Entwicklung einer internetbasierten Lieferantenplattform für Qualitätsdaten

Nils Nieuwenhuis (3180646)

Fontys Hochschule für Technik und Logistik

Wirtschaftsinformatik

Bocholt, 22. Februar 2018

Zusammenfassung

Dieses Dokument basiert auf den Ergebnissen einer Bachelorarbeit bei der Flender GmbH in Bocholt. Das Ziel der Arbeit war es, einen Prototypen für eine Lieferantenplattform zur Speicherung und Übertragung von Qualitätsdaten zu entwickeln, um das Unternehmen, dessen Hauptmerkmal die Fertigung von Getrieben und Kupplungen für die Antriebstechnik ist, in diesem Bereich zu unterstützen.

Nach einer Analyse der bisherigen Applikation bezüglich bestehender Probleme und Anforderungen für die Zukunft ist der Entwurf der Systemarchitektur der neuen Applikation erfolgt. Die abschließenden Kapitel dieser Arbeit fokussieren die Umsetzung der grafischen Oberfläche mit ihrer zugehörigen Programmlogik und dem Softwarewerkzeug Servoy.

Während der Arbeit ist ein Prototyp entstanden, der in der Lage ist, Qualitätsdaten in einer grafischen Oberfläche entgegenzunehmen und zu speichern. Das bedeutet, dass die Flender GmbH im Anschluss an die Arbeit den Prototypen als Ausgangsbasis für die Fertigstellung der Lieferantenplattform verwenden kann.

Summary

This document is based on the results of a bachelor thesis at Flender GmbH in Bocholt. The aim of the work was to develop a prototype for a supplier platform for storing and transmitting quality data in order to support the company, whose main characteristic is the production of gearboxes and clutches for drive technology, in this area.

After an analysis of the existing application with regard to existing problems and requirements for the future, the system architecture of the new application was designed. The final chapters of this work focus on the implementation of the graphical user interface with its associated program logic and the Servoy software tool.

During the work a prototype was created, which is able to receive and save quality data in a graphical user interface. This means that Flender GmbH can use the prototype as a basis for the completion of the supplier platform after the work.

Statement of Authenticity

Issued by the FHTenL Examination Board, September 2017

I, the undersigned, hereby certify that I have compiled and written this document and the underlying work / pieces of work without assistance from anyone except the specifically assigned academic supervisor. This work is solely my own, and I am solely responsible for the content, organization, and making of this document and the underlying work / pieces of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents / pieces of work provided by my course / my academic institution, and I understand that this document and the underlying pieces of work will not be accepted for evaluation or for the award of academic credits if it is determined that they have not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g. ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document and the underlying work / pieces of work will not be accepted for evaluation or for the award of academic credits if it is determined that they embody plagiarism.

Name (in capital letters):

_____

Student number: _____

Place / Date: _____

Signature: _____

Inhaltsverzeichnis

Abkürzungsverzeichnis

CRM *Customer Relationship Management*
CSS *Cascading Style Sheets*
ERM *Entity-Relationship-Modell*
ERP *Enterprise-Resource-Planning*
FleQs *Flender Qualitätssystem*
JDBC *Java Database Connectivity*
PLM *Product Lifecycle Management*
RAD *Rapid Application Development*
SCM *Supply Chain Management*
VCS *Versionsverwaltungssystem*
VM *Virtuelle Maschine*
VPN *Virtual Private Network*
WYSIWYG *What you see is what you get*

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

In dieser Arbeit geht es um die Ergebnisse eines Projekts, das im Rahmen einer Bachelorarbeit bei der Flender GmbH in Bocholt stattfand. Auf den folgenden Seiten werden der Ablauf, die Vorgehensweise und die Ergebnisse der Arbeit beschrieben. Während der Bachelorarbeit wurde ein Prototyp für eine Internetschnittstelle für Lieferanten zur Pflege von Qualitätsdaten entwickelt, mit dem Ziel, eine bestehende Lösung zu ersetzen.

Der Umstieg einer veralteten Softwareanwendung zur Qualitätsdatenerfassung auf FoxPro-Basis auf eine moderne Architektur und Programmierumgebung erfordert auch die Ablösung der nicht mehr dem Stand der Technik entsprechenden Kommunikation mit externen Partnern (Lieferant, Kunde). Hier werden aktuell über Mailnachrichten PDF-Dokumente ausgetauscht. Diese Dokumente können nur verwaltet, aber nicht für weitergehende Analysen genutzt werden. Eine bereits bestehende, nur unzureichend ausgestaltete Schnittstelle scheitert an der Akzeptanz der Anwender, da der aktuelle Aufwand zur Stammdatenpflege als Voraussetzung für die Erfassung der Daten sehr hoch ist.

Über neue Kommunikationstechniken wie einer internetbasierten Austauschplattform soll ein Lieferant die Möglichkeit erhalten, seine zu meldenden Qualitätsdaten selbst über eine Erfassungsmaske an die Flender GmbH zurückzumelden. Der Aufwand für die Stammdatenpflege soll dabei möglichst gering gehalten werden. Die über die Lieferantenplattform erfassten Daten sollen nach einer automatisierten Prüfung der Daten auf Plausibilität in die neue Qualitätsdatenbank gespeichert werden.

Auf den folgenden Seiten dieser Arbeit wird das Projektumfeld genauer beschrieben, gefolgt von der Schnittstellenanalyse. Danach wird genauer auf den Applikationsaufbau und der abschließenden Umsetzung der Schnittstelle eingegangen.

1.1 Unternehmen

Das Unternehmen Flender GmbH stellt Getriebe und Kupplungen für die Antriebstechnik her und ist eine hundertprozentige Tochtergesellschaft der Siemens AG. Neben den Branchen Papier, Zement, Mining und Oil, werden auch Getriebe für erneuerbare Energiegewinnung produziert. Die Flender GmbH liefert und produziert weltweit. Der Hauptsitz des Unternehmens liegt in Deutschland, in der Stadt Bocholt; weitere Produktionsstandorte befinden sich in Voerde, Penig, Graffenstaden (Frankreich), Elgin (USA), Chennai, Kharapur (Indien) und Tianjin (China). An der Umsetzung des in diesem Dokument beschriebenen übergeordneten Projekts sind die IT Abteilung und die Qualitätsabteilung, sowie Fachabteilungen als Kunden beteiligt.

1.1.1 Die IT Abteilung

Die IT Abteilung unterstützt die operativen Fachbereiche bei der Geschäftsprozessoptimierung für SCM, PLM und CRM. Die Abteilung berät den internen Kunden bei der Umsetzung komplexer Anforderungen hinsichtlich Auswahl und Implementierung geeigneter IT-Tools und unterstützt bei der Betreuung dieser Systeme. Die Abteilung besteht aus 80 Mitarbeitern weltweit. Die hier beschriebene Arbeit findet im Umfeld dieser Abteilung statt.

1.1.2 Die Qualitätsabteilung

Das Qualitätsmanagement umfasst alle Tätigkeiten und Zielsetzungen zur Sicherung der Produkt- und Prozessqualität. Zu berücksichtigen sind hierbei Aspekte der Wirtschaftlichkeit, Gesetzgebung, Umwelt und Forderungen des Kunden. Zu den klassischen Aufgaben des Qualitätsmanagements zählen hierbei Qualitätsplanung, Qualitätslenkung, Qualitätsprüfung und Qualitätsverbesserung. Jede operative Geschäftseinheit hat ihr eigenes Qualitätsmanagement, die wiederum einer zentralen Qualitätsmanagementabteilung unterstellt sind.

2 Projektumfeld

Dieses Kapitel gibt einen Überblick über das Projektumfeld und setzt sich zusammen aus dem Projektmanagement und der Projektplanung. Das Projektmanagement beschreibt die beteiligten Projektparteien und wie diese organisiert sind, sowie die eigentliche Zielsetzung des Projekts gefolgt von den Interessengruppen beziehungsweise Stakeholdern.

Die Projektplanung geht genauer auf den Ablauf des Projekts und die Vorgehensweise ein, sowie den Status vor Projektbeginn. Abschließend werden das Anforderungsmanagement, das Qualitätsmanagement und das Risikomanagement im Rahmen der Planung beschrieben.

2.1 Projektmanagement

Dieses Kapitel beschreibt das Projektmanagement beginnend mit dem Projektteam und der Organisation im Projekt. Im Folgenden wird das Projektziel sowie abschließend die Stakeholder beschrieben.

2.1.1 Team

Das übergeordnete Projektteam besteht insgesamt aus zehn Mitarbeitern der Flender GmbH, zusammengesetzt aus einem Projektleiter aus der Qualitätsabteilung und einem *Scrum* Master aus der IT-Abteilung. Das Entwicklerteam besteht aus drei Mitarbeitern, wozu auch der Verfasser dieser Arbeit gehört. Die Fachabteilung wird durch fünf Mitarbeiter der Flender GmbH vertreten, die als Kunden fungieren.

2.1.2 Organisation

Zur Planung und Organisation des Projektes und der Mitarbeiter wird die vom Auftraggeber vorgegebene Projektmanagementsoftware *OpenProject* eingesetzt. Diese unterstützt auch die agile Entwicklungsmethode *Scrum* mit hilfreichen Funktionen. *Scrum* wird vorranging während der Implementierungsphase eingesetzt und ist ebenfalls vorgegeben. Das Projekt gliedert sich in die drei Phasen Schnittstellenanalyse, Applikationsaufbau und Umsetzung.

Diese drei Phasen bilden den Ausgangspunkt in *OpenProject*, wobei einer Phase mehrere Arbeitsabschnitte (Sections) zugeteilt sind. Bei Bedarf kann einer Section eine Aufgabe (Task) zugewiesen werden. Jede dieser Komponenten hat einen Zeitraum, einen Verantwortlichen, eine Beschreibung, einen Status und Beziehungen zu anderen Komponenten. Zur besseren Übersicht können die Komponenten in einem Gant-Chart dargestellt werden, wie man das beispielsweise aus Microsoft Project kennt. Zur besseren Differenzierbarkeit können Komponenten einer Version zugewiesen werden, so wird auch die agile Projektplanung abgebildet, indem ein *Backlog* und ein *Sprint* als verschiedene Versionen angelegt werden. Zusätzlich wird für die

agile Projektplanung eine spezielle Ansicht bereitgestellt, bei der *Backlogs* als Liste auf der rechten Seite angezeigt werden und Sprints als Liste auf der linken Seite. Alle diese Funktionen sorgen dafür, dass jeder im Projektteam genau weiß, wie der aktuelle Status ist, wer an welchem Arbeitspaket arbeitet und was als nächstes geplant ist.

Während der Implementierungsphase wird außerdem ein sogenanntes Versionsverwaltungssystem (VCS) eingesetzt, wobei die Wahl in diesem Projekt auf *Git* fiel, welches von Linus Torvalds, dem Entwickler des Betriebssystems Linux, entworfen und entwickelt wurde. *Git* kommt bereits in anderen Teilprojekten zum Einsatz und bietet die folgenden Eigenschaften. Der Quellcode kann leicht von anderen Entwicklern verwendet werden, wenn Entwickler aus dem Team ausscheiden oder andere Aufgaben bekommen. Es können alle Änderungen, die am Code vorgenommen wurden, zurückverfolgt werden und gegebenenfalls auf einen alten Stand zurückgesetzt werden. Sofern ein Server beim Einsatz des Verwaltungssystems verwendet wird, ist die Arbeit zentral und sicher gespeichert. Bei Problemen können andere Entwickler sich schnell einen Überblick verschaffen und unterstützen. Basierend auf diesen Gründen wurde von der Projektleitung entschieden, *Git* als Versionsverwaltungssystem einzusetzen.

Die Kommunikation im Team erfolgt im Wesentlichen über E-Mail und gemeinsame Meetings. Zusätzlich wird sich über eine im Unternehmen eingesetzte Kommunikationssoftware namens *Circuit* von dem Unternehmen Unify ausgetauscht. Mit Hilfe von *Circuit* kann man über Sprache, Video und Chat in Kontakt treten, außerdem bietet es die Möglichkeit des Screensharings und der Dateifreigabe. Vereinzelt wird auch die Kommentarfunktion der Projektmanagementsoftware *OpenProject* eingesetzt. Zusätzlich wurden jeden Dienstag und Freitag Statusmeetings mit dem Betreuer dieser Arbeit abgehalten, um eventuell auftretenden Problemen entgegenzuwirken und den allgemeinen Status der Arbeit zu kommunizieren.

Die zu entwickelnde Applikation entsteht in einer Servoyumgebung. Diese wurde vom Auftraggeber vorgegeben. Servoy ist eine plattformunabhängige auf dem *Rapid Application Development (RAD) Konzept* basierende Umgebung, die komplett in *Java* entwickelt wurde und aus drei Teilen besteht: Dem Server, dem Client und der Developer-Applikation. Servoy kann

mit jeder Datenbank verwendet werden, die eine *JDBC*-Schnittstelle zur Verfügung stellt. Außerdem bietet *Servoy* verschlüsselte Datenübertragung, installationslose Client-Anwendungen (*Java-Web-Start Technologie*), Internationalisierung, Rechtemanagement und Versionierung.

2.1.3 Zielsetzung

Ziel des Projekts ist die Entwicklung einer neuen Internetschnittstelle zur Erfassung unterschiedlicher Qualitätsprüfungen, die von externen Lieferanten durchgeführt werden, am Beispiel von Maßprotokollen. (Diese Protokolle dienen zum Vergleich von Soll-, Istwerten.)

Über die browserbasierte Schnittstelle soll ein Lieferant die Möglichkeit erhalten, seine zu meldenden Qualitätsdaten selbst über eine Erfassungsmaske an die Flender GmbH zurückzumelden. Die so erfassten Daten können nach einer automatisierten Prüfung auf Plausibilität und Einhaltung der Toleranzen in die neue Qualitätsdatenbank gespeichert werden. Außerdem soll der Lieferant die Möglichkeit erhalten, Qualitätszeugnisse im PDF Format nach erfolgreicher Datenübertragung zu generieren. Zum Ende der Bachelorarbeit soll ein Prototyp entwickelt worden sein mit der Möglichkeit, Maßprotokollprüfungsdaten manuell einzugeben und an die Flender GmbH zu übertragen, sodass anschließend die Implementierung mit User Acceptance Test und Rollout durchgeführt werden kann.

2.1.4 Stakeholder

- **Qualitätsabteilung:** Die Qualitätsabteilung ist nicht mehr gezwungen das alte System zu pflegen und kann mit dem neuen System zusätzlich Auswertungen erstellen. Außerdem sind alle qualitätsrelevanten Daten an einem zentralen Ort gespeichert.
- **IT-Abteilung:** Die IT-Abteilung unterstützt die Qualitätsabteilung bei der Einführung des neuen Systems und entwickelt die Internetschnittstelle.

- **Lieferanten:** Lieferanten können an einem zentralen Ort Ihre Qualitätsdaten an die Flender GmbH weitergeben und sehen umgehend, welchen Anforderungen Ihre Daten entsprechen müssen und ob diese erfüllt sind.
- **Fachabteilungen:** Die Fachabteilungen müssen sich nicht mehr über verschiedene Kommunikationswege mit dem Lieferanten austauschen, da es nur noch einen zentralen Kommunikationsweg gibt und alle Daten an einem Ort gespeichert werden.

2.2 Projektplanung

Im Folgenden wird die Projektplanung genauer erläutert, beginnend mit dem zeitlichen Ablauf, gefolgt von der Vorgehensweise. Weil bereits eine Internetschnittstelle existiert, die abgelöst werden soll, wird danach auf den Status vor Beginn des Projekts eingegangen. Abschließend folgt eine detailliertere Beschreibung des Qualitätsmanagements und des Risikomanagements.

2.2.1 Ablauf

Der geplante Ablauf des Projekts kann der folgenden Grafik entnommen werden. Die roten Rauten stehen für Meilensteine und die blauen Balken für die Projektphasen. Die einzelnen Phasen werden im Verlauf dieses Abschnitts genauer erläutert.

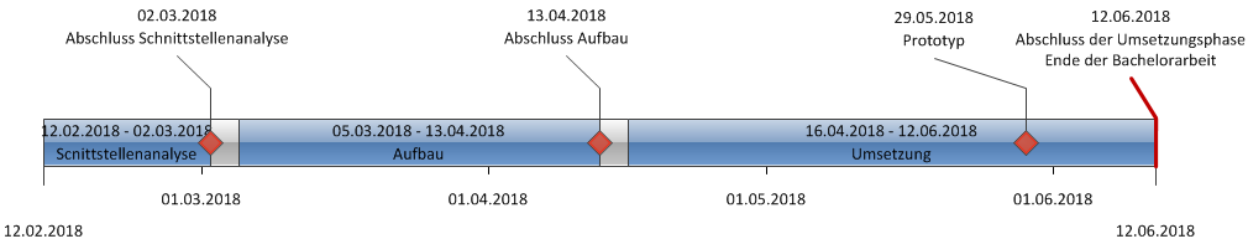


Abbildung 1: Zeitlicher Ablauf

Während der Schnittstellenanalyse werden die Anforderungen des Kunden bzw. der Fachabteilungen aufgenommen. Mehrere Meetings mit den beteiligten Parteien dienen in dieser Phase dazu, den derzeitigen Ablauf darzulegen. Die dabei auftretenden Probleme, sowie

Anforderungen für die Zukunft und die jetzige Funktionalität der Schnittstelle werden festgehalten. Zu Beginn des Projekts wurde außerdem in einem dreitätigen Workshop die bestehende Servoyumgebung, auf der die neue Lösung basieren soll, erklärt und ein Überblick über die laufenden Projekte gegeben. Die Durchführung erster Abstimmungen hatte zum Ziel, die Rahmenbedingungen des Teilprojekts, die generellen Programmiervorgaben und das zu realisierende Datenmodell festzulegen.

In der Applikationsaufbauphase wird ein Datenmodell entwickelt, das applikationsübergreifend eingesetzt werden soll, wobei auch die Anforderungen für die Internetschnittstelle mit eingebracht werden. Das heißt, die neue Qualitätsdatenbank[1] und die neue Lieferantenplattform werden auf demselben Datenmodell arbeiten. Aus diesem Grund werden häufige Abstimmungen mit den Entwicklern stattfinden, um ein Modell zu entwickeln, das allen Anforderungen gerecht wird. Das Ergebnis dieser Arbeit wird in einem *ERM*-Modell festgehalten. Außerdem werden Mockups für die Applikation erstellt, die von regelmäßigen Abstimmungen sowohl mit den Fachabteilungen als auch mit den Kunden begleitet werden. Forschungsaspekte ergeben sich aus der Servoyumgebung, die für die neue Schnittstelle eingesetzt werden soll. Dazu gehört die Beantwortung der Fragen, auf welcher Plattform die Applikation bereitgestellt wird, welche Infrastruktur benötigt wird und wie die Betreuung gewährleistet wird.

Während der Umsetzungsphase werden die zuvor entwickelten Mockups, nach erfolgreicher Abstimmung mit dem Kunden, in der Servoyumgebung umgesetzt. Darüber hinaus erfolgt die Implementierung der entsprechenden Business Logic inklusive verschiedener Tests für die jeweiligen Funktionen. Außerdem wird das Datenbankmodell in eine Testdatenbank überführt, um die Funktionalität der Software zu testen und eventuelle Anpassungen am Datenmodell vorzunehmen. Kurz vor dem Ende dieser Phase soll der Prototyp fertiggestellt sein, um die Software dem Kunden sowie der Fachabteilung vorzuführen.

[1] 4.1 Systemübersicht S.18

2.2.2 Vorgehensweise

Die Vorgehensweise in den ersten beiden Projektphasen kann man am ehesten mit dem Wasserfallmodell erklären. Diese wurde nicht explizit vorher gewählt, sondern ergibt sich aus dem Umstand, dass die Entwicklungsmethode *Scrum* die Vorbereitung eines Projekts nicht berücksichtigt. Die beiden ersten Phasen haben einen definierten Start- und Endzeitpunkt mit Endergebnissen und bauen aufeinander auf. Außerdem enthält jede Phase untergeordnete Aktivitäten, die zum Abschluss der jeweiligen Phase ebenfalls abgeschlossen sein müssen. Auf die Erstellung eines Lastenheftes und Pflichtenheftes wurde jedoch verzichtet, da diese durch das *Product Backlog* der *Scrum* Entwicklungsmethode ersetzt werden. Wie diese Vorgehensweise mit Hilfe einer Projektmanagementsoftware genau aufgebaut ist, kann dem Organisationsabschnitt aus dem vorherigen Unterkapitel entnommen werden (siehe Kapitel 2.1.2. Organisation)

Bei der Umsetzungsphase wird ein agiler Entwicklungsansatz mit Hilfe der *Scrum*-Methode verwendet. Dazu werden die Anforderungen in *User-Stories* übersetzt und in einem *Product Backlog* zusammengefasst, mit *Storypoints* bewertet und priorisiert, wobei sich die Priorität im Verlauf der Implementierung ändern kann. Die eigentliche Umsetzung erfolgt in vier *Sprints*, die jeweils vierzehn Tage dauern. Vor jedem dieser *Sprints* wird ein sogenanntes *Sprint Planning Meeting* abgehalten, in dem *Product Backlog User-Stories* entnommen werden, die dann dem kommenden *Sprint* zugeordnet werden. Durch die Bewertung der *User-Stories* kann nach einer Eingewöhnungszeit besser entschieden werden, wie viele Stories in einem *Sprint* abgeschlossen werden können. Nach dem Start eines *Sprints* können die enthaltenen *User-Stories* nicht mehr geändert werden und es werden zweimal wöchentlich kurze Meetings absolviert, um den aktuellen Status zu kommunizieren. Am Ende eines Sprints wird ein *Sprint Review Meeting* abgehalten, in dem die Ergebnisse dem *Product Owner* und gegebenenfalls anderen Interessenten präsentiert werden. Das Feedback und neue Anforderungen fließen dann in das nächste *Sprint Planning Meeting* mit ein und der Prozess beginnt von vorne.

2.2.3 Status vor Beginn der Arbeit

Zum Zeitpunkt der Arbeit wird bereits eine Internetschnittstelle auf Basis von *SAP NetWeaver* zur Rückmeldung von Qualitätsdaten (Maßprotokolle) externer Lieferanten eingesetzt. Nach erfolgreicher Anmeldung kann der Lieferant zu seinen laufenden Bestellungen Qualitätsdaten in Form von CSV-Dateien hochladen. Anschließend erfolgt eine Plausibilitätsprüfung der Daten und der Hinweis auf eventuell fehlerhafte Eingaben. Nach erfolgreicher Prüfung werden die Daten in ein ERP System überführt, gespeichert und dem Anwender als Qualitätszeugnis in Form einer herunterladbaren PDF-Datei zur Verfügung gestellt. Die aktuelle Technologie der Schnittstelle wird in Zukunft nicht mehr unterstützt und soll deshalb im September 2018 abgelöst und mit einer neuen Lösung ersetzt werden.

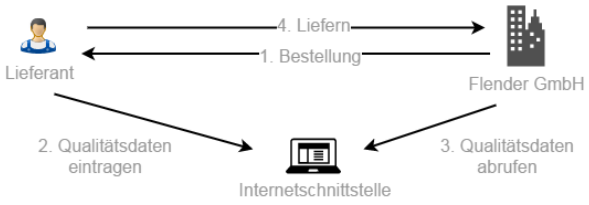


Abbildung 2: Ablauf Qualitätsdatenerfassung

2.2.4 Qualitätsmanagement

Um die Projektqualität und die Produktqualität eines Projekts zu gewährleisten, werden zu Beginn Qualitätsrichtlinien definiert und schriftlich festgehalten. Diese Qualitätsrichtlinien geben vor, nach welchen Qualitätsstandards Ergebnisse entstehen müssen und wie diese zu erreichen sind. Im Folgenden stehen die Qualitätsrichtlinien, die in Kooperation mit dem Auftraggeber entstanden sind, für das in diesem Dokument beschriebene Projekt.

Dokumente:

- Alle projektrelevanten Dokumente sind mit dem Office-Paket Microsoft Office 2010 verfasst, das Standard im Unternehmen ist, oder liegen im PDF-Format vor.

- Alle Dokumente sind mindestens einmal von einer anderen Person zu prüfen und zwar entweder innerhalb oder außerhalb des Teams.
- Eine verbesserte Version eines Dokuments wird nochmals überprüft, um sicherzustellen, dass das Feedback verarbeitet wurde.
- Wenn ein Dokument eine Prüfung nicht besteht, wird es weiter verbessert.
- Um diese Qualitätsstandards zu gewährleisten, ist jedem Dokument eine Dokumentenhistorie beigefügt, die folgendermaßen aufgebaut ist:

Version	Status	Datum	Verantwortlicher	Änderungsgrund
-	-	-	-	-

Abbildung 3: Dokumentenhistorie

Programmcode:

- Programmcode, der für ein neues Feature, einen Fix oder eine andere Änderung geschrieben wird, wird in einem neuen Branch entwickelt.
- Der Programmcode muss getestet werden, andernfalls werden Änderungen nicht in den Masterbranch übernommen.
- Der gesamte Programmcode ist zu dokumentieren, entweder durch Programmkommentare oder durch eine schriftliche Dokumentation.
- Beim Programmcode wird der von Servoy zur Verfügung gestellte Coding StyleGuide verwendet (siehe Anhang F Servoy Coding StyleGuide). Auf Codetests wird verzichtet, weil es zum Zeitpunkt der Arbeit keine Tests für die Servoyumgebung gibt.
- Zum Zeitpunkt der Arbeit konnte das Unternehmen keine Ressourcen für Continuous Integration und Continuous Delivery bereitstellen, weshalb im Rahmen dieser Arbeit darauf verzichtet wird.

2.2.5 Risikomanagement

Alle Projekte unterliegen gewissen Risiken, weshalb versucht wird, im Rahmen der Projektplanung mögliche Risiken zu erkennen und Maßnahmen dagegen zu definieren. Beim Risikomanagement erstellt man eine Übersicht über eventuell auftretende Risiken, indem man sie zur besseren Einschätzung in ihre Bestandteile zerlegt. Ein Bestandteil von einem solchen Risiko ist die Eintrittswahrscheinlichkeit. Hinzu kommt die Tragweite, die beschreibt, wie stark sich das Risiko auf den Projekterfolg auswirkt. Je höher diese beiden Faktoren sind, desto größer ist das Risiko. Zum besseren Verständnis werden das Risiko und sein Effekt auf das Projekt meist genauer beschrieben und um Maßnahmen zur Vermeidung ergänzt. In der folgenden Risikoübersicht wird die Eintrittswahrscheinlichkeit prozentual dargestellt und die Tragweite mit einer Skala von eins bis zehn, wobei eins für so gut wie keine Auswirkung steht und zehn für eine sehr gravierende.

RisikoID	Risiko	Effekt	Maßnahmen	Eintrittswahrscheinlichkeit	Tragweite
1	Anforderungen werden falsch verstanden	Unnötige Entwicklungsarbeit und falsche Funktionen	Anforderungen werden schriftlich festgehalten und mit dem Kunden abgestimmt. Regelmäßige Vorstellung der Zwischenergebnisse	10%	6
2	Zu wenig Verständnis bei der Servoyprogrammierung	Deadlines werden nicht rechtzeitig eingehalten, die Applikation wird nicht fertig	Rechtzeitig Probleme kommunizieren und gegebenenfalls Hilfe von Servoy anfordern bzw. intensive Schulungen vor Implementierungsbeginn	40%	6
3	Die Stagingumgebung steht nicht rechtzeitig zur Verfügung	Die Applikation kann nicht deployed werden. Verschiebung Produktivsetzung	Frühzeitiges Beantragen der nötigen Ressourcen oder bestehende Ressourcen nutzen	50%	7
4	Datenbankmodell funktioniert nicht	Altdaten können nicht übernommen werden. Performanceprobleme.	Rechtzeitige Migration von Altdaten in eine Testdatenbank um gegebenenfalls Änderungen am Datenmodell vorzunehmen. Review des Datenmodells	20%	7
5	Datensynchronisation funktioniert nicht	Fehlende Daten	Regelmäßiges Testen der Synchronisationsfunktionalität	30%	9
6	Remote Function Call (RFC) zum SAP-System funktioniert nicht	Fehlende Daten	Frühzeitiges Implementieren der RFC-Schnittstelle und regelmäßige Funktionstests	30%	9

Tabelle 1: Risikoübersicht

3 Schnittstellenanalyse

In diesem Kapitel geht es um die Analyse der aktuell im Unternehmen eingesetzten Internetschnittstelle, die in Kapitel 2.2.3 bereits kurz erläutert wurde. Wie bereits erwähnt, wurden zur Analyse mehrere Meetings mit den Fachabteilungen abgehalten, um einen Einblick in die jetzige Situation und zukünftige Anforderungen zu bekommen. Der Hauptfokus der Bachelorarbeit liegt zunächst auf sogenannten Maßprotokollen. Diese Protokolle dienen zum Vergleich von Soll- und Istwerten. Allerdings wurde ebenfalls mit anderen Fachabteilungen gesprochen, die die Anforderung haben, auch andere Protokolle und Prüfungen abzubilden, wie beispielsweise Gußprotokolle. Dadurch kann die neue Applikation nach einem generischen Ansatz geplant werden, um spätere Erweiterungen zu vereinfachen. Gußprotokolle enthalten zusammengefasst die Ergebnisse von chemischen Analysen und haben einen ähnlichen Aufbau wie Maßprotokolle, weshalb es nahe liegt, diese mit abzubilden.

Als erstes werden in diesem Kapitel die Ist-Situation, die positiven Eigenschaften der aktuellen Schnittstelle, weitere Verbesserungsmöglichkeiten und „Berührungsängste“ genauer beschrieben. Es folgt die Soll-Situation, in der beschrieben wird, wie die neue Schnittstelle aussehen beziehungsweise funktionieren soll und welche Anforderungen erfüllt sein müssen. Abschließend erfolgt die Beschreibung der Lösung dieser Analysearbeit mit einem Anwendungsfalldiagramm.

3.1 Ist-Situation

Die jetzige Internetschnittstelle arbeitet Hand in Hand mit dem im Unternehmen eingesetzten ERP-System SAP. Hierbei beschränkt sich die Stammdatenübertragung allerdings auf das Produktivsystem. Zur weiteren Verarbeitung werden die Daten jedoch auch im Testsystem benötigt, um zum Beispiel den Druck von Zeugnissen zu prüfen. Diese Daten werden zurzeit händisch von einem System in das andere übertragen. Hinzu kommt die Notwendigkeit der Klassifizierung für jedes Maß aus dem Protokoll im SAP-System. Damit Tests mit den Daten gemacht werden können, müssen diese außerdem in der PLM-Software und dem Testsystem des Unternehmens verfügbar sein.

Für die Erstellung von Maßprotokollen und ihren Sollwerten, die dann in der Schnittstelle angezeigt werden und um den zuvor beschriebenen manuellen Aufwand am System zu reduzieren, wird von der Fachabteilung eine Excel-Vorlage (CSV-Datei) verwendet, die dazu dient, die Daten ins SAP System übertragen zu können.

Außerdem wurde die Schnittstelle nicht bei allen Lieferanten ausgerollt, so dass sie die Schnittstelle nicht nutzen und ihre Daten mit Hilfe von Excel oder anderen Verfahren an die Flender GmbH weitergeben, die diese Daten dann prüfen und bei Fehlern wiederum den Lieferanten kontaktieren müssen, was unter Umständen auch mehrmals passieren kann. Ein weiteres Problem sind Lieferanten, die zur Übertragung der Daten eine CSV-Datei zur Verfügung gestellt bekommen, diese dann mit Excel öffnen, Excel automatisch führende Nullen löscht und es dann zu Problemen in der weiteren Verarbeitung kommt. Zusammengefasst gibt es im Moment keinen einheitlichen Übertragungsweg für Qualitätsdaten von Lieferanten.

Die Usability und die Performance der aktuellen Applikation führen zu Einschränkungen bei der Nutzung. Die zur Verfügung gestellten Filter funktionieren oft nicht oder nur in Verbindung mit langen Antwortzeiten, die nicht nur in diesem Fall entstehen. Was wiederum für Auswertungen benötigt wird, ist die komplette Teilidentifikation (Bestellnummer, Position, laufende Nummer), die aktuell nur manuell aus den drei genannten Feldern zusammengesetzt wird.

Die Plausibilitätsprüfung nach dem erfolgreichen Übertragen von Qualitätsdaten auf Lieferantenseite funktioniert sehr gut. Es wird sofort grafisch angezeigt, ob die Maße im Toleranzbereich liegen oder nicht. Falls Daten nicht im Toleranzbereich liegen, kann man eine Meldung hinterlegen, die als Ausnahmeantrag für eine Abweichung zu verstehen ist und vorher beantragt werden muss.

3.2 Soll-Situation

Die neu zu entwickelnde Internetschnittstelle auf Basis des Softwarewerkzeugs Servoy soll in einer mehrsprachigen Weboberfläche aus dem Internet für externe Lieferanten erreichbar sein. Die Mehrsprachigkeit soll sich zunächst auf die deutsche und englische Sprache beschränken, mit der Möglichkeit, im weiteren Betrieb Sprachen zur Oberfläche hinzuzufügen.

Die Schnittstelle soll dem Anwender seinen Arbeitsvorrat anzeigen können und ihm die Möglichkeit bieten, aus diesem Vorrat einen Datensatz zu wählen, für den Messdaten übertragen werden können.

Die Messdaten, die ein Lieferant an die Flender GmbH zu übertragen hat und die dabei einzuhaltenden Vorgaben sollen in der Schnittstelle vorgegeben sein, sodass für den Lieferanten sofort ersichtlich ist, welche Daten er einzugeben hat. Während dieser Eingabe soll sofort kommuniziert werden, ob die Daten valide sind. Bei nicht validen Daten soll daraufhin eine Möglichkeit existieren, einen Ausnahmeantrag anzugeben. Außerdem soll es möglich sein, mehrere Datensätze mit Hilfe eines Uploads einer Datei einzugeben und im Anschluss das Ergebnis dieses Uploads angezeigt zu bekommen, inklusive der Prüfung auf Plausibilität. Zusätzlich soll der Anwender in der Lage sein, Anhänge zu einem Maßprotokoll hochzuladen.

Nach erfolgreicher Übertragung der Daten erhält der Lieferant die Möglichkeit, sich ein Zeugnis beziehungsweise Prüfprotokoll des aktuellen Maßprotokolls im PDF-Format herunterzuladen. Diese Zeugnisse sollen nur herunterladbar sein, wenn die Daten zuvor geprüft und für gültig befunden wurden.

3.3 Ergebnis

Das Ergebnis der Schnittstellenanalyse sind Anwendungsfälle, die aus den folgenden Eigenschaften bestehen: einem Titel, einem Akteur, einem Szenario, einem Trigger und einer Vorbedingung. Im Folgenden wird eine Übersicht aller Anwendungsfälle aufgeführt, worauf die Darstellung eines beispielhaften Anwendungsfallbeispiels folgt. Alle Anwendungsfälle können dem Anhang entnommen werden (siehe Anhang B Anwendungsfälle).

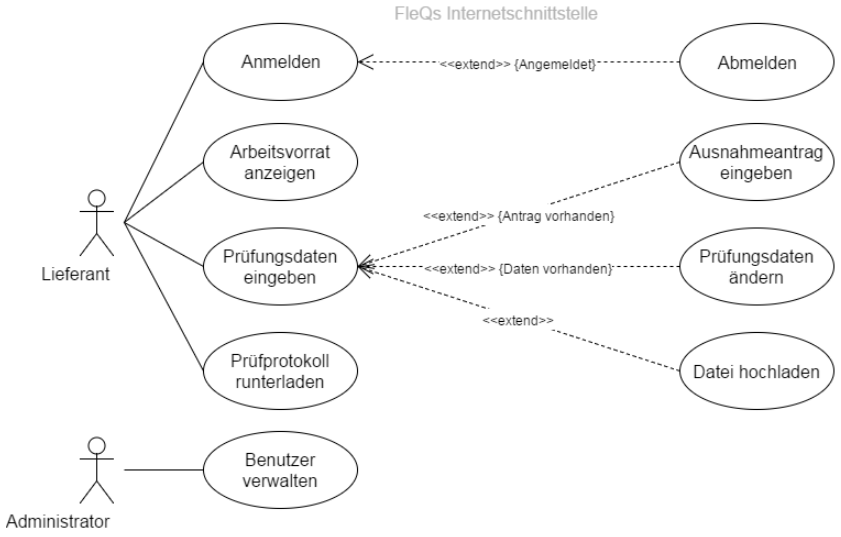


Abbildung 4: Anwendungsfalldiagramm

Name	Prüfungsdaten eingeben
Akteur	Lieferant
Szenario	1. Der Anwender gibt seine Prüfparameter ein. 2. Der Anwender gibt seine Istwerte ein. 3. Der Anwender sieht ob seine Daten valide sind. 4. Der Anwender speichert seine Eingaben
Trigger	Der Anwender hat eine Bestellung aus dem Arbeitsvorrat ausgewählt.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Prüfdatenerfassung. - Es sind offene Bestellungen vorhanden. - Es wurde eine Bestellung ausgewählt. - Es wurde ein Prüfling ausgewählt - Der Prüfling ist nicht abgeschlossen. - Die Bestellung ist nicht gesperrt.

Abbildung 5: Anwendungsfallbeispiel

4 Applikationsaufbau

Dieses Kapitel beschreibt den Aufbau der zu erstellenden Schnittstelle mit dem Namen „Flender Qualitätssystem Lieferantenplattform (FleQs Lieferantenplattform)“ und der zugehörigen Komponenten. Als erstes werden die beteiligten Systeme und deren Beziehungen durch eine grafische Übersicht genauer erläutert. Dazu wurde im Vorfeld eine Forschung betrieben, um die möglichen Plattformen zum Bereitstellen der Schnittstelle herauszuarbeiten und anhand vordefinierter Kriterien eine Empfehlung auszusprechen. Diese Forschungsarbeit wird im nachfolgenden Kapitel noch genauer erläutert.

Anschließend wird das zugrundeliegende Datenbankmodell der Applikation beschrieben. Dieses Modell wird nicht nur von der in diesem Dokument beschriebenen Schnittstelle verwendet, sondern applikationsübergreifend eingesetzt. Die Beschreibung des Datenmodells in diesem Dokument bezieht sich dabei ausschließlich auf die für die FleQs Lieferantenplattform relevanten Aspekte und wurde mit Hilfe von Microsoft Visio 2010 erstellt.

Abschließend wird das Design anhand weiterer Mockups beschrieben und genauer erklärt. Bei der Erstellung wurde die bestehende Schnittstelle als Ausgangspunkt genommen. Nach dem Entwurf einer Version wurde diese diskutiert und das Feedback floss in die nächste Version mit ein, bis man bei einem für alle Parteien zufriedenstellenden Ergebnis angelangt war.

4.1 Systemübersicht

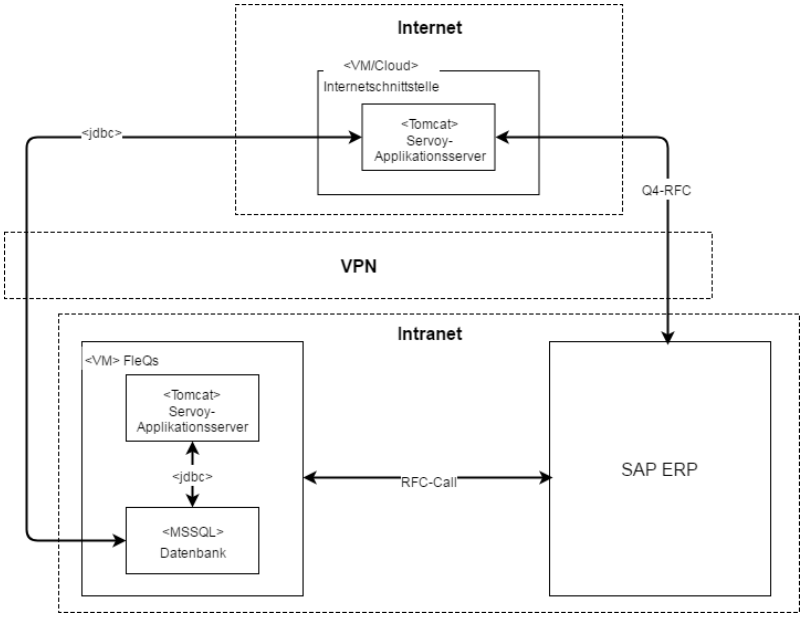


Abbildung 6: Highfly Systemübersicht

An der Systemlandschaft der Internetschnittstelle sind drei Systeme beteiligt: die eigentliche Internetschnittstelle, das Flender Qualitätssystem und das SAP ERP System. Zum Zeitpunkt der Bachelorarbeit war die finale Infrastruktur noch nicht festgelegt worden, weshalb die Entwicklung auf einem Testsystem erfolgte. Die hier dargestellte Systemübersicht ist das wahrscheinlichste Szenario der Produktivumgebung.

Für die Entscheidung, auf welcher Plattform die Produktivumgebung entstehen soll, wurde eine Forschungsarbeit erstellt, die die möglichen Plattformen aufzeigt und bewertet und dabei gewisse Kriterien berücksichtigt, die zuvor in Zusammenarbeit mit dem Unternehmen aufgestellt worden sind. Die Kriterien wurden hinsichtlich folgender Kategorien ausgewählt: Betreuung, Funktion, Sicherheit, Kosten und Performance. In der Arbeit sind die einzelnen Kriterien in tabellarischer Form aufgelistet. In Bezug auf die möglichen Plattformen werden zudem einerseits ein Server als virtuelle Maschine (VM) und andererseits eine Cloudplattform als Varianten gegenübergestellt. Abschließend wird in der Forschungsarbeit eine Empfehlung für eine Plattform anhand eines Beispielszenarios dargestellt. Der Fokus dieses Szenarios liegt dabei auf der Unterteilung in

getrennte Subnetze und der damit verbundenen Zugriffskontrolle (siehe Anhang D Forschungsarbeit).

Die Schnittstelle ist im Gegensatz zu den anderen Systemen aus dem Internet erreichbar und besteht aus nur einer Komponente. Es handelt sich um einen containerbasierten Tomcat-Server, der einen Servoy-Applikationsserver bereitstellt, auf dem die Schnittstelle als sogenannte Servoy-Lösung läuft. Diese Komponente wird mit Hilfe eines Cloudanbieters zur Verfügung gestellt. Zusätzlich kommuniziert die Schnittstelle über einen VPN-Tunnel mit einem SAP ERP System, um die Ausnahmeanträge zu prüfen, die ein Anwender bei einem sich nicht im Toleranzbereich befindlichen Prüfwert hinterlegen kann.

Ein weiteres System ist das FleQs (Flender Qualitätssystem), das ebenfalls auf Servoy basiert und aus einem Tomcat-Server und einer Microsoft SQL Datenbank besteht. Dieses System läuft auf einem virtualisierten Microsoft Server 2012 System innerhalb der Flender GmbH und ist nur im Intranet des Unternehmens erreichbar. Das FleQs ist die Neuentwicklung eines Qualitätssystems und ersetzt die auf Microsoft FoxPro basierende alte Software. Mit diesem System können Prüfungen zu Bauteilen, Aufträgen und Lieferungen erfasst, dokumentiert und rückverfolgt werden. Das FleQs und die FleQs Lieferantenplattform arbeiten mit demselben Datenmodell. Die Vorgaben, die in der Lieferantenplattform angezeigt werden, werden mit dem FleQs System generiert und die Daten, die ein Lieferant an das Unternehmen übergibt, werden wiederum im FleQs System zur weiteren Verarbeitung benötigt. Beide Systeme greifen auf dieselbe Datenbank zu, wobei die Lieferantenplattform mit Hilfe eines VPN-Tunnels die Daten des Unternehmensnetzwerks einsehen kann. Damit das FleQs Zugriff auf Daten, wie Bestellungen und Aufträge hat, besitzt es außerdem eine Schnittstelle zum SAP ERP System.

Das letzte an der Systemarchitektur beteiligte System ist das bereits erwähnte SAP ERP System, welches als Datenbasis für das FleQs System fungiert und Stammdaten, sowie Bewegungsdaten des Unternehmens beinhaltet. Dieses System befindet sich auf einem Server innerhalb der Siemens AG. Außerdem werden über dieses System die Ausnahmeanträge für die Lieferantenplattform angelegt.

4.2 Datenbankmodell

Das hier gezeigte Datenmodell zeigt nur einen Ausschnitt der für diese Arbeit relevanten Tabellen. Zu beachten ist, dass das Modell Fremdschlüssel enthält, um es besser zu beschreiben. Die eigentliche Datenbank enthält keinerlei Fremdschlüssel und die Verknüpfung der Tabellen erfolgt ausschließlich im Quellcode, mit Hilfe von Servoy-Relationen (siehe Kapitel 5.1), festgelegt durch das übergeordnete Projektteam. Außerdem wurde bestimmt, die Tabellennamen klein zu schreiben. Bei der Beschreibung der Tabellen werden nur die funktionsrelevanten Attribute beschrieben und nicht die Metadaten, wie Erstellzeit, Erstellbenutzer oder Ähnliches.

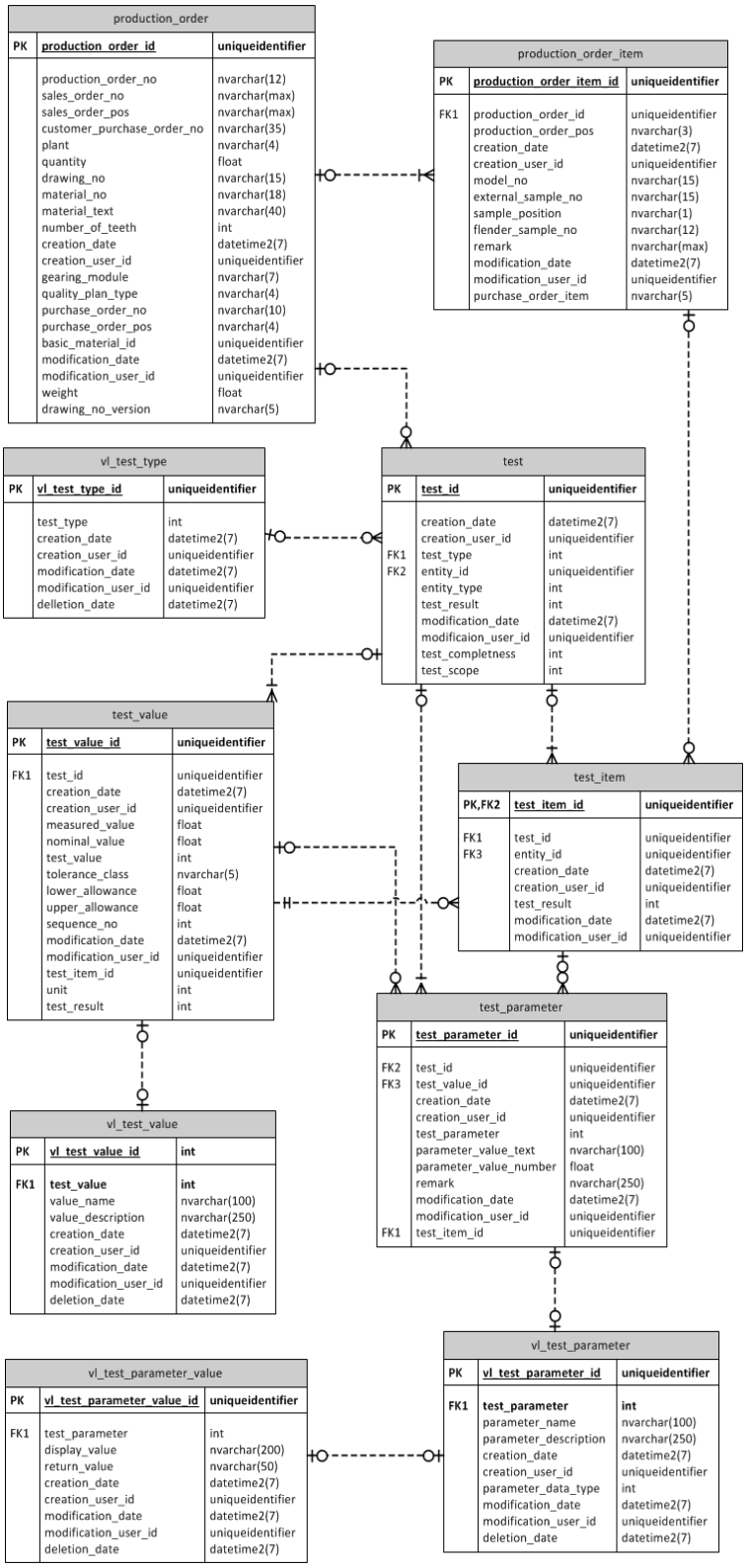


Abbildung 7: Datenbankmodell

Der Ausgangspunkt in diesem Modell ist die Testtabelle (test), mit der eine Prüfung mit ihren Metadaten beschrieben wird, wobei das Attribut entity_id besonders interessant ist. Dieses Attribut kann ein beliebiges Objekt aus der Datenbank enthalten, wie zum Beispiel einen Fertigungsauftrag, eine Bestellung oder ein Material. Im Fall von Maßprotokollen enthält sie einen Datensatz der Tabelle production_order (production_order_id) und ist vom Typ Maßprotokoll (test_typ). Wenn eine neue Bestellung generiert wird, entsteht ein Eintrag in der Fertigungsauftragstabelle (production_order) mit einer Bestellnummer, aber ohne Fertigungsauftrag. Der Auftrag wird zu einem späteren Zeitpunkt der Bestellung zugeordnet und eingetragen.

Die test_item Tabelle, welche ein Teil innerhalb eines Tests beschreibt, ist über die test_id mit einem Test verknüpft. Diese Tabelle hat wiederum eine entity_id und enthält beim Test für Maßprotokolle eine Fertigungsauftragsteilidentnummer (production_order_item_id). Die Fertigungsauftragsteiltabelle (production_order_item) enthält im Wesentlichen die Kopfdaten eines Maßprotokolltests, wie die Zeichnungsnummer (drawing_no) und die laufende Nummer (purchase_order_item). Bei der Zuordnung zu einem Fertigungsauftrag erhält sie außerdem eine Fertigungsauftragsteilenummer (production_order_pos).

Die nächste Tabelle ist die test_value Tabelle. In dieser Tabelle werden die eigentlichen Prüfwerte gespeichert. Beim Maßprotokoll besteht sie aus den folgenden Vorgaben: Sollwert (nominal_value), untere Toleranzgrenze (lower_allowance), obere Toleranzgrenze (upper allowance), Toleranzklasse (tolerance_class) und dem Istwert (measured_value). Diese Tabelle ist im Fall von Maßprotokollen mit der Testtabelle (test) und der Testbestandteiltabelle (test_item) verknüpft, kann aber im Hinblick auf andere Prüfungen auch nur mit einer von beiden Tabellen verknüpft sein.

Um die Prüfwerte genauer zu beschreiben existiert die vl_test_value Tabelle. Eine Prüfung kann zum Beispiel vom Typ OK_NOT_OK sein, wenn ein Prüfwert nur aussagt, ob etwas in Ordnung oder nicht in Ordnung ist. Im Fall von Maßprotokollen ist ein Wert in der Regel vom Typ MEASUREMENT, also Messwert.

Um Parameter zu einer Prüfung (test), einem Prüfungsbestandteil (test_item) oder einem Prüfungswert (test_value) zu speichern, existiert die Tabelle test_parameter. In dieser Tabelle steht der Wert des Parameters (parameter_value_text oder parameter_value_number) und der Parametertyp (test_parameter). Der Parametertyp ist in der Tabelle vl_test_parameter definiert. Er besteht aus einem Namen (parameter_name), zum Beispiel LADLE_SIGN für Gießzeichen, und einem Datentyp (parameter_data_type). Der Datentyp sagt aus, ob es sich um einen Parameter mit Text oder einen Parameter mit einer Zahl handelt.

4.3 Mockups

Um die grafische Oberfläche der FleQs Lieferantenplattform besser zu planen und um die Oberfläche mit dem Kunden abzusprechen, wurden mehrere Versionen von Mockups erstellt. Die Ausgangslage dieser Modelle war, wie bereits erwähnt, die bestehende Lieferantenschnittstelle. Nach der Fertigstellung einer Version wurde diese mit dem Projektteam und dem Kunden besprochen. Das Ergebnis dieser Besprechung waren Anforderungen für das nächste Modell. So sind sechs Versionen entstanden, wovon die letzte Version als Ausgangspunkt für die Implementierung festgelegt wurde. Im Rahmen dieses Dokuments wird diese Version nachfolgend anhand von Bildern erläutert. Die Version enthält drei Ansichten mit den folgenden Funktionen: eine Ansicht für den Login des Anwenders, eine für den Arbeitsvorrat und die letzte für die Prüfdatenerfassung.

4.3.1 Login

Die erste Ansicht die ein Anwender zu sehen bekommt, wenn er die Schnittstelle aufruft, ist die Login-Ansicht. In dieser Ansicht kann der Anwender die Anzeigesprache auswählen und sich mit seinen Zugangsdaten anmelden.

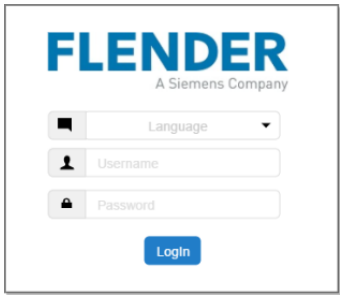
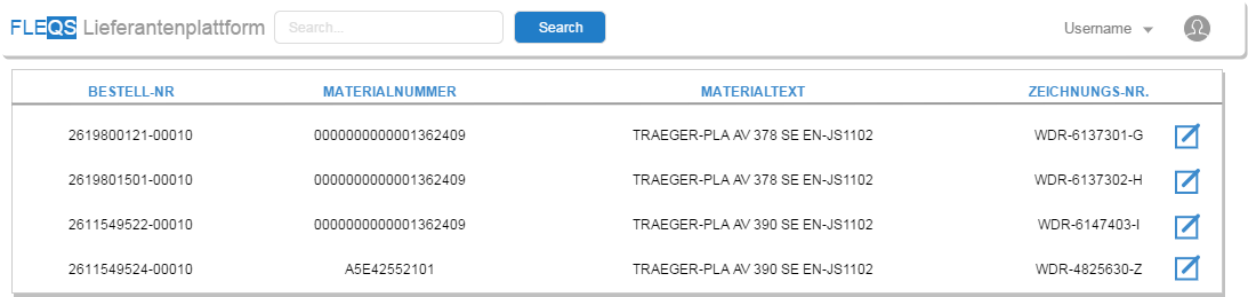


Abbildung 8: Login Mockup

4.3.2 Arbeitsvorrat

Nach erfolgreicher Anmeldung kommt der Anwender zu seinem Arbeitsvorrat. In diesem Arbeitsvorrat werden nur die aktuellen Bestellungen angezeigt, zu denen der Lieferant noch Qualitätsdaten zu übertragen hat. Nach dem Klick auf einen Eintrag landet der Anwender in der Prüfdatenerfassung.



BESTELL-NR	MATERIALNUMMER	MATERIALTEXT	ZEICHNUNGS-NR.
2619800121-00010	0000000000001362409	TRAEGER-PLA AV 378 SE EN-JS1102	WDR-6137301-G
2619801501-00010	0000000000001362409	TRAEGER-PLA AV 378 SE EN-JS1102	WDR-6137302-H
2611549522-00010	0000000000001362409	TRAEGER-PLA AV 390 SE EN-JS1102	WDR-6147403-I
2611549524-00010	A5E42552101	TRAEGER-PLA AV 390 SE EN-JS1102	WDR-4825630-Z

Abbildung 9: Arbeisvorrat Mockup

4.3.3 Prüfdatenerfassung

In der Prüfdatenerfassung wird dem Lieferanten im oberen Bereich die Möglichkeit gegeben, die Kopfdaten zu einer Maßprotokollprüfung einzugeben. Diese sind abhängig von der unten links auswählbaren laufenden Nummer. Zusätzlich kann eine neue laufende Nummer mit Hilfe des Buttons unter den laufenden Nummern angelegt werden. Auf der unteren rechten Seite können die eigentlichen Prüfwerte eingetragen werden, mit anschließender Anzeige, ob der eingetragene

Wert in Ordnung ist oder nicht. Zusätzlich kann hier ein Ausnahmeantrag angegeben werden. Abschließend können im oberen Bereich die Daten gespeichert werden, es kann zum Arbeitsvorrat zurückgekehrt werden und es kann eine Datei mit den Prüfwerten hochgeladen werden. Daraufhin wird die Tabelle unten rechts automatisch mit diesen Werten gefüllt.

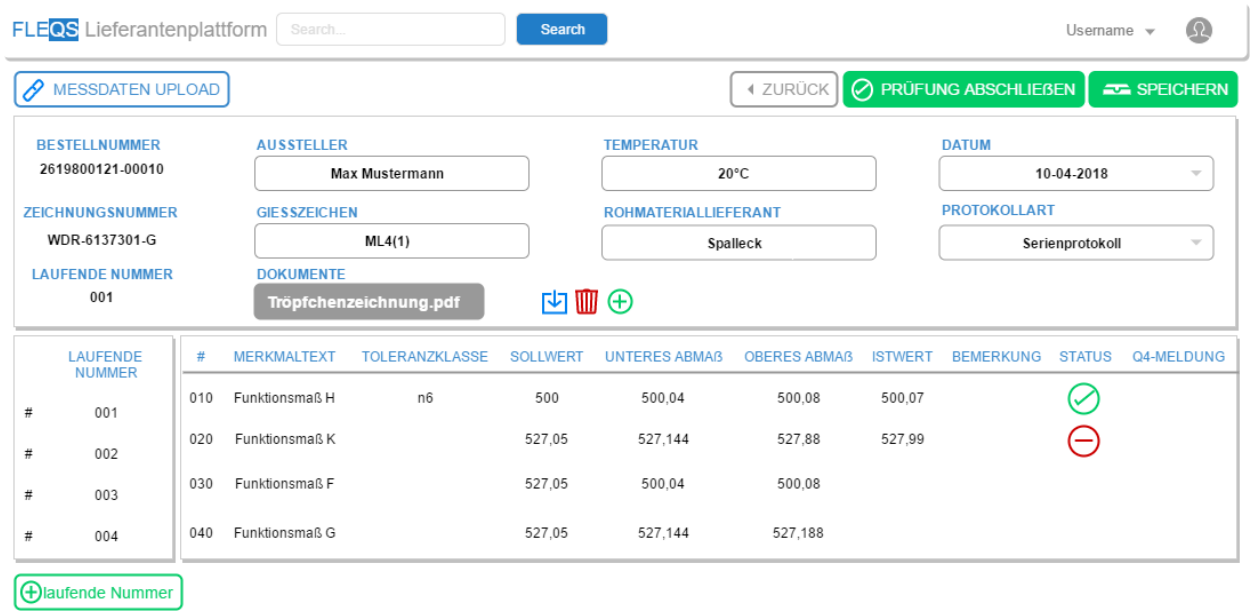


Abbildung 10: Prüfdatenerfassung Mockup

5 Umsetzung

In diesem Kapitel geht es um die Umsetzung der in dieser Arbeit beschriebenen Lieferantenplattform für Qualitätsdaten mit dem Softwaretool Servoy und dem zugehörigen Servoy Developer. Zunächst wird Servoy mit seinen wichtigsten Konzepten genauer erklärt. Danach wird die grafische Oberfläche anhand eines beispielhaften Servoy Formulars gezeigt und es wird erläutert, wie dieser Zustand erreicht wurde. Abschließend wird die Servoy Business Logic mit einem Klassendiagramm genauer beschrieben. Wie Servoyprogrammcode aussieht und wie dieser getestet wird kann einem Beispiel im Anhang entnommen werden(siehe Anhang G Servoy Programmcodebeispiel). Das nachfolgende Bild zeigt den Projektfortschritt zum Zeitpunkt der Realisierung der Prüfdatenerfassung in OpenProject, der gesamte *Product Backlog* kann dem Anhang entnommen werden (siehe Anhang C Product Backlog).

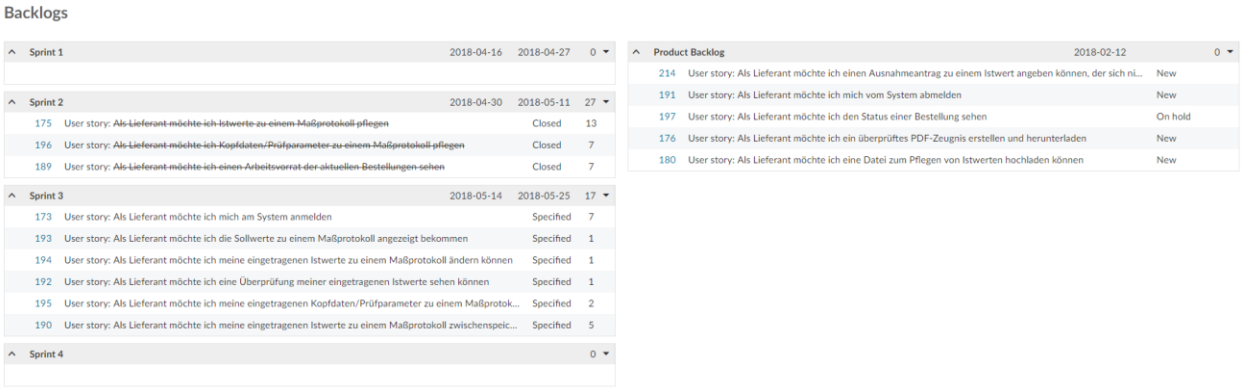


Abbildung 11: Projektfortschritt

5.1 Servoykonzepte

Solutions: Eine Solution ist eine einzelne Anwendung und enthält Formulare, Business Logic und Datenschichtendefinitionen.

Formulare: Ein Formular ist das Servoy-Objekt, das zum Bearbeiten, Anzeigen, Ändern, Einfügen, Suchen und Löschen von Daten verwendet wird und in der Regel auf einer einzigen Datenbanktabelle basiert, was aber nicht obligatorisch ist. Den Formularen können darüber hinaus nur Methoden und Variablen zugeordnet werden, die im Rahmen des Formulars liegen. Vererbte Formulare können auch Methoden und Variablen haben, die ebenfalls vom übergeordneten Formular vererbt werden.

Datasources: Eine Datasource ist der Bezeichner für eine externe Quelle von tabellarischen Daten, in den meisten Fällen eine Tabelle aus einer Datenbank. Servoy übernimmt die komplette Verwaltung der Datenbank, sodass keine Backendapplikation entwickelt werden muss.

Datenprovider: Ein Datenprovider verwaltet Informationen und ist nicht nur eine Datenbankspalte, sondern kann auch eine Berechnung, Aggregation, globale Variable oder Formularvariable sein. Datenprovider können einem Feld oder auch Beschriftungen zugeordnet werden.

Methoden: Servoy's Scripting Engine verwendet Methoden, um zusammengehörenden Code zu identifizieren. Eine Servoy-Methode ist im Wesentlichen mit einer JavaScript-Funktion gleichzusetzen. Alle Methoden in Servoy gehören zu einem Scope, im Fall von Formularen spricht man auch von einem Formscope. Darüber hinaus gibt es allgemeine Top-Level Scopes, die Methoden enthalten, die keinem Form angehören und zum Beispiel für die Erstellung von Klassen verwendet werden. Sie befinden sich direkt unterhalb einer Solution.

Plugins: Ein Plugin erweitert die Funktionalität von Servoy um die Programmiersprache Java. In Java geschrieben, können sie in einer Lösung durch Scripting angesprochen werden. Plugins können dem Servoy-Programmierer zusätzliche Funktionen hinzufügen und auf Ressourcen außerhalb von Servoy verweisen.

Foundsets: Ein Foundset ist ein Servoy-Objekt, das eine Abfrage aus einer einzelnen Datenbanktabelle repräsentiert. Foundsets übernehmen das Caching von Daten für die Anzeige, müssen dafür aber nicht die Datenbank anfordern. Jedes Formular, das auf einer Tabelle in Servoy basiert, hat ein zugrundeliegendes Foundset, das die Daten dafür verwaltet. Formulare, die auf derselben Tabelle basieren, teilen sich ein Foundset. Foundsets können durch Scripting erstellt, aufgerufen, geändert, kontrolliert und angewendet werden.

Relationen: Eine Relation ist ein Servoy-Objekt, mit dem die Schlüsselbeziehung zwischen zwei Datenbanktabellen definiert wird.

Calculations: Eine Calculation ist ein abgeleiteter Wert, der sich wie eine weitere Spalte in der Datenbanktabelle verhält. Die Berechnungsdaten sind das Ergebnis einer JavaScript-Funktion.

5.2 Grafische Oberfläche

Das nachfolgende Bild zeigt die grafische Oberfläche des Servoy-Formulars für die Prüfdatenerfassung (DimensionalProtocolTest). Der Servoy Developer verwendet für die Erstellung von Formularen einen WYSIWYG Form-Designer. Mit diesem Designer können verschiedene Container auf dem Formular platziert werden, wie zum Beispiel ein div-Container. Durch das Hinzufügen von CSS-Klassen können den Containern Layout-Funktionen gegeben werden, die außerdem beliebig verschachtelt werden können. Bei dem folgenden Formular wurde mit CSS-Grid gearbeitet. Komponenten werden hinzugefügt, indem sie im Designer in einen Container gezogen werden. Komponenten können beispielsweise Buttons, Dropdown-Felder oder Tabellen sein und haben bestimmte Eigenschaften wie Größe, Anzeigetext oder Ausrichtung, die direkt im Designer bearbeitet werden können. Genauso wie den Containern können Komponenten CSS-Klassen hinzugefügt werden. Des Weiteren bietet Servoy die Möglichkeit, Formulare voneinander erben zu lassen, so erbt das hier zu sehende Formular beispielsweise die Navbar seines Elternformulars.

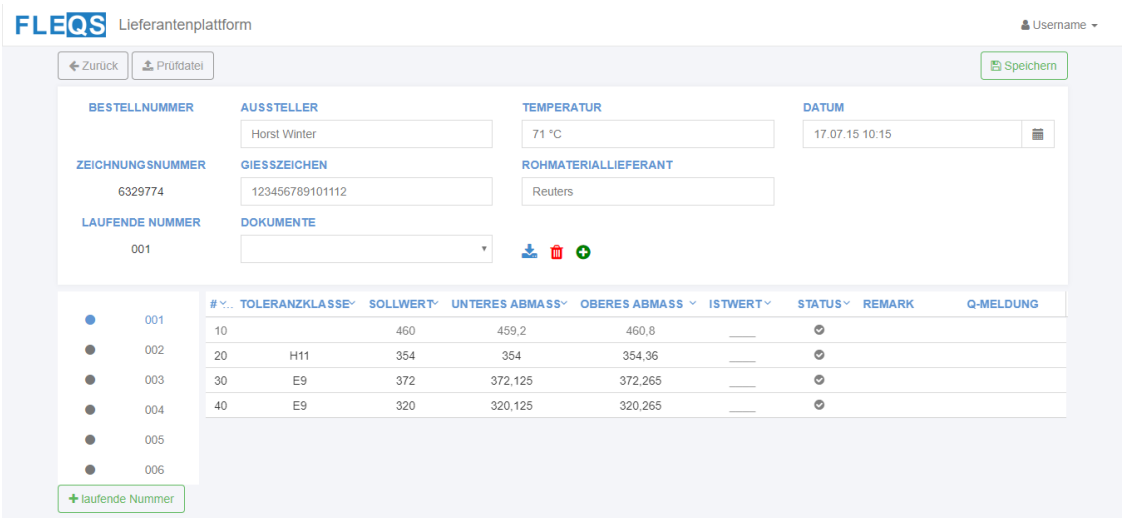


Abbildung 12: Grafische Oberfläche Prüfdatenerfassung

5.3 Business Logic

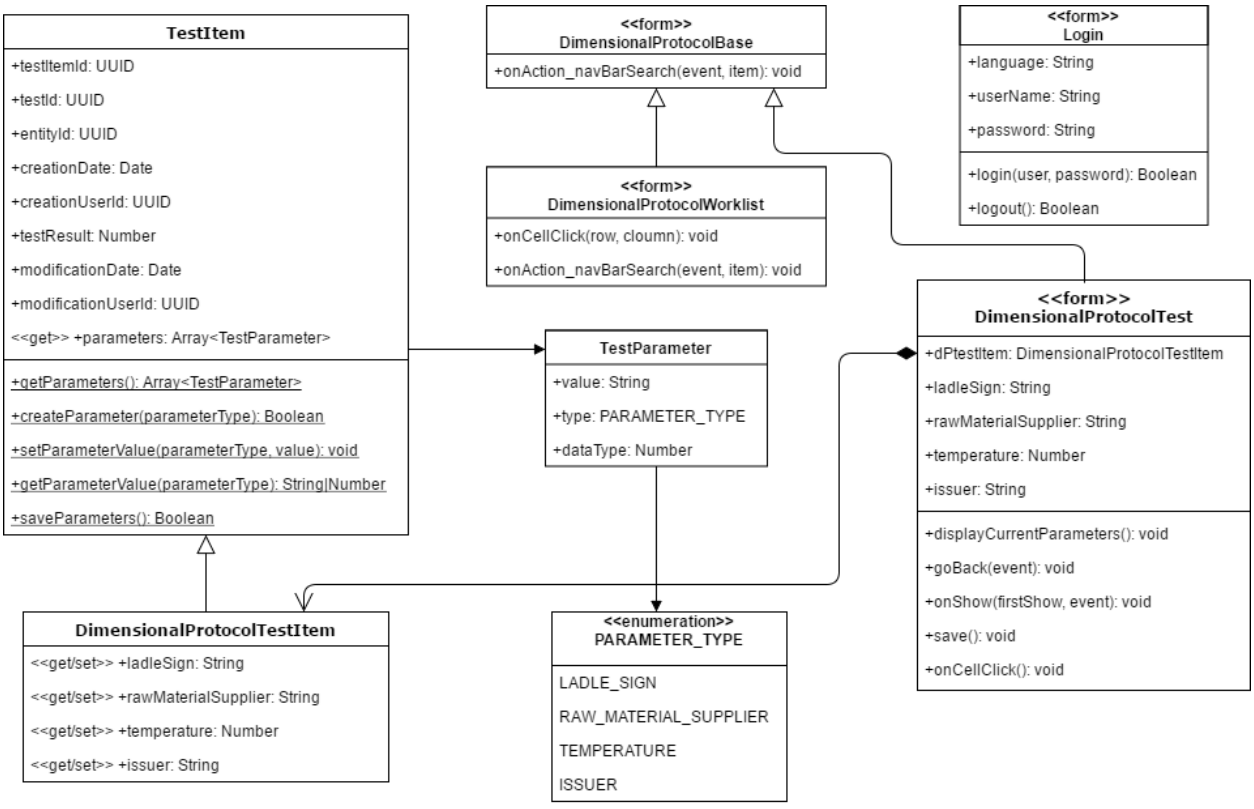


Abbildung 13: Klassendiagramm FleQs Lieferantenportal

Die hier zu sehende Abbildung zeigt das FleQs Lieferantenportal anhand seines Klassendiagramms. Servoy Formulare können, wie bereits erwähnt, ihre eigenen Methoden und Variablen haben und sind in diesem Diagramm mit <<form>> gekennzeichnet. Der Ausgangspunkt ist die Klasse TestItem, die einen Teil eines Tests beschreibt und zusätzlich ein Array mit den zugehörigen Prüfparametern enthält. Diesem Array ist eine Getter-Methode zugeordnet, die beim Aufruf alle aktuellen Parameter zu einem Testbestandteil aus der Datenbank abruft. Die Kindsklasse DimensionalProtocolTestItem besitzt zusätzlich Felder für Parameter einer Maßprotokollprüfung. Diese Felder besitzen ebenfalls eine Getter-Methode. Diese Methode ruft die Elternmethode zum Abrufen eines Prüfwerts auf und gibt den Wert des jeweiligen Parameters aus dem Parameterarray der Elternklasse zurück. Zusätzlich besitzen die Parameterfelder Setter-Methoden, die wiederum eine Elternmethode zum Setzen eines Prüfparameters im Parameterarray der Elternklasse aufrufen. Parameteränderungen werden nur temporär gespeichert, um Datenbankzugriffe zu reduzieren, weshalb es eine Methode in der Elternklasse gibt, um alle temporären Parameteränderungen in der Datenbank zu speichern.

Für Parameter gibt es eine eigenständige Klasse namens TestParameter, die zum besseren Verständnis ein Enum für Prüfparametertypen verwendet und beim Array der Elternklasse TestItem zum Einsatz kommt. Wie in der Abbildung zu sehen, erben die beiden Formklassen DimensionalProtocolWorklist und DimensionalProtocolTest von dem DimensionalProtocolBase Form, um das gleiche Layout und die gleiche Navbar zu verwenden, wobei die Suchfeld-Methode der Navbar von beiden Formularen überschrieben wird. Beide Formulare besitzen außerdem eine Methode, um auf Klicks in einer Tabelle zu reagieren. Das DimensionalProtocolTest Formular ist das Formular für die Prüfdatenerfassung. Es verwendet Variablen zum Anzeigen der Prüfparameterwerte und hat Funktionen, die wiederum geerbte Funktionen der Klasse DimensionalProtocolTestItem verwenden, um diese Werte zu aktualisieren beziehungsweise anzuzeigen und zu speichern, sowie eine Methode, um zum vorherigen Formular zurückzukehren. Das letzte Formular der Abbildung zeigt das Login Formular mit zwei Funktionen zum Ein- oder Ausloggen.

6 Fazit und Ausblick

Im Rahmen dieser Arbeit wurde eine internetbasierte Lieferantenplattform für Qualitätsdaten entwickelt. Aus diesem Grund wurde die bestehende Plattform analysiert, eine Infrastruktur ausgearbeitet, Entwürfe für eine grafische Oberfläche erstellt und ein Prototyp der Plattform entwickelt.

Das Ergebnis ist ein Prototyp zum Übertragen und Speichern von Qualitätsdaten, bei dem die wesentlichen Funktionen der neuen Lieferantenplattform umgesetzt sind und positive Reaktionen beim Kunden auslösen.

Das Arbeiten mit dem Softwarewerkzeug Servoy gestaltete sich anspruchsvoller als zunächst angenommen, auch wenn Servoy einem einige Dinge erleichtert, Stichwort Backendapplikation. Es erfordert ein hohes Verständnis der Servoykonzepte und Funktionen, um diese sinnvoll einzusetzen. Außerdem wird eine ältere JavaScript Version verwendet, sodass neuere Sprachfunktionen nicht genutzt werden können.

In der Finalisierungsphase, die nach der Bachelorarbeit stattfindet, wird die Lieferantenplattform fertiggestellt. Danach werden im Rahmen einer dedizierten Testphase Funktionstests und Anwendertests durchgeführt und in diesem Zusammenhang auch User-Acceptance Tests. Wenn diese Tests erfolgreich abgeschlossen sind, wird der technische Go-Live vollzogen, Stammdaten werden übernommen und abschließend werden die Lieferanten geschult.

7 Literaturverzeichnis

Wikipedia (2018): Servoy, [online] https://de.wikipedia.org/wiki/Servoy [09.03.2018]

Jacobsen, Jens (2018): User Stories – weniger Schreiben, mehr Gestalten, [online] https://www.usabilityblog.de/user-stories-weniger-schreiben-mehr-gestalten/ [15.03.2018].

Wikipedia (2018): Qualitätsmanagement im Projektmanagement, [online] https://de.wikipedia.org/wiki/Qualit%C3%A4tsmanagement_im_Projektmanagement [17.03.2018].

Projektmanagement Handbuch (2018): Risikomanagement, [online] https://www.projektmanagementhandbuch.de/handbuch/projektplanung/risikomanagement/ [17.03.2018].

Scrum Akademie (2018): User Stories, Epics & Themes, [online] https://www.scrumakademie.de/product-owner/wissen/user-stories-epics-themes/ [25.04.2018].

Servoy Wiki (2018): Intro to Servoy Concepts, [online] https://wiki.servoy.com/display/DOCS/Intro+to+Servoy+Concepts [21.05.2018].

Anhang

A Projektplan

Projektplan

Nils Nieuwenhuis (3180646)

Fontys Hochschule für Technik und Logistik

Wirtschaftsinformatik

Bocholt, 13. Februar 2018

Inhaltsverzeichnis

1 Einleitung

Dieser Projektplan beschreibt den Verlauf einer Bachelorarbeit, welche Teil eines Studiums im Fach „Business Informatics“ an der Fontys Hochschule für Technik und Logistik in Venlo ist. Die Bachelorarbeit findet statt bei dem Unternehmen Flender GmbH in Bocholt.

Dieses Dokument beginnt mit der Beschreibung des Unternehmens, dem Team in dem gearbeitet wird und der Organisation des Projektes. Danach wird genauer auf das Projekt eingegangen, welches Bestandteil der Bachelorarbeit ist.

1.1 Unternehmen

Das mittelständische Unternehmen Flender GmbH mit über 5.000 Mitarbeitern ist eine hundertprozentige Siemens AG-Tochter und stellt Getriebe und Kupplungen für die verschiedensten industriellen Anwendungen (Wind, Bahn, Marine, Zement, Mining u.v.m.) her. Die IT Abteilung unterstützt die operativen Fachbereiche bei der Geschäftsprozessoptimierung für SCM, PLM und CRM und berät den internen Kunden bei der Umsetzung komplexer Anforderungen hinsichtlich Auswahl und Implementierung geeigneter IT-Tools. Die Abteilung besteht aus 80 Mitarbeitern weltweit.

2 Projektmanagement

Das Projekt ist ein Teilprojekt des neuen Flender Qualitätssystems (FleQs) und beinhaltet eine Lieferantenplattform zur Erfassung von Qualitätsdaten externer Lieferanten.

2.1 Team

Das übergeordnete Projektteam besteht insgesamt aus zehn Mitarbeitern der Flender GmbH, zusammengesetzt aus einem Projektleiter aus der Qualitätsabteilung und einem *Scrum* Master aus der IT-Abteilung. Das Entwicklerteam besteht aus drei Mitarbeitern, wozu auch der Verfasser dieses Plans gehört. Die Fachabteilung wird durch fünf Mitarbeiter der Flender GmbH vertreten, die als Kunden fungieren.

2.2 Organisation

Zur Planung und Organisation des Projektes und der Mitarbeiter wird die vom Auftraggeber vorgegebene Projektmanagementsoftware *OpenProject* eingesetzt. Diese unterstützt auch die agile Entwicklungsmethode *Scrum* mit hilfreichen Funktionen. *Scrum* wird vorranging während der Implementierungsphase eingesetzt und ist ebenfalls vorgegeben. Das Projekt gliedert sich in die drei Phasen Schnittstellenanalyse, Applikationsaufbau und Umsetzung.

Diese drei Phasen bilden den Ausgangspunkt in *OpenProject*, wobei einer Phase mehrere Arbeitsabschnitte (Sections) zugeteilt sind. Bei Bedarf kann einer Section eine Aufgabe (Task) zugewiesen werden. Jede dieser Komponenten hat einen Zeitraum, einen Verantwortlichen, eine Beschreibung, einen Status und Beziehungen zu anderen Komponenten. Zur besseren Übersicht können die Komponenten in einem Gant-Chart dargestellt werden, wie man das beispielsweise aus Microsoft Project kennt. Zur besseren Differenzierbarkeit können Komponenten einer Version zugewiesen werden, so wird auch die agile Projektplanung abgebildet, indem ein *Backlog* und ein als verschiedene Versionen angelegt werden. Zusätzlich wird für die agile Projektplanung eine spezielle Ansicht bereitgestellt, bei der *Backlogs* als Liste auf der rechten Seite angezeigt werden und Sprints als Liste auf der linken Seite. Alle diese Funktionen sorgen dafür, dass jeder im Projektteam genau weiß, wie der aktuelle Status ist, wer an welchem Arbeitspaket arbeitet und was als nächstes geplant ist.

Während der Implementierungsphase wird außerdem ein sogenanntes Versionsverwaltungssystem (VCS) eingesetzt, wobei die Wahl in diesem Projekt auf *Git* fiel, welches von Linus Torvalds, dem Entwickler des Betriebssystems Linux, entworfen und entwickelt wurde. *Git* kommt bereits in anderen Teilprojekten zum Einsatz und bietet die folgenden Eigenschaften. Der Quellcode kann leicht von anderen Entwicklern verwendet werden, wenn Entwickler aus dem Team ausscheiden oder andere Aufgaben bekommen. Es können alle Änderungen, die am Code vorgenommen wurden, zurückverfolgt werden und gegebenenfalls auf einen alten Stand zurückgesetzt werden. Sofern ein Server beim Einsatz des Verwaltungssystems verwendet wird, ist die Arbeit zentral und sicher gespeichert. Bei Problemen können andere Entwickler sich schnell einen Überblick verschaffen und unterstützen. Basierend auf diesen Gründen wurde von der Projektleitung entschieden, *Git* als Versionsverwaltungssystem einzusetzen.

Die Kommunikation im Team erfolgt im Wesentlichen über E-Mail und gemeinsame Meetings. Zusätzlich wird sich über eine im Unternehmen eingesetzte Kommunikationssoftware namens *Circuit* von dem Unternehmen Unify ausgetauscht. Mit Hilfe von *Circuit* kann man über Sprache, Video und Chat in Kontakt treten, außerdem bietet es die Möglichkeit des Screensharings und der Dateifreigabe. Vereinzelt wird auch die Kommentarfunktion der Projektmanagementsoftware *OpenProject* eingesetzt.

Die zu entwickelnde Applikation entsteht in einer Servoyumgebung. Diese wurde vom Auftraggeber vorgegeben. Servoy ist eine plattformunabhängige auf dem *Rapid Application Development (RAD) Konzept* basierende Umgebung, die komplett in *Java* entwickelt wurde und aus drei Teilen besteht: Dem Server, dem Client und der Developer-Applikation. Servoy kann mit jeder Datenbank verwendet werden, die eine *JDBC*-Schnittstelle zur Verfügung stellt. Außerdem bietet *Servoy* verschlüsselte Datenübertragung, installationslose Client-Anwendungen (*Java-Web-Start Technologie*), Internationalisierung, Rechtemanagement und Versionierung.

2.3 Vorgehensweise

Die Vorgehensweise in den ersten beiden Projektphasen kann man am ehesten mit dem Wasserfallmodell erklären. Diese wurde nicht explizit vorher gewählt, sie ergibt sich aus dem Umstand, dass die Entwicklungsmethode *Scrum* die Vorbereitung eines Projekts nicht mit beachtet. Die beiden ersten Phasen haben einen definierten Start- und Endzeitpunkt mit Endergebnissen und bauen aufeinander auf. Außerdem enthält jede Phase untergeordnete Aktivitäten, die zum Abschluss der jeweiligen Phase nötig sind. Auf die Erstellung eines Lastenheftes und Pflichtenheftes wurde jedoch verzichtet, sie werden quasi durch das Backlog der *Scrum* Entwicklungsmethode ersetzt. Wie diese Vorgehensweise mit Hilfe einer Projektmanagementsoftware genau aufgebaut ist, kann dem Organisationsabschnitt aus dem vorherigen Unterkapitel entnommen werden.

Bei der Umsetzungsphase wird ein agiler Entwicklungsansatz mit Hilfe der *Scrum*-Methode verwendet. Dazu werden die Anforderungen in *User-Stories* übersetzt und in einem *Product Backlog* zusammengefasst, mit *Storypoints* bewertet und priorisiert, wobei sich die Priorität, Bewertung im Verlauf der Implementierung ändern kann. Die eigentliche Umsetzung erfolgt in vier *Sprints*, die jeweils vierzehn Tage dauern. Vor jedem dieser *Sprints* wird ein sogenanntes *Sprint Planning Meeting* abgehalten, indem dem *Product Backlog User-Stories* entnommen werden, die dann dem kommenden *Sprint* zugeordnet werden. Durch die Bewertung der *User-Stories* kann nach einer Eingewöhnungszeit besser entschieden werden, wie viele Stories in einem *Sprint* abgeschlossen werden können. Nach dem Start eines *Sprints* können die enthaltenen *User-Stories* nicht mehr geändert werden und es werden alle zwei Tage kurze Meetings praktiziert, um den aktuellen Status zu kommunizieren. Am Ende eines Sprints wird ein *Sprint Review Meeting* abgehalten, in dem die Ergebnisse dem *Product Owner* und gegebenenfalls anderen Interessenten präsentiert werden. Das Feedback und neue Anforderungen fließen dann in das nächste *Sprint Planning Meeting* mit ein und der Prozess beginnt von vorne.

2.4 Ziel des Projekts

Ziel des Projekts ist die Entwicklung eines Prototypen für eine Internetschnittstelle zur Erfassung unterschiedlicher Qualitätsprüfungen, die von externen Lieferanten durchgeführt werden am Beispiel von Maßprotokollen. (Diese Protokolle dienen zum Vergleich von Soll-, Istwerten.)

Über die browserbasierte Schnittstelle soll ein Lieferant die Möglichkeit erhalten, seine zu meldenden Qualitätsdaten selbst über eine Erfassungsmaske an die Flender GmbH zurückzumelden. Die so erfassten Daten sollen nach einer automatisierten Prüfung auf Plausibilität und Einhaltung der Toleranzen in die neue Qualitätsdatenbank gespeichert werden. Außerdem soll der Lieferant die Möglichkeit erhalten Qualitätszeugnisse im PDF Format nach erfolgreicher Datenübertragung zu generieren. Zum Ende der Bachelorarbeit soll die Entwicklung abgeschlossen sein, sodass anschließend die Implementierung mit User Acceptance Test und Rollout durchgeführt werden kann.

2.5 Status vor Beginn der Arbeit

Zum jetzigen Zeitpunkt wird bereits eine Internetschnittstelle auf Basis von *SAP NetWeaver* zur Rückmeldung von Qualitätsdaten (Maßprotokolle) externer Lieferanten eingesetzt. Nach erfolgreicher Anmeldung kann der Lieferant zu seinen laufenden Bestellungen Qualitätsdaten in Form von CSV-Dateien hochladen. Anschließend erfolgt eine Plausibilitätsprüfung der Daten und der Hinweis auf eventuell fehlerhafte Eingaben. Nach erfolgreicher Prüfung werden die Daten in ein ERP System überführt, gespeichert und der Anwender kann ein Qualitätszeugnis in Form einer PDF-Datei herunterladen. Die aktuelle Technologie der Schnittstelle wird in Zukunft nicht mehr von *SAP* unterstützt und soll deshalb im September 2018 abgelöst und mit einer neuen Lösung ersetzt werden.

2.6 Projektmeilensteine

Meilenstein	Enddatum
Abschluss der Schnittstellenanalyse	02.03.2018
Abschluss des Applikationsaufbaus	13.04.2018
Prototyp	29.05.2018
Abschluss der Umsetzung	12.06.2018
Abschluss der Bachelorarbeit	**12.06.2018**

2.7 Risikomanagement

RisikoID	Risiko	Effekt	Maßnahmen	Eintrittswahrscheinlichkeit	Tragweite
1	Anforderungen werden falsch verstanden	Unnötige Entwicklungsarbeit und falsche Funktionen	Anforderungen werden schriftlich festgehalten und mit dem Kunden abgestimmt. Regelmäßige Vorstellung der Zwischenergebnisse	10%	6
2	Zu wenig Verständnis bei der Servoyprogrammierung	Deadlines werden nicht rechtzeitig eingehalten, die Applikation wird nicht fertig	Rechtzeitig Probleme kommunizieren und gegebenenfalls Hilfe von Servoy anfordern bzw. intensive Schulungen vor Implementierungsbeginn	40%	6
3	Die Stagingumgebung steht nicht rechtzeitig zur Verfügung	Die Applikation kann nicht deployed werden. Verschiebung Produktivsetzung	Frühzeitiges Beantragen der nötigen Ressourcen oder bestehende Ressourcen nutzen	50%	7
4	Datenbankmodell funktioniert nicht	Altdaten können nicht übernommen werden. Performanceprobleme.	Rechtzeitige Migration von Altdaten in eine Testdatenbank um gegebenenfalls Änderungen am Datenmodell vorzunehmen. Review des Datenmodells	20%	7
5	Datensynchronisation funktioniert nicht	Fehlende Daten	Regelmäßiges Testen der Synchronisationsfunktionalität	30%	9
6	Remote Function Call (RFC) zum SAP-System funktioniert nicht	Fehlende Daten	Frühzeitiges Implementieren der RFC-Schnittstelle und regelmäßige Funktionstests	30%	9

3 Arbeitsbeschreibung

Dieses Kapitel beschreibt meine Aufgaben während des Projekts, anhand von drei Projektphasen. Beginnend mit der Schnittstellenanalyse, dem nachfolgenden Applikationsaufbau und der abschließenden Umsetzung.

3.1 Schnittstellenanalyse

Während der Analysephase besteht die Aufgabe darin, die Anforderungen des Kunden bzw. der Fachabteilungen aufzunehmen und zu verstehen. Dazu werden mehrere Meetings mit den beteiligten Parteien abgehalten, in denen der jetzige Ablauf gezeigt wird, die dabei auftretenden Probleme, sowie Anforderungen für die Zukunft. Zu Beginn des Projektes wurde in einem dreitätigen Workshop die bestehende Servoyumgebung, auf der die neue Lösung basieren soll, erklärt und ein Überblick über die laufenden Projekte gegeben. Es wurden erste Abstimmungen durchgeführt bezüglich der Rahmenbedingungen des Teilprojekts, den generellen Programmiervorgaben und des zu realisierenden Datenmodells.

3.1.1 Vorgehensweise und Ergebnisse

Um die aktuelle Schnittstelle zu analysieren werden mehrere Meetings mit den Fachabteilungen veranstaltet und die Ergebnisse dieser Meetings werden in Form von Anforderungen schriftlich festgehalten. Zur Kontrolle werden die schriftlichen Ergebnisse anderen Projektmitgliedern zu Verfügung gestellt und gegebenenfalls angepasst, bis alle Beteiligten mit dem Ergebnis zufrieden sind. Das Endergebnis dieser Arbeit sind User-Stories für die Umsetzungsphase des Projekts.

3.2 Applikationsaufbau

In der Designphase besteht die Aufgabe darin, das Datenmodell mit zu entwickeln, dass applikationsübergreifend eingesetzt werden soll und dabei die Anforderungen für die Internetschnittstelle mit einzubringen. Das heißt die neue Qualitätsdatenbank und die neue Internetschnittstelle werden auf demselben Datenmodell arbeiten. Aus diesem Grund wird sich häufig mit den Entwicklern ausgetauscht, um ein Modell zu entwickeln dass allen Anforderungen gerecht wird. Das Ergebnis dieser Arbeit wird in einem ERM-Modell festgehalten. Außerdem werden Mockups für die Applikation erstellt und es wird sich dabei regelmäßig mit den Fachabteilungen bzw. Kunden abgestimmt. Forschungsaspekte ergeben sich aus der Servoyumgebung, die für die neue Schnittstelle eingesetzt werden soll. Dazu gehört die Beantwortung der Fragen, auf welcher Plattform die Applikation bereitgestellt wird, welche Infrastruktur benötigt wird und wie die Betreuung gewährleistet wird.

3.2.1 Vorgehensweise und Ergebnisse

In dieser Phase wird zunächst ein ERM-Modell zusammen mit den am Projekt beteiligten Entwicklern erstellt, deren Wissen aus der Betreuung des Altsystems entscheidend ist für die neue Datenstruktur. Um die Qualität des Modells zu gewährleisten wird es allen Beteiligten zur Verfügung gestellt und gegebenenfalls angepasst. Bei der Mockuperstellung wird sich regelmäßig mit den Fachabteilungen ausgetauscht um deren Anmerkungen bei der Erstellung mit einfließen zu lassen und so ein qualitatives Ergebnis zu erzielen. Die fertigen Mockups werden dem gesamten Team zur Verfügung gestellt.

3.3 Umsetzung

Zu den Aufgaben während der Implementierungsphase gehört es die zuvor entwickelten Mockups, nach erfolgreicher Abstimmung mit dem Kunden in der Servoyumgebung umzusetzen

und für die Funktionen die entsprechende Business Logic zu implementieren. Außerdem wird das Datenbankmodell in eine Testdatenbank überführt um die Funktionalität der Software zu testen und eventuell Anpassungen am Datenmodell vorzunehmen. In der Implementierungsphase wird mit der bereits erwähnten Scrum-Methode gearbeitet und alle Fortschritte werden in *OpenProject* protokolliert. Kurz vor dem Ende dieser Phase soll der Prototyp fertigstellt sein, um die Software dem Kunden bzw. der Fachabteilung vorzuführen.

3.3.1 Vorgehensweise und Ergebnisse

Bei der Umsetzung wird mit Hilfe von *OpenProject* nach der Scrum-Methode gearbeitet, sodass alle Arbeitsvorgänge nachvollziehbar in der Projektmanagementsoftware dokumentiert sind. Sämtlicher Programmcode wird mit sinnvollen Tests erstellt und dokumentiert um die Qualität zu gewährleisten.

B Anwendungsfälle

FleQs Lieferantenplattform Anwendungsfälle

Dokumentenhistorie

Version	Status	Datum	Verantwortlicher	Änderungsgrund
1.0	Erstellt	06.04.2018	Nils Nieuwenhuis	
1.1	Ergänzt	13.04.2018	Nils Nieuwenhuis	Ergänzungen

1 Anmelden

Name	Anmelden
Akteur	Lieferant
Szenario	1. Anwender wählt die Sprache aus. 2. Anwender gibt seinen Benutzernamen ein. 3. Anwender gibt sein Passwort ein. 4. Anwender klickt auf den Login-Button.
Trigger	Ein Lieferant möchte seine Qualitätsdaten übergeben und die Webseite wird aufgerufen.
Vorbedingung	- Ein gültiges Benutzerkonto

2 Abmelden

Name	Abmelden
Akteur	Lieferant
Szenario	1. Anwender klickt auf den Logout-Button.
Trigger	Der Anwender unterbricht seine Arbeit oder hat sie abgeschlossen.
Vorbedingung	- Der Anwender ist angemeldet.

3 Arbeitsvorrat anzeigen

Name	Arbeitsvorrat anzeigen
Akteur	Lieferant
Szenario	1. Anwender meldet sich an. 2. Anwender sieht seine aktuellen Bestellungen. 3. Der Anwender wählt eine Bestellung aus. 4. Der Anwender sieht die zur Bestellung gehörenden Prüflinge. 5. Der Anwender wählt einen Prüfling aus.
Trigger	Der Anwender hat sich angemeldet.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Prüfdatenerfassung. - Es sind offene Bestellungen vorhanden.

4 Prüfungsdaten eingeben

Name	Prüfungsdaten eingeben
Akteur	Lieferant
Szenario	1. Der Anwender gibt seine Prüfparameter ein. 2. Der Anwender gibt seine Istwerte ein. 3. Der Anwender sieht ob seine Daten valide sind. 4. Der Anwender speichert seine Eingaben
Trigger	Der Anwender hat eine Bestellung aus dem Arbeitsvorrat ausgewählt.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Prüfdatenerfassung. - Es sind offene Bestellungen vorhanden. - Es wurde eine Bestellung ausgewählt. - Es wurde ein Prüfling ausgewählt - Der Prüfling ist nicht abgeschlossen. - Die Bestellung ist nicht gesperrt.

5 Prüfungsdaten ändern

Name	Prüfungsdaten ändern
Akteur	Lieferant
Szenario	1. Anwender ändert Daten. 2. Anwender speichert seine Änderungen.
Trigger	Falsche oder fehlende Eingaben wurden vom Anwender gemacht.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Prüfdatenerfassung. - Es sind offene Bestellungen vorhanden. - Es wurde eine Bestellung ausgewählt. - Es wurde ein Prüfling ausgewählt. - Der Prüfling ist nicht abgeschlossen. - Die Bestellung ist nicht gesperrt. - Es wurden Eingaben gemacht.

6 Ausnahmeantrag eingeben

Name	Ausnahmeantrag eingeben
Akteur	Lieferant
Szenario	1. Anwender gibt Ausnahmeantrag ein. 2. Anwender sieht ob sein Antrag gültig ist. 3. Anwender speichert seine Änderungen.
Trigger	Die eingegebenen Daten des Anwenders sind nicht valide.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Prüfdatenerfassung. - Es sind offene Bestellungen vorhanden. - Es wurde eine Bestellung ausgewählt. - Es wurde ein Prüfling ausgewählt. - Der Prüfling ist nicht abgeschlossen. - Die Bestellung ist nicht gesperrt. - Es wurden ungültige Eingaben gemacht. - Ein Ausnahmeantrag ist vorhanden.

7 Datei hochladen

Name	Datei hochladen
Akteur	Lieferant
Szenario	1. Anwender klickt auf Upload-Button. 2. Anwender wählt eine Datei aus. 3. Anwender klickt auf Upload.
Trigger	Der Anwender möchte zusätzliche Dateien zu einer Prüfung zur Verfügung stellen.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Prüfdatenerfassung. - Es sind offene Bestellungen vorhanden. - Es wurde eine Bestellung ausgewählt. - Es wurde ein Prüfling ausgewählt - Der Prüfling ist nicht abgeschlossen. - Die Bestellung ist nicht gesperrt.

8 Prüfprotokoll runterladen

Name	Prüfprotokoll runterladen
Akteur	Lieferant
Szenario	1. Anwender klickt auf den Download Button. 2. Anwender wählt den Zielort aus. 3. Anwender klickt auf Speichern.
Trigger	Der Anwender hat alle Daten komplett ausgefüllt und braucht das Prüfprotokoll zur weiteren Verarbeitung.
Vorbedingung	- Der Anwender ist angemeldet. - Der Anwender befindet sich in der Dokumentenverwaltung - Es sind Bestellungen vorhanden. - Der Prüfling des gewünschten Protokolls ist abgeschlossen. - Die Daten sind geprüft.

9 Benutzer verwalten

Name	Benutzer verwalten
Akteur	Administrator
Szenario	1. Anwender meldet sich an der Benutzerdatenbank an. 2. Anwender erstellt oder löscht einen Benutzer.
Trigger	Ein neuer Lieferant möchte die Schnittstelle verwenden oder ein bestehender Lieferant möchte die Schnittstelle nicht mehr verwenden.
Vorbedingung	- Der Benutzer ist ein realer gültiger Lieferant.

C Product Backlog

THEMA	TYP	ÜBERGEORDNETE AUFGABE	VERSION
Anmelden	Epic	-	Epics
Als Lieferant möchte ich mich am System anmelden	User story	#208	Sprint 3
Abmelden	Epic	-	Epics
Als Lieferant möchte ich mich vom System abmelden	User story	#209	Product Backlog
Arbeitsvorrat anzeigen	Epic	-	Epics
Als Lieferant möchte ich einen Arbeitsvorrat der aktuellen Bestell...	User story	#223	Sprint 2
Als Lieferant möchte ich den Status einer Bestellung sehen	User story	#223	Product Backlog
Daten zu einer Qualitätsprüfung eingeben	Epic	-	Epics
Als Lieferant möchte ich Istwerte zu einem Maßprotokoll pflegen	User story	#224	Sprint 2
Als Lieferant möchte ich meine eingetragenen Istwerte zu einem ...	User story	#224	Sprint 3
Als Lieferant möchte ich eine Überprüfung meiner eingetragenen...	User story	#224	Sprint 3
Als Lieferant möchte ich die Sollwerte zu einem Maßprotokoll an...	User story	#224	Sprint 3
Als Lieferant möchte ich Kopfdaten/Prüfparameter zu einem Ma...	User story	#224	Sprint 2
Daten zu einer Qualitätsprüfung ändern	Epic	-	Epics
Als Lieferant möchte ich meine eingetragenen Istwerte zu einem ...	User story	#225	Sprint 3
Als Lieferant möchte ich meine eingetragenen Kopfdaten/Prüfpa...	User story	#225	Sprint 3
Ausnahmeantrag eingeben	Epic	-	Epics
Als Lieferant möchte ich einen Ausnahmeantrag zu einem Istwert...	User story	#226	Product Backlog
Datei zu einer Qualitätsprüfung hochladen	Epic	-	Epics
Als Lieferant möchte ich eine Datei zum Pflegen von Istwerten h...	User story	#227	Product Backlog

Abbildung 14: Product Backlog

D Forschungsarbeit

Erforschung der möglichen Plattformen zur Bereitstellung von Webapplikationen innerhalb der Flender GmbH

Nils Nieuwenhuis

Fontys Hochschule für Technik und Logistik

Wirtschaftsinformatik

Bocholt, 02. April 2018

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument behandelt die Forschung der möglichen Plattformen für Webapplikationen innerhalb der Flender GmbH, im Rahmen der Bachelorarbeit zur Erstellung einer neuen Lieferantenplattform für das Flender Qualitätssystem (FleQs).

Als erstes werden die Kriterien, die bei der Wahl einer geeigneten Plattform beachtet werden müssen, gesammelt und beschrieben. Danach werden die Kriterien mit einer Bewertung versehen und anschließend die verschiedenen Möglichkeiten aufgezählt. Die Möglichkeiten werden kurz beschrieben und mit einer Bewertungsmatrix wird gezeigt, welche Kriterien von der jeweiligen Möglichkeit erfüllt werden. Abschließend wird eine Empfehlung für eine Plattform beschrieben und erklärt.

2 Kriterien

Die Kriterien die für eine Plattform zur Bereitstellung von Webapplikationen herangezogen werden, gliedern sich in fünf Hauptkriterien: Betreuung, Funktionen, Sicherheit, Kosten und Performance. Nachfolgend sind diese mit ihren zugehörigen Kriterien beschrieben. Zur Bestimmung der Kriterien wurden Interviews mit Inhouse IT-Beratern, Entwicklern und Mitarbeitern der IT-Infrastruktur der Flender GmbH geführt.

2.1 Betreuungskriterien

2.1.1 Backups

ID	2.1.1
Titel	Backups
Beschreibung	Es werden automatisch Backups von den genutzten Services angefertigt.
Grund	Bei Problemen kann zu einem früheren funktionierenden Zustand zurückgekehrt werden.
Quelle	Entwickler, Berater, IT- Infrastruktur
Gewichtung	3

2.1.2 Updates

ID	2.1.2
Titel	Updates
Beschreibung	Automatische Installation von Updates, insbesondere Sicherheitsupdates.
Grund	In der heutigen Zeit sind Updates ein absolutes Muss, um die Sicherheit des Systems aufrecht zu erhalten.
Quelle	Entwickler, Berater, IT- Infrastruktur
Gewichtung	3

2.1.3 Verfügbarkeit

ID	2.1.3
Titel	Verfügbarkeit
Beschreibung	Systeme müssen 24/7 zur Verfügung stehen mit einer maximalen Ausfallzeit von einem Tag.
Grund	Um Verzögerungen und Informationsverlust zu vermeiden.
Quelle	Berater
Gewichtung	3

2.2 Funktionskriterien

2.2.1 Microsoft SQL Server

ID	2.2.1
Titel	Microsoft SQL Server
Beschreibung	Die Plattform sollte als Datenbankserver mindestens Microsoft SQL Server unterstützen.
Grund	Die geplanten Datenbanken des Unternehmens basieren auf Microsoft SQL Server.
Quelle	Entwickler, Berater
Gewichtung	3

2.2.2 VPN Verbindung

ID	2.2.2
Titel	VPN Verbindung
Beschreibung	Die Plattform sollte VPN-Verbindungen unterstützen.
Grund	Applikationen benötigen Daten aus den Unternehmenssystemen.
Quelle	Entwickler, Berater
Gewichtung	3

2.2.3 Skalierbarkeit

ID	2.2.3
Titel	Skalierbarkeit
Beschreibung	Es können zu jeder Zeit mehr Ressourcen oder weniger Ressourcen genutzt werden.
Grund	Applikationen können im Laufe der Zeit mehr oder weniger Ressourcen benötigen bzw. weitere Applikationen kommen hinzu.
Quelle	Entwickler, Berater
Gewichtung	2

2.2.4 Flexibilität

ID	2.2.5
Titel	Flexibilität
Beschreibung	Die Plattform bietet Möglichkeiten weitere Services wie Load-Balancer, Reverse-Proxys, Monitoring-Systeme etc. zur Systemkonfiguration hinzuzufügen.
Grund	Bei wachsender Ressourcenverwendung, werden Services zur Verwaltung und Betreuung benötigt.
Quelle	Entwickler, Berater
Gewichtung	2

2.2.5 Container-Virtualisierung

ID	2.2.6
Titel	Container-Virtualisierung
Beschreibung	Die Plattform bietet die Möglichkeit Container Virtualisierung zur Bereitstellung von Applikationen zu nutzen.
Grund	Die Flender GmbH möchte in Zukunft Applikationen containerbasiert bereitstellen.
Quelle	Entwickler, Berater
Gewichtung	1

2.3 Sicherheitskriterien

2.3.1 Berechtigungskonzept

ID	2.3.1
Titel	Berechtigungskonzept
Beschreibung	Ein Berechtigungskonzept um die Rechte der Anwender auf der Plattform zu definieren.
Grund	Nicht jeder Anwender soll die Möglichkeit haben alles konfigurieren zu können.
Quelle	Entwickler, Berater, IT- Infrastruktur
Gewichtung	3

2.3.2 Monitoring

ID	2.3.2
Titel	Monitoring
Beschreibung	Die Möglichkeit Vorgänge am System nachzuverfolgen.
Grund	Eventuell auftretende Probleme oder falsche Eingaben können zurückverfolgt werden und leichter behoben werden.
Quelle	IT- Infrastruktur
Gewichtung	2

2.3.3 Verschlüsselung

ID	2.3.3
Titel	Verschlüsselung
Beschreibung	Der Aufruf der auf der Plattform zur Verfügung gestellten Applikationen bzw. Services soll ausschließlich verschlüsselt erfolgen.
Grund	Um die Anwender und den Datenverkehr vor Missbrauch zu schützen.
Quelle	IT- Infrastruktur
Gewichtung	3

2.3.4 Sicherheitszonen

ID	2.3.4
Titel	Sicherheitszonen
Beschreibung	Services oder Systeme können in verschiedenen Zonen bereitgestellt werden. Eine Zone ist zum Beispiel nicht aus dem Internet erreichbar, eine andere dafür schon.
Grund	Aus Sicherheitsgründen dürfen nicht alle Applikationen aus dem Internet erreichbar sein, z.B. Datenbanken.
Quelle	IT- Infrastruktur
Gewichtung	3

2.4 Kostenkriterien

Aufgrund in Verhandlung befindlicher Verträge mit dem IT-Dienstleister der Flender GmbH ist zum Zeitpunkt dieser Forschung kein Vergleich möglich.

2.5 Performancekriterien

2.5.1 Servoy Systemanforderungen

ID	2.5.1
Titel	Servoy Systemanforderungen
Beschreibung	Die folgenden Anforderungen müssen von der Plattform unterstützt werden:
Grund	Das Unternehmen plant mehrere Servoyapplikationen umzusetzen.
Quelle	Berater
Gewichtung	3

Servoy Application Server

Windows	Linux	OS X
Intel Pentium III, IV, M, D 1Ghz or higher	Intel Pentium III, IV, M, D 1Ghz or higher	Intel CPU 1 Ghz or higher
1.5Gb RAM	1.5Gb RAM	1.5Gb RAM
850Mb Free diskspace	850Mb Free diskspace	850Mb Free diskspace
All client and server Windows versions since Windows 2000 (see List of Microsoft Windows versions)	Java Standard Edition runtime Environment 7.0 or higher	Mac OS X 10.4 or higher
Java Standard Edition runtime Environment 7.0 or higher		Java Standard Edition runtime Environment 7.0 or higher
Networking: TCP/IP		

2.6 Bewertung der Kriterien

Die Bewertung der einzelnen Kriterien erfolgt mit einer Skala von eins bis drei. Drei bedeutet, dass das Kriterium unerlässlich für die Funktionsweise ist und eins, dass das Kriterium geringen Einfluss auf die Funktionsweise hat.

Name	Kriterium	Gewichtung
Betreuungskriterien		
Backups	2.1.1	3
Updates	2.1.2	3
Verfügbarkeit	2.1.3	3
Funktionskriterien		
Microsoft SQL Server	2.2.1	3
VPN Verbindung	2.2.2	3
Skalierbarkeit	2.2.3	2
Flexibilität	2.2.4	2
Container-Virtualisierung	2.2.5	1
Sicherheitskriterien		
Berechtigungskonzept	2.3.1	3
Monitoring	2.3.2	2
Verschlüsselung	2.3.3	3
Sicherheitszonen	2.3.4	3
Kostenkriterien		
Performancekriterien		
Servoy Systemanforderungen	2.5.1	3

3 Plattformen

In diesem Kapitel werden die einzelnen Plattformen, die für die Flender GmbH in Frage kommen, beschrieben und bewertet. Die Bewertung erfolgt mit der zuvor aufgestellten Gewichtungsmatrix und mit einem Erfüllungsgrad von eins bis fünf, wobei eins für nicht erfüllt steht und fünf dafür, dass das Kriterium voll erfüllt ist. Der Erfüllungsgrad wird mit der Gewichtung multipliziert um eine Punktezahl pro Kriterium zu erhalten. Diese Punkte werden pro Plattform zu einer Gesamtpunktzahl zusammengefasst.

3.1 Server-VM innerhalb der Flender GmbH

Diese Plattform wäre eine virtuelle Maschine bereitgestellt vom IT-Dienstleister, der für die IT-Infrastruktur der Flender GmbH zuständig ist.

Die virtuelle Maschine würde auf Windows Server 2012 R2 basieren und komplett vom Dienstleister betreut. Die Entwickler würden Zugang zum System bekommen um die benötigte Software auf dem System zu installieren.

Name	Gewichtung	Erfüllungsgrad	Punkte
Backups	3	5	15
Updates	3	5	15
Verfügbarkeit	3	4	12
Microsoft SQL Server	3	5	15
VPN Verbindung	3	5	15
Skalierbarkeit	2	2	4
Flexibilität	2	2	4
Container-Virtualisierung	1	1	1
Berechtigungskonzept	3	5	15
Monitoring	2	3	6
Verschlüsselung	3	5	15
Sicherheitszonen	3	3	9
Servoy Systemanforderungen	3	5	15
Gesamtpunktzahl	**141**		

3.2 Cloudplattform

Die Cloudanbieter die für die Flender GmbH in Frage kommen sind Amazon Web Services und Microsoft Azure, weil die Muttergesellschaft, die Siemens AG, bereits Rahmenverträge mit diesen Anbietern geschlossen hat und die Flender GmbH diese verwenden kann.

Die Cloudplattform würde zentral von der IT-Abteilung verwaltet, wobei einzelne Entwickler die nötigen Rechte bekommen würden um die benötige Infrastruktur aufzusetzen. Die Betreuung würde dabei durch die Entwickler erfolgen.

Name	**Gewichtung**	**Efüllungsgrad**	**Punkte**
Backups	3	5	15
Updates	3	5	15
Verfügbarkeit	3	5	15
Microsoft SQL Server	3	5	15
VPN Verbindung	3	5	15
Skalierbarkeit	2	5	10
Flexibilität	2	5	10
Container-Virtualisierung	1	5	5
Berechtigungskonzept	3	5	15
Monitoring	2	5	10
Verschlüsselung	3	5	15
Sicherheitszonen	3	5	15
Servoy Systemanforderungen	3	5	15
Gesamtpunktzahl	**170**		

4 Empfehlung

Basierend auf der Bewertung empfehle ich die Umsetzung in einer Cloudplattform. Nachfolgend anhand von einem Beispielaufbau bei Amazon Web Services (AWS) gezeigt. AWS bietet zudem viele Möglichkeiten für Container Virtualisierung, die das Unternehmen in Zukunft verwenden möchte.

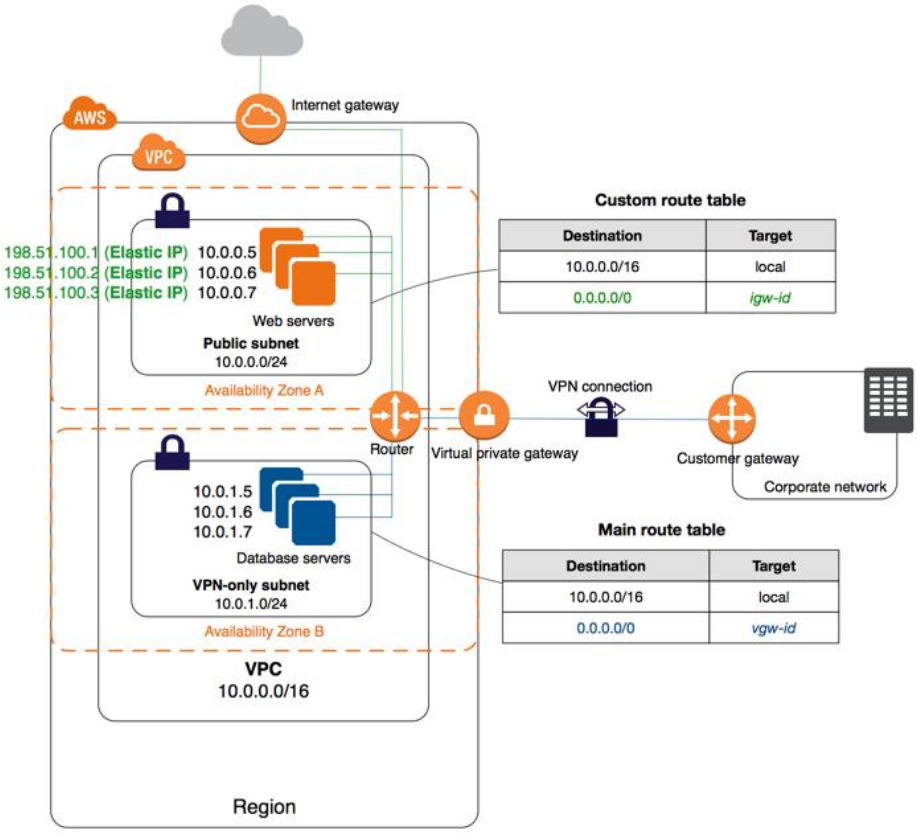


Abbildung 1: AWS Beispielaufbau
(https://docs.aws.amazon.com/de_de/AmazonVPC/latest/UserGuide/images/Case3_Diagram.png)

Der hier gezeigte Aufbau enthält die folgenden Optionen[1]:

- Eine Virtual Private Cloud (VPC). Eine VPC ist ein virtuelles Netzwerk innerhalb der Amazon Umgebung.
- Ein öffentliches Subnetz, ein öffentliches Subnetz ist ein Subnetz, das einer Routing-Tabelle zugeordnet ist, die über eine Route zu einem Internet-Gateway verfügt und damit aus dem Internet erreichbar ist
- Ein reines privates VPN-Subnetz, ein privates Subnetz ist nur innerhalb der Virtual Private Cloud erreichbar.
- Ein Internet-Gateway. Dies ermöglicht der VPC die Verbindung zum Internet und zu anderen AWS-Services.
- Eine VPN-Verbindung zwischen der VPC und dem Unternehmensnetzwerk. Die VPN-Verbindung besteht aus einem Virtual Private Gateway auf der Seite von Amazon, der VPN-Verbindung, sowie einem Gateway auf Unternehmensseite der VPN-Verbindung.
- Instanzen im privaten Subnetzbereich, um es den Instanzen zu ermöglichen, miteinander und mit anderen Instanzen in der VPC zu kommunizieren. Bei diesen Instanzen handelt es sich in der Regel um Backend-Server, die keinen eingehenden Datenverkehr vom Internet akzeptieren müssen, aber Datenverkehr von Ihrem Netzwerk senden und empfangen können.
- Instanzen im öffentlichen Subnetz mit Elastic IP-Adressen. Dabei handelt es sich um öffentliche IPv4-Adressen, durch die der Zugriff vom Internet ermöglicht wird. Den Instanzen können beim Start anstatt der Elastic IP-Adressen öffentliche IPv4-Adressen zugewiesen werden. Bei diesen Instanzen handelt sich in der Regel um Frontend-Server, die Applikationen hosten, die Datenverkehr vom Internet akzeptieren müssen.
- Eine benutzerdefinierte Routing-Tabelle, die dem öffentlichen Subnetz zugeordnet ist. Diese Routing-Tabelle enthält einen Eintrag, der es Instanzen im Subnetz ermöglicht, mit anderen Instanzen in der VPC zu kommunizieren, sowie einen Eintrag, der es Instanzen im Subnetz ermöglicht, direkt mit dem Internet zu kommunizieren.
- Die Haupt-Routing-Tabelle, die dem privaten Subnetz zugeordnet ist. Diese Routing-Tabelle enthält einen Eintrag, der es Instanzen im Subnetz ermöglicht, mit anderen Instanzen in der VPC zu kommunizieren, sowie einen Eintrag, der es Instanzen im Subnetz ermöglicht, direkt mit Ihrem Netzwerk zu kommunizieren.

[1] Amazon: „Szenario 3: VPC mit öffentlichem und privatem Subnetz und von AWS verwaltetem VPN-Zugang“, unter https://docs.aws.amazon.com/de_de/AmazonVPC/latest/UserGuide/VPC_Scenario3.html (05.05.2018)

- Eine Amazon Elastic Compute Cloud (Amazon EC2) Auto Scalling Gruppe um Instanzen automatisch bereit zu stellen und je nach Bedarf mehrere Instanzen, nach vorher definierten Regeln, zu starten oder zu stoppen.
- Alternativ einen Amazon Elastic Container Service (Amazon ECS), zum Ausführen von Docker-Containern in einem Cluster von Amazon-EC2 Instanzen.

Literaturverzeichnis

Amazon (2018a): Szenario 3: VPC mit öffentlichem und privatem Subnetz und von AWS verwaltetem VPN-Zugang, [online] https://docs.aws.amazon.com/de_de/AmazonVPC/latest/UserGuide/VPC_Scenario3.html [05.05.2018].

(2018b): Amazon Elastic Container Service – Dokumentation, [online] https://aws.amazon.com/de/documentation/ecs/?icmpid=docs_menu [10.05.2018].

(2018c): Was ist Amazon EC2 Auto Scaling?, [online] https://docs.aws.amazon.com/de_de/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html [aufgerufen 10.05.2018].

E Servoy-Formulare

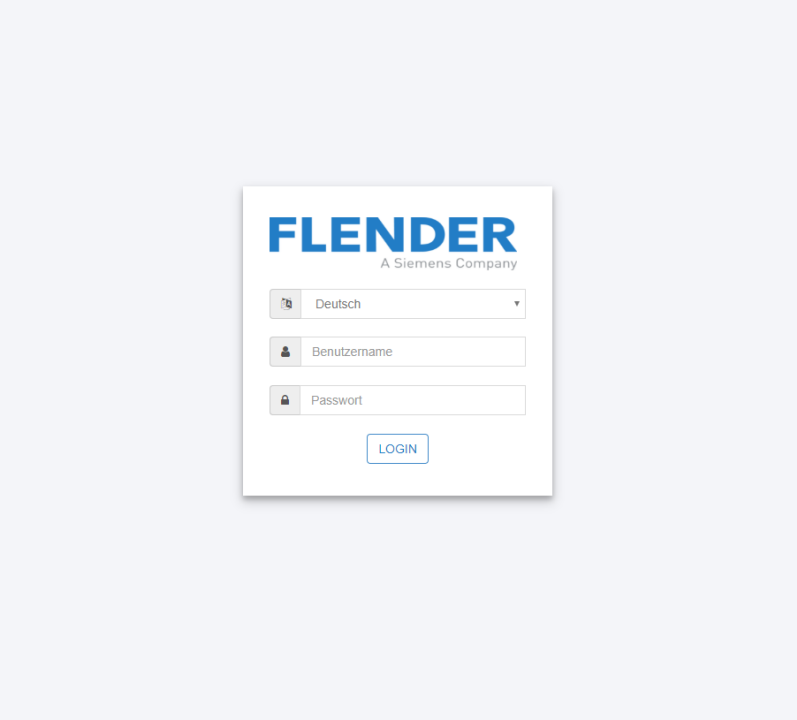


Abbildung 15: Screenshot Login Formular

BESTELLNUMMER	MATERIALNUMMER	MATERIALTEXT	ZEICHNUNGSNUMMER
	1517882	STIRNRAD LP 622,0/1,00 1.6587	6251236
4501225280 10	1246924	WELLE-HOHL D170/53X750 42CrMo4V	87312688d
	1588338	STIRD-PLD RP 534,5/1,52 SE 1.6587	6358112
	617229	STRDWELLE LP 225,0/3	00000518159l
	1271831	TEIL 1 BCK 63/240 ENG;IS400-18-LT	6068863n
	1241961	ADAPTER E2/3 HA4 34CRNIMO6	6030744b
2610697606 10	23087306075	STRDWELLE LP 451,3/2,11 1.6587	87306071f
	954807	STIRNRAD LP 125,0/1,53 1.6587	5743956b
	A5E38562032	TEIL 2 191.9X55.5X82 ZBG	a5e38562032aab
	1539717	TEIL 2 184X86 ZBG	6315936
	1608639	STRDWELLE LP 482,0/3,92 SE LK1 1.6587	6329774
	654689	STIRNRAD RP 478,0/4,14 SE 1.6587	6002757h
	650743	BLZN 225,0X790 34CrNiMo6	81128l27a
	2400006925	STIRNRAD D209,72/90X56 02.4000 06925	6096629
	1593088	STIRNRAD RP 482,0/4,91 SE 1.6587	6348792c
	9900069	RUPEX TEIL 1/2	
	540926	RADKRANZ 160 17,5R CC484K	5540926e
	1442480	STIRNRAD LP 320,0/2,03 1.6582	57549l35a
4501215028 10	1286928	STIRNRAD LP 679,0/3,86 SE 1.6587	6082490c
	1392762	STIRNRAD R 160,0/3,94 711-24-410EE	6186516

Abbildung 16: Screenshot DimensionalProtocolWorklist Formular

Abbildung 17: Screenshot DimensionalProtocolTest Formular

F Servoy Coding StyleGuide

Coding StyleGuide

For all JavaScript related conventions, lease see http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml

General

- names are camel-cased
- Always use curly brackets for block statements

Constants

- Names are all upper case, using underscore as separators

Constructor functions

- Names are single words and the name starts with a capital

Exceptions

- Names are camelCased, with a slight variation: they also start with a capital letter
- Name ends with "Exception"

Module naming

- names describe the purpose of the module and use abbreviations sparsely
- Module that are integral part of the BAP are prefixed with 'svy', for example svyMonitoring
- Modules that are completely standalone and can be used outside the BAP are prefixed with 'mod', for example modDialogs or modGoogleAnalytics
- UnitTest modules have the same name as the module they are intended to test, postfixed with '_test', i.e. modGoogleAnalytics_test
- Modules that implement only a RESTful API on top of another module have the same name as the module they belong to, postfixed with '_api'

Value Lists

- All BAP value lists should be prefixed 'svy'
- Table names when used (table/relation-based) should be implied in the name, i.e. 'svyUsers$active', where the table name is 'users'. If the name is 'sec_users' drop the prefix.

Relation Names

- Normal Table-to-Table relations should follow the default naming. i.e. 'customers_to_orders'
- Table-to-Table relations which have additional relation items should use the default name appended with a short descriptive name which is delimited by a '$' character - table_to_table$<description>
  i.e. 'customers_to_orders$status_cancelled'
- Global relations which involve a single table should always begin with the table name and be appended with a short descriptive name which is delimited by a '$' character - table_name$<description>, i.e. customers$active
- Container relations should take the form of <table_name>$container, i.e. customers$container

G Servoy Programmcodebeispiel

```
/**
 * @type {JSFoundSet}
 *
 * @properties={typeid:35,uuid:"8DCF18E1-E435-4A00-BDDA-258CA669D2A4",variableType:-4}
 */
var productionOrderFoundset;
/**
 * @type {JSFoundSet}
 *
 * @properties={typeid:35,uuid:"5AE82D41-BE30-4B05-B533-494A348F2EAB",variableType:-4}
 */
var productionOrderItemFoundset;
/**
 * @type {JSFoundSet}
 *
 * @properties={typeid:35,uuid:"363914BA-8EB5-443B-B5DD-3DC28881FD6D",variableType:-4}
 */
var testFoundset;
/**
 * @type {JSFoundSet}
 *
 * @properties={typeid:35,uuid:"D8D714EB-A759-45E4-9977-66654B7E9A34",variableType:-4}
 */
var testItemFoundset;

/**
 * @type {scopes.TestItem.TestItem}
 *
 * @properties={typeid:35,uuid:"96D218BD-0922-435F-8D2A-23BFD8553835",variableType:-4}
 */
var testItem;

/**
 * @type {JSRecord<db:/fleqs/test_item>}
 *
 * @properties={typeid:35,uuid:"2B56B8A9-EB27-416E-8702-84D6BFF99A44",variableType:-4}
 */
var actualTestItemRecord;

/**
 * @AllowToRunInFind
 *
 * @properties={typeid:24,uuid:"EE4D47E6-DA96-4D3B-ADBF-9C341A8B1A3E"}
 */
function setUp() {
```

```
    // start database transaction
    databaseManager.startTransaction();

    // get the production order foundset
    productionOrderFoundset = datasources.db.fleqs.production_order.getFoundSet();

    // create a new record in the foundset and fill the new record with testdata
    var productionOrderRecord = productionOrderFoundset.getRecord(productionOrderFoundset.newRecord());
    productionOrderRecord.production_order_no = '12345678901213';
    productionOrderRecord.quantity = 1;
    productionOrderRecord.customer_purchase_order_no = '12345678901213';
    productionOrderRecord.material_no = '123456789012';
    productionOrderRecord.drawing_no = '1234567';
    productionOrderRecord.drawing_no_version = 'A';

    // save the new record in the database
    if (databaseManager.saveData(productionOrderRecord)) {
        application.output('UnitTest ProductionOrder was temporally created!');
    } else {
        application.output('UnitTest ProductionOrder wasnt created!');
        databaseManager.rollbackTransaction();
    }

    // set the foundset to the created production oder
    if (productionOrderFoundset.find()) {
        productionOrderFoundset.production_order_no = '12345678901213';
        productionOrderFoundset.quantity = 1;
        productionOrderFoundset.customer_purchase_order_no = '12345678901213';
        productionOrderFoundset.search();
    }

    // get the production order item foundset
    productionOrderItemFoundset = datasources.db.fleqs.production_order_item.getFoundSet();

    // create a new record in the foundset and fill the new record with testdata
    var productionOrderItemRecord =
productionOrderItemFoundset.getRecord(productionOrderItemFoundset.newRecord());
    productionOrderItemRecord.production_order_id = productionOrderFoundset.production_order_id;
    productionOrderItemRecord.production_order_pos = '1';

    // save the new record in the database
    if (databaseManager.saveData(productionOrderItemRecord)) {
        application.output('UnitTest ProductionOrderItem was temporally created!');
    } else {
        application.output('UnitTest ProductionOrderItem wasnt created!');
```

```
    databaseManager.rollbackTransaction();
}

// set the foundset to the created production oder item
if (productionOrderItemFoundset.find()) {
    productionOrderItemFoundset.production_order_id = productionOrderFoundset.production_order_id;
    productionOrderItemFoundset.production_order_pos = '1';
    productionOrderItemFoundset.search();
}

// get the test foundset
testFoundset = datasources.db.fleqs.test.getFoundSet();

// create a new record in the foundset and fill the new record with testdata
var testRecord = testFoundset.getRecord(testFoundset.newRecord());
testRecord.test_type = 21;
testRecord.entity_id = productionOrderFoundset.production_order_id;
testRecord.entity_type = 1;

// save the new record in the database
if (databaseManager.saveData(testRecord)) {
    application.output('UnitTest Test was temporally created!');
} else {
    application.output('UnitTest Test wasnt created!');
    databaseManager.rollbackTransaction();
}

// set the foundset to the created test
if (testFoundset.find()) {
    testFoundset.test_type = 21;
    testFoundset.entity_id = productionOrderFoundset.production_order_id;
    testFoundset.entity_type = 1
    testFoundset.search()
}

// get the test item foundset
testItemFoundset = datasources.db.fleqs.test_item.getFoundSet();

// create a new record in the foundset and fill the new record with testdata
var testItemRecord = testItemFoundset.getRecord(testItemFoundset.newRecord());
testItemRecord.creation_user_id = null;
testItemRecord.test_id = testFoundset.test_id;
testItemRecord.entity_id = productionOrderItemFoundset.production_order_item_id;

// save the new record in the database
```

```
    if (databaseManager.saveData(testItemRecord)) {
        application.output('UnitTest TestItem was temporally created!');
    } else {
        application.output('UnitTest TestItem wasnt created!');
        databaseManager.rollbackTransaction();
    }

    // set the foundset to the created test item
    if (testItemFoundset.find()) {
        testItemFoundset.test_item_id = testItemFoundset.test_item_id;
        testItemFoundset.entity_id = productionOrderItemFoundset.production_order_item_id;
        testItemFoundset.test_id = testFoundset.test_id;
        testItemFoundset.search();
    }

    // get the current test item record
    actualTestItemRecord = testItemFoundset.getRecord(testItemFoundset.getSelectedIndex());
}

/**
 * @properties={typeid:24,uuid:"4C87C2C9-54C4-4264-A27D-D6936994B434"}
 * @AllowToRunInFind
 */
function test_createParameter() {
    // create a new DimensionalProtocolTestItem object with the current test item record
    testItem = new scopes.DimensionalProtocolTestItem.DimensionalProtocolTestItem(actualTestItemRecord);

    // get the ladleSign value which creates a new parameter because the test item has no parameter at the
moment
    testItem.ladleSign;

    // get the parameter foundset and set it to the new created parameter
    var parameterFoundset = datasources.db.fleqs.test_parameter.getFoundSet();
    if (parameterFoundset.find()){
        parameterFoundset.test_id = testFoundset.test_id;
        parameterFoundset.test_item_id = testItemFoundset.test_item_id;
        parameterFoundset.test_parameter = scopes.TestParameter.PARAMETER_TYPE.LADLE_SIGN;
        parameterFoundset.search();
    }

    // check if the database has a record
    jsunit.assertEquals(1, parameterFoundset.getSize());
}

/**
```

```
 * @AllowToRunInFind
 *
 * @properties={typeid:24,uuid:"2E078055-9017-49C0-92A3-B9FD62B65D7F"}
 */
function test_setParameterValue() {
    // create a new DimensionalProtocolTestItem object with the current test item record
    testItem = new scopes.DimensionalProtocolTestItem.DimensionalProtocolTestItem(actualTestItemRecord);

    // get the ladleSign value which creates a new parameter because the test item has no parameter at the
moment
    testItem.ladleSign;

    // set the parameter value
    testItem.ladleSign = 'test';

    // save temporary changes in the database
    testItem.saveParameters();

    // get the parameter foundset and set it to the new created parameter
    var parameterFoundset = datasources.db.fleqs.test_parameter.getFoundSet();
    if (parameterFoundset.find()){
        parameterFoundset.test_id = testFoundset.test_id;
        parameterFoundset.test_item_id = testItemFoundset.test_item_id;
        parameterFoundset.test_parameter = scopes.TestParameter.PARAMETER_TYPE.LADLE_SIGN;
        parameterFoundset.search();
    }

    // check if the value is present in the database
    jsunit.assertEquals('test', parameterFoundset.parameter_value_text);
}

/**
 * @properties={typeid:24,uuid:"AE3F909E-D583-4BEE-A6BF-61B8A9C58B7B"}
 */
function test_getParameter() {
    // create a new DimensionalProtocolTestItem object with the current test item record
    testItem = new scopes.DimensionalProtocolTestItem.DimensionalProtocolTestItem(actualTestItemRecord);

    // get the ladleSign value which creates a new parameter because the test item has no parameter at the
moment
    testItem.ladleSign;

    // set the parameter value
    testItem.ladleSign = 'test';
```

```
    // save temporary changes in the database
    testItem.saveParameters();

    // check if the getter returns the value from the database
    jsunit.assertEquals('test', testItem.ladleSign);
}

/**
 * @properties={typeid:24,uuid:"AD3A70D4-8FB5-4A50-9FD3-4148092C9F1D"}
 */
function tearDown() {
    // rollback all changes in the database
    databaseManager.rollbackTransaction();
}
```