neu");
	ResultSet rese = statements.executeQuery();
	// Solange es eine weitere Zeile gibt
	while (rese.next()) {
		// Überprüfe ob der Messstellenname schon vorhanden ist
		idint=rese.getInt("IDtest");
		messstellelesen=rese.getString("c_messstellenname");
		if (messstellelesen.contains(messstellenname)){
			checkschleife=0;
		}
	}
	// Falls Messstellenname noch nicht vorhanden ist
	if (checkschleife==1){
		// Schreibe die Messwerte in die Tabelle
		idint=idint+1;
		ID=Integer.toString(idint);
		if (ID.equals("0")){
			ID="1";
		}
		// Insert für v40
		PreparedStatement statement = con.prepareStatement("
		INSERT INTO app_fc_neu (IDtest, c_messstellenname,c_messwert)
		VALUES ('"+ID+"','"+messstellenname+" V40"+"','"+v40+"')");
		statement.executeQuery();
		idint=idint+1;
		ID=Integer.toString(idint);
		// Insert für v100
		statement = con.prepareStatement("INSERT INTO app_fc_neu (
		IDtest, c_messstellenname,c_messwert) VALUES    (
		'"+ID+"','"+messstellenname+"V100"+"','"+v100+"')");
		statement.executeQuery();
		idint=idint+1;
		ID=Integer.toString(idint);
		// Insert für VI
		statement = con.prepareStatement("INSERT INTO app_fc_neu (
		IDtest,    c_messstellenname,c_messwert) VALUES(
		'"+ID+"','"+messstellenname+"VI"+"','"+VI+"')");
		statement.executeQuery();
```

```
                    }
                    con.close();
                }
            }
}
```

Anhang A2 Quelltext BeanShell Tool Prozess Eingabemaske Tool 1

```
import java.net.*;
import java.io.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.sql.DataSource;
import org.joget.apps.app.service.AppUtil;
import org.joget.apps.form.model.Element;
import org.joget.apps.form.model.FormData;
import org.joget.apps.form.model.FormRow;
import org.joget.apps.form.model.FormRowSet;
import org.joget.commons.util.LogUtil;
import java.util.*;
import java.lang.*;

String ueberpruefen="zahl";
String responseBodynew="";
int result= 0;
int resultneu=0;
Connection con = null;//Verbindungsvariable
String stelle="";//messstellenname
String ipadresse="";
String ordnername="";
String teststring="test";
String responseBody="";
String data="";
String wert="";//eingetragener Wert fuer die Messstelle
DataSource ds = (DataSource)AppUtil.getApplicationContext().getBean("setupDataSource");
con = ds.getConnection();//Baue Verbindung zur Datenbank auf
if(!con.isClosed()) {//Falls sie hergestelt werden konnte
        // Hole alle eintraege zu einstellungen
        PreparedStatement stamt = con.prepareStatement("SELECT * from app_fd_einstellungen");
        ResultSet res = stamt.executeQuery();//fuehre die Query aus
        while (res.next()) {
                ipadresse=res.getString("c_ipadresse");
                ordnername=res.getString("c_ordnername");
        }
        // Hole alle eintraege zur eingabe
        PreparedStatement stmt = con.prepareStatement("SELECT * from app_fd_eingabeoption");
        ResultSet rs = stmt.executeQuery();//fuehre die Query aus
        while (rs.next()) {// fuer jede zeile tue
                stelle=rs.getString("c_messstellenname");//hole den namen der messstelle
                stelle=stelle.replaceAll(" ","%20");
                wert=rs.getString("c_messwert");//hole den wert der messstelle
                if (wert.equals("")){
                        wert="Platzhalter";
                }
                ueberpruefen="zahl";
                if (wert.contains(",")){
                        wert=wert.replaceAll(",",".");
                }
                try {
                        double retval = Double.parseDouble(wert);
                }
                catch(NumberFormatException e){
                        ueberpruefen="string";
```

```
		}
		//Baue den string zum Schreiben zusammen
		// System..Labor ist die Ordnerstruktur in inmation
		//String data ="{\n\"items\": [ \n{\n\"p\": \"/System/Training/Labor/"+stelle+"\",\n\"v\": \""+wert+
		"\" ,\n}\n]\n}";
		//Setze die Url fuer die inmationbfrage zusammen
		String urlneu = "http://"+ipadresse+":8002/api/v2/read";
		urlneu=urlneu+"?identifier="+ordnername+stelle;
		// /System...Labor/ ist die Ordnerstruktur ueber den einzelnen Stellen
		// stelle die Verbindung her
		URLConnection verbindungneu = new URL(urlneu).openConnection();
		verbindungneu.setRequestProperty("Content-Type","application/json");// setze die Parameter
		verbindungneu.setRequestProperty("Accept","application/json");//
		verbindungneu.setRequestProperty("username","so");//
		verbindungneu.setRequestProperty("password","inmation");//
		int resultneu = verbindungneu.getResponseCode();//Falls result 200 steht die Verbindung
		// in responseneu steht die Antwortals Inputstream
		InputStream responseneu = verbindungneu.getInputStream();
		BufferedReader burneu = new BufferedReader(new InputStreamReader(
		responseneu, "utf8"));// Hier beginnt die Umwandlung des Inputstreams in einen String
		StringBuffer sbneu = new StringBuffer();
		String lineneu = "";
		while ((lineneu = burneu.readLine()) != null) {
			sbneu.append(lineneu);
		}
		responseBody = sbneu.toString();
		String[] parts=responseBody.split("t\":");
		String[] teile=parts[1].split("\"");
		String zeit=teile[1];
		burneu.close();
		responseneu.close();
		stelle=stelle.replaceAll("%20"," ");
		if (ueberpruefen.equals("string")){
			data ="{\n\"items\": [ \n{\n\"p\": \""+ordnername+stelle+"\",\n\"v\": \""+wert+
			"\" ,\n\"t\":\""+zeit+"\",\n}\n]\n}";
		}
		else{
			data ="{\n\"items\": [ \n{\n\"p\": \""+ordnername+stelle+"\",\n\"v\": "+wert+
			" ,\n\"t\": \""+zeit+"\",\n}\n]\n}";
		}
		// baue die Url fuer den inmationschreibbefehl
		URL url = new URL("http://"+ipadresse+":8002/api/v2/write");
		// stelle die verbindung her
		HttpURLConnection verbindung = (HttpURLConnection) url.openConnection();
		verbindung.setRequestMethod("POST");// setze parameter
		verbindung.setRequestProperty("Content-Type", "application/json");//
		verbindung.setRequestProperty("Accept","application/json");//
		verbindung.setRequestProperty("username", "so");//
		verbindung.setRequestProperty("password", "inmation");//
		verbindung.setDoOutput(true);//
		verbindung.getOutputStream().write(data.getBytes("UTF-8"));// schreibe den befehl
		result= verbindung.getResponseCode();// falls result 200 ist hat es geklappt
}
// Hole alle eintraege vom Messgerät
PreparedStatement stamtneu = con.prepareStatement("SELECT * from app_fc_neu");
ResultSet resneueste = stamtneu.executeQuery();
while (resneueste.next()) {
		responseBodynew="Platzhalter";
		String messstellennamefcneu = resneueste.getString("c_messstellenname");
```

```
                messstellennamefcneu=messstellennamefcneu.replaceAll(" ","%20");
                //Setze die Url fuer die inmationbfrage zusammen
                String urlnew = "http://"+ipadresse+":8002/api/v2/read";
                urlnew=urlnew+"?identifier="+ordnername+messstellennamefcneu;
                // stelle die Verbindung her
                URLConnection verbindungnew = new URL(urlnew).openConnection();
                verbindungnew.setRequestProperty("Content-Type","application/json");// setze die Parameter
                verbindungnew.setRequestProperty("Accept","application/json");//
                verbindungnew.setRequestProperty("username","so");//
                verbindungnew.setRequestProperty("password","inmation");//
                resultneu = verbindungnew.getResponseCode();//Falls result 200 steht die Verbindung
                if (resultneu==200){
                        // in response steht die Antwort als Inputstream
                        InputStream response = verbindungnew.getInputStream();
                        BufferedReader bur = new BufferedReader(new InputStreamReader(
                        response, "utf8"));// Hier beginnt die Umwandlung des Inputstreams in einen String
                        StringBuffer sb = new StringBuffer();//
                        String line = "";
                        while ((line = bur.readLine()) != null) {//
                                sb.append(line);//
                        }
                        // hier endet die Umwandlung nun ist responsebody die antwort als string
                        responseBodynew = sb.toString();
                        // falls responsebody platzhalter enthaelt
                        if (responseBodynew.contains("Platzhalter")==false){
                                messstellennamefcneu=messstellennamefcneu.replaceAll("%20"," ");
                                PreparedStatement anweisungnew = con.prepareStatement(
                                "Delete from app_fc_neu  where c_messstellenname='
                                "+messstellennamefcneu+"'");
                                // Schreibe in die Tabelle die einzelnen Stellen
                                anweisungnew.executeQuery();
                        }
                        bur.close();
                        response.close();
                }
        }
        con.close();//Ende
}
```

Anhang A3 Quelltext BeanShell Tool Prozess Eingabemaske Tool 0

```
import java.net.*;
import java.io.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.sql.DataSource;
import org.joget.apps.app.service.AppUtil;
import org.joget.apps.form.model.Element;
import org.joget.apps.form.model.FormData;
import org.joget.apps.form.model.FormRow;
import org.joget.workflow.model.service.*;
import org.joget.apps.form.model.FormRowSet;
import org.joget.commons.util.LogUtil;
import java.util.*;
import java.lang.*;

int result=0;
String url="";
Connection con = null; // Variable für die Verbindung
String stelle="";// Variable für den Stellennamen
String responseBody="";// Variable für die Antwort der inmationabfrage
String ipAdresse="";
String ordnername="";
int k=0;//Zaehlvariable fuer die Messstellen um diese in eine Tabelle zu ueberfuehren
DataSource ds = (DataSource)AppUtil.getApplicationContext().getBean("setupDataSource");//
con = ds.getConnection();//Verbindung zur Datenbank aufbauen
if(!con.isClosed()) {//Falls die Verbindung hergestellt werden konnte
        // Loesche die Zeilen in der          Eingabetabelle
        PreparedStatement stmt = con.prepareStatement("Delete from app_fd_eingabeoption");
        stmt.executeQuery();//Fuehre die query aus
        // Hole alle Messstellen
        PreparedStatement statements = con.prepareStatement("SELECT * from app_fd_einstellungen");
        ResultSet rese = statements.executeQuery();//Fuehre die Query aus
        while (rese.next()) {
                ipAdresse=rese.getString("c_ipadresse");
                ordnername=rese.getString("c_ordnername");
        }
        // Hole alle Messstellen
        PreparedStatement statement = con.prepareStatement("SELECT * from app_fd_messstellen");
        ResultSet res = statement.executeQuery();//Fuehre die Query aus
        while (res.next()) {// Fuer jede Zeile tue
                k=k+1;//Zaehle k eins hoch
                stelle=res.getString("c_namederstelle");// Hole den Stellennamen
                stelle=stelle.replaceAll(" ","%20");
                url = "http://"+ipAdresse+":8002/api/v2/read";
                url=url+"?identifier="+ordnername+stelle;//Setze die Url fuer die inmationbfrage zusammen
                URLConnection verbindung = new URL(url).openConnection();// stelle die Verbindung her
                verbindung.setRequestProperty("Content-Type","application/json");// setze die Parameter
                verbindung.setRequestProperty("Accept","application/json");//
                verbindung.setRequestProperty("username","so");//
                verbindung.setRequestProperty("password","inmation");//
                result = verbindung.getResponseCode();//Falls result 200 steht die Verbindung
                if (result==200){
                        WorkflowManager wm = (WorkflowManager) pluginManager.getBean(
                        "workflowManager");
                        wm.activityVariable(workflowAssignment.getActivityId(),"status", "ok");
```

```
                        // in response steht die Antwort als Inputstream
                        InputStream response = verbindung.getInputStream();
                        // Hier beginnt die Umwandlung des Inputstreams in einen String
                        BufferedReader bur = new BufferedReader(new InputStreamReader(
                        response, "utf8"));
                        StringBuffer sb = new StringBuffer();
                        String line = "";
                        while ((line = bur.readLine()) != null) {
                                sb.append(line);
                        }
                        responseBody = sb.toString();
                        // hier endet die Umwandlung        nun ist responsebody die antwort als string
                        if (responseBody.contains("Platzhalter")){// falls responsebody platzhalter enthaelt
                                stelle=stelle.replaceAll("%20"," ");
                                PreparedStatement anweisung = con.prepareStatement("INSERT INTO
                                app_fd_eingabeoption (id,c_messstellenname,c_messwert) VALUES
                                ('"+Integer.toString(k)+"','"+stelle+"','')");
                                anweisung.executeQuery();// Schreibe in die Tabelle die einzelnen Stellen
                        }
                }
        }
        // Hole alle Messstellen
        PreparedStatement statementneu = con.prepareStatement("SELECT * from app_fd_eingabeoption");
        ResultSet resneu = statementneu.executeQuery();//Fuehre die Query aus
        while (resneu.next()) {
                String messstellennamemessgeraet=resneu.getString("c_messstellenname");
                PreparedStatement statementneuer = con.prepareStatement("SELECT * from app_fc_neu
                Wherec_messstellenname='"+messstellennamemessgeraet+"'");// Hole alle Messstellen
                ResultSet resneuer = statementneuer.executeQuery();//Fuehre die Query aus
                while (resneuer.next()) {
                        String messwertmessgeraet=resneuer.getString("c_messwert");
                        PreparedStatement anweisungneu = con.prepareStatement("Update
                         app_fd_eingabeoption set c_messwert='"+messwertmessgeraet+"' where
                        c_messstellenname='"+messstellennamemessgeraet +"'");
                        anweisungneu.executeQuery();
                        WorkflowManager wm = (WorkflowManager) pluginManager.getBean("
                        workflowManager");
                        wm.activityVariable(workflowAssignment.getActivityId(),"status", "ok");
                }
        }
        con.close();// Ende
}
```

Anhang B inmation Swagger

Anhang B1 Get /api/v2/read

Curl

```
--header 'Accept: application/json' --header 'username: so' --header 'password: inmation' 'http://localhost:8002/api/v2/read?identi
```

Request URL

```
http://localhost:8002/api/v2/read?identifier=%2FSystem%2FTraining%2FLabor%2F2SN12%20CP
```

Response Body


```
{
  "data": [
    {
      "p": "/System/Training/Labor/2SN12 CP",
      "v": 80,
      "q": 0,
      "t": "2019-04-29T14:15:25.393Z"
    }
  ]
}
```

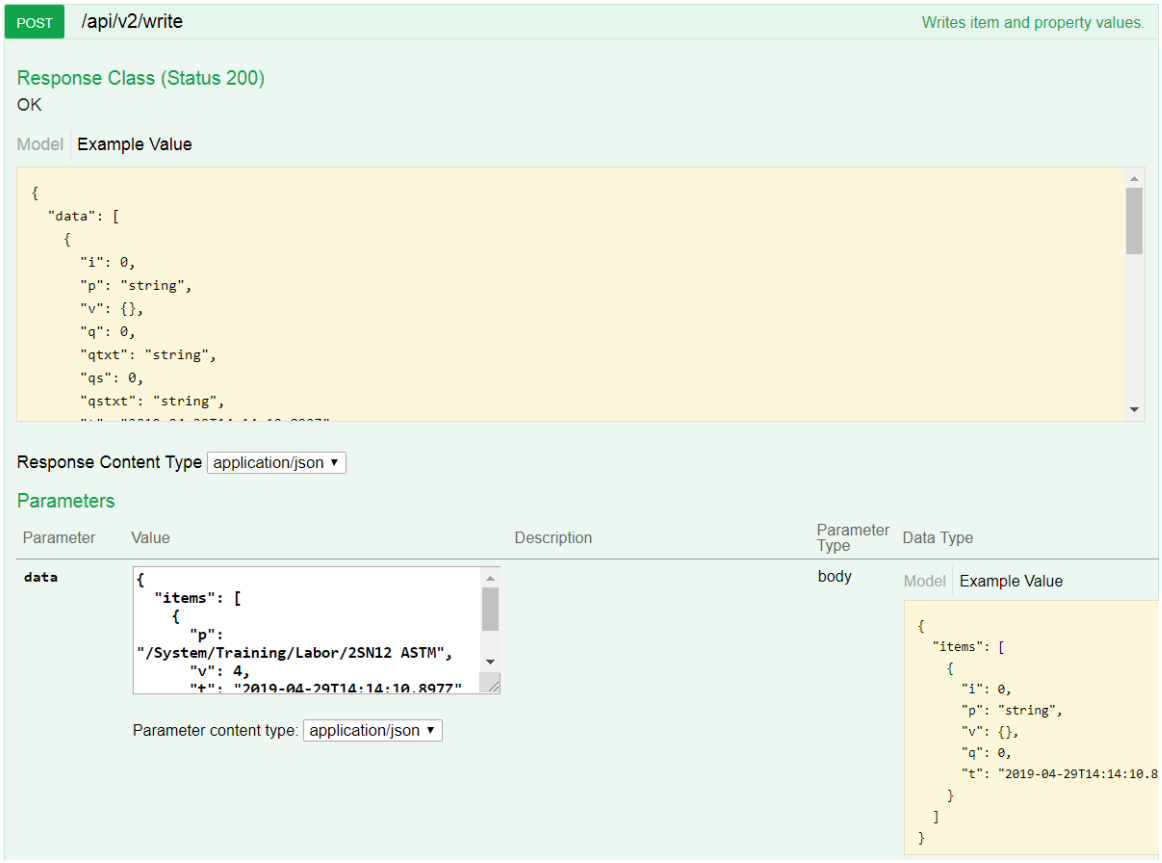

Response Code

```
200
```

Response Headers


```
{
  "date": "Mon, 29 Apr 2019 14:15:25 GMT",
  "server": "Microsoft-HTTPAPI/2.0",
  "content-length": "96",
  "content-type": "application/json; charset=utf-8"
}
```

Anhang B2 Post /api/v2/write



Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'username: so' --header 'passw
   "items": [ \
     { \
       "p": "/System/Training/Labor/2SN12 ASTM", \
       "v": 4, \
       "t": "2019-04-29T14:14:10.897Z" \
     } \
   ] \
 }' 'http://localhost:8002/api/v2/write'
```

Request URL

```
http://localhost:8002/api/v2/write
```

Response Body


```
{
  "data": [
    {
      "p": "/System/Training/Labor/2SN12 ASTM",
      "v": 4,
      "t": "2019-04-29T14:14:10.897Z"
    }
  ]
}
```


Response Code

```
200
```

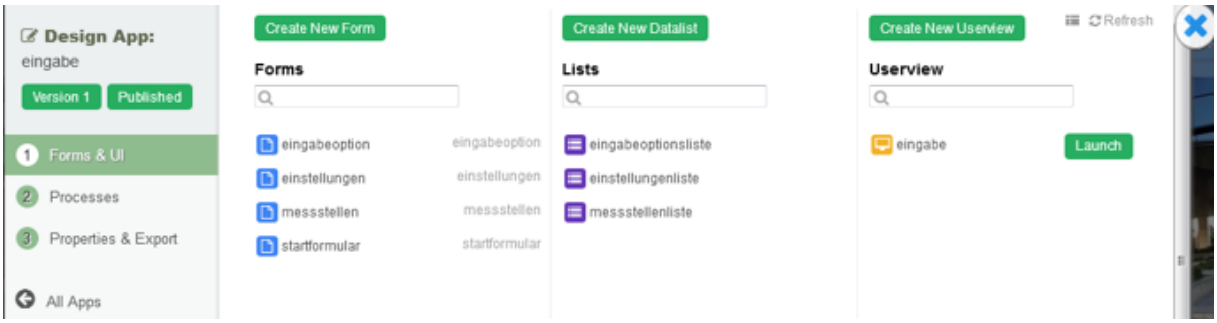
Response Headers


```
{
  "access-control-allow-origin": "*",
  "date": "Mon, 29 Apr 2019 14:18:32 GMT",
  "server": "Microsoft-HTTPAPI/2.0",
  "content-length": "89",
  "content-type": "application/json; charset=utf-8"
}
```

Anhang C Joget App Screenshots

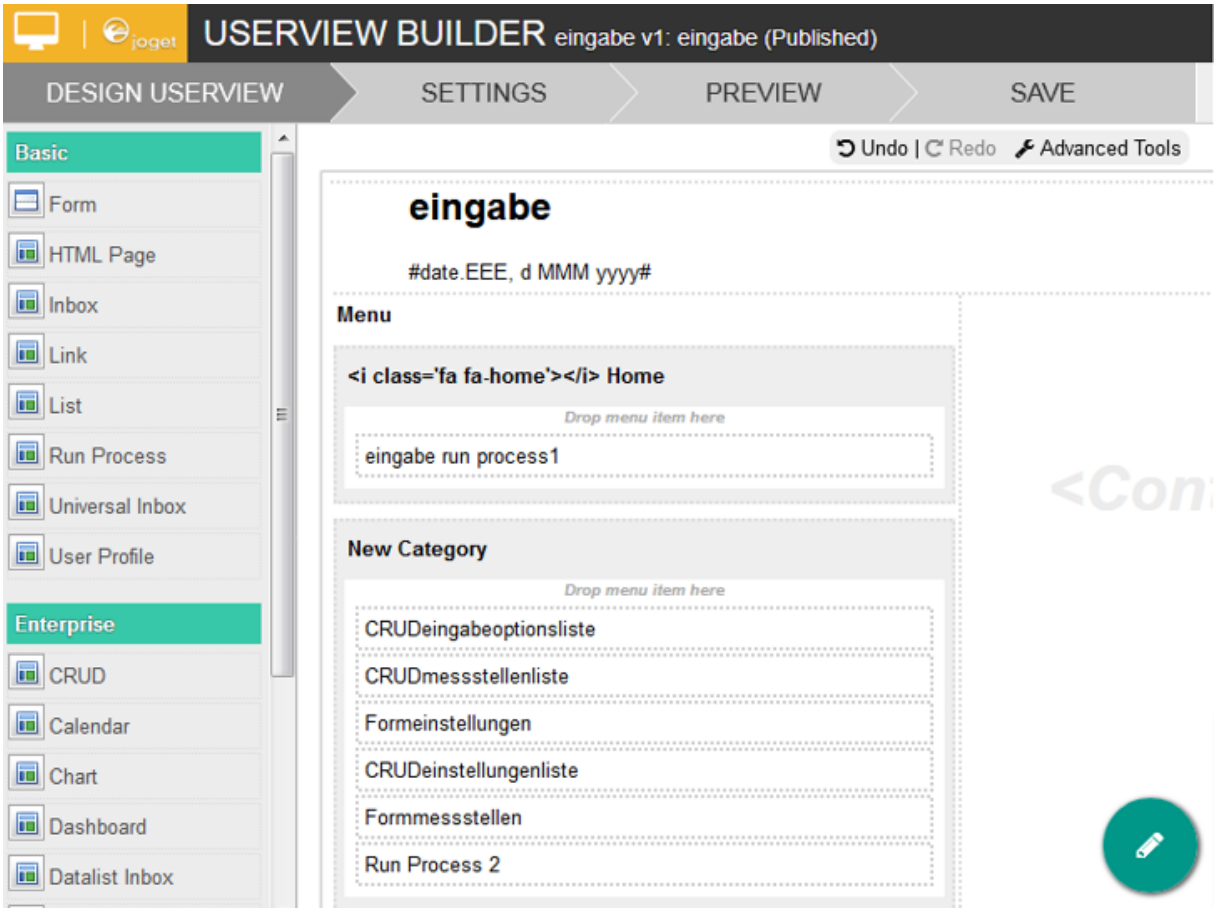
Anhang C1 Design App

Zu sehen ist das Design-App-Fenster der eingabe-App mit den Forms „eingabeoption“, „einstellungen“, „messstellen“ und „startformular“, den Lists „eingabeoptionsliste“, „einstellungenliste“ und „messstellenliste“ und dem Userview „eingabe“.



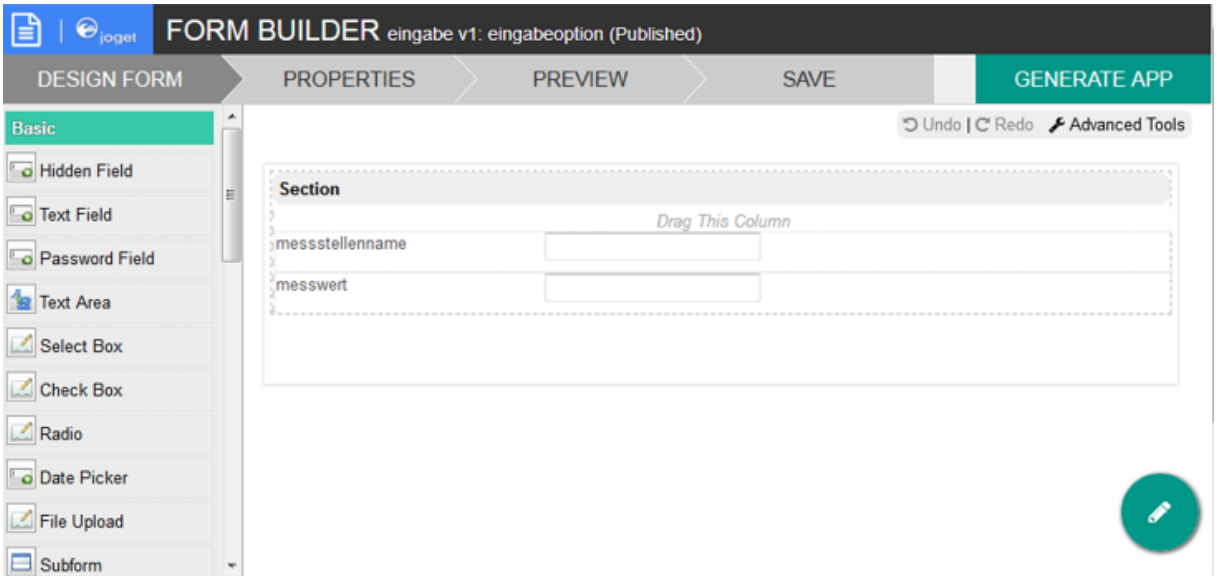
Anhang C2 Userview „eingabe“

Zu sehen sind die Reiter, mit denen auf die erstellten *Forms*, *Datalists* und Prozesse verwiesen wird. „process 1“ ist der Prozess Eingabemaske und „Process 2“ ist der Prozess Messgerät.



Anhang C3 Form „eingabeoption“

Zu sehen ist der *Form Builder* der Form „eingabeoption“, in das die Messwerte von den Laboranten eingetragen werden.

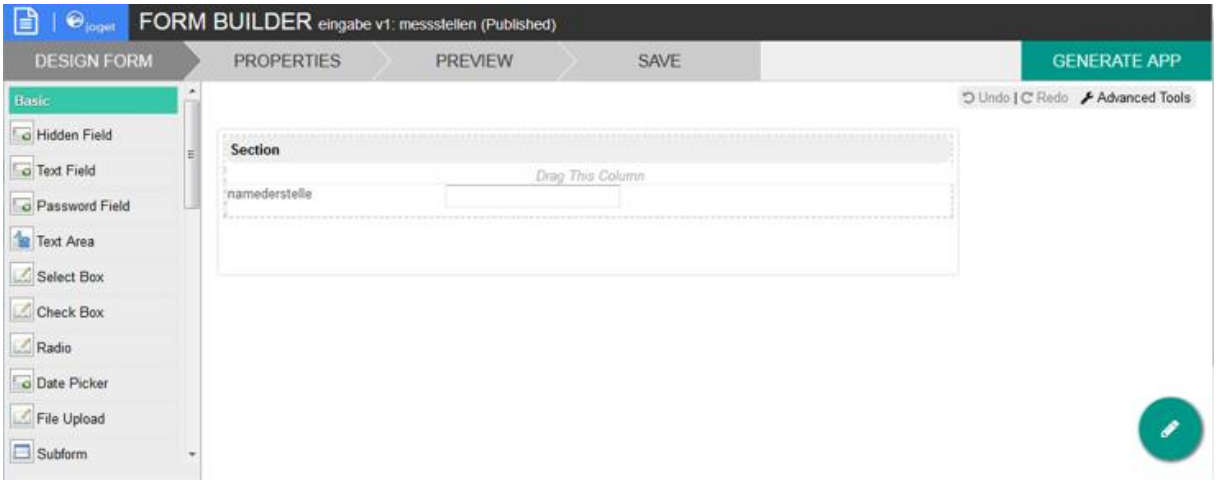


Anhang C4 Form „einstellungen“

Zu sehen ist der *Form Builder* der *Form* „einstellungen“, in das die IP-Adresse des inmation-Servers sowie den Ordnername des über den Laborelementen angeordneten Ordnerpfades enthält.

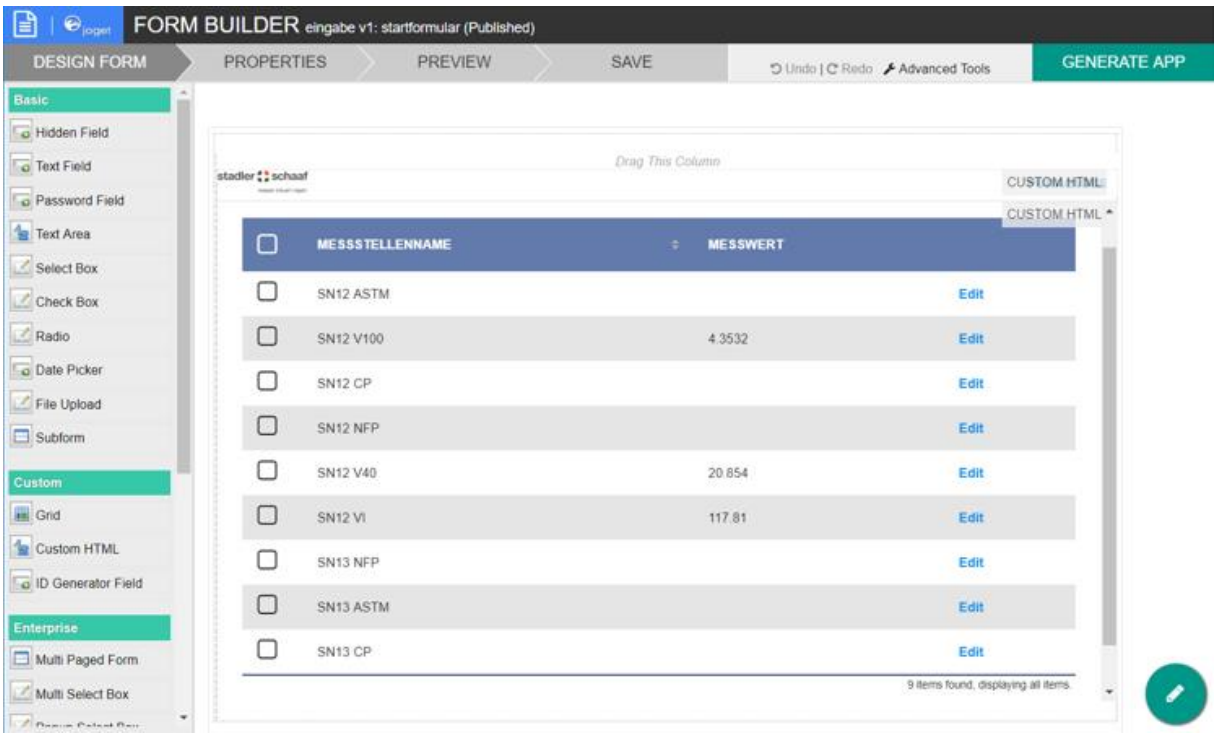
Anhang C5 Form „messstellen“

Zu sehen ist der *Form Builder* der *Form* „messstellen“, in das die in inmation angelegten Messstellen und Messgrößen eingetragen werden können.



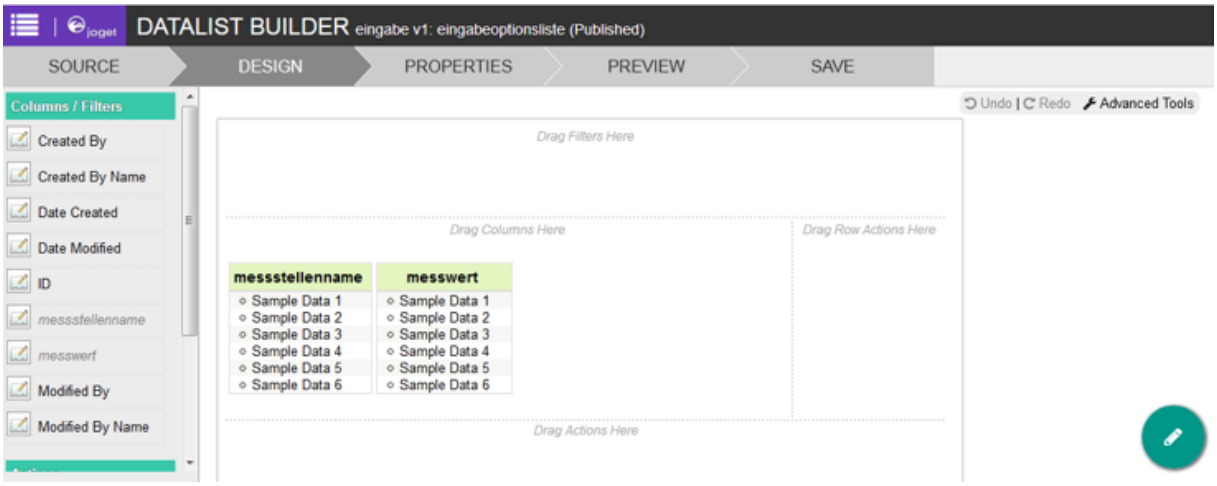
Anhang C6 Form „startformular“

Zu sehen ist der *Form Builder* der *Form* „startformular“, das die Eingabemaske enthält.



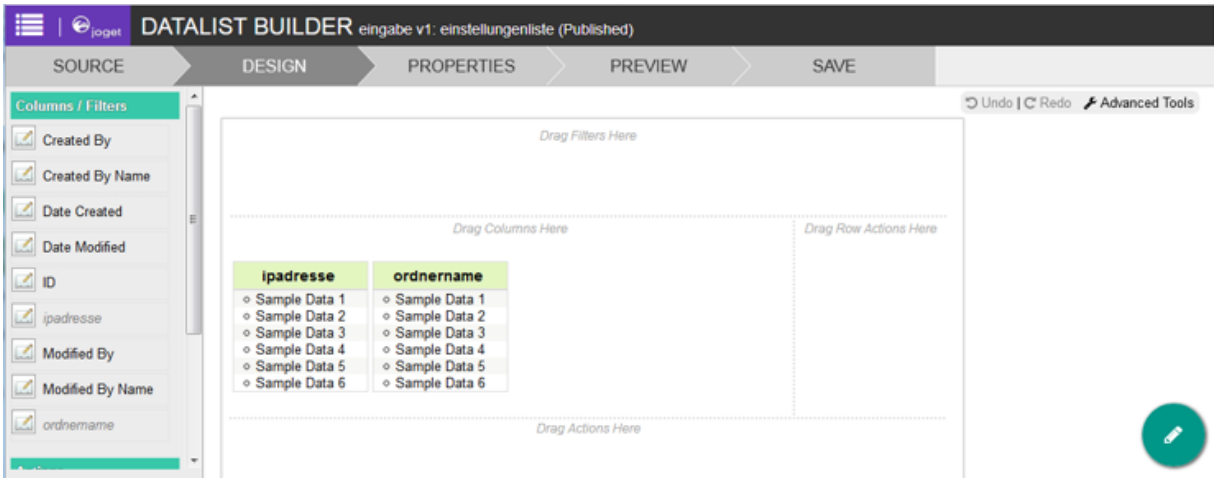
Anhang C7 List „eingabeoptionsliste”

Zu sehen ist der *Datalist Builder* der *List* „eingabeoptionsliste”, in der die Eingaben in die *Form* „eingabeoption“ abgespeichert werden.



Anhang C8 List „einstellungenliste“

Zu sehen ist der *Datalist Builder* der *List* „einstellungenliste”, in der die Eingaben in die *Form* „einstellungen“ abgespeichert werden.



Anhang C9 List „messstellenliste“

Zu sehen ist der *Datalist Builder* der *List* „messstellenliste”, in der die Eingaben in die *Form* „messstellen“ abgespeichert werden.