HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Gema

Seriennummernverwaltung Gema GmbH

Bachelorarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Frühlingssemester 2013

Autor(en):	Lorenz Bösch
	Christoph Bühler
	Patrick Lötscher
Betreuer:	Prof. Hansjörg Huser

Aufgabenstellung

Seriennummernverwaltung Gema GmbH

Aufgabenstellung für Lorenz Bösch, Christoph Bühler, Patrick Lötscher

Ausgangslage:

Die bestehende Seriennummernverwaltung ersetzen und auf den neuesten Stand bringen. Aktuell ist die Applikation, welche eingesetzt wird, eine WinForms .NET Framework 2.0 Applikation, welche direkt mit dem SQL Server kommuniziert. Die Applikation hat enorme Performanceprobleme und ist nicht auf dem Stand der Technik.

Ziel der Arbeit:

Überarbeitung der Software nach den Standards des Softwareengineerings. Die Zielapplikation soll als Webapplikation auf einem IIS laufen. Zum Abschluss der Arbeit soll eine komplette und performante Lösung entstanden sein, welche sich danach im produktiven Umfeld verwenden lässt.

Der Projektumfang besteht aus:

- der fertigen Applikation, welche im Mindesten dieselben Funktionen und Features besitzt, wie die aktuelle Applikation
- einer sauberen Dokumentation (SAD, Projektdokumentation), sodass die Applikation vom Businesspartner selbst erweitert und angepasst werden kann
- einer Migration der bestehenden Daten

Resultate

- Lauffähige Software gemäss Spezifikation
- Projekt- und Produktdokumentation
- Dokumente gemäss Vorgaben des Studiengangs.

Auftraggeber

Gema
Marco Schlegel
IT-Leiter Gema

Projektteam

- Christoph Bühler, c1buehle@hsr.ch
- Lorenz Bösch, lboesch@hsr.ch
- Patrick Lötscher, ploetsch@hsr.ch

Betreuung HSR

Hansjörg Huser
hhuser@hsr.ch
Tel: 055 222 49 12 (HSR Raum 6.010)

Projektabwicklung

Termine

- Beginn der Arbeit: Mo., 18. Feb. 2013
- Abgabetermin Kurzfassung/Poster/Mgmt-Summary zum Review: 7.Juni 2013
- Abgabetermin (inkl. Poster): 14. Juni 2013, 12.00 Uhr
- Präsentation der Bachelorarbeiten, 14. Juni 2013 16-20h
- Mündliche BA-Prüfung : 16. August 2013
- Zwischenbesprechung/Review mit Auftraggeber nach Projektplan

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von mind. 360 Stunden pro Student. (2 Wochen zu 45h=90h, 14 Wochen zu 20h=280h)

Hinweise für die Gliederung und Abwicklung des Projektes:

Gliedern Sie Ihre Arbeit in 4 bis 5 Teilschritte. Schliessen Sie jeden Teilschritt mit einem Meilenstein ab. Definieren Sie für jeden Meilenstein, welche Resultate dann vorliegen müssen!

Folgende Teilschritte bzw. Meilensteine sollten Sie in Ihrer Planung vorsehen:

Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen),

- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
  Letzter Meilenstein: Systemtest abgeschlossen
  Termin: ca. eine Woche vor Abgabe
- Entwickeln Sie Ihre SW in einem iterativen, inkrementellen Prozess: Planen Sie möglichst früh einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In die folgenden Phasen können Sie dieses Kernsystem schrittweise ausbauen und testen.
- Falls Sie in Ihrer Arbeit neue oder Ihnen unbekannte Technologien einsetzen, sollten Sie parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Setzen Sie konsequent Unit-Tests ein! Verwalten Sie ihre Software und Dokumente auf einem geeigneten Repository. Stellen Sie sicher, dass der/die Betreuer jederzeit Zugriff auf das Repository haben und dass das Projekt anhand des Repositories jederzeit wiederhergestellt werden kann.
- Achten Sie auf die Einhaltung guter Programmier- und Designprinzipien
  - DRY, high cohesion, loose coupling, etc.
  - Clean Code!
- Halten Sie sich im Übrigen an die Vorgaben aus dem Modul SE-Projekt.

Projektadministration

- Führen Sie ein individuelles Projekttagebuch aus dem ersichtlich wird, welche Arbeiten Sie durchgeführt haben (inkl. Zeitaufwand). Diese Angaben sollten u.a. eine individuelle Beurteilung ermöglichen.
- Dokumentieren Sie Ihre Arbeiten laufend. Legen Sie Ihre Projektdokumentation mit der aktuellen Planung und den Beschreibungen der Arbeitsresultate elektronisch in einem Projektordner ab. Dieser Projektordner sollte jederzeit einsehbar sein (z.B.: svn-Server oder File-Share).

Inhalt der Dokumentation

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

- Dokumente gemäss Vorgabe: https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html
- Aufgabenstellung
- Technischer Bericht
- Projektdokumentation
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.

Fortschrittsbesprechung:

Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt.

Teilnehmer: Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber

Termin: jeweils Dienstag, 15:00 Uhr, Raum 6.010 (Abweichungen werden rechtzeitig kommuniziert)

Alle Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber per E-Mail (spätestens innerhalb von 2 Tagen) zugestellt wird.

Standard-Traktanden sind:

- Was wurde erreicht, was ist geplant, welche Probleme stehen an
- Review von Code/Dokumentation (Abgabe jeweils einen Tag vor dem Meeting

Falls notwendig, können weitere Besprechungen / Diskussionen einberufen werden.

Beurteilung

Gesichtspunkte	**Gewicht**
Organisation, Durchführung	1/5
Berichte (Abstract, Mgmt Summary, techn. und persönliche Berichte) sowie Gliederung, Darstellung, Sprache)	1/5
Inhalt, mündliche Prüfung *)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Rapperswil, 18. Feb. 2013

Hansjörg Huser

Abstract

Für die Gema GmbH soll eine neue Applikation zur Verwaltung der Seriennummern der Pulverbeschichtungsanlagen und deren Komponenten erstellt werden. Diese Seriennummern werden zur eindeutigen Identifikation benötigt, falls Probleme auftreten und ein möglicher Garantiefall daraus resultiert. Der Entschluss, eine neue Applikation zu erstellen, wurde gefasst, da die bereits bestehende Lösung grosse Einschränkungen in Sachen Performance und Wartbarkeit mit sich bringt. Die neue Seriennummernverwaltung soll diese Probleme beheben und die Arbeitsprozesse für die Mitarbeiter vereinfachen und optimieren.

Im Rahmen dieser Bachelorarbeit wurde eine komplett neue, voll funktionsfähige Software für den produktiven Einsatz erstellt, die alle bereits vorhandenen Funktionen der alten Software zur Verfügung stellt. Dazu gehört auch die Datenübernahme aller bereits existierenden Produkte.

Zu den Hauptaufgaben gehört der Import von Aufträgen, Kunden und Adressen, die bereits in einer separaten ERP Lösung Namens „ProConcept“ verwaltet werden. Diese Daten werden alle 10 Minuten mit der Datenbank der Seriennummernverwaltung abgeglichen. Die auf dem .NET Framework basierende ASP.NET Webapplikation, welche neu bei allen Mitarbeitern zum Einsatz kommt, ermöglicht das Erstellen und die Verwaltung einzelner Produkte mit ihrer Seriennummer. Die Möglichkeit zum Massenimport von Produkten wird mittels CSV erreicht. Die für einen Auftrag benötigten Produkte werden über die Webapplikation zugewiesen. Ein auf C# und WPF basierter Barcode Client kommt dort zum Einsatz, wo viele Produkte gleichzeitig einem Auftrag zugewiesen werden müssen.

Mit Hilfe von Crystal Reports können Garantiekarten, Typenschilder und weitere Berichte erstellt werden. Auf den Typenschildern, welche an den Produkten angebracht werden, wird neuerdings ein DataMatrix Code aufgedruckt.

Management Summary

Ausgangslage

Die Gema GmbH produziert Pulverbeschichtungsanlagen, die aus einzelnen Produkten gefertigt werden. Diese Produkte müssen mit einer Seriennummer versehen sein, damit sie eindeutig identifiziert werden können. Eine Identifikation ist dann nötig, wenn Probleme auftreten und ein möglicher Garantiefall daraus resultiert.

Die heute eingesetzte Applikation zur Erfassung dieser Produkte hat jedoch einige Einschränkungen. Eines der grössten Probleme ist die Performance. Die Benutzer mussten teilweise mehrere Sekunden warten, bis eine Aktion durchgeführt wurde. Eine weitere Einschränkung liegt bei der Anpassbarkeit. So können zum Beispiel nicht ohne weiteres neue Attribute einem Produkttyp hinzugefügt werden. Änderungen die mehrere Produktionslinien betreffen, müssen für jede dieser Produktionslinien einzeln durchgeführt werden. Spezialfälle in der Garantieabwicklung können durch die vorhandene Datenstruktur nicht realisiert werden. Ebenfalls mangelt es der aktuellen Applikation an Benutzerfreundlichkeit. Die Prinzipien für das Design einer Benutzeroberfläche werden oftmals verletzt. Bei Änderungen muss die Applikation neu verteilt werden, was der Wartbarkeit nicht zugutekommt.

Aus diesen Gründen wurde entschieden, im Rahmen einer Bachelorarbeit eine neue Applikation zu entwickeln.

Vorgehen

In einem ersten Schritt wurden die funktionalen Anforderungen an die neue Applikation zusammen mit dem Businesspartner erarbeitet. Dabei wurde in erster Linie darauf geachtet, dass alle Funktionen der alten Applikation spezifiziert sind. Zusätzlich wurden neue Anforderungen wie beispielsweise das Scannen von Produkten via Barcode und das Generieren von diesen Barcodes aufgenommen. Durch den Einsatz von Barcodes soll das bisher mühsame Eintippen von einzelnen Produkten und deren Verknüpfung zu Aufträgen eliminiert werden. Ferner wurde ein persönliches Gespräch mit allen Produktionsmitarbeitern geführt, um zu erfahren, welche Änderungen oder Erweiterungen für die neue Applikation sinnvoll wären.

In einem zweiten Schritt wurden Papierprototypen anhand der gesammelten Informationen erstellt. Beim Erstellen dieser Papierprototypen wurden hauptsächlich zwei Ziele verfolgt: zum einen sollten alle alten und neu gewünschten Funktionalitäten implementiert sein, zum anderen sollte das GUI vor allem den Anforderungen in Sachen Benutzerfreundlichkeit, Wartbarkeit und Erweiterbarkeit gerecht werden. Ein wichtiger Schritt dabei war, von den einzelnen Ansichten für jede Produktionslinie wegzukommen und eine einheitliche Eingabemaske für alle Benutzer zu schaffen. Mit Hilfe dieser Papierprototypen wurden die wichtigsten Abläufe mit einigen Benutzern durchgespielt. Das Ziel dabei war, zu erfahren, ob die erarbeiteten GUI Prototypen intuitiv und verständlich sind. Den Benutzern sollte so auch die Möglichkeit gegeben werden, direkten Einfluss auf die Gestaltung der neuen Applikation nehmen zu können.

Nach der Evaluationsphase wurde direkt mit der Implementation eines Prototypen der Webapplikation, des Reportings und des Barcode Scanners begonnen. Durch den damit erreichten architektonischen Durchstich konnten die grössten technischen Risiken bereits in den Anfängen des Projekts minimiert werden. Dazu gehörte, dass keine Erfahrungen im Bereich Reporting und Barcode und nur geringe Erfahrungen mit den eingesetzten Frontendsprachen sowie deren Interoperabilität vorhanden waren.

Auf die erfolgreiche Implementation der Prototypen folgte die eigentliche Entwicklung der Applikation. Parallel zur Entwicklung wurden Tests geschrieben, die zu einer fehlerfreien Implementation beigetragen haben. Nach der Fertigstellung der Applikation, wurde diese beim Businesspartner für einen ersten Integrationstest mit ausgewählten Benutzern eingerichtet. Eine erste Datenübernahme erfolgte im Vorfeld, damit bereits Referenzdaten zur Verfügung standen. Änderungswünsche oder Fehler wurden implementiert respektive behoben und anschliessend erfolgte die Freigabe zum Test für alle Benutzer. Dabei sollten diese die neue Applikation parallel zur alten bedienen und etwaige letzte Fehler eruieren. Abschliessend erfolgte der Abnahmetest beim Kunden auf der produktiven Umgebung.

Ergebnis

Das Ergebnis dieser Bachelorarbeit ist eine voll funktionsfähige Applikation, die von der Gema GmbH produktiv eingesetzt wird. Durch die neue Architektur können früher beklagte Performanceprobleme ausgeschlossen werden.

Zur Hauptanwendung dient die Webapplikation, die bei allen Benutzern zum Einsatz kommt und über die Aufträge und Produkte bearbeitet werden. Die Aufträge inklusive Kundeninformationen werden in einer externen ERP-Software erstellt und verwaltet. Alle 10 Minuten werden diese mit der Datenbank der Seriennummernverwaltung abgeglichen. Zusätzlich werden auftragsspezifische Attribute hinzugefügt, die bearbeitet werden können. Produkte, aus denen später die Pulverbeschichtungsanlagen gefertigt werden, können auf zwei Arten über die Seriennummernverwaltung erfasst werden: entweder werden sie manuell über die Webapplikation erfasst oder per Datei (CSV) eingelesen, die kommagetrennte Werte enthält, wobei ersteres den Normalfall repräsentiert. Eine CSV-Datei kommt momentan bei zwei Produkttypen zu Stande, da alle diese Produkte im Voraus getestet werden und die somit erhaltenen Messdaten zum Erstellen verwendet werden können.

Damit möglichst dynamisch Produkte erstellt werden können, werden diese in Produktkategorien und Produkttypen unterteilt. Ein Produkttyp kann zum Beispiel ein „OptiCenter OC01“ sein, das einer Produktkategorie „Pulversystem“ zugewiesen ist. Diesem Produkttyp können nun dynamisch Produktattribute wie beispielsweise die Frequenz oder eine Druckspanne hinzugefügt oder entfernt werden. Dabei können dieselben Produktattribute für mehrere Produkttypen verwendet werden. Zusätzlich können die einzelnen Produktionslinien verwaltet werden. Dadurch besteht die Möglichkeit, den Produktionslinien Produktkategorien zuzuweisen, indem diesen nur die benötigten Produktkategorien angezeigt werden können.

Geht ein Auftrag in die Produktion, werden diesem die benötigen Produkte über die Seriennummernverwaltung zugewiesen. Es kommt auch vor, dass ein Produkt aus anderen Produkten besteht. Aus diesem Grund können Produkte auch anderen Produkten zugewiesen werden. Bei einigen Produktionslinien müssen sehr viele Produktzuweisungen durchgeführt werden. Damit nicht von Hand alle Produkte einzeln zugewiesen werden müssen, kommt ein separater Barcode Client zum Einsatz. Mit Hilfe eines Barcode Scanners wird zuerst ein Auftrag, der einen DataMatrix Code enthält, eingelesen. Danach werden die Produkte, welche ebenfalls über einen DataMatrix Code verfügen, eingelesen und direkt dem Auftrag zugewiesen. Per „Drag and Drop“ ist es möglich, die Zuweisung zu verändern, bevor man sie speichert.

Durch den Einsatz von Crystal Reports kann die Seriennummernverwaltung Garantiekarten und Typenschilder, welche auf dem Produkt angebracht werden, erstellen. Die Typenschilder besitzen einen Barcode, der die Seriennummer des jeweiligen Produkts beinhaltet.

Eine ausgeklügelte Auftrags- und Produktsuche vereinfacht den Mitarbeitern die Suche nach Aufträgen und Produkten über die Seriennummer oder spezifische Attribute. Zusätzlich wird den Mitarbeitern des Marketings ermöglicht, diverse Auswertungen zu erstellen.

Die Authentifizierung des Benutzers an der Seriennummernverwaltung wird anhand der Windows Authentifikation durchgeführt. Es werden die Login-Daten des an der Domäne angemeldeten Benutzers verwendet. Somit fällt das manuelle Login weg.

Durch die Möglichkeit der dynamischen Produkterstellung können Anpassungen einfach durchgeführt werden, ohne dass die Applikation neu verteilt werden muss. Dies gilt ebenfalls für das dynamische generieren von Reports. Der Einsatz von Barcodes und der Import von CSV-Dateien bringt eine markante Optimierung in den Abläufen mit sich. Die neu designte Benutzeroberfläche entspricht gängigen Designprinzipien und ermöglicht so den Benutzern ein intuitives Arbeiten.

Ausblick

Die Applikation löst per 01. Juli 2013 die bestehende Lösung komplett ab.

Mögliche zukünftige Erweiterungen der Seriennummernverwaltung könnten zum Beispiel sein, dass Kunden der Gema GmbH selber über eine extern aufrufbare Webseite den Garantiestatus eines Produktes abrufen können. Dies wäre relativ einfach realisierbar, da es sich bei der Seriennummernverwaltung bereits um eine webbasierte Applikation handelt.

Durch den Einsatz von DataMatrix Codes hält man sich die Option offen, weitere Informationen als nur die Seriennummer zu speichern. Es wäre möglich, alle Informationen, die zu einem Produkt vorhanden sind, abzuspeichern. Zusätzlich wäre ein Link auf das Handbuch denkbar.

Inhalt

1. Einleitung

Die Seriennummernverwaltung ist eine Software, welche von der Gema GmbH eingesetzt wird. Diese wurde erstellt, um eine effiziente Verwaltung aller von der Gema hergestellten Produkte zu gewährleisten. Die Software wurde nicht nach den gängigen Designprinzipien eines Softwareingenieurs erstellt und hat unter anderem deswegen Probleme mit der Performance. Das Ziel der Bachelorarbeit ist es nun, diese Software neu zu schreiben und diese Probleme zu beseitigen. Zusätzlich sind neue Anforderungen hinzugekommen, welche ebenfalls erfüllt werden müssen.

Die Applikation soll einfach zu Warten sein und die Arbeitsprozesse optimieren. Aus diesem Grund wird eine ASP.NET Webapplikation erstellt, welche die Probleme der alten Software löst. So wird die Rechenlast auf den Webserver ausgelagert und die Funktionalität zentral gesteuert. Es muss nicht mehr die gesamte Applikation neu verteilt werden, wenn neue Features oder Änderungen dazukommen.

2. Grundlagen

2.1. IST-Zustand

Der aktuelle Prozess ist in folgendem Bild beschrieben:

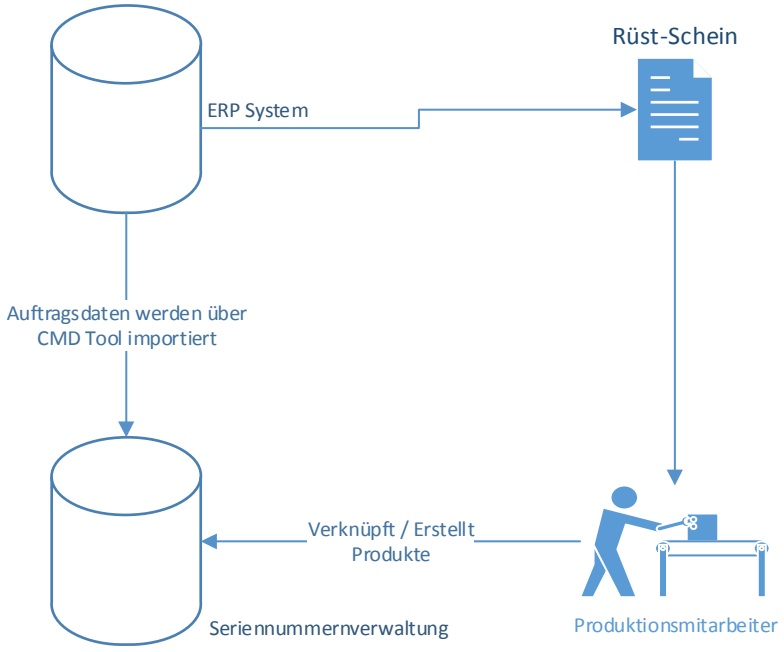


Abbildung 1: IST-Prozess Seriennummernverwaltung

Die Seriennummernverwaltung wird alle 10 Minuten mit neuen Aufträgen proaktiv bestückt. Dies wird mittels eines „scheduled task“ auf dem Datenbankserver der aktuellen Software realisiert. Dieser Task stösst seinerseits ein kleines CMD-Tool an, welches die ERP – View, welche es extra zu diesem Zweck gibt, nach den neuesten Einträgen durchsucht und diese dann in die Datenbank importiert. Das Ganze geschieht über einen OR-Mapper, was Performanceeinbussen mit sich bringt.

Es gibt zwei unterschiedliche Arten von Linien:

1. Produkte werden fortlaufend produziert, quasi auf Vorrat. Im alten System gibt es ein paar wenige Produkte, die nun ohne Auftrag erfasst werden können. Für die Produkte, welche zwar auf Vorrat produziert wurden, jedoch zwingend einen Auftrag benötigen, wurde ein Dummy Auftrag erfasst.
2. Produkte werden nach Rüstschein produziert, haben also einen konkreten Auftrag. Die Mitarbeiter erfassen das Produkt direkt beim entsprechenden Auftrag und verknüpfen allfällige Komponenten (andere Produkte)

Im System ist für die zwei unterschiedlichen Arten jedoch keine Unterscheidung vorhanden.

Hat der Mitarbeiter einen Auftrag abgeschlossen, so kann er Garantiekarten und Typenschilder für seine Produkte drucken lassen. Dies geschieht direkt über den eingetragenen Drucker. Eine Vorschau ist zwar möglich, jedoch nur als PDF, was eine Nachbearbeitung ausschliesst.

2.1.1. Bestehende Lösung

Die bestehende Lösung setzt folgende Technologien ein:

- Microsoft .NET Framework 2.0
- WinForms 2.0 Applikation
- MS SQL Server Express 2005
- DevExpress WinForms Components v.9

Die eingesetzten Technologien sind nicht mehr zeitgemäss. Unteranderem dieser Umstand sorgt für einige Probleme:

- Performanceprobleme aufgrund des aktuellen Datenmodells
  - Pro Produkt separate Tabelle (ca. 58 Tabellen)
- Performanceprobleme aufgrund der älteren Computer (Rechenlast für Clients zu hoch)
- Reporting schwierig, da inkonsistente Datenhaltung betrieben wurde
- Bei Änderungen muss die gesamte Applikation neu verteilt werden
- Designprinzipien (z.B.: DRY[1]) wurden nicht bzw. ungenügend eingehalten
- Hohe Netzwerklast

[1] DRY – Don't repeat yourself

2.2. Soll-Zustand

2.2.1. Prozess

In folgender Grafik ist der für die Seriennummernverwaltung relevante Soll-Prozess grafisch dargestellt:

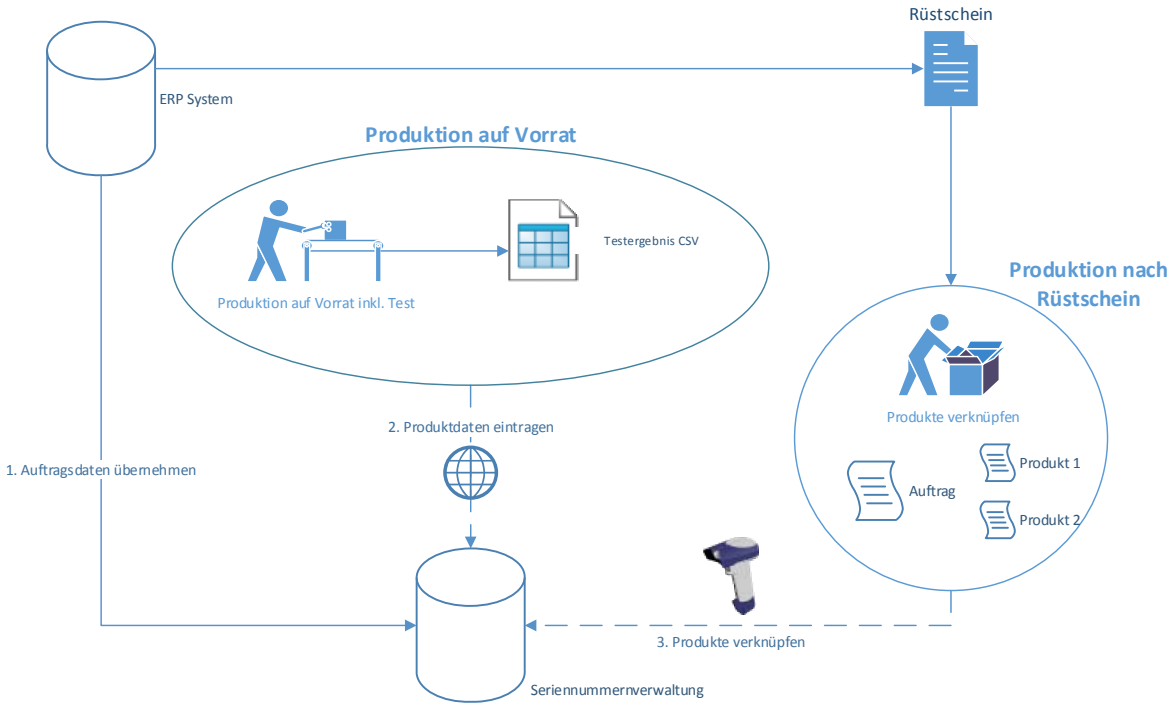


Abbildung 2: Relevanter Geschäftsprozess der Gema GmbH (SOLL-Prozess)

Die Komponenten werden in der Tabelle beschrieben:

Komponente	Beschreibung
ERP System	In der Gema GmbH wird ProConcept als ERP System verwendet. Die ganze Auftragsabwicklung sowie die Fakturierung werden darüber gehandhabt.
Rüstschein	Aus den Aufträgen, die im ERP System verwaltet werden, wird ein Rüstschein für die entsprechenden Linien erstellt.
Seriennummernverwaltung	Ist die neu zu entwickelnde Lösung, welche die Seriennummern der Produkte sowie die Zuordnung von Produkten zu Aufträgen verwaltet.
Produktion auf Vorrat	Die meisten Produkte werden auf Vorrat, also ohne konkreten Auftrag produziert. Danach werden sie einem Test unterzogen. Die Testdaten sowie die Seriennummer des Produkts werden dank einer Teststation in einer CSV-Datei abgelegt. Die Produkte werden danach zwischengelagert, bis Sie für einen effektiven Auftrag benötigt werden.
Produktion nach Rüstschein	Aufgrund eines Rüstscheins werden produzierte Produkte einem Auftrag oder einem anderen Produkt zugeordnet und zur Weiterverarbeitung oder zum Versand bereitgestellt.

Tabelle 1: Beschreibung der Komponenten

Der Prozessablauf sieht folgendermassen aus:

1. **Auftragsdaten übernehmen**: Die Auftragsdaten, die im ERP System enthalten sind, werden in die Seriennummernverwaltung gespiegelt.
2. **Produktdaten eintragen**: Die Linien-Mitarbeiter pflegen das generierte CSV mit den Testdaten und der Seriennummer in die Seriennummernverwaltung. Sie greifen mit dem Browser über eine Webschnittstelle zu und laden die CSV-Datei hoch. Falls keine CSV-Datei generiert wurde, erfassen sie sämtliche Produkte von Hand.
3. **Produkte verknüpfen**: Wenn Produkte untereinander oder aber mit einem Auftrag verknüpft werden müssen, wird dies ebenfalls in die Seriennummernverwaltung eingetragen. Die Linienmitarbeiter haben dazu zwei Möglichkeiten:
    a. Die Verknüpfung wird ebenfalls via Browser eingetragen. Es ist auch möglich, komplett neue Produkte direkt einem Auftrag zuzuweisen.
    b. Sie können den Auftragsbarcode und anschliessend die untergeordneten Produkte anhand eines Bar- oder QR-Codes scannen und die Verknüpfungen danach hochladen.

2.2.2. Software

Neu soll die Software folgende Kriterien erfüllen:

- Einheitlich und korrekt gestalteter Code (Schichtenarchitektur)
- Aktuelle Technologien (.NET 4.5, Entity Framework) einsetzen
- Einheitliche und aufgeräumte Datenstruktur
- Einfaches Reporting möglich
- Rechenlast umverteilen auf die Server, anstatt auf den Clients

Alles in allem muss die Software den gängigen Designprinzipien Rechnung tragen. Die Wartbarkeit soll ebenfalls erhöht werden. So soll die Applikation nicht mehr an zig Rechner verteilt werden müssen, sondern soll auf einem Server als Webapplikation zur Verfügung stehen.

2.2.3. Einschränkungen

Es wurde vermehrt der Wunsch geäussert, die Seriennummernverwaltung mehr mit dem ERP-System zu koppeln. Dadurch könnte beispielsweise überprüft werden, ob einem Auftrag die richtigen Produkte in der richtigen Anzahl hinzugefügt wurden. Fehler bei der Erfassung könnten so minimiert werden. Diesem Wunsch konnte leider nicht nachgekommen werden, da der Zeitaufwand für die Realisierung der Schnittstelle zum ERP-System zu hoch gewesen wäre beziehungsweise keine Ressourcen dafür freigemacht werden können.

3. Anforderungsspezifikationen

3.1. Allgemeines

3.1.1. Produktperspektive

Die aktuelle Software ist ein „Fat-Client“, welcher auf den Computern der Produktionslinien und betroffener Personen läuft. Diese Art der Software erschwert ein Deployment und bietet keine plattformübergreifende Nutzungsmöglichkeit. Ein weiteres Problem stellt die Performance dar, da die Produktionscomputer meist nicht mit den komplexeren Rechenaufgaben klar kommen. Mit dem webbasierten Ansatz der „Seriennummernverwaltung Gema GmbH“ wird das Deployment vereinfacht und die Rechenlast auf einen Server verlegt.

3.1.2. Produktfunktion

Client:

- Anzeigen der Daten
- CRUD Operationen durchführen

Webserver (IIS):

- Verwaltung der Daten
- Verwaltung der Clients
- Schnittstelle zwischen Clients und Daten

Datenbankserver (MSSQL):

- Datenhaltung
- Eventuelle Aufbereitung durch Reports und / oder Views

3.1.3. Annahmen

Technisch:

- Skalierbarkeit des Servers ist nicht nötig, da die Anzahl der Benutzer vorerst nicht mehr als 100 betragen wird. Wird die Webplattform später für Kunden veröffentlicht, so kann die Skalierbarkeit mittels mehreren synchronisierten IIS Webservern realisiert werden
- Die Skalierbarkeit des MSSQL Servers kann mittels DB-Replikation bewerkstelligt werden
- Internet Explorer 8 ist im Minimum vorhanden
- Windows 7 ist im Minimum vorhanden

3.1.4. Abhängigkeiten

- Windows Server 2008 R2 Umgebung
- IIS 7
- .NET Framework 4.5
- Internet Explorer 8 (min.)
- MSSQL Datenbankserver

3.2. UseCases

3.2.1. Webapplikation

3.2.1.1. UseCase Diagramm

Abbildung 3: UseCase Diagramm Webapplikation

3.2.1.2. Actors & Stakeholders

Administrator

Der Administrator stellt die EDV Abteilung der Gema GmbH dar. Er hat alle Möglichkeiten im System. Er kann Reports erstellen und im System hinterlegen. Diese können dann von den Benutzern generiert und ausgedruckt werden. Eine weitere wichtige Funktion des Administrators ist das Erstellen von Produkttypen. Diese sorgen für die nötige Flexibilität bei der Produktspeicherung.

Benutzer

Der Benutzer ist der Mitarbeiter einer Produktionslinie. Er kann die Daten anzeigen und neue Produkte in seiner Produktionslinie erfassen und verwalten. Mittels eines Massenimports können mehrere Produkte auf einmal in das System eingetragen werden.

System
Das System ist der Webserver mit dem IIS System. Das System liefert die Webapplikation aus und sorgt dafür, dass neue Aufträge, welche im ProConcept eingetragen werden, in das System eingetragen und synchronisiert werden.

3.2.1.3. Beschreibung „Brief“

Attribute verwalten
Der Administrator kann von Produkttypen benötigte Attribute erstellen, auslesen und modifizieren. Ein Löschen der Attribute ist nicht erlaubt, da dies die Integrität der Daten verletzen könnte. Die Attribute sind spezielle Werte, welche von einem bestimmten Produkttyp verwendet werden.

Produkttyp verwalten
Der Administrator hat die Möglichkeit, die Produkttypen zu verwalten. Dies beinhaltet Erstellen, Auslesen und Modifikation. Wegen der Datenintegrität der Datenbank kann keine Löschung vorgenommen werden. Die Produkttypen sind einer Produktkategorie zugeordnet, welche für die Produktionslinien von Belang sind. Die Produkttypen werden für eine bestimmte Kategorie verwendet und klassifizieren so ein Produkt.

Reports erstellen
Um ein generisches Reporting zu gewährleisten, kann der Administrator neue Reports erstellen und im System registrieren. Diese Reports können dann von Benutzern generiert (ausgeführt) werden und in einem anderen Schritt gedruckt und genutzt werden. Der Administrator nutzt dafür die im „Detailkonzept Reporting“ vorgestellte Methode, um die Reports zu erstellen.

Aufträge importieren
Das System kann mittels eines Cronjob oder einer ähnlichen Technologie die neuen Auftragsnummern mit Kundeninformationen aus dem ProConcept importieren. Es werden die neusten Aufträge hinzugefügt und die bestehenden auf Änderungen überprüft.

Dazu werden die Auftragsnummer und Zusatzinformationen (Industriecode, Kunde, etc.) aus einer definierten View ausgelesen und danach in der MSSQL Datenbank eingefügt.

Frachtbrief / Etiketten drucken
Ein Benutzer hat die Möglichkeit, für bestimmte Aufträge (z.B.: Linie 1 Produktionsaufträge) einen Frachtbrief und / oder eine Frachtetikette auszudrucken. Diese müssen vor dem Druck bearbeitbar sein. Dies ermöglicht den Benutzern, spezielle Lieferadressen einzutragen.

Produkte verwalten
Der Benutzer kann neue Produkte erfassen, bearbeiten und löschen. Dies ist die Kernfunktion für den Benutzer. Ein Benutzer hat eine Übersicht mit allen von der Produktionslinie eingetragenen Produkten und kann in dieser Maske die einzelnen Produkte verwalten.

Produkt mit Auftrag verknüpfen
Nicht alle Produkte werden bei der Erfassung im System gleich einem Auftrag zugeordnet. Aus diesem Grund hat der Benutzer nachträglich noch die Möglichkeit, ein Produkt mit einem Auftrag zu verknüpfen. Er wählt dazu den Auftrag aus und fügt diesem alle gewünschten Produkte hinzu.

Seriennummer suchen
Die einzelnen Benutzer (Produktion, Marketing, etc.) können über eine Suchmaske eine Seriennummer suchen. Eine erweiterte Suchmaske erlaubt das Suchen nach Attributen und Typen.

Massenimport von Produkten

Der Benutzer hat die Möglichkeit, mittels einer Technik (Barcode Scanner oder Massenimport über Excel) mehrere Produkte hinzuzufügen. Dies stellt in gewissen Produktionslinien ein wichtiger UseCase dar, da das Eintragen dieser Menge von Produkten viel Arbeitszeit verschlingt.

Typenschild drucken

Der Benutzer einer Produktionslinie kann über den Webserver ein Typenschild für ein Produkt ausdrucken. Diese Typenschilder enthalten wichtige Informationen über das hergestellte Gerät (z.B.: Stromverbrauch, Spannung, Luftverbrauch, etc.) und gewisse CE und Konformitätssymbole. Die Typenschilder werden von einem Thermaldrucker ausgedruckt, welcher seinen Auftrag direkt vom System erhält.

Garantiekarte drucken

Zur Fertigstellung eines Auftrages ist es unabdingbar, dass eine Garantiekarte ausgedruckt wird. Auf dieser Garantiekarte sind sämtliche dem Auftrag zugeordneten Produkte, mit eventuellen Unterprodukten, mit Seriennummer und Typenbezeichnung aufgelistet. Die Karte wird mit dem Reportingsystem („Detailkonzept Reporting“) generiert und ausgedruckt.

Report generieren

Der Benutzer kann spezielle (linienspezifische) Reports generieren lassen. Diese Reports wurden zuvor vom Administrator erstellt und im System hinterlegt. Diese Reports können spezifische Garantiekarten, Typenschilder oder auch Frachtetiketten und Frachtlisten sein.

Barcode generieren

Während der Benutzer die Typenschilder drucken lässt, besteht die Möglichkeit, einen Barcode zu generieren. Dieser Barcode (normaler Barcode oder QR-Code) enthält sämtliche produktspezifischen Informationen und kann dem Kunden helfen, digital auf diese Daten zuzugreifen. Dieser Barcode ist auf den Typenschildern enthalten.

Exportieren nach CSV

Bei der Auftrags-/Produktsuche besteht die Möglichkeit, das Suchresultat als CSV (Comma-separated values) Datei zu speichern. Aus Performancegründen zeigt die Suche maximal 1000 Einträge an. In der exportierten CSV Datei sind jedoch alle Suchresultate enthalten.

Produktkategorie (CRUD)

Im Administrationsmenü besitzen die Administratoren die Möglichkeit, die verschiedenen Kategorien („Pistole“, „Handgerät“, „Kabine“, etc.) zu verwalten. Es können neue erstellt werden und bestehende modifiziert werden. Die Kategorien können auch gelöscht werden, jedoch nur, wenn sie keinem Typen zugewiesen sind.

Produktionslinie (CRUD)

Die Administratoren haben über das entsprechende Menü Zugang zur Linienverwaltung. Die Produktionslinien dienen der Aufteilung der Produktkategorien. Ein Mitglied einer Produktionslinie erhält beim Erstellen eines Produktes nur diejenigen Kategorien, welcher es zugewiesen ist. Auch hier ist Erstellen, Modifizieren und Löschen möglich. Analog zu den anderen, datenbankgespeicherten Objekten kann nur gelöscht werden, wenn keine Abhängigkeiten mehr bestehen.

3.2.1.4. Beschreibung „Fully Dressed“

Produkte verwalten

Der Use Case „Prudukte verwalten“ kann noch weiter unterteilt werden. Im Folgenden wird nur der Use Case „Produkt erfassen“ genauer beschrieben, da bearbeiten und löschen keine spezielle Funktionalität benötigen:

Use Case Name	Produkt erfassen
Primary Actor	Benutzer
Stakeholders and Interests	Benutzer: • Möchte einem Auftrag ein neues Produkt zuweisen
Preconditions	• Person ist eingeloggt und im System als Benutzer autorisiert. • Es existiert mindestens ein Produkttyp im System
Postconditions	• Produkt wurde erfolgreich gespeichert • Seriennummer ist im System eingetragen und dem Produkt zugeordnet
Basic Scenario	1. Der Benutzer wählt in der Produktübersicht die Funktion, um ein neues Produkt zu erfassen 2. Das System zeigt in einer Liste die erfassten Produkttypen an. 3. Der Benutzer wählt den gewünschten Produkttypen aus. 4. Das System zeigt dem Benutzer die Erfassungsmaske für diesen spezifischen Produkttyp an und füllt das Seriennummernfeld mit einem Vorschlag ab. 5. Der Benutzer erfasst die allgemeinen Produktinformationen: Seriennummer, Beschreibung 6. Der Benutzer erfasst die zusätzlichen Felder und speichert das Produkt 7. Das System speichert das Produkt und zeigt eine Info-Meldung an
Erweiterung	5.a Der Benutzer wählt einen neuen Produkttypen aus 1. Weiter bei Schritt 5
Mögliche Fehlerfälle	6. a) Ein Feld, welches im Produkttyp als Muss definiert ist, wurde leer gelassen -> Eine Fehlermeldung wird angezeigt b) In einem Feld ist die Validierung mittels Regex, welcher im Attribut festgelegt wurde, fehlgeschlagen -> Eine Fehlermeldung wird angezeigt
Spezielle Anforderungen	Keine
Eintrittshäufigkeit	Je nach Produktionslinie zwischen 10 und 50 Mal pro Tag

Tabelle 2: „Fully Dressed“ Beschreibung für UseCase „Produkt erfassen“

3.2.2. Barcode Client

3.2.2.1. UseCase Diagramm

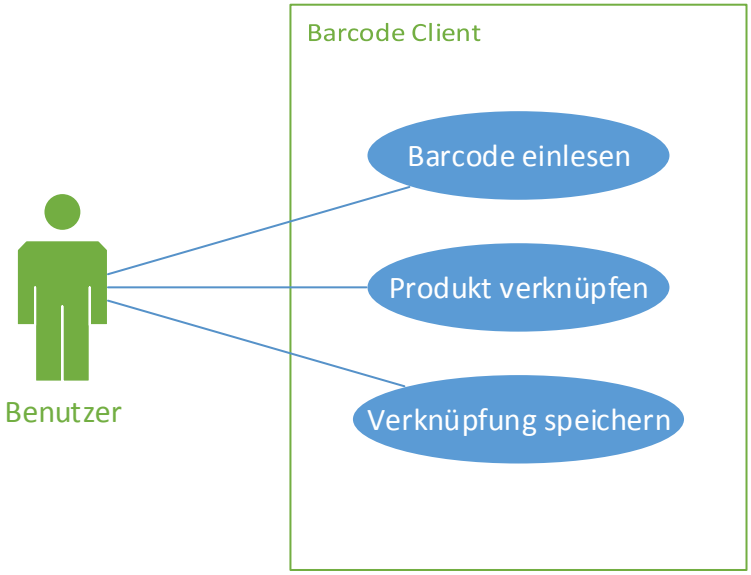


Abbildung 4: UseCase Diagramm Barcode Client

3.2.2.2. Actors & Stakeholder

Benutzer

Der Benutzer ist ein Mitarbeiter aus der Produktion, der einen Barcode Scanner zur Verfügung hat. Mit Hilfe des Barcode Scanners können mehrere Produkte auf einmal in das System eingetragen und untereinander verknüpft werden.

3.2.2.3. Beschreibung „Brief“

Barcode einlesen

Der Benutzer liest mit dem Barcode Scanner den Barcode eines Auftrags oder Produkts ein. Das Produkt muss bereits im System vorhanden, darf aber noch nicht verknüpft sein. Produkte werden automatisch dem aktiven Auftrag zugewiesen.

Produkt verknüpfen

Der Benutzer hat die Möglichkeit, ein Produkt mit einem Auftrag oder mit einem anderen Produkt manuell neu zu verknüpfen. Dies ist vor allem dann nötig, wenn zum Beispiel ein Produkt falsch zugewiesen wurde beim Einlesen oder wenn ein Produkt einem anderen Produkt als Komponente hinzugefügt werden soll.

Verknüpfung speichern

Damit die getätigten Verknüpfungen übernommen werden, müssen sie gespeichert werden. Dazu müssen die entsprechenden Produkte bearbeitet werden.

3.3. Paper Prototyping

3.3.1. Wireframes

3.3.1.1. Auftragssuche

In der Auftragssuche soll dem Benutzer, wie der Name schon sagt, die Möglichkeit geboten werden, nach einem oder mehreren Aufträgen zu suchen. Dies kann über die Auftragsnummer oder später definierten Attributen wie z.B. Kunde, Projektname, Land, Sprache, Industriecode, etc. geschehen.

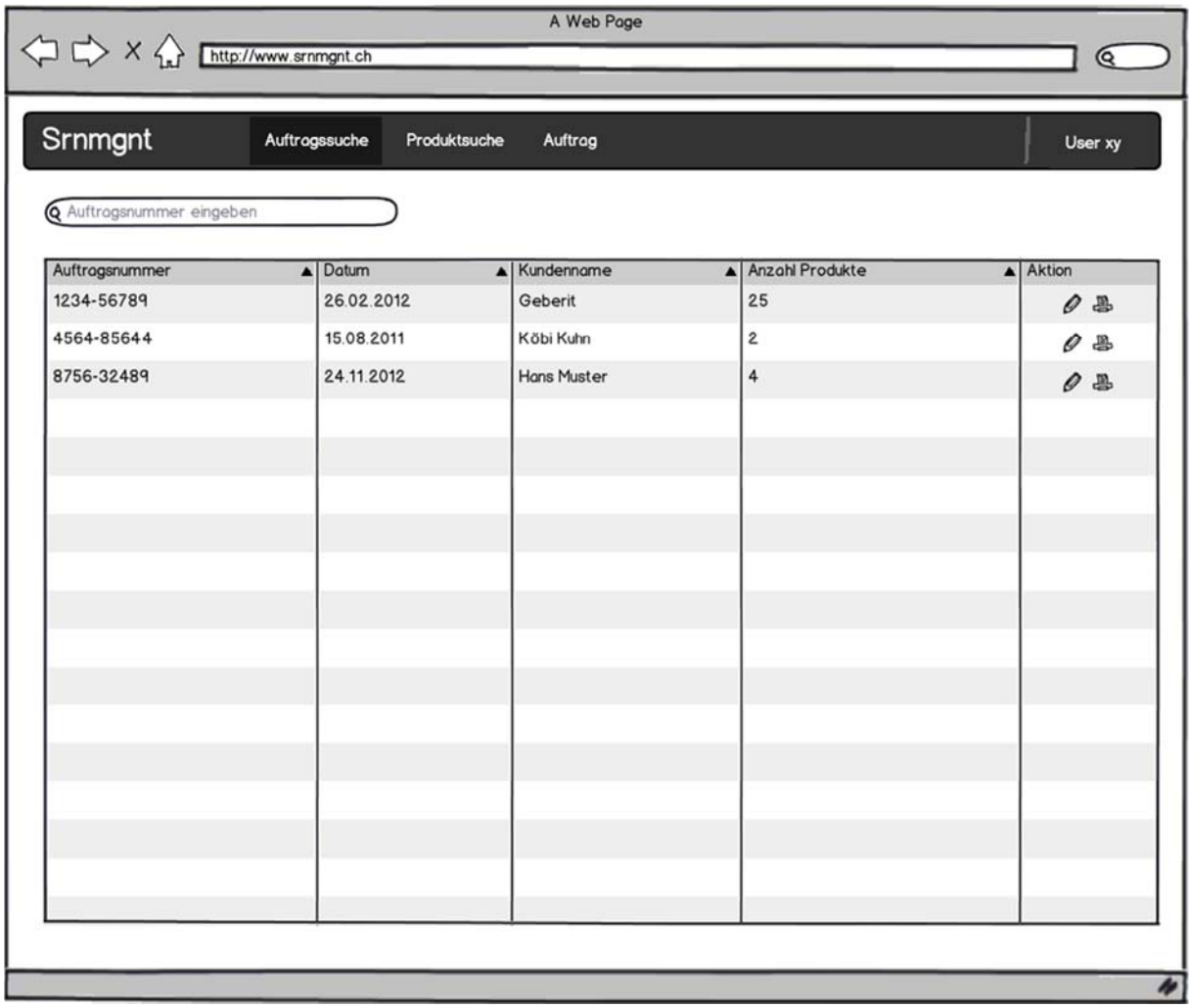


Abbildung 5: Wireframe Auftragssuche

3.3.1.2. Produktsuche

In der Produktsuche kann nach spezifischen Produkten anhand der Seriennummer gesucht werden. Zusätzlich soll auch die Möglichkeit bestehen, nach einer Kategorie, einem Produkttyp oder einem oder mehreren Attributen suchen zu können.

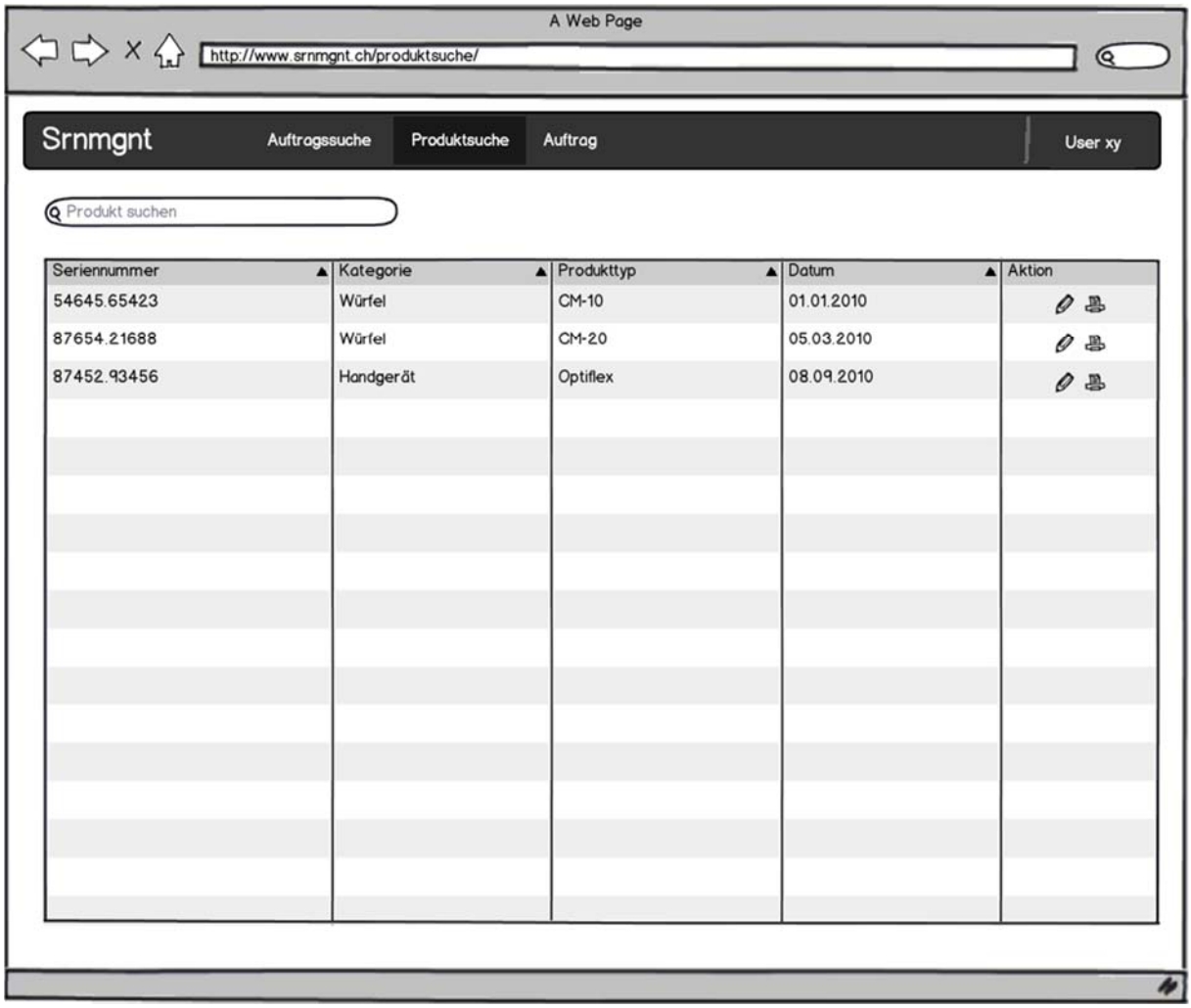


Abbildung 6: Wireframe Produktsuche

3.3.1.3. Auftragsübersicht

Die Auftragsübersicht zeigt alle relevanten Informationen zu einem Auftrag an. Dazu muss zuerst die Auftragsnummer eingegeben werden. Über die Auftragssuche gelangt man ebenfalls auf die Auftragsübersicht. Die Informationen zum Kunden und die zugewiesenen Produkte sind hier auf einen Blick ersichtlich. Eine wichtige Funktion ist das Hinzufügen von Produkten. Dabei kann zwischen einem neuen Produkt, einem bereits erfassten Produkt oder einem Occasionsprodukt gewählt werden. Garantiekarten zum Auftrag können über den dafür vorgesehenen Button ausgedruckt werden.

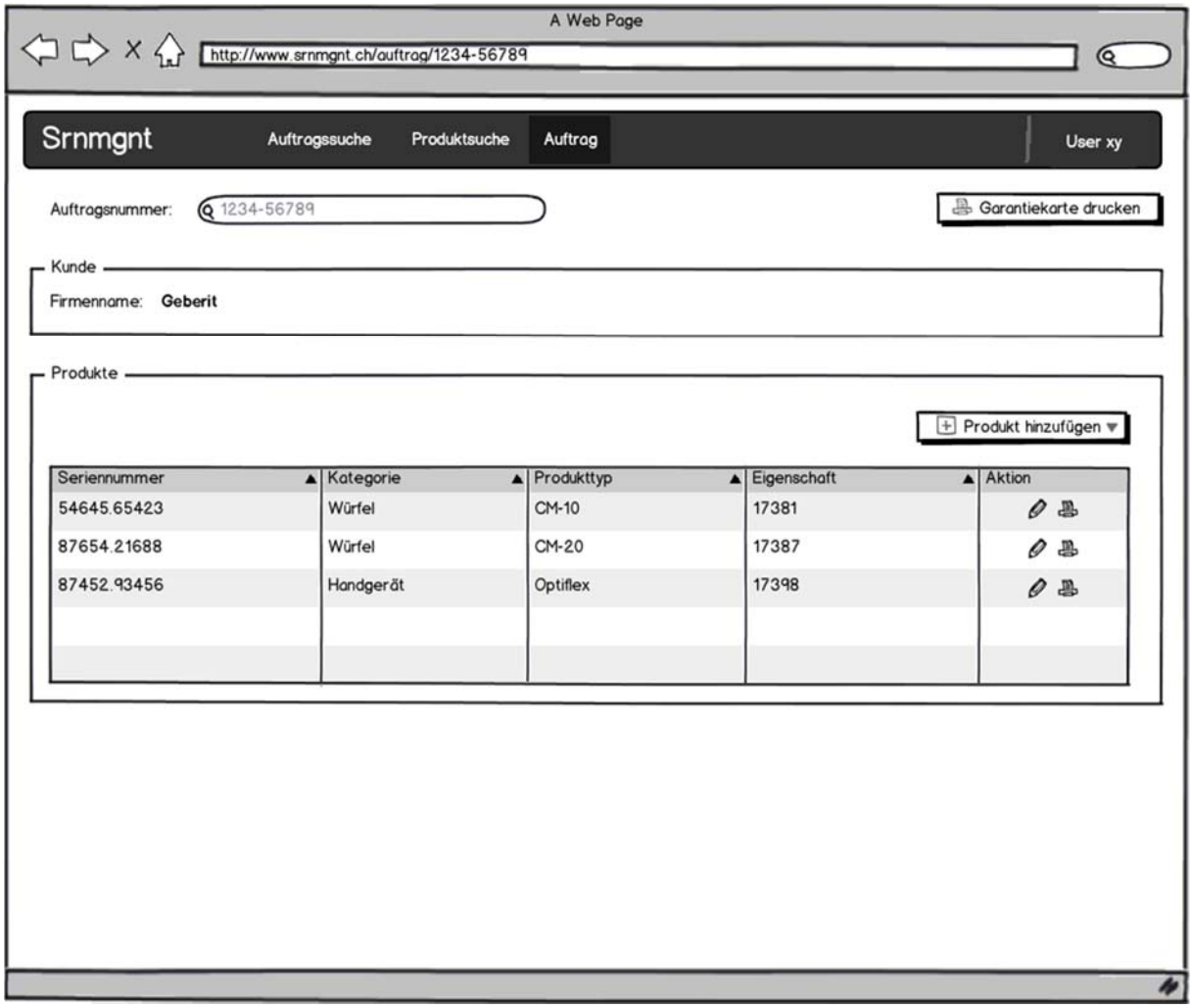


Abbildung 7: Wireframe Auftragsübersicht

3.3.1.4. Erweiterte Auftragsübersicht

Zusätzlich zur normalen Auftragsübersicht sollen auftragsrelevante Daten für gewisse Benutzergruppen, wie zum Beispiel das EAB, angezeigt werden. Um auf diese Ansicht zu gelangen, muss sich der Benutzer entweder in der Gruppe der Administratoren oder Jobinformation befinden.

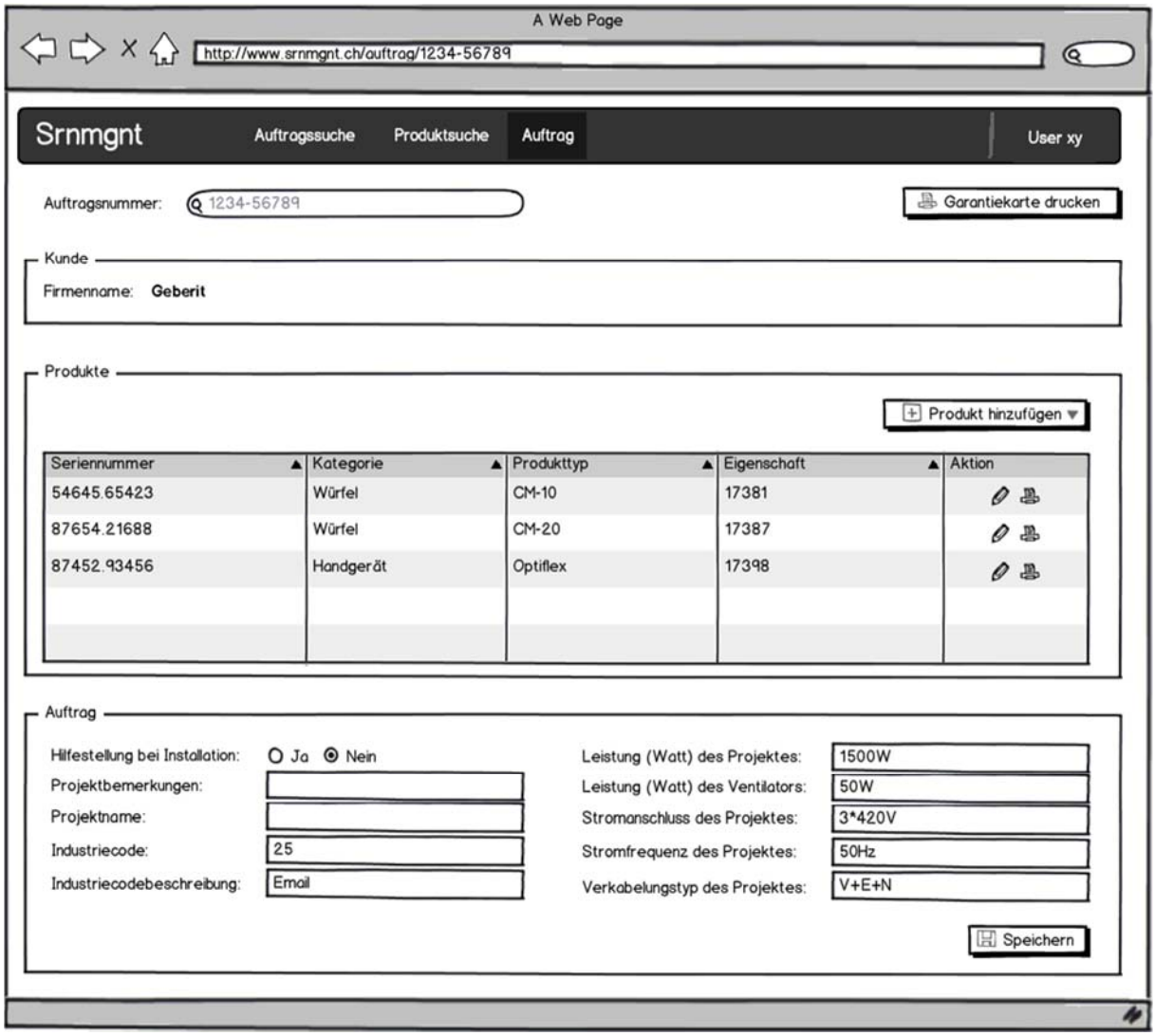


Abbildung 8: Wireframe erweiterte Auftragsübersicht

3.3.1.5. Produktdetails

Wenn ein Produkt in der Auftragsübersicht angewählt wird oder man das Produkt in der Produktsuche auswählt, gelangt man auf die Produktübersicht. Anhand der Breadcrumbs ist ersichtlich, welchem Auftrag das Produkt zugeordnet ist, sofern das Produkt bereits zugeordnet wurde. Alle Informationen zum Produkt werden auf dieser Ansicht angezeigt und können bis auf die Seriennummer und den Produkttyp bearbeitet werden. Die Produkte, die diesem Produkt zugeordnet sind, sind als Komponenten ersichtlich. Dem Produkt kann ein neues Produkt, ein bereits bestehendes Produkt oder ein Occasionsprodukt hinzugefügt werden. Wird ein neues Produkt hinzugefügt, gelangt man ebenfalls auf diese Seite, mit dem Unterschied, dass alle Felder leer sind.

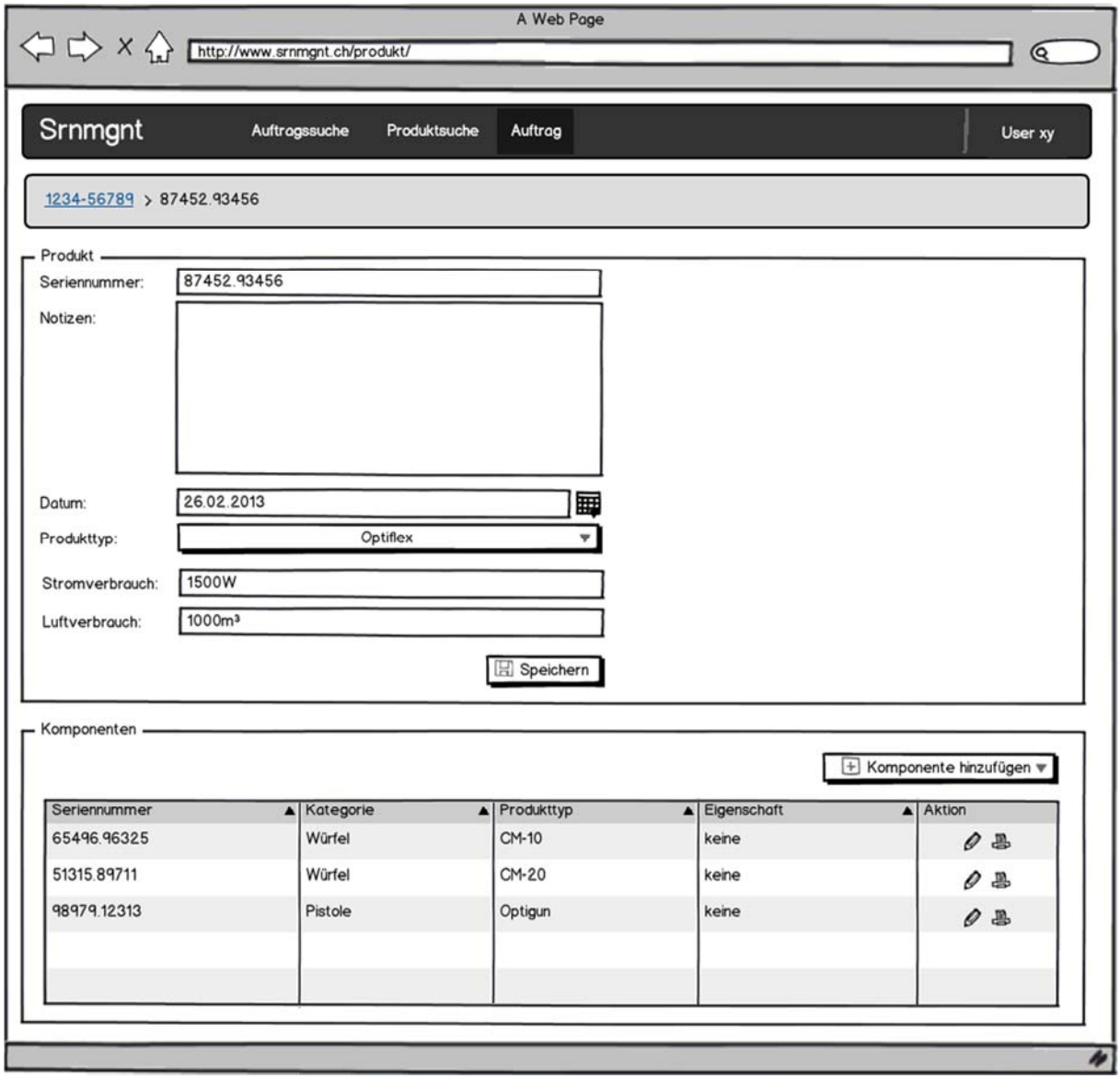


Abbildung 9: Wireframe Produktdetails

3.3.1.6. Produktkomponente

Hierbei handelt es sich um die Produktübersicht einer Komponente. Für jedes Produkt besteht dasselbe Layout. Dass es sich um eine Komponente handelt, ist anhand der Breadcrumbs ersichtlich. Durch dieses Design ist es möglich, eine beliebige Tiefe in der Verschachtelung zu erreichen.

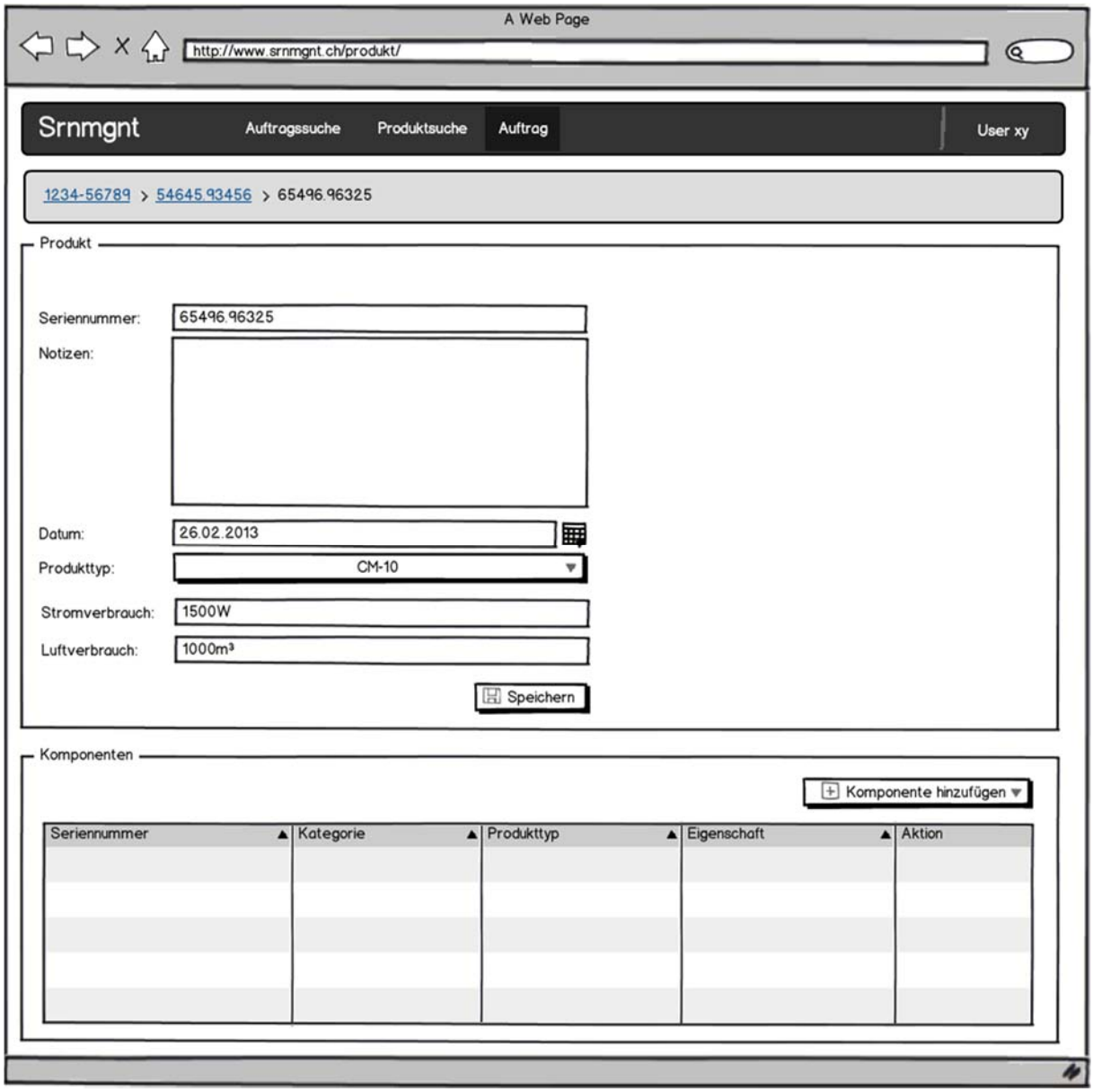


Abbildung 10: Wireframe Produktkomponente

3.3.1.7. Bestehendes Produkt hinzufügen

Möchte man ein bestehendes Produkt einem Auftrag oder einem anderen Produkt als Komponente hinzufügen, wird ein Overlay geöffnet. Die verfügbaren, noch nicht zugewiesenen Produkte werden auf der linken Seite angezeigt. Die zu verknüpfenden Produkte können nun ausgewählt auf die rechte Seite verschoben werden. Beim Speichern werden die ausgewählten Produkte dem Auftrag oder dem Produkt hinzugefügt. Das Overlay wird nach dem Speichervorgang wieder geschlossen.

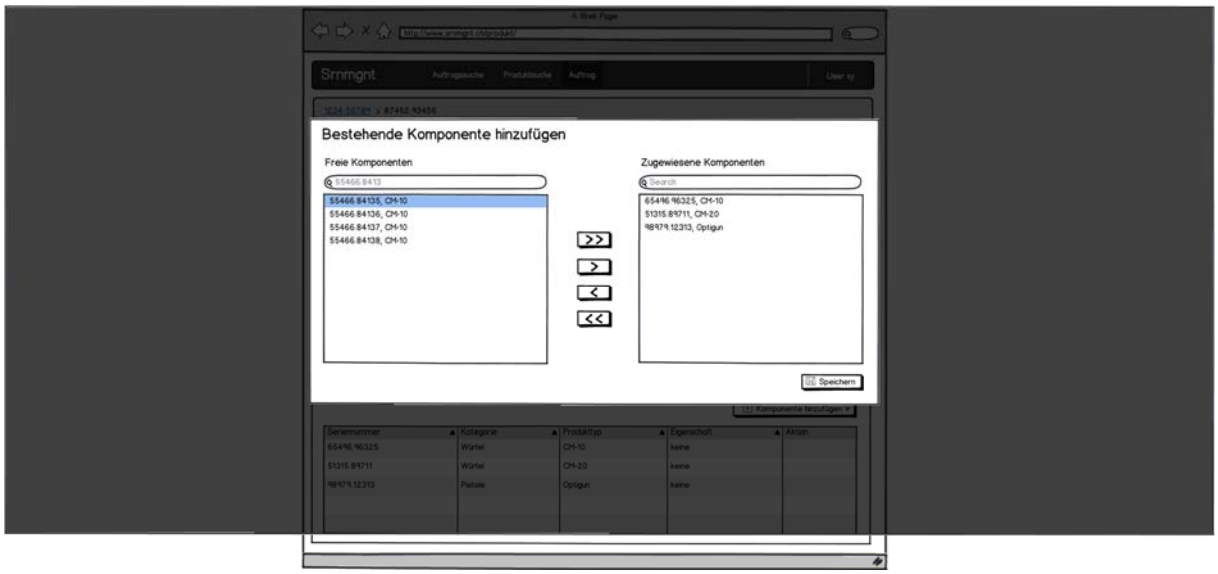


Abbildung 11: Wireframe bestehendes Produkt hinzufügen

3.3.1.8. Barcodeübersicht

Der separate Barcode Client zeigt in dieser Übersicht alle bereits eingescannten Produkte an. Hierbei handelt es sich um Produkte, die in der Datenbank noch nicht vorhanden sind. Im Barcode sind alle Informationen zum Erstellen des Produkts vorhanden. Über den „Starten“ Button soll der Scanvorgang gestartet werden. Der „Abschliessen“ Button soll den Speichervorgang auf der Datenbank durchführen.

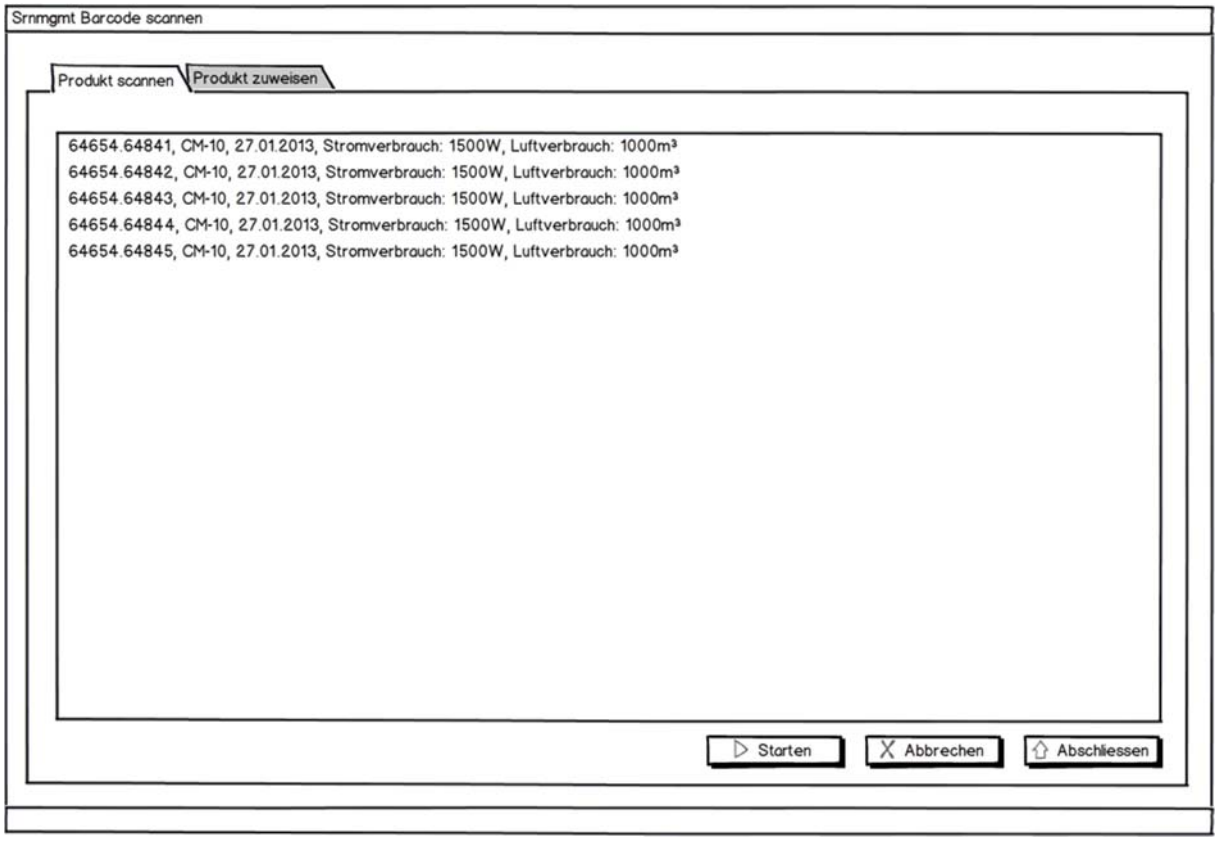


Abbildung 12: Wireframe Barcodeübersicht

3.3.1.9. Barcodezuweisung

In der Produktzuweisung des Barcode Clients sollen bereits im System vorhandene Produkte einfach mit dem Barcode Scanner verknüpft werden können. Dazu muss zuerst der Auftrag eingescannt werden. Nachfolgende, eingescannte Produkte werden diesem Auftrag hinzugefügt. Beim einmaligen Scan eines Steuerungscodes ist es zusätzlich möglich, einem Produkt eine Komponente hinzuzufügen. Die eingescannten Produkte werden solange als Komponente eines Produkts hinzugefügt, bis der Steuerungscode nochmals gescannt wird. Die nächsten Produkte werden somit wieder dem Auftrag hinzugefügt. Sobald ein neuer Auftrag eingescannt wird, werden alle neu gescannten Produkte diesem neuen Auftrag zugeordnet. Zusätzlich ist es möglich, per „Drag and Drop“ die einzelnen Produkte neu zuzuweisen. Somit können mögliche Fehler beim Scanner einfach behoben werden, ohne dass ein erneuter Scann nötig wäre.

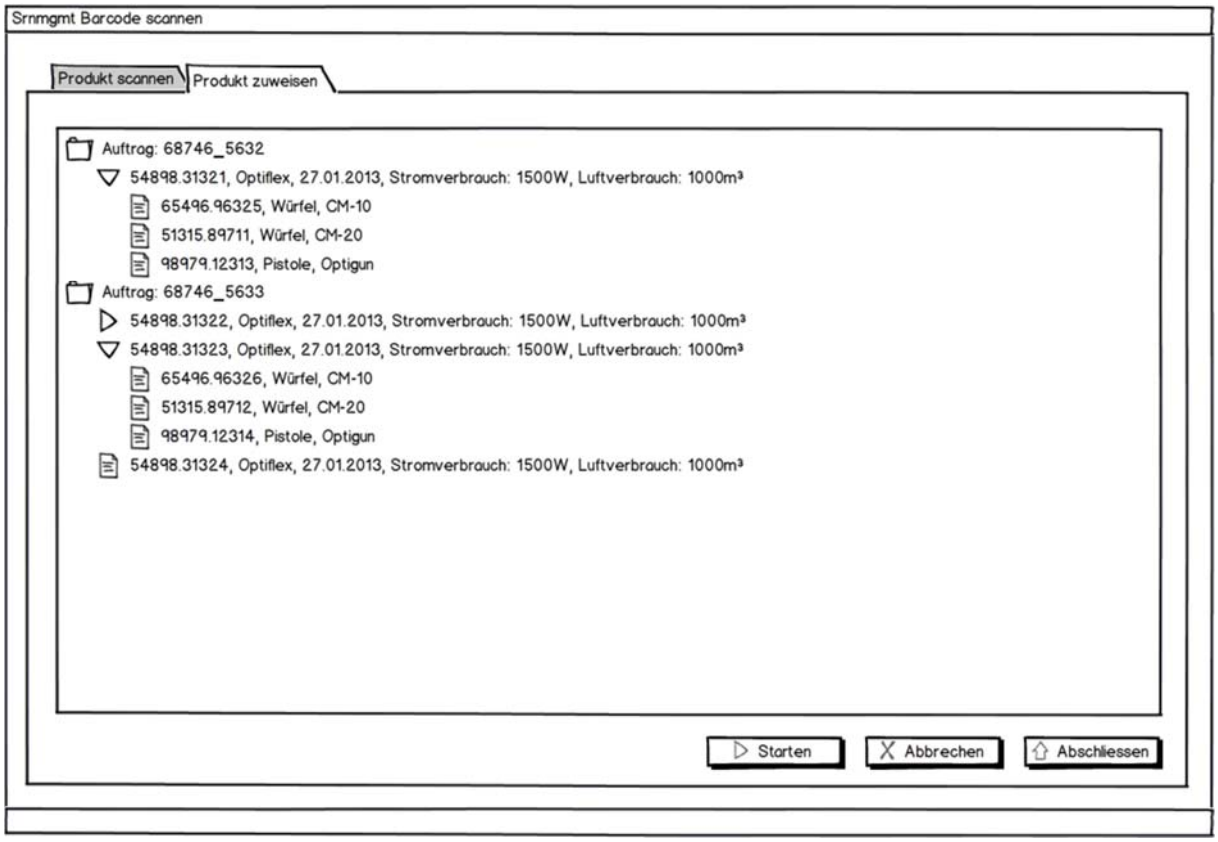


Abbildung 13: Wireframe Barcodezuweisung

3.3.2. Findings

3.3.2.1. Testsubjekt 1

- Produkttyp zuoberst auswählen
- Unterkomponenten werden nur von Zeit zu Zeit eingefügt
- Wie findet man Produkte ohne Seriennummer?
- „Typenschilder drucken“ – Button fehlt
- Typenschilder: Nur einige anzeigen, nicht alle
- Produkte ohne Auftrag eintragen

3.3.2.2. Testsubjekt 2

- Was hat der Kunde bestellt? Das wäre wichtig bei der Auftragsübersicht zu sehen. (Schnittstelle ProConcept)
- 20 Geräte eintragen: Typ auswählen, Anzahl (mit automatischer Seriennummernvergabe)
- Etiketten müssen bearbeitbar sein
- (Wenn die Seriennummer nicht eindeutig ist, reicht eine Warnung)

3.3.2.3. Testsubjekt 3

- Verknüpfung mit ProConcept wäre sinnvoll
- Verknüpfung von Komponenten zu Auftrag sollte ersichtlich sein
- Produkte auf Service buchen -> Es gibt keinen Service-Auftrag, nur eine Rechnung
    - Eventuell Rechnungsnummer als Auftragsnummer
- Produkt, das zur Reparatur zurück kommt und repariert wurde, soll nicht aus dem alten Auftrag gelöscht und beim neuen Auftrag hinzugefügt werden, sondern inaktiv gesetzt werden können, damit ein Tracking möglich ist.
- Datenleichen sollen abgeschrieben werden oder als nächstes versendet werden können
    - Zum Beispiel wenn eine Pistole mehr als 3 Monate im Lager ist, soll eine Warnung angezeigt werden
- Im Ersatzteillager geht es zur Zeit nicht, eine Garantiekarte zu drucken
- Speichern Button ganz am Schluss anzeigen
- Bei Hauptkomponenten alle Details anzeigen
    - Gesamte Produkthierarchie anzeigen
- Aus ProConcept Artikelnummer übernehmen

3.4. Nichtfunktionale Anforderungen

3.4.1. Mengenanforderungen

3.4.1.1. Datenhaltung

Es sollen minimal 1‘000‘000 Produkt „Records“ in der Tabelle verwaltet werden können. Die Tabelle mit den Attributwerten der verschiedenen Produkttypen soll ebenfalls Einträge im Millionenbereich enthalten können. Dies wird durch den MS SQL Server gewährleistet.

3.4.1.2. Client – Server Kommunikation

Es sollen mindestens 500 Clients gleichzeitig verarbeitet werden können. Aktuell werden ca. 20-30 Clients genutzt, bis zu einer geschätzten maximalen Anzahl von 70 Clients. Der IIS von Microsoft kann ohne Probleme diese Anzahl Clients mit der Applikation versorgen. Ist eine Skalierung notwendig, so kann dies mittels Loadbalancing realisiert werden.

4. Analyse

4.1. Applikationsdomain

4.1.1. Domainmodell

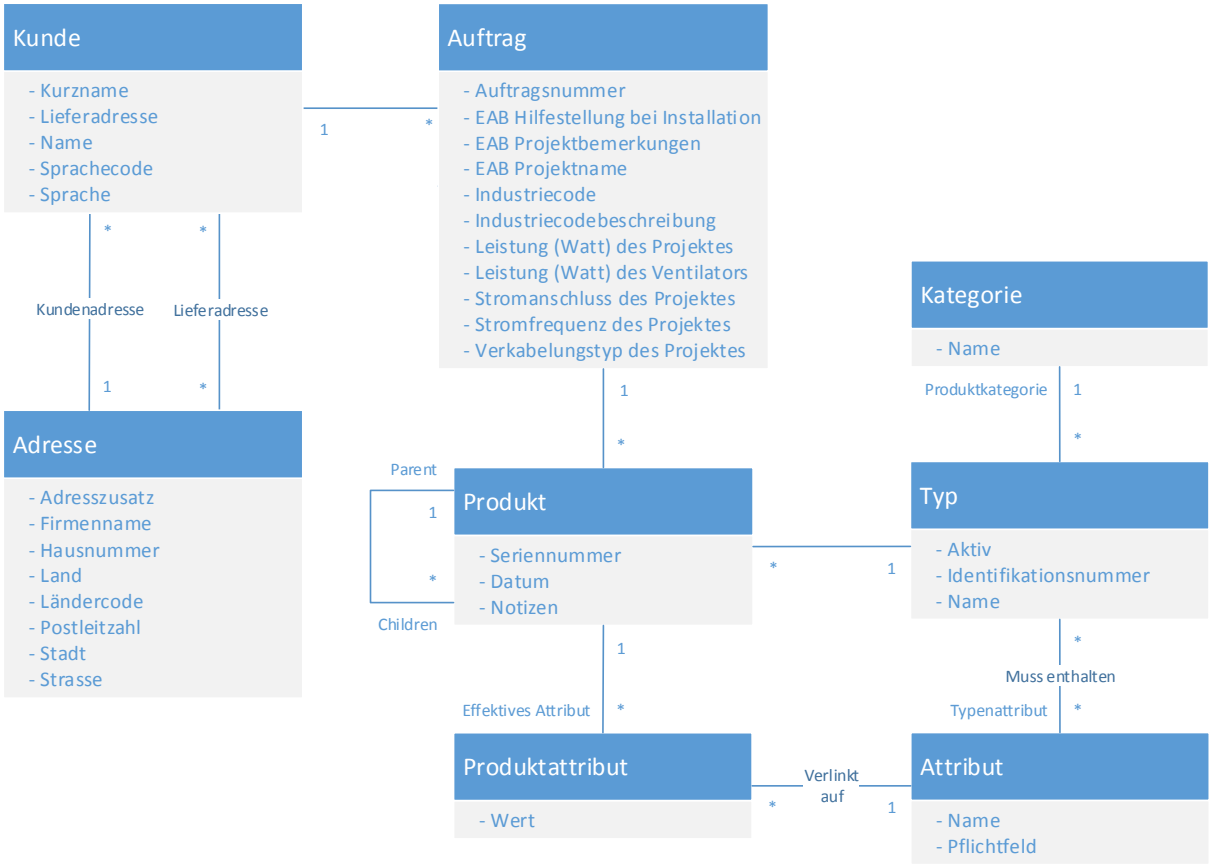


Abbildung 14: Domainmodell Srnmgmt

4.1.2. Klassen

4.1.2.1. Auftrag

Diese Klasse enthält alle relevanten Auftragsinformationen. Die Werte für die Felder werden mittels eines automatisierten „Task“ mit einer ProConcept Datenbank synchronisiert. Ein Auftrag wird mit einem Kunden verknüpft und mit allen auftragsspezifischen Produkten. Einige Attribute werden von der elektronischen Auftragsbearbeitung (EAB) ausgefüllt und verwaltet.

Attribute

Attribut	Typ	Beschreibung
Auftragsnummer	String	Auftragsnummer (aus ProConcept)
EAB Hilfestellung	Boolean	Definiert, ob die EAB – Abteilung bei der Installation der Anlage Hilfe vor Ort leistet
EAB Projektbemerkungen	String	Allgemeine Projektbemerkungen, welche die Installation oder Ähnliches betreffen
EAB Projektname	String	Kurzname (interner Projektname)
Industriecode	Int	Spezifischer Industriecode
Industriecodebeschreibung	String	Beschreibung des Industriecodes
Leistung des Projektes	Int	Eine übergeordnete Leistungsangabe (Watt), in welcher das Element mit dem grössten Leistungsbedarf eingetragen wird. Dient hauptsächlich der vereinfachten Suche nach Projekten
Leistung des Ventilators	Int	Leistungsangabe des Lüfters, welcher in der Stromversorgung (ICS) der ganzen Anlage verbaut ist
Stromanschluss des Projektes	String	Anschluss der Anlage beim Kunden (z.B.: 3x420 V; 230 V; etc.)
Stromfrequenz des Projektes	Byte: • 50 • 60	Die Frequenz (Hz) des Stromanschlusses. Kann nur einen der aufgeführten Werte enthalten
Verkabelungstyp des Projektes	String: • V+E+N • V+PEN • V+PE	Netzanschlussart beim Kunden. Kann nur einen der aufgeführten Werte enthalten

Tabelle 3: Attribute der Auftragsklasse

4.1.2.2. Kunde

Enthält die kundenrelevanten Daten. Diese werden bei der Synchronisation mit ProConcept mitgeliefert und abgeglichen. Ein Kunde muss eine Adresse besitzen und kann variabel viele Lieferadressen besitzen.

Attribute

Attribut	Typ	Beschreibung
Name	String	Name des Kunden (z.B.: Gema Powdercoating)
Kurzname	String	Gema-intern genutzter Kurzname des Kunden
Sprache	String	Die Korrespondenzsprache des Kunden
Sprachcode	String	Sprachcode (z.B.: DE) der Korrespondenzsprache

Tabelle 4: Attribute der Kundenklasse

4.1.2.3. Adresse

Enthält die gesammelten Adressen der Kunden. Ein Kunde ist mit dieser Klasse verknüpft. Die Adresse kann eine Lieferadresse oder eine „normale“ Kundenadresse darstellen.

Attribute

Attribut	Typ	Beschreibung
Firmenname	String	Der Firmenname des Kunden (kann bei Lieferadressen abweichen)
Land	String	Land der Adresse
Ländercode	String	Internationaler Ländercode des Landes
Strasse	String	Strasse der Adresse
Adresszusatz	String	Spezieller Adresszusatz (z.B.: Stockwerk, Postbox, etc.)
Hausnummer	Short	Hausnummer der Adresse
Postleitzahl	String	Postleitzahl der Adresse
Stadt	String	Ortsname der Adresse

Tabelle 5: Attribute der Adressklasse

4.1.2.4. Produkt

Die Kernklasse der Applikation. Die dazugehörige Tabelle enthält alle Produkte, welche eingetragen werden. Ein Produkt hat eine eindeutige Seriennummer und ist mit einem Produkttyp verknüpft. Dieser Typ wiederum definiert, welche zusätzlichen Attribute zu einem Produkt eingetragen werden müssen. Ein Produkt kann ebenfalls Unterprodukte („Children“) enthalten, welche so beispielsweise verbaut wurden. So kann eine Produkthierarchie abgebildet werden.

Attribute

Attribut	Typ	Beschreibung
Datum	Date	Einfügedatum des Produktes
Seriennummer	String	Seriennummer des Produktes. Diese wird zu gewissen Teilen vom Produkttyp vorgegeben und ist im Normalfall eine fortlaufende Nummer
Notizen	String	Spezielle Notizen zu einem Produkt

Tabelle 6: Attribute der Produktklasse

4.1.2.5. Kategorie

Die Produktkategorie. Diese ist der übergeordnete „Typ“ zu den Produkttypen. Eine Produktkategorie kann beispielsweise „Handgerät“ oder „Beschichtungspistole“ sein. Anhand dieser Kategorie wird entschieden, welche Produkttypen dem Benutzer zur Auswahl angezeigt werden.

Attribute

Attribut	Typ	Beschreibung
Name	String	Name der Kategorie

Tabelle 7: Attribute der Kategorienklasse

4.1.2.6. Typ

Der Typ eines Produktes. Der Typ definiert die Zusatzattribute eines Produktes und kann von den Administratoren dynamisch erstellt werden. Es soll möglich sein, dynamisch neue Typen zu definieren und direkt zu verwenden. Ein Typ selbst kann nicht (bzw. nur unter bestimmten Umständen) gelöscht werden. Ein solcher Typ kann beispielsweise „CM-10“ in der Kategorie „Steuerungswürfel“ sein.

Attribute

Attribut	Typ	Beschreibung
Aktiv	Boolean	Definiert, ob der Typ zurzeit aktiv ist und auch von den Benutzern gewählt werden kann. Ist ein Typ in keinem Produkt verwendet, so kann er gelöscht werden. Aus Gründen der Datenintegrität können verwendete Typen nicht gelöscht werden
Identifikationsnummer	String	Definiert den ersten Teil der Seriennummer eines Produktes. Die Identifikationsnummer ist optional und kann pro Typ definiert werden. Ist eine Nummer definiert (z.B.: „17001.“), so wird bei der Erstellung eines Produktes eine Seriennummer vorgeschlagen
Name	String	Name des Produkttyps

Tabelle 8: Attribute der Typenklasse

4.1.2.7. Attribut

Ein Attribut definiert eine zusätzliche Eigenschaft eines Produktes. Enthalten ist der Name des Attributes. Die Attribute können als Pflichtfeld definiert werden und erhalten eventuell ein Überprüfungskriterium für die oberflächliche Validierung. Attribute sind im Typ definiert und ermöglichen ein dynamisches Erstellen von Produkttypen.

Attribute

Attribut	Typ	Beschreibung
Name	String	Name des Attributes
Pflichtfeld	Boolean	Definiert, ob das Attribut ein Pflichtattribut darstellt

Tabelle 9: Attribute der Attributsklasse

4.1.2.8. Produktattribut

Ein Produktattribut ist der effektive Wert eines Attributes. Der Produkttyp definiert die zu verwendenden Attribute, jedoch nur das Produkt kann letztendlich die Attribute enthalten. Mittels dieser Produktattribute werden jedem Produkt ein Satz von Attributen und deren Werte zugewiesen.

Attribute

Attribut	Typ	Beschreibung
Wert	String	Wert des Attributes

Tabelle 10: Attribute der Produktattributsklasse

4.1.3. Beziehungen

Multiplizität	Source Klasse	Destination Klasse	Beschreibung
1:*	Adresse	Kunde	Jeder Kunde besitzt eine Kundenadresse.
:	Adresse	Kunde	Da die Kundenadresse nicht zwingend die Lieferadresse des Kunden ist, können einem Kunden noch mehrere Lieferadressen zugeordnet werden.
1:*	Kunde	Auftrag	Ein Kunde kann mehrere Aufträge aufgeben.
1:*	Auftrag	Produkt	Einem Auftrag können mehrere Produkte zugeordnet werden. Es ist jedoch auch möglich, dass ein Auftrag ohne Produkt existiert.
***:1**	Produkt	Typ	Jedes Produkt stammt von einem gewissen Produkttyp, wobei diese Typen mehrfach verwendet werden können.
***:1**	Typ	Kategorie	Jeder Typ wird wiederum einer Kategorie zugeordnet.
:	Typ	Attribut	Ein Typ spezifiziert die zusätzlichen Attribute für das konkrete Produkt. Deshalb besitzt ein Typ *-Attribute, wobei diese Attribute für mehrere Typen wiederverwendet werden können (daher *:*).
1:*	Produkt	Produktattribut	Ein Produkt enthält eine Liste mit den zusätzlich im Produkttyp definierten Attributen.
***:1**	Produktattribut	Attribut	Ein Produktattribut enthält eine Referenz auf ein Attribut. Ebenfalls ist der konkrete Wert des Attributes darin enthalten.
1:*	Produkt	Produkt	Zwischen den Produkten besteht eine Parent-Child Verbindung. Das bedeutet, dass jedes Produkt beliebig viele Unterprodukte beinhalten kann. Die Produkte ohne Parent sind Hauptprodukte.

Tabelle 11: Beziehungen im Domainmodell

4.2. Systemsequenzen

4.2.1. Produkt erfassen

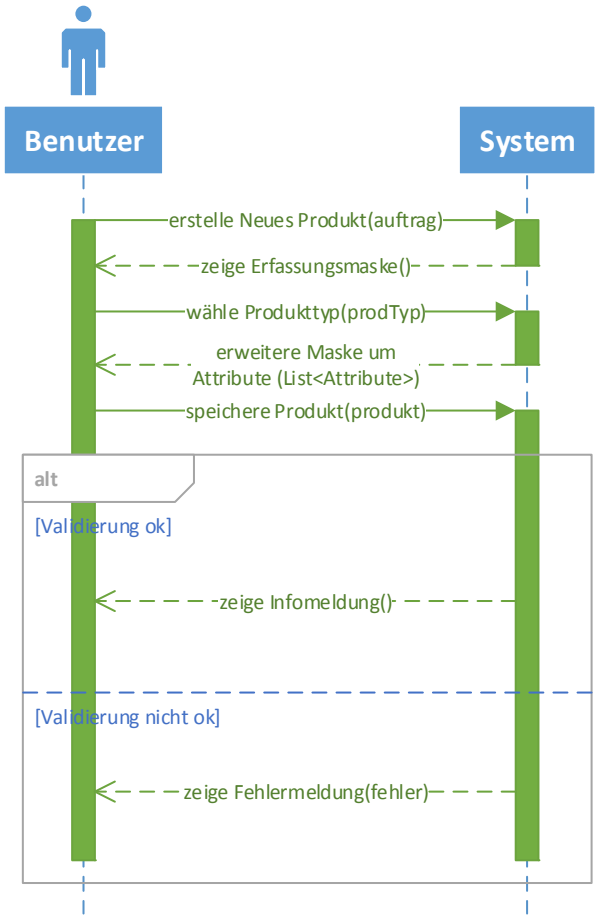


Abbildung 15: Systemsequenzdiagramm zu „Produkt erfassen“

4.3. Systemoperationen

Systemoperationen:

- erstelleNeuesProdukt(auftrag)
- wähleProdukttyp(produkttyp)
- speichereProdukt(produkt)

4.3.1. Contract CO1: erstelleNeuesProdukt

Operation	
Cross References	Use Cases: • Produkt (CRUD)
Preconditions	Benutzer ist authentifiziert und darf für die ausgewählte Produktionslinie Objekte eintragen
Postconditions	Lokal ist das Objekt erstellt und wartet auf die Speicherung

Tabelle 12: Operationcontract CO1

4.3.2. Contract CO2: wähleProdukttyp

Operation	
Cross References	Use Cases: • Produkt (CRUD)
Preconditions	Objekt wurde lokal erstellt und Benutzer ist authentifiziert
Postconditions	Objekt enthält die zusätzlichen – durch den Typ definierten – Attribute

Tabelle 13: Operationcontract CO2

4.3.3. Contract CO3: speichereProdukt

Operation	
Cross References	Use Cases: • Produkt (CRUD) • Massenimport von Produkten • Produkt mit Auftrag verknüpfen
Preconditions	Objekt ist erstellt und die zusätzlichen, vorgegebenen Attribute sind ausgefüllt. Die Validierung muss erfolgreich sein.
Postconditions	Objekt ist in der Datenbank gespeichert und enthält eine eindeutige ID

Tabelle 14: Operationcontract CO3

4.4. Benutzerauthentifizierung

In diesem Abschnitt wird das Konzept zur Authentifizierung und Autorisierung am System beschrieben.

4.4.1. Authentifizierungsmethoden

4.4.1.1. „Anonymous“

Eine anonyme Authentifizierung kommt für uns nicht in Frage, da wir ein gewisses Usermanagement unterstützen müssen.

4.4.1.2. ASP.NET Impersonation

Die gesamte ASP Applikation (inklusive aller Benutzer) läuft unter einem definierten Benutzer. Dieser wird im IIS konfiguriert und sollte so wenige Berechtigungen wie möglich besitzen. Eine richtige Authentifizierung für einzelne Benutzer ist somit nicht möglich oder muss zusätzlich erstellt werden. Somit fällt auch diese Authentifizierungsmethode aus unserem Raster heraus.

Vorteile

- Konfiguration einfach
- Nur ein Server CAL
- Benutzer kann wenig Schaden anrichten

Nachteile

- Keine Differenzierung der verschiedenen Benutzer
- Berechtigungsebenen können nicht dargestellt werden

4.4.1.3. Basic Authentication

Bei der „Basic Authentication“ werden bei jedem http Request ein bestimmter Benutzername und ein Passwort mitgeliefert. Diese Informationen sind mit dem Base64 Verfahren codiert. Diese Codierung bietet keinen wirklichen Schutz vor Entschlüsselung und ist somit als „stand alone“ Authentifizierung nicht zu empfehlen. Um die Sicherheit zu gewährleisten muss eine verschlüsselte Verbindung (SSL / TLS) verwendet werden.

Die Headerinformation der Basic Authentication sieht wie folgt aus:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Vorteile

- Sehr einfach zu implementieren
- Schnelle Authentifizierung
- Erlaubt Differenzierung der verschiedenen Benutzer

Nachteile

- Ohne SSL / TLS ist keine wirkliche Sicherheit vorhanden
- Keine automatisierte Authentifizierung möglich

4.4.1.4. Forms Authentication

Bei dieser Form der Authentifizierung wird dem Benutzer zu Beginn ein Formular in der Webapplikation gezeigt, in welchem er seine Benutzerdaten (Name, Passwort) eingeben kann. Diese Daten werden dann an den Server übertragen, welcher somit den Benutzer authentifiziert. Der Server stellt mittels Session oder Cookie sicher, dass der Benutzer nicht für jeden Request seine Daten eingeben muss.

Dies stellt eine gängige Methode zur Authentifizierung dar und wird meist mittels SSL / TLS zusätzlich gesichert.

Vorteile

- Differenzierung der Benutzer möglich
- „Richtige“ Authentifizierung, welche auch eine Sicherheit darstellt (im Vergleich zu Basic Authentication)
- Benutzersystem wird zu grossen Teilen vom .NET Framework zur Verfügung gestellt

Nachteile

- Keine automatisierte Authentifizierung
- Wenn der Verkehr über http läuft, wird das Passwort im Klartext übertragen

4.4.1.5. Windows Authentication

Hierbei wird der bestehende, von Windows genutzte Authentifizierungsmechanismus verwendet. Die zugrundeliegende Methode nutzt – je nach Verfügbarkeit und Konfiguration – den NTLM Mechanismus oder die Kerberos v5 Variante. Damit die Windowsauthentifizierung reibungslos funktioniert, müssen folgende Kriterien erfüllt sein [1]:

- Client und Webserver müssen in derselben Domäne sein[2]
- Mindestens Internet Explorer 2.0 muss vorhanden sein
- http Proxyverbindungen sind nicht verlangt
- Kerberos v5 benötigt eine Verbindung zum Domänenkontroller

Vorteile

- Differenzierung der Nutzer möglich
- Automatische Authentifizierung (Intranet)
- Benutzer müssen keinen Benutzernamen und kein Passwort eingeben
- Verwaltung der Benutzerberechtigungen über Active Directory möglich
- Benutzersystem wird sehr gut von .NET unterstützt

Nachteile

- Initialaufwand für Administrator hoch, da die gesamte Gruppenstruktur angelegt werden muss
- Wenn der Computer nicht gesperrt ist, kann einfach mit einem anderen Benutzer das Programm genutzt werden

4.4.2. Entscheid

Nach der Evaluation der einzelnen Möglichkeiten erschien die Variante „**Windows Authentication**“ am vielversprechendsten. Die Methode ermöglicht es, die ganze Benutzerstruktur auf das AD abzuwälzen und die Rechteverwaltung ebenfalls über das AD zu steuern. Ein weiterer grosser Vorteil ist, dass die Benutzer kein Passwort mehr eingeben müssen und sich so nicht noch mehr verschiedene Benutzeraccounts merken müssen. Nach dem eingängig erwähnten Initialaufwand kann die Benutzerverwaltung effizient über das AD gemacht werden. Im Weiteren wird das genaue Benutzerkonzept ausgearbeitet.

[2] Dies ist nicht zwingend nötig. Wird „von aussen“ her auf den Webserver zugegriffen, so erscheint ein Anmeldefenster. In diesem muss man sich zwingend mit einem in der Domäne vorhanden Benutzer authentifizieren. Jedoch muss dann darauf geachtet werden, dass die Verbindung gesichert ist.

4.4.3. Umsetzung

4.4.3.1. Umgebung

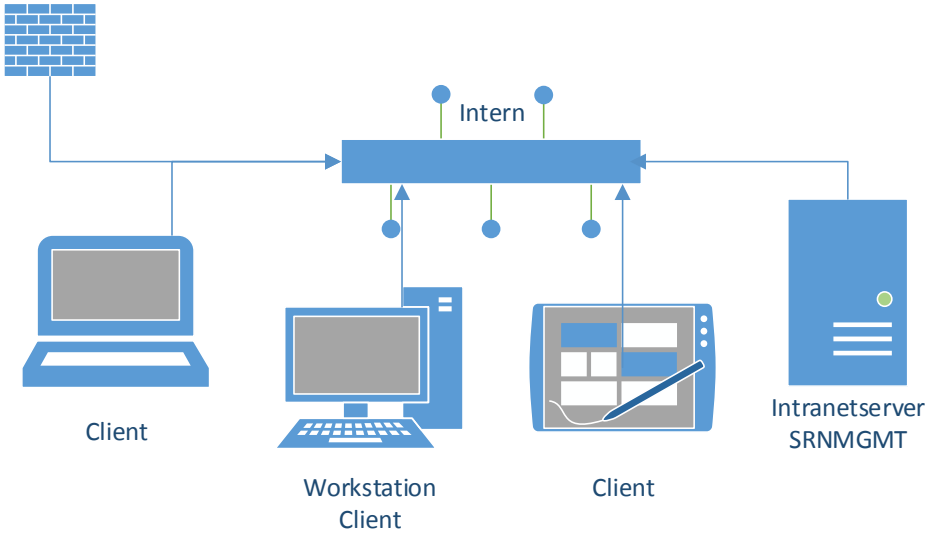


Abbildung 16: Umgebung für das Intranet

4.4.3.2. Anforderungen

Zwingende Anforderungen an die Benutzerverwaltung sind:

- Jede Produktionslinie hat eine AD-Gruppe (um Zugriff auf das linienspezifische Menü zu erhalten)
- Keine Unterscheidung zwischen Read und Write
- Es soll eine Gruppe geben, welche „übersteuern“ kann, das heisst, Personen in dieser Gruppe können trotzdem auf andere Gruppen zugreifen
- Administratorengruppe, welche die Verwaltung der Typen, Attribute etc. übernimmt

Optionale, mit der Gema zu diskutierende Anforderungen:

- Grundlegender Zugriff über separate Gruppe steuern
- Read / Write Trennung
- Eigene Gruppen für Reporting / Barcode

4.4.4. Struktogramm

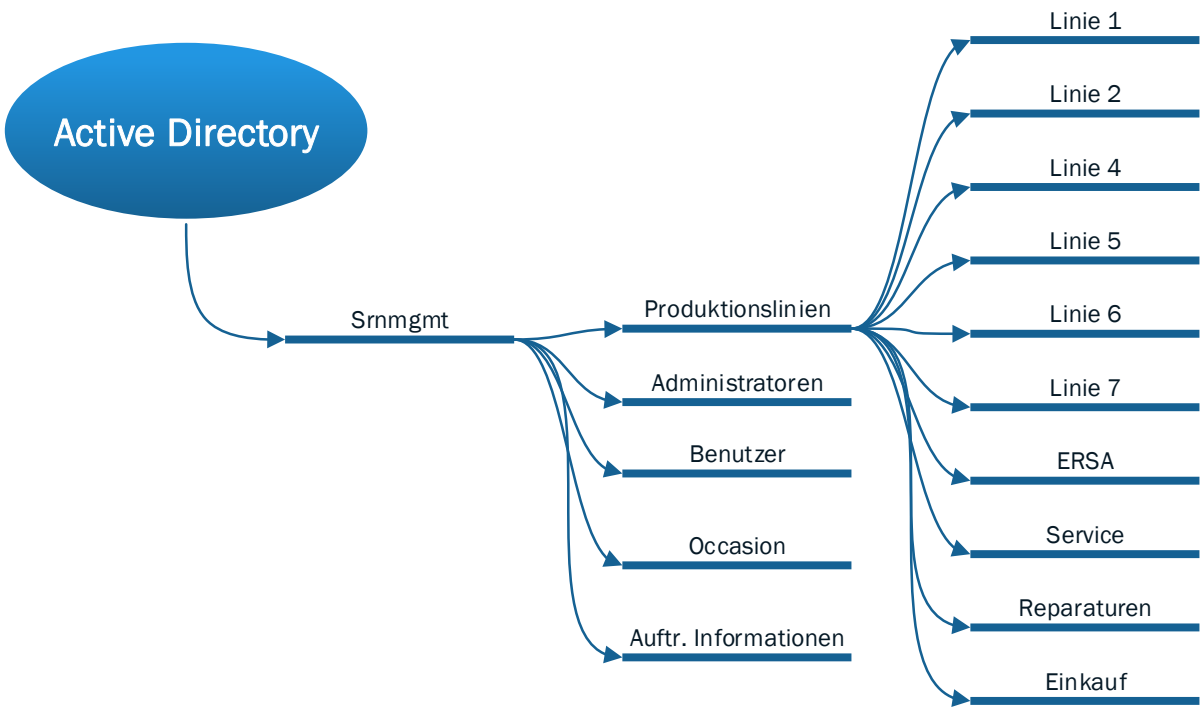


Abbildung 17: Struktogramm der Benutzergruppen

4.4.5. Organisation Units

4.4.5.1. Srnmgmt

In dieser OU sind alle für die Seriennummernverwaltung relevanten Gruppen und verschachtelten OUs hinterlegt. Sie dient der allgemeinen Abgrenzung zur restlichen AD-Struktur.

4.4.5.2. Produktionslinien

Diese OU enthält sämtliche Gruppen für die einzelnen Produktionslinien. Alle Benutzer, welche hier einer Gruppe zugeordnet werden, können auf das jeweilige Linienmenü zugreifen. Die Spezialgruppe „Alle Linien“ ist ebenfalls hier abgelegt. Die Bindung zwischen einer Gruppe und der Produktionslinie geschieht auf Applikationsebene und kann in der Datenbank der Webapplikation angegeben werden.

4.4.6. Benutzergruppen

4.4.6.1. Administratoren

Diese Gruppe enthält alle Administratoren der Seriennummernverwaltung. Diese haben Zugriff auf sämtliche Linienmenüs sowie das Administrationsmenü. In besagtem Menü können die Produkttypen und verschiedenen Produktattribute verwaltet werden, sowie neue Reports hinzugefügt werden.

4.4.6.2. Benutzer

Diese Benutzergruppe enthält alle AD Benutzer, welchen es erlaubt ist, auf die Applikation zuzugreifen. Die Mitgliedschaft dieser Gruppe ist notwendig, um auf die Datenbank und die IIS Applikation zu verbinden.

4.4.6.3. Einzelne Produktionslinien

Jede Produktionslinie erhält eine eigene Verwaltungsgruppe (Bsp.: „Line1“). In diesen Gruppen werden alle zugeteilten Benutzer hinterlegt. Ein Benutzer kann grundsätzlich in mehreren Gruppen sein, sofern dies sein Arbeitsprozess verlangt. Aufgrund dieser Liniengruppen bestimmten sich die nutzbaren Produktkategorien und Produkttypen.

4.4.6.4. Occasion

Alle Benutzer (oder Benutzergruppen), welche dieser Gruppe zugeordnet sind, können ein bestehendes Produkt als Occasionsprodukt einem anderen Produkt oder Auftrag hinzufügen. Geschieht dies, so wird für das Produkt ein Occasionseintrag (Historie Eintrag) angelegt und das Produkt als Occasion markiert.

4.4.6.5. Auftragsinformationen (Auftr. Informationen)

Die Benutzer dieser Gruppe erhalten ein zusätzliches Informationsmenü in der Auftragsübersicht. Diese zusätzlichen Informationen betreffen Projektinformationen und allgemeine auftragsspezifische Eckdaten. In der Auftragssuche kann jeder Benutzer alle Felder suchen, jedoch erhalten nur die Benutzer dieser Gruppe Zugang zur Änderungsmaske.

4.4.7. .NET Anbindung

In der Webapplikation wird dies mittels eines LDAP Konnektors realisiert. Die Webapplikation hat die Möglichkeit, auf den AD Service zuzugreifen und die einzelnen Benutzergruppen abzufragen. Dies geschieht mit einigen Hilfsfunktionen, welche vom .NET Framework zur Verfügung gestellt werden. Die Abfrage muss jedoch selbst ausgewertet werden.

Die „Controller“, welche eingesetzt werden, um einem Client die Webseite auszuliefern, unterstützen ein .NET Attribut, welches erlaubt, Benutzergruppen zu hinterlegen. Nur den angegebenen Benutzergruppen ist es erlaubt, die Aktion (Abruf der Seite, eintragen von Produkten, etc.) auszuführen.

4.4.8. Konfiguration

Um nicht bei jeder Namens-, Benutzer-, Gruppenänderung die Applikation neu verteilen zu müssen, wird ein möglichst konfigurierbarer Ansatz gewählt. Den Administratoren soll es ermöglicht sein, über die Applikationseinstellungen in der IIS-Konsole die Benutzergruppen und allgemeinen Einstellungen anzupassen, wie das folgende Bild demonstriert.

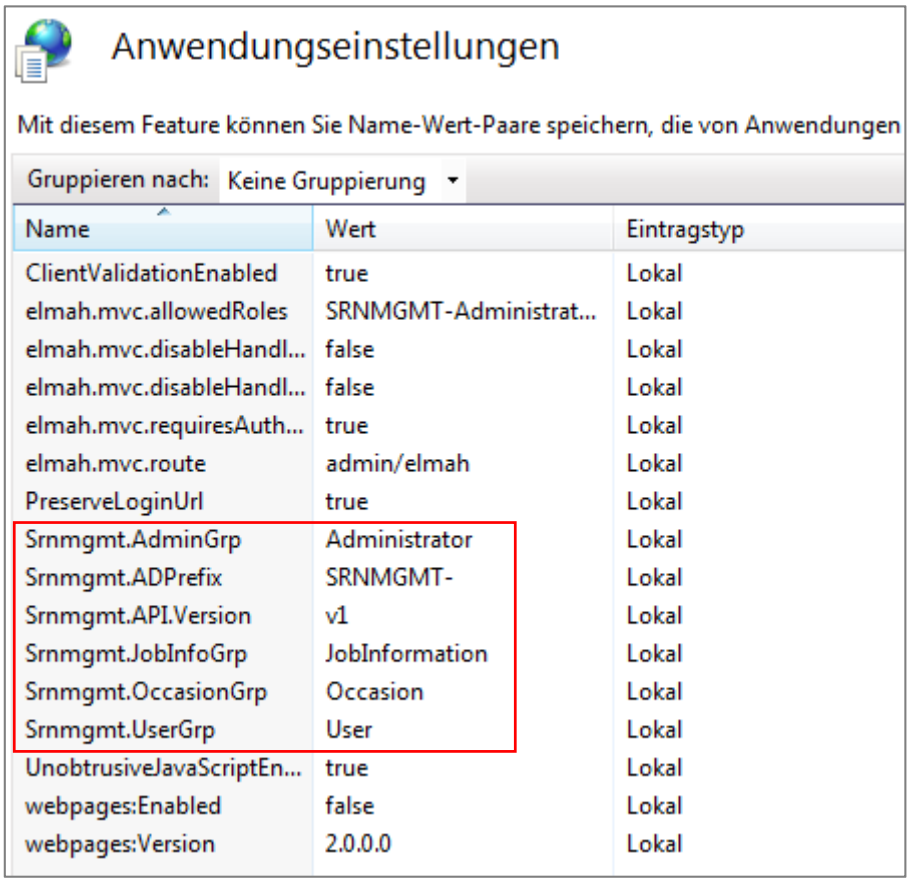


Abbildung 18: Einstellungen der Webapplikation über IIS

4.5. Reporting

4.5.1. Ausgangslage

Die etablierten Reports wie zum Beispiel „Typenschilder drucken“ sollen auch in die neue Applikation übernommen werden. Zudem möchte man sich die Möglichkeit offen halten, neue Reports möglichst einfach der Applikation hinzufügen beziehungsweise alte Reports bearbeiten zu können.

Das Ziel ist es, das neue Reporting so generisch und auf der anderen Seite auch so einfach wie möglich zu gestalten.

Ebenso sollen dem Benutzer auch Möglichkeiten geboten werden, dass die generierten Reports vor dem Drucken noch angepasst werden können. Wünschenswert wäre es deshalb, dass die Benutzer auf bekannte Tools wie zum Beispiel Microsoft Word zurückgreifen können.

4.5.2. Komponenten

Ein Report besteht grundsätzlich immer aus denselben Komponenten:

Komponente	Beschreibung
Daten	Das zentrale Element eines Reports sind die Daten. Nach Generierung des Reports sollen die Daten besser für den Benutzer aufbereitet sein. Die Daten sind allerdings in sehr unterschiedlicher Form vorhanden. Es ist denkbar, dass man dem Report gleich eine Liste mit entsprechenden Datensätzen übergibt oder aber nur eine Verbindung zur Datenbank und zusätzlich ein SQL-Statement.
Parameter	Ein Report kann optional beliebig viele Parameter beinhalten. Diese werden meistens dazu benötigt, die vorhandenen Daten zu Filtern (z. B. Datums-Range, Kundenname etc.)
Design-Vorlage	Mittels Design-Vorlage kann definiert werden, wie die Daten aufbereitet werden sollen. Dazu gehören Farbe, Schriftgrösse, Platzierung von Daten und so weiter. Auch die Design-Vorlagen werden je nach Technologie unterschiedlich abgelegt. Denkbare Formate sind XML-Dateien oder Word Vorlagen (.dot).

Tabelle 15: Komponenten eines Reports

In der folgenden Grafik sieht man, wie die Komponenten zusammenspielen:

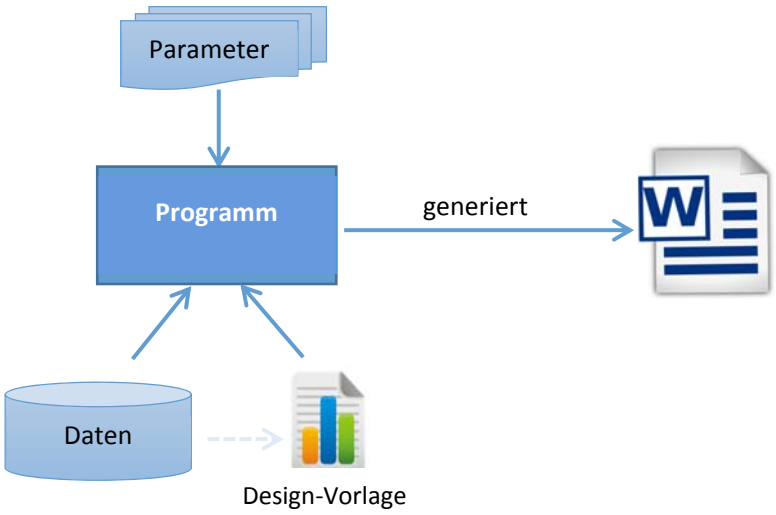


Abbildung 19: Komponenten für das Reporting

Damit ein Benutzer nun einen bestimmten Report generieren kann, muss er nur noch die entsprechende Design-Vorlage auswählen und die benötigten Parameter angeben. Danach generiert das Programm, unterstützt durch ein bestimmtes Framework, den Report.

Im folgenden Kapitel soll das richtige Framework ausgewählt werden. Viele dieser Frameworks stellen bereits Mittel zur Verfügung, damit man nur noch einen vordefinierten Report auswählen und mit Parametern abfüllen muss, um das gewünschte File zu erhalten.

4.5.3. Technologien

4.5.3.1. SQL Server Reporting Services (SSRS)

SQL Server Reporting Services ist ein von Microsoft selbst entwickeltes Berichtgenerierungssystem. Es ist serverbasiert und bereits im Microsoft SQL Server enthalten. Über eine Intranet Seite können neue Reports hochgeladen und durch autorisierte Benutzer auch generiert werden. Allenfalls benötigte Parameter werden dabei automatisch abgefragt. Ebenso ist eine einfache Integration ins ASP.NET oder SharePoint möglich.

Das Design wird in einem eigenen Reportdesigner festgelegt und danach in RDL-Files abgelegt. In diesen Reports kann direkt eine Datasource definiert werden oder man gibt beim Erzeugen die entsprechenden Daten mit. Die generierten Reports können danach in verschiedene Formate wie PDF, Excel und Word exportiert werden.

Vorteile

- Framework ist sehr ausgereift und setzt bereits viel von der geforderten Funktionalität um
  - Reporting kann sehr generisch gestaltet werden
- Wird bereits im Microsoft SQL Server mitgeliefert -> keine zusätzliche Lizenz mehr nötig
- Internationalisierung wird unterstützt

Nachteile

- Läuft nur in Microsoft-Umgebungen
- Der Report wird mit speziellen Files definiert und nicht direkt in Word
  - Es ist noch kein Know-how für RDL-Files vorhanden

4.5.3.2. Crystal Reports (CR)

Crystal Reports ist ein von der Firma SAP entwickeltes Software-Framework zur Erstellung von Berichten. Es ist gut ins .NET Framework und daher auch in ASP.NET integriert. Die Vorlagen der Reports werden in RPT-Files gespeichert. In diese Vorlagen kann entweder direkt eine Datenbankverbindung gebunden oder zur Generierungszeit eine Liste mit Daten übergeben werden. Für das Design des Reports gibt es einen speziellen Designer, der auch mit Visual Studio benutzt werden kann. Die generierten Reports können danach entweder in verschiedene Formate wie PDF, Excel und Word exportiert oder gleich mit dem ReportViewer im Browser angezeigt werden.

Vorteile

- Framework ist sehr ausgereift und setzt bereits viel von der geforderten Funktionalität um
  - Reporting kann sehr generisch gestaltet werden
- CR ist in der Firma bereits im Einsatz (Lizenz schon vorhanden)
  - Das Know-how für das Design der Reports und Erstellen der RTP-Files ist vorhanden
- Da CR direkt in der Applikation läuft, hat man gute Kontrolle über die Security
- Internationalisierung wird unterstützt
- Dank dem Word-Export ist eine Nachbearbeitung durch Benutzer möglich

Nachteile

- Der Report wird mit speziellen Files definiert und nicht direkt in Word
- CR ist teuer, man benötigt zusätzliche Lizenzen

4.5.3.3. Microsoft Word

Die dritte Möglichkeit besteht darin, Microsoft Word oder allgemein Microsoft Office direkt zu verwenden. Im .NET Framework sind bereits gute Schnittstellen vorhanden, um Office-Dokumente nach Wunsch zu bearbeiten. So ist es zum Beispiel möglich, mit einer .Dot-Datei, also einer Vorlage, ein Word-Dokument zu erstellen und dieses mit Daten abzufüllen.

Das Problem dieser Lösung liegt darin, dass sämtliche Logik für die Erstellung selbst codiert werden müsste. Somit wird es schwierig, ein generisches Reporting zur Verfügung zu stellen, ausser man bildet quasi selbständig einen Reporting Service nach. Dafür können die Reports direkt in Microsoft Word designt werden und es wird kein zusätzliches Know-how für das Designen benötigt.

Vorteile

- Design wird gerade in Microsoft Word definiert
- Da man alles selber programmiert, ist man unabhängig und sehr flexibel

Nachteile

- Man muss Logik, die schon in bewährten Frameworks umgesetzt wurde, nachbilden. Dies sollte möglichst verhindert werden -> Don't Repeat Yourself
- Internationalisierung nicht standardmässig unterstützt
- Falls neue Reports hinzugefügt werden, muss ziemlich sicher auch der Code angepasst werden

4.5.4. Technologie Entscheid

Die dritte Variante nur mit Microsoft Word ohne eigentliches Reporting-Framework ist nicht zu empfehlen, da viele Funktionen selbst realisiert werden müssten, die man in den anderen beiden Lösungen bereits erhält.

Der Entscheid zwischen SSRS und CR ist im Grossen und Ganzen Geschmacksache. Der grösste Nachteil von SSRS, dass es nur in einer reinen Microsoft Umgebung läuft, fällt hier nicht ins Gewicht, da es später nur in einer Microsoft Umgebung laufen wird. Ebenso der Nachteil von CR, dass eine Lizenz benötigt wird, kann vernachlässigt werden, da die Firma bereits eine Lizenz besitzt. CR hat noch den kleinen Vorteil, dass in der Firma bereits Know-how für das Erstellen von Reports (also den RTP-Files) vorhanden ist. Aus diesen Gründen wurde entschieden, **Crystal Reports** zu verwenden.

4.6. CSV-Import

4.6.1. Anforderungen

Der CSV Import dient dem allgemeinen Importieren von Produkten in die Webapplikation. Somit muss es über diese Funktion möglich sein, sämtliche Produkte und Produkttypen (dynamisch) zu importieren. Folgende Anforderungen sind dazu gegeben:

- Datei soll beliebig benannt werden können (mit Dateiendung „.csv“)
- Produkte müssen eindeutig einem Typen zugewiesen werden können
- Produkten müssen die entsprechenden Typenattribute zugewiesen werden können
- Produkten müssen Unterprodukte (z.B. Pistole → Kaskade)[3] zugewiesen werden können
- Zusätzlich enthaltene Daten oder Leerzeilen müssen ignoriert werden
- Es müssen verschiedene Datumsformate akzeptiert werden

4.6.2. Definitionen

Um eine einheitliche Möglichkeit zum Importieren über CSV zu ermöglichen, müssen folgend beschriebene Dinge definiert und eingehalten sein. Diese Definitionen sind notwendig, da die Datenstruktur der Webapplikation sehr dynamisch ist und erweiterbar sein soll. Mitunter sind in der gesamten Applikation dynamische Produkttypen möglich, welche eine variable Anzahl an Attributen enthalten können. Um diese veränderbare Struktur in einem Import abzubilden, müssen gewisse Strukturen im CSV festgelegt sein.

4.6.2.1. Allgemeine Definitionen

Um einen reibungslosen Ablauf des Imports zu ermöglichen, sind folgende allgemeine Definitionen einzuhalten:

- Dateibenennung: Grundsätzlich egal, solange das Datei interne Format mit CSV übereinstimmt
- Allgemeine CSV-Format Eigenschaften (Komma- oder Semikolon getrennt)
- CSV muss eine Kopfzeile mit den Spaltenüberschriften enthalten
- Seriennummer muss der Typenidentifikation entsprechen (Regex)
- Nach Möglichkeit leere Spalten vermeiden (nicht zwingend, jedoch auch aus übersichtstechnischen Gründen zu empfehlen)[4]

[3] Diese genannten Produkte sind Beispiele der Gema. Es handelt sich dabei um eine Pulverbeschichtungspistole, welche eine Kaskade mit Seriennummer enthält.
[4] Spalten, welche keine Überschrift haben, werden komplett überlesen bzw. ignoriert.

4.6.2.2. Pflichtfelder und Namensgebung

Da die CSV nicht lokalisiert sein müssen und der Import auf eine möglichst einfache Art realisiert werden soll, werden für die erzwungenen Felder vorgegebene Namen verwendet. Diese speziellen Felder werden im Folgenden aufgelistet mit der Information, ob sie zwingend oder optional sind:

- Seriennummer des Produktes: „SerNr“ – zwingend
- (Produktions-)Datum des Produktes: „Date“ – zwingend
- Typ des Produktes: „Type“ – zwingend
- Produktkategorie des Produktes: „Cat“ – zwingend (für die Eindeutigkeit der Typen)
- Notizen des Produktes: „Notes“ – optional

Ist eines der zwingenden Felder bzw. der zwingenden Spalten im CSV nicht vorhanden, so wirft die Applikation einen Fehler und der Importvorgang wird abgebrochen.

4.6.2.3. Unterprodukte

Da der von der Webapplikation genutzte Weg zur Verknüpfung von Produkten hier nicht angewendet werden kann, muss ein „Top-Down“ Ansatz ermöglicht werden. Dies bedeutet, dass die Unterprodukte bereits in der Datenbank oder vor dem Elternprodukt im CSV bestehen müssen. Um einem Produkt ein Unterprodukt zuzuweisen, kann im CSV eine Spalte mit einem beinahe beliebigen Namen gegeben werden. Der Name der Spalte muss lediglich folgenden String enthalten: „SubSerNr“. Sind solche Spalten vorhanden, so muss sie nicht zwingend ausgefüllt sein. Die Spalte wird nur in Betracht gezogen, wenn auch ein Wert darin steht. Nachfolgend sind die Randbedingungen zu Unterprodukten aufgelistet:

- Spalte „SubSerNr“ nicht zwingend
- Name der Spalte muss lediglich den String „SubSerNr“ enthalten, somit sind folgende Namen gültig:
  - „SubSerNr1“
  - „MySubSerNr“
  - „KaskadeSubSerNr“
  - …
- Ist kein Wert in der Spalte eingetragen, wird die Spalte ignoriert (es besteht kein Ausfüllzwang)
- Das zu verknüpfende Produkt (das Unterprodukt) muss bereits in der Datenbank ODER oberhalb der aktuellen Zeile bestehen. Somit können beispielsweise Pistole und Kaskade in einem CSV importiert werden: (Beispiel) Zeile 15 enthält die Kaskade mit Nummer E12345 und Zeile 47 die Pistole mit Nummer 15001.12345 und enthaltener Kaskade E12345
- Wird ein Unterprodukt nicht gefunden (in der DB oder im aktuellen Import), so wird das zu importierende Produkt mit einem Fehler behaftet und nicht importiert

4.6.2.4. Typen und Typattribute

Da die Typen für die Produkte dynamisch sind und auch dynamisch mit Attributen ausgestattet werden können, müssen gewisse Richtlinien eingehalten werden, um einen möglichst offenen Import zu ermöglichen. Die Werte in der „Type“-Spalte müssen genau den Typenbezeichnungen entsprechen, da ein „Fuzzymatching“ zu vielen Fehlern führen würde. Folgende Regeln sind beim Import für Typen zu befolgen:

- Typenname muss genau dem Namen der Datenbank entsprechen (kann über Administrationsmenü der EDV abgerufen werden)
- Falls der Typenname nicht eindeutig ist, so wird der erste gefundene Typ verwendet, jedoch wird auf die Produktekategorie gefiltert (es können auch inaktive Typen verwendet werden!)
- Wird ein Typ nicht gefunden, so wird das zu importierende Produkt mit einem Fehler gekennzeichnet und der Import fährt beim nächsten Produkt weiter

Da die Typen eine variable Anzahl an Attributen haben können und diese Attribute ihrerseits zwingend oder nicht zwingend sein können, müssen auch hier gewisse Regeln eingehalten werden:

- Die einzelnen Produktattribute werden in eigene Spalten aufgeteilt
- Ist in einem Typ ein Attribut nicht zugewiesen, so wird es nicht verknüpft (ermöglicht den Import von mehreren Produkttypen in einem CSV, zum Beispiel Pistolen und Kaskaden)
- Der Spaltenname muss exakt dem deutschen oder englischen Namen des Attributes in der Datenbank entsprechen (inklusive Leerzeichen und Klammerausdrücken o. Ä.)
- Ist ein Attribut nicht als benötigt / zwingend markiert, so muss die Spalte nicht vorhanden sein
- Ist ein Attribut nicht als benötigt / zwingend markiert, so muss die vorhandene Spalte nicht ausgefüllt sein
- Ist ein zwingendes Attribut nicht ausgefüllt oder fehlt die ganze Spalte, so wird ein Fehler verursacht, was dazu führt, dass das Produkt nicht importiert wird. Der restliche Import wird fortgesetzt

4.6.2.5. Datum

Da der Import verschiedene Zeitformate erkennen muss, sind nachfolgend einige Beispiele aufgelistet:

- dd.MM.yyyy (17.04.2013)
- yyyy-MM-dd (2013-04-17)
- yyyy-MM-dd-HH:mm:ss (2013-04-17-23:59:59 – SPS Format)

Die oben genannten Formate funktionieren garantiert und sind getestet. Grundsätzlich sollten alle nach ISO 8806 formatierten Formate sowie alle gängigen Datenformate funktionieren. Jedoch kann es sein, dass spezielle Datenformate (wie das der SPS Steuerung) nicht als konventionelle Formate gelten. Diese müssen speziell in die Software eingepflegt werden.

4.6.3. Datenprüfung

Um Fehler in der Datenbank zu vermeiden, werden verschiedene Schutz- und Prüfmechanismen für den Import angewandt. Grundsätzlich gibt es sechs Fehlerquellen, welche im Folgenden aufgelistet und beschrieben sind:

- Das Fehlen von zwingend benötigten Grundfeldern (SerNr, Date, Type): führt zum Abbruch des Imports
- Die Seriennummer ist nicht eindeutig bzw. doppelt
- Der Typ kann nicht gefunden werden
- Das Datum kann nicht übersetzt (geparst) werden
- Das Unterprodukt wurde nicht gefunden (in der DB oder vor dem aktuellen Produkt im Import)
- Das Produkt ist ungültig. Dieser Fehler ist gegliedert in mehrere Teile, jedoch wird dem Benutzer nur die Meldung der Ungültigkeit angezeigt. Folgende Fehler können zu dieser Meldung führen:
    - Seriennummer passt nicht zum Typ (Regex)
    - Zwingende Attribute des Typs fehlen
    - Ungültige Attribute wurden zugewiesen
    - Nicht mehr aktive Attribute wurden zugewiesen

Mit Ausnahme des Fehlers mit fehlenden Grundfeldern überspringen alle anderen Fehler das aktuelle zu importierende Produkt (das entsprechende Produkt wird mittels Fehlermeldung gekennzeichnet). Der genannte Ausnahmefehler führt zum Abbruch des Imports.

4.6.4. Beispiele

Nachfolgend werden einige Beispiele für solche CSV Files aufgelistet (jeweils in tabellarischer und effektiver Form). Für alle Beispiele gelten folgende Grundbedingungen:

- Typ „P01“ ist eine Pulverbeschichtungspistole mit:
    - Produktkategorie „Pistole“
    - einem erzwungenen Attribut „Monteur“
    - Identifikations-Regex „15201[.]\d{5}“ (15201.<Laufnummer>)
- Typ „K01“ ist eine Pistolenkaskade mit:
    - Produktkategorie „Kaskade“
    - Einem erzwungenen Attribut „Strom-Messwert“
    - Einem nicht erzwungenen Attribut „Farbe“
    - Identifikations-Regex „E\d{5}“ (E<Laufnummer>)

4.6.4.1. Erfolgreicher Import eines Typs

Tabellarisch

SerNr	Date	Type	Cat	Notes	Monteur
15201.00001	14.02.2013	P01	Pistole		HM
15201.00002	14.02.2013	P01	Pistole		HM
15201.00003	14.02.2013	P01	Pistole	Speziell.	HM
15201.00004	14.02.2013	P01	Pistole		HM
15201.00005	14.02.2013	P01	Pistole		HM

Tabelle 16: Import eines einzelnen Typen

CSV

```
SerNr;Date;Type;Cat;Notes;Monteur
15201.00001;14.02.2013;P01;Pistole;;HM
15201.00002;14.02.2013;P01;Pistole;;HM
15201.00003;14.02.2013;P01;Pistole;Speziell.;HM
15201.00004;14.02.2013;P01;Pistole;;HM
15201.00005;14.02.2013;P01;Pistole;;HM
```

4.6.4.2. Erfolgreicher Import mehrerer Typen

Tabellarisch

SerNr	Date	Type	Cat	Notes	Monteur	Strom-Messwert	Farbe
15201.00001	14.02.2013	P01	Pistole		HM		
15201.00002	14.02.2013	P01	Pistole		HM		
E01234	14.02.2013	K01	Kaskade	Speziell.		1500	
E01235	14.02.2013	K01	Kaskade			1700	Blau

Tabelle 17: Import mit mehreren Typen

CSV

```
SerNr;Date;Type;Cat;Notes;Monteur;Strom-Messwert;Farbe
15201.00001;14.02.2013;P01;Pistole;;HM;;
15201.00002;14.02.2013;P01;Pistole;;HM;;
E01234;14.02.2013;K01;Kaskade;Speziell.;;1500;
E01235;14.02.2013;K01;Kaskade;;;1700;Blau
```

4.6.4.3. Erfolgreicher Import mit Unterprodukten

Tabellarisch

SerNr	Date	Type	Cat	SubSerNr	Monteur	Strom-Messwert
E00001	14.02.2013	K01	Kaskade			1200
E00002	14.02.2013	K01	Kaskade			500
E00003	14.02.2013	K01	Kaskade			700
15201.00001	14.02.2013	P01	Pistole	E00001	HM	
15201.00002	14.02.2013	P01	Pistole	E00002	HM	

Tabelle 18: Import mit Unterprodukten

CSV

```
SerNr;Date;Type;Cat;SubSerNr;Monteur;Strom-Messwert
E00001;14.02.2013;K01;Kaskade;;;1200
E00002;14.02.2013;K01;Kaskade;;;500
E00003;14.02.2013;K01;Kaskade;;;700
15201.00001;14.02.2013;P01;Pistole;E00001;HM;
15201.00002;14.02.2013;P01;Pistole;E00002;HM;
```

4.6.4.4. Fehlerhafte Imports

Fehlerhafte Seriennummern

Tabellarisch

SerNr	Date	Type	Cat	Notes	Monteur
15201.00001	14.02.2013	P01	Pistole		HM
00001.00002	14.02.2013	P01	Pistole		HM
15201.00003	14.02.2013	P01	Pistole	Speziell.	HM
15201.00004	14.02.2013	P01	Pistole		HM
15201.00005	14.02.2013	P01	Pistole		HM

Tabelle 19: Import mit fehlerhaften Seriennummern

CSV

```
SerNr;Date;Type;Cat;Notes;Monteur
15201.00001;14.02.2013;P01;Pistole;;HM
00001.00002;14.02.2013;P01;Pistole;;HM
15201.00003;14.02.2013;P01;Pistole;Speziell.;HM
15201.00004;14.02.2013;P01;Pistole;;HM
15201.00005;14.02.2013;P01;Pistole;;HM
```

Doppelte Seriennummern

Tabellarisch

SerNr	Date	Type	Cat	Notes	Monteur
15201.00001	14.02.2013	P01	Pistole		HM
15201.00002	14.02.2013	P01	Pistole		HM
15201.00001	14.02.2013	P01	Pistole	Speziell.	HM
15201.00004	14.02.2013	P01	Pistole		HM
15201.00005	14.02.2013	P01	Pistole		HM

Tabelle 20: Import mit doppelten Seriennummern

CSV

```
SerNr;Date;Type;cat;Notes;Monteur
15201.00001;14.02.2013;P01;Pistole;;HM
15201.00002;14.02.2013;P01;Pistole;;HM
15201.00001;14.02.2013;P01;Pistole;Speziell.;HM
15201.00004;14.02.2013;P01;Pistole;;HM
15201.00005;14.02.2013;P01;Pistole;;HM
```

Fehlerhaftes Datum

Tabellarisch

SerNr	Date	Type	Cat	Notes	Monteur
15201.00001	14.02.2013	P01	Pistole		HM
15201.00002	14.02.2013	P01	Pistole		HM
15201.00003	14.02.20xx	P01	Pistole	Speziell.	HM
15201.00004	14.02.20yy	P01	Pistole		HM
15201.00005	14	P01	Pistole		HM

Tabelle 21: Import mit fehlerhaften Datumsformat

CSV

```
SerNr;Date;Type;Cat;Notes;Monteur
15201.00001;14.02.2013;P01;Pistole;;HM
15201.00002;14.02.2013;P01;Pistole;;HM
15201.00003;14.02.20xx;P01;Pistole;Speziell.;HM
15201.00004;14.02.20yy;P01;Pistole;;HM
15201.00005;14;P01;Pistole;;HM
```

Fehlerhafter Produkttyp

Tabellarisch

SerNr	Date	Type	Cat	Notes	Monteur
15201.00001	14.02.2013	P01	Pistole		HM
15201.00002	14.02.2013	P0x	Pistole		HM
15201.00003	14.02.2013	P05	Pistole	Speziell.	HM
15201.00004	14.02.2013	P01	Pistole		HM
15201.00005	14.02.2013	P01	Pistole		HM

Tabelle 22: Import mit fehlendem / fehlerhaften Typ

CSV

```
SerNr;Date;Type;Cat;Notes;Monteur
15201.00001;14.02.2013;P01;Pistole;;HM
15201.00002;14.02.2013;P0x;Pistole;;HM
15201.00003;14.02.2013;P05;Pistole;Speziell.;HM
15201.00004;14.02.2013;P01;Pistole;;HM
15201.00005;14.02.2013;P01;Pistole;;HM
```

Fehlende Attribute

Tabellarisch

SerNr	Date	Type	Cat	Notes
15201.00001	14.02.2013	P01	Pistole	
15201.00002	14.02.2013	P01	Pistole	
15201.00003	14.02.2013	P01	Pistole	Speziell.
15201.00004	14.02.2013	P01	Pistole	
15201.00005	14.02.2013	P01	Pistole	

Tabelle 23: Import mit fehlenden Attributen

CSV

```
SerNr;Date;Type;Cat;Notes
15201.00001;14.02.2013;P01;Pistole;
15201.00002;14.02.2013;P01;Pistole;
15201.00003;14.02.2013;P01;Pistole;Speziell.
15201.00004;14.02.2013;P01;Pistole;
15201.00005;14.02.2013;P01;Pistole;
```

4.7. Barcode-Client

4.7.1. Ausgangslage

Während der ersten Besprechung mit der Gema stellte sich heraus, dass neu in der Software eine Schnittstelle zum Einlesen und Erstellen von Barcodes gewünscht ist. Zum einen sollen die vom Lieferanten erhaltenen Produkte über einen Barcode verfügen, damit diese bei Erhalt automatisch in die Software eingelesen werden können. Zum anderen soll optional beim Druck der Typenschilder die Möglichkeit bestehen, einen Barcode zu generieren. Dieser Barcode enthält sämtliche produktspezifischen Informationen und kann dem Kunden helfen, digital auf diese Daten zuzugreifen. In diesem Rahmen stellt sich die Frage, welche Arten von Barcode und welche Barcode-Scanner sich für die erwähnten Funktionalitäten am besten eignen.

4.7.2. Barcode

4.7.2.1. Variante 1: EAN-13

Beschreibung

 7612345678900	Präfix 76 kennzeichnet die Schweiz	2-stellig	76
	Herstellercode vergeben von EAN Schweiz Teilnehmernummer (Identifikation des EAN Mitgliedes)	7-stellig	1234567
	Artikelnummer vergeben vom Hersteller Produkte- und/oder Adressidentifikation	3-stellig	890
	Prüfziffer	1-stellig	0

Tabelle 24: EAN Ländercodes [2]

Der Standard EAN Strichcode hat 13 Ziffern welche jeweils einen Wert zwischen 0-9 annehmen können. Jedes Zeichen besteht aus 11 Elementen. Alle Striche und Lücken tragen Informationen. Er wird hauptsächlich in Lebensmittelsupermärkten verwendet. Die Produkte sind mit der Europäischen Artikelnummer (EAN), als Barcode verschlüsselt, bedruckt. Der EAN-Code enthält keine Daten wie zum Beispiel den Preis. Er enthält lediglich eine Anzahl verschiedener Zahlen, mithilfe dieser kann zum Beispiel eine Kasse den Preis im System abrufen. Der EAN-Code ist somit wertlos, sofern keine Bezugsquelle für die Daten existiert.

Vorteile

- Einfache Implementation

Nachteile

- Muss in korrekter Position eingescannt werden
- Es können keine Daten gespeichert werden
- Registration notwendig
- Lizenzgebühren fallen an

4.7.2.2. Variante 2: PDF417

Beschreibung

Abbildung 20: PDF417 Barcode

PDF417 steht für „portable data file“ und gehört zur Familie der Stacked-Codes. Es ist ein mehrfach gestapelter Strichcode (Stapelcode), der aus mehreren untereinander angeordneten Strichcodes besteht. Die Zeichen sind in „Codewörtern“ verschlüsselt. Jedes Codewort besteht aus 17 Modulen, aufgeteilt in 4 Stricke und 4 Lücken. Die Anzahl der Zeilen variiert zwischen 3 bis 90 Zeilen und jede dieser Zeilen enthält einen Zeilenindikator zur Orientierung für das Lesegerät. Um den Inhalt der Gesamtnachricht abzusichern, dienen zwei Codewörter als Prüfzeichen. Es können weitere Codewörter (max. 512) eingefügt werden.

Vorteile

- Hohe Datendichte
- Datenkapazität: 2710 numerische Buchstaben, 1850 alphanumerische Buchstaben, 1108 Bytes (binäre Daten)
- Fehlerkorrektur
- Flexibilität in der Anpassung von Informationen auf eine gegebene Fläche durch variable Höhe, Breite und Informationsdichte

Nachteile

- Hohe Auflösung für den Druck oder die Anzeige nötig
- Schwierige Implementation
- Muss in korrekter Position eingescannt werden

4.7.2.3. Variante 3: QR Code

Beschreibung

Abbildung 21: QR Code

QR Codes gehören zur Familie der 2D-Barcodes, die im Gegensatz zu herkömmlichen Barcodes sowohl horizontal wie auch vertikal Informationen enthalten. QR steht für „quick response“, was so viel wie „schnelle Antwort“ bedeutet. Jeder Punkt in einem QR Code repräsentiert ein Bit.

Der ursprüngliche Einsatzzweck war die Kennzeichnung von Baugruppen und Elementen in logistischen Prozessen in der Automobilproduktion des Toyota-Konzerns. Heute ist der QR Code sehr verbreitet für die codierte Abbildung von Webadressen (URLs), Telefonnummern, Adressen, informierende Texte, vCards, WLAN-Zugangsdaten oder Geodaten.

Der QR-Code ist als öffentlicher Standard etabliert und die Verwendung ist lizenz- und kostenfrei.

Vorteile

- Hohe Datendichte
- Datenkapazität: 7089 numerische Buchstaben, 4296 alphanumerische Buchstaben, 2953 Bytes (binäre Daten)
- Fehlerkorrektur: QR Codes offerieren eine eingebaute Fehlerkorrektur. Es können folgende Fehlerkorrektur-Levels gewählt werden: low, medium, quartile, high. Abhängig vom Fehlerkorrektur-Level ist es möglich, zwischen 7%(low) und 30%(high) der unlesbaren Daten wiederherzustellen.
- Einfache Implementation
- Grosse Verbreitung
- Kann in jeder Position eingescannt werden. Das wird durch die drei Quadrate in den Ecken ermöglicht, die zur Orientierung dienen. Scanner können so den QR Code mit high speed decodieren.

Nachteile

- GS1 QR Code wurde noch nicht standardisiert
- Bei grossem Inhalt muss der QR Code grösser gedruckt werden, damit er problemlos eingelesen werden kann
- Der Barcode-Scanner ist teurer

4.7.2.4. Variante 4: DataMatrix Code

Abbildung 22: DataMatrix Code

Der DataMatrix Code gehört ebenfalls zur Familie der 2D-Barcodes. Er besteht aus schwarzen und weissen Feldern die in einer quadratischen Form angeordnet sind.

Er wird meist dort eingesetzt, wo kleine Objekte markiert werden müssen, da er die Fähigkeit besitzt, viele Daten auf kleinem Raum zu speichern.

Der Gebrauch des Datamatrix Code ist ebenfalls lizenz- und kostenfrei.

Vorteile

- Hohe Datendichte
- 2335 alphanumerische Buchstaben
- Fehlerkorrektur: ca. 30%
- Kleinste Version: 10x10 Felder (QR Code: 21x21 Felder)
- Kleinerer Overhead als der QR Code, sofern nicht viele Daten gespeichert werden
- Einfache Implementation
- GS1 Datamatrix Code bereits standardisiert
- Kann in jeder Position eingescannt werden

Nachteile

- Der Barcode-Scanner ist teurer
- Der Datamatrix Code wird grösser als der QR Code ab einer Länge von 89 Feldern, respektive 721 alphanumerischen Buchstaben, sofern die Fehlerkorrekturrate bei beiden Codes bei 30% liegt.

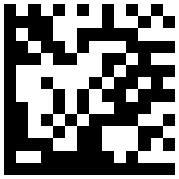
4.7.3. Barcode Entscheid

Aktuell wird nur die Seriennummer eines Produktes im Barcode enthalten sein. Später jedoch wäre es denkbar, auch weitere Produktinformationen im Barcode zu speichern. Da der neu eingesetzte Barcode auch zukünftigen Anforderungen genügen muss, ist der Einsatz eines 2D Barcodes nötig. Am besten eignet sich hierfür der **DataMatrix Code**, da der Platzbedarf trotz 30% Fehlerkorrektur bei geringer Datenmenge sehr klein ist. Ein weiterer Vorteil ist die mögliche Implementation des **GS1 Standards**. Es soll somit ein GS1 DataMatrix Code mit einer Symbolgrösse von 18x18 zum Einsatz kommen, der auf einer Grösse von 12x12mm abgebildet wird. Dabei ist es möglich, bis zu 25 alphanumerische Buchstaben zu speichern.

4.7.4. GS1 Implementation

Zur Codierung der Daten in einem GS1 DataMatrix Code wir der Application Identifier (AI) Standard verwendet. Folgende Codes wurden bereits spezifiziert:

Produkt:
Format: (21)<Seriennummer>
Beispiel zur Codierung: FNC12115201.00100
Ausgabe beim Lesen: FNC12115201.00100



Auftrag:
Format: (91)<Auftragsnummer>
Beispiel zur Codierung: FNC1915013-00001
Ausgabe beim Lesen: FNC1915013-00001

Komponente:
Format: (92)Component
Beispiel zur Codierung: FNC192Component
Ausgabe beim Lesen: FNC192Component

Ein GS1 DataMatrix Code startet immer mit einem FNC1 als Startzeichen. Dies dient zur Identifikation. Die Klammern um die AIs sind nur zur besseren Ansicht vorhanden. Sie gehören nicht zu den Daten und werden daher auch nicht codiert. AIs, die eine variable Feldlänge haben, müssen mit dem Funktionszeichen 1 (FNC1) abgeschlossen werden, sofern nicht die maximale Feldlänge ausgenutzt wird. Das ist jedoch nur nötig, wenn es sich dabei nicht um den letzten AI im Code handelt.

Wenn also nur die Seriennummer codiert wird, wird kein FNC1 am Schluss benötigt. Falls jedoch zum Beispiel noch das Produktionsdatum folgen sollte, muss das Funktionszeichen vorhanden sein, da die Seriennummer nicht 20 Zeichen lang ist: FNC12115201.00100FNC111130420

Liste der verfügbaren AIs: http://www.gs1.ch/gs1-system/das-gs1-system/barcodes-identification#Section7

Spezifikation für die Generierung: http://www.gs1.ch/docs/default-source/gs1-system-document/gs1-datamatrix-introduction-and-technical-overview.pdf?sfvrsn=2

4.7.5. Schnittstellen

4.7.5.1. USB Keyboard

Um die Daten des Scanners/Imagers zu erhalten, wird dieser im Normalfall auf USB-KBD (Keyboard) gestellt. Der Scanner/Imager greift somit auf die im System vorhandenen Treiber für USB-Keyboard Tastaturen zu und gibt die gelesenen Daten in das Textfeld aus, welches gerade den Fokus besitzt. Die Ausgabe wird standardmässig mit einem „Enter“ abgeschlossen, was jedoch konfigurierbar ist.

4.7.5.2. Virtuelle COM-Schnittstelle

Die Daten des Scanners/Imagers werden per USB an den PC gesendet. Der Treiber emuliert einen virtuellen COM-Port, über den mit dem Scanner/Imager kommuniziert werden kann. Als Entwickler kann eine Serial API verwendet werden, um eine Verbindung zum Scanner/Imager herstellen und Daten übermitteln zu können.

4.7.5.3. OPOS und die Unterstützung in .NET

Beschreibung

Die Entwicklung einer POS (Point-of-Sale oder Point-of-Service) Applikation stellte die Entwickler vor eine grosse Herausforderung, da jeder Anbieter von POS Geräten, wie zum Beispiel POS Drucker, Barcode Scanner und Kredit/Debit Kartenleser, verschiedene Features, Funktionalitäten und Interfaces zur Verfügung stellte. OPOS ist der erste weit verbreitete POS Gerätestandard. OPOS wurde von Microsoft, NCR, Epson und Fujits-ICL initiiert, um POS Hardware in Applikationen für das Windows Betriebssystem zu integrieren. OPOS verwendet die COM Technologie und ist somit sprachunabhängig.

OPOS via COM

Microsofts .NET COM Interoperabilitätsunterstützung ermöglicht die einfache Integration des OPOS Common Control Objects in .NET Applikationen. Man kann direkt COM Objekte referenzieren und Visual Studio baut automatisch den Interoperabilitäts-Layer auf.

POS für .NET

POS für .NET ist eine Klassenbibliothek, die POS Entwicklern erlaubt, Microsoft .NET Technologien in ihren Produkten zu verwenden. Es wird ein einfaches Interface für .NET Applikationen angeboten, um mit den POS Geräten zu interagieren.

Nachteil

Die letzte Version von POS für .NET ist v1.12 und wurde im Jahr 2008 veröffentlicht. Es gibt diverse Probleme beim Einsatz von POS, die jedoch unter Umständen umgangen werden können:

- POS für .NET funktioniert nicht in einer 64Bit Umgebung
- POS für .NET funktioniert nicht ab .NET 4.0

4.7.6. Schnittstellen Entscheid

Die Verwendung von **COM-Schnittstellen** ist in .NET standardmässig implementiert. Aus diesem Grund bietet sich die Verwendung der virtuellen COM-Schnittstelle an. Es kann auf Events (eingehende Daten) reagiert werden, was die Implementation erleichtert.

5. Software Design

5.1. Systemübersicht

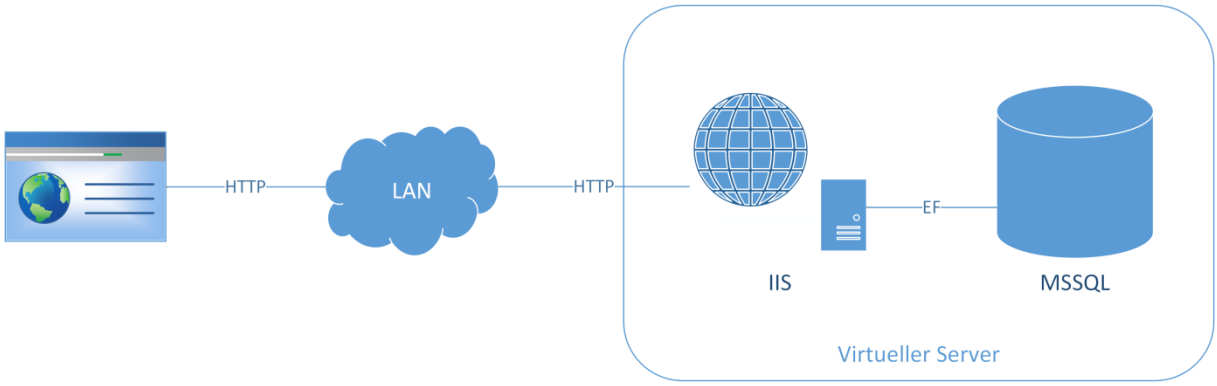


Abbildung 23: Systemübersicht Webapplikation

5.1.1. Client

Der Client wird von den Mitarbeitern der Gema verwendet. Er stellt dem Benutzer alle nötigen Daten für die Verwaltung der Seriennummern zur Verfügung. Die Kommunikation zwischen Client und Server erfolgt wie HTTP.

5.1.2. Server / API

Die zentrale Komponente für die Seriennummernverwaltung stellt der Webserver dar. Er verwaltet die Anfragen der Clients und führt die Manipulationen auf der Datenbank aus.

5.1.3. Datenbank

Die Datenbank ist für die Persistenz zuständig. Sie beinhaltet sämtliche Daten zu Seriennummern, Produkten, Aufträgen (was von ProConcept her verfügbar ist) und Kunden.

5.2. Architektonische Ziele & Einschränkungen

5.2.1. Ziele

5.2.1.1. Erweiterbarkeit

Es soll einfach möglich sein, Erweiterungen zu implementieren, ohne dass alle Clients ein Update benötigen. Durch den Einsatz von webbasierten Technologien müssen Änderungen nur auf den Server deployt werden. Die neuen Features sind für den Client sofort über den Browser verfügbar.

5.2.1.2. Usability

Die Usability steht ganz klar im Vordergrund. Der Benutzer soll eine gewohnte und einfache Bedienung vorfinden, damit im Alltag ein effizientes Arbeiten möglich ist. Ein vorgängiges Paper Protoyping direkt beim Kunden soll etwaige Mängel in den Papierprototypen aufdecken und so die Usability maximieren.

5.2.1.3. Geringe Client-Auslastung

Die bestehenden Clients bei den verschiedenen Produktionslinien verfügen nicht über genügend Ressourcen, um Berechnungen/Abfragen an den Server in kurzer Zeit durchführen zu können. Der Einsatz von webbasierten Technologien gibt uns die Möglichkeit, Berechnungen und Abfragen an die Datenbank direkt auf dem Server auszuführen. Nur die Resultate werden dem Client übermittelt und angezeigt.

5.2.1.4. Security

Der Zugriff auf die Seriennummernverwaltung soll durch eine Authentifizierung gesichert werden. Damit nicht zusätzlich eine Benutzerverwaltung verwaltet werden muss, bietet sich die Authentifizierung über das Active Directory an. Der Vorteil an dieser Lösung ist, dass die Benutzer bereits im Active Directory bestehen. Es müssen nur ein paar zusätzliche Gruppen definiert werden. Dem Benutzer wird so ein einfacher Login-Prozess mit seinen vorhandenen Benutzerdaten geboten und dem Administrator wird die Verwaltung um einiges vereinfacht.

5.2.2. Einschränkungen

5.2.2.1. Zugriff auf lokale Ressourcen

Damit Barcodes / Matrix-Codes eingescannt werden können, ist ein Zugriff auf die lokalen Ressourcen wie zum Beispiel HID, OPOS oder die COM-Schnittstelle nötig. Aus diesem Grund wird ein separater Fat-Client benötigt.

5.2.2.2. Client / Frontend

Die Architektur auf dem Client ist dynamisch, sie kann also ohne Probleme ändern. Da eine Webapplikation erstellt wird, können auf der Clientseite nur browserspezifische Abhängigkeiten entstehen. Mittels verschiedener Scripts wird verhindert, dass Fehler bei Internetexplorer Versionen kleiner als 9 auftreten. Alle Internetexplorer Versionen < 8 werden nicht unterstützt. Dies wurde so entschieden, um Technologien wie Bootstrap und AngularJS gut zu unterstützen, was einen enormen Performancevorteil in der Programmierung bringt.

5.3. Logische Architektur

In folgender Abbildung sind die logischen Abhängigkeiten zwischen den Assemblies dargestellt:

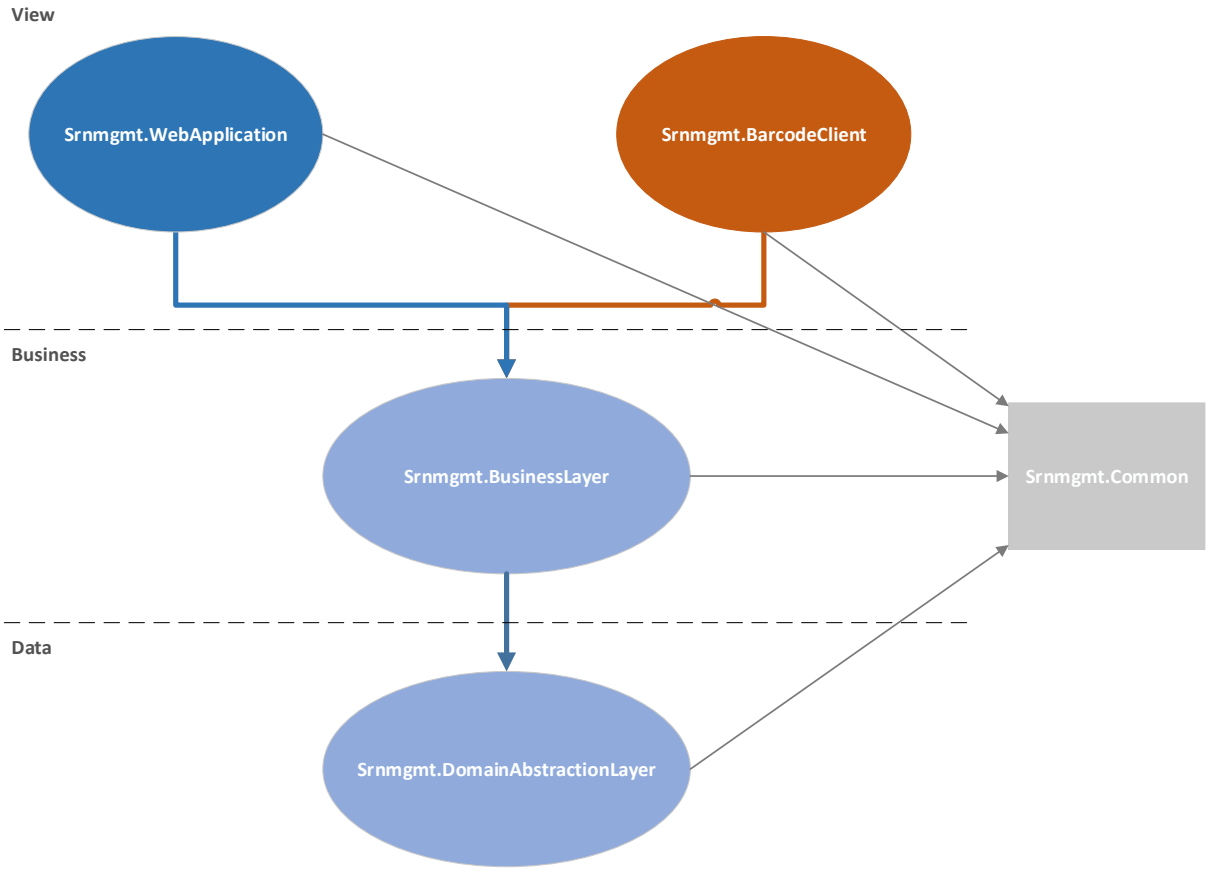


Abbildung 24: Dependencygraph Seriennummernverwaltung

Wie aus der Grafik ersichtlich ist, wurde die Anwendung in verschiedene Schichten aufgeteilt. Diese sind im Wesentlichen View, Business sowie Data. Die Kommunikation läuft immer von oben nach unten. Es bestehen keine zirkulären Abhängigkeiten und die Kopplung kann entsprechend tief gehalten werden.

Eine Ausnahme bildet die Assembly Common, welche eine sehr hohe Kopplung aufweist. In diesem Assembly sind hauptsächlich die Datentransferobjekte enthalten. Diese Kopplung wurde bewusst in Kauf genommen. Da die Klassen die Daten halten, bleiben sie ab einem gewissen Zeitpunkt stabil. Und falls sich die Daten der Applikation ändern sollten, hat dies meist auch Auswirkungen auf alle Layer.

Die einzelnen Layer werden in folgenden Abschnitten genauer beschrieben.

5.3.1. WebApplication

Der Applicationlayer enthält die eigentliche Benutzerapplikation. Die ASP.NET MVC4 Applikation enthält die Frontend Controller sowie die API Controller. Die ViewController liefern die kompilierte „CSHTML" Seite an den Client und fügen so Lokalisierungen und dynamische Inhalte ein. Die API Controller liefern serialisierte Daten aus und ermöglichen so ein dynamisches Nachladen von Inhalten.

Als Designprinzip wurde hier eine spezielle Trennung von Back- und Frontendrendering gewählt. Alles was beim Aufruf der Seite klar definiert ist, wird von der Microsoft RAZOR Engine erledigt, zum Beispiel die Lokalisierung. Sämtliche dynamischen Inhalte wie Produkte, Kategorien, etc. werden von AngularJS dynamisch nachgeladen und angezeigt.

5.3.2. BarcodeClient

Der Barcode Client besteht aus einer View und einem ViewModel. Bis auf die Zuweisung des „DataContext" wurde darauf verzichtet, Implementationen im Code Behind vorzunehmen. Die ganze Logik der View befindet sich im ViewModel. Durch den Einsatz des MVVM Patterns wird eine klare Trennung der Aufgaben (separation of concerns) gewährleistet. Das ViewModel und Model ist so einfacher in Isolation zu testen. Durch die damit erreichte lose Kopplung hat man vor allem Vorteile in Bezug auf die Wartbarkeit und Erweiterbarkeit.

Der Barcode Client greift über den Businesslayer auf die Datenbank zu, um Produkte und Aufträge zu verifizieren und deren Zuordnung zu speichern. Ebenfalls benötigt er den Zugriff auf den Commonlayer, um auf die DataTransferObjects (DTOs) zugreifen und Methodextensions verwenden zu können.

5.3.3. Businesslayer (BL)

Diese Schicht ist zuständig für die gesamte Businesslogik. Hier werden die Zugriffe auf die Datenbank ausgeführt und die neuen Objekte erstellt. Will der Barcodescanner oder die Webapplikation ein neues Produkt anlegen, so wird dies über eine entsprechende Funktion im Businesslayer gekapselt.

Im Businesslayer ist auch die gesamte Validation angesiedelt. Die Businesskomponente überprüft die einzelnen Objekte auf deren Vollständigkeit und meldet Fehler mittels einer Exception.

Mittels Exceptions können Spezialfälle sowie auch „Doppel-Returns"[5] vermieden werden. Tritt in der Datenbank oder bei einer DB-Operation ein Fehler auf, so wird eine entsprechende Exception geworfen. Diese muss dann im Backend der Webapplikation beziehungsweise im Barcodescanner aufgefangen und korrekt verarbeitet werden.

[5] Damit ist gemeint, dass eine Funktion nicht zwei verschiedene Typen retournieren kann. Also beispielsweise, wenn die Funktion „bool" zurückliefern sollte, jedoch im Fehlerfall einen „string" retournieren will.

5.3.4. Domainabstractionlayer (DAL)

In diesem Layer wird die Datenbank abgebildet. Der Layer enthält lediglich das Entity-Framework Modell. Darin sind sämtliche Entities, Assotiationen und Funktionen abgebildet. Der Businesslayer greift auf das Entity-Framework (EF) zu und verwaltet so die Datenbankobjekte.

Der Ansatz dazu ist „Code-First“, dabei wird zuerst das Modell im EF erstellt und daraus dann die Datenbank mit allen Relationen erstellt. Leider kann dies nicht vollständig umgesetzt werden, da beispielsweise „User Defined Functions“ (UDF) und „Stored Procedures“ (SProc) nicht über den Code-First Ansatz definiert werden können. Diese müssen eigens mittels eines SQL Scripts in der Datenbank angelegt werden.

Aus diesem Grund wird zum Schluss des Projektes ein ganzheitliches SQL-Skript erstellt, welches die ganze Datenbank aufbaut, die erforderlichen Benutzer registriert und die verwendeten UDFs erstellt.

5.3.5. Commonlayer (CL)

In dieser Softwareschicht sind alle gemeinsam genutzten Objekte enthalten. Hier werden die DataTransferObjects (DTOs) abgelegt. Der Commonlayer soll als Unterstützung für alle anderen Layers genutzt werden. Der DAL nutzt die hier enthaltenen Enums für die Entities, der BL nutzt die DTOs und die Webapplikation die DTOs und gewisse Helperfunktionen. Somit ist dieser Layer in den meisten der anderen Layer verknüpft. Wichtig dabei ist es, darauf zu achten, dass keine zirkulären Referenzen entstehen.

5.3.6. Testing

In der Anwendung sind noch weitere Projekte vorhanden, die für das Testing benötigt werden. Diese garantieren einen reibungslosen Ablauf der Applikation. Getestet wird vor allem die Hintergrundfunktionalität (BL, CL, DAL, sowie API Controller), um das Funktionieren der Software zu garantieren (Unit-Tests). Diese Assemblies wurden nicht in die Übersicht aufgenommen.

5.3.7. Wichtige Schnittstellen

View -> Business: Sämtliche Zugriffe auf den Business Layer von einer darüber liegenden Schicht geschehen über Interfaces. Die eigentliche Implementation bleibt so verborgen und die Unit-Tests können gegen die Interfaces geschrieben werden. Die View-Klassen, die einen Service-Zugriff brauchen, programmieren ebenfalls gegen das Interface. Mit dem Ansatz der „Dependency Injection“ wird die konkrete Implementation jeweils in den Konstruktor „injiziert“.

Business -> Data: Der Businesslayer enthält die Methoden, um von einem Entity Objekt in die entsprechenden DTO’s zu konvertieren und umgekehrt. Diese Funktionalität wurde mithilfe von Methodextensions gelöst.

5.4. Wichtige Systemabläufe

5.4.1. Neues Produkt für einen Auftrag erfassen

In der Auftragsübersicht eines spezifischen Auftrags hat der Benutzer die Möglichkeit, dem Auftrag ein neues Produkt hinzuzufügen. Dazu klickt er auf den Button „Produkt hinzufügen“ und nachfolgend wählt er „Neues Produkt hinzufügen“ aus. Anschliessend wird er auf die neue Seite weitergeleitet, auf der das Produkt erfasst wird. Der Benutzer muss hier alle relevanten Daten des Produktes eingeben, dazu gehören die Kategorie, der Produkttyp, ob es sich um einen Mehrfacheintrag handelt und die Seriennummer. Zusätzlich können je nach Produkttyp noch weitere Attribute vorhanden sein, die ausgefüllt werden müssen. Mit einem Klick auf „Speichern“ bestätigt der Benutzer die Eingaben. Nach erfolgreicher Verifikation wird das Produkt erstellt und dem Auftrag zugewiesen. Dem Benutzer wird nun das soeben erstellte Produkt angezeigt.

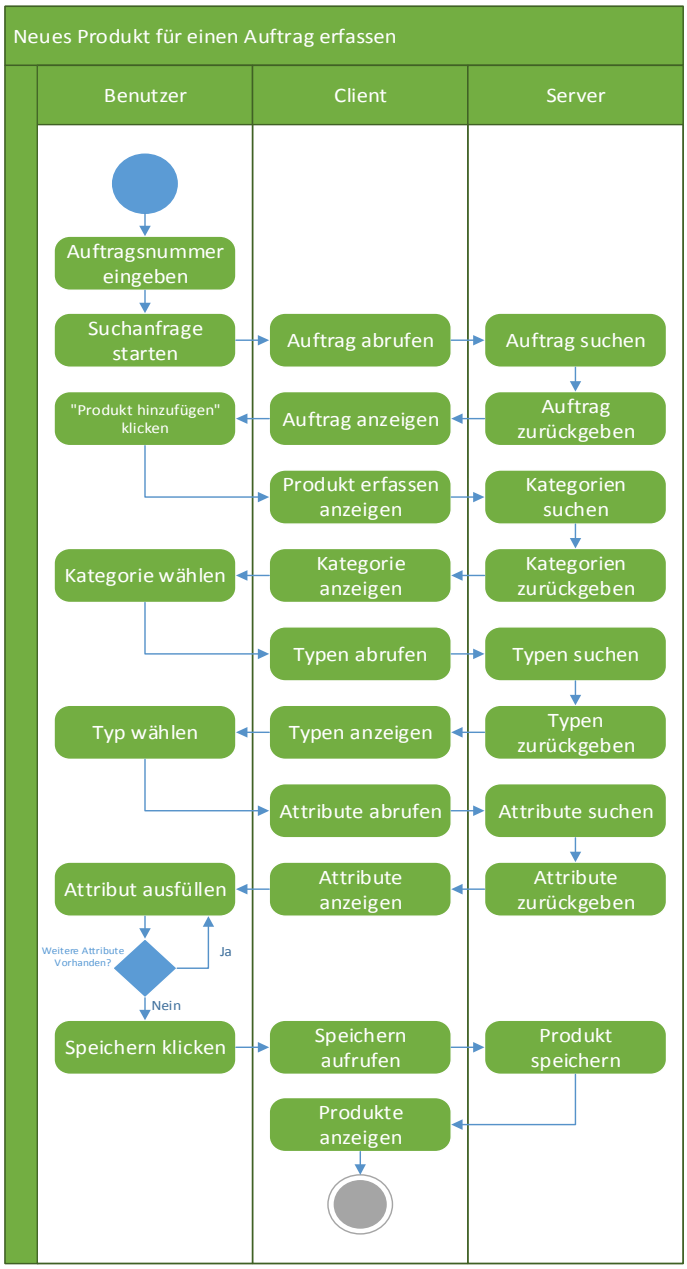


Abbildung 25: Ablaufdiagramm „neues Produkt für einen Auftrag erfassen“

5.4.2. Bestehende Produkte einem Auftrag hinzufügen

Bereits existierende Produkte können in der Auftragsübersicht über den Button „Produkt hinzufügen“ und anschliessender Auswahl von „Bestehendes Produkt hinzufügen“ einem Auftrag hinzugefügt werden. Nachdem sich das Overlay geöffnet hat, muss erst die Kategorie und dann der Produkttyp ausgewählt werden. Danach erscheinen in der linken Spalte die verfügbaren Produkte. Nach der Auswahl der gewünschten Produkte können diese mit dem Button „>“ in die rechte Spalte zu den verknüpften Produkten verschoben werden. Durch die Bestätigung mit dem „Speichern“ Button wird die Verknüpfung der Produkte zum Auftrag durchgeführt und das Overlay wird geschlossen. Nun ist wieder die Auftragsübersicht mit der aktualisierten Produktetabelle ersichtlich.

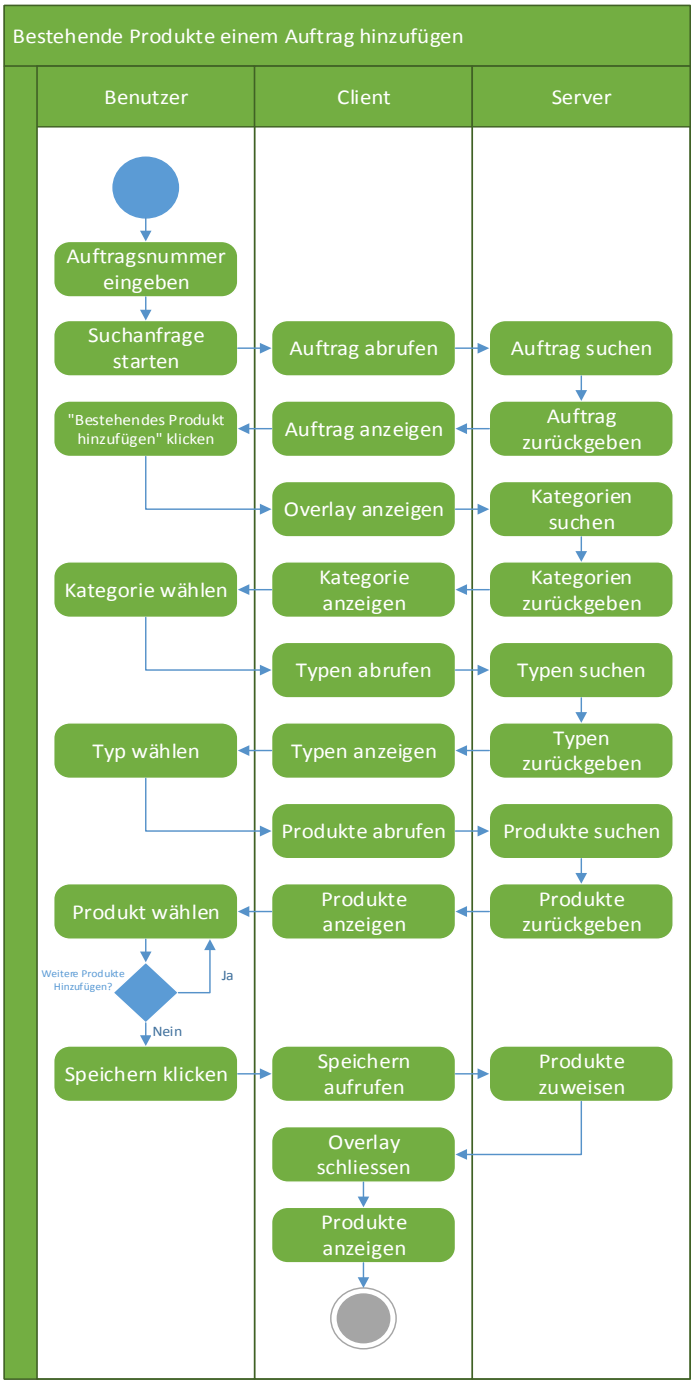


Abbildung 26: Ablaufdiagramm für „bestehendes Produkt zum Auftrag hinzufügen“

5.5. Lokalisierung

5.5.1. Grundlagen

Eine ASP.NET Webapplikation wird grundsätzlich über „Resources“ lokalisiert. Diese Ressourcen sind „.resx“ – Dateien, welche eine simple Key-Value Struktur haben. Die Nutzung dieser Ressourcen ist sehr einfach und basiert auf den Browsereinstellungen. Im Header des Browserrequests wird die gewünschte Sprache mitgeliefert und wenn diese gefunden wird, kann dem Benutzer seine geforderte Sprache angezeigt werden. Ist jedoch kein Resx zur verlangten Sprache vorhanden, so wird die Standardsprache verwendet. [3]

Von der Gema GmbH kam der Wunsch nach Lokalisierung mit der Einschränkung, sich auf Deutsch und Englisch festzulegen. Aus diesem Grund werden gewisse Lokalisierungen über die Datenbank gelöst und sind nicht im Resx vorhanden. Im Folgenden wird beschrieben, wie die Lokalisierung umgesetzt ist und was wie lokalisiert wurde.

Die gesamte Lokalisierung wird von der RazorEngine[6] verwaltet. Diese Templating-Engine parst die .cshtml Dateien und fügt die entsprechende Sprache ein. Die Engine ist im Standardaufbau einer ASP.NET MVC4 Applikation dabei und muss nicht extra installiert werden.

5.5.2. Sprachinformationen

Die gewünschte Sprache wird vom Browser aufgrund der lokalen Windows-Spracheinstellung übermittelt. Ein solcher Request sieht folgendermassen aus:

```
GET /srnmgmt HTTP/1.1
Host: localhost
Connection: keep-alive
Cache-Control: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/27.0.1453.94 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
```

Zwecks Lesbarkeit wurde die Autorisierung aus dem Request gelöscht.

Der für die Lokalisierung relevante Part ist die „Accept-Language“ – Sektion. Sie bestimmt, welche Sprache der Browser gerne hätte. Diese Sprachinformationen werden von der Webapplikation ausgelesen und zwischengespeichert. Ist die verlangte Sprache vorhanden, so wird diese dem Benutzer zurückgegeben. Ist die Sprache nicht vorhanden, so wird die Standardsprache Deutsch verwendet. Die Standardsprache wurde in Absprache mit der Gema auf Deutsch gesetzt, da die meisten Benutzer einen deutschen Sprachhintergrund haben.

[6] https://github.com/Antaris/RazorEngine

Sobald der Benutzer auf die Webapplikation zugreift, wird die verlangte Sprache in die Session eingetragen:

```
protected void Application_AcquireRequestState(object sender, EventArgs e)
{
    if (HttpContext.Current.Session == null) return;
    var ci = Session["Culture"] as CultureInfo;
    if (ci == null)
    {
        var lang = "de";
        if (HttpContext.Current.Request.UserLanguages != null &&
            HttpContext.Current.Request.UserLanguages.Length != 0)
        {
            lang = HttpContext.Current.Request.UserLanguages[0].Substring(0, 2);
        }
        ci = new CultureInfo(lang);
        Session["Culture"] = ci;
    }
    Thread.CurrentThread.CurrentUICulture = ci;
    Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(ci.Name);
}
```

Der Benutzer kann jedoch die Sprache umstellen, indem er in der Webapplikation mittels des Sprache-Buttons diese ändert:

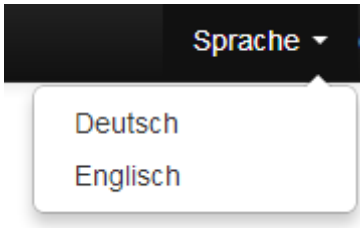


Abbildung 27: Button zum Wechseln der Sprache

Diese beiden Buttons ermöglichen es dem Benutzer, die Sprache zu wechseln. Der Benutzer steuert durch diese Links einen bestimmten „SettingsController“ an, welcher dann die vorhandene „CultureInfo“ in der Session mit der neuen überschreibt:

```
[GET("changeculture?{lang}")]
public ActionResult ChangeCulture(string lang)
{
    Session["Culture"] = new CultureInfo(lang);
    return Redirect(Url.Action("Index", "Job"));
}
```

Ist diese CultureInfo gültig und das Resx dazu vorhanden, so wird der Benutzer die neue Sprache geliefert bekommen.

5.5.3. Resx – Ressourcen

In den sogenannten „Resx“-Files sind die Sprachen abgelegt. Diese werden als kompilierte DLLs zur Applikation geliefert und geladen. Wird eine neue Sprache eingebunden, so muss die entsprechende DLL (auf dem Server) ausgeliefert werden. Es ist notwendig, die Applikation neu zu verteilen, jedoch geschieht dies nur auf dem IIS-Server und nicht mehr auf allen Clients. [4]

Die Zuweisung der Sprache zum spezifischen Assembly geschieht über die Benennung der Resx Datei. Die Standardsprache wird einfach über den Namen und die Endung „.resx“ definiert (im Falle der Webapplikation „SharedStrings.resx“). Jede weitere Sprache enthält einen Namen nach folgendem Schema: <<Name der Ressource>>.<<Sprachcode>>.resx. Die englische Sprachdatei der Webapplikation heisst somit: „SharedStrings.en.resx“.

5.5.4. Datenbankgelagerte Lokalisierung

Grundsätzlich wurde so viel wie möglich über die erwähnten Resx Dateien lokalisiert. Diese Ressourcen können dynamische Inhalte wohl lokalisieren, jedoch wird stark davon abgeraten, da die Applikationsdateien manipuliert werden [5]. Um unsere dynamischen Inhalte zu lokalisieren, wurde deswegen ein spezieller Weg mittels Serialisierung gewählt.

In Absprache mit der Gema wurde festgelegt, dass keine neuen Sprachen ausser Deutsch und Englisch hinzugefügt werden. Dies soll jedoch nicht der Anreiz einer hart codierten Lösung sein, weshalb eine dynamische Lösung gefunden werden musste. Die dynamischen Inhalte, welche lokalisiert werden müssen (Produktkategorie, Produktattribut, Reportattribut) enthalten ein spezielles Konstrukt für die Namensgebung. Wird ein solches Objekt (in der Webapplikation) erstellt, so muss ein Name für jede Sprache (DE, EN) angegeben werden. Im Backend wird diese Liste von Namen in JSON serialisiert und in der Datenbank als String gespeichert. Nachfolgend wird ein kleines Beispiel mit der C# Liste und dem Datenbankäquivalent aufgezeigt:

```
var names = new Dictionary<string, string>();
names.Add("de", "Pistole");
names.Add("en", "Powdergun");
```

wird zu:

```
{"de":"Pistole","en":"Powdergun"}
```

Der Vorteil bei der Serialisierung in JSON ist, dass die Sprachauswahl dynamisch bleibt. Es ist also weiterhin möglich, eine weitere Sprache einzupflegen.

5.5.5. JavaScript Lokalisierung

Ein weiteres Problem bei der Lokalisierung mit Hilfe der RazorEngine stellt die Lokalisierung von JavaScript Variablen dar. Für gewisse Elemente wie Bestätigungsformularen oder Ähnlichem muss JavaScript Code verwendet werden, um nicht den Server unnötig zu belasten. Diese Elemente beinhalten Text und dieser soll in verschiedenen Sprachen verfügbar sein. Es gibt mehrere Ansätze bei der Lokalisierung von JavaScript Variablen:

- Die lokalisierten Werte in einem Script-Tag einfügen
- „Hidden“ – Divs mit den lokalisierten Werten einfügen
- Lokalisierte Werte per AJAX vom Server nachladen

Der für die Webapplikation praktikabelste Weg ist die Variante mit dem Script-Tag. Es handelt sich um einige wenige Variablen, welche eingefügt werden müssen, weswegen diese in einem Script-Tag eingebunden werden. Die Vorteile dieser Methode sind:

- Weniger Overhead im Vergleich zur DIV Variante
- Keine unnötigen Anfragen an den Server
- RazorEngine kann die Lokalisierung vornehmen (sehr schnell)

Auf diese Weise wurden folgende Elemente lokalisiert:

- Nachrichten für den Benutzer (Fehlermeldungen, Warnhinweise), welche nicht statisch vorhanden sind, also durch den NotificationManager hervorgerufene Meldungen
- Bestätigungsnachrichten (Löschen, Entfernen)
- Pagination-Links

Dieser spezielle Script-Tag wurde im Basislayout für alle .cshtml Dateien angelegt und sieht folgendermassen aus:

```
<script>
    appLanguage = "@Culture.Substring(0,2)";
    baseLocation = @Html.JsEncode(AppSettings.BaseLocation);
    apiLocation = @Html.JsEncode(AppSettings.ApiLocation);
    lblPaginationFirst = @Html.JsEncode(@ViewRes.SharedStrings.lblPaginationFirst);
    ...
</script>
```

5.5.6. Enum Lokalisierung

Um die Enums nicht mehrfach führen zu müssen (Datenbank, Businesslayer, View), wurde entschieden, sie in das Common Assembly auszulagern. Um auch die Werte eines Enums zu lokalisieren wurde in der WebApplication unter Helpers die Klasse EnumHelper.cs hinzugefügt. Mit dieser Hilfsklasse ist es nun möglich, die entsprechenden Sprachwerte für einen Enum-Key in einem Resx-File abzulegen.

Bei den Methodenaufrufen muss das entsprechende Ressourcen File angegeben werden. Dieses wird danach nach folgendem Muster durchsucht: „Enum-Name“.„Enum-Key“. Falls ein Eintrag vorhanden ist, wird der Value davon verwendet, ansonsten wird lediglich der Enum-Key verwendet.

Folgendes Enum ist gegeben:

```
public enum PowerFrequency : byte
{
    Frequency_50_Hz = 0,
    Frequency_60_Hz = 1
}
```

Um nun mit der Hilfsmethode die lokalisierten Werte auszulesen, müssen folgende Einträge in einem Ressourcen File erfasst werden:

	PowerFrequency.Frequency_50_Hz	50 Hz
	PowerFrequency.Frequency_60_Hz	60 Hz

Abbildung 28: Enumlokalisierung im .resx File

Der Befehl dazu lautet wie folgt:

```
EnumHelper.GetEnumValuesAsKeyValuePair<PowerFrequency>(typeof(@ViewRes.SharedStrings));
```

5.6. Benutzerinterface

5.6.1. AngularJS

AngularJS ist ein JavaScript-Framework, welches das MVVM (Model – View – ViewModel) Designpattern im Frontend realisiert. Dem Frontenddesigner wird somit ermöglicht, seinen UI-Code zu entkoppeln. Die Kommunikation zwischen Controller (ViewModel) und dem GUI (View) basiert auf sogenannten „Bindings“. Diese Bindings übernehmen das Speichern von Werten in die Variablen des ViewModels oder das Aktualisieren der GUI-Werte, wenn sich der Wert im ViewModel ändert. Dieses Prinzip wird von Microsoft mit den WPF-Bindings schon länger verwendet. Einer der grossen Vorteile dieses Designpatterns ist die Testbarkeit des ViewModels. Man kann so indirekt die GUI-Logik testen.

Nachfolgend ist ein kleines „Hello World“ Beispiel eines AngularJS Scripts, welches von der Homepage von AngularJS übernommen wurde [6], beschrieben:

```
<!doctype html>
<html ng-app>
  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
  </head>
  <body>
    <div>
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <h1>Hello {{yourName}}!</h1>
    </div>
  </body>
</html>
```

AngularJS verbindet hier die gesamte Seite mit einem Standardkontroller, welcher die einzelnen Variablen enthält. Sobald der Benutzer einen Wert in das Input-Feld füllt, wird die Variable im Kontroller angelegt, falls sie nicht vorhanden ist und danach aktualisiert. Sobald die Variable `yourName` einen Wert enthält oder sich dieser ändert, wird die Änderung übernommen und der neue Wert im Titel unter dem Inputfeld angezeigt.

Dieses Konstrukt alleine würde jedoch nicht die Freiheiten eines C#-WPF-MVVM Projektes ermöglichen, weshalb mittels hinterlegten „Kontrollern“ mehr Funktionalität eingebaut werden kann [6]:

```
<!doctype html>
<html ng-app>
  <head>
    <script src=".../angular.min.js"></script>
  </head>
  <body>
    <div ng-controller="testCtrl">
      <label>Name:</label>
      <input type="text" ng-model="yourName" placeholder="Enter a name here">
      <hr>
      <button ng-click="greetMe()">Greet Me!</button>
    </div>
    <script>
        function testCtrl($scope){
            $scope.greetMe = function(){
                alert('Hello ' + $scope.yourName);
            };
        }
    </script>
  </body>
</html>
```

In obigem Beispiel wird ein benutzerdefinierter Kontroller an die View angehängt und mit einer Funktion erweitert. Diese Funktion ist nun in der View nutzbar. AngularJS verlinkt alles über Dependency Injection (DI), somit wird die JS Funktion `function testCtrl($scope){/*…*/}` automatisch als Kontroller für den umgebenden DIV der View eingesetzt. Wird nun der Button „Greet Me!“ geklickt, so wird die Funktion `greetMe()` im Kontroller ausgeführt, welche einen JS-Alert mit dem Namen in der `yourName` Variable ausgibt.

AngularJS ist somit ein sehr gutes Framework, wenn der Programmierer sich bereits mit dem MVVM Pattern auskennt. AngularJS wurde aus folgenden Gründen verwendet:

- Schnell zu erlernen (vor allem wenn Kenntnisse in C# / WPF / MVVM vorhanden sind)
- Entkoppeln der GUI Logik von den Kontrollern
- Vereinfachtes Ressourcenmanagement (API Ressourcen)
- Wiederverwendbarkeit von Elementen wird gefördert
- Das Team wollte sich mit AngularJS auseinandersetzen, da es von Google für Google Mail verwendet wird

5.6.1.1. Grundkonzept Srnmgmt

Da die Konzepte von AngularJS nahe an den Prinzipien des MVVM Patterns von Microsoft liegen, wird es in diesem Projekt verwendet. Da jedoch die Hello-World Beispiele verständlicherweise nicht ausreichen, wurden gewisse Grundkonzepte für die Nutzung in der Seriennummernverwaltung festgelegt. Diese Grundkonzepte umfassen:

- Code Guidelines für Controller
- Auslagerung von Direktiven und Ressourcen
- Nutzung eines Ressourcenservices
- Nutzung von wiederverwendbaren Direktiven

Code Guidelines

Damit der Code für einen projektfremden Programmierer nicht zum unüberschaubaren Desaster wird, wurden folgende Guidelines eingehalten:

```
angular.module('testApp', ['dependency']).
    controller('testCtrl', ['$scope', 'injectedVar', function ($scope, injectedVar) {
        //variables

        //init

        //events

        //public functions

        //private functions
    }]);
```

Diese Darstellung wurde für alle Kontroller verwendet und macht den Kontrollercode übersichtlich. Die spezielle Art und Weise, wie der Kontroller erstellt wird (über die `controller(…)` Funktion des AngularJS Modules) sorgt dafür, dass der Code die Minification überlebt. Dies wird in einem späteren Abschnitt beschrieben.

5.6.1.2. Services

In diesem Abschnitt werden die einzelnen, programmierten Services beschrieben. Einzige Ausnahme dabei sind die Ressourcenservices, welche in einem eigenen Abschnitt erläutert werden, da sie eine zentrale Rolle bei der Kommunikation mit der API darstellen. Alle Services sind in der Datei „./Scripts/Shared/srnmgmtServices.js“ abgelegt und in einzelne Module verpackt, damit sie als alleinstehende Abhängigkeit geladen werden können. Zentral dabei ist: die Services registrieren sich selbst im `$rootScope`, damit die Kontroller direkt über die Servicevariable darauf zugreifen können, ohne die Services erst selbst in einer Variable zu registrieren.

LocalizationService

Dieser Service ermöglicht das Auslesen von sprachspezifischen Attributen eines Objekts. Der Service ist relativ klein, jedoch wird er an vielen Stellen verwendet. Der Service enthält die folgenden Elemente:

Name	Typ	Beschreibung
appLanguage	private variable	Diese Variable enthält die aktuelle „CultureInfo“. Diese wird beim Aufbau der Seite in eine JavaScript Variable eingefügt. Ist diese Variable nicht vorhanden, so wird „de“ als Standardwert verwendet.
getAppLanguage()	public function	Diese Funktion liefert lediglich den Wert der `appLanguage` Variablen.
getLocalizedName(obj)	public function	Dies ist die zentrale Funktion des Services. Man kann ihr ein datenbanklokalisiertes Objekt übergeben und erhält den lokalisierten Wert. Wird beispielsweise ein Produktattribut mit Name `{"de":"Pistole","en":"Powdergun"}` übergeben, so wird bei der deutschen Spracheinstellung „Pistole“ zurückgeliefert. Ist die Sprachvariable nicht gesetzt oder hat das Objekt kein korrespondierendes Namensfeld, so wird automatisch der Standardwert „de“ bzw. das ganze Objekt zurückgeliefert.

Tabelle 25: Elemente des Localizationservices

Dieser Service ist zentral für alle datenbanklokalisierten Objekte wie:

- Produkttyp
- Produktattribut
- Produktkategorie

DownloadService

Der DownloadService ist das zentrale Element für alle Downloadaufgaben. Er erstellt den einzigartigen Token für den Download und verarbeitet die Antworten des Servers. Die Elemente des Services sind:

Name	Typ	Beschreibung
options	private variable	Die Variable `options` enthält die Optionen für den angezeigten Benutzerdialog. Dieser Dialog blockiert das UI für sämtliche anderen Interaktionen. Zweck dieses Locks ist, dass der Benutzer keine Interaktion während des Downloads machen kann.
openDialogs	private variable	In diesem Array sind alle offenen Dialoge gespeichert. Dies dient dem Management und der Vermeidung von Fehlern bei mehreren Downloads.
checkDownloadProgress(token)	private function	In dieser privaten Funktion findet das Polling statt. Es wird einmal pro Sekunde überprüft, ob ein Cookie mit dem angegebenen Downloadtoken übertragen wurde. Ist dieses Cookie in der Session enthalten, so wird der UI-Lock aufgelöst und der Benutzer kann weiterarbeiten, da sein Download beendet ist.
deleteDownload(token)	private function	Funktion um einen offenen Downloaddialog zu schliessen und das Cookie zu entfernen
prepareDownload(path)	private function	Funktion, welche den zufälligen Token erstellt und den Pfad präpariert. Returniert wird ein Objekt, welches beide Eigenschaften enthält.
downloadFile(path)	public function	Die zentrale Funktion, welche den GET-Downloadprozess anstösst. Es wird ein `downloadToken` generiert, welcher dann an den Server übermittelt wird. Danach blockiert die Funktion das GUI und startet den Download und den Cookieprüfer.
downloadFileOverPost(path, data)	public function	Muss eine Datei über die HTTP POST Methode abgerufen werden, so muss eine spezielle Technik genutzt werden. Der Browser kann eine Datei nicht über POST herunterladen. Genutzt wird diese Methode, falls zum Beispiel die Parameter zu lange für einen GET-Abruf werden.

Tabelle 26: Elemente des Downloadservices

Ablauf des GET-Downloads:

1. Anfrage mit einem zufälligen Token an den Server senden
2. Service blockiert das UI
3. Server verarbeitet die Anfrage
4. Service prüft jede Sekunde, ob ein Cookie mit dem Token auftaucht
    a. Wenn ja: Download ist abgeschlossen
    b. Wenn nein: Download läuft noch
5. Service beendet den UI-Lock

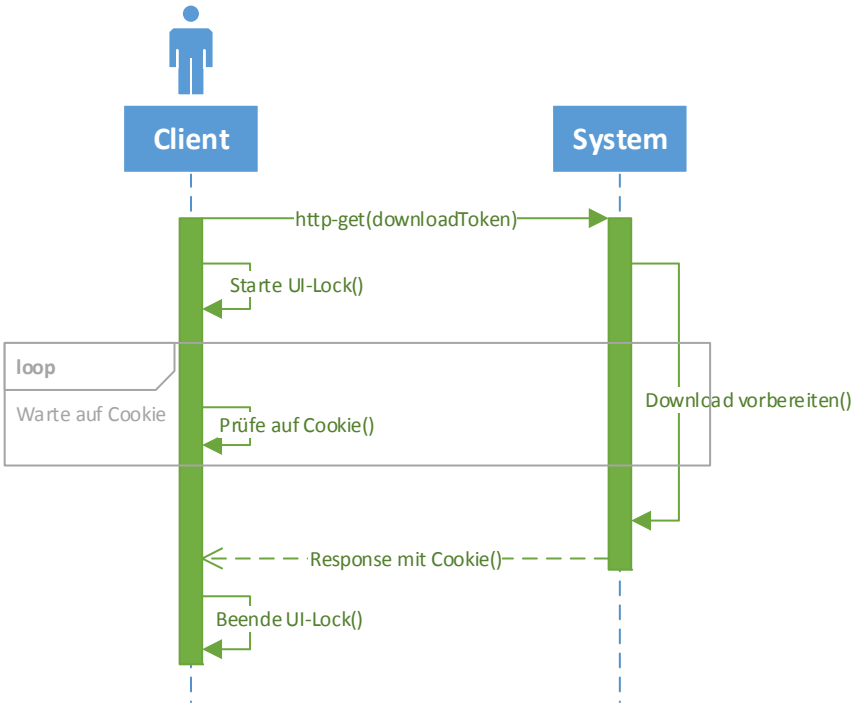


Abbildung 29: GET-Downloadprozess des Services

Ablauf des POST-Downloads:

1. POST-Anfrage mit einem zufälligen Token an den Server
2. Service blockiert das UI
3. Service wartet auf die Beendigung der Anfrage
    a. Erfolgreich:
        i. UI-Lock aufheben
        ii. Dateidownload über GET starten (mit demselben Token)
        iii. Download aus der Liste entfernen (auf dem Server)
    b. Error:
        i. UI-Lock aufheben
        ii. Weiterleitung auf Errorseite

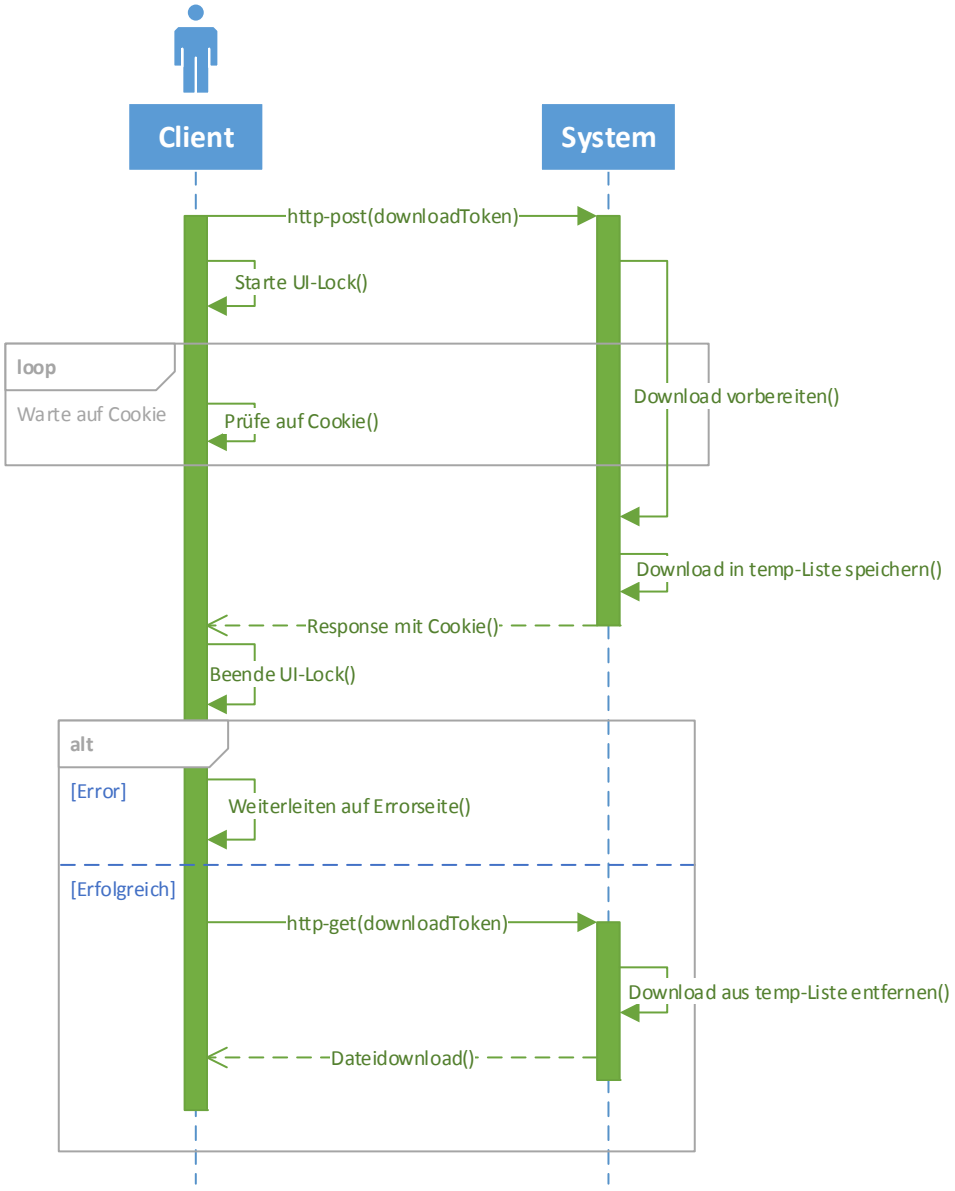


Abbildung 30: POST-Downloadprozess des Services

Dieser spezielle Downloadprozess speichert den durch die POST-Methode generierten Filestream in einer Key/Value Liste. Ist der Schlüssel bereits in der Liste eingetragen, so wird eine Exception geworfen. Sobald die POST-Anfrage den Download bereitgestellt hat, wird er in diese Liste eingetragen und die Anfrage sendet das Cookie zur Beendigung des Prozesses. Sobald der Client das Cookie entdeckt, wird der UI-Lock beendet und der Client startet dieselbe Anfrage nochmals, jedoch über HTTP GET anstatt POST. Da die Anfrage mit demselben Token kommt, sucht der Server in der Downloadliste (welche im Kontroller implementiert ist) nach dem Token und retourniert den entsprechenden Filestream. Sobald der Stream ausgeliefert wurde, löscht der Server den gelieferten Eintrag in der Liste. Um Concurrency-Probleme zu vermeiden, wird die Liste bei den Zugriffen gelockt.

Es ist theoretisch möglich, dass ein POST ohne anschliessendes GET ausgeführt werden kann (wenn nach dem POST Vorgang der Browser beendet wird). Das Problem ist dann, dass der Server die entsprechenden Einträge nicht löscht. Jedoch wurde diese Verfahrensweise genutzt, da:

- Das Risiko sehr klein ist
- Es ein einfacher und effizienter Ansatz ist
- Aus Zeitgründen keine umfangreichere Option genutzt werden konnte

Sollte es jedoch, entgegen den Erwartungen, zu gehäuften Fehlern kommen, so muss das Konzept überarbeitet werden. Möglich wäre ein eigener Thread, welcher alle fünf Minuten die Liste komplett leert. Weiter wäre eine Managementklasse denkbar, welche die Liste und das Listenmanagement übernimmt.

NotificationService

Der NotificationService enthält alle Elemente, welche zur Kommunikation mit dem Benutzer notwendig sind. Dazu gehören eine kleine Direktive, zwei Animationen und der Service selbst. In diesem Modul wurde explizit die Direktive eingefügt, da es sich um keine wiederverwendbare Direktive handelt. Sie ist nur im Zusammenhang mit dem Service verwendbar. Der Service hat zwei Funktionen:

- Dem Benutzer beständige Fehlermeldungen anzeigen
- Dem Benutzer flüchtige Meldungen anzeigen (Info, Warning, etc.)

Der Service enthält folgende Elemente:

Name	Typ	Beschreibung
alerts	private variable	Diese Variable enthält alle offenen Nachrichten für den Benutzer. Die Nachrichten, welche in dieser Variablen gespeichert sind, müssen vom Benutzer geschlossen werden.
shortNotification	private variable	Die Variable enthält das Objekt, welches eine Kurznachricht darstellt.
getAlerts	public function	Diese Funktion liefert den Array mit den Notifikationen zurück.
closeAlert(index)	public function	Diese Funktion wird vom Benutzer aufgerufen, wenn er auf das [X] der Nachricht klickt. Sie schliesst die Meldung und löscht die Nachricht aus dem Array.
createAlert(type, msg)	public function	Muss dem Benutzer eine beständige Notifikation (die der Benutzer wegklicken muss) angezeigt werden, zum Beispiel bei einem Fehler, so wird diese Funktion genutzt. Man kann den Typen der Nachricht und die Nachricht selbst angeben. Die Typen sind: • „info“ • „warning“ • „success“ • „error“
getShortNotification()	public function	Liefert die Variable `shortNotification` zurück.
showShortNotification(type, msg)	public function	Wird dem Benutzer eine Kurznachricht angezeigt (Nachricht, welche nach 5 Sekunden selbstständig verschwindet), so muss diese Funktion aufgerufen werden. Die Parameter der Funktion sind dieselben wie die der `createAlert(type, msg)` Funktion.

Tabelle 27: Elemente des Notificationservices

DataDisplayService

Der DataDisplayService hat zwei Aufgaben: erstens realisiert er die Pagination und zweitens die Sortierung von Tabellen. Dieser Service wird bei allen Tabellen verwendet und ist somit ein zentrales Element bei der Darstellung von Daten. Der Service enthält die folgenden Elemente

Name	Typ	Beschreibung
sortVariables	private variable	Enthält die Variablen für die Sortierung. Zu diesen Variablen gehören: • `sortColumn`: Die aktuell sortierte Spalte • `reverse`: Information, ob die Spalte auf- oder abwärts sortiert ist
paginationVariables	private variable	Enthält, analog zur `sortVariables` Variable, die Eigenschaften für die Pagination. Zu diesen Eigenschaften zählen: • `elements`: Die aktuellen Elemente der Tabelle • `currentPage`: Die aktuelle Seite • `pageSize`: Die maximale Grösse einer Seite • `maxSize`: Die maximale Länge der Pagination-Bar
sorting	public variable	Öffentliche Variable, welche als Kapselung für die Funktionen der Sortierung fungiert. Die Funktionen sind: • `getSortColumn()`: Retourniert die aktuell sortierte Spalte • `setSortColumn(col)`: Setzt die zu sortierende Spalte • `getReverse()`: Prüft, ob die Spalte auf- oder abwärts sortiert ist • `isSorted(col)`: Prüft, ob die angegebene Spalte sortiert ist
pagination	public variable	Öffentliche Variable für die Funktionen der Pagination, welche sind (jeweils get/set sind in einem Punkt beschrieben, da es sich nur um das Auslesen, bzw. Schreiben der Variable handelt): • `setElements(arr)`: Setzt die Daten der Pagination neu • `getNumberOfPages()`: Retourniert die Anzahl der benötigten Seiten • `CurrentPage`: Liest / Schreibt die aktuelle Seite • `PageSize`: Liest / Schreibt die aktuelle Seitengrösse • `MaxSize`: Liest / Schreibt die aktuelle, maximale Anzahl an angezeigten Paginations-Buttons

Tabelle 28: Elemente des Datadisplayservices

5.6.1.3. Direktiven

Direktiven sind wiederverwendbare Codefragmete, welche es erlauben, keine Codeduplikate vorzufinden. Im Falle der Webapplikation wurden diese Direktiven verwendet, um mehrfach genutzte Formulare und Validationsprüfungen zu realisieren. Im Folgenden sind die Direktiven, welche eingesetzt wurden, beschrieben, mit Ausnahme der später erwähnten externen Direktiven und der „Dynamisches HTML Formular“ Direktive, welche in einem eigenen Abschnitt erläutert wird. Grundsätzlich sind alle Direktiven in zwei Typen zu unterteilen: „UI-Direktive“ und „Validations-Direktive“. Eine UI-Direktive ersetzt sich selbst mit einem HTML-Template und ermöglicht so die Wiederverwendung von Formularen oder Ähnlichem und eine Validations-Direktive validiert Inputfelder. Alle Direktiven, mit Ausnahme der ganzen Pakete, sind in der Datei „./Scripts/Shared/srnmgmtDirectives.js“ zu finden.

Externe Direktiven

Die einzelnen Gesamtpakete an Direktiven (wie z.B. „angular-strap“) sind in eigene Dateien ausgelagert. Jedoch mussten gewisse externe Direktiven noch angepasst werden, um mit unseren Anforderungen zu passen. So wurde beispielsweise bei der Pagination-Direktive von „ui.bootstrap“ direkt eine Abhängigkeit des Services eingefügt, da ohne diesen Service gar keine Darstellung möglich wäre. Diese externen Direktiven werden nur kurz erwähnt und die eigenen Änderungen erläutert. Diese Direktiven sind aus dem UI.Bootstrap Paket. [7]

Die externen Direktiven sind:

- `ui.bootstrap.buttons`: Checkbox und Radiobuttons. Änderungen:
  - Eigene Werte für Radiobuttons eingefügt
  - Für `true` / `false` können verschiedene Texte wie „Ja“ / „Nein“ definiert werden
- `ui.bootstrap.typeahead`: Typeahead für Inputfelder (Eingabehilfe). Änderungen:
  - Templatepfad angepasst
- `ui.bootstrap.pagination`: Paginationdirektive für die Benutzerlinks. Änderungen:
  - Templatepfad angepasst
  - Lokalisierung über window-Variablen eingebaut
- `ui.bootstrap.dialog`: Direktive zum Anzeigen von Benutzermeldungen. Änderungen:
  - Templatepfad angepasst
- `ui.bootstrap.transition`: Direktive für die Animation von Aktionen

ListPropertySelector

Abbildung 31: UI-Direktive ListPropertySelector

Diese UI-Direktive ermöglicht es, ein Set von Objekten einem anderen Set hinzuzufügen oder zu entfernen. Die nummerierten Elemente entsprechen:

1. dem Attribut „left-list-header“, einem Textattribut, welches einen dynamischen Titel ermöglicht
2. dem Attribut „right-list-header“, einem Textattribut, welches einen dynamischen Titel ermöglicht
3. dem Filter für die linke Liste
4. dem Filter für die rechte Liste
5. der linken Liste
6. der rechten Liste
7. den Buttons, welche die markierten Elemente hin- und herschieben

Die Direktive fügt einem Container die Elemente in der rechten Spalte hinzu. Zu diesem Zweck müssen zwei Quellen angegeben werden: `left-source` und `right-source`. Diese werden dann durch die Direktive bearbeitet. Wird ein neues Element hinzugefügt, so wird der Container der rechten Liste bearbeitet.

AddExistingProductForm

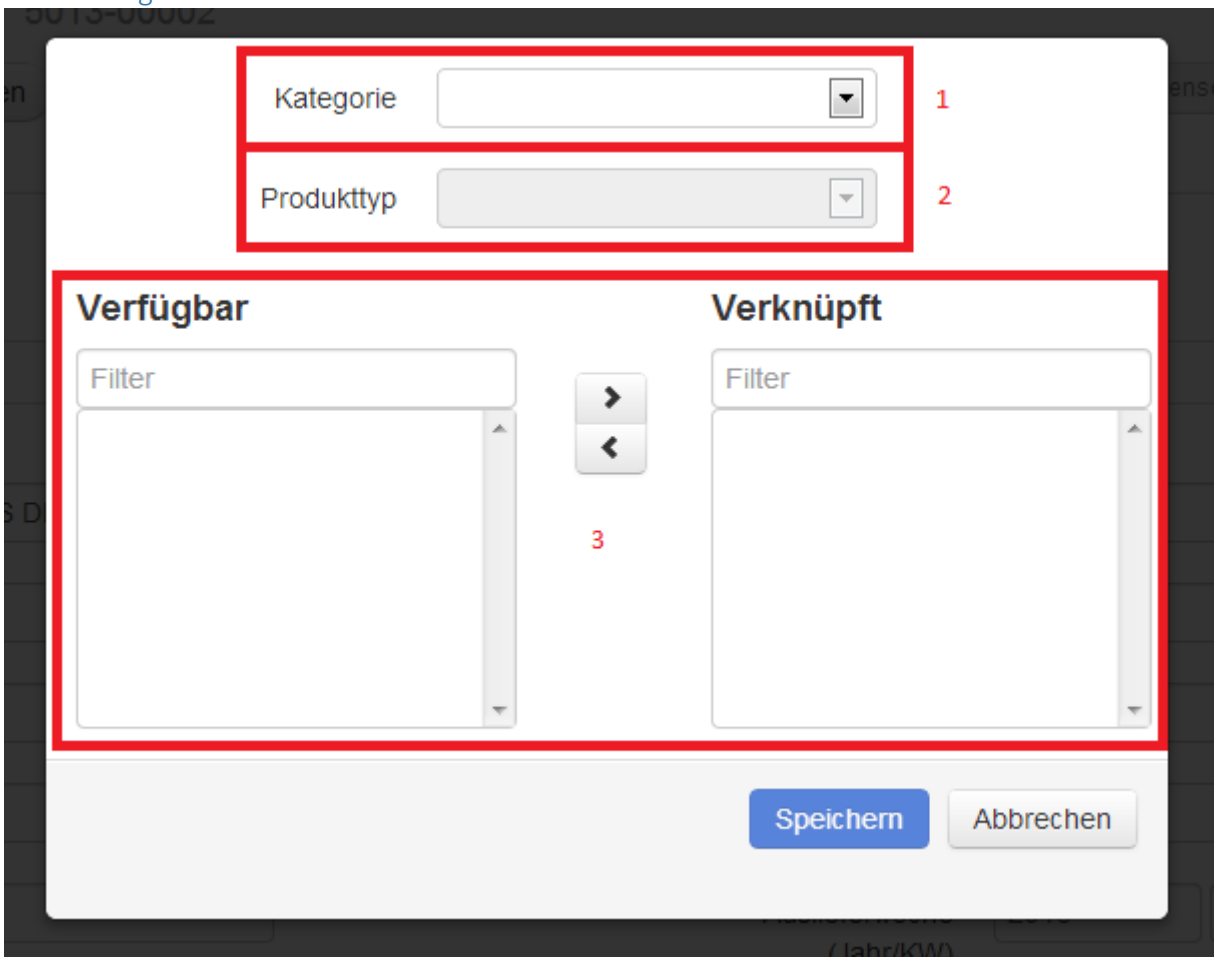


Abbildung 32: UI-Direktive AddExistingProductForm

Diese UI-Direktive wird dazu verwendet, bestehende Produkte zu einem Auftrag oder einem Überprodukt hinzuzufügen. Der Benutzer wählt eine Produktkategorie und einen Produkttyp. Danach werden die verfügbaren Produkte des Typs angezeigt und er kann diese mittels der Buttons mit dem Auftrag bzw. dem Elternprodukt verknüpfen. Die nummerierten Elemente sind:

1. die Kategorie-Auswahl für den Benutzer
2. die Typ-Auswahl für den Benutzer
3. ListPropertySelector Direktive für das Ändern der Container

Die Direktive enthält eine kleine Sub-Direktive (`addExistingProductLiButton`), welche einen lokalisierten, speziellen Button zur Verfügung stellt, um den Dialog zu öffnen.

AddUsedProductForm

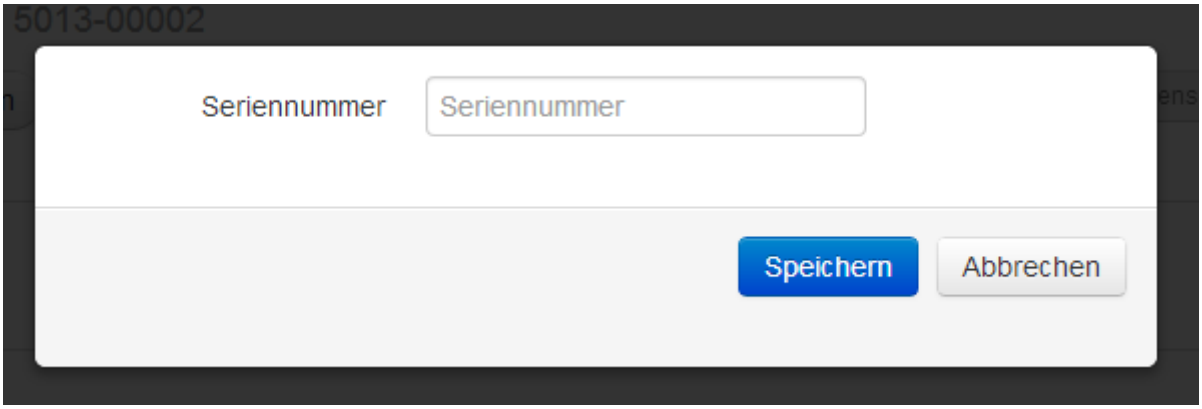


Abbildung 33: UI-Direktive AddUsedProductForm

Um einem Auftrag ein Occasionsprodukt hinzuzufügen, wurde eine UI-Direktive geschrieben, welche dies ermöglicht. Sie besteht im Wesentlichen nur aus einem Eingabefeld für eine Seriennummer. Wurde diese bereits verwendet, so wird ein Eintrag in der Historie gemacht, ansonsten wird das Produkt wie ein normales „existing product“ behandelt.

Numberonly

Abbildung 34: Validation-Direktive Numberonly

Die Direktive `numberonly` ist eine Validations-Direktive, welche dafür sorgt, dass in einem Inputfeld nur nummerische Werte eingegeben werden können. Es wird ein Fehler auf dem gesamten Inputfeld generiert, wenn der eingegebene Input nicht nummerischen Werten entspricht.

UniqueSerialnumber

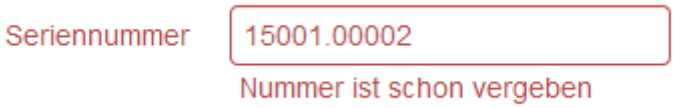


Abbildung 35: Validation-Direktive UniqueSerialnumber

Mittels der Validations-Direktive `unique-srn` wird überprüft, ob eine Seriennummer bereits in der Datenbank eingetragen ist. Zu diesem Zweck wird, sobald der Benutzer nach einer Eingabe 0,8 Sekunden Pause mit Eingaben macht, eine Abfrage an die API gesendet. Die URL für diese Abfrage ist `./api/v1/isuniquesrn?serialnumber=<<number>>`. Der API-Controller überprüft nun, ob die Seriennummer bereits eingetragen ist und retourniert ein Objekt mit einem booleschen Wert, anhand dessen der Fehlerstatus gesetzt wird oder nicht.

Die Anfrage wird nicht ausgeführt wenn:

- Kein „string“ im Inputfeld ist
- Das Feld keinen Wert hat
- Die Direktive nicht ausgeführt werden darf (Konjunktion ist negativ)
- Der Wert im Inputfeld kleiner als 4 Zeichen ist (um Überflutungen zu verhindern)
- Der Benutzer innerhalb von 0,8 Sekunden eine weitere Taste drückt

5.6.2. Dynamisches HTML Formular

Ein besonderes Herzstück des Frontend ist das dynamische Formular, welches anhand der Attribute eines Typs oder Reports generiert wird. Dieses Formular ist in einer UI-Direktive und einem entsprechenden spezialisierten Template. Im Folgenden werden die Funktionen und die Rahmenbedingungen genauer erläutert.

Die Direktive `dynamicAttributeForm` ist ein Formular, welches sich mit dem AngularJS HTML-Attribut `ng-repeat` dynamisch aufbauen kann. Der Backendcode ist vergleichsweise trivial, da er nur die Formularfelder auf Fehler in der Eingabe (nummerische Werte, erzwungene Felder) überprüft und die Variable mit den Attributen austauscht. Wichtig für die Funktion sind das Template und die Datenstruktur, welche eingehalten werden muss.

Produkt-, und Reportattribute können fünf Typen haben:

- `string`: ein normales, alphanummerisches Feld
- `date`: ein Datumsfeld
- `number`: ein nummerischer Wert
- `boolean`: ein boolescher Wert (true / false)
- `selection`: eine Auswahl verschiedener alphanummerischer Werte

Zusätzlich kann ein Attribut erzwungen sein oder nicht. Dies geschieht über ein Flag beim Erstellen bzw. Modifizieren des Attributes.

Wird bei der Erstellung des Attributes der Typ `selection` ausgewählt, so wird dem Administrator ein zusätzliches Feld angezeigt, in welchem er die Auswahlmöglichkeiten Komma-getrennt eingeben muss. Diese Komma-getrennten Auswahlen werden im Hintergrund in einen Array abgefüllt und so an den Server übermittelt.

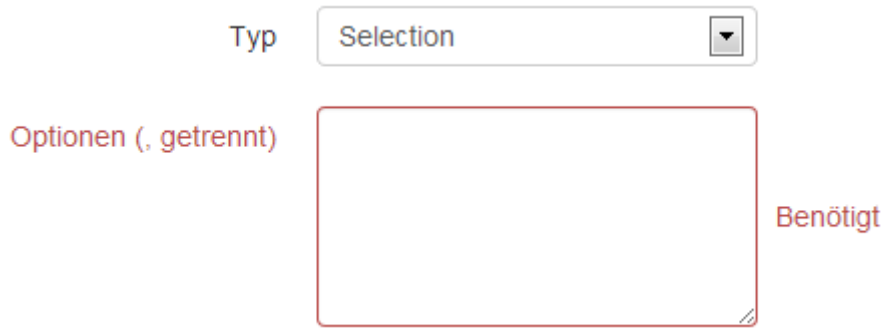


Abbildung 36: Angabe der Auswahlmöglichkeiten

Damit das dynamische Formular funktioniert, muss folgende Datenstruktur vorhanden sein:

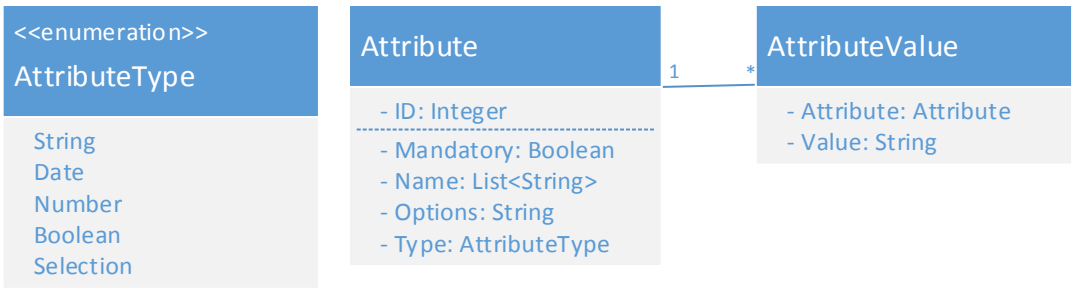


Abbildung 37: Datenstruktur dynamisches HTML Formular

Im `attributeContainer` der Direktive wird eine Liste von `AttributeValue` Objekten übergeben. Diese werden überprüft und in eine interne Liste übertragen. Danach baut sich das Formular anhand der Informationen im `Attribute` Objekt auf. Je nach Attributtyp wird ein anderes Inputfeld angezeigt, was mittels AngularJS `ng-switch` realisiert wird. Werden die Werte verändert, so werden diese direkt in Container gespeichert, welche per Referenz mit dem zu bearbeitenden Objekt verknüpft sind. Somit wird das Produkt beziehungsweise der Report mit den Attributwerten befüllt.

Abbildung 38: Ansicht dynamisches HTML Formular

Der Attributtyp `selection` besitzt zusätzlich eine spezielle Eigenschaft. Da die Datenbankanbindung keine variablen Datentypen zulässt, muss ein einheitliches Format erreicht werden. In der Datenbank werden diese Werte als „String“ abgespeichert und bei der Speicherung bzw. beim Laden des Wertes in den entsprechenden Typ konvertiert. Da nun aber die Selektion mehrere Auswahlen enthalten kann, werden die angegebenen Auswahloptionen im JSON Format abgelegt. Diese Optionen werden ausgelesen und dem Benutzer angezeigt.

Ist ein Wert nicht mehr in der Auswahl vorhanden, jedoch im Wert-Feld eingetragen, so wird er automatisch zur Auswahl hinzugefügt. Der Sinn dabei ist, dass bei Änderungen an den Optionen trotzdem die alten Werte behalten werden können, da ansonsten das Formular einen Fehler meldet. Es kann zum Beispiel in einem ersten Schritt aus den Auswahlmöglichkeiten A, B, C die C gewählt und gespeichert werden. Ist C später nicht mehr Teil der Auswahl (A, B), so wird ein Objekt, welches im Wert-Feld „C“ besitzt, trotzdem die Auswahl C erhalten. Wird jedoch ein anderer Wert (A, B) verwendet, verschwindet C aus der Auswahl.

Das dynamische Formular wird aktuell für die Erstellung und das Bearbeiten von Produkten sowie Reporten genutzt. Da es eine AngularJS Direktive ist, kann das Formular jedoch auch an anderen Stellen genutzt werden, solange die Datenstruktur eingehalten wird.

5.6.3. Ressourcenservices

Eine bestimmte Gruppe von Services wurde nicht bei den obigen AngularJS Services aufgelistet, da es sich um spezielle Services handelt. Diese Ressourcenservices dienen der Kapselung der Ressourcen. Zu Beginn der Implementation wurden die Ressourcen (API-Schnittstellen für AngularJS) direkt als Abhängigkeiten genutzt. Das Problem dabei ist, dass die Anzahl Ressourcen schnell anstieg und damit auch in jedem Kontroller mit dynamischen Daten die Anzahl Abhängigkeiten anstieg. Um zu verhindern, dass bei kleineren Änderungen oder bei neuen Ressourcen sämtliche Kontroller angepasst werden müssen, wurden diese Services eingefügt. Wenn eine neue Ressource hinzugefügt wird, so ist die Abhängigkeit an einer zentralen Stelle und die Funktionalität wird danach vom Ressourcenservice zur Verfügung gestellt.

Alle Ressourcen sind in der Datei „./Scripts/Shared/srnmgmtResources.js“ zu finden. Eine Ressource selbst sieht folgendermassen aus:

```
angular.module('srnmgmt.resource.job', ['ngResource']).
    factory('job', ['$resource', function ($resource) {
        var job = $resource(window.baseLocation + window.apiLocation + 'jobs/:jobNumber',
                  { jobNumber: '@Jobnumber' }, {
            search: { method: 'GET', isArray: true, params: { jobNumber: '' } },
            update: { method: 'PUT' }
        });
        job.prototype.getReadableEabNotes = function () {
            if (!hasFieldValue(this.EabNotes) || !angular.isString(this.EabNotes))
                return '';
            return (this.EabNotes.length > 60) ?
                    this.EabNotes.substring(0, 60) + '...' : this.EabNotes;
        };
        return job;
    }]);
```

Die Ressource besteht aus einer URL und REST-Operationen. Sind von der API an der Stelle der angegebenen URL die verschiedenen REST-Operationen zur Verfügung gestellt, erledigt die Ressource das gesamte Handling selbst. Dem Ressourcenobjekt können nun auch noch spezialisierte Funktionen hinzugefügt werden (`job.prototype.getReadableEabNotes = function(){/*…*/}`), um die Ressource zu erweitern. Eine Ressource stellt danach gewisse Funktionalitäten bereit.

Damit dies funktioniert, müssen gewisse Parameter festgelegt werden. Im obigen Beispiel wird die URL auf „./api/v1/jobs/:jobNumber“ festgelegt. Für die Ressource bedeutet dies, dass die Basis der Ressource „…/jobs“ ist. Nun wird, ausgehend von der aufgerufenen Methode, eine ID angehängt oder nicht. Diese ID wird ebenfalls in der URL festgelegt und ist in diesem Beispiel `:jobNumber`.

Die Operatoren sind (ausgegangen von einer URL „./myobject/:ID“):

JS-Funktion	HTML Abfrage	URL	Bedeutung
.query()	GET	./myobject	„Query“ liefert alle Elemente zurück. Es wird ein GET auf die URL ohne ID ausgeführt. Gewisse Schnittstellen unterstützen diese Option nicht, da zu viele Ergebnisse geliefert würden.
.get(ID)	GET	./myobject/ID	„Get“ liefert genau ein Objekt zurück. Die Anfrage wird an die Ressource mit dem eindeutigen Schlüssel gesendet. Als Resultat erhält man das Objekt mit dem angegebenen Schlüssel.
.save(obj)	POST	./myobject	„Save“ speichert ein Objekt **ohne** ID. Im Normalfall wird davon ausgegangen, dass die Datenbank den eindeutigen Schlüssel generiert. Somit wird ein Objekt ohne ID (in den POST Daten) an den Server gesendet. Der Server speichert das Objekt in der Datenbank und liefert das ganze Objekt mit eingetragener ID zurück.
.update(ID, obj)	PUT	./myobject/ID	„Update“ kann über POST oder PUT realisiert werden. In unserem Design wurde PUT verwendet, um die beiden Operationen klar zu trennen. Dem Server wird eine ID und ein Objekt (in den POST Daten) geliefert. Mit den Daten wird das identifizierte Objekt aktualisiert und das aktualisierte Objekt zurückgesendet.
.delete(ID)	DELETE	./myobject/ID	„Delete“ löscht ein Objekt in der Datenbank. Dem Server wird die ID überliefert, welche er zu Löschen hat. Da aus Integritätsgründen nicht alle Objekte gleich gelöscht werden dürfen, wird erst geprüft, ob Referenzen auf das Objekt vorhanden sind. Kann der Server den Löschvorgang ausführen, wird `deactivated: false` geliefert. Kann der Server das Objekt nicht löschen, so wird es deaktiviert („Active“-Flag auf „false“ setzen) und `deactivated: true` wird zurückgeliefert.

Tabelle 29: REST-Operationen auf Ressourcen

Damit, wie oben erwähnt, nicht zu viele Abhängigkeiten entstehen und die Anzahl der Ressourcen sich ständig erhöht hat, wurden die Ressourcenservices „productService“, „jobService“ und „reportService“ als Kapselung dazwischen gestellt. So hat nun beispielsweise die Produktsuche nur die Abhängigkeit auf den Produkteservice, anstatt fünf bis sechs verschiedene Anhängigkeiten auf die einzelnen Ressourcen zu haben.

Diese Services enthalten alle relevanten Operationen auf den Ressourcen und sorgen für eine einheitliche Fehlerbehandlung. Diese Operationen reichen vom Erstellen über Modifikation bis hin zum Löschen und anderen Operationen. Auch Suchoperationen und das Erstellen von Verknüpfungen (Produkt einem Auftrag anhängen) werden über diese Services realisiert.

Da diese Operationen asynchron verlaufen (AJAX), wird mit dem gängigen „Promise“-Modell [8] gearbeitet. Eine Operation liefert ein Versprechen zurück, welches sich meldet, sobald es erfüllt ist. Durch dieses Modell wird der Fluss der Applikation nicht gestoppt.

5.6.4. Bundling und Minification

5.6.4.1. Grundlagen

Bundling (das Bündeln von Anfragen) und Minification (das Entfernen von überflüssigen Zeichen und Kommentaren im Code) ist ein wichtiger Bestandteil der Applikation. Diese Techniken sorgen dafür, dass der Browser des Benutzers nicht zu viele Anfragen an den Server stellen muss. Diese Funktionalität wird von der „Optimization“ Referenz einer ASP.NET Applikation zur Verfügung gestellt.

In der Webapplikation werden diese Techniken für sämtliche JavaScript und CSS Dateien verwendet. So werden dem Programmierer die vollständigen Dateien geliefert, jedoch dem Benutzer auf der Produktionsumgebung nur die gebündelten und komprimierten Dateien.

Nachfolgend ist ein Beispiel einer nicht-komprimierten Funktion in JavaScript dargestellt:

```
function sum(variableOne, variableTwo, variableThree){
       //comment
       var subTotal = variableOne + variableTwo;
       return subTotal + variableThree;
}
```

Diese Funktion hat lange Parameternamen sowie eine Zwischenvariable im Code. Komprimiert sieht die Funktion folgendermassen aus:

```
function sum(e,t,n){var r=e+t;return r+n}
```

Die Idee dahinter ist, die überflüssigen Leerzeichen und Kommentare zu eliminieren. Dies sorgt für kleinere Antworten des Servers und schnelleres Parsing des Browsers.

Das Bündeln von Abfragen sorgt ebenfalls dafür, dass die Seite im Browser schneller geladen wird. Ein durchschnittlicher Browser kann ca. 5-10 Abfragen gleichzeitig ausführen. Da jedoch die Webapplikationen immer komplexer und dynamischer werden, ist diese Zahl meist zu tief. In unserem Projekt muss der Browser für die Auftragsübersicht ca. 25 Abfragen erledigen. Da dies zu Performanceeinbussen führt, werden diese Abfragen gebündelt. Dies bedeutet, dass mehrere JavaScript oder CSS Dateien in eine Abfrage gepackt werden. So werden aus beispielsweise 10 JavaScript Dateien (bzw. Abfragen) lediglich eine Abfrage.

Konfiguriert werden diese Bündel in der Klasse `BundleConfig` der Webapplikation. Ein solches Bundle kann folgendermassen definiert sein:

```
bundles.Add(new ScriptBundle("~/scripts/angular")
    .Include("~/Scripts/jquery/jquery-{version}.js")
    .Include("~/Scripts/angularJS/angular.js")
    .Include("~/Scripts/angularJS/angular-resource.js")
    .Include("~/Scripts/angularJS/angular-cookies.js"));
```

In obigem Beispiel werden vier JavaScript Dateien in eine Abfrage („./scripts/angular“) gepackt. Diese Optimierungsmethoden sind für die Produktivumgebung sehr nützlich, jedoch für den Programmierer eher mühsam zu debuggen. Deshalb wird die Optimierung nur bei den produktiven Builds eingeschaltet.

5.6.4.2. AngularJS und Minification

Da AngularJS über DI funktioniert, kann die Kompression des JavaScript Codes zu Problemen führen. Wie im Abschnitt AngularJS anhand eines Beispiels erklärt wird, kann ein AngularJS Controller in mehreren Notationen geschrieben werden. Wird die Notation verwendet, in der der Controller über eine globale Funktion definiert wird, so hat AngularJS nach einer Komprimierung keine Ahnung mehr, welche Variablen injiziert werden müssen. Diese Notation und ihre Komprimierung sehen wie folgt aus:

```
function TestCtrl($scope, $http, $timeout){
	$scope.yourName = 'Max';
	$scope.greetMe = function(){
		alert('Hello ' + $scope.yourName);
	}
}
```

Komprimiert kann der `$scope` und die `$http` bzw. `$timeout` Variable nicht mehr injiziert werden, da AngularJS anhand des Namens die Variablen identifiziert:

```
function TestCtrl(e,t,n){e.yourName="Max";e.greetMe=function(){alert("Hello "+e.yourName)}}
```

Um diesem Problem entgegen zu wirken, ist es in AngularJS möglich, die Variablen anzugeben, welche injiziert werden sollen. Dies geschieht über eine spezielle Konstruktor-Notation:

```
angular.module('testApp', []).
	controller('TestCtrl', ['$scope', '$http', '$timeout', function($scope, $http,
	                                                                $timeout){
		$scope.yourName = 'Max';
		$scope.greetMe = function(){
			alert('Hello ' + $scope.yourName);
		}
	}]);
```

Diese Notation enthält die zu injizierenden Variablen in String-Form. Diese Form darf nicht komprimiert werden, da sie eventuell wichtige Daten enthält. Das komprimierte und funktionierende Äquivalent zum obigen Code wäre:

```
angular.module("testApp",[]).controller("TestCtrl",["$scope","$http","$timeout",
function(e,t,n){e.yourName="Max";e.greetMe=function(){alert("Hello "+e.yourName)}}])
```

5.7. Reporting

Es wurde darauf geachtet, dass der Reporting Teil generisch ist. Aus diesem Grund haben die Administratoren die Möglichkeit, neue Report Files im Crystal Report Format „.rpt“ hochzuladen. Die Metadaten zu jedem Report werden beim erstmaligen Hochladen des Files ausgelesen und auf der Datenbank abgelegt. Dadurch müssen diese nicht bei jedem Generieren eines Berichts neu ausgelesen werden.

Das Designmodell für das Reporting sieht folgendermassen aus:

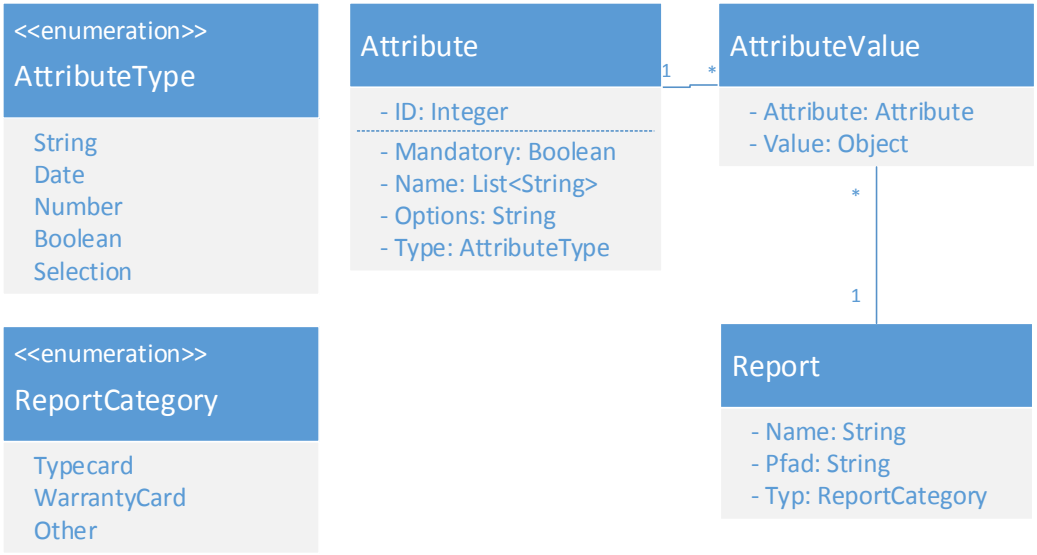


Abbildung 39: Designmodell Reporting

Die Klassen sind in folgender Tabelle beschrieben:

Klasse / Enum	Beschreibung
AttributeType	Enumeration mit den möglichen Parametertypen.
Attribute	Beschreibt einen Parameter eines Reports. Dies umfasst in erster Linie den Namen des Parameters sowie den Typ.
AttributeValue	Enthält den Wert eines Parameters und den Parameter selbst.
Report	Beschreibt einen eigentlichen Report. Von diesem Objekt sind sämtliche Metadaten eines Report Files erreichbar. Es hält den Namen und die Kategorie des Reports und den Pfad zum Report File.
ReportCategory	Enumeration mit den möglichen Report Kategorien.

Tabelle 30: Klassen für das Reporting

5.7.1. Das RPT File

Mithilfe von Designern kann das RPT File erstellt werden. In folgender Abbildung ist das Bearbeiten eines Files mit einem Designer sichtbar:

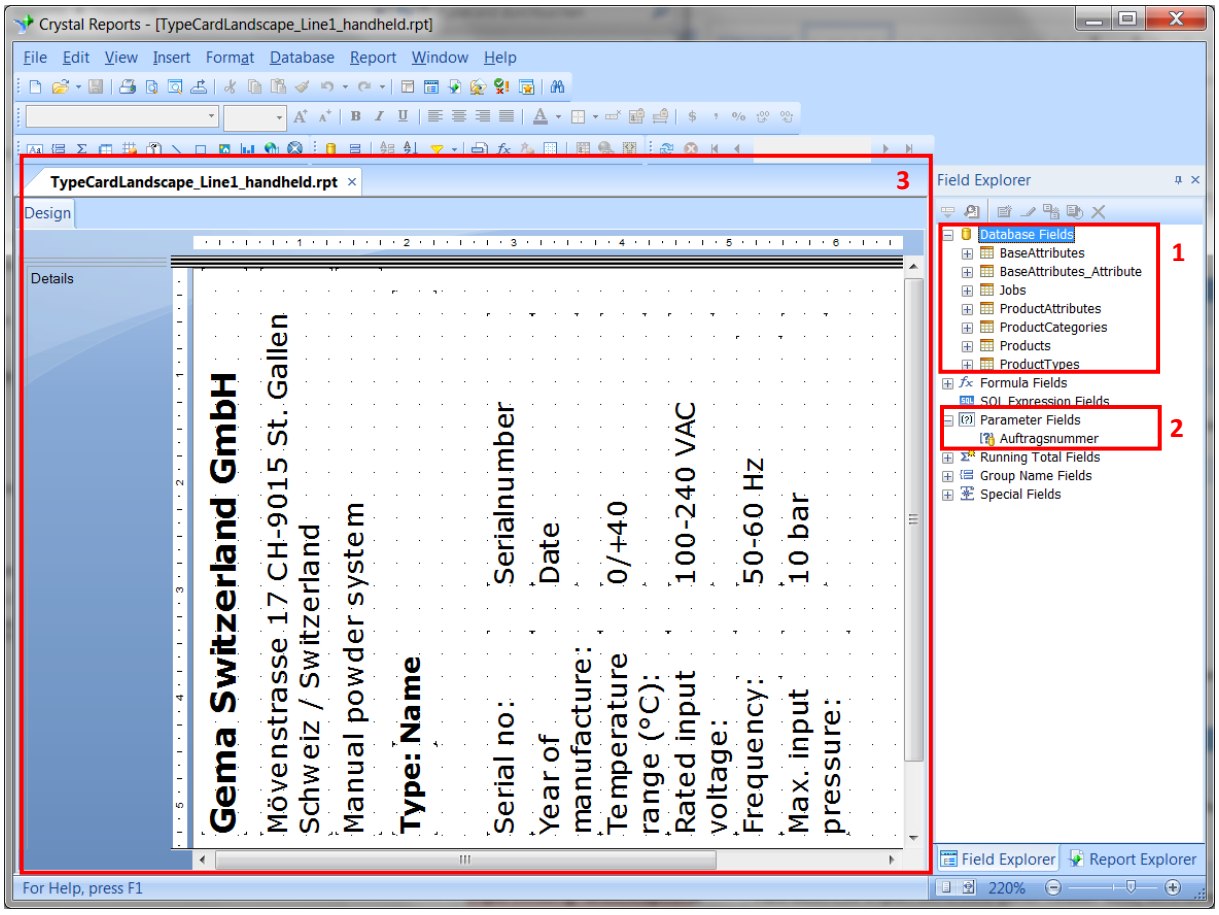


Abbildung 40: Designen eines RPT Files

1. *Datenbank*: Im File werden die Verbindung und die Abfrage zur Datenbank gespeichert. Die Verbindung zur Datenbank kann durch wenige Klicks angepasst werden. Dies ist hilfreich, wenn zum Beispiel die Entwicklungs- und die Produktivumgebung andere Verbindungen voraussetzen.
2. *Parameter*: Sämtliche Parameter, die für die Generierung des Reports benötigt werden, können hier eingetragen werden. Diese Parameter werden oftmals für die Datenbankabfrage eingesetzt.
3. *Design*: Hier ist das eigentliche Designfenster. Durch Drag and Drop ist das Zeichnen von Reports entsprechend einfach. Auf den Umgang mit dem Designer soll hier jedoch nicht weiter eingegangen werden.

5.7.2. Barcodes generieren

Um in den Reports Datamatrix Codes generieren zu können, musste eine Erweiterung zu Crystal Reports besorgt werden. Nach Vergleichen von mehreren Anbietern wurde entschieden, „DataMatrix Font & Encoder 5“ von Morovia[7] einzusetzen. Mit Hilfe einer DLL, die auf dem Server installiert wird, werden in den Formelfeldern spezielle Funktionen zur Verfügung gestellt. Das Formelfeld codiert den Text, welcher im Datamatrix Code stehen soll. Mit der richtigen Font wird die Ausgabe des Formelfelds als Datamatrix angezeigt. Die Font muss bei jedem Client verteilt werden.

Folgende Formel erzeugt einen Datamatrix Code einer Seriennummer:

```
StringVar CompleteBarcodeString:="";
StringVar DataToEncode:= "21" + {Products.Serialnumber};
NumberVar i:=0;
NumberVar Segments:=
DataMatrixEncodeSet(DataToEncode, 4);
For i:=0 to Segments Do
(
       CompleteBarcodeString := CompleteBarcodeString + DataMatrixEncodeGet(i);
);
CompleteBarcodeString
```

Eine genaue Anleitung kann beim Anbieter selbst gefunden werden **Es ist eine ungültige Quelle angegeben.**.

5.7.3. Hochladen eines RPT Files

Der Administrator kann nun via Web GUI solche Report Files hochladen. Das File wird auf dem Server im „App_Data“ Ordner abgelegt. Anschliessend werden die Parameter des Reports ausgelesen und in der Datenbank abgelegt. Folgender Code wird benötigt, um über die Parameter eines Reports zu iterieren:

```
var crystalReportFile = new ReportClass{FileName = Folder + report.Path};
foreach (ParameterFieldDefinition param in
    crystalReportFile.DataDefinition.ParameterFields)
{
       … // Convert param to Attribute/AttributeType/AttributeValue
}
```

Wie bereits einmal angetönt, wurde das Konzept mit Attribute, AttributeType und AttributeValue für die Reportparameter verwendet. Die ausgelesenen Parameter werden nun in dieses Konzept umgewandelt. Dies hat den Vorteil, dass man Methoden auslagern und für beide Teile benutzen kann.

Zu beachten ist, dass nur Parameter ausgelesen werden, die nicht mit „SRPm-„ beginnen. Somit ist es möglich, auch Parameter in einem Report zu definieren, die nicht durch die Benutzer ausgefüllt werden müssen. Dies ist vor allem im Zusammenhang mit Subreports wichtig. Subreports haben oftmals Parameter, die jedoch vom Mainreport gesetzt werden und nicht noch durch den Benutzer ausgefüllt werden müssen.

Ebenso wird der Parametername als deutscher Attributname übernommen. Die englische Übersetzung ist nach dem Upload durch den Administrator vorzunehmen. Alle ausgelesenen Parameter werden als Pflichtattribut gesetzt, das heisst der Benutzer muss sie in jedem Fall ausfüllen.

[7] http://www.morovia.com/fonts/datamatrix/

Der Ablauf ist in folgender Grafik ersichtlich:

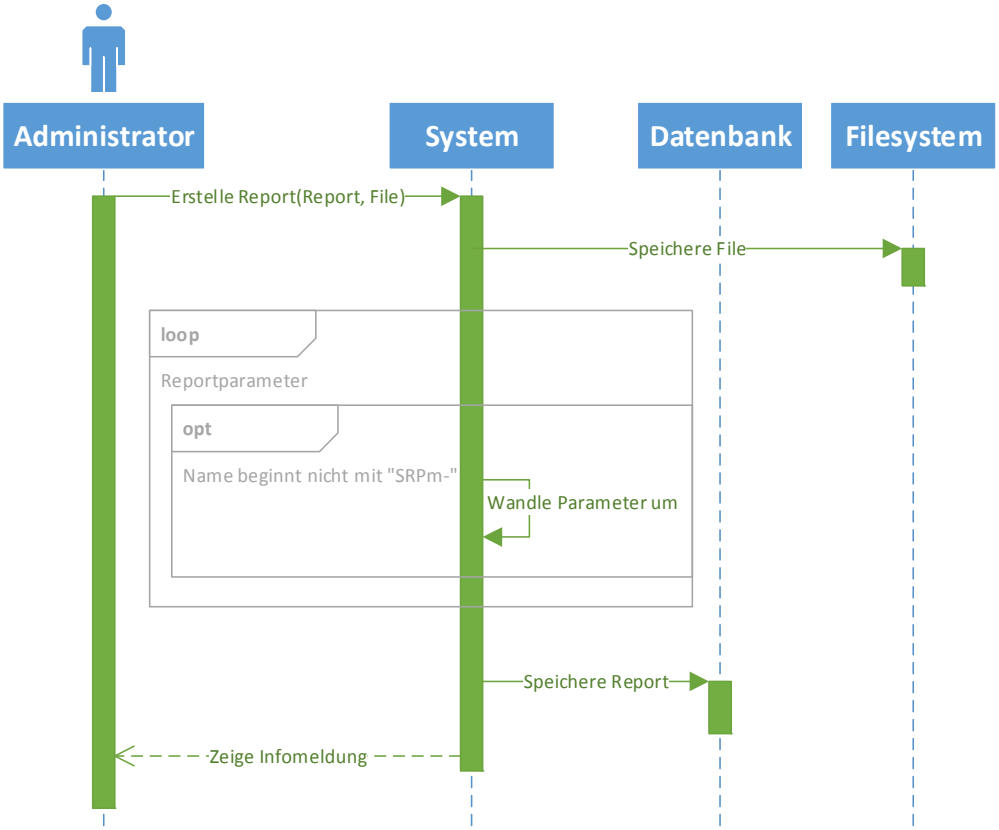


Abbildung 41: Ablauf für Upload eines Reports

5.7.4. Generierung eines Reports

Folgendes Sequenzdiagramm gibt einen Überblick über den Ablauf bei der Reportgenerierung:

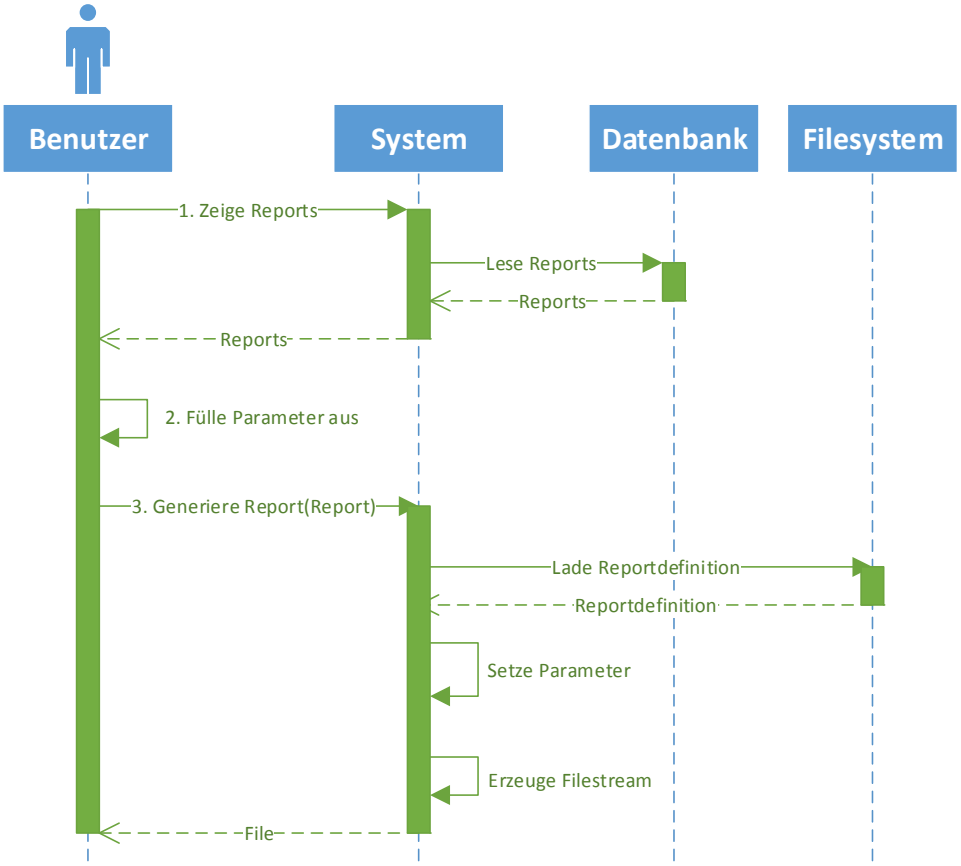


Abbildung 42: Ablauf zur Generierung eines Reports

Schritt 1: Der Benutzer hat an mehreren Stellen der Applikation die Möglichkeit, Reports zu generieren. Welche Reports dabei verfügbar sind, hängt allein von den Einträgen von Report Objekten in der Datenbank ab. Die Files im App_Data Ordner werden dazu nicht beachtet. Dies wurde aus Performancegründen so entschieden. Es wäre auch möglich gewesen, die Metadaten bei jeder Abfrage der verfügbaren Reports neu auszulesen. Da die Reportdefinitionen jedoch eher statisch sind, ist dieser Overhead nicht zu verantworten.

Schritt 2: Mithilfe des generischen HTML-Formulars (5.6.2 Dynamisches HTML Formular) muss der Benutzer nun sämtliche Parameter/Attribute ausfüllen. Zusätzlich zu den Parametern, die der Report schon hat, wird noch ein Parameter für das Dateiformat hinzugefügt. Dies ermöglicht dem Benutzer, den Report als PDF oder als Word zu exportieren.

Schritt 3: Erst wenn der Benutzer den Bericht effektiv generieren will, wird wieder auf das RTP File im App_Data Ordner zugegriffen. Zuerst wird es geladen und anschliessend mit den neusten Daten aktualisiert:

```
var crystalReportFile = new ReportClass
       {
              FileName = Folder + reportToGenerate.Path
       };
crystalReportFile.Load();
crystalReportFile.Refresh();
```

Nun müssen noch die Parameter des Reports gesetzt werden. Dazu wird durch die AttributeValues iteriert, die vom Benutzer kommen. Um den Parameter zu setzen, ist folgender Befehl nötig:

```
crystalReportFile.SetParameterValue(param.Attribute.Name["de"], param.Value);
```

Da der deutsche Attributname als Schlüssel für das Setzen eines Parameterwerts benützt wird, ist es auch nicht möglich, diesen anzupassen.

5.7.5. Möglichkeiten zum Anpassen der Reports

Es können beliebig Reports hochgeladen und auch wieder gelöscht werden. Um einen hochgeladenen Report zu ändern gibt es mehrere Möglichkeiten, wobei einige Punkte beachtet werden müssen:

1. Das RPT File kann direkt auf dem Server bearbeitet werden. Solange nur kleine Designänderungen vorgenommen werden (z.B. Schriftgrösse anpassen, Text ändern etc), funktioniert diese Variante sehr gut. Falls jedoch neue Parameter hinzukommen oder bestehende Parameter umbenannt werden, wird dies zu Fehlern in der Anwendung führen, da die Metadaten nicht mehr übereinstimmen.
2. Das RPT File kann auf dem Server direkt ausgetauscht werden. Hier gibt es dieselbe Einschränkung wie beim ersten Punkt. Solange die Parameter unverändert bleiben, treten keine Probleme auf.
3. Der bestehende Report wird via Web GUI gelöscht und danach neu hinzugefügt. Dadurch werden auch die Metadaten neu ausgelesen und die Probleme der ersten beiden Varianten entfallen.

5.8. Exceptions und Logging

5.8.1. Exceptions

Die selber definierten Exceptions befinden sich im Namespace „Srnmgmt.Common.Helpers“, damit sie von der WebApp wie auch vom BarcodeClient verwendet werden können.

Exception	Beschreibung
GenericSrnmgmtException	Diese Exception wird standardmässig geworfen, sofern eine Exception nicht speziell behandelt wird.
BadRequestException	Eine BadRequestException kann in der WebApp auftreten, falls eine Anfrage an den Server fehlgeschlagen (z.B.: ID nicht angegeben) ist.
EmptySearchRequestException	Tritt auf, wenn eine Suchanfrage leer ist.
ImportMissingFieldsException	Falls beim CSV Import ein erzwungener Header fehlt, wird diese Exception geworfen.
ObjectNotFoundException	Wird geworfen, wenn das gesuchte Element nicht in der Datenbank gefunden werden konnte.
AttributeNotActiveException	Sollte ein Attribut eines Produkts, das jedoch auf inaktiv gesetzt ist, einen Wert enthalten, so wird diese Exception geworfen.
ProductTypeNotActiveException	Wenn ein Produkt vom Typ eines inaktiven Produkttyps ist, wird diese Exception geworfen.
SerialnumberNotUniqueException	Die Exception tritt beim Erstellen eines Produkts auf, dessen Seriennummer bereits vergeben ist.
SerialnumberFormatException	Diese Exception wird dann geworfen, wenn die Seriennummer des Produkts nicht mit dem Regex dieses Produkttyps übereinstimmt.
MissingAttributeException	Sollte ein erforderliches Attribut eines Produkts nicht gesetzt sein, wird diese Exception geworfen.
NonexistentAttributesException	Wenn ein Produkt ein Attribut enthält, das eigentlich für diesen Produkttyp nicht definiert ist, wird diese Exception geworfen.
EmptyAttributeException	Tritt auf, wenn ein Attribut nicht ausgefüllt wurde, sofern das Ausfüllen obligatorisch ist.
DateNotSetException	Bei falschem / nicht gesetztem Datum erscheint diese Exception.
WrongTypeException	Diese Exception tritt dann auf, wenn beim Speichern eines Produkts dessen Typ nicht mehr mit dem Typ des Produkts in der DB übereinstimmt.
RecursiveAttachmentException	Wird dann geworfen, wenn ein Produkt sich selber als Child zugewiesen wird.

Tabelle 31: Liste der benutzerdefinierten Ausnahmen

5.8.2. Logging

Um den Administratoren ein Werkzeug für die Fehlerbehebung an die Hand zu geben, wurde ein Loggingframework eingebunden. Dieses Framework loggt alle Exceptions, welche in der Webapplikation auftreten. Diese Exceptions werden dann als xml Datei in einem Verzeichnis („~/App_Data/Log“) abgelegt. Die Administratoren können über das Adminmenü unter dem Menüpunkt „Error-Log“ eine Übersicht aufrufen. Ausserdem können genauere Details des Fehlers betrachtet werden (Session, Requestinformationen, Pfade, etc.). Das Framework hängt sich mittels eines Requestfilters zwischen den IIS und den Client. Sobald ein Fehler auftaucht, wird der Filter notifiziert. Der Filter ruft dann die entsprechenden Informationen ab und speichert diese.

Die Konfiguration dieses Frameworks geschieht über die Web.config Datei. In ihr können die Pfade zum Errorlog sowie die Gruppen, welche darauf zugreifen dürfen, konfiguriert werden.

```
<appSettings>
    /*…*/
    <add key="elmah.mvc.disableHandler" value="false" />
    <add key="elmah.mvc.disableHandleErrorFilter" value="false" />
    <add key="elmah.mvc.requiresAuthentication" value="true" />
    <add key="elmah.mvc.allowedRoles" value="SRNMGMT-Administrator" />
    <add key="elmah.mvc.route" value="admin/elmah" />
    /*…*/
  </appSettings>
```

Error Log for srnmgmt on Srv

RSS FEED | RSS DIGEST | DOWNLOAD LOG | HELP | ABOUT

Errors 1 to 15 of total 226 (page 1 of 16). Start with 10, 15, 20, 25, 30, 50 or 100 errors per page.

Host	Code	Type	Error	User	Date	Time
LORENZPC	0	BadRequest	This was a bad request. Bad format, missing parameters or wrong address. Details...		11.06.2013	11:10
LORENZPC	0	BadRequest	This was a bad request. Bad format, missing parameters or wrong address. Details...		11.06.2013	11:10
LORENZPC	0	BadRequest	This was a bad request. Bad format, missing parameters or wrong address. Details...		11.06.2013	10:55
LORENZPC	0	ObjectNotFound	The requested object was not found in the database. Details...		07.06.2013	12:59
LORENZPC	400	Http	The length of the query string for this request exceeds the configured maxQueryStringLength value. Details...		07.06.2013	10:33
LORENZPC	0	COM	Zeichenfolge ist nicht numerisch. Details: errorKind Fehler in Datei "asfd {9890EB96-FD1C-4703-9226-FA5D6593E0AB}.rpt": Fehler in Formel serNr: 'ToNumber(Right ({?Seriennummer}, InStr ({?Seriennummer}, '.'))) + {Command.ser}' Zeichenfolge ist nicht numerisch. Details: errorKind. Details...		06.06.2013	15:09
LORENZPC	0	COM	Daten konnten nicht aus Datenbank abgerufen werden. Details: [Datenbankanbietercode: 3701] Daten konnten nicht aus Datenbank abgerufen werden. Fehler in der Datei Rdddkkk {9EBC1CCD-47BA-4485-BC6F-9FE712F805B7}.rpt: Daten konnten nicht aus Datenbank abgerufen werden. Details: [Datenbankanbietercode: 3701] Details...		06.06.2013	15:02
LORENZPC	0	COM	Daten konnten nicht aus Datenbank abgerufen werden. Details: [Datenbankanbietercode: 3701] Daten konnten nicht aus Datenbank abgerufen werden. Fehler in der Datei asdf {3FCE1212-1D9F-4DCB-B2D1-AB67A61A4817}.rpt: Daten konnten nicht aus Datenbank abgerufen werden. Details: [Datenbankanbietercode: 3701] Details...		06.06.2013	14:37
LORENZPC	0	ObjectNotFound	The requested object was not found in the database. Details...		06.06.2013	14:29

Abbildung 43: Ausschnitt aus dem Error-Log

5.9. Besonderheiten

5.9.1. Vorhersage der Seriennummern

Da die Seriennummer eine Identifikation besitzt, können gewisse Typen von Seriennummern vorhergesagt werden. Leider sind viele Seriennummern ein wirres Gemisch aus Zahlen und Buchstaben. Für diese Seriennummern kann keine akkurate Vorhersage gemacht werden. Für alle „gängigen“ Seriennummern (Formate wie: A0000, 15001.01250) können Annahmen getroffen werden. Diese Annahmen werden folgendermassen erstellt:

1. Neuestes Produkt mit dem angegebenen Typen suchen
    a. Falls keines gefunden wird, so wird die Identifikation möglichst genau geparst und dann zurückgegeben
2. Seriennummer des besagten Produktes auslesen und von rechts nach links parsen
3. Sobald ein nichtnummerisches Zeichen entdeckt wird (z.B. der Punkt), wird der ganze Zahlenbereich rechts des Zeichens abgekapselt
4. Den ausgewählten Bereich in eine Zahl konvertieren und inkrementieren
5. Vorderen Teil der Nummer mit dem inkrementierten Teil zusammenfügen
6. Falls die erstellte Seriennummer nicht eindeutig ist, so wird der Vorgang (bis zu 1000 Mal) wiederholt
7. Seriennummer zurückliefern

5.9.2. Validierung

Um auf allen Ebenen die Korrektheit der Daten zu garantieren, wird an verschiedensten Stellen validiert. Dies wurde so umgesetzt, da die Daten über die API (Browser / Webapplikation) oder über eine Desktopapplikation mittels Businesslayer bearbeitet werden können. Im Folgenden sind die Validierungen der verschiedenen Schichten beschrieben:

- **Webapplikation GUI**: Die Validierung geschieht über das UI. Der Benutzer kann kein Formular auf dem normalen Weg (Klick auf „Speichern“) absenden, ohne die Felder ausgefüllt zu haben. Dies ist eine einfache, jedoch effiziente Form der Validierung, kann aber nicht ohne Backendvalidierung funktionieren, da theoretisch Daten verloren gehen oder die Requests modifiziert werden könnten.
- **Webapplikation API**: Gewisse Validierung geschieht bereits im API / View Controller. Sendet ein Browser beispielsweise einen Request zur Änderung eines Objektes mit einer leeren ID, so wird der Request mit einem „BadRequest“ quittiert.
- **Businesslayer**: Die zentrale Validierungsschnittstelle. Hier werden die Produkte und anderen Objekte auf Herz und Nieren geprüft. Es wird sichergestellt, dass die ID eindeutig ist, die erforderlichen Attribute vorhanden sind, die Seriennummer eines Produktes zum Typen passt, etc.
- **Datenbank**: In der Datenbank wird lediglich mit „Constraints“ validiert. Sie enthalten gewisse Bedingungen über die Datenbankfelder wie die Gesamtlänge des Seriennummernfeldes.

Die wichtigste Validierung findet dabei im BL statt. Der BL ist die zentrale Schnittstelle zu den Daten und somit ist er die letzte Schnittstelle vor der Datenbank. Es ist also essentiell, dass die genaueste Validierung im BL erfolgt.

5.9.3. Nebenläufigkeit

Grundsätzlich wird die Nebenläufigkeit vom IIS bzw. der ASP.NET Engine geregelt. Die ASP.NET Engine erzeugt für jeden Request einen API / MVC Controller. Ein ganzer Zyklus eines Requests besteht aus **Es ist eine ungültige Quelle angegeben.**:

Schritt	Details
Ersten Request der Applikation empfangen	Routen aus dem „Global.asax“ File in der `RouteTable` registrieren.
Routing ausführen	Die `UrlRoutingModule` suchen in der `RouteTable` nach der passenden Route und leitet den Request dorthin.
MVC Request Handler erstellen	Der `MvcRouteHandler` instanziiert einen `MvcHandler` und übergibt diesem den Request-Kontext.
Kontroller erstellen	Der `MvcHandler` identifiziert den zuständigen Kontroller anhand des Kontexts und instanziiert dann den Kontroller.
Kontroller ausführen	Der `MvcHandler` führt die `Execute` Methode des Kontrollers aus.
Aktion ausführen	Der `ControllerActionInvoker`, welcher mit dem Kontroller assoziiert ist, identifiziert die auszuführende Methode auf dem Kontroller und führt diese mit den entsprechenden Attributen aus.
Resultat ausführen	Die Methode erhält den Benutzerinput, führt sich aus und erstellt das entsprechende programmierte Resultat. Danach wird das Resultat zurückgeliefert.

Tabelle 32: Zyklus eines MVC Requests

Dieser Zyklus wird pro Request abgehandelt (mit Ausnahme des ersten Schrittes). So treten keine Probleme mit der Nebenläufigkeit auf, ausser man erstellt `static` Variablen in den Kontrollern.

Das Management der Nebenläufigkeit in Bezug auf die Datenhaltung wird von der Datenbank selbst übernommen.

Jedoch können Probleme wie „Lost Update“ auftreten, da keine Timestamps oder Ähnliches verwendet werden. Dieses Problem ist bekannt und das Risiko wurde bewusst eingegangen, da die Probleme nur selten auftreten können und die zeitlichen Ressourcen beschränkt waren. Die einzelnen Abteilungen der Gema sind in Produktionslinien aufgeteilt, welche alle ihre klar definierten Produkte zugewiesen haben. Somit könnte es nur innerhalb einer Produktionslinie zu Konflikten führen. Da jedoch die Mitarbeiter das Produkt, welches sie eintragen, auch aktuell bearbeiten, tritt dieses Problem nur selten auf.

5.9.4. Behandlung von Sonderzeichen

Da URLs keine Sonderzeichen enthalten dürfen und gewisse Zeichen vom IIS falsch interpretiert werden, musste ein gewisses Escaping gemacht werden. Die Sonderzeichen werden vor der Verwendung als Link vom JavaScript in eine nutzbare Form gebracht und danach vom Server im API Controller wieder übersetzt. Nachfolgend ist eine Tabelle mit den ersetzten Sonderzeichen und ihren Platzhaltern.

Sonderzeichen	Platzhalter
„.“	„_po_“
„/“	„_sl_“
„:“	„_dp_“

Tabelle 33: Ersetzte Sonderzeichen

5.10. Barcode Client

5.10.1. Designentscheid

Die Entscheidung, den Barcode Client als separaten Fat-Client zu realisieren, wurde getroffen, weil der Zugriff auf einen COM-Port über eine Webapplikation nur sehr mühsam realisierbar ist. Bei einer Webapplikation wäre es zwar denkbar, den Inhalt des Barcodes über den USB-Keyboard Treiber in ein Textfeld ausgeben zu lassen. Dazu muss die Applikation jedoch immer geöffnet und der Fokus muss auf ein Textfeld gesetzt sein. Mit einem Fat-Client hat man die Möglichkeit, auf Events zu reagieren. Wird etwas eingescannt, kann er unabhängig, ob das Programm den Fokus besitzt oder nicht, darauf entsprechend reagieren.

5.10.2. Funktionalitäten

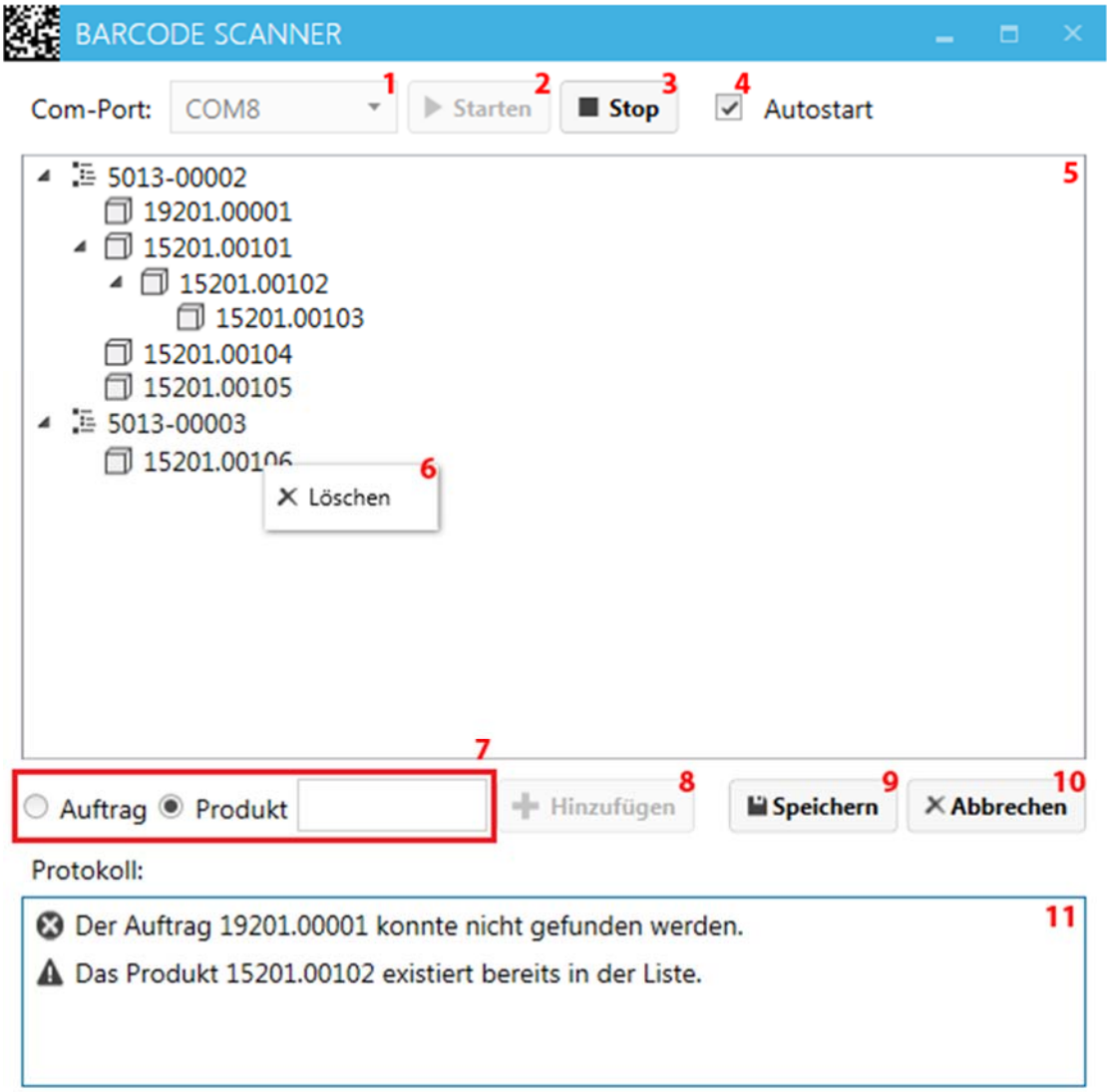


Abbildung 44: Barcode Client

1. Auswahl der verfügbaren COM-Port Schnittstellen.
2. Start-Button, um die Verbindung mit dem ausgewählten COM-Port herzustellen.
3. Stop-Button, um die Verbindung mit dem ausgewählten COM-Port zu beenden.
4. Auto-Start Checkbox, damit die Verbindung zum letzten ausgewählten COM-Port automatisch beim Öffnen des Barcode-Clients gestartet wird.
5. TreeView, die alle gescannten Aufträge und Produkte anzeigt. Die einzelnen Elemente der TreeView können mittels „Drag and Drop“ verschoben werden.
6. Rechtsklickmenü zum Löschen eines Auftrags oder Produkts.
7. Manuelle Eingabe eines Auftrags oder Produkts.
8. Hinzufügen-Button, um die manuelle Eingabe zu bestätigen
9. Speichern-Button zum Speichern der in der TreeView ersichtlichen Zuweisung.
10. Abbrechen-Button zur Löschung aller Einträge in der TreeView.
11. ListView, die wichtige Informationen anzeigt. Die Informationen werden in 4 Kategorien unterteilt: success, info, warning, error

5.10.3. Systemablauf

Damit der Benutzer mit der Zuweisung der Produkte starten kann, muss er den Barcode Client starten. Er muss als erstes den COM-Port auswählen, an dem der Barcode Scanner angeschlossen ist. Mit einem Klickt auf „Starten“ wird die Verbindung zum ausgewählten COM-Port hergestellt. Nun muss der Benutzer als erstes einen Auftrag scannen. Jedes Produkt, das nachfolgend eingescannt wird, wird diesem Auftrag zugeordnet. Es besteht ebenfalls die Möglichkeit, einem Produkt eine Komponente hinzuzufügen. Dazu muss zuerst ein spezieller „Komponenten Barcode“ gescannt werden. Solange nicht wieder der „Komponenten Barcode“ gescannt wird, werden alle gescannten Produkte als Komponente interpretiert. Eine Komponente ist ebenfalls ein Produkt, das jedoch dazu benötigt wird, um ein anderes Produkt herzustellen. Der Benutzer hat jederzeit die Möglichkeit, einen neuen Auftrag zu scannen. Alle nachfolgenden gescannten Produkte und deren Komponenten werden von nun an dem neuen Auftrag zugewiesen. Nachdem alle gewünschten Produkte einem oder mehreren Aufträgen zugewiesen wurden, kann der Benutzer mit einem Klick auf „Speichern“ die Zuweisung bestätigen.

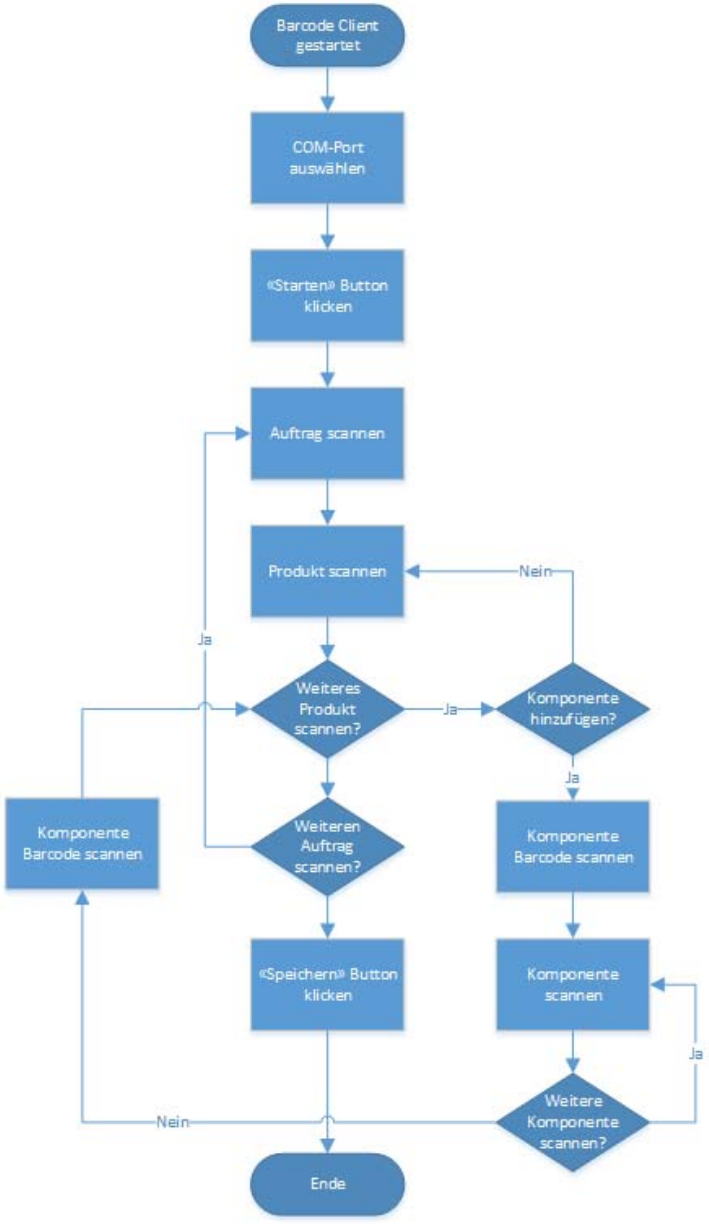


Abbildung 45: Systemablauf Barcode Client

5.10.4. Datenprüfung

Jeder eingescannte Barcode wird zuerst auf seine Richtigkeit überprüft. Erst nach einer erfolgreichen Überprüfung wird das gescannte Objekt der Liste hinzugefügt.

Zuerst wird sichergestellt, dass es sich überhaupt um einen GS1 DataMatrix Code handelt. Dazu müssen mindestens 2 Zahlen am Anfang der erhaltenen Daten stehen. Sollte eine dieser Überprüfungen fehlschlagen, wird eine Error-Meldung ausgegeben.

Anhand des jetzt ermittelten Application Identifiers (AI) wird überprüft, ob es sich um einen Auftrag, ein Produkt oder einen Steuerungscode für eine Komponente handelt. Für die verschiedenen Möglichkeiten gibt es unterschiedliche Anforderungen, die erfüllt werden müssen.

Produkt (AI: 21):

- Wurde bereits ein Auftrag eingescannt? Nein: Error-Meldung
- Existiert das Produkt bereits in der Liste? Ja: Error-Meldung
- Existiert das Produkt in der Datenbank? Nein: Error-Meldung
- Wurde das Produkt bereits einem Auftrag zugewiesen? Ja: Error-Meldung

Auftrag (AI: 91):

- Existiert der Auftrag bereits in der Liste? Ja: Error-Meldung
- Existiert der Auftrag in der Datenbank? Nein: Error-Meldung

Der Steuerungscode für die Komponente (AI: 92) benötigt keine zusätzliche Überprüfung.

5.10.5. Benutzerfeedback

5.10.5.1. Storyboard

Für ein Benutzerfeedback wird eine Storyboard Animation verwendet. Aufgrund der Meldung werden unterschiedliche Farbanimationen durchgeführt. Eine Farbanimation startet bei Weiss, geht innerhalb von 0.3 Sekunden auf die jeweilige Farbe über und kehrt dann wieder zurück zu weiss. Folgende Meldungen können auftreten:

Success-Meldung (grün): Tritt auf, wenn etwas erfolgreich durchgeführt werden konnte.

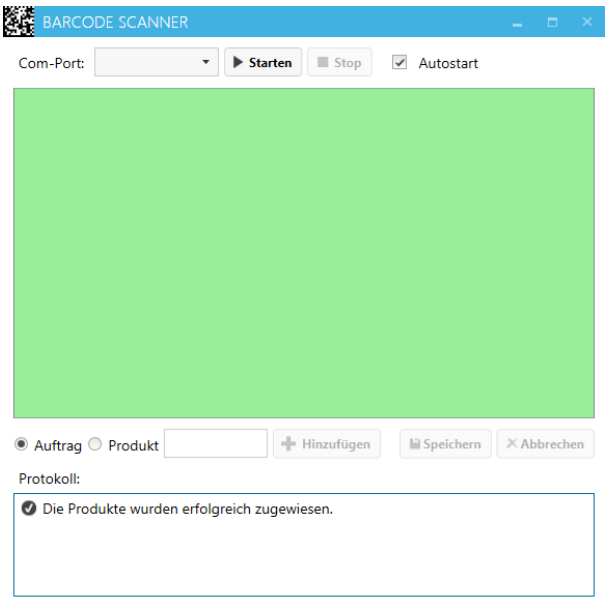


Abbildung 46: Barcode Storyboard Success

Info-Meldung (blau): Tritt bei einer allgemeinen Information für den Benutzer auf.

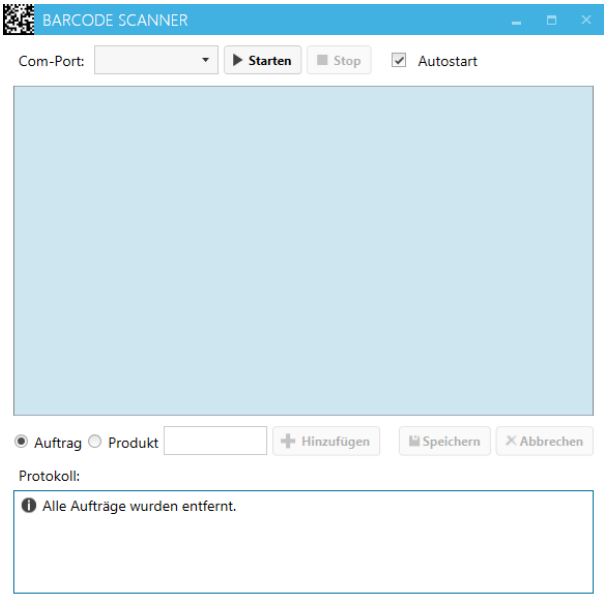


Abbildung 47: Barcode Storyboard Info

Warning-Meldung (gelb): Tritt auf, wenn etwas nicht erfolgreich durchgeführt werden konnte.

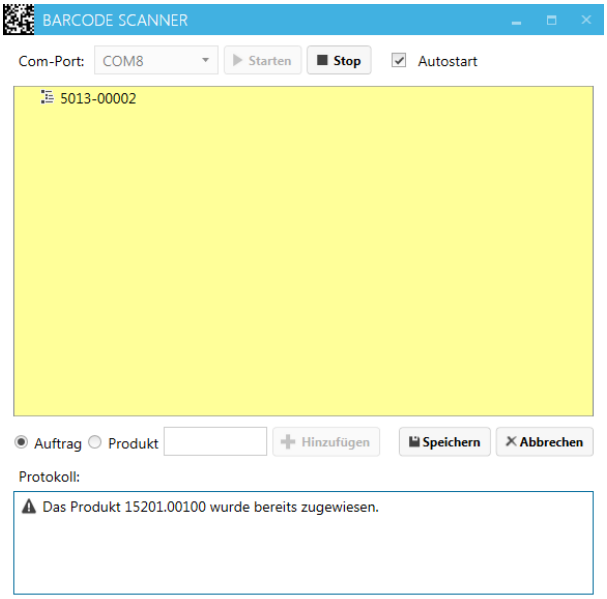


Abbildung 48: Barcode Storyboard Warning

Error-Meldung (rot): Tritt bei Fehlern auf.

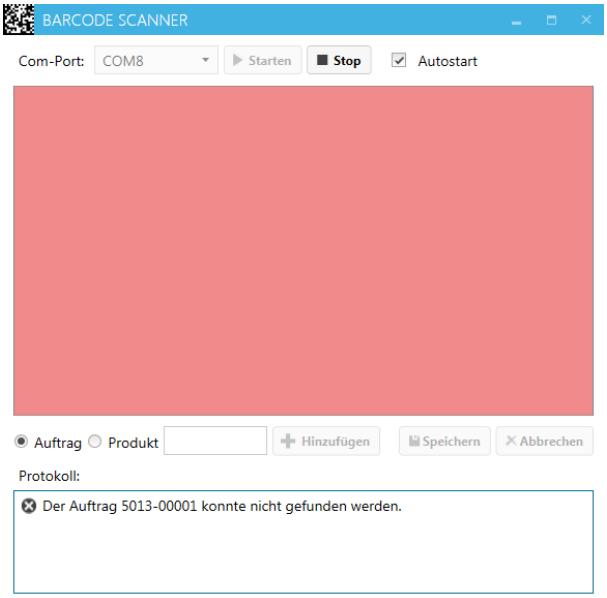


Abbildung 49: Barcode Storyboard Error

5.10.5.2. Protokoll

Wie auch beim Storyboard gibt es im Protokoll die vier verschiedenen Meldungen:

- Success
- Info
- Warning
- Error

Hier werden jedoch nur wichtige Informationen angezeigt, die dem Benutzer helfen. Dazu gehören vor allem Errors und Warnings, damit dieser nachvollziehen kann, warum etwas nicht wie gewünscht funktioniert hat.

5.10.6. Attached Properties

In den Objekten, wo ein gewünschtes Verhalten nicht standardmässig vorhanden ist, kommt das Attached Property zum Einsatz. Ein Attached Property wird als eine Art globales Property verwendet, das für jedes Objekt verwendbar ist. Es wäre auch möglich dieses Verhalten im Code-Behind zu realisieren. Jedoch wurde absichtlich darauf verzichtet, dort irgendwelche Implementationen zu realisieren.

Folgende Attached Properties wurden in der Klasse AttechedBehaviour realisiert:

Name	Beschreibung
PreviewMouseLeftButtonDown	Wird in der TreeView und für die ComboBox verwendet, um den PreviewMouseLeftButtonDownCommand, respektive UpdateComboBoxCommand aufzurufen, sobald die linke Maustaste über dem jeweiligen Objekt gedrückt wird.
MouseMove	Wird in der TreeView verwendet und immer dann aufgerufen, wenn die Maus bewegt wird. Der MouseMoveCommand führt dann die entsprechende Funktionalität aus.
DragOver	Ruft den DragOverCommand auf, wenn ein Element über ein anderes Element in der TreeView gezogen wird.
Drop	Wenn das „gezogene“ Element in der TreeView losgelassen (gedropt) wird, dann wird der DropCommand aufgerufen.
Close	Ruft den CloseCommand auf, wenn das Programm beendet wird, damit allenfalls eine noch offene Verbindung zur Seriellen Schnittstelle geschlossen wird.
ScrollToEnd	Sorgt dafür, dass die ListView für das Protokoll immer das unterste Element anzeigt.

Tabelle 34: Programmierte "attached properties"

Die ersten vier Attached Properties werden für die Drag and Drop Funktionalität benötigt.

5.11. Deployment

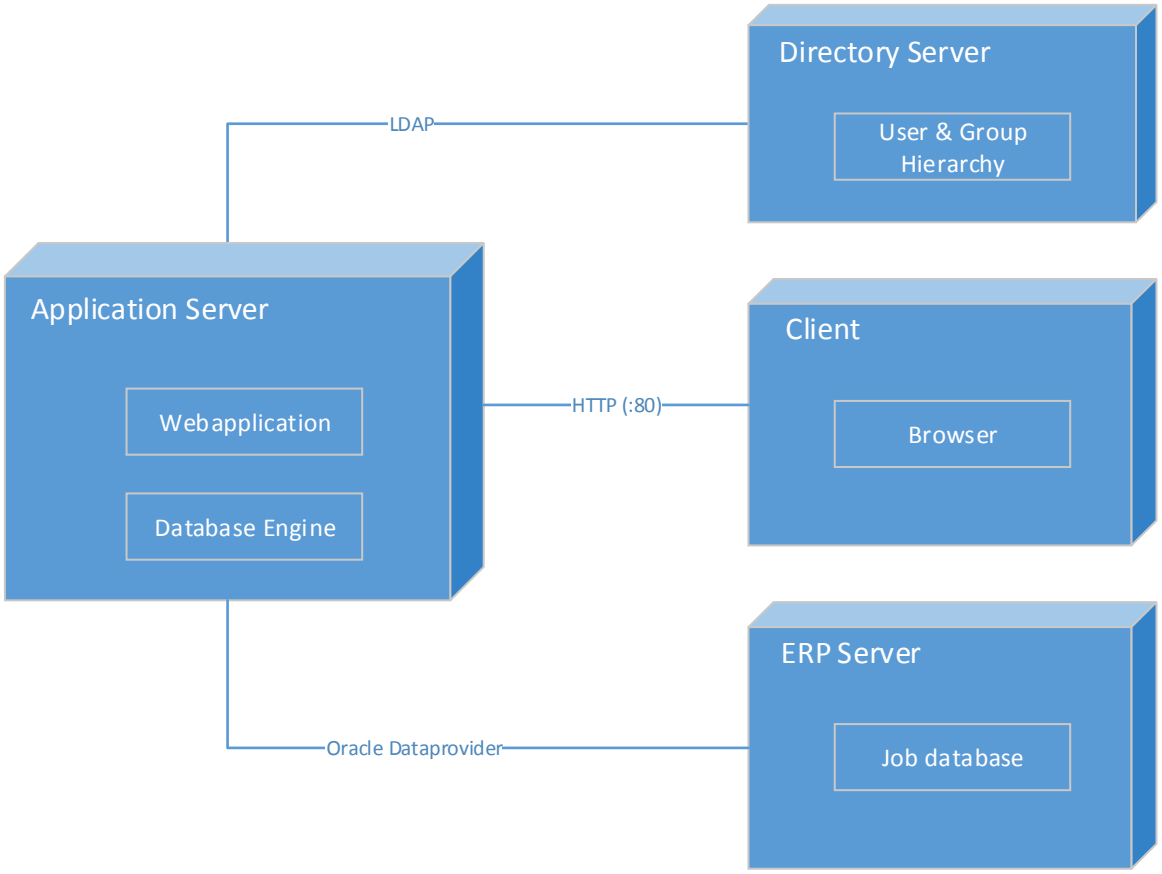


Abbildung 50: Deploymentdiagramm Produktivumgebung

Die produktive Umgebung teilt sich in vier Hauptnodes auf. Der zentrale Node ist der Application Server, welcher die ganze Applikation hostet. Darauf befindet sich der Webserver (IIS) mit der kompilierten Webapplikation. Des Weiteren ist auf dem Applikationsserver die Datenbank (MS SQL Express) enthalten.

Der Directory Server ist der Domänenkontroller der Gema. Er enthält die ganze Benutzerhierarchie der Firma. Per LDAP kann der Applikationsserver die Berechtigungen und Gruppenzugehörigkeiten der Benutzer auslesen und anhand dieser Informationen bestimmen, welcher Benutzer wie viele Rechte besitzt.

Der Client besitzt lediglich den Browser (nebst anderer Workstationapplikationen) und greift über das HTTP Protokoll über Port 80 auf den Applikationsserver zu.

Der ERP Server enthält alle auftragsspezifischen Daten und ist die zentrale Quelle für diese Informationen. Die Informationen werden periodisch vom Applikationsserver über den Oracle Datenprovider abgeholt und in der eigenen Datenbank gespeichert. So wird die Ausfallsicherheit der Webapplikation garantiert.

5.12. Datenhaltung

5.12.1. Schnittstelle

Als Schnittstelle für die Datenhaltung wurde das Entity Framework (EF) von Microsoft verwendet. Das EF ist ein sehr schneller und zuverlässiger OR-Mapper, welcher sich in der C# Community sehr etabliert hat. EF wurde von Microsoft als OpenSource Software freigegeben.

Prinzipiell wird EF auf zwei Wegen verwendet:

- Code-First: Erst wird die Datenstruktur modelliert und anhand des Modells die Datenbank erstellt
- Data-First: Die Datenbank wird unabhängig vom Programmcode modelliert und anhand der Datenbank wird das EF – Modell erstellt

In der Webapplikation wurde der Code-First Ansatz verwendet, da die Datenstruktur grundlegend überarbeitet wurde. Nachdem das Modell (Abschnitt Entity Framework Model) erstellt ist, generiert das EF einen Entity[8]-Container. Dieser Container verwaltet alle Verbindungen zur Datenbank und cached die Abfragen. Muss beispielsweise im Businesslayer ein Zugriff auf die Datenbank erfolgen, so ist dies folgendermassen möglich:

LINQ Query Syntax

```
using(var db = new EntityContainer())
{
        var query = from o in db.Objects
                    select o;
        return query;
}
```

LINQ Lambda Syntax

```
using(var db = new EntityContainer())
{
        var query = db.Objects.Select(o => o);
        return query;
}
```

Das „using“-Statement sorgt für den reibungslosen Auf- und Abbau der Verbindung zur Datenbank.

[8] Die einzelnen Klassen werden im EF als Entities bezeichnet

5.12.2. Entity Framework Model

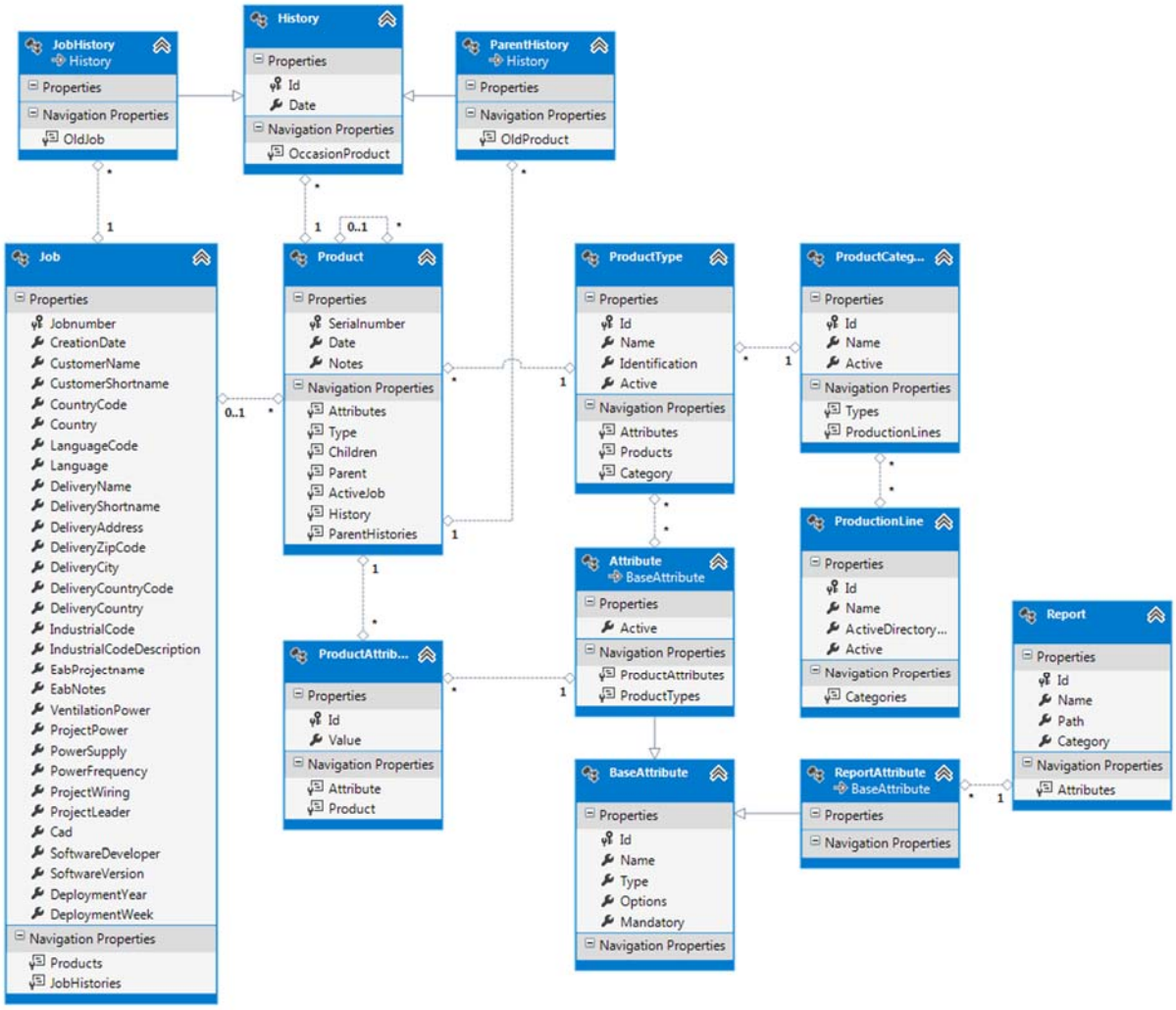


Abbildung 51: Entity Framework Model

5.12.3. Abweichungen zum Designmodell

Die eigentliche Implementierung weicht vor allem im „Auftrags“ Objekt vom Domainmodell ab. Geplant war das Aufteilen der Kunden- und Lieferdaten vom eigentlichen Auftrag. Jedoch würde dies die Synchronisierung erheblich erschweren und zu keinem befriedigenden Ergebnis führen. Da die Synchronisation über den Oracle Datenprovider läuft, wurde mittels eines SQL-Merge Scripts die ERP View mit der eigenen Auftragstabelle synchronisiert. Da die Daten der Quelle nicht normalisiert sind und in einer zusammengefassten View daherkommen, wurden sie nicht mehr normalisiert. Die Auftragsdaten werden jedoch aktualisiert, falls sich im ERP zum Beispiel eine Kundenadresse ändert.

5.12.4. Spezialisierungen

5.12.4.1. User defined functions

Da das EF keine UDFs definieren kann, müssen diese auf eine spezielle Art und Weise genutzt werden. SProcs können ebenfalls nicht ohne Aufwand genutzt werden. Für die Webapplikation mussten gewisse Funktionen in die Datenbank ausgelagert werden:

- Vergleich von Nummern (grösser als, kleiner als)
- Vergleich von Daten (vor einem bestimmten Datum, nach einem bestimmten Datum)

Um eine solche UDF zu nutzen, müssen folgende Schritte unternommen werden:

1. Die Funktion definieren

```
CREATE FUNCTION DateObjectLessThan
       (
       @left date,
       @right date
       )
RETURNS bit
AS
BEGIN
       declare @ret bit;
       select @ret = case when @left < @right then 1 else 0 end;
       return @ret;
END
```

2. Die Funktion importieren über EF-Modelldesigner

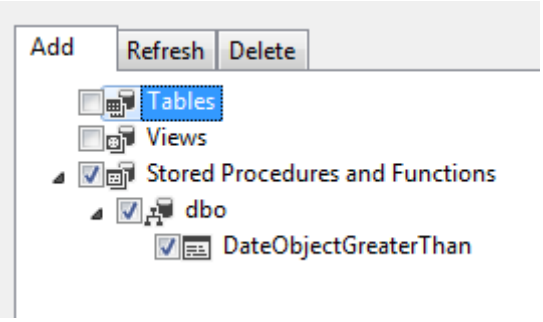


Abbildung 52: Importieren von SProcs und UDFs

3. Funktion im Container definieren

```
public partial class EntityContainer
{
    [EdmFunction("Entities.Store", "DateObjectLessThan")]
    public bool DateObjectLessThan(DateTime left, DateTime right)
    {
        throw NotImplementedException("Not usable without db");
    }
}
```

Nach diesen drei Schritten sind die UDFs normal über den DB-Kontext nutzbar:

```
query = query.Where(o => db.DateObjectLessThan(o.CreationDate, searchQuery.DateTo.Value));
```

Wichtig: wird die Datenbank aus dem EF-Modell generiert (in beiden Ansätzen so), so werden die Funktionen nicht erstellt. Sollte also die Datenbank gelöscht werden und mittels eines Scripts generiert werden, so müssen die Funktionen wieder eigens implementiert werden.

5.13. Zusätzliche Programme

5.13.1. DataConverter

Der Datenkonverter dient dazu, die Daten aus der alten Datenbank zu übernehmen und im neuen Format in der neuen Datenbank zu speichern. Diese Datenübernahme erfolgt einmal bei der Einführung der Applikation. Somit kann man die alte Applikation ganz abschalten und die Benutzer müssen nicht in zwei Systemen nach den Daten suchen.

Hierbei handelt es sich um einen einmaligen ETL (Extract, Transform, Load) Prozess. Das heisst, die Daten müssen zuerst von der alten Datenbank geladen, danach in die neue Form transformiert und abschliessend in die neue Datenbank gespeichert werden. Es gibt viele Tools, mit welchen sich ein solcher Prozess abbilden lässt. Das Tool „Pentaho Data Integration“ wurde deshalb für den Datenkonverter in Betracht gezogen [9]. Aufgrund des kleinen Know-hows wurde dann jedoch entschieden, den gesamten Datenkonverter mittels SQL zu realisieren.

Das Know-how im Team war hier entsprechend höher, und da es sich um eine einmalige Übernahme handelt sind die vielen und teilweise eher schwer verständlichen Codefragmente vertretbar. Die Scripts sind im Repository im Ordner „Data Converter“ enthalten und werden in folgender Tabelle genauer beschrieben:

File	Beschreibung
00_RUN_FULL_DATACONVERTER.sql	Über dieses File kann der Datenkonverter gestartet werden. Damit die richtigen Files ausgeführt werden, kann man zuerst den Dateipfad setzen. Um das File starten zu können, muss es im SQLCMD Mode ausgeführt werden.
01_GeneralSettings.sql	Ist zuständig für die Erstellung sämtlicher Arttribute sowie die Definition von Variablen. Die Zuweisung der Attribute zu konkreten Typen geschieht erst später.
02_DataConverterLine1.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Handgerät“, “Erweiterung“ und „Pumpe“. Sorgt dafür, dass in den Typnamen kein „®“ mehr vorhanden ist.
03_DataConverterLine2.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Lichtgitter“, “Lichtgitter Controller“, „Drehgeber“, „Schrank“, „Würfel“ und „Würfel Mainboard“.
04_DataConverterLine4.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Achse“, “Motor“ und „Einschub“.
05_DataConverterLine5.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Pistole“, “Kaskade“ und „Spannungstester“.
06_DataConverterLine6.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Sieb“ und „Pulver Management“. Da die Seriennummern der Pulverzentren teilweise schon von anderen Produkten benutzt wurden, wird bei Bedarf der Prefix „PC“ vor die Seriennummer gehängt.
07_DataConverterLine7.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Kabine“, „Frischpulversystem“, „Siebmaschine“ und „Handkabine“. Da die Seriennummern der Frischpulversysteme teilweise schon von anderen Produkten benutzt wurden, wird bei Bedarf der Prefix „FPS“ vor die Seriennummer gehängt.
08_EAB.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Steuerung“ und „Spannungsquelle“. Falls eine Steuerung keine Seriennummer besitzt, wird sie auf „ctrl“ + Laufnummer gesetzt. Ebenso werden die Steuerungen mit einem Bindestrich in der Seriennummer nicht übernommen. Die Attribute einer Steuerung werden im neuen System direkt dem Auftrag zugewiesen.
09_Purchasing.sql	Übernimmt sämtliche Produkte und Typen der Kategorien „Filter“, „Feuerschutz“ und „Anderes Gerät“.
10_Final_Connections.sql	Stellt die linienübergreifenden Verbindungen zwischen den Produkten her.
11_Additional_Jobinfo.sql	Übernimmt die zusätzlichen Auftragsinformationen.
12_Line1_CleanTypes.sql	Führt gleiche Typen mit Schreibfehlern zusammen und weist sie der richtigen Kategorie zu.
13_History.sql	Sorgt dafür, dass die History Einträge erstellt werden.

Tabelle 35: Beschreibung der Datenkonverter Files

5.13.1.1. Voraussetzungen

- Damit der Datenkonverter korrekt läuft, ist die Datenbank neu aufzusetzen.
- Die Einträge für die Aufträge müssen schon in der Datenbank vorhanden sein. Falls ein Produkt mit einem Auftrag verknüpft werden soll, der noch nicht in der Datenbank vorhanden ist, wird ein Dummy-Auftrag angelegt.

5.13.1.2. Umsetzung

Um über die Daten zu iterieren, wurden Cursors verwendet. Dazu gibt man zuerst die gewünschte SQL Abfrage an und kann danach mit einem Do-While durch die Ergebnisse iterieren. Die Cursors sind jeweils als „Fast_Forward“ deklariert worden. Dies bringt erhebliche Performancegewinne mit sich, da die Cursors nur in eine Richtung gelesen werden können. Für unsere Datenbearbeitung reicht dies jedoch aus.

Die meisten Inserts wurden durch ein Try-Catch geschützt. Falls ein Fehler auftreten sollte, wird die Anzahl Fehler um eins erhöht und die Fehlermeldung ausgegeben. Danach wird der nächste Eintrag aus dem Cursor ausgelesen. Am Ende des Skripts werden danach die Gesamtanzahl Fehler ausgegeben.

5.13.1.3. Test

Für den Datenkonverter wurden keine Tests geschrieben, da dies den Rahmen gesprengt hätte. Während der Implementation wurde er jedoch mehrmals auf die produktiven Daten angewandt und das Ergebnis danach von den Endbenutzern beurteilt. Anschliessend wurde dieses Feedback wieder in den Datenkonverter aufgenommen.

Auf dem Server werden mithilfe des Datenkonverters innerhalb von ca. 6min über 270‘000 Produkte übernommen.

5.13.2. SQL Server Backup

Um die Daten der Webapplikation zu sichern, könnte ein spezialisierter Backup-Agent eingesetzt werden. Da jedoch keine SQL Backup-Agenten bei der Gema eingesetzt werden, wird die gesamte Datenbank geklont und gesichert. Zu diesem Zweck wurde ein geplanter Task auf dem Server eingerichtet, welcher ein Powershell-Script ausführt. Das Script bereitet das Logfile vor und startet dann die SQL Backup Abfrage per „sqlcmd“. Ist die Abfrage erfolgreich, so kann das normale Backup die Datei wegsichern. Schlägt der Backup fehl, so wird eine E-Mail generiert, welche den im Script definierten Empfängern das Log und eine Fehlermeldung zusendet.

Die Scripts sind im Repository unter den Namen

- backup-query.sql
- backup-script.ps1

zu finden.

5.13.3. SQL Server Jobmerge

Damit in der Webapplikation immer die neusten Auftragsdaten vorhanden sind, werden diese direkt mit der ERP Datenbank synchronisiert. Diese ERP Datenbank enthält alle relevanten Daten, wie Kundennummer, Kundenadresse und Lieferadressen. Um diese Daten zu synchronisieren, wurde in der alten Lösung ein kleines CMD-Tool eingesetzt, welches anhand eines Zeitstempels die neuesten Einträge gesucht und synchronisiert hat.

Neu wird diese Aufgabe über ein SQL Script mittels geplantem Task realisiert, da dies deutlich schnellere Laufzeiten aufweist. Ein spezieller SQL Befehl „merge“ synchronisiert dabei zwei Tabellen und vergleicht die Einträge. So können im SQL Code selbst verschiedene Aktionen für die nicht gefundenen Einträge sowie für die gefundenen, aber nicht exakt gleichen Einträge definiert werden. Um dieses SQL Script anzustossen, wird wie erwähnt ein geplanter Task genutzt. Dieser Task startet ein Powershell-Script, welches die Fehlerbehandlung übernimmt. Analog zum Backupscript startet das angestossene Powershell-Script den SQL Merge mittels „sqlcmd“ und schreibt die Ergebnisse in ein Logfile. Schlägt die Synchronisierung fehl, so wird eine E-Mail an die eingetragenen Personen versendet.

Die Scripts sind im Repository unter den Namen

- merge-query.sql
- merge-script.ps1

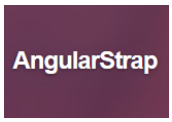
zu finden.

5.13.4. Webapplication-Deploymentscript

Damit für die Gema das Deployment der Webapplikation möglichst einfach ist, wurde ein kleines Script erstellt, welches diese Aufgabe übernimmt. Es erstellt ein Logfile mit den Resultaten und Ausgaben des Deployments und passt die Ordnerberechtigungen an. Dies geschieht, da ein spezieller Ordner „Reports“ erstellt werden muss, auf welchem die Nutzer Schreibzugriff haben müssen. In diesem Ordner werden die Reportdefinitionfiles (.rpt) abgelegt.

Das Script ist im Repository unter dem Namen „websrvDeployment-script.ps1“ zu finden.

5.14. Eingesetzte Technologien

Technologie	Version		Beschreibung
AngularJS	1.1.4		AngularJS ist ein MVVM Framework für Webprogrammierer. Es ermöglicht die getrennte Programmierung von GUI-Logik und Clientseitiger Backend-Logik.
AngularStrap	0.7.3		AngularStrap sind verschiedene AngularJS Direktiven, welche die Bootstrap Elemente kapseln. So wird das Angular – Binding auf Bootstrap Komponenten ermöglicht.
Angular-ui.Bootstrap	0.3.0		Angular-UI besteht ebenfalls aus AngularJS Direktiven, welche Bootstrap Elemente kapseln. Jedoch handelt es sich dabei um Basiskomponenten, welche wenig Logik mitführen.
ASP.NET MVC4	4.0.20710		Die Grundlage der Webapplikation stellt ASP.NET dar. Es ist das Webframework der Microsoft und läuft auf einem IIS Server mit .NET Framework 4.5. Enthalten darin ist die Razor Engine, welche das View-Rendering übernimmt.
AttributeRouting	3.5.6	-	Erweiterung der ASP.NET Applikation. Ermöglicht das attribuierte Routing von Client-Anfragen.
Bootstrap Datepicker	1.0.2	-	Bootstraperweiterung, welche ein Datumsfeld für HTML realisiert.
Crystal Reports	13.0.5		Reporting-Framework der SAP. Liefert einen Designer für das Erstellen und Warten der Reports. Die Runtime ermöglicht dem Server das Generieren von erstellten Reports.
Elmah.MVC	2.0.2	-	ELMAH (Error Logging Modules and Handlers) ist ein Framework, welches das automatisierte Logging der Webapplikation übernimmt. Es kann vielfältig konfiguriert werden und bietet zahlreiche Logging-Features.
jQuery	1.9.1		jQuery ist die heute am gängigsten genutzte JavaScript Library, die es gibt. Man findet beinahe keine Webapplikation, welche in „normalem“ JS programmiert ist. jQuery bietet viele Funktionen zur DOM Manipulation und ähnlichem.
Json.NET	5.0.3	-	Für sämtliche JSON Manipulationen in .NET ist Json.NET eine umfangreiche Library, welche von allen grossen Projekten genutzt wird. Sie bietet alle Manipulationen sowie Konvertierung von und zu JSON.
MahApps.Metro	0.10.1.1		Dies ist eine reine „Style-Library“. Sie wird verwendet, um in WPF Applikationen die heute gängigen Windows 8 Metro-Styles zu nutzen. Der Einarbeitungsaufwand ist verschwindend klein und die Resultate gross.

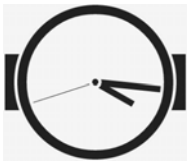
Technologie	Version		Beschreibung
momentJS	2.0.0		Diese JavaScript-Library sorgt für ein effizientes und korrektes Manipulieren von Daten in Browsern. Jeder Browser implementiert sein eigenes Datum-Parsing, was gezwungenermassen zu Problemen führt. momentJS hilft, dies zu umgehen und eine einheitliche Schnittstelle zu haben.
Moq	4.0.10827		Moq ist ein Mocking-Framework für .NET. Es wird verwendet, um Klassen für das Unit-Testing zu mocken. So kann beispielsweise der eingeloggte Benutzer gemockt und für Tests benutzt werden.
Morovia DataMatrix Font&Encoder	5		Ist ein Produkt von Morovia, welches unter anderem eine Truetype Font für das Anzeigen von Datamatrix-Codes beinhaltet. Wird für das Reporting eingesetzt, um Barcodes für die Typenschilder zu generieren.
Ng-Upload	0.2.0	-	Ng-Upload ist eine kleine Direktive für AngularJS, um ein korrektes Fileuploading in Angular zur Verfügung zu haben.
Twitter Bootstrap	2.3.0		Das meistgenutzte Designframework der Welt. Bootstrap stellt eine riesige Bibliothek an CSS Klassen zur Verfügung, welche ohne grossen Designaufwand eine schöne Webapplikation entstehen lassen.
Unity	2.1.505.2		Depencency Injection für ASP.NET MVC. Dient der Entkopplung der Businesslogik und erhöht die Testbarkeit der entsprechenden Softwareschichten.

Tabelle 36: Eingesetzte Technologien

6. Qualitätssicherung

6.1. Kennzahlen

Da sich die Codequalität nicht einfach beschreiben lässt, sind nachfolgend einige Kennzahlen aufgelistet und illustriert, welche einen Überblick über die Applikationen und ihre Schichten geben sollen.

Das ganze Projekt (inkl. Barcodeclient, jedoch ohne Tests) umfasst 3342 Zeilen Code, welche folgendermassen aufgeteilt sind:

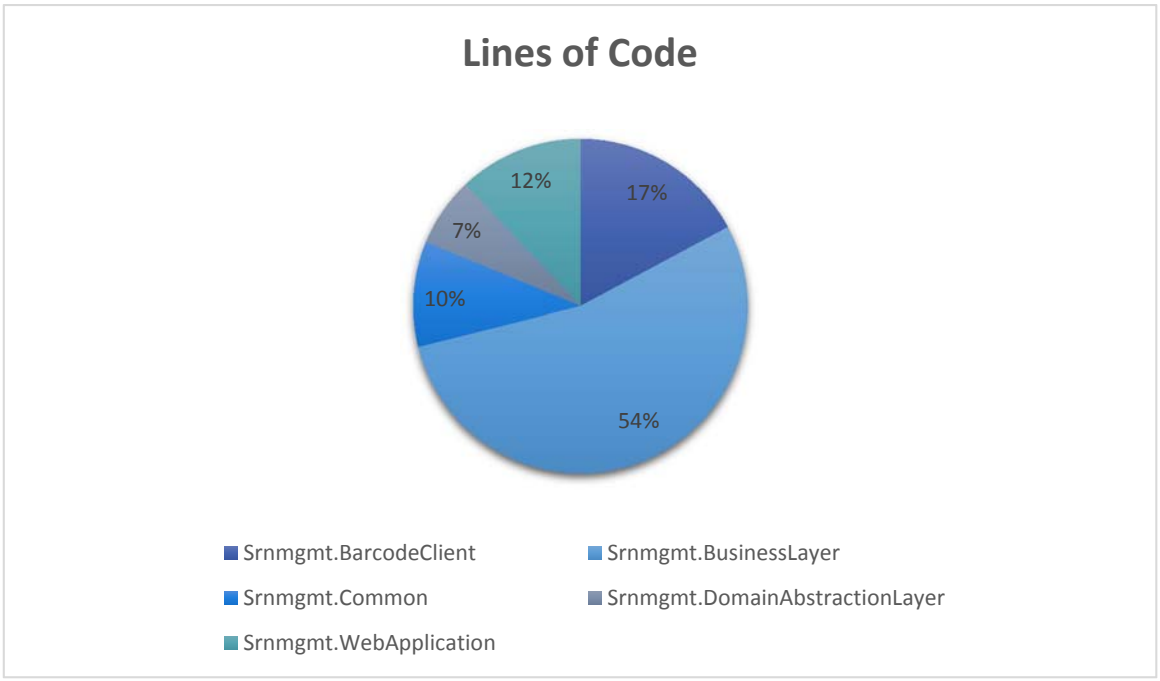


Abbildung 53: Lines of Code

Die zentrale Schicht ist der Businesslayer. In ihm sind alle relevanten Operationen von und zur Datenbank enthalten. Das Ziel dieser Architektur ist es, eine zentrale Schnittstelle zur Datenschicht zu haben.

Der nächste aufgeführte Wert ist das „Class Coupling“ (kleinere Werte sind besser). Hierbei handelt es sich um einen Index, wie viele Klassen einzigartig und nicht abhängig von anderen sind. Ein gutes Softwaredesign hat eine hohe Kohäsion und eine kleine Kupplung der Klassen. Bewertet werden sämtliche Referenzen wie Rückgabewerte, Variablen, Templates und vieles mehr.

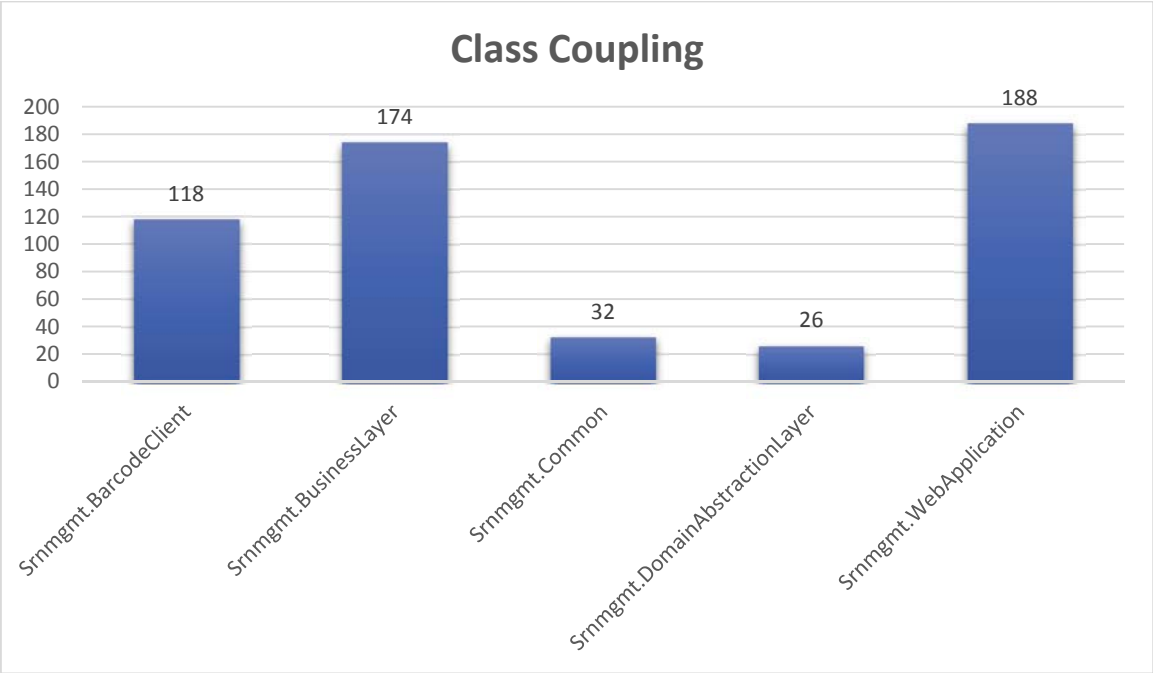


Abbildung 54: Class Coupling

Die Kopplung wurde durch die Einführung des Dependency Injection (DI) Prinzips deutlich verringert.

Eine weitere wichtige Kennzahl ist die „Cyclomatic Complexity“ (kleinere Werte sind besser). Sie beschreibt die Anzahl der verschiedenen Codepfade, welche beschritten werden können. Grundsätzlich werden alle verschiedenen Weichen (IF, WHILE, FOR, etc.) gezählt.

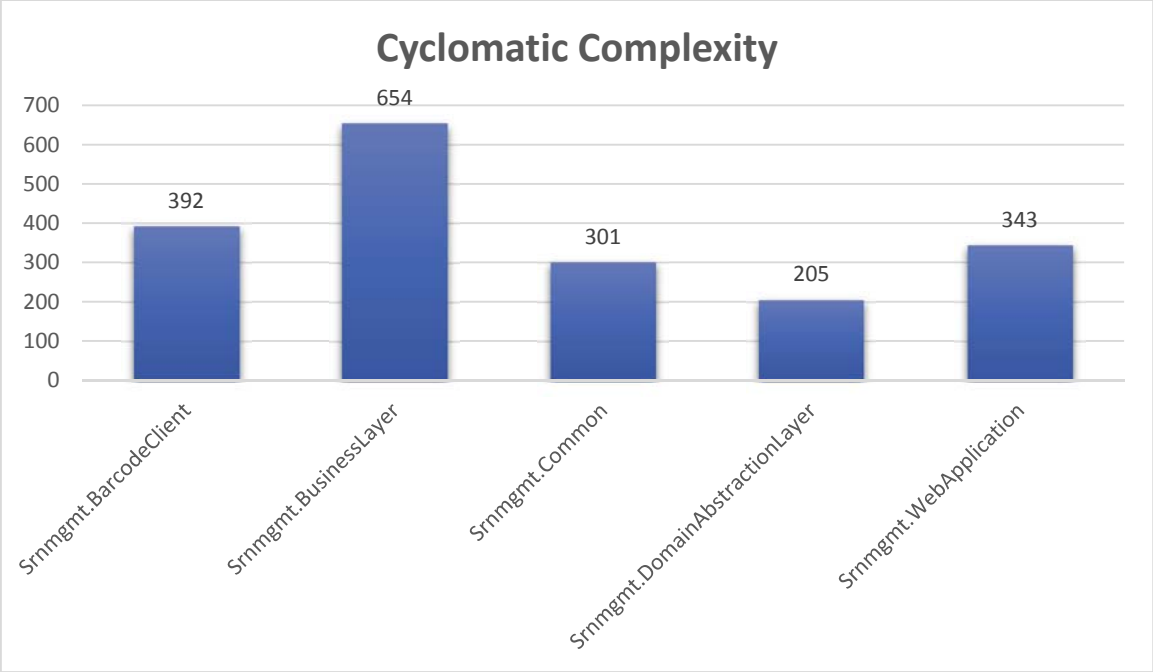


Abbildung 55: Cyclomatic Complexity

Aus den oben erwähnten Kennzahlen und der Halstead-Metrik[9] wird die „Maintainability“ berechnet. Dieser Index gibt an, wie einfach der Code zugänglich ist. Ist dieser Wert hoch, so ist der Code verständlicher und einfacher zu warten bzw. zu bearbeiten. Die Formel für diesen Index ergibt sich wie folgt:

$$Index = MAX\left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171}\right)$$

$$HV = Halstead\ Volume$$
$$CC = Cyclomatic\ Complexity$$
$$LoC = Lines\ of\ Code$$

Die Indizes für das Projekt:

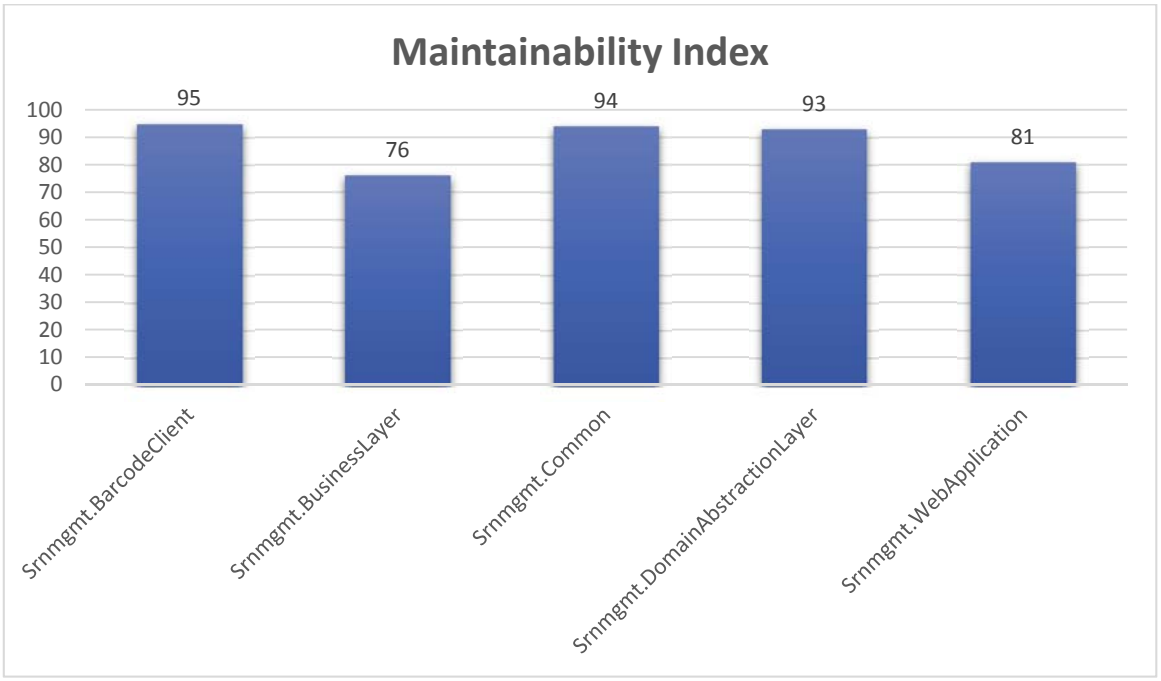


Abbildung 56: Maintainability Index

Der Maintainability Index kommt auf eine Skala von 0 bis 100 zu liegen. Um sich ein Bild machen zu können, sollte man die Indexrange kennen (hier gilt: höhere Werte sind besser):

- 0-9: Schlecht
- 10-19: Mittel
- 20-100: Gut

[9] Eine Softwaremetrik welche die Komplexität eines Codes angibt. [31]

Die letzte wichtige Kennzahl zur Qualitätssicherung ist die „Code Coverage“ der Testprojekte. Sie gibt an, wie viele verschiedene Codepfade getestet sind. Die durchschnittliche Coverage des Projektes beträgt 60.45%. Diese setzt sich folgendermassen zusammen:

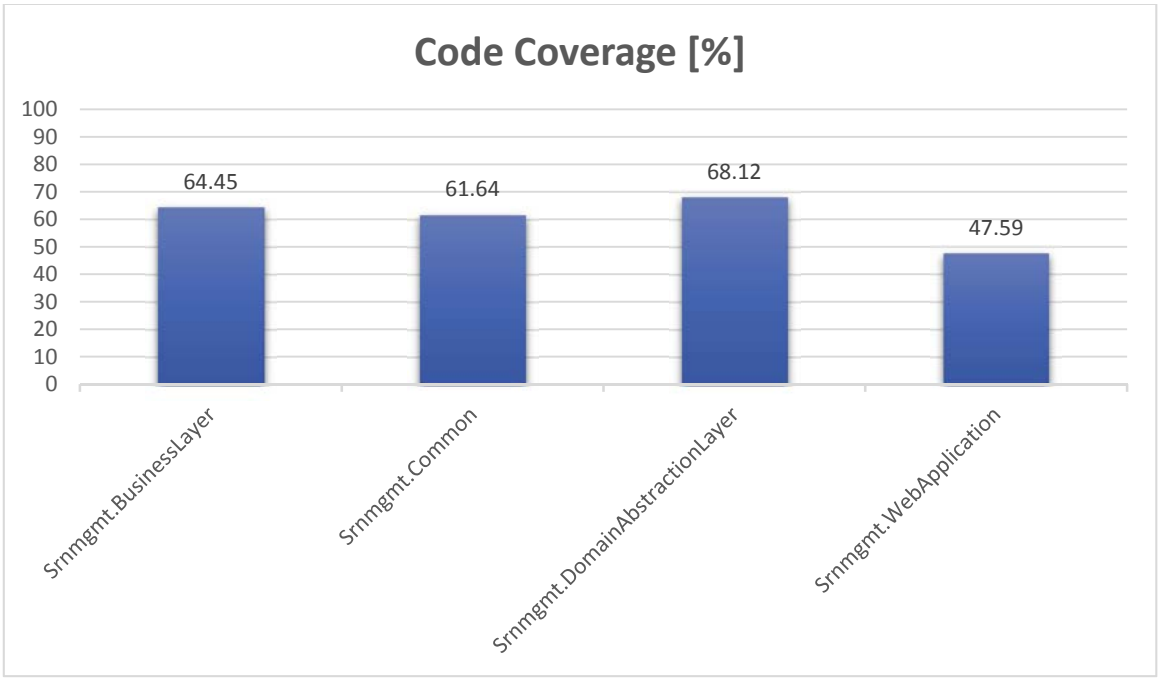


Abbildung 57: Code Coverage

Diese Zahlen sind eher tief, da immer eine Abdeckung von 100% angestrebt werden sollte. Da aber selbst mit einer 100%-Abdeckung keine Garantie für gute Tests gegeben ist, wurde der Fokus des Testens auf Tests mit den effektiven Endbenutzern gesetzt.

6.2. Testphase bei Gema GmbH

Anfangs Mai 2013 wurde ein erster Prototyp der Applikation bei der Gema GmbH vor Ort in die produktive Umgebung aufgenommen. Die Benutzer wurden dazu angehalten, sich ein wenig mit der neuen Applikation vertraut zu machen und Fehler und fehlende Funktionen zu melden. Bis zum Ende des Projekts wurden insgesamt acht Versionen verteilt und getestet. Während am Anfang teilweise auch noch gröbere Implementationsfehler gefunden wurden, entdeckte man gegen Ende des Projekts mehrheitlich fachliche Fehler.

Ganz am Schluss des Projekts wurde eine einwöchige Testphase durchgeführt. Die Mitarbeiter erhielten zuerst eine kleine Schulung. Danach mussten sie während dieser Woche parallel das alte und das neue System benutzen. Die neue Applikation wurde so einem eigentlichen Härtetest unterzogen und es konnte überprüft werden, ob die Funktionalität des alten Systems vollständig abgedeckt ist.

Bei den meisten gefundenen Fehlern ging es um kleine Layout Anpassungen bei den Reports. Ebenso wurden noch ein paar wenige fachliche Fehler beim Datenkonverter gefunden (z. B. Typen doppelt vorhanden mit leicht unterschiedlicher Bezeichnung). Diese Fehler konnten noch behoben werden. Folgende Anpassungen bzw. Funktionsanfragen konnten jedoch aufgrund des Feature- und Codefreezes nicht mehr berücksichtigt werden:

- Typenschilder direkt aus der Produktansicht drucken. Ist technisch relativ einfach umzusetzen. Workaround: Typenschilder des entsprechenden Auftrags generieren, nur die Seite des gewünschten Typenschilds drucken
- Typenschilder direkt drucken, ohne Filedownload und Vorschau. Erfordert einen hohen Aufwand, da das Drucken eines Filedownloads aus dem Browser nicht so einfach ist. Der Benutzer würde sich ca. 2 Klicks sparen.

Die Akzeptanz der Benutzer ist von Linie zu Linie unterschiedlich.

- Bei der Linie 5, wo viele Produkte auf Vorrat eingetragen werden, sind die Mitarbeiter ein wenig kritisch. Dies liegt daran, dass im alten System mit sehr spezialisierten Masken das Erfassen teilweise leichter gefallen ist (wenn auch aufgrund der Performanceprobleme nicht unbedingt schneller). Das grösste Problem dabei ist, dass die Teststationen für die Produkte dieser Linie noch nicht fertiggestellt sind. Die Erfassung der Produkte geschieht momentan noch von Hand. Mit der Einführung der Teststationen, welche automatisch CSV-Files für die Erfassung erstellt und die manuelle Erfassung überflüssig macht, wird die Akzeptanz sicherlich drastisch ansteigen.
- Die anderen Linien sind mit dem neuen System sehr zufrieden, wobei eine Linie es kaum erwarten kann, bis das alte System abgelöst wird

Es ist klar, dass die Einführung einer neuen Applikation immer ein wenig Zeit benötigt, bis sich alle Mitarbeitenden damit identifizieren können.

6.3. Abnahmetest

Am Schluss des Projekts wurde ein Abnahmetest beim Kunden vor Ort durchgeführt. Nach dem Abnahmetest konnten 71 der 81 Abnahmetests auf Grün gesetzt werden, und nur gerade zwei mussten auf Rot gesetzt werden.

Das bedeutet, dass fast 90% aller Tests erfolgreich waren und nur wenig mehr als 2% nicht zufriedenstellend sind. Dies ist ein gutes Ergebnis, vor allem wenn man bedenkt, dass die meisten Fehler im Administrationsbereich liegen. Die Admins sind ausgebildete IT-Fachkräfte, die für dieses Programm geschult wurden und den weiteren Betrieb sicherstellen.

In der folgenden Liste sind die vorhandenen Fehler und Unschönheiten aufgeführt:

- *Kritisch*: Bei der Mehrfacheintragung eines Produkts kann es zu einem unerwarteten Fehler kommen. Wenn beispielsweise die Seriennummer „5“ im System vorhanden ist und eine Mehrfacheintragung ab der Seriennummer „1“ für 10 Produkte durchgeführt wird, wird kein einziges Produkt eingetragen. Grund dafür ist, dass die Funktion Produkte mit der Seriennummer von 1-10 einfügen will und die 5 bereits vergeben ist. Sauber wäre, wenn er die 5 überspringen würde und dafür noch die Seriennummer 11 vergeben würde.
- *Kritisch*: Wenn beim Hochladen eines Reportfiles ein falsches File (z.B. .txt Datei anstatt .rpt) hochgeladen werden soll, wird einfach ein leerer Eintrag erstellt. Diesen Eintrag kann man danach auch nicht mehr löschen. Es sollte jedoch schon beim Upload geprüft werden, ob das File gültig ist oder nicht. *(Nur Adminbereich)*
- Wenn bei „Occasion hinzufügen“ eine Seriennummer eingegeben wird, die noch nirgends verlinkt ist, wird das Produkt dennoch dem Auftrag zugewiesen. Eigentlich müsste eine Fehlermeldung kommen.
- Überall dort, wo Filter und Pagination zusammen auf eine Tabelle angewandt werden, tritt dasselbe Problem auf. Es wird zwar sauber gefiltert, jedoch wird die Pagination nicht nachgeführt. *(Nur Adminbereich)*
- Sowohl bei der Produktionslinie wie bei den Produkttypen gibt es noch Probleme mit dem löschen. *(Nur Adminbereich)*
    - Die Produktionslinie kann nicht gelöscht werden, wenn noch Produktkategorien zugewiesen sind, was nicht sinnvoll ist. Bei den Produktionslinien wird das Attribut „Aktiv/Inaktiv“ gar nicht benötigt, da es keine Konsistenzprobleme geben kann.
    - Sobald Produkttypen noch Attribute zugewiesen haben, können sie nicht gelöscht, sondern nur auf inaktiv gesetzt werden. Bei den Produkttypen dürfen allerdings die zugewiesenen Attribute keine Auswirkung auf das Löschen haben. Wenn ein Produkttyp gelöscht werden soll, muss die Verbindung zwischen Typ und Attribut getrennt werden.
- Beim Hochladen eines Reports kann das File auch weggelassen werden, obwohl es ein Pflichtfeld ist. *(Nur Adminbereich)*
- Bei einem Attribut vom Typ Selection kann das Pflichtfeld „Optionen“ umgangen werden, indem einfach ein Leerzeichen mit Komma eingetippt wird. Dies ist allerdings nicht sehr sinnvoll. *(Nur Adminbereich)*

Die gefundenen Fehler können nun priorisiert und falls gewünscht in einer neuen Version behoben werden. Erfreulich ist, dass kein Fehler entdeckt wurde, der dem Arbeiten mit der neuen Applikation im Weg steht. Einer erfolgreichen Umstellung auf das neue System steht also nichts mehr im Weg.

7. Ausblick

Ursprünglich wurde der Go-Live beim Businesspartner innerhalb der Arbeit geplant. Aufgrund verschiedenster Ferienabsenzen bei der Gema wurde dieser Punkt jedoch auf den 1. Juli verschoben. Das Projektteam wird die Gema bei der Einführung am 1. Juli unterstützen und beratend zur Seite stehen. Ebenfalls wird dem zuständigen Entwickler bei der Gema eine kurze Einführung in die Projektstruktur gegeben. Während der Einarbeitungsphase wird auch ein gewisser Support per Mail möglich sein.

Zu den möglichen Erweiterungen, bzw. Anpassungen gehören:

- Ausdruck der Typenschilder direkt über an den Server angebundene Drucker
- Ausdruck aus der Produktemaske
- Weitere Optimierung des Geschäftsprozesses

Da nun die Datenstruktur sehr generisch gehalten ist und die Informationen per WebAPI verfügbar sind, könnte man sich auch überlegen, eine zweite Webapplikation für Kunden zu erstellen. Über diese Schnittstelle könnten Kunden dann Informationen (Handbücher, Garantieansprüche, Zusatzartikel) über die von Ihnen gekauften Produkte einholen. Es wäre sicherlich empfehlenswert, diese Kundenapplikation zu kapseln und die WebAPI oder direkt den Businesslayer als Schnittstelle zu den Daten zu nutzen.

8. Appendix

8.1. Abkürzungsverzeichnis

Abkürzung	Bedeutung
(G)UI	(Graphical) User Interface
AD	Active Directory
AI	Application Identifier
BL	Business Layer
CAL	Client Access License
CL	Common Layer
CMD	Command (Line)
CO	Contract of Operation
CR	Cristal Reports
CRUD	Create, Read, Update, Delete
CSV	Comma-Separated-Value
DAL	Domain Abstraction Layer
DI	Dependency Injection
DOM	Document Object Model
DRY	Don't Repeat Yourself
EAB	Elektronische Auftragsbearbeitung
EF	Entitiy Framework
ERP	Enterprise Resource Planning
ETL	Extract – Transform – Load
IIS	Internet Information Service
JSON	JavaScript Object Notation
MVVM	Model – View – ViewModel
NTLM	NT Lan Manager
ORM	Object Relational Mapping (OR-Mapper)
OU	Organisation Unit
POS	Point-Of-Sale / Point-Of-Service
REST	Representational State Transfer
SProc	Stored Procedure
SQL	Structured Query Language
SSL	Secure Socket Layer
SSRS	SQL Server Reporting Services
TLS	Transport Layer Security
UDF	User Defined Function

Tabelle 37: Abkürzungsverzeichnis

8.2. Abbildungsverzeichnis

8.3. Tabellenverzeichnis

8.4. Literaturverzeichnis

[1] Microsoft, „Configure Windows Authentication (IIS 7),“ 2009. [Online]. Available: http://technet.microsoft.com/en-us/library/cc754628(v=ws.10).aspx.

[2] RUOSS-KISTLER, „RUOSS-KISTLER AG,“ [Online]. Available: http://www.ruoss-kistler.ch/frameload.htm?http://www.ruoss-kistler.ch/handel/hilfe/barcode_typen.htm.

[3] Microsoft, „Walkthrough: Using Resources for Localization with ASP.NET,“ 30 03 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/fw69ke6f(v=vs.100).aspx.

[4] Microsoft, „How to deploy the resx files in VS2008,“ 17 05 2010. [Online]. Available: http://forums.asp.net/t/1558861.aspx/1.

[5] P. Dyda, „Stackoverflow: How to use Resources.resx dynamically i,e add new items dynamically,“ 16 04 2011. [Online]. Available: http://stackoverflow.com/questions/5686370/how-to-use-resources-resx-dynamically-i-e-add-new-items-dynamically.

[6] Google, „AngularJS - Superheroic JavaScript MVW Framework,“ Google, 2012. [Online]. Available: http://angularjs.org.

[7] Angular-UI team, „UI Bootstrap,“ 2013. [Online]. Available: http://angular-ui.github.io/bootstrap/.

[8] F. Copes, „Deferreds and Promises in JavaScript,“ 20 12 2012. [Online]. Available: http://flaviocopes.com/deferred-and-promises-in-javascript/.

[9] P. K. Project. [Online]. Available: http://kettle.pentaho.com/. [Zugriff am 10 06 2013].

[10] "My Digital Life," 03 06 2013. [Online]. Available: http://www.mydigitallife.info/how-to-create-hidden-user-account-hide-user-account-from-welcome-screen-in-windows/.

[11] R. Mühsig, „Code Inside,“ 29 04 2010. [Online]. Available: http://code-inside.de/blog/2010/04/29/howto-net-4-0-asp-net-mvc-on-iis-7-5-pagehandlerfactory-integrated-has-a-bad-module-managedpipelinehandler/. [Zugriff am 03 06 2013].

[12] xmorera, „Stackoverflow,“ 12 05 2011. [Online]. Available: http://stackoverflow.com/questions/5985153/asp-net-impersonation-problem.

[13] Wikipedia, „Form-based authentication,“ 11 10 2012. [Online]. Available: http://en.wikipedia.org/wiki/Form-based_authentication.

[14] Wikipedia, „Basic access authentication,“ 19 02 2013. [Online]. Available: http://en.wikipedia.org/wiki/Basic_access_authentication.

[15] R. Strahl, „Understanding ASP.NET Impersonation Security,“ 18 05 2005. [Online]. Available: http://www.west-wind.com/weblog/posts/2005/May/18/Understanding-ASPNET-Impersonation-Security.

[16] „Visualstudio Magazine - CR vs. SSRS," 05 03 2013. [Online]. Available: http://visualstudiomagazine.com/articles/2010/12/14/crystal-reports-vs-ssrs.aspx.

[17] "Codeproject Crystal Report," 05 03 2013. [Online]. Available: http://www.codeproject.com/Articles/166291/Generate-a-report-using-Crystal-Reports-in-Visual.

[18] „Codeproject - Localization in SSRS," 05 03 2013. [Online]. Available: http://www.codeproject.com/Articles/294636/Localizing-SQL-Server-Reporting-Services-Reports.

[19] „Bryans Blog - CR export to Word," 05 03 2013. [Online]. Available: http://www.fryan0911.com/2009/05/aspnet-how-to-export-crystal-report-to.html.

[20] „ASP.NET Forum - Localization in CR," 05 03 2013. [Online]. Available: http://forums.asp.net/t/1531037.aspx/1.

[21] „ASP.NET Forum - CR vs. SSRS," 05 03 2013. [Online]. Available: http://forums.asp.net/t/720164.aspx/1.

[22] Wendy, „Passbook Ready," 14 8 2012. [Online]. Available: http://passbookready.com/qrvspdf417/.

[23] G. Switzerland, „GS1," [Online]. Available: http://www.gs1.ch/de/leistungsbereiche/identification-communication/standardisation/GS1-System/barcode-identification/datentraeger/050-datamatrix.php.

[24] M. C. Services, „Monroe Consulting Services," [Online]. Available: http://www.monroecs.com/posfordotnet/opos_dotnet.htm.

[25] e. oreilly, „qrworld," 26 September 2001. [Online]. Available: http://qrworld.wordpress.com/2011/09/26/qr-codes-versus-data-matrix/.

[26] jkealey, „Lava Blast Software Blog," 6 6 2011. [Online]. Available: http://blog.lavablast.com/post/2011/06/06/Using-Microsoft-POS-for-NET-in-2011.aspx.

[27] Datalogic, „Datalogic," [Online]. Available: http://www.datalogic.com/deu/produkte/automatic-data-capture/mehrzweck-handscanner/gryphon-i-gd4400-2d-pd-174.html.

[28] Datalogic, „Datalogic," [Online]. Available: http://www.datalogic.com/deu/produkte/automatic-data-capture/industrielle-handscanner/powerscan-pd8500-pd-187.html.

[29] crazycatman, „HubPages," [Online]. Available: http://crazycatman.hubpages.com/hub/QR-Code-vs-Barcode.

[30] Google, „AngularJS - API Reference," Google, 2012. [Online]. Available: http://docs.angularjs.org/api/.

[31] Wikipedia, „Halstead-Metrik," 1 04 2013. [Online]. Available: http://de.wikipedia.org/wiki/Halstead-Metrik.

[32] A. J. Kattan, „Form authentication and authorization in ASP.NET," 21 04 2006. [Online]. Available:

http://www.codeproject.com/Articles/13872/Form-authentication-and-authorization-in-ASP-NET.