HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

HILDING ANDERS

Produktionsoptimierungen bei BICO

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2016

Autor(en):	Tobias Schmitz
	Philipp Walder
Betreuer:	Dr. Daniel Keller
Projektpartner:	Hilding Anders Switzerland AG, Schänis

1 Inhalt

2 Aufgabenstellung 7
3 Abstract 8
4 Management Summary 9
4.1 Ausgangslage 9
4.2 Vorgehen und Technologien 9
4.3 Ergebnis 9
5 Technischer Bericht 10
5.1 Einleitung und Übersicht 10
5.1.1 Eine wandelnde Aufgabendefinition 10
5.1.2 Abgrenzung zur Vorarbeit 10
5.2 Planung 11
5.2.1 Projektplanung 11
Wichtige Termine 11
Iterationen 12
5.2.2 UseCases 13
1 Fehlmaterial 13
2 Q-Issue 13
3 Update herunterladen 13
4 Optimierung 13
5 Web-APP 13
6 Rollenwechsel 13
Diagramm 14
5.2.3 Weitere Funktionale Anforderungen 15
5.2.4 Nicht Funktionale Anforderungen 15
Erreichbarkeit und Plattformabhängigkeiten 15
Konfigurierbarkeit 15
Benutzbarkeit 15
5.2.5 Domainanalyse 16
Domainmodell 16
Domainklassenbeschreibungen 16
models.dao 16
AndroidMissingMaterial 16
Config 16
Pathoptimizer 17
PathCoreList 17

PathCorePallet 17
PathCore 18
PathSector 18
SectorPosition 18
SectorConnection 19
SectorPlan 19
SectorPlanPosition 19
5.3 Entwicklung 21
5.3.1 Übersicht 21
Verwendete Technologien 21
5.3.2 Implementation 22
Android 22
Rollenzuschnitt 23
Server-Infrastruktur 25
Optimierung 28
Datenanalyse 28
Das Lager 28
Zuteilung der Kerne an Ihre Lagerposition. 29
Kernlisten 30
Umsetzung 31
Path-Finder 31
Listenimport 31
Plan erstellen 31
Berechnung des Fahrweges 33
CSV für Plan erstellen 34
Neue Liste generieren 34
resetLists 35
resetPlan 35
5.3.3 Schnittstellenbeschreibung 36
5.3.4 Qualitätssicherung 39
Code Analyse 41
Integrationtests Protokoll 45
Android 45
Rollenzuschnitt 46
Import 47
Listen löschen 47

Konfiguration 48
Optimierung 48
Usabilitytest Protokoll 49
Fehlmaterial 49
Q-Issue 50
Rollenzuschnitt 50
Protokoll Peerreviews 51
5.4 Deployment 52
5.5 Schlussfolgerung und Ausblick 53
5.5.1 Empfehlungen an BICO 53
5.6 Glossar 55
5.7 Literaturverzeichnis 56
5.8 Abbildungsverzeichnis 57
5.9 Tabellenverzeichnis 58
A Anhang 59
A.1 Projektdokumente 59
A.1.1 Anleitungen 59
Android 59
Installationsanleitung 59
Funktionsbeschreibung 59
Qualitäts-Mangel 59
Fehlmaterial 59
Kernlisten 59
Web-Applikation 60
Materialanforderung 60
Abmelden 60
Update App 60
Thin App 60
Einstellungen 60
Funktionen Web-App 60
Produktionslisten 60
Kernlisten 60
Handmusterlisten 60
Reissverschlusslisten 60
Einzugslisten 60
Gesamtlisten 61

Management 61
Dateiimport 61
Excel importieren 61
Neue Liste generieren 61
Excel hochladen 61
Bilder und Kerndaten laden 61
Zuschnittlager hochladen 61
Administration Listen 61
Administration Kerne 61
Administration RV 61
Priorisierung 61
Produktübersicht 61
Auditlog 61
KPI 62
Materialanforderungen 62
Fehlmaterial 62
Zuschnitt 62
Admin 62
Benutzerübersicht 62
Konfiguration 62
Logs 63
Optimierung 63
Download App 63
Optimierung 63
Installation 66
Was wird benötigt: 66
Aufsetzten MySQL Datenbank 67
Installationsschritte: 67
Datenbank anpassen 67
Services starten 68
Funktion überprüfen 70
A.1.2 Wandel der Aufgabendefinition 72
Ausgeschriebene Aufgabe 72
Version 1 73
Version 2 73
Version 3 74

Version 4 75
Version 5 77
Version 6 79
Version 80
A.1.3 Wireframes 82
WebApp 82
Android App 83
A.1.4 SSDs 85
Fehlmaterial 85
Q-Issue 86
Optimierung 86
A.1.5 Contracts 87
sendMissingMaterial (UC1 Fehlmaterial) 87
email (UC2 Q-Issue) 87
runOptimizationFor (UC4 Optimierung) 87

Studienarbeit im Herbst-Semester 2016
Produktions-Optimierungen bei BICO

Industriepartner

HildingAnders Switzerland AG
Biltnerstrasse 42
CH-8718 Schänis
hildinganders.com bico.ch

Ausgangslage

Bei Hilding Anders in Schänis SG werden die in der Schweiz gut bekannten Bico Matratzen hergestellt. Etwa 150 Mitarbeitende produzieren bis zu 800 Matratzen pro Tag, jede einzelne auf Mass gemacht. Logistische nicht ganz einfach: die Mass-Schaumstoffkerne müssen in der Produktion mit den ausgewählten Polster- und Dekstoffen zusammengebracht werden. Dabei wird auch ein grosses Lager an Matratzenkernen, Stoffen und Reissverschlüssen verwaltet.
Die Produktionsplanung wird heute noch grossteils mittels Excel-Tabellen und ausgedruckten Listen gemacht; neuerdings unterstützt eine Web-Applikation die Produktionsüberwachung und eine Android App die Gabelstaplerfahrer.

Aufgabenstellung

1. Bestehende Android App in Produktionsplanung erweitern:
   - Einfaches Melden von Fehlmaterial
   - Navigationshilfe bzw. Wegoptimierung für Gabelstapler

2. Web-Applikation erweitern:
   - Modellierung der Matratzenkern-Lagerplätze
   - Aufzeigen von Optimierungsmöglichkeiten bei Lagerung und Routing der Gabelstaplerfahrer

3. Verbessern der Server-Infrastruktur
   - Server-Neustart optimieren
   - Bedienungsanleitungen für Installation und Wartung des Servers
   - div. Detailverbesserungen, z.B. Zugriffe auf Logs, App-Installation via APK

16.12.2016
Rapperswil, Datum


Unterschrift des HSR-Betreuers, Daniel Keller

3 Abstract

Diese Studienarbeit umfasst folgende Technologien und Gebiete:

Da es sich um eine Client-Server Applikation handelt, ist auf der Serverseite die Umsetzung mit Java (Play-Framework) anzutreffen. Die Clientseite beschreibt zwei Verschiedene Technologieansätze. Zum einen kann die Applikation von einem Web Frontend aus bedient werden, welches mit der Templating-Engine Twirl, HTML und CSS mit Bootstrap, und Javascript mit JQuery realisiert wurde, zum anderen stehen zwei Versionen einer Android-Applikation den Benutzern bereit.

Die Projektdurchführung wurde mit den RUP Iterationen geplant, die Implementation agile ausgeführt. Zur Hilfestellung wurde als Versionsverwaltungssystem GIT eingesetzt.

In dieser Arbeit wurden diverse Verbesserungen und Erweiterungen, sowohl an den Android-Applikationen, als auch an der Serverseite und dem Web Frontend umgesetzt. Wichtige neue Funktionalitäten sind hierbei die Möglichkeit Fehlmaterial zu erfassen, oder Qualitätsmängel einem Verantwortlichen zukommen zu lassen, sowie eine dynamisch filterbare und sortierbare Listenansicht. Diese Filtermöglichkeiten umfassen zum Beispiel Bezugs- oder Liniennummer.

Des Weiteren wurden diverse Wartungs- und Verbesserungsarbeiten an der bestehenden Infrastruktur der Applikation durchgeführt. Ein weiterer Teil dieser Arbeit umfasst die Analyse und das Aufzeigen von Optimierungspotential im Lagerverwaltungssystem des Auftraggebers.

Im Bereich Optimierung wurde zuerst der Dijkstra-Algorithmus zum finden des Shortest-Path implementiert. Hiernach werden Vorschläge zur Optimierung generiert. Diese umfassen einerseits das Neuzuteilen der Matratzenkerne an einen neuen Ort und andererseits das Umsortieren der Reihenfolge der zu bearbeitenden Bestellungen.

4 Management Summary

4.1 Ausgangslage

Bei Hilding Anders in Schänis SG werden die in der Schweiz gut bekannten Bico Matratzen hergestellt. Etwa 150 Mitarbeiter produzieren bis zu 800 Matratzen pro Tag, jede einzelne auf Mass gemacht. Logistisch nicht ganz einfach: die Mass-Schaumstoffkerne müssen in der Produktion mit den ausgewählten Polster- und Deckstoffen zusammengebracht werden. Dabei wird auch ein grosses Lager an Matratzenkernen, Stoffen und Reissverschlüssen verwaltet.

Die Produktionsplanung wird heute noch grossteils mittels Excel-Tabellen und ausgedruckten Listen gemacht; neuerdings unterstützt eine Web-Applikation die Produktionsüberwachung und eine Android App die Gabelstaplerfahrer.

4.2 Vorgehen und Technologien

Unsere Vorgänger Philippe Naegeli und Raffael Ioannone haben mit der Web-Applikation und der App für Gabelstaplerfahrer eine gute Grundlage geschaffen, auf der wir aufbauen konnten.

Zuerst mussten wir jedoch den Server wieder zum Laufen bringen, da er vor dem Sommer abstürzte und nicht wieder gestartet wurde. Als dieser wieder online war, konnten wir, nach diversen Gesprächen mit dem technischen Verantwortlichen, einem Gabelstaplerfahrer und Mitarbeiter beim Nähen, Vorschläge für Verbesserungen machen. Aus vielen Ideen wurden konkrete und umsetzbare Pakete, die vorwiegend Verbesserung und Automatisierung von bestehenden Funktionen und vier neue Funktionen enthielten.

Die Android App sollte die Möglichkeit bieten, fehlendes oder fehlerhaftes Material auf die Web-Applikation zu melden, diese Funktionen wurden später auch noch in ein Smartphone App genommen um jeglichen Mitarbeitern das Melden zu ermöglichen.

Die Web-Applikation sollte eine Grundlage für spätere Optimierungen am Lagersystem (Lageroptimierung oder Wegoptimierung) schaffen und bestehende Funktionen verbessern oder automatisieren.

4.3 Ergebnis

Die Web-Applikation kann nun die von der Administration generierten Listen automatisch auf den Server laden und ein Mitarbeiter hat die Möglichkeit alte Listen zu Löschen. Im Bereich der Optimierung kann man eine oder mehrere Palettenlisten, wie sie ein Gabelstaplerfahrer verwendet hochladen und eine Lageranordnung aussuchen, dann wird einem die Anzahl durchfahrenen Sektoren angegeben. Diese Werte können nun mit verschiedenen Lageranordnungen verglichen werden um zu sehen ob Optimierungspotential vorhanden ist.

Auf der Android Seite haben wir die gewünschten Funktionen mit einem Barcodereader und Fotos umgesetzt. So kann man nun bei defekten Materialien Fotos aufnehmen und diese direkt per Email an den Verantwortlichen senden. Fehlende Kerne können über den Barcode identifiziert werden und es wird ein Eintrag auf dem Server erstellt.

5 Technischer Bericht

5.1 Einleitung und Übersicht

5.1.1 Eine wandelnde Aufgabendefinition

Am Anfang standen grobe Ideen seitens der Hilding Anders, was sie sich für die Zukunft wünschten und ein paar kleinere Fixes. Im Gespräch waren Navigationshilfe für Gabelstaplerfahrer, Wegoptimierung im bestehenden Lager, Optimierung am Lager und Zusatzfunktionen für die Android App.

Die Navigationshilfe sollte auf dem Tablet anzeigen, welches der kürzeste Weg zum nächsten Matratzenkern ist. Es wäre jedoch schwierig umzusetzen gewesen, da man im Lager keinen GPS empfang hat und die W-Lan-Abdeckung nicht genügend ist.

Bei der Optimierung haben wir uns nach einem Gespräch mit einem Professor der HSR, welcher sich mit Simulationen bestens auskennt[1], mit der Firma auf eine Shortest-Path Analyse geeinigt. Des Weiteren galt es herauszufinden, welches Potential in einer optimaleren Verteilung der Kerne im Lager liegt.

Zusätzlich kam das Bedürfnis einer Optimierung beim Zuschnitt der Stoffe auf. Dort haben wir wirkliches Potential gesehen, da man mit einfacher Voraussicht viel Arbeit einsparen kann. Somit wurde dieses Thema ein bedeutender Teil unserer Arbeit, weil auch die Mitarbeiter begeistert waren und uns immer wieder neuen Input gaben, wie wir ihnen das Leben noch etwas vereinfachen können.

Ebenfalls dazu kam eine abgespeckte Version der bereits vorhanden Android App mit den Funktionen, die wir an der bestehenden App ergänzen sollten. Jetzt kann ein Lagermitarbeiter mit seinem Tablet fehlendes oder defektes Material einscannen und die Meldung direkt an den Verantwortlichen weiterleiten. Zusätzlich können andere Mitarbeiter sich die mobile Version der App auf das Smartphone laden und diese Funktionen ebenfalls nutzen.

5.1.2 Abgrenzung zur Vorarbeit

Im Frühlingssemester 2015 wurde in Form einer Bachelorarbeit eine Machbarkeitsstudie und einen User-Acceptancetest einer Android App durchgeführt, um zu evaluieren, ob eine Ablösung der Excel-Listen mit einem Tablet möglich ist.

Im Herbstsemester 2015 wurde in einer Studienarbeit ein Prototyp der Android App erstellt, welcher bereits ein Algorithmus zur Gruppierung der Kerne implementiert und die Kernliste mit Bildern und Sektor Informationen anzeigt.

In einer Bachelorarbeit im Frühlingssemester 2016 wurde eine Android-Applikation für Gabelstaplerfahrer und eine dazugehörige Webapplikation für stationäre Arbeitsplätze erstellt. Mit der Mobile-App ist es möglich, Listen von Kernen anzusehen und abzuarbeiten. Die Webapplikation dient dazu, die Daten auf unterschiedliche Weisen importieren zu können. Zudem wird den Büroangestellten eine Ansicht auf die weiteren Listen angeboten.

[1] Gespräch mit Prof. Dr. Andreas Rinkel am 05.10.16

5.2 Planung

5.2.1 Projektplanung

Nach einem verzögerten Start aufgrund administrativer Komplikationen begannen wir unsere Arbeit mit ca. 10 Tagen Verspätung. Sehr früh teilten wir unsere Arbeit in die vier RUP-Phase Inception, Elaboration, Construction und Transition und versuchten direkt bei Hilding Anders vorbei zu gehen um persönlichen Kontakt herzustellen und das Problem zu begutachten.

Hiervon leitet sich Folgende Projektplanung ab.

Wichtige Termine

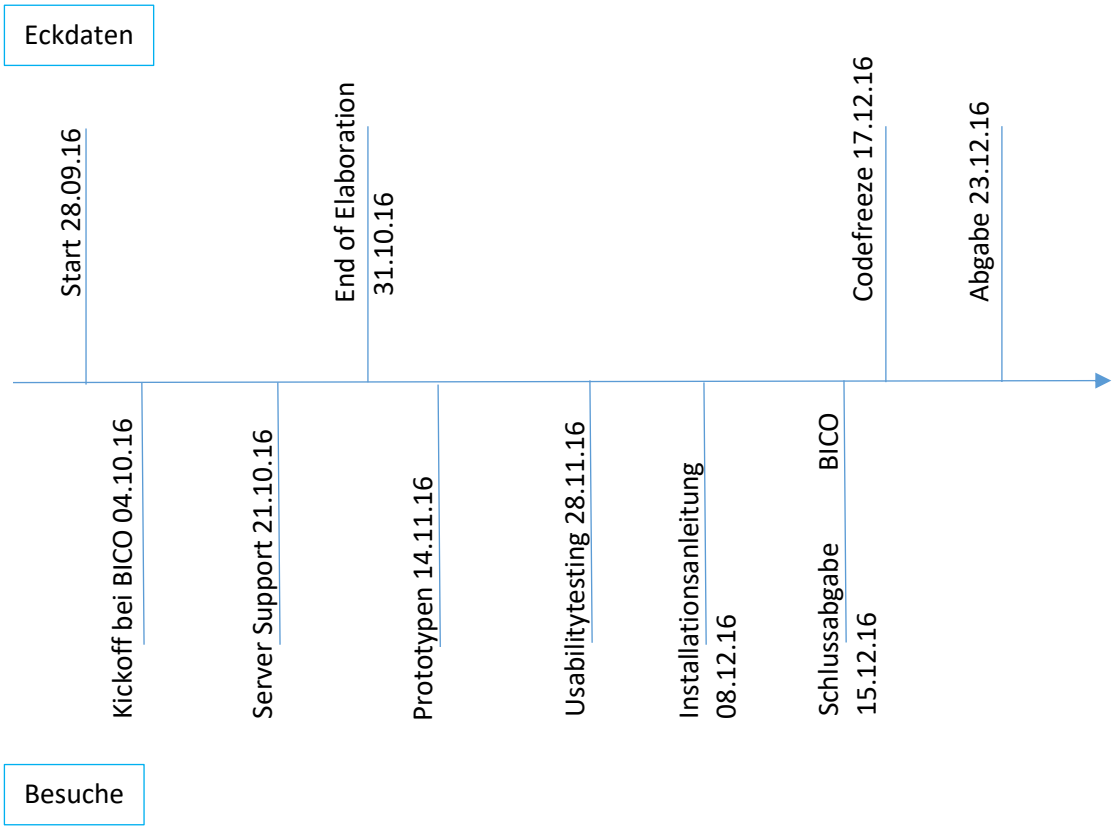


Abbildung 1 Termine der Planung

Tabelle 1 Termine der Planung

28.09.2016	Start der Arbeit
04.10.2016	Kickoff des Projektes bei der BICO
21.10.2016	Support mit dem Server
31.10.2016	End of Elaboration
14.11.2016	Präsentieren der ersten Prototypen (Android App) und Server Setup
28.11.2016	Usabilitytests
08.12.2016	Installationsanleitung durchgehen und letzten Verbesserungen definieren
15.12.2016	Schlussabgabe bei BICO (Präsentation IST Zustand)
17.12.2016	Codefreeze
23.12.2016	Projektabgabe an Betreuer und BICO

Iterationen

Tabelle 2 Iterationsplanung

Iteration	Start	Ende	Bemerkungen
Inception	28.09.2016	03.10.2016	Einarbeiten in alte Arbeit
E1	04.10.2016	17.10.2016	Aufgaben Analyse
E2	18.10.2016	31.10.2016	Planung und Prototypen
C1	01.11.2016	14.11.2016	Entwicklung
C2	15.11.2016	28.11.2016	Entwicklung und Usabilitytests
C3	29.11.2016	12.12.2016	Entwicklung
T1	13.12.2016	23.12.2016	Schlussabgabe und Dokumentation fertigstellen

5.2.2 UseCases

1 Fehlmaterial

UC1: Ein Mitarbeiter sieht, dass an einem Lagerplatz kein Kern mehr vorhanden ist oder das Kanban-Minimum wurde erreicht. Mit der Android App scannt er den Barcode (Barcode an Säule auch wenn kein Material mehr vorhanden ist), wählt ob es eine Kanban-Nachbestellung oder Fehlmaterial ist und gibt seinen Namen ein. Diese Daten kann er absenden und sie werden in der Web-App unter Fehlmaterial angezeigt.

UC1.1: Ein Gabelstaplerfahrer kann angeben, wie viele Kerne ihm fehlen um seinen aktuellen Auftrag zu beenden.

UC1.2: Ein Gabelstaplerfahrer kann einen Termin eingeben, bis wann er eine Nachlieferung braucht.

UC1.3: Ein Mitarbeiter hat die Möglichkeit einen Kommentar zur Meldung zu schreiben für zusätzliche Informationen.

2 Q-Issue

UC2: Ein Mitarbeiter sieht defektes Material und kann mit der Android-App eine Meldung dazu erstellen. Damit möglichst gut auf das Problem reagiert werden kann, macht der Mitarbeiter ein Foto des Defekts, scannt den Barcode des Artikels und schreibt einen Kommentar dazu. Die Meldung geht per Email direkt an eine Mailbox für Materialfehler.

3 Update herunterladen

Ein Nutzer der Android-Applikation kann sich direkt aus dem App die neuste Version vom Server herunterladen.

4 Optimierung

UC4.1: Ein Mitarbeiter mit Administrator-Berechtigungen kann in der Optimierung Tageslisten hochladen und mit dem Standardalgorithmus sehen, wie viele Sektoren ein Gabelstapler abgefahren hat, um alle Paletten abzuarbeiten.

UC4.2: Er kann sich mit Verbrauchslisten (welche Kerne wurden wie oft verkauft) ein optimiertes Lager zusammenstellen lassen. Dann kann er die hochgeladenen Tageslisten im optimierten Lager simulieren und so vergleichen, ob ein Verbesserungspotential vorhanden ist.

Er kann jegliche bestehenden Lagerpläne hochladen und diese auch zur Simulation verwenden.

5 Web-APP

UC5.1: Ein Administrator kann die Logs des Servers einfach einsehen, um im Fehlerfall Hintergrundinformationen zu erhalten.

UC5.2: Für den Rollenwechsel kann man eine aktualisierte Version des Lagers hochladen

UC5.3: Um das Überfüllen des Servers zu verhindern, kann ein Mitarbeiter Fehlmaterial-Meldungen wieder löschen.

6 Rollenwechsel

UC6: Die Mitarbeiter beim Stoffzuschnitt haben die Möglichkeit in der Web-App verschiedene Sortierungen der aktuellen Aufträge anzusehen. So können sie nicht nur die Kernliste der Paletten, die gerade zusammengestellt werden, anschauen, sondern auch welche Stoffe an diesem Tag noch folgen werden. So können sie vorausschauen und evtl. schon Zuschnitte vorzeihen um überflüssige Rollenwechsel zu vermeiden.

Diagramm

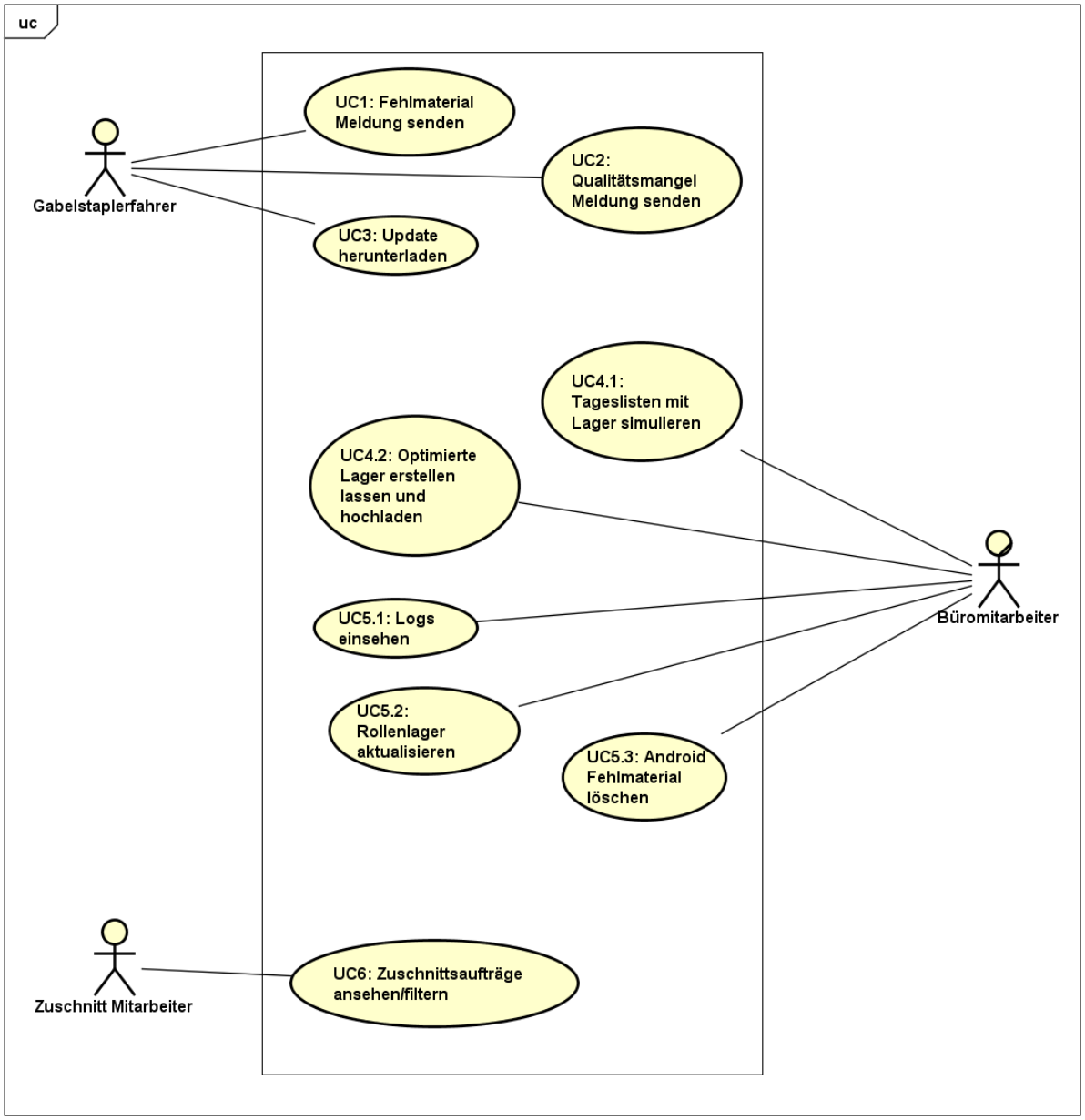


Abbildung 2 UseCase Diagramm

5.2.3 Weitere Funktionale Anforderungen

- Die Daten des Q-Issues werden bei bestehender Verbindung zum Internet als E-Mail an die Konfigurierte Adresse gesendet. Und so für weitere Verwendung zugänglich gemacht.
- Die Daten des Fehlmaterials werden, bei bestehender Verbindung zum Server, gesendet und gespeichert. Die Daten sind danach Im Webinterface für berechtigte Benutzer einsehbar.
- Die Funktion zum Melden eines Qualitäts-Issues, als auch die Funktion zum Eintragen von Fehlendem Material werden in einer Android Applikation als .apk Datei für die Mitarbeiter zur Verfügung gestellt, und können falls nötig auch aktualisiert werden.

5.2.4 Nicht Funktionale Anforderungen

Erreichbarkeit und Plattformabhängigkeiten

- Sowohl die Funktion zum Melden eines Qualitäts-Issues, als auch die Funktion zum Eintragen von Fehlendem Material soll für jeden Mitarbeiter in Schänis vor Ort Auf Android Geräten ab Version 4.2 jederzeit möglich sein.
- Für die Mitarbeiter, welche für den Rollenwechsel und das dortige Zuschneiden verantwortlich sind, soll es zur Arbeitszeit möglich sein, die Liste der benötigten Rollen, über einen aktuellen Webbrowser, sortiert und gruppiert nach Bedarf anzuzeigen.

Konfigurierbarkeit

Einzelne Einstellungen für die Serververwaltung und den Import/Export von Daten ins System sollen von befugten Benutzern im Webinterface gesetzt und verändert werden können. Hierzu zählen:

- Mail-Adresse für den Versand der Q-Issues
- Versionen der Android Applikation
- SharedFolder für die Logfiles der Applikation
- SharedFolder für den automatischen Import von Excel-Listen
- Automatisches Importieren AN/AUS
- Automatisches Freischalten AN/AUS

Benutzbarkeit

- Falsche Benutzereingaben sollen dem Benutzer zurückgemeldet werden.

5.2.5 Domainanalyse

Domainmodell

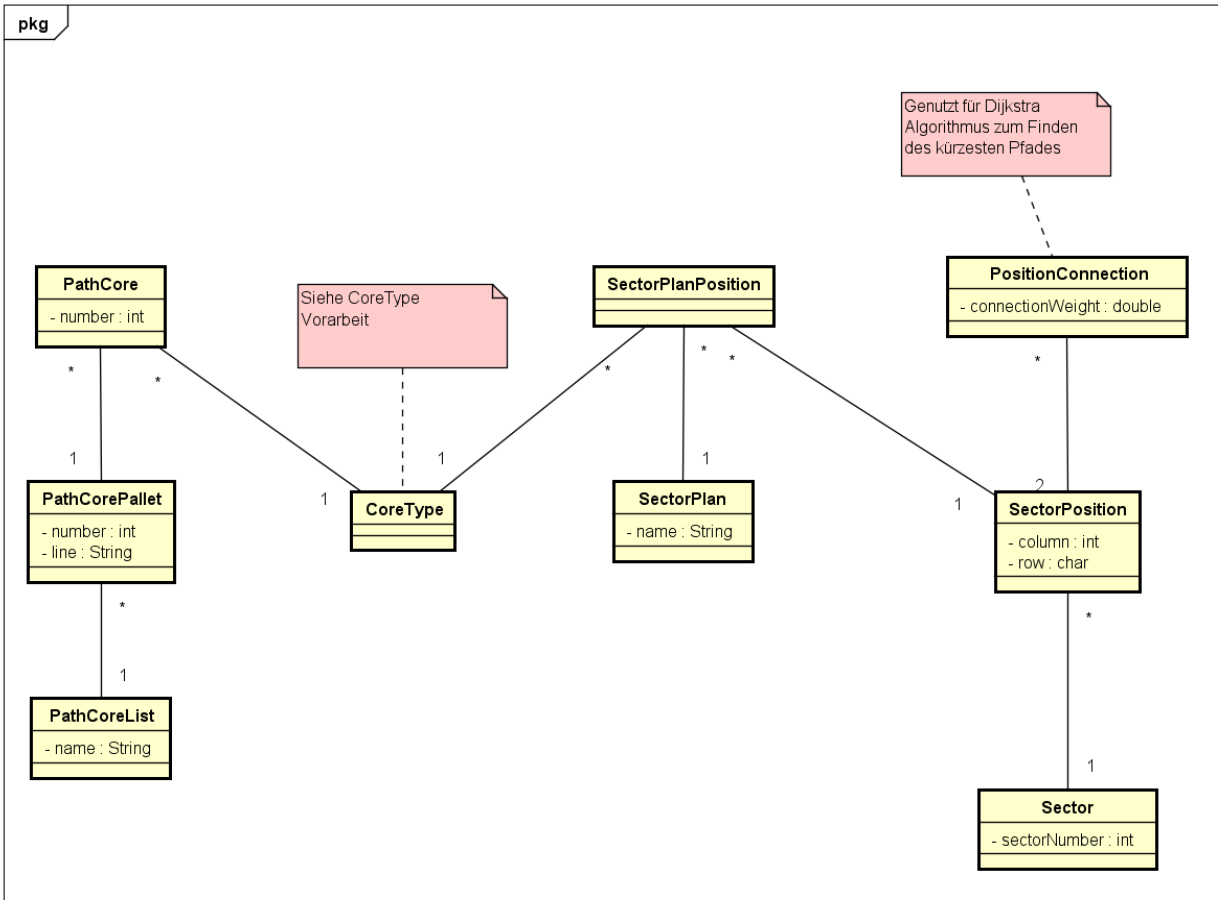


Abbildung 3 Domainmodell

Domainklassenbeschreibungen

models.dao

AndroidMissingMaterial

Diese Klasse speichert das Fehlmaterial, welches von der Android App aufgenommen und geschickt werden kann.

Tabelle 3 Properties AndroidMissingMaterial

Property	Beschreibung
missingMaterialType	Beschreibt, ob das Fehlmaterial eine einfache Fehlmaterialposition, oder eine Kanban Nachbestellung ist.
barcode	Der Barcode, welcher gescannt wurde, oder eingegeben wurde.
issuer	Benutzername des Verfassers.
amount	Wieviel nachbestellt werden soll.
date	Datum an welchem das Material benötigt wird
comment	Kommentar des Verfassers

Config

Diese Klasse speichert die dynamischen Konfigurationsdaten, welche bei laufender Applikation veränderbar sein sollen.

Tabelle 4 Properties Config

Property	Beschreibung
qIssueEmail	E-Mail-Adresse welche vom Android App genutzt wird um die Emails für die Q-Issues zu generieren
pathUploadExcel	Ordnerpfad welcher unter Management → Datenimport im Tab Excel importieren verwendet wird. Dieses Directory wird nach allen xls und xlsx Dateien durchsucht. Diese können auf Knopfdruck Importiert werden.
tempUploadPath	Ein Ordnerpfad welcher gebraucht wird um Dateien zwischenzuspeichern.
pathImportPictures	Der Pfad aus welchem die Kernbilder geladen werden.
pathImportCoretypes	Der Pfad aus welchem die Kerndaten geladen werden. Dieser Pfad muss auf eine .csv Datei zeigen
autoImportExcelFolder	Der Pfad auswelchem der Import Task die Dateien periodisch Importiert.
autoImportExcel	Zeigt an, ob der Import Task die Dateien periodisch importieren soll.
autoActivateLists	Zeigt an, ob die Importierten Listen automatisch freigegeben werden sollen.
androidAppFolder	Der Pfad aus welchem die Android APKs geladen werden sollen. Für den Download.
androidAppFile	Die Android APK für die Tablet Version.
androidThinFile	Die Android APK für die Mobile Version

Pathoptimizer

PathCoreList

Die Klasse Repräsentiert eine Excel-Liste. Die PathCoreList wird verwendet um die alten Excel Listen zu repräsentieren, um die Länge des Fahrtweges zu berechnen welcher gebraucht wurde.

Tabelle 5 Dependencies PathCoreList

Abhängigkeit zu	Beschreibung
PathCorePallet	Eine PathCoreList beinhaltet mehrere PathCorePallets. Dies sind die einzelnen Paletten, welche auf der Liste waren

Tabelle 6 Properties PathCoreList

Property	Beschreibung
name	Der Dateiname der alten Excelliste

PathCorePallet

Die Klasse repräsentiert eine Palette aus der alten Excel Liste. Eine Palette wird vom PathOptimizer jeweils wie Folgt abgefahren: Gestartet wird jede Palette jeweils neu im Liftsektor. Dann wird mittels einer Implementation des Dijkstra-Algorithmus der kürzeste Pfad zur ersten Kernposition bestimmt. Dieser Pfad wird zum Aktuellen gesamt Pfad addiert. Von der neuen Position aus wird nun für jede weitere Kernposition dasselbe gemacht. Am Ende der Palette wird nochmals der kürzeste Pfad von der letzten Position aus zum Liftsektor zurück berechnet und addiert.

Tabelle 7 Dependencies PathCorePallet

Abhängigkeit zu	Beschreibung
PathCoreList	Ein PathCorePallet gehört zu einer PathCoreList.
PathCore	Ein PathCorePallet hat einen oder mehrere PathCores. Diese repräsentieren die Kernpositionen in der Palette.

Tabelle 8 Properties PathCorePallet

Property	Beschreibung
line	Die Linie zu welcher die Kerne auf der Palette gehören
palletNumber	Die Palettennummer.

PathCore

Die Klasse repräsentiert eine Kernposition der zugehörigen PathCorePallet.

Tabelle 9 Dependencies PathCore

Abhängigkeit zu	Beschreibung
PathCorePallet	Ein PathCore gehört zu einem PathCorePallet
CoreType	Der Kerntyp, welcher auf dieser Liste repräsentiert wird

Tabelle 10 Properties PathCore

Property	Beschreibung
number	Die Nummer der Position auf der Liste.

PathSector

Die Klasse repräsentiert einen Sektor im Lager.

Tabelle 11 Dependencies PathSector

Abhängigkeit zu	Beschreibung
SectorPosition	Ein PathSector hat mehrere SectorPositions.

Tabelle 12 Properties PathSector

Property	Beschreibung
sectorNumber	Die Sektor Nummer ist eine Zahl von 1 bis 5

SectorPosition

Ein Sektor ist eingeteilt in einem Schachbrett Muster. Die horizontalen Linien sind jeweils mit einem Buchstaben gekennzeichnet, die vertikalen Linien jeweils mit einer Zahl. Diese Felder sind SectorPositions. Jede SectorPosition repräsentiert ein eindeutiges Feld im Lagerplan.

Tabelle 13 Dependencies SectorPosition

Abhängigkeit zu	Beschreibung
PathSector	Eine SectorPosition gehört zu einem PathSector.
SectorConnection	Eine SectorPosition hat mehrere SectorConnections.

SectorPlanPosition	Eine SectorPlanPosition beschreibt den Standort eines Kernes im Lager. Eine SectorPosition kann mehrere SectorPlanPositions beinhalten.

Tabelle 14 Properties SectorPosition

Property	Beschreibung
col	Vertikale Koordinatenbezeichnung. Eine Zahl.
row	Horizontale Koordinatenbezeichnung. Ein Buchstabe.
planX	Absolute Horizontale Koordinatenbezeichnung im Lagerplan.
planY	Absolute Vertikale Koordinatenbezeichnung im Lagerplan.

SectorConnection

Die Klasse repräsentiert die möglichen Verbindungen zwischen mehreren SectorPositions. Sie wird vor allem für den Path-Finding Algorithmus gebraucht.

Tabelle 15 Dependencies SectorConnection

Abhängigkeit zu	Beschreibung
SectorPosition	Eine SectorConnection verbindet zwei SectorPositions

Tabelle 16 Properties SectorConnection

Property	Beschreibung
connectionWeight	Gewichtung für die Verbindung. wird benötigt, um die Hauptfahrtwege im Lager abzubilden

SectorPlan

Die Klasse repräsentiert den Lagerplan der gelagerten Kerne, sprich bestimmt die Positionen der Kerne im Lager. Da Vorschläge für ein optimiertes Lager generiert werden können, reicht ein einzelner Plan nicht aus.

Tabelle 17 Dependencies SectorPlan

Abhängigkeit zu	Beschreibung
SectorPlanPosition	Ein SectorPlan hat mehrere SectorPlanPositions

Tabelle 18 Properties SectorPlan

Property	Beschreibung
name	Name des Lagerplans

SectorPlanPosition

Verbindet einen Kern mit einem SectorPlan und der SectorPosition für diesen Plan.

Tabelle 19 Dependencies SectorPlanPosition

Abhängigkeit zu	Beschreibung
SectorPlan	Eine SectorPlanPosition gehört zu einem SectorPlan.
SectorPosition	Eine SectorPlanPosition weist einem Kern seine SectorPosition zu.
CoreType	Der Kern welcher platziert werden soll.

5.3 Entwicklung

5.3.1 Übersicht

Diese Arbeit wurde auf der Basis der Vorarbeit aufgebaut. Folgend ist der Aufbau der Arbeit nochmals graphisch dargestellt.

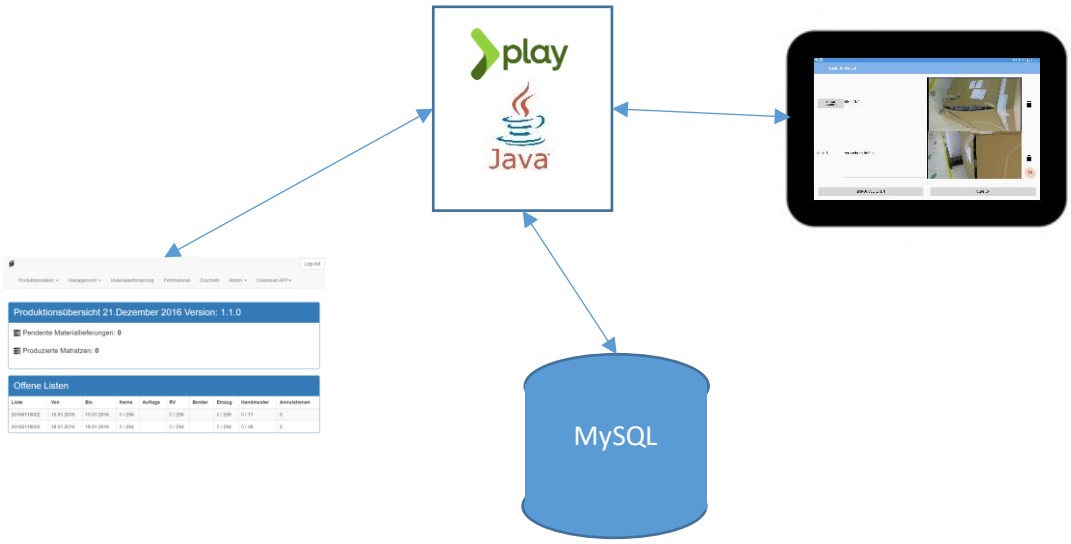


Abbildung 4 Systemübersicht

Verwendete Technologien

Tabelle 20 Verwendete Technologien

Verwendete Technologien	
Android	Java
	Android SDK
	Android Studio
	greenDAO OR-Mapper für SQLite (ausschliesslich in Vorarbeit)
	GSON: JSON Serialization
	Volley: Library für http Requests zum Server
Server	Java
	Java Play-Framework
	JPA Hibernate OR-Mapper für die MySQL Datenbank
	Logback Logger im Playframework
	Guice Dependency Injection
	GSON: JSON Serialization
	JUnit
	IntelliJ
Server - Webfrontend	Twirl Templating Engine (Scala)
	JavaScript
	JQuery
	HTML5, CSS, Bootstrap

5.3.2 Implementation

Android

In der Android App sollte es möglich sein, dass ein Mitarbeiter Meldungen über fehlendes oder fehlerhaftes Material an den verantwortlichen Büromitarbeiter melden kann. Da sich die Büromitarbeiter nicht sehr oft im Lager aufhalten können sie solche Probleme schwer alleine finden und sind auf die Hilfe der Gabelstaplerfahrer angewiesen, die pro Tag mehrere Kilometer durch das Lager machen.

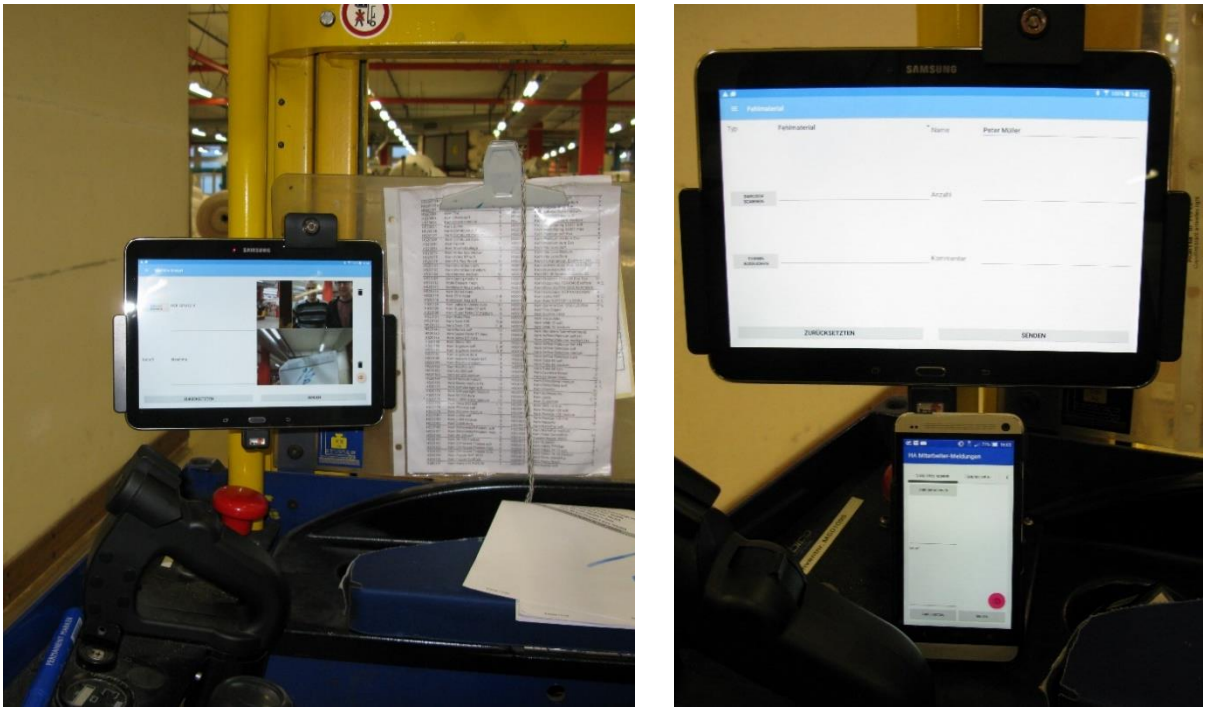


Abbildung 5 Android Applikation im Einsatz

Die Lagermitarbeiter haben nun eine Erweiterung für ihre Produktions-App erhalten und können mit dieser, allfällige Probleme schnell weiterleiten. Für fehlerhaftes Material gibt es Qualitäts-Mangel-Meldungen und für fehlendes oder fast aufgebrauchtes Material die Fehlmaterial-Meldungen.

Bei Qualitäts-Mängeln ist es wichtig, dass die Hilding Anders ihrem Lieferanten beweisen kann, dass der Defekt nicht ihre Schuld ist. Darum kann man bei Qualitätsmeldungen mehrere Fotos als Beweis mitsenden. Die Meldung wird als Email an den Qualitätsverantwortlichen gesendet, deshalb kann man bereits in der Applikation den Betreff der Email eingeben. Die Empfängeradresse wird automatisch vom Server geladen, somit muss der Mitarbeiter nicht wissen, an wen er diese Mail senden muss. Mit dem eingescannten Barcode kann man dem Büromitarbeiter die Arbeit noch etwas vereinfachen, da er dann schneller weiss, wen er kontaktieren muss. Das automatisch generierte Email kann natürlich auch noch von Hand ergänzt werden um zusätzliche Informationen anzufügen.

Beim Fehlmaterial wird eine Meldung erstellt, welche an den Server geschickt wird und die verantwortlichen Büromitarbeiter müssen diese selber kontrollieren, werden also nicht benachrichtigt. Die Minimalanforderungen an eine solche Meldung sind der Barcode, also was fehlt, der Name des Erstellers der Meldung, um welcher automatisch aus den Anmeldedaten gelesen wird, und der Typ der Meldung, also ob gar nichts mehr vorhanden ist oder ob das Kanban-Minimum erreicht wurde. Als weitere Informationen kann der Lagermitarbeiter einen Termin setzten, bis wann er sich Nachschub wünscht, eine Anzahl an Kernen die er brauchen wird, und einen Kommentar falls noch etwas nicht genannt werden konnte.

Diese beiden Funktionen der Tablet-App wurden auch noch in einer Smartphone App umgesetzt um allen Mitarbeitern der Hilding Anders die Möglichkeit zu geben, Fehler zu melden. Da diese App der Einfachheit halber ohne Login auskommt, hat sie noch eine Seite mit Einstellungen, in der man den Namen für die Fehlmaterialmeldung setzen kann und zur Überprüfung wird die Emailadresse für Qualitätsmeldungen angezeigt.

Implementation:

Mit dem Knopf zum Scannen des Barcodes wird ein neuer Intent eröffnet, welcher ein Barcodescann-App öffnet. Dieser übernimmt dann die Erkennung des Barcodes und gibt uns den Barcode als Text zurück. (Dies wird auch in der Fehlmaterial App so genutzt.) Für das Aufnehmen der Fotos benutzen wir die normale Kamera-App und speichern das Foto zuerst auf dem lokalen Speicher. Von dort nehmen wir es, komprimieren es und fügen es in eine spezielle Liste, damit wir mehrere Fotos in einer Scrollview darstellen können. Beim Senden werden die Daten aus dem Betreff und dem Barcode in das Email eingefügt und die Fotos, die sich momentan in der Scrollview befinden, werden dem Mail als Anlagen angehängt.

Die beiden Ansichten Qualitätsmangel und Fehlmaterial benutzten beide Shared Preferences um die gemachten Eingaben zwischen zu speichern und holen diese beim onReturn auch wieder. Will man die geladenen Daten nicht kann man diese zurücksetzten. Dabei werden die Shared Preferences und die Liste mit den aufgenommenen Fotos gelöscht, respektive auf null gesetzt. Beide Ansichten öffnen nach dem Abschliessen des Auftrags den Navigationdrawer, da wir davon ausgehen, dass wenn man ein Problem gesehen und gemeldet hat, man nicht gleich wieder eines sieht. Muss man allerdings mehrere Meldungen hintereinander aufgeben, ist man nur ein Klick davon entfernt.

Beim Fehlmaterial wird der Einfachheit halber ein MissingMaterialDTO, also ein Transfer Objekt erstellt. Alle gemachten Angaben werden in das DTO gepackt und mit einem http Post-Request an den Server gesendet. Falls dieser Post nicht richtig versendet oder empfangen werden kann gibt die Methode entsprechende Rückmeldungen. Bei erfolgreichem Versenden wird dies auch angezeigt. Beim Öffnen der View sucht diese automatisch in den Shared Preferences nach vorhandenen Daten, die nicht gesendet oder gelöscht wurden. Somit kann die App auch geschlossen werden ohne, dass Daten verloren gehen.

Die Smartphone App wurde durch eine Einstellungs-Ansicht ergänzt. Dort kann man seinen Namen angeben, um diesen bei Fehlmaterial-Meldungen nicht jedes Mal von Hand eingeben zu müssen. Da wird die Smartphone-App mit einer Tab View anstelle einem Navigationdrawer erstellt haben, da dieser zu viel Aufwand für die wenigen Funktionen gewesen wäre, werden die Seiten beim Wechsel nicht neu geladen. Deshalb ist es nun notwendig, dass man nachdem man den Namen eingetragen hat, diesen mit dem Speichern Knopf in die Shared Preferences speichern muss und dann in der Fehlmaterial Ansicht das Feld aktualisieren muss um den Namen wieder zu holen. Ist er jedoch einmal gesetzt sollte ihn sich die App merken. Ebenfalls angezeigt wird die E-Mail-Adresse, für Qualitätsmeldungen. Ist diese nicht verfügbar, hat man keine Verbindung zum Server und kann somit weder E-Mails noch Fehlmaterialmeldungen senden. Direkt übernommen wurden die Funktionen für eine Verbindung in die Webansicht und den Download der neusten Version der App. Selbstverständlich wird die mobile App heruntergeladen für das Update.

Rollenzuschnitt

Bei Rollenzuschnitt mussten die Arbeiterinnen mehrmals täglich die Rollen mit dem Deckstoff wechseln um entsprechend den Aufträgen zuschneiden zu können. Als Auftragsübersicht benutzten sie Excellisten, meistens waren das 3-4 Seiten je 40 Aufträgen. Dies ist sehr unübersichtlich und umständlich. In der Zukunft soll an diesem Platz ein PC stehen und die Mitarbeiterinnen sollen im Browser eine Übersicht über ihre Aufträge haben.

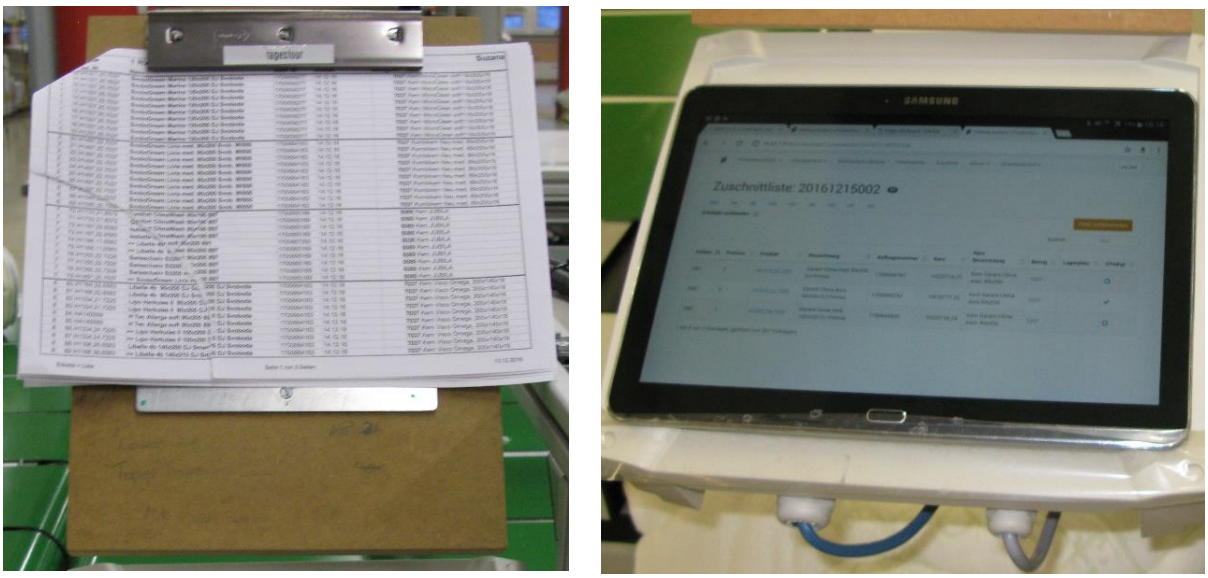


Abbildung 6 Rollenzuschnitt Vergleich

Nun kann man Auftragslinien einzeln darstellen um die Masse an Aufträgen etwas einzudämmen. Ebenfalls ist es möglich, direkt auf eine Bezugsnummer zu klicken um diese als Filterkriterium einzutragen. Dann werden nur noch Aufträge mit dieser Bezugsnummer angezeigt, so hat man die Möglichkeit, vorauszuschauen wie oft und wann ein Stoff noch gebraucht wird. Dasselbe ist mit der Produktnummer möglich, so können alle gleichen Produkte angezeigt werden, falls die Bezugsnummer alleine zuviele Aufträge anzeigt.

Erledigte Zuschnitte können abgehakt werden, was hilft einen Überblick über die getane Arbeit zu haben. Der Einfachheit halber können erledigte Positionen auch komplett ausgeblendet werden, damit nur noch offene Aufträge angezeigt werden.

Implemantation:

Wir haben diese Seite so umgesetzt, dass wir zur Genereierung der Liste die gleiche Methode verwenden, wie bei den Gesamtlisten, damit wir alle verfügbaren Informationen haben. Diese Liste wird dann einmal koplett durchlaufen um alle vorkommenden Linien zu finden, diese werden dann in der Linienliste der View übergeben. Weiter bekommt die Methode, welche am Ende die Seite rendert einen Filter, welche Linie angezeigt werden soll, anfangs werden immer alle Linien angezeigt. Wird auf der Seite ein Filter gewählt, wird sie neu geladen und im Hintergrund wird eine neue Liste erstellt, die nur Elemente mit dem gewählten Listenattribut enthalten. Diese neue Liste wird dann beim Rendern mitgegeben.

Das Filtern nach Bezugs- oder Produktnummer funktioniert so, dass ein Link auf diesen Nummern ein JS Befehl auslöst, welcher den Inhalt des Links in das Suchfeld der Tabelle schreibt. Mit dme Knopf zum Zurücksetzten des Filters wird einfach ein leerer String in das Suchfeld geschrieben.

Zum ausbleden der erledigten Positionen hat jede Position ein Attribut, das anzeigt ob sie schon erledigt wurde. Mit einem Klick kann der Zustand geändert werden, so kann auch etwas, dass als erledigt markiert wurde wieder auf nicht erledigt gesetzt werden. Bei setzten des Hakens zum Ausblenden der Erledigten, überprüft JS die ganze Tabelle und blendet alle Erledigten aus. Nimmt man den Haken wieder weg, werden alle wieder angezeigt.

Server-Infrastruktur

Bevor hier die Anforderungen, welche uns bei diesem Teil der Implementation gestellt wurde, behandelt werden, wird auf eine über viele dieser Erweiterungen greifende Funktionalität geschrieben. Die neue Konfiguration.

Hierfür wurde die schon im Kapitel Modelklassenbeschreibungen beschriebene Klasse Config erstellt, welche die Konfiguration zur Datenbank schreibt. Des Weiteren wurde in der Administrationsansicht eine Seite hinzugefügt wo diese Konfigurationsmöglichkeiten dem Benutzer zur Verfügung gestellt werden.

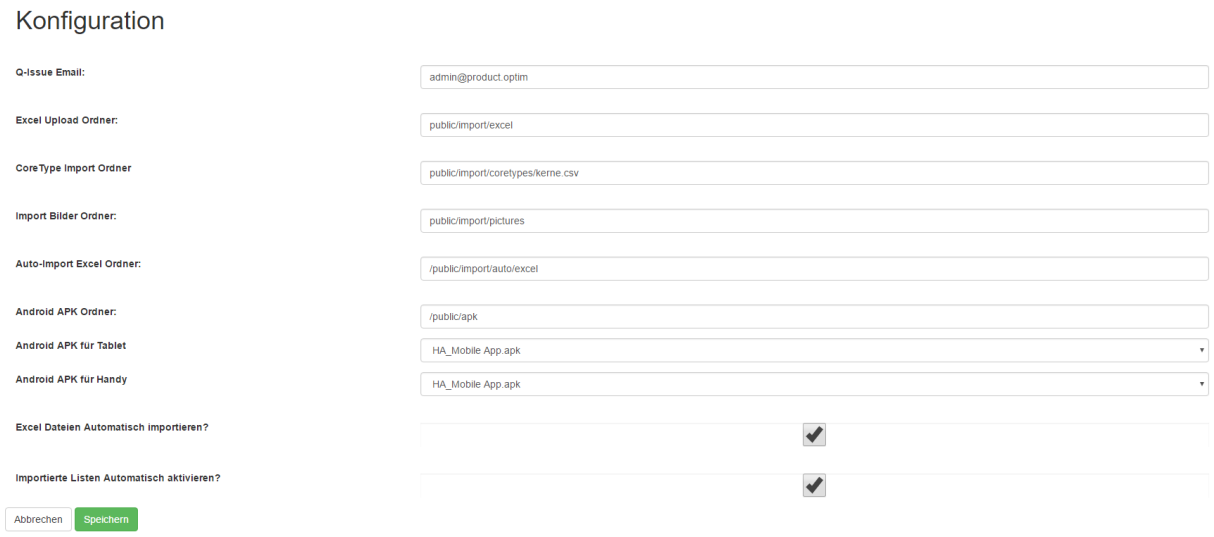


Abbildung 7 Konfigurationsansicht

Die Anforderungen an die Verbesserung der Server-Infrastruktur waren die Folgenden:

Es soll sowohl möglich sein .xls und .xlsx Dateien mit den jeweiligen Listen zu importieren. In der Vorarbeit waren bis jetzt nur .xls Dateien unterstützt worden.

Für diesen Task mussten bei der Library welche die Office-Dokumente liest lediglich jeweils eine andere Klasse eingesetzt werden (Package der Klassen von org.apache.poi.xssf.usermodel zu org.apache.poi.ss.usermodel):

- XSSFWorkbook wechseln zu Workbook
- XSSFSheet wechseln zu Sheet
- XSSFRow wechseln zu Row

Danach musste noch dafür gesorgt werden, dass alle Fileextension Filter beide Endungen .xls und .xlsx unterstützen.

Die importierten Listen sollen automatisch zur Benutzung in der Produktion freigegeben werden.

Für diese Funktion musste jeweils nach dem Importieren der Liste, jeweils noch die Liste auf aktiviert gestellt werden.

Es soll möglich sein, Listen automatisch aus einem konfigurierbaren Ordner oder Network-Share zu laden und zu importieren. Dies soll jeweils in einem Konfigurierbaren Intervall geschehen, mit dem Default-Wert von 15 Minuten.

Dies stellte sich als eine schwierigere Aufgabe heraus, da wir dadurch, dass der neu erstellte Import-Task als Guice Module erstellt wurde, mussten wir uns zusätzlich zum Iterieren über den Konfigurierten

Ordner, auch noch um das Sicherstellen der Datenbankverbindung in diesem Externen Module kümmern, was aufgrund der in der Vorarbeit Verwendeten Libraryversionen nicht sonderlich leichtfiel.

Hier wäre ein Refactoring der gesamten static Methods in den Klassen evtl. ein besserer Ansatz gewesen, dies wäre jedoch eine relativ komplexe Angelegenheit geworden.

Listen welche noch nicht bearbeitet wurden, sollen beim erneuten Import der Daten überschrieben werden.

Auch bei dieser Funktion mussten einige Checks in den Importprozess gebaut werden. Der schwerste Teil dieser Implementationsänderung war jedoch die korrekten Abhängigkeiten für das Löschen aller anhängenden Objekte auf der Domainebene der Vorarbeit. Diese waren schlicht weg einfach gar nicht vorhanden.

Listen sollen gelöscht werden können.

Siehe Beschreibung vorheriges Unterkapitel

Wenn der Server neu startet soll dafür gesorgt sein, dass die Applikation wieder von selber starten kann.

Aufgrund der neu geforderten Konfigurierbarkeitsanforderungen haben wir uns hier für die Distribution eines ZIP Archives für die Installation des Servers entschieden.

Jedoch blieb hier nun immer noch die Frage, wie wir nun die Applikation wieder automatisch unter Windows starten können.

Zuerst sind wir bei einer Recherche im Internet auf «Schtasks.exe»[Reference] gestossen. Dies haben wir evaluiert, jedoch sind wir bei Tests mit dem Tool auf Limitierungen gestossen, welche es für unsere Applikation nicht besonders geeignet erschienen lies.

Danach sind wir über gegangen zu der eigentlich üblichsten Variante. Wir haben den Startup Task über Windows Services gelöst. Hierfür sind wir bei unseren Recherchen über ein hilfreiches Tool (https://nssm.cc/) gestossen zur simplen Anbindung von .bat Scripts an einen Windows Service. Dies eignete sich aus dem Grund besonders gut, da die Standard Distribution welche mit dem play-framework eigenen Distribution Prozess jeweils für Windowssysteme ein .bat Script mit erstellt, welches so mit relativ kurzen .bat Scripts einfach an den Windows Service angeschlossen werden kann.

Auf die Logs soll einfacher Zugriff gewährleistet werden.

Wir haben den Logger so konfiguriert, dass er den Pfad für die Logs aus der normalen Konfigurationsdatei liest. Des Weiteren haben wir auf den sogenannten «RollingFileAppender» eingesetzt und so konfiguriert, dass Logdateien, welche grösser als 5MB werden, automatisch kopiert werden und für 30 Tage archiviert werden, so hat man immer schnell die aktuellen Logs zur Hand.

Zudem werden die Logs nun direkt in der Administrationsansicht der Applikation angezeigt.

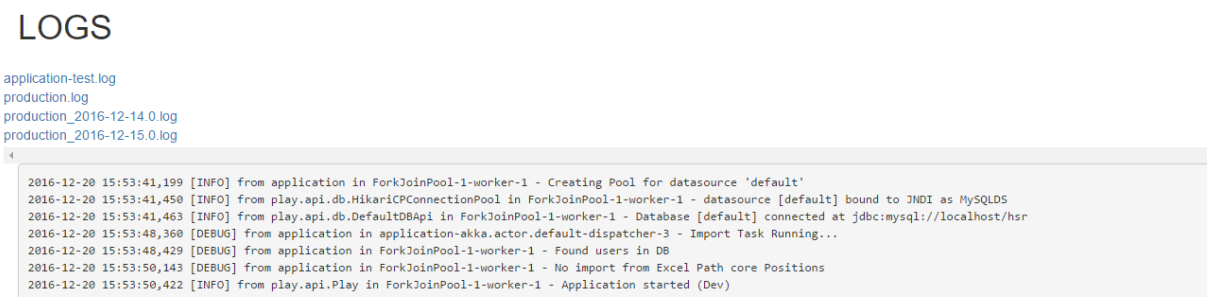


Abbildung 8 Log-Übersicht

Überarbeiten einiger Fehlermeldungen auf der Android-Applikation, sodass bei Netzwerkfehlern eindeutig erkennbar ist, wo der Fehler liegt.

Fehlermeldungen wo nötig, wurden Angepasst, oder neu hinzugefügt.

Neuste Android-App installieren und jeweils auf Aktualisierungen prüfen.

Hierfür musste erstmal einen Möglichkeit geschaffen werden, wie man an die .apk Files für die beiden Android-App Versionen herankommt. Dafür wurde im AndroidController einige neue HTTP GET Routes erstellt. Diese Lesen den Konfigurierbaren Filenamen und senden die jeweiligen Files als Download.

In den Android Applikationen wurde nun der DownloadManager genutzt um zu prüfe, ob sich die Files auf dem Server geändert hatten, wenn Ja, dann wird die neue Version heruntergeladen und dem Benutzer zur Installation aufgerufen.

Bilder für die Kerne sollen, beim erneuten Import aktualisiert werden.

An dieser Funktion musste nicht viel geändert werden, da die Änderungen bereits übernommen wurden

Kerndaten sollen beim erneuten Import aktualisiert werden.

Gleich wie bei den Bildern war diese Anforderung bereits abgedeckt.

Des Weiteren haben wir die zwei Funktionen «Bilder Importieren» und «Kerndaten Laden» zusammengenommen, da die eine Funktion eh immer nach der andern Ausgeführt wird und diese voneinander Abhängig sind

Anleitungen und Erleichterungen zur Installation der Server-Applikation

Siehe Beschreibung Startup-Task und Anleitungen im Anhang dieses Berichts

Optimierung

Datenanalyse

Wichtig für die Lageroptimierung war zuerst einmal die Datenanalyse. Dazu zählt hier auch die Planung des benötigten Datenmodelles, welches für die Funktionen, welche zu implementieren waren von Nöten sind.

Dieses Datenmodell ist auch schon im Kapitel Domainanalyse etwas unter die Lupe genommen worden. Soll hier aber nun noch genauer behandelt werden und der Datenursprung genau definiert werden.

Dieses Datenmodell kann genauer nochmals in drei Untersektionen geteilt werden, welche alle ihre eigene Funktionalität haben. Zum ersten haben wir das physische Lager selber, zum zweiten haben wir die Zuteilung der Kerntypen an Ihre Lagerpositionen und zum dritten haben wir die Kernlisten, diese beschreiben die Reihenfolge, in welcher die Kerne aus dem Lager auf die einzelnen Palletten geladen werden.

Das Lager

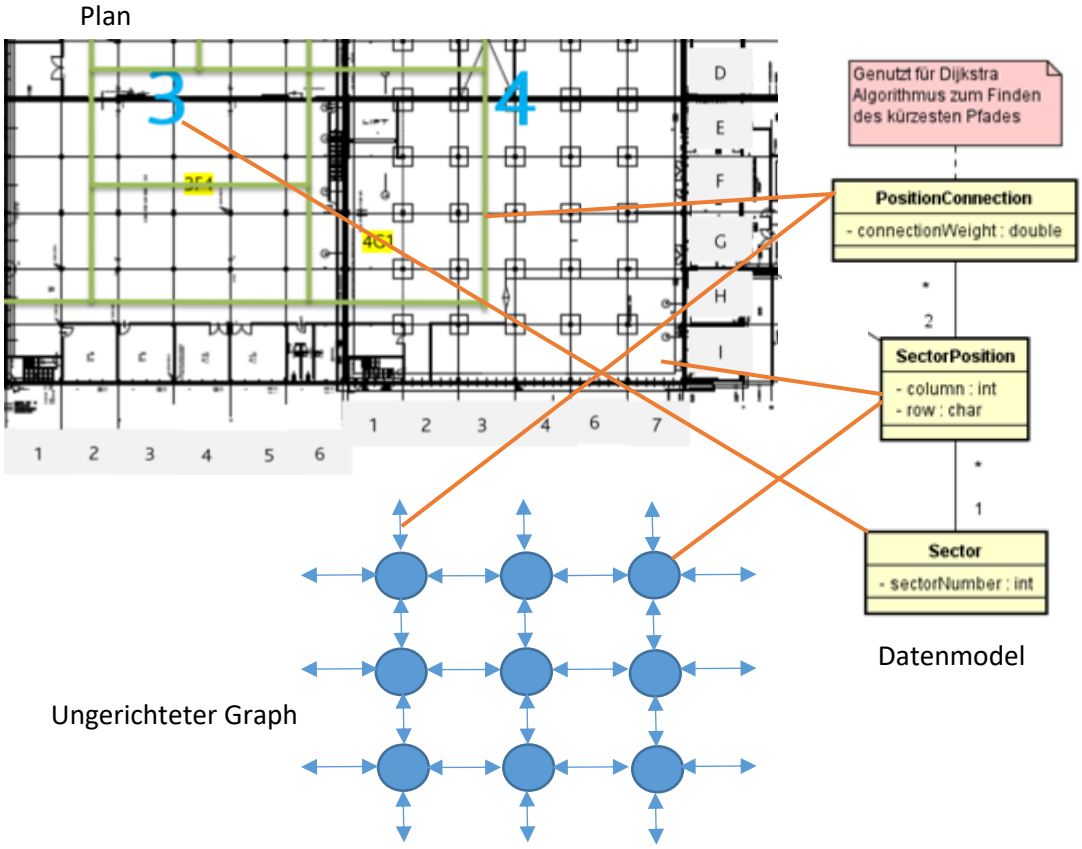


Abbildung 9 Lagerplan Datenübersicht

Die Quelle der Lagerdaten ist der Lagerplan (In der Übersicht linksoben), welcher von der Firma bereitgestellt wurde. Nach diesem Plan wurde das Datenmodell (in der Übersicht rechts) entwickelt um die möglichen Fahrtwege durch das Lager zu erhalten, und auch um die Kürzesten Wege durch dieses Lager zu finden. Dieses Datenmodell hat die Struktur eines ungerichteten Graph für den Path-finding Algorithmus. Die orangen Linien in der Übersicht bedeuten, dass dies die gleichen Daten sind. So ist jeweils der mögliche Fahrweg zwischen zwei Positionen durch eine «PositionConnection» dargestellt und durch den Pfeil im ungerichteten Graph. Der Sektor im Plan ist durch die grosse blaue Nummer gekennzeichnet und als «Sector» im Modell gespeichert. Die Positionen sind im Lagerplan

durch die Reihen und Spalten Kennzeichnung gegeben, im Modell als «SectorPosition» gespeichert und im ungerichteten Graph als blauer Knoten Dargestellt. So kann eine SectorPosition eindeutig über die Reihen- und Spaltenkennzeichnung, sowie die Sektornummer davor gekennzeichnet werden. Beispielsweise «1B3», steht für die Position in der Reihe B und Spalte 3, im Sektor mit der Sektornummer 1. Dies ist so möglich, da es dank dem Buchstaben zwischen den Zahlen keine Verwechslung geben kann

Zuteilung der Kerne an Ihre Lagerposition.

Für die Zuteilung der Kerne an ihre Sektor-Positionen, benötigen wir mehrere Sektorpläne. Dies ist nötig für einerseits die Zuteilungen, welche aus einer Excel-Liste importiert werden, wie auch für die neu zu generierenden Zuteilungen der Kerne an neue Positionen nach Häufigkeit des Kernes in den Produktionslisten.

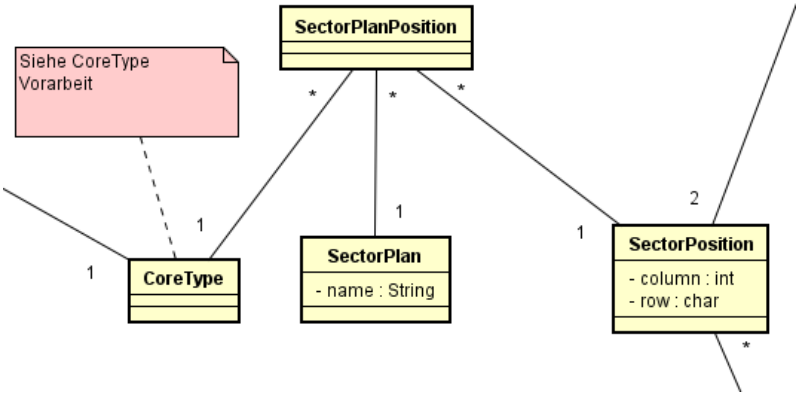


Abbildung 10 Daten der Lagerpositionen

Beim Import aus einer Excel-Liste werden in der Tabelle vier Spalten benötigt. Die Kernnummer, die Kernbezeichnung, die Sektornummer und die Sektor-Position.

	A	B	E	V	W
1	ArtNr	ArtBez	LB	Sektor[1-5]	KoordinatenA1
2	HS20000.61	Kern Baby 60x120x8	5	3	D5
3	HS20000.63	Kern Baby 70x140x8	20	3	D5
7	HS20001.22	Monokern 90x200x12	21	1	B3

Abbildung 11 Datenherkunft

Kernlisten

	A	B	C	D	E	F	G	H	I	J
1	Kernliste			2 RV HR	Pl.Nr	20160111002		3V5	EmailListe	
3	Pal	Nr	Art. Nr	Name	Auftr. Nr	Enddat.	PANR	Kern Nr.	Kern	Linie
4	1	10	HC1110.22.6037	>> Curem Heaven S260 90x200 DJ Curem	1700614336	11.01.16	4003260	HS20216.24	Kern Curem Heaven 25	HA
5	1	11	HC1111.22.6037	Curem Heaven Forte S260 90x200 DJ Cur em	1700614337	11.01.16	4003261	HS20691.22	Kern Curem Heaven Forte 25 2.0	HA
6	1	12	H1347.12.7230	>> AirRelax 90x190 DJ Premio	1700614510	11.01.16	4003199	HS20704.22	Kern AirRelax 90x200x20 cm	HA
7	1	13	H1347.27.7230	AirRelax 160x200 DJ Premio	1700614507	11.01.16	4003200	HS20704.27	Kern AirRelax 160x200x20 cm	HA
8	1	14	H1347.27.7230	AirRelax 160x200 DJ Premio	1700614507	11.01.16	4003200	HS20704.27	Kern AirRelax 160x200x20 cm	HA
9	1	10	H1196.22.6585	VitaLuxe 4b soft 90x200 DJ Prestige	1700613343	11.01.16	4003165	HS20224.22	Kern VitaLuxe soft II 89x200	HB
10	1	11	H1197.22.6585	VitaLuxe 4b med 90x200 DJ Prestige	1700613343	11.01.16	4003166	HS20225.22	Kern VitaLuxe med II 89x200	HB
11	1	12	H1197.22.6585	VitaLuxe 4b med 90x200 DJ Prestige	1700614649	11.01.16	4003167	HS20225.22	Kern VitaLuxe med II 89x200	HB
12	1	13	H1197.22.6585	VitaLuxe 4b med 90x200 DJ Prestige	1700614649	11.01.16	4003167	HS20225.22	Kern VitaLuxe med II 89x200	HB
13	1	14	H1199.22.6585	Finesse 4b soft 90x200 DJ Prestige	1700615305	11.01.16	4003168	HS20242.22	Kern Finesse II soft 89x200	HB
14	1	15	H1199.22.6585	Finesse 4b soft 90x200 DJ Prestige	1700613343	11.01.16	4003169	HS20242.22	Kern Finesse II soft 89x200	HB
15	1	16	H1200.22.6585	Finesse 4b med 90x200 DJ Prestige	1700615305	11.01.16	4003171	HS20243.22	Kern Finesse II m 89	HB
16	1	17	H1200.22.6585	Finesse 4b med 90x200 DJ Prestige	1700613400	11.01.16	4003172	HS20243.22	Kern Finesse II m 89	HB
17	2	20	H1383.22.7238	Climabal Luxe II d 90x200 DJ Select	1700615179	11.01.16	4003221	HS20231.22	Kern AirPulse II d 89x200x22	HB
18	2	21	H1201.22.6585	Finesse 4b dura 90x200 DJ Prestige	1700613400	11.01.16	4003176	HS20244.22	Kern Finesse II d 89	HB
19	2	22	H1319.21.7217	ClimaDream med. 80x200 DJ Coolmax HCS20	1700604917	11.01.16	4003178	HS20678.21	Kern ClimaDream med 79x199x22	HB
20	2	23	H1319.22.7217	ClimaDream med. 90x200 DJ Coolmax HCS20	1700613715	11.01.16	4003179	HS20678.22	Kern ClimaDream med 89x199x22	HB
21	2	24	H1319.22.7217	ClimaDream med. 90x200 DJ Coolmax HCS20	1700613715	11.01.16	4003179	HS20678.22	Kern ClimaDream med 89x199x22	HB

Abbildung 12 Kernliste in Excel

Hierfür müssen die Listen mit den jeweiligen Palletten gespeichert werden. Eine Liste wird durch eine sogenannte PathCoreList repräsentiert. Der Name der PathCoreList ist der Dateiname des Excelfiles. Die Palletten speichern welche Palettennummer sie bekommen haben, sowie zur Filterung auch ihre zugehörige Produktionslinie. Der PathCore auf der Palette speichert jeweils noch seine Position innerhalb der Palette und zu welchem Kerntyp er gehört.

Die Listen werden auch aus Excelfiles Importiert, genauer gesagt aus den Produktionslisten der Firma. Sie können aber auch für die Optimierung generiert werden.

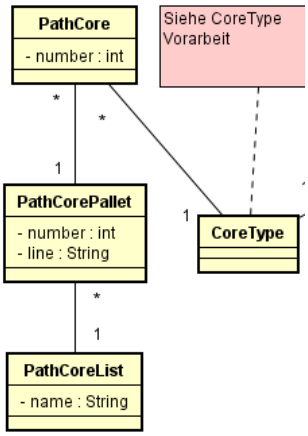


Abbildung 13 Datenspeicherung

Umsetzung

Path-Finder

Der Pathfinding-Algorithmus der bei dieser Arbeit zum Einsatz kommt, ist eine Implementation des Dijkstra-Algorithmus. Der Algorithmus sucht sich den Kürzesten Pfad anhand eines ungerichteten Graph. Die Implementation musste für dieses Domainmodell etwas angepasst werden. Sie basiert auf folgender Quelle: http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html

Listenimport

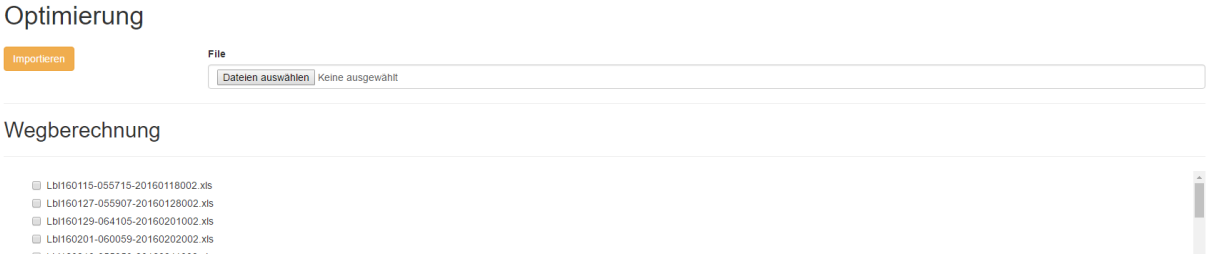


Abbildung 14 Listenimport

Das Excelfile wird hochgeladen und gelesen. Mit dem Namen der Datei wird eine neue PathCoreList erzeugt.

Nun wird aus der Kernliste im Excelfile pro Kern ein neuer PathCore erzeugt und dem richtigen CoreType zugewiesen. Ist die dazugehörige PathCorePallet noch nicht vorhanden, so wird auch diese mit der Nummer und der Produktionslinie angelegt.

Die neue Liste wird nach dem generieren unter «Wegberechnung» angezeigt.

Plan erstellen

Eine Neue Kernzuordnung zu Ihren Positionen im Lager kann auf zwei Arten erstellt werden. Einerseits kann Sie durch einen Algorithmus erstellt werden, welcher nach der Häufigkeit der kerne geht, und die Kerne so neu im Lager positioniert, oder die Positionen können aus einem Excelfile gelesen werden.

Abbildung 15 Plan erstellen aus Excelfile

Bei dieser Methode muss der Benutzer das File hochladen, und die benötigten Spalten (Kernnummer, Kern Bezeichnung, Sektornummer und Sektor-Position) angeben. Dann muss auch noch ein Name für

den neuen Plan angegeben werden, und das Excel Sheet in welchem die Daten sind. Hieraus wird nun ein neuer Lagerplan mit den Kern Zuordnungen zu den SectorPositions erstellt.

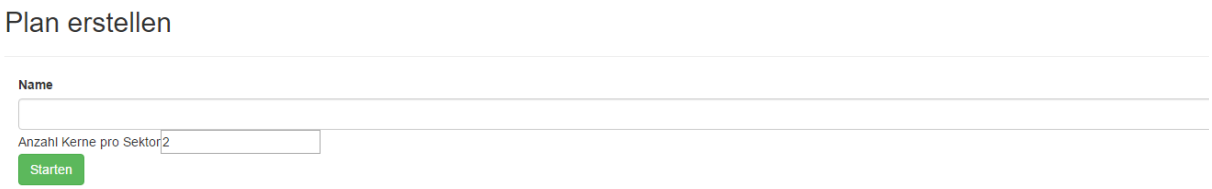


Abbildung 16 Plan generieren

Zum anderen kann der Plan auch generiert werden.

Hierfür muss ein Name angegeben werden und die Anzahl Kerne, welche pro Sektor Verteilt werden sollen.

Das Erste was diese Funktion macht, ist den neuen Plan generieren.

Danach erstellt es eine nach Häufigkeit der Kerne absteigend sortierte Liste aller geladenen Kerne.

```
final Map<CoreType, Integer> counterCores = new HashMap<>();
for (CoreType pathCore : pathCores) {
    counterCores.put(pathCore, 1 + (counterCores.containsKey(pathCore) ? counterCores.get(pathCore) : 0));
}

List<CoreType> coreTypes = new ArrayList<>(counterCores.keySet());
coreTypes.sort((x, y) -> counterCores.get(y) - counterCores.get(x));
```

Abbildung 17 Sortierung der Kerne

Damit hat man die Liste der Kerne, welche man in dieser Reihenfolge so nahe wie möglich am Liftsektor positionieren will.

Um auf die Distanz zu kommen, welche eine Position vom Liftsektor entfernt ist, müssen wir diese Distanzen mit dem Pathfinding-Algorithmus berechnen lassen, und eine Liste erstellen, diesmal aber in aufsteigender Reihenfolge, sprich den Sektor mit der kürzesten Distanz zuerst.

```
PathFinder pathFinder = new PathFinder().init().from(getLiftSector());
final Map<SectorPosition, Integer> distanceToPositions = new HashMap<>();
for (SectorPosition sectorPosition : AbstractEntity.getList(SectorPosition.class)) {
    distanceToPositions.put(sectorPosition, pathFinder.to(sectorPosition).size());
}

List<SectorPosition> sectorPositions = new ArrayList<>(distanceToPositions.keySet());
sectorPositions.sort(Comparator.comparingInt(distanceToPositions::get));
```

Abbildung 18 Sortierung der Distanzen

Danach wird über diese beiden Listen Iteriert und je nach angegebener Zahl des Benutzers die Menge der Kerne auf den Sektoren verteilt.

Egal mit welcher dieser beiden Methoden der Plan erstellt wird, wird er danach auch unter Wegberechnung angezeigt.

Berechnung des Fahrweges

Wegberechnung

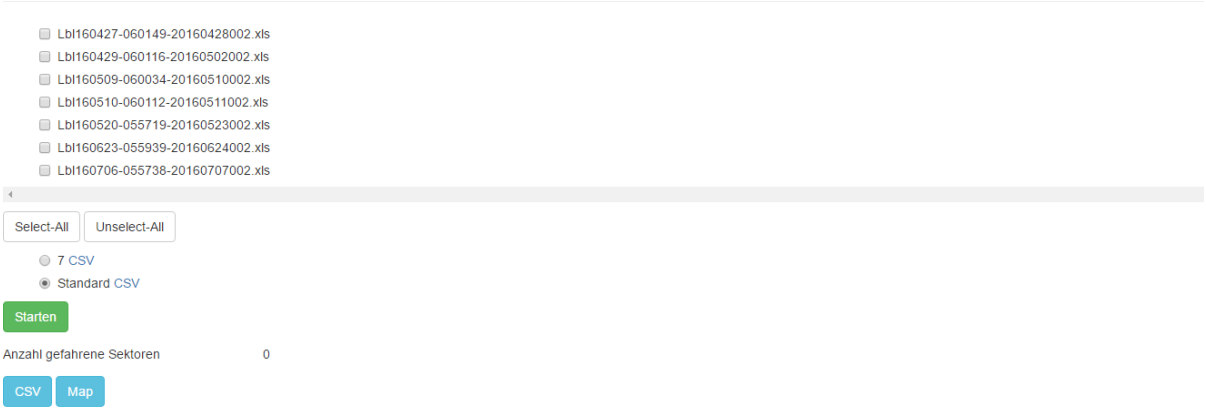


Abbildung 19 Wegberechnung

Für die Berechnung des Fahrweges kann eine oder mehrere Listen ausgewählt werden, auf welchen danach eine Durchfahrt Simuliert wird.

Dies läuft wie folgt ab:

Für jede der selektierten Listen wird die Mehode PathOptimizer.run mit der Liste und dem entsprechend gewählten Plan ausgeführt. Diese Methode merkt sich für jede Palette in der Liste zuerst einmal die Lift-Position als erste Position. Das Bedeutet, dass für jede neue Palette beim Lift gestartet wird. Von dort wird mit Hilfe des Pathfinding-Algorithmus die der Weg zur ersten Kernposition auf der Palette berechnet. Diese Position wird nun als die Letzte Position abgespeichert, damit der Pathfinding-Algorithmus nun ab dieser weiter laufen kann. Dies wird für jeden Kern auf der Palette wiederholt. Am Ende einer Palette wird nochmals der Weg zurück zum Lift mit berechnet.

Dieser Prozess ergibt den gesamten Fahrweg einer Liste zurück.

Im Controller wird aus dem Fahrweg noch das entsprechende CSV für das anzeigen generiert.

In der Übersicht sieht nun der Nutzer die Anzahl der gefahrenen Sektoren.

Sollten nicht alle Kerne im Lagerplan gefunden worden sein, kommt eine folgende Meldung

Abbildung 20 Wegberechnung Fehlende Kernpositionen

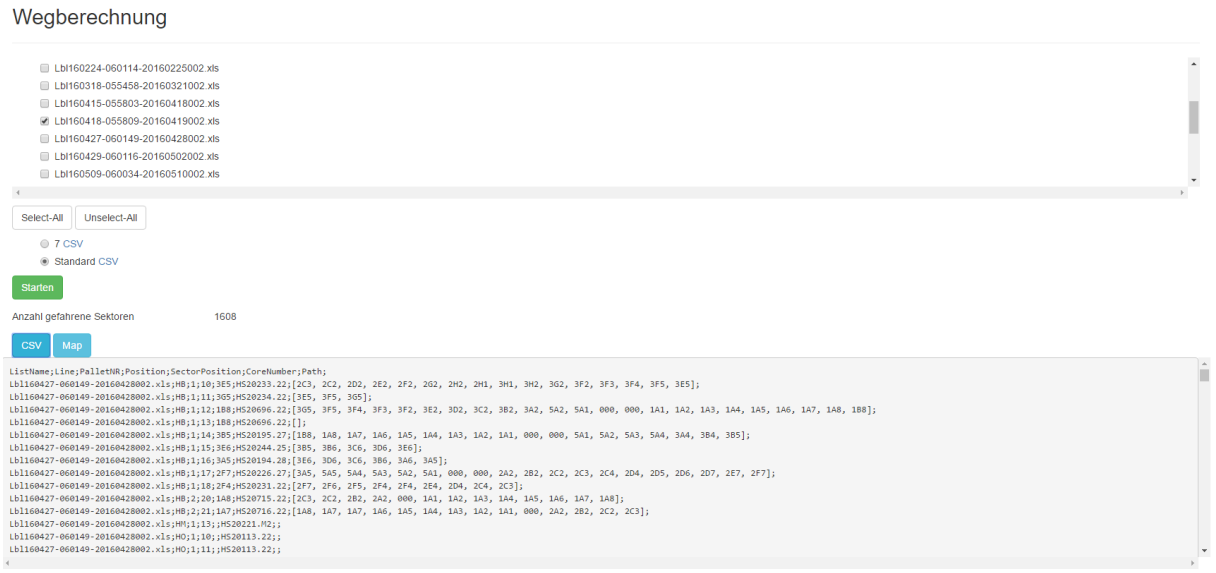


Abbildung 21 CSV

Das generierte CSV kann durch klicken auf den blauen CSV Button angezeigt werden.

Durch klicken auf den Button Map kann eine Heatmap der gefahrenen Strecke angezeigt werden.

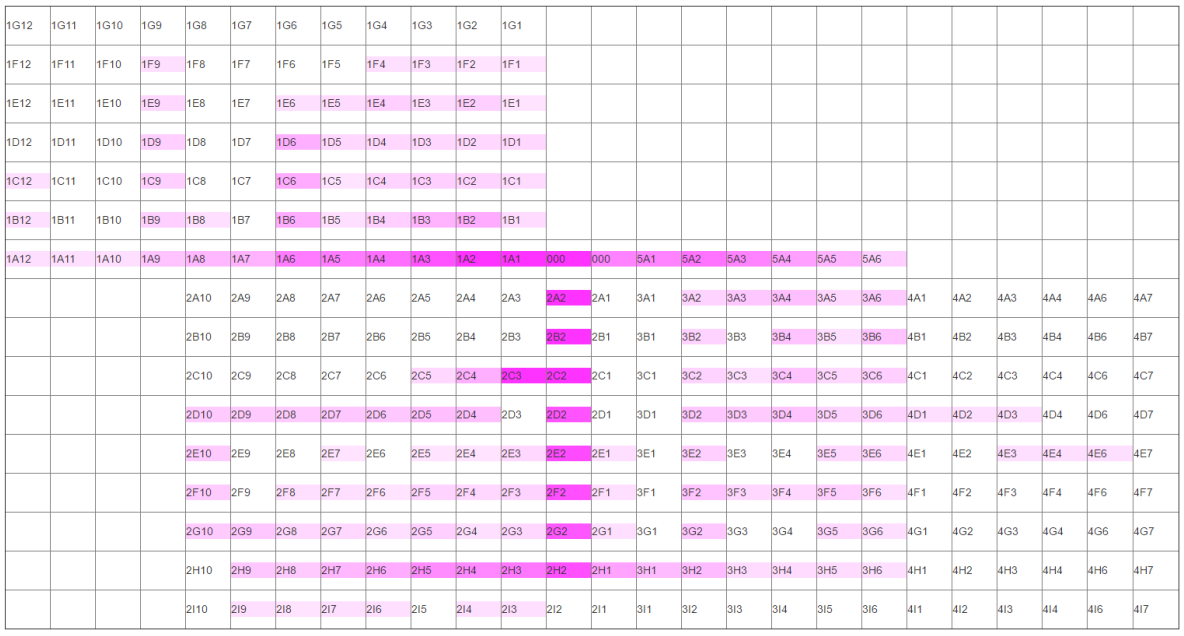


Abbildung 22 Heatmap

CSV für Plan erstellen

Hier wird für jeden Kern der im Plan positioniert wurde ein Eintrag im CSV File generiert.

Neue Liste generieren

Schliesslich gibt es noch die Möglichkeit anhand einer alten Liste und eines Planes eine neue optimierte Liste zu generieren. Diese Funktion dient lediglich der Veranschaulichung des Potentials der Optimierung. Die Funktion ist insofern nicht zur Generierung realer neuer Produktionslisten zu brauchen, als sie nicht die Stapelungsregeln des in der Firma genutzten Algorithmus berücksichtigt.

Sie versucht jedoch anhand des Folgend erklärten Algorithmus eine möglichst optimale Liste zu generieren.

Abbildung 23 Listen erzeugen

Die Funktion ist durch das klicken des Buttons «Neue Listen erstellen» erreichbar.

Folgend werden die neuen Listen generiert:

Der Nutzer gibt an, wie die neue Liste heissen soll, anhand welcher Liste eine neue optimierte Liste erstellt werden soll, auf welchem Lagerplan die Liste optimiert werden soll, und die Anzahl möglicher Kerne auf einer Palette für die neue Liste.

Zuerst wird für jede Produktionslinie in der alten Liste, eine Sammlung aller benötigten Kerne erstellt, damit die Produktionslisten später nicht durcheinandergemischt werden.

Hiernach wird Folgender Ablauf für alle Produktionslinien und deren Kerne durchlaufen.

Zuerst wird eine neue PathCorePallet erstellt und der neuen Liste und der Produktionslinie zugewiesen.

Als aktuelle Position wird sich der Lift-Sektor gemerkt.

Solange in der Produktionsliniensammlung noch Kerne vorhanden sind, wird die Sammlung aufsteigend nach der Distanz zwischen der Kernposition im Lagerplan und der aktuell gemerkten Position sortiert, sodass der Kern, welcher am nächsten zu der gemerkten Position zuoberst ist.

```
final Map<PathCore, Integer> counterPositions = new HashMap<>();
for (PathCore pathCore : pcl) {
    SectorPlanPosition position = pathCore.getCoreType().getSectorPlanPositionFor(baseSectorPlan.getName());
    counterPositions.put(pathCore, pathFinder.to(position.getSectorPosition()).size());
}
pcl.sort(Comparator.comparingInt(counterPositions::get));
```

Abbildung 24 Sortieren nach Distanz zur aktuellen Position

Der Oberste Kern wird aus der Sammlung genommen, und auf die Palette gespeichert. Die Position des Kerns wird als aktuelle Position gespeichert.

Ist eine Palette Voll, was anhand der eingegeben Anzahl bestimmbar ist, so wird die aktuelle Position wieder auf den Lift-Sektor gesetzt, wodurch dort wieder neu gestartet wird.

Am Ende kommt eine Neue Liste dabei heraus, welche Versucht jeweils die kürzesten Distanzen zwischen zwei Kernen hintereinander zu Nutzen.

resetLists

Mit dieser Funktion werden alle importierten und generierten Listen entfernt.

resetPlan

Mit dieser Funktion werden alle importierten und generierten Pläne entfernt.

5.3.3 Schnittstellenbeschreibung

Tabelle 21 Schnittstellenspezifikation

Nr	Request	Pfad und Methode (@controllers.)	Beschreibung
		##Android	
1	POST	/android/missingMaterial/add AndroidController. addAndroidMissingMaterial()	Dieser Methode wird ein MissingMaterialDTO als Post-Data mitgegeben (JSON Format), welches dann auf dem Server gespeichert wird.
2	GET	/android/getAppName AndroidController. getCurrentAppName()	Gibt den Namen der aktuellsten Version der Tablet App zurück.
3	GET	/android/getApp AndroidController. getCurrentApp()	Gibt die neuste Version der Tablet App zum Aktualisieren mit. Als APK File.
4	GET	/android/getQIssueMail AndroidController. getQIssueMail()	Gibt die aktuell eingetragene E-Mail-Adresse für Qualitätsmeldungen zurück.
5	GET	/android/getThinAppName AndroidController. getCurrentThinAppName()	Gibt den Namen der aktuellsten Version der Mobile App zurück.
6	GET	/android/getThinApp AndroidController. getCurrentThinApp()	Gibt die neuste Version der Mobile App zum Aktualisieren mit. Als APK File
		##Config	
7	GET	/webapp/config/view ConfigController.overview()	Öffnet die Konfigurationsübersicht.
8	POST	/webapp/config/edit ConfigController.edit()	Gibt alle Daten in der Konfig-Ansicht dem Server, damit dieser sie speichern kann.
		##Logs	
9	GET	/webapp/log/:file DownloadController. log(file: String)	Gibt das Logfile mit dem Namen des file Parameters zurück.
10	GET	/webapp/log DownloadController.log(file: String = "")	Gibt das default Logfile zurück.
		##Cutview	
11	GET	/webapp/cutviewList ListController.cutviewOverview()	Öffnet die Übersicht aller verfügbaren Produktionslisten zur Auswahl des benötigten Tages.

12	GET	/webapp/cutviewDetail/:id/:filter ListController. cutviewDetail (id: Long, filter:String)	Öffnet eine Produktionsliste wobei id der Name=Tag der Liste ist und filter der zu öffnende Tab von Linien ist.
13	GET	/webapp/toggleCoverDone/ :id/:coverListId/:filter TransferController. toggleCoverDone (id:Long, coverListId:Long, filter:String)	Setzt ob ein Position einer Produktionsliste Erledigt wurde (oder zurück), die Attribute sind id, die Id der Position in der Liste, coverListId, die Id der Produktionsliste, und filter ist der zu öffnende Tab von Linien.
14	GET	/webapp/import/coverPosition CarrierController. importCoverPosition()	Öffnet die Seite zum Hochladen eines Rollenlagers. Das ist die Zuteilung von Stoffrollen auf eine Lagerposition im Rollenlager.
15	POST	/webapp/import/coverPosition ImporterController. uploadCoverPosition()	Hier wird eine Rollenlagerliste mitgegeben, welches im Server gespeichert wird. Die vorherige Liste wird gelöscht.
		##Path Optimization	
16	POST	/webapp/path/loadcorepos PathOptimizerController. readCorePositions()	Hier kann eine Exceldatei mitgegeben werden für den Import der Lagerpositionen der Kerne.
17	GET	/webapp/path/optimize PathOptimizerController. runOptimizationFor (sectorPlan:String, selectedPathCoreLists: java.util.List[String])	Man gibt der Methode einen Lagerplan und eine importierte Kernliste mit. Der Algorithmus fährt dann die Kernliste ab und gibt zurück, wie viele Sektoren er durchquert hat.
18	GET	/webapp/path/overview PathOptimizerController. showOptimization()	Öffnet die Ansicht der Optimierung.
19	POST	/webapp/path/upload ImporterController. uploadPathSheet	Gibt man dieser Methode eine oder mehrere Produktionslisten als Excelfiles mit, kann man sie später für die Optimierung auswählen.
20	GET	/webapp/path/generate PathOptimizerController. generatePlan(name:String, useSpecNumber:String = "TRUE", specNumber:String, useLists:String = "FALSE")	Erstellt anhand der Häufigkeit der importierten Kerne einen neuen Lagerplan.

21	GET	/webapp/path/resetPlan PathOptimizerController. resetSectorPlan()	Löscht die generierten und importierten Lagerpläne.
22	GET	/webapp/path/resetList PathOptimizerController. resetLists()	Löscht die generierten und importierten Kernlisten.
23	GET	/webapp/path/generatePathCore List PathOptimizerController. generateNewPathCoreList(newPathCoreListName, oldPathCoreList, basePlanName, numberOfCores = "10")	Generiert eine optimierte Liste anhand einer anderen Liste und einem Lagerplan.
24	GET	/webapp/path/getCSVSectorPla /:name PathOptimizerController. getCSVSectorPlan (name:String)	Gibt den Lagerplan als CSV File zurück.

5.3.4 Qualitätssicherung

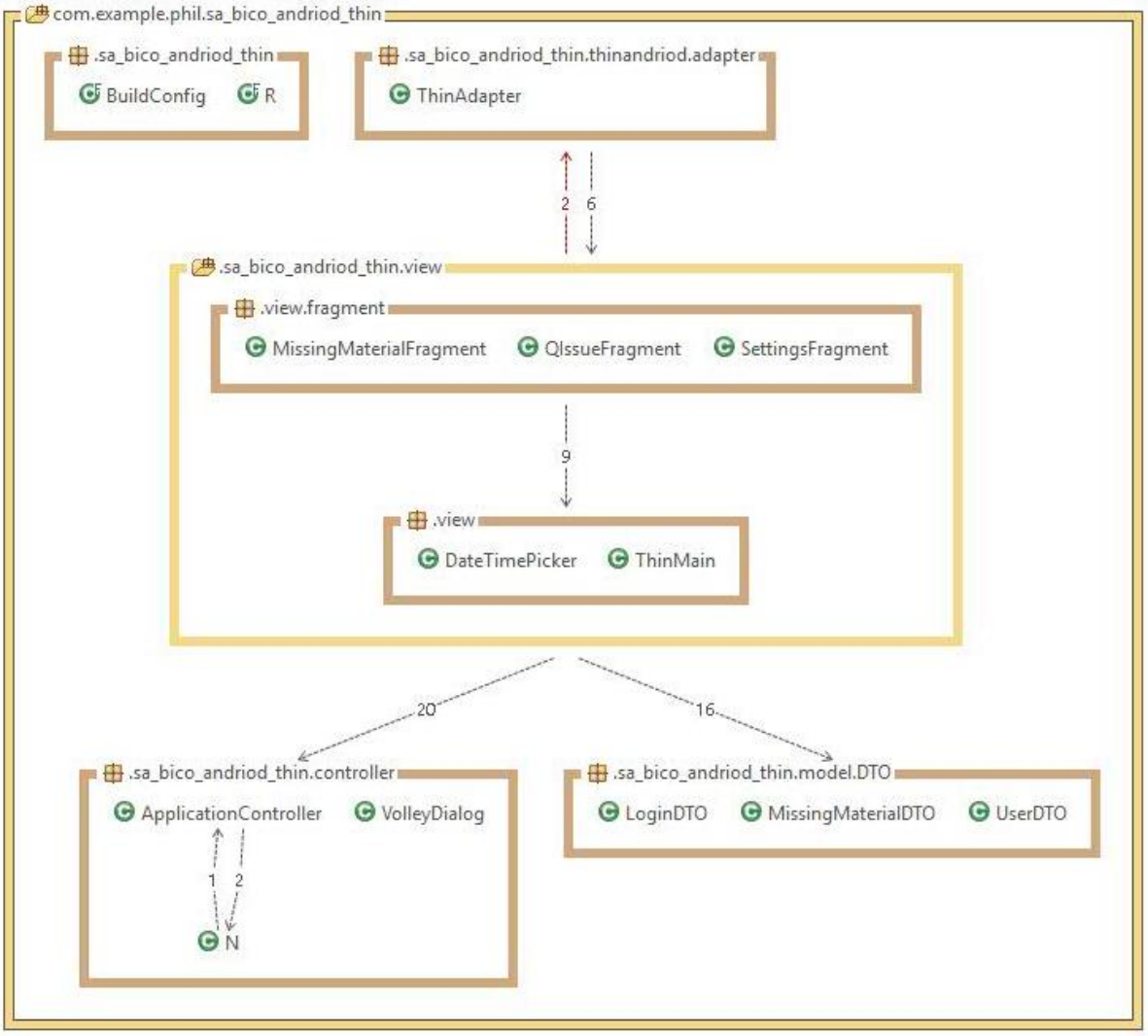


Abbildung 25 Stan Analyse Android

Zu sehen ist die Übersicht der Packages der Mobile App. Diverse Klassen konnten vom Tablet App übernommen werden: MissingMaterialFragment, QIssueFragment, DateTimePicker, LoginDTO, MissingMaterialDTO, UserDTO und VolleyDialog.

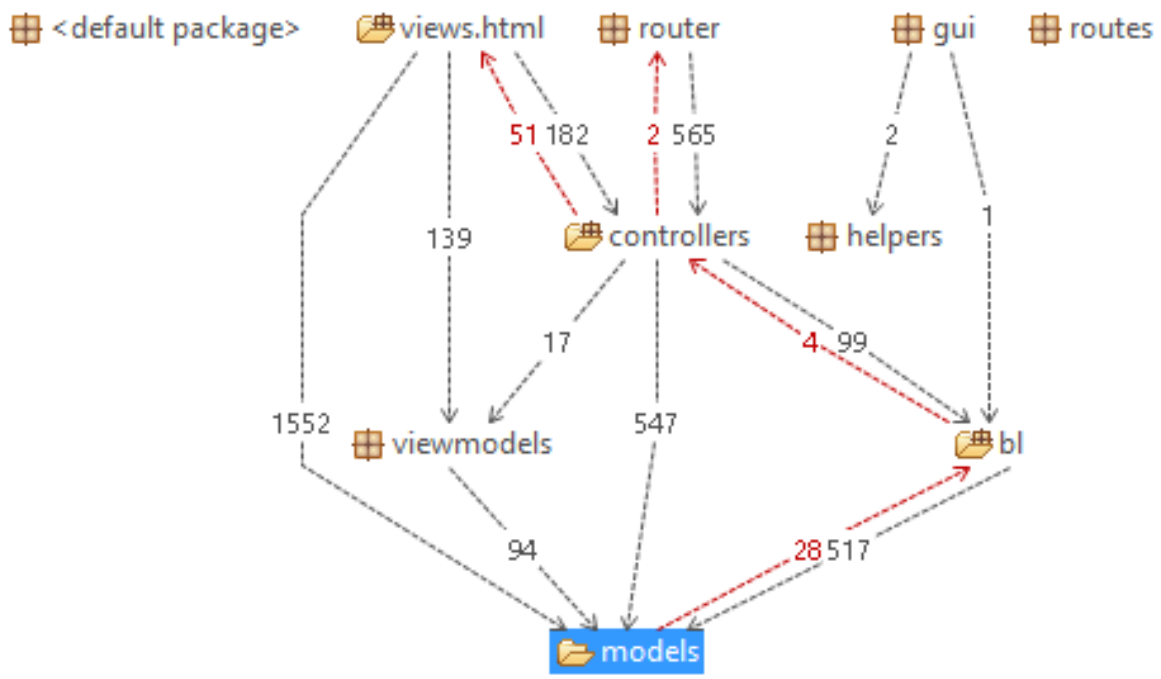


Abbildung 26 Stan Analyse Server

Der Server wurde mehrheitlich übernommen und wir haben an diversen Orten noch Änderungen vorgenommen.

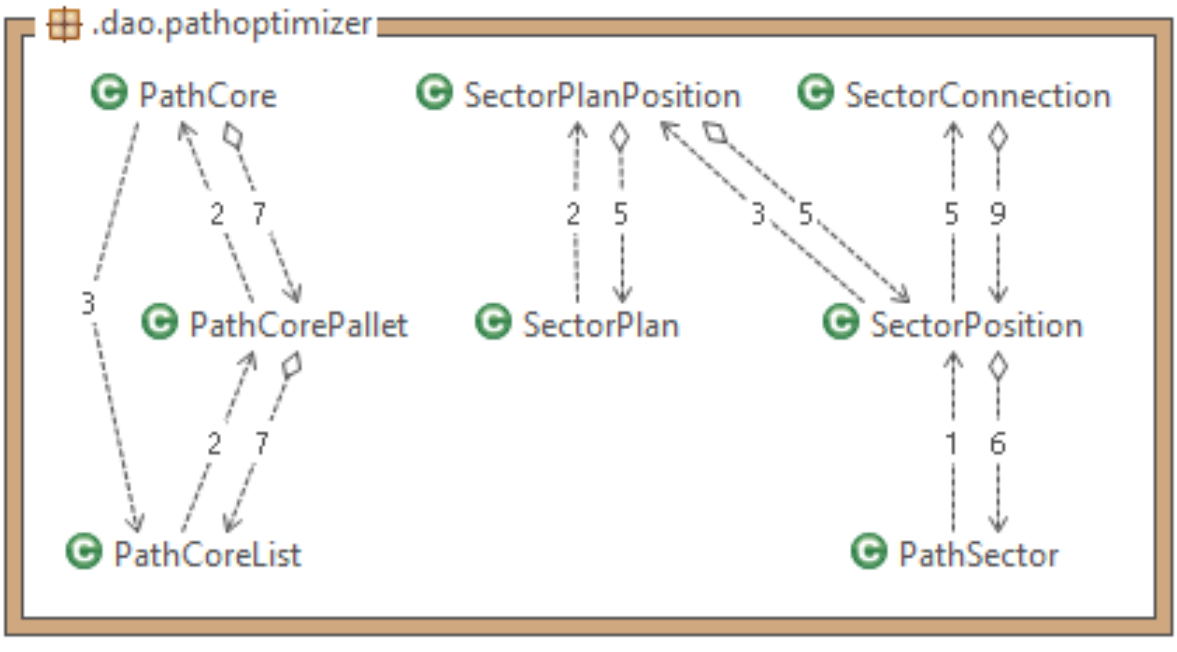


Abbildung 27 Stan Analyse Optimierungsdomain

Das ist die Übersicht über die Klassen der Optimierung, welche komplett von uns erstellt wurde.

Code Analyse

Legende:

LOC = Lines of Code

(rec) = recursive

LOCp = Lines of production Code

LOCt = Lines of test Code

Ev(G) = Essential Cyclomatic Complexity

Iv(G) = Design Complexity

V(G) = Cyclomatic Complexity

Android Tablet App

Package metrics | Module metrics | File type metrics | Project metrics

package	LOC	LOC(rec)	LOCp	LOCp(rec)	LOCt	LOCt(rec)
com.hildinganders.mobileclient.view.fragment	2'552	2'552	2'552	2'552	0	0
com.hildinganders.mobileclient.model.greenDAO	1'962	1'962	1'962	1'962	0	0
layout	1'199	1'199	1'199	1'199	0	0
com.hildinganders.mobileclient.view	439	2'991	439	2'991	0	0
com.hildinganders.mobileclient.controller.adapter	429	429	429	429	0	0
com.hildinganders.mobileclient.controller	290	719	290	719	0	0
com.hildinganders.mobileclient.model.DTO	271	271	271	271	0	0
values	242	242	242	242	0	0
menu	86	86	86	86	0	0
com.hildinganders.generator	85	85	85	85	0	0
com.hildinganders.mobileclient.model	19	2'252	19	2'252	0	0
drawable	8	8	8	8	0	0
		7'582		7'582		0
com		6'047		6'047		0
com.hildinganders.mobileclient		5'962		5'962		0
com.hildinganders		6'047		6'047		0
Total	**7'582**		**7'582**		**0**	
Average	631.83		631.83		0.00	

Abbildung 28 Android Code Metrics

Die meisten Lines Of Code (LOC) befinden sich im Fragments-Package, da dort die meisten http Requests abgesetzt werden, welche extrem viel Code benötigen, um alle möglichen Antworten zu behandeln.

Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

method	ev(G)	iv(G)	v(G)
com.hildinganders.mobileclient.model.greenDAO.MattressCoreDao.bindValues(SQLiteStatement,MattressCore)	1	12	15
com.hildinganders.mobileclient.model.greenDAO.BulkPositionDao.bindValues(SQLiteStatement,BulkPosition)	1	12	14
com.hildinganders.mobileclient.model.greenDAO.MattressCoreDao.readEntity(Cursor,int)	1	12	12
com.hildinganders.mobileclient.model.greenDAO.BulkPositionDao.readEntity(Cursor,int)	1	12	12
com.hildinganders.mobileclient.controller.adapter.PalletListAdapter.onBindViewHolder(ViewHolder,int)	1	12	12
com.hildinganders.mobileclient.model.greenDAO.MattressCoreDao.readEntity(Cursor,MattressCore,int)	1	12	12
com.hildinganders.mobileclient.model.greenDAO.BulkPositionDao.readEntity(Cursor,BulkPosition,int)	1	12	12
com.hildinganders.mobileclient.controller.adapter.BulkCoreListAdapter.onBindViewHolder(ViewHolder,int)	1	10	10
com.hildinganders.mobileclient.controller.adapter.CoreListAdapter.onBindViewHolder(ViewHolder,int)	1	10	10
com.hildinganders.mobileclient.view.fragment.CorelistsFragment.getBulkCoreLists()	3	7	9
com.hildinganders.mobileclient.view.fragment.BulkCorelistFragment.postSetWarehouseCounter(Integer,AlertDialog,BulkPosition)	1	8	8
com.hildinganders.mobileclient.view.fragment.MissingMaterialFragment.onCreateView(LayoutInflater,ViewGroup,Bundle)	2	7	8

Abbildung 29 Android Complexity Metrics

Unsere komplexeste Methode ist das onCreateView() der MissingMaterial-Ansicht (unterste Zeile) und ist somit nicht einmal im roten Bereich. Sie ist unsere komplexeste Methode, da dort ein Spinner, 4 Buttons und 4 Eingabefelder eingelesen werden müssen.

Android Mobile App

Package metrics | Module metrics | File type metrics | Project metrics

package	LOC	LOC(rec)	LOCp	LOCp(rec)	LOCt	LOCt(rec)
com.example.phil.sa_bico_andriod_thin.view.fragment	773	773	773	773	0	0
layout	475	475	475	475	0	0
com.example.phil.sa_bico_andriod_thin.controller	266	266	266	266	0	0
com.example.phil.sa_bico_andriod_thin.view	170	943	170	943	0	0
values	131	131	131	131	0	0
com.example.phil.sa_bico_andriod_thin.model.DTO	55	55	55	55	0	0
com.example.phil.sa_bico_andriod_thin.thinandriod.adapter	36	36	36	36	0	0
com.example.phil.sa_bico_andriod_thin	35	1'335	0	1'300	35	35
com		1'335		1'300		35
		1'941		1'906		35
com.example.phil		1'335		1'300		35
com.example.phil.sa_bico_andriod_thin.model		55		55		0
com.example.phil.sa_bico_andriod_thin.thinandriod		36		36		0
com.example		1'335		1'300		35
Total	**1'941**		**1'906**		**35**	
Average	242.62		238.25		4.38	

Abbildung 30 Android Code Metrics

Die Mobile App ist deutlich kleiner wie die Tablet App aber die grössten Funktionen sind immer noch im Fragment Package.

Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

method	ev(G)	iv(G)	v(G)
com.example.phil.sa_bico_andriod_thin.view.ThinMain.postLogin(LoginDTO)	1	8	8
com.example.phil.sa_bico_andriod_thin.view.fragment.MissingMaterialFragment.onCreateView(LayoutInflater,ViewGroup,Bundle)	2	7	8
com.example.phil.sa_bico_andriod_thin.view.fragment.MissingMaterialFragment.send(MissingMaterialDTO)	1	8	8
com.example.phil.sa_bico_andriod_thin.controller.ApplicationController.updateFromServer(Boolean)	2	6	6
com.example.phil.sa_bico_andriod_thin.view.fragment.QIssueFragment.onCreateView(LayoutInflater,ViewGroup,Bundle)	1	5	5
com.example.phil.sa_bico_andriod_thin.view.fragment.QIssueFragment.compress(String)	2	3	5
com.example.phil.sa_bico_andriod_thin.view.fragment.QIssueFragment.onActivityResult(int,int,Intent)	1	4	4
com.example.phil.sa_bico_andriod_thin.thinandriod.adapter.ThinAdapter.getItem(int)	4	2	4
com.example.phil.sa_bico_andriod_thin.controller.ApplicationController.registerDownloader(String)	1	4	4

Abbildung 31 Android Complexity Metric

Da die Mobile App so klein ist, ist es einfacher, ihre Komplexität klein zu halten.

Server

Package metrics | Module metrics | File type metrics | Project metrics

package	LOC	LOC(rec)	LOCp	LOCp(rec)	LOCt	LOCt(rec)
models.dao	3'477	3'975	2'778	3'276	699	699
views.lists	1'934	1'934	1'934	1'934	0	0
controllers	1'768	1'768	1'768	1'768	0	0
views.carrier_management	1'438	1'438	1'438	1'438	0	0
	932	14'156	932	13'193	0	963
bl.importer	696	696	696	696	0	0
bl.exporter	663	663	663	663	0	0
models.dao.pathoptimizer	498	498	498	498	0	0
bl	398	2'391	398	2'391	0	0
models.dto	328	328	328	328	0	0
bl.pathoptimizer	309	309	309	309	0	0
viewmodels	266	266	266	266	0	0
views.path_optimization	258	258	258	258	0	0
views	253	4'168	253	4'168	0	0
bl.comparator	155	155	155	155	0	0
routes	153	153	0	0	153	153
views.users	142	142	142	142	0	0
views.kpi	85	85	85	85	0	0
bl.util	70	70	70	70	0	0
gui	64	64	0	0	64	64
bl.security	59	59	59	59	0	0
views.transfer	58	58	58	58	0	0
helpers	47	47	0	0	47	47
models.enums	36	36	36	36	0	0
bl.logging	29	29	29	29	0	0
META-INF	28	28	28	28	0	0
bl.exceptions	12	12	12	12	0	0
models		4'339		3'640		699
Total	**14'156**		**13'193**		**963**	
Average	524.30		488.63		35.67	

Abbildung 32 Server Code Metrics

Der Server ist das grösste Stück in unserer Arbeit, obwohl wir davon eher einen kleinen Teil geschrieben haben.

Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

method	ev(G)	iv(G)	v(G)
bl.pathoptimizer.PathOptimizer.generateNewPathCoreList(PathCoreList,SectorPlan,int,String)	7	10	12
bl.importer.ExcelParser.processMovexSheet(String)	3	12	12
bl.PathOptimizerImporter.readCorePositionsPath(String,String,String,String,String,String,String)	6	8	10
bl.pathoptimizer.PathOptimizer.generateNewSectorPlan(String,boolean,int,boolean)	2	9	10
bl.PathOptimizerImporter.generateGrid(int,int,long,int,int,int,int)	1	5	9
models.dao.BulkProductionList.getStatus()	4	6	7
controllers.ImporterController.uploadPathSheet()	1	5	6
controllers.PathOptimizerController.runOptimizationFor(String,List<String>)	1	5	6
controllers.DownloadController.log(String)	1	5	6
controllers.AndroidController.postFinishPallet()	2	5	5
bl.importer.DatabaseImporter.createCarriers(String,String)	1	5	5
controllers.ImporterController.addZipperType()	2	4	5
controllers.ImporterController.uploadExcel()	1	5	5
bl.comparator.ProductionListOverviewComparator.compare(ProductionList,ProductionList)	3	5	5
bl.pathoptimizer.PathOptimizer.getCoreTypesFor(boolean)	1	5	5
bl.importer.ExcelImporter.processFile(String,File)	1	4	5
models.dao.User.authenticate(String,String)	3	4	5
bl.util.FileHelper.filesInFolder(File,List<String>)	1	4	5
controllers.ImporterController.uploadCoverPosition()	1	5	5
models.dao.GeneratedList.getOpenGeneratedListsHandpattern()	1	5	5
models.dao.OrderItem.setSpecialSize()	1	4	5
bl.PrettyPrinter.adjustColumnWidths(String[][],int[])	1	4	5
bl.comparator.ProductionListOverviewComparator.compareEmptyLists(GeneratedList,GeneratedList)	3	1	5

Abbildung 33 Server Complexity Metric

Bei der Optimierung haben wir die komplexesten Methoden. Diese hätte man durch Auslagern kleiner Methoden noch übersichtlicher gestalten können, wofür aber die Zeit nicht mehr reichte.

Integrationtests Protokoll

Aufgrund eingeschränkter Platzverhältnissen in den Tabellen werden folgende Abkürzungen verwendet:

Bc=Barcode

MM=MissingMaterial=Fehlmaterial

NavDraw=Navigationdrawer

QM=Qualitätsmeldung

Android

Tabelle 22 Integrationtests Android

Schritt Nr.	Ausgangslage	Aktion	Soll-Ausgabe	Ist-Ausgabe	Gut/nichtGut
1	NavDraw ist offen	Auswahl Fehlmaterial	Öffnet Fehlmaterial	Öffnet Fehlmaterial	Ok
2	Soll (1)	Senden drücken	Meldung: Bc und Name eingeben	Meldung: Bc und Name eingeben	Ok
3	Soll (1)	Drücke Bc Scannen	Öffnet Kamera mit Bc App	Öffnet Kamera mit Bc App	Ok
4	Soll (3)	Bc Aufnehmen	Schreibt Bc in Bc-Feld	Schreibt Bc in Bc-Feld	Ok
5	Soll (4)	Senden drücken	MM an Server gesendet und Erfolgsmeldung	MM an Server gesendet und Erfolgsmeldung	Ok
6	Soll (4)	Drücke Termin aussuchen	Kalenderansicht öffnet	Kalenderansicht öffnet	Ok
7	Soll (6)	Wähle Termin	Schreibt Termin in Terminfeld	Schreibt Termin in Terminfeld	Ok
8	Soll (7)	Senden drücken	MM an Server gesendet und Erfolgsmeldung	MM an Server gesendet und Erfolgsmeldung	Ok
9	Soll (7)	Wähle Typ Nachbestellung	Spinner zeigt Nachbestellung	Spinner zeigt Nachbestellung	Ok
10	Soll (9)	Senden drücken	MM an Server gesendet und Erfolgsmeldung	MM an Server gesendet und Erfolgsmeldung	Ok
11	Soll (9)	Wähle eine Anzahl	Zeige Anzahl in Feld	Zeige Anzahl in Feld	Ok
12	Soll (11)	Senden drücken	MM an Server gesendet und Erfolgsmeldung	MM an Server gesendet und Erfolgsmeldung	Ok
13	Soll (11)	Schreibe einen Kommentar	Zeige Kommentar	Zeige Kommentar	Ok
14	Soll (13)	Senden drücken	MM an Server gesendet und Erfolgsmeldung	MM an Server gesendet und Erfolgsmeldung	Ok

15	Soll (13)	Zurücksetzten drücken	Löscht alle Eingaben	Löscht alle Eingaben	Ok
16	Soll (15)	Drücke auf NavDraw	Öffnet NavDraw	Öffnet NavDraw	Ok
17	Soll (16)	Wähle Qualitäts-Mangel	Öffnet QM	Öffnet QM	Ok
18	Soll (17)	Drücke Bc Scannen	Öffnet Kamera mit Bc App	Öffnet Kamera mit Bc App	Ok
19	Soll (18)	Bc Aufnehmen	Schreibt Bc in Bc-Feld	Schreibt Bc in Bc-Feld	Ok
20	Soll (19)	Foto aufnehmen	Foto speichern & anzeigen	Foto speichern & anzeigen	Ok
21	Soll (20)	Foto löschen	Foto wird aus Liste entfernt	Foto wird aus Liste entfernt	Ok
22	Soll (20)	Betreff eingeben	Betreff anzeigen	Betreff anzeigen	Ok
23	Soll (22)	Zurücksetzten drücken	Löscht alle Eingaben	Löscht alle Eingaben	Ok
24	Soll (22)	Senden drücken	Öffnet Mail-assistent	Öffnet Mail-assistent	Ok
25	Soll (24)	Mail versenden	Erfolgsmeldung & NavDraw öffnet sich	NavDraw öffnet sich	Keine Meldung vom Mailclient
26	Soll (25)	Update drücken	Öffnet Installations-assistent mit neuster Version	Öffnet Installations-assistent mit neuster Version	Ok

Rollenzuschnitt

Tabelle 23 Integrationtests Rollenzuschnitt

Schritt Nr.	Ausgangslage	Aktion	Soll-Ausgabe	Ist-Ausgabe	Gut/nichtGut
1	Zuschnitt Übersicht geöffnet	Productionlist aussuchen	Öffnet Liste im Tab «Alle»	Öffnet Liste im Tab «Alle»	Ok
2	Soll (1)	Wähle Linie «HM»	Zeige nur Positionen mit Linie «HM»	Zeige nur Positionen mit Linie «HM»	Ok
3	Soll (1)	Klicke Bezugsnummer 6548	Zeige nur Positionen mit Bezugsnummer 6548	Zeige nur Positionen mit Bezugsnummer 6548	Ok
4	Soll (3)	Klicke Filter zurücksetzten	Zeige alle Positionen, Filterfeld leer	Zeige alle Positionen, Filterfeld leer	Ok
5	Soll (1)	Markiere 1. Position als Erledigt	Zeige 1. Position als Erledigt	Zeige 1. Position als Erledigt	Ok

6	Soll (5)	Drücke “Erledigte ausblenden”	1.Position wird ausgeblendet	1.Position wird ausgeblendet	Ok
7	Soll (6)	Drücke «Erledigte ausblenden»	1.Position als erledigt angezeigt	1.Position als erledigt angezeigt	Ok
8	Soll (7))	Markiere 1. Position als nicht erledigt	Zeige alle Positionen gleich	Zeige alle Positionen gleich	Ok

Import

Tabelle 24 Integrationtests Import

Schritt Nr.	Ausgangslage	Aktion	Soll-Ausgabe	Ist-Ausgabe	Gut/nichtGut
1	Importansicht geöffnet	Klicke auf Rollenlager importieren	Öffnet Tab Rollenlager importieren	Öffnet Tab Rollenlager importieren	Ok
2	Soll (1)	Datei aussuchen	Zeigt Datei im Importfeld	Zeigt Datei im Importfeld	Ok
3	Soll (2)	Klicke importieren	Importiert Datei	Importiert Datei	Ok
4	Soll (3)	Klicke auf Bilder & Kerne importieren	Öffne Tab Bilder & Kerne importieren	Öffne Tab Bilder & Kerne importieren	Ok
5	Soll (4)	Klicke Knopf importieren	Kernbilder und Kerndaten werden importiert	Kernbilder und Kerndaten werden importiert	Ok
6	Soll (5)	Klicke auf Excel hochladen	Öffne Tab Excel hochladen	Öffne Tab Excel hochladen	Ok
7	Soll (6)	Klicke auf Datei auswählen	Zeige gewählte Datei an	Zeige gewählte Datei an	Ok
8	Soll (7)	Klicke Knopf importieren	Importiere gewählte Datei	Importiere gewählte Datei	Ok

Listen löschen

Tabelle 25 Integrationtests Listen löschen

Schritt Nr.	Ausgangslage	Aktion	Soll-Ausgabe	Ist-Ausgabe	Gut/nichtGut
1	Management-Administration Listen ist geöffnet	Tagesliste löschen	Löschwarnung, Tagesliste verschwindet aus Ansicht	Löschwarnung, Tagesliste verschwindet aus Ansicht	Ok
2	Soll (1)	Produktionsliste löschen	Löschwarnung, Produktionsliste verschwindet aus Ansicht	Löschwarnung, Produktionsliste verschwindet aus Ansicht	Ok

Konfiguration

Tabelle 26 Integrationtests Konfiguration

Schritt Nr.	Ausgangslage	Aktion	Soll-Ausgabe	Ist-Ausgabe	Gut/nichtGut
1	Admin – Konfiguration ist geöffnet	Änderung vornehmen & speichern drücken	Änderung wird übernommen und im Server gespeichert	Änderung wird übernommen und im Server gespeichert	Ok

Optimierung

Tabelle 27 Integrationtests Optimierung

Schritt Nr.	Ausgangslage	Aktion	Soll-Ausgabe	Ist-Ausgabe	Gut/nichtGut
1	Admin – Optimierung ist geöffnet	Eine oder mehrere Dateien importieren	Dateien werden in Liste angezeigt	Dateien werden in Liste angezeigt	Ok
2	Soll (1)	Eine oder mehrere Listen wählen & Wegberechnung starten	Ausgabe der durchfahrenen Sektoren	Ausgabe der durchfahrenen Sektoren	Ok
3	Soll (2)	Lager erstellen mit Verbrauchslisten	Lager steht zur Auswahl beim Start der Optimierung	Lager steht zur Auswahl beim Start der Optimierung	Ok
4	Soll (2)	Drücke Map	Zeigt Heatmap mit gefahrenen Sektoren	Zeigt Heatmap mit gefahrenen Sektoren	Ok

Usabilitytest Protokoll

Fehlmaterial

1. Sie fahren mit dem Gabelstapler durch das Lager und sehen einen leeren Lagersektor. Um das Fehlen des Materials weiter zu leiten holen sie ihr Handy/Tablet und füllen die Fehlmaterial-Seite aus. Da an der Säule des Lagerplatzes ein Barcode des Produkts hängt können sie diesen direkt einlesen. Sie wissen per Zufall, dass der nächste Auftrag für diesen Kern am 12.12.16 kommt und es werden 12 Stück benötigt.
2. Sie gehen vor Feierabend noch durch das Lager und sehen einen Lagersektor mit nur einer halben Palette. Sie wissen jedoch, dass dort immer mindestens eine ganze Palette liegen müsste(Kanban). Der Barcode auf den verbleibenden Kernen ist leider nicht scanbar. Da es pro Palette 10 Kerne hat müssen sie mindestens so viele nachbestellen. Da sie von keinem bekannten Bedarfstermin wissen können sie auch kein Datum angeben. Sie wissen aber, dass dieser Kern gut läuft und wollen sehr bald Nachschub, fügen sie dies der Meldung an.

Aufgetretene Probleme:

Tabelle 28 Usabilitytest Fehlmaterial

NR	Problem	Lösung	Erledigt
1	Die Datumseingabe ist nicht Intuitiv.	Hier soll mit dem Einbinden eines Datepickers Abhilfe geschaffen werden. Der Datepicker ist über einen gut erkennbaren Button erreichbar	JA
2	Jedes Mal den Namen eingeben ist Umständlich	Der Name soll im Tablett App vom Login genommen werden. Im Mobile App wird eine entsprechende Konfigurationsmöglichkeit bereitgestellt.	JA
3	Die Barcode Funktion ist nicht sonderlich Intuitiv. Der Button ist zu weit vom Input-Feld entfernt	Den Button gut sichtbar neben dem Input platzieren.	JA
4	Die Darstellung der Inputfelder ist ziemlich Verwirrend.	In einem besseren Layout gestalten. Besser ein Tabellen-Layout Verwenden, statt zwei Linear-Layouts.	JA
5	Tastatur ist auf dem Bildschirm bei einzelnen Inputs durch andere Inputs noch im Weg.	Layout dynamisch mit Tastatur anpassen. Tastatur beim Wegklicken von Inputfeld ausblenden.	Ja
6	Bug: Es erscheint eine Fehlermeldung, welche überhaupt nicht in den Kontext passt, nach einer Bestimmten Reihenfolge von Aktionen: Fehlmaterial Senden → Zurück zur Kernliste → Fehlmaterial-View wieder öffnen → Popup mit Fehlermeldung in neuer View!	Testen und Fehler finden.	JA
7	Wenn ein Nutzer die Ansicht verlässt, gehen die eingegebenen Daten verloren. Dies war unklar.	Daten persistieren, bis diese gelöscht oder gesendet werden.	JA

Q-Issue

3. Sie gehen durch das Lager und sehen einen beschädigten Kern. Mit dem Android-App machen sie ein Foto der Beschädigung, scannen den Barcode ein und beschreiben Ort und Art der Beschädigung.

Aufgetretene Probleme:

Tabelle 29 Usabilitytest Q-Issue

NR	Problem	Lösung	Erledigt
1	Wenn ein Nutzer die Ansicht verlässt, gehen die eingegebenen Daten verloren. Dies war unklar.	Daten persistieren, bis diese gelöscht oder gesendet werden.	JA
2	Es fehlt die Möglichkeit mehrere Bilder zu schicken.	Es soll möglich sein mehrere Bilder aufzunehmen und in einer scrollbaren View dargestellt zu bekommen.	JA
3	Ein Beispielbild mit dem Text «Bild aufnehmen» war nicht ganz so Intuitiv.	Durch einen gut sichtbaren Button ersetzen	JA
4	Kommentar Feld ist für den Nutzen überflüssig.	Der Betreff der Email soll direkt aus dem Kommentarfeld genommen werden. Das Feld entsprechend umbenannt	JA
5	Bug: Senden der grossen Bilder kann bei manchen Mail-Servern fehlschlagen.	Bilder auf Android komprimieren und dann Senden.	JA
6	Tastatur ist auf dem Bildschirm bei einzelnen Inputs durch andere Inputs noch im Weg.	Layout dynamisch mit Tastatur anpassen. Tastatur beim Wegklicken von Inputfeld ausblenden.	Ja

Rollenzuschnitt

4. Sie haben am Morgen gerade die erste Rolle zum Zuschnitt geholt und wollen wissen wie viele Teile am heutigen Tag von diesem Stoff gebraucht werden.

Hier bekamen wir sehr viel Feedback, da diese Feature zum ersten Mal umgesetzt und ausprobiert wurde. Wir wollen die Liste nun um folgende Funktionen erweitern:

- «Erledigt» Feld, um zugeschnittene Positionen abzuhaken.
  - In einer neuen Spalte in der Liste umgesetzt.
- Filtermöglichkeit nach Bezugs- und Produktnummern.
  - Links auf den Nummern aktivieren den Filter, Button löscht den Filter wieder.
- Filtermöglichkeit nach Linie.
  - Filter über eine Tabulatoren-Ansicht realisiert.

Diese Funktionen konnten alle umgesetzt werden.

Protokoll Peerreviews

Tabelle 30 Protokoll Peerreviews

Nr.	Ausgangslage Problem	Aktion	Behoben
1	Bug in Navigation zwischen Detailansicht für Reisverschlusslisten, Handmusterlisten und Einzugslisten	Wechseln von der Id auf die jeweiligen Namen, da diese von den Routes erwartet werden	14.12.2016
2	Config.pathUploadExcel wird für zwei Funktionen verwendet.	tempUpload Directory erstellen	14.12.2016
3	CoverPositionImporter viele Strings in Code	Constants extrahieren.	14.12.2016
4	Duplicated Code für Excel Listen Import	ExcelParser.parseExcelFileSingleSheet als generische Funktion verwendet	14.12.2016
5	deleteOldCoverPositions SQL Query kann vereinfacht werden	AbstratEntity.getList verwenden	14.12.2016
6	remLists	Umbenennen für Namensgebung	14.12.2016
7	AndroidMissingMaterial Naming Verwechselbar	Alles auf AndroidMissingMaterial benennen	14.12.2016
8	Filterung der listen. Code zu Komplex und unverständlich	Rewrite der Methoden	14.12.2016
9	setCoverDone und setCoverUndone können zu einer Funktion kombiniert werden	toggleCoverDone erstellt	14.12.2016
10	ConfigController hat überflüssige generische toString Methode.	Entfernen + Auslagern einiger String Constants	14.12.2016
11	Reihenfolge in Config View verwirrend	Neu anordnen	14.12.2016
12	Optimizer hat viele Strings	String Constants auslagern + Umbenennung von unverständlichen Variable Namen.	14.12.2016
13	Imports ungenutzt	Optimieren der Imports	14.12.2016

5.4 Deployment

Unser System besteht aus mehreren Komponenten, dem Server, einer WebApp und zwei verschiedenen Android Applikationen. Diese werden wir nun einzeln betrachten:

Die kleiner der beiden Android Applikationen ist eine Smartphone-App. Sie benötigt keine Datenbank, um Daten zwischen zu speichern werden nur Sharedpreferences genutzt. Die App kommuniziert über zwei Wege, erstens kann sie http-Requests abschicken, zum Beispiel um Fehlmaterial-Meldungen zu machen, oder um die E-Mail-Adresse für Q-Issue-Meldungen abzufragen. Mit dieser Adresse kann die App später E-Mails versenden.

Die grössere App ist für Tablets ausgelegt aber auch sie hat keine lokale Datenbank. Zusätzlich zu den Funktionen der kleinen App kann sie noch die Produktionslisten vom Server abfragen und den Status der Listen zurücksenden.

Die WebApp sollte in allen gängigen Browsern aufrufbar sein und zeigt die Daten und Funktionen vom Server an.

Auf dem Server selbst läuft eine Instanz der MySQL Datenbank, die wir zur Persistierung aller Daten benutzten. Auf dem Server benutzten wir das Play! Framework um unsere Applikation zu hosten.

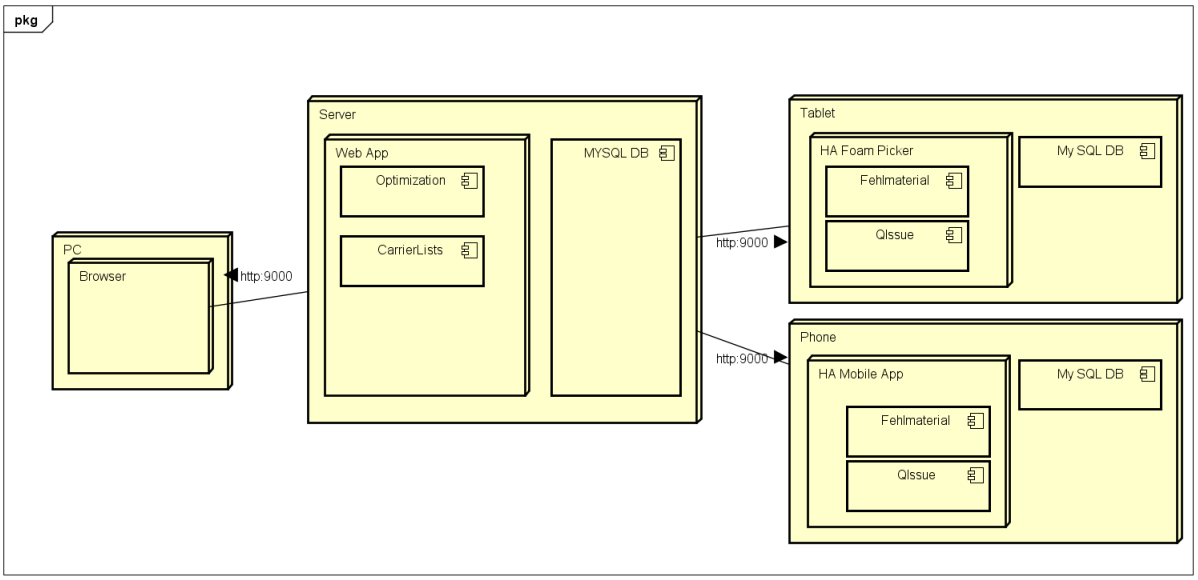


Abbildung 34 Deployment

5.5 Schlussfolgerung und Ausblick

5.5.1 Empfehlungen an BICO

Hier wird nochmals auf den Optimierungsprozess eingegangen, und eine Empfehlung für die Firma abgegeben, wie in Zukunft mit dem Lager gearbeitet werden könnte.

Um eine Empfehlung abzugeben, wird ein Testlauf der Optimierungsalgorithmen mit Folgenden Inputdaten vollzogen.

Die Folgenden in der rechten Grafik aufgelisteten Listen wurden aus Excelfiles importiert:

Nun wird ein neuer Lagerplan generiert mit 5 Kernen pro Sektor und dem Namen «5».

Zum Test wird nun die Liste «Lbl160115-055715-20160118002.xls» gewählt und zuerst der Lagerplan «Standard». Die Wegberechnung wird ausgeführt.

Skipped cores = 88 no Position found in Plan

88 Kerne wurden somit nicht im Lagerplan lokalisiert, dieser Lagerplan wurde uns von der Firma abgegeben.

Anzahl gefahrene Sektoren 1536

Dennoch brauchte die restliche Liste 1536 Sektoren, welche durchfahren wurden für diesen Test.

Lbl160114-055842-20160115002.xls
Lbl160115-055715-20160118002.xls
Lbl160127-055907-20160128002.xls
Lbl160129-064105-20160201002.xls
Lbl160201-060059-20160202002.xls
Lbl160208-055636-20160209002.xls
Lbl160210-055953-20160211002.xls
Lbl160212-055935-20160215002.xls
Lbl160218-055724-20160219002.xls
Lbl160219-055844-20160222002.xls
Lbl160219-060106-20160222002.xls
Lbl160224-060114-20160225002.xls
Lbl160318-055458-20160321002.xls
Lbl160415-055803-20160418002.xls
Lbl160418-055809-20160419002.xls
Lbl160427-060149-20160428002.xls
Lbl160429-060116-20160502002.xls
Lbl160509-060034-20160510002.xls
Lbl160510-060112-20160511002.xls
Lbl160511-055829-20160512002.xls
Lbl160513-060023-20160517002.xls
Lbl160520-055719-20160523002.xls
Lbl160602-060013-20160603002.xls
Lbl160603-055823-20160606002.xls
Lbl160615-060209-20160616002.xls
Lbl160623-055939-20160624002.xls
Lbl160706-055738-20160707002.xls
Lbl160711-055838-20160712002.xls
Lbl160822-060154-20160823002.xls
Lbl160914-055547-20160915002.xls
Lbl160915-055923-20160916002.xls
Lbl161003-060103-20161004002.xls
Lbl161005-055834-20161006002.xls
Lbl161007-060230-20161010002.xls
Lbl161014-055618-20161017002.xls
Lbl161021-055855-20161024002.xls

Abbildung 35 Importierte Listen

Für einen zweiten Versuch wird nun die gleiche Liste genommen, aber der Lagerplan «5». Hier kommen wir zwar auf ein ähnliches Ergebnis:

Anzahl gefahrene Sektoren 1523

Hier ist jedoch zu beachten, dass auch die vorher übersprungenen Kerne hier mitberechnet wurden. Wieviel hier eingespart wurde, können wir ohne die Positionsdaten der Firma nicht sagen.

Hier noch ein Vergleich der beiden Heatmaps

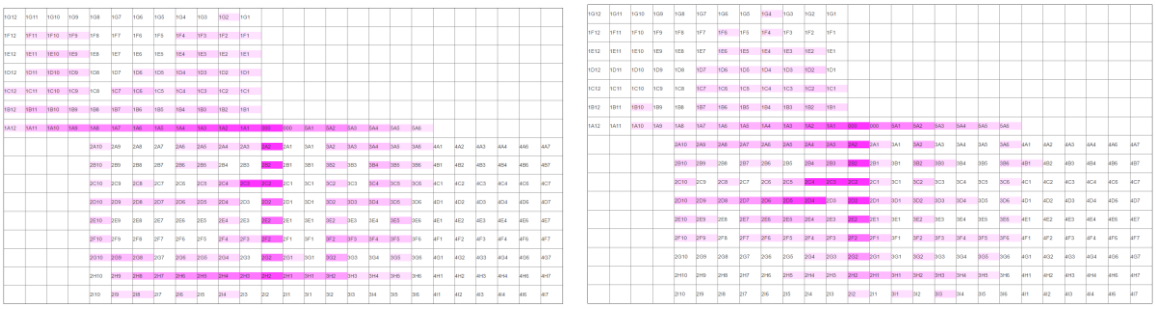


Abbildung 36 Vergleich der Heatmaps

Des Weiteren ist diese Berechnung auch stark abhängig von der Anzahl Kerne, welche auf einer SectorPosition verteilt werden können.

Hier eine Liste mit jeweilig mehr Kernen pro SectorPosition und dem jeweiligen Resultat mit der Liste.

6 Kerne:

Anzahl gefahrene Sektoren	1471

7 Kerne:

Anzahl gefahrene Sektoren	1386

8 Kerne:

Anzahl gefahrene Sektoren	1255

Eine weitere Option welche wir hier Testen wollen ist das neu generieren der Liste, abgestimmt auf den Plan «Standard».

Name der neuen Liste: «Lbl160115-055715-20160118002.xls-St» und Anzahl Kerne pro Palette 10.

Nach dem Lauf der Wegberechnung kommt wieder die Fehlermeldung:

Skipped cores = 88 no Position found in Plan

Aber ein neues Ergebnis:

Anzahl gefahrene Sektoren	1093

Somit kann durch die Optimierung der Reihenfolge doch noch einiges eingespart werden. Von anfangs 1536 Sektoren welche durchfahren werden, zu 1093 Sektoren.

Wieviel bei der Neuverteilung gespart wird, hängt von den fehlenden Daten und der Anzahl Kerne ab, welche neu pro SectorPosition verteilt werden.

Somit ist unser Vorschlag an die Firma zur Optimierung des Lagers der Folgende:

Die Fehlenden Daten sollten aufgearbeitet werden, danach können diese zur Optimierung importiert werden. So lässt sich erstmal feststellen, wieviel Potential bei der Lagerumverteilung noch schlummert.

Zum Verbessern der Reihenfolge, in welcher die Paletten beladen werden, schlagen wir der Firma vor ihren Algorithmus so zu erweitern, dass er die aktuelle Lagerposition mitberücksichtigt.

Dies ist in unseren Augen der wichtigste Aspekt an so einer Optimierung, da dann unabhängig von der Anordnung im Lager, ein möglichst optimaler Weg erzeugt werden kann. So kann immer der kürzeste Weg zu einem Kern in der zu Produzierenden Liste gesucht werden.

Von der neuen Position aus, kann dann das Spiel wiederholt werden, und der nächste Kern gesucht werden. Wenn eine Palette voll ist, so kann wieder beim Lift gestartet werden. Hier muss jedoch noch im Gegensatz zu dem in unserer Arbeit verwendeten Algorithmus, die Stapelregeln für die Matratzen berücksichtigt werden. Jedoch scheint es sich zu lohnen, wenn man die Verbesserung von 1536 auf 1093 betrachtet.

5.6 Glossar

MissingMaterial ist Fehlmaterial, also Material von dem nur noch wenig oder gar nichts mehr vorhanden ist.

QIssue / Qualitätsmangel sind Meldungen die auf Fehler an einem Produkt oder dessen Verpackung hinweisen.

Barcode verweist klar auf die Codes mit der Produktbezeichnung bei Hilding Anders.

QIssue-Mail ist die E-Mail-Adresse, die verwendet wird um Qualitätsmeldungen zu versenden.

WebApp ist die Ansicht der Serverfunktionen in einem Browser.

Zuschnittlager ist eine Exceltabell auf der die verschiedenen Stoffe vom Zuschnitt einem Lagerplatz im Rollenlager zugeteilt sind.

Fehlmaterial via App sind Fehlmaterial-Meldungen, die via Android App eingegangen sind und werden anders behandelt als solche, die aus der WebApp gemeldet werden.

Zuschnitt ist die Ansicht für den Zuschnitt der Boden- und Deckplatten für die Kerne und hat nichts mit dem Kernzuschnitt zu tun.

Admin ist die Seite mit Administrator-Funktionen und kann nur von Benutzern mit Manager-Rolle angesehen und bearbeitet werden.

Konfiguration erlaubt es gewisse Einstellungen am Server vorzunehmen.

Optimierung erlaubt es, die gefahrenen Wege der Gabelstapler zu vergleichen um mögliches Optimierungspotential zu finden.

Download APP stellt die neusten Versionen der Android Apps zur Verfügung.

SectorPosition Gespeicherte Sektorposition.

SectorPlan Gespeicherter Sektorplan (Lagerplan).

5.7 Literaturverzeichnis

- Philippe Naegeli, Raffael Ioannone 2016: Digitalisierung des Produktionsprozesses, Rapperswil.
- Oracle. Java Persistence API. http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html, 2016. [Online; last accessed Dezember-2016]
- Google. Google Volley. https://android.googlesource.com/platform/frameworks/volley, 2016. [Online; last accessed Dezember-2016].
- Google. Google Material Icons. https://design.google.com/icons/ 2016. [Online; last accessed Dezember-2016].
- Java Play Framework. https://www.playframework.com/documentation/2.5.x/JavaHome, 2016. [Online; last accessed Dezember-2016].
- Bootstrap. http://getbootstrap.com/, 2016. [Online; last accessed Dezember-2016].
- Hibernate (JPA). http://hibernate.org/, 2016. [Online; last accessed Dezember-2016].
- JQuery. http://api.jquery.com/, 2016. [Online; last accessed Dezember-2016].

5.8 Abbildungsverzeichnis

Abbildung 1 Termine der Planung 11
Abbildung 2 UseCase Diagramm 14
Abbildung 3 Domainmodell 16
Abbildung 4 Systemübersicht 21
Abbildung 5 Android Applikation im Einsatz 22
Abbildung 6 Rollenzuschnitt Vergleich 24
Abbildung 7 Konfigurationsansicht 25
Abbildung 8 Log-Übersicht 27
Abbildung 9 Lagerplan Datenübersicht 28
Abbildung 10 Daten der Lagerpositionen 29
Abbildung 11 Datenherkunft 29
Abbildung 12 Kernliste in Excel 30
Abbildung 13 Datenspeicherung 30
Abbildung 14 Listenimport 31
Abbildung 15 Plan erstellen aus Excelfile 31
Abbildung 16 Plan generieren 32
Abbildung 17 Sortierung der Kerne 32
Abbildung 18 Sortierung der Distanzen 32
Abbildung 19 Wegberechnung 33
Abbildung 20 Wegberechnung Fehlende Kernpositionen 33
Abbildung 21 CSV 34
Abbildung 22 Heatmap 34
Abbildung 23 Listen erzeugen 35
Abbildung 24 Sortieren nach Distanz zur aktuellen Position 35
Abbildung 25 Stan Analyse Android 39
Abbildung 26 Stan Analyse Server 40
Abbildung 27 Stan Analyse Optimierungsdomain 40
Abbildung 28 Android Code Metrics 41
Abbildung 29 Android Complexity Metrics 42
Abbildung 30 Android Code Metrics 42
Abbildung 31 Android Complexity Metric 42
Abbildung 32 Server Code Metrics 43
Abbildung 33 Server Complexity Metric 44
Abbildung 34 Deployment 52
Abbildung 35 Importierte Listen 53
Abbildung 36 Vergleich der Heatmaps 53

5.9 Tabellenverzeichnis

Tabelle 1 Termine der Planung 11
Tabelle 2 Iterationsplanung 12
Tabelle 3 Properties AndroidMissingMaterial 16
Tabelle 4 Properties Config 17
Tabelle 5 Dependencies PathCoreList 17
Tabelle 6 Properties PathCoreList 17
Tabelle 7 Dependencies PathCorePallet 18
Tabelle 8 Properties PathCorePallet 18
Tabelle 9 Dependencies PathCore 18
Tabelle 10 Properties PathCore 18
Tabelle 11 Dependencies PathSector 18
Tabelle 12 Properties PathSector 18
Tabelle 13 Dependencies SectorPosition 18
Tabelle 14 Properties SectorPosition 19
Tabelle 15 Dependencies SectorConnection 19
Tabelle 16 Properties SectorConnection 19
Tabelle 17 Dependencies SectorPlan 19
Tabelle 18 Properties SectorPlan 19
Tabelle 19 Dependencies SectorPlanPosition 19
Tabelle 20 Verwendete Technologien 21
Tabelle 21 Schnittstellenspezifikation 36
Tabelle 22 Integrationtests Android 45
Tabelle 23 Integrationtests Rollenzuschnitt 46
Tabelle 24 Integrationtests Import 47
Tabelle 25 Integrationtests Listen löschen 47
Tabelle 26 Integrationtests Konfiguration 48
Tabelle 27 Integrationtests Optimierung 48
Tabelle 28 Usabilitytest Fehlmaterial 49
Tabelle 29 Usabilitytest Q-Issue 50
Tabelle 30 Protokoll Peerreviews 51

A Anhang

A.1 Projektdokumente

A.1.1 Anleitungen

Android

Installationsanleitung

Die neuste Android App wird als .apk-Datei vom Administrator in den Ordner public/apk gelegt, damit sie zum Download freigegeben werden kann. Auf der Web-Ansicht kann der Administrator unter Konfiguration die aktuellste Version der App aussuchen, welche dann zum Download freigegeben ist.

Jegliche Nutzer können über die Web-Ansicht die neuste Version mit der Taste «Download App» auf ihr Gerät laden. Dabei spielt es keine Rolle, ob es ein Smartphone, Tablet oder PC ist. Vom PC aus kann die App nicht gestartet werden, sondern muss auf ein Android-Gerät verschoben werden.

Ist die App auf einem Android-Gerät, kann mit der Installation begonnen werden. Beim Öffnen des apk wird man automatisch gefragt, ob man es installieren will. Dazu benötigt man die Berechtigung, Applikationen von fremden Quellen zu installieren, auf welche auch hingewiesen wird, falls man diese nicht hat. Es gilt dort den Haken zu setzten unter Sicherheit, Unbekannte Herkunft.

Hat man die App bereits, kann man im Menu den Knopf «Update App» wählen und es wird überprüft, ob eine neuere Version zur Verfügung steht. Ist dies der Fall, kann diese direkt heruntergeladen werden. Bei der folgenden Installation, müssen auch die nötigen Berechtigungen vorhanden sein.

Funktionsbeschreibung

Qualitäts-Mangel

Sieht man im Lager ein Objekt, dass offensichtlich defekt ist, kann man eine Meldung mit dem Qualitätsmangel an die verantwortliche Person senden. Mit dem Knopf «Barcode scannen», liest man den Barcode des Objekts ein. Der Barcode kann auch mit der Tastatur eingegeben werden. Beim Betreff schreibt man, was das Problem ist, z.B. «Kern schmutzig». Dann drückt man den Kamera-Knopf und macht ein oder mehrere Fotos des Mangels, mit der Taste «Senden» wird ein Email geöffnet, dass eigentlich sendebereit ist. Falls notwendig, kann man noch zusätzliche Informationen in das Mail schreiben, und dieses dann abschicken.

Hat man bereits Angaben in den Feldern, die man nicht mehr braucht, kann man mit der Taste «Zurücksetzten» diese löschen.

Fehlmaterial

Hat es von einer Sorte von Kernen keine mehr an Lager oder ist die minimale Anzahl (Kanban) erreicht, kann man eine Fehlmaterial-Meldung erstellen. Dabei wählt man zuerst den Typ aus, ob gar nichts mehr vorhanden ist (Fehlmaterial) oder ob das Kanban-Minimum erreicht wurde (Nachbestellung). Danach wird mit dem Knopf «Barcode scannen» der Barcode aufgenommen (kann auch mit der Tastatur eingegeben werden), jetzt ist die Meldung bereits fertig. Falls noch zusätzliche Informationen vorhanden sind, kann man ein Termin aussuchen, bis wann man das fehlende Material braucht, wieviel man davon braucht und einen Kommentar dazu abgeben.

Mit der Sendebestätigung wird alles weitergeleitet, mit «Zurücksetzten», kann man die eingetragenen Daten wieder löschen.

Kernlisten

Bei den Kernlisten kann die aktuelle Tagesliste heruntergeladen werden, in welcher alle Paletten zu sehen sind. Diese kann man sich selbst zuteilen oder die «Automatische Zuweisung» einschalten. Beinhaltet eine Palette einen Kern, der zugeschnitten werden muss, wird dies bei der Palette und beim

Kern selbst angezeigt. Die Palette ist eine bestimmte Reihenfolge von Kernen, ist man sich nicht sicher, ob man den richtigen Kern hat, kann man einfach das zugehörige Bild öffnen um sicher zu gehen.

Hat man alle Kerne einer Palette aufgeladen, kann man sich eine neue Palette aussuchen und die alte wird als erledigt markiert.

Web-Applikation

Dieser Knopf öffnet den Browser und stellt eine Verbindung zur Web-Ansicht her. Dort hat man andere Funktionen als in der Android-App, welche aber eher für administrative Arbeiten benutzt werden.

Materialanforderung

Dieser Knopf öffnet den Browser und stellt eine Verbindung zur Ansicht mit allen Fehlmaterialien her. So kann man seine Meldungen auch auf dem Tablet verfolgen, und sieht den aktuellen Status und ob diese bereits wieder verfügbar sind.

Abmelden

Mit der Taste «Abmelden», wird der aktuelle Benutzer ausgeloggt. Danach kommt man auf das Anmeldefenster und kann einen beliebigen Benutzer anmelden.

Update App

«Update App» schaut ob eine neuere Version der App verfügbar ist. Die Anleitung zur Installation ist im ersten Teil gegeben.

Thin App

Die handheld Version der App kann auf den meisten Herkömmlichen Android-Smartphones installiert werden. Die Funktionalität der Tabs Qualitätsmangel und Fehlmaterial wurde grundsätzlich übernommen. Die einzige Änderung wurde beim Fehlmaterial gemacht, dort wird der Name nun nicht mehr von den Logindaten geholt, sondern muss in den Einstellungen gespeichert werden und dann auf der Fehlmaterial-Ansicht aktualisiert werden.

Einstellungen

Die QIssue Emailadresse wird wie auf der Tablet-App vom Server geholt, sollte dieser nicht verfügbar sein, muss beim Öffnen der effektiven Email die Adresse von Hand eingegeben werden.

Mit Web-App öffnen wird der Browser des Smartphones geöffnet und eine Verbindung zur Web-Applikation hergestellt. Dort kann auch die neuste Version der App heruntergeladen werden.

Funktionen Web-App

Produktionslisten

Kernlisten

Hier kann man eine Tagesliste aussuchen welche dann alle Paletten dieses Tages anzeigt. Innerhalb der Paletten werden die einzelnen Kerne mit Bild und Zuschnitts Hinweis angezeigt.

Handmusterlisten

Hier werden in einer Tagesliste nur die Handmuster-Linien und die einzelnen Zuschnitte angezeigt.

Reissverschlusslisten

Hier werden die benötigten Reissverschlüsse für die Paletten einer Tagesliste angezeigt.

Einzugslisten

Hier sieht man, wie viele Matratzen der Tagesproduktion schon abgeschlossen sind, und welche Kerne noch fehlen.

Gesamtlisten

Hier sieht man alle einzelnen Aufträge eines Tages, mit allen vorhandenen Angaben.

Management

Dateiimport

Excel importieren

In der Konfiguration kann ein Ordner Angegeben werden: Admin → Konfiguration → Excel Upload Ordner

Aus diesem Ordner werden hier alle .xls und alle .xslsx Files angezeigt und können Auf Knopfdruck importiert werden.

Neue Liste generieren

Wurde ein Excel manuell hochgeladen, kann hier mit einem ausgewählten Algorithmus eine Palettenliste erstellt werden. Dieses Feature sollte aber nur in Ausnahmefällen benutzt werden, da der Autoimport eigentlich alles richtig erledigen sollte.

Excel hochladen

Hier können Tageslisten aus dem Movex hochgeladen werden. Dies kann aber auch automatisch gemacht werden, wenn man es unter «Konfiguration» (Administratorberechtigung nötig) aktiviert. Beim Importieren kann man auswählen ob man die Liste mit oder ohne Algorithmus hochladen will.

Bilder und Kerndaten laden

Auf Knopfdruck, werden aus den angegebenen Verzeichnissen die neusten Kerndaten und die passenden Bilder dazu auf den Server geladen.

Zuschnittlager hochladen

Um die Position eines Stoffes beim Zuschnitt anzeigen zu können sollte immer die aktuellste Version des Zuschnittlagers hochgeladen werden.

Administration Listen

Hier werden alle Tageslisten mit ihrem Status angezeigt. Man kann sie zur Bearbeitung freigeben oder noch einmal löschen, um eine aktualisierte Version hochzuladen.

Administration Kerne

Hier sieht man alle verfügbaren Kernmodelle und in welchem Sektor sie liegen. Man kann Kernmodelle und Sektoren hinzufügen und Kerne editieren.

Administration RV

Hier werden alle verfügbaren Reissverschlüsse gezeigt und man kann zusätzlich eine Markierung setzten, falls ein Reissverschluss nicht mehr gerüstet werden muss.

Priorisierung

Um gewisse Paletten zu priorisieren kann man in dieser Ansicht jeder Palette eine Priorität von A-C geben.

Produktübersicht

Hier werden alle Aufträge eines Tages angezeigt.

Auditlog

Hier kann man den Auditlog herunterladen.

KPI

Bei kompletter Auftragsabwicklung über die Web-App, wird eine Statistik erstellt welche man hier einsehen kann.

Materialanforderungen

Hier werden die Materialanforderungen angezeigt.

Fehlmaterial

Unter Fehlmaterial gibt es zwei verschiedene Sorten von Fehlmaterial. Zum einen die Materialien von denen man weiss, dass sie Fehlen und man sie in der Web-Ansicht zum Ausstand hinzufügt, erscheinen in der oberen Liste. Diese Liste kann auch exportiert werden, falls notwendig

Zuschnitt

Hier werden die benötigten Zuschnitte angezeigt. Zuoberst kann man aussuchen, welche Linie man sehen will. Mit einem Klick auf die Bezugsnummer, wird die Liste gefiltert und es werden nur noch die Aufträge angezeigt, welche die gleiche Bezugsnummer haben. Mit dem Knopf «Filter zurücksetzten» werden wieder alle Zuschnitte angezeigt. Hat man einen Zuschnitt gemacht, kann er als «Erledigt» markiert werden. Um eine bessere Übersicht über die noch vorhandene Arbeit zu haben, kann man die erledigten Zuschnitte ausblenden.

Die Tabelle mit den Aufträgen kann beliebig sortiert werden indem man auf die zwei Pfeile neben dem Spaltentitel klickt.

Admin

Dieser Teil kann nur von Benutzern mit Manager-Rolle gesehen werden.

Benutzerübersicht

Hier werden alle zugelassenen Benutzer und ihre Rolle angezeigt. Neue Benutzer können sehr einfach erstellt werden, da man nur einen Namen, eine E-Mail und ein Passwort braucht. Benutzerkonten, die nicht mehr gebraucht werden können einfach gelöscht werden.

Konfiguration

Hier können diverse Einstellungen gemacht werden, die das Verhalten des Servers beeinflussen. Änderungen sollten mit dem «Aktualisieren» Knopf bestätigt werden.

Q-Issue E-Mail: Setzt, an wen eine Qualitäts-Meldung über die Android-Applikation gesendet wird.

Excel Upload Ordner: Beim manuellen Upload bekommt man eine Auswahl aller, in diesem Ordner gespeicherten Dateien, um diese manuell zu importieren.

CoreType Importer Ordner: Beim Laden der Kerndaten und Bilder wird in diesem Ordner nach Kerndaten gesucht.

Import Bilder Ordner: Beim Laden der Kerndaten und Bilder wird in diesem Ordner nach Bildern der Kerne gesucht.

Auto-Import Excel Ordern: Dieser Pfad sollte auf den Ordner zeigen, in dem die Tageslisten gespeichert werden. Von hier aus werden sie automatisch in den Server importiert.

Android APK Ordner: Hier sucht der Server nach .apk-Dateien, die er weiter unten, unter «Android APK Name» zur Auswahl stellt.

Android APK Name: Hier wird gewählt, welche Version der App beim Klicken auf «Download App» heruntergeladen werden soll.

Logs

Dieser Tab zeigt die Logs des Servers an, um in einem Problemfall zusätzliche Informationen zu haben.

Optimierung

Hier können im oberen Teil Tageslisten, zu Testzwecken, hochgeladen werden.

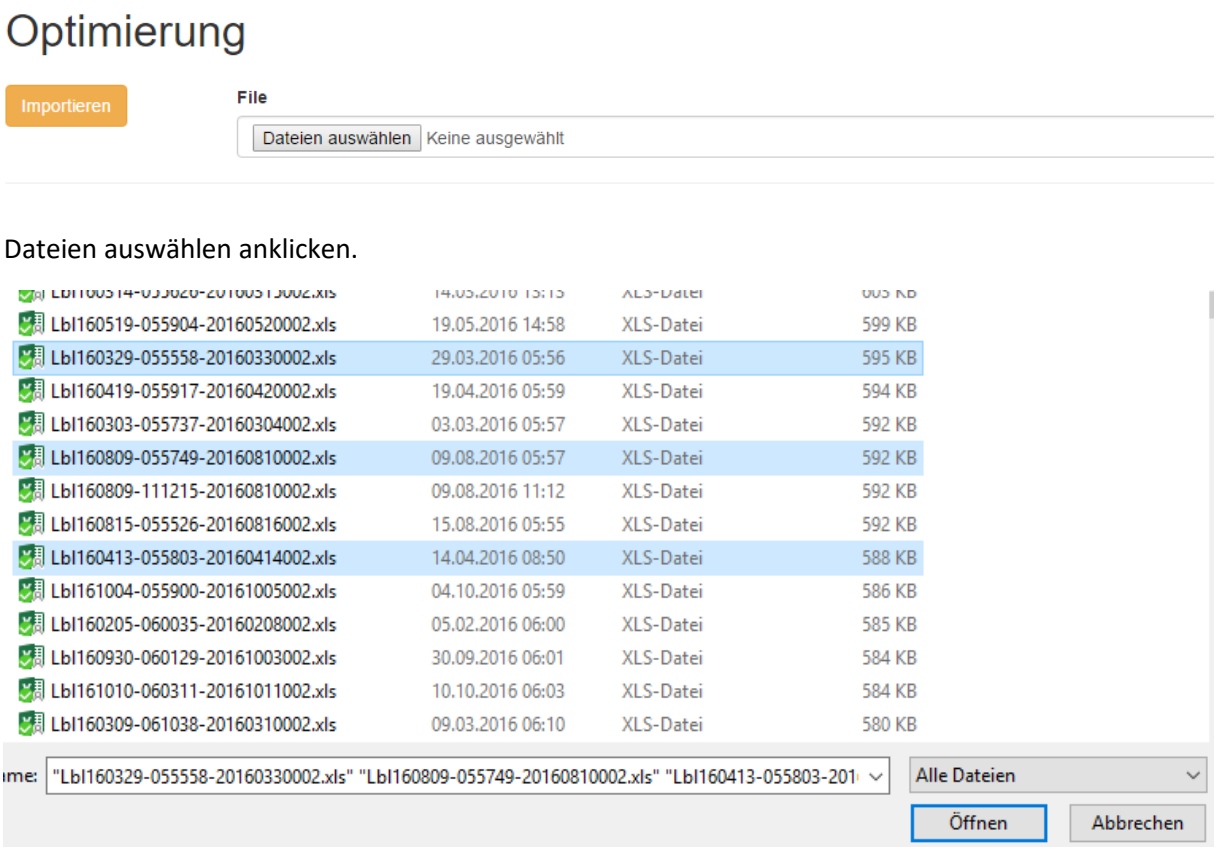
Danach können die gewünschten Listen ausgewählt und die Optimierung gestartet werden. Als Resultat erhält man die vom Gabelstapler gefahrenen Sektoren. So kann man die Listen unter einander Vergleichen.

Download App

Dieser Knopf lädt die aktuellste Version der Android-App herunter.

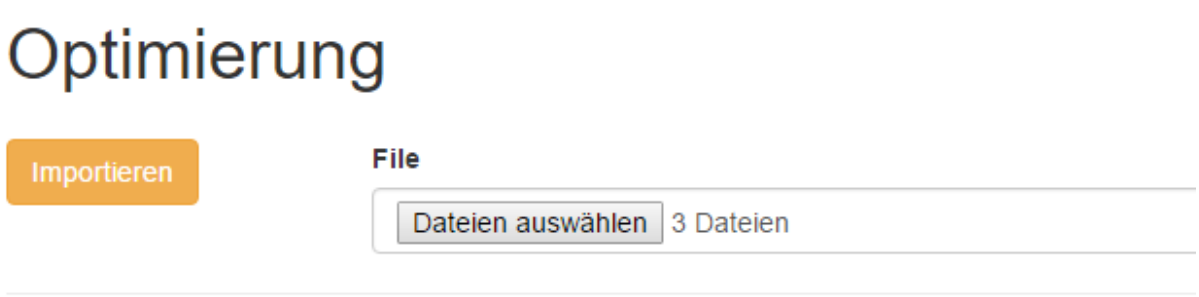
Optimierung

Zuerst müssen die alten Listen geladen werden.

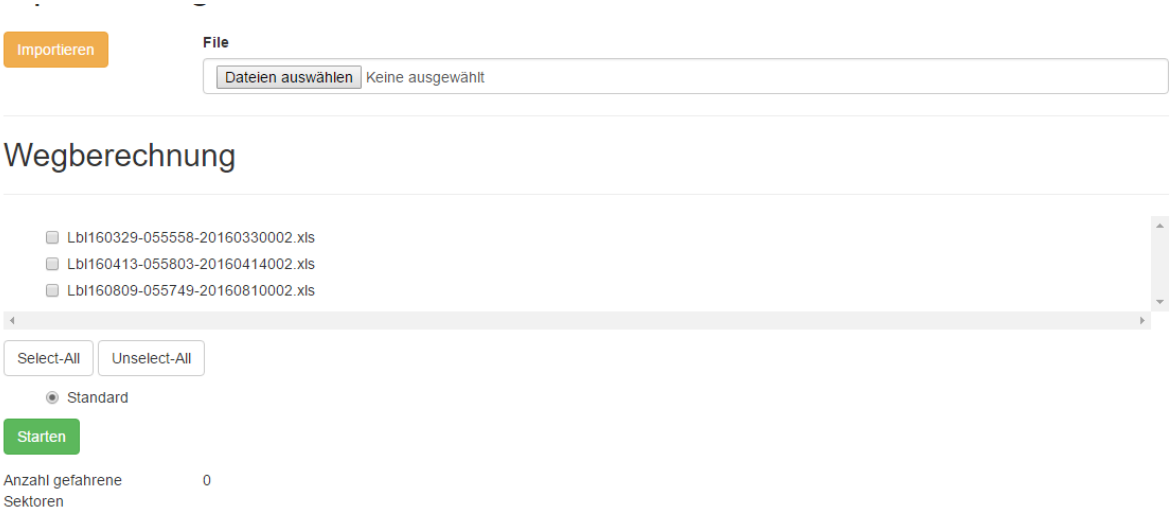


Dateien auswählen anklicken.

Nun die zu importierenden Listen auswählen. Es können mehrere mit Ctrl und Shift Taste ausgewählt werden. Dann auf Öffnen klicken. (Achtung es sollten nicht mehr als ca. 20 Dateien auf einmal geladen werden, weil sonst der Request an den Server zu lange dauert und die Datenbank die Verbindung abbricht. Diesen Fehler konnten wir leider hier nicht mehr beheben. Die Aktion kann ber für mehr Listen einfach wiederholt werden.)



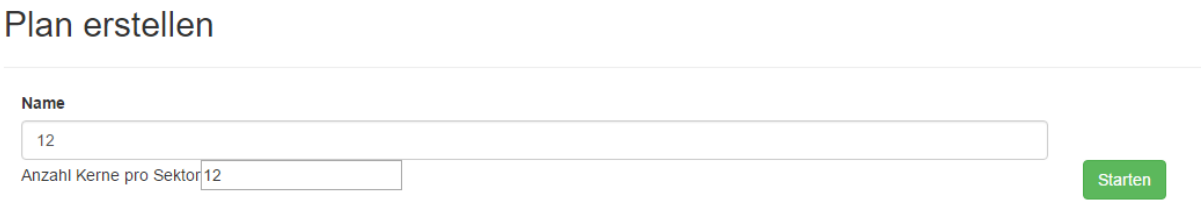
Importieren anklicken.



Nun sieht man die Importierten Listen in dieser Übersicht.

Oben sind die Listen und darunter direkt die Lagerpläne.

Die Lagerpläne kann man auf zwei Arten erzeugen:



Einerseits kann man Anhand der Häufigkeit der Kerne in den oben importierten Listen (nur diese werden berücksichtigt) generieren lassen.

Hierzu einen Namen eingeben und die Anzahl Kerne welche auf eine SectorPosition passen (mit SectorPosition ist zB 1A12 vom Lagerplan gemeint)

Dann starten klicken.

Andererseits kann man ein File mit den Lagerpositionen importieren, welche man wünscht. Hier könnte man auch noch weitere «Experimente» aus Excelfiles probieren.

Hier gilt es einige Dinge zu berücksichtigen, damit der Import nicht fehlschlägt.

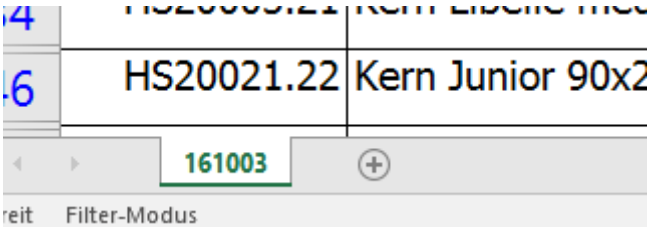
	A	B	E	V	W
1	ArtNr	ArtBez	LB	Sektor[1-5]	KoordinatenA1
2	HS20000.61	Kern Baby 60x120x8	5	3	D5
3	HS20000.63	Kern Baby 70x140x8	20	3	D5
7	HS20001.22	Monokern 90x200x12	21	1	B3

1. Vier Spalten sind nötig, in welchen die Informationen wie oben Beispielhaft angezeigt sind. Artikel-NR, Artikel Bezeichnung, Sektor und Sektorkoordinaten
2. Alle Spalten sind in der 1. Reihe des Excel-Sheets mit einem Spaltentitel versehen, siehe oben blau hinterlegt.
3. In der Titelreihe darf es keine Zeilenumbrüche haben, weil diese nicht im Webinterface mit eingegeben werden können.

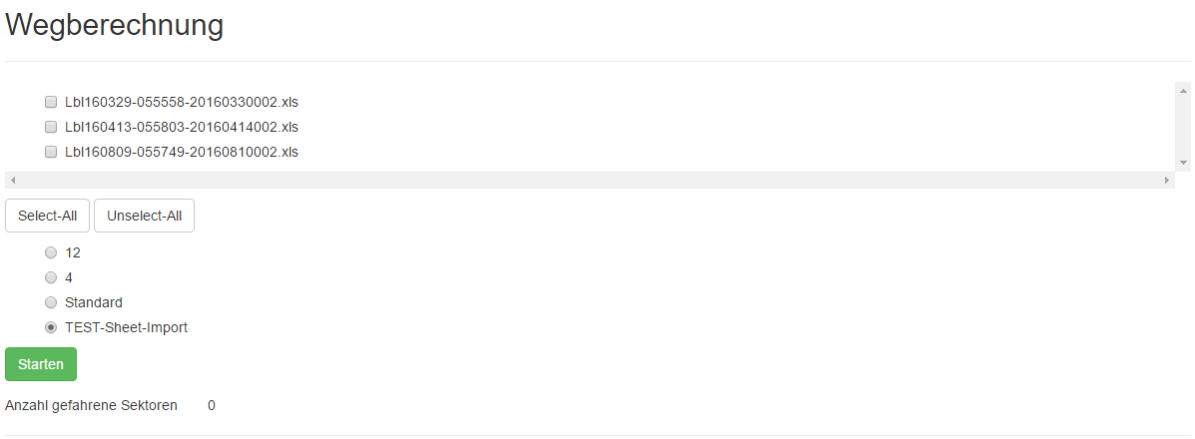
Nun müssen die Inputfelder ausgefüllt werden. Wieder einen Namen eingeben.

Dann müssen die Spaltentitel aus dem Excel Sheet übernommen werden und in die Felder eingetragen werden.

Dann auch noch den Sheet-Name von Excel übernehmen.



Die Datei unten auswählen und Importieren klicken

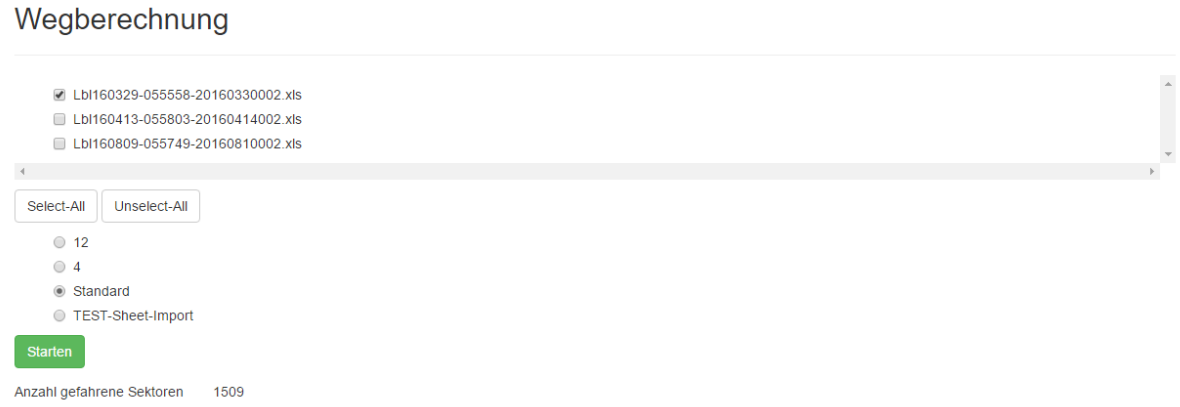


Oben wird nun auch der neu Importierte Plan angezeigt

Nun kann man die Berechnung starten.

Dazu kann man eine oder mehrere Listen auswählen und unten einen der Lagerpläne.

Dann Starten klicken.



Ganz unten kann das ganze wieder rückgängig gemacht werden.

resetPlan löscht alle generierten Pläne

resetLists löscht alle importierten Listen.

Installation

Die Server-Infrastruktur wird als *.ZIP File ausgeliefert.

In diesem ZIP sind alle benötigten Dateien, für das Laufenlassen, und das Konfigurieren der Serverapplikation.

Was wird benötigt:

Hier eine Liste mit Dingen, die benötigt werden, bevor mit der Konfiguration und Installation begonnen werden kann:

- ZIP File

- MySQL Datenbank mit DB-Nutzernamen und DB-Passwort

Aufsetzten MySQL Datenbank

Da die Datenbank funktioniert und eigentlich nicht verändert werden muss, werden wir diesen Teil kurzhalten.

Zum Aufsetzten der Datenbank mit MySQL Workbench gibt es diverse Anleitungen im Internet, hier eine Möglichkeit:

https://dev.mysql.com/doc/workbench/en/wb-home.html auf Englisch.

Installationsschritte:

Zuerst muss ein Ordner ausgesucht werden, in welchem die Schlussendlichen Dateien gespeichert werden sollen.

Für unsere Anleitung verwenden wir den Pfad: C:\BicoServer (Muss eventuell neu erstellt werden).

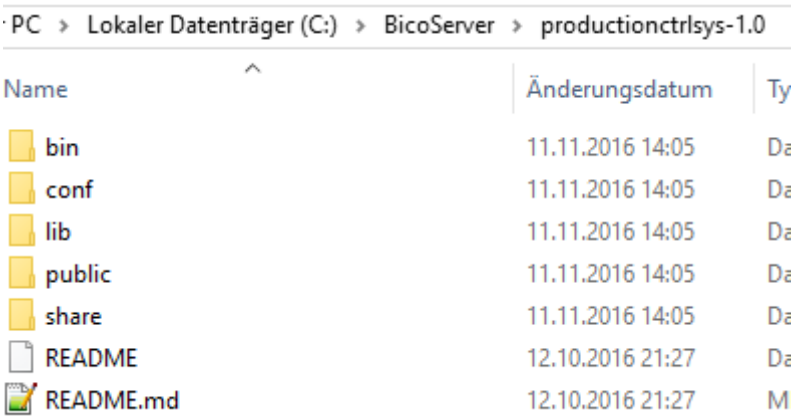
Dann entpacken wir das ZIP File in diesem Ordner.

Datenbank anpassen

Nun sollte in dem Ordner ein weiterer Ordner mit dem Namen «productionctrlsys-X.X» liegen wobei X.X die Versionsnummer ist. In unserem Beispiel «productionctrlsys-1.0»

Den Ordner productionctrlsys-1.0 öffnen.

Nun sollte eine Liste Weiterer Ordner zu sehen sein.



Im Ordner «conf» können nun einige Einstellungen vorgenommen werden.

Wenn man «conf» öffnet sieht man eine Datei «production.conf»

In der Datei production.conf können Einstellungen zur Datenbank und Zum Benutzerkonto angegeben werden, welches beim ersten Start der Applikation erstellt werden soll.

```
include "application"

db.default {
  driver = com.mysql.jdbc.Driver
  url = "jdbc:mysql://localhost/hsr"
  username = HSR
  password = hsrweb_16
  jndiName = MySQLDS
}

bico.logpath=./logs/

bico.db.defaultuser {
  user = manager
  password = "12345"
  email = "test@webapp.url"
}

bico.import.scheduler{
  time_unit = MINUTES
  time_interval = 15.0
}
```

Das Format für die Datenbank URL ist wie folgt:

Jdbc:mysql://{HOST}/{DATENBANK_NAME}

Datenbank Nutzername und Passwort können daruntergesetzt werden.

Im Beispiel also existiert auf dem Computer(localhost) eine Datenbank mit dem namen «hsr»

Es existiert ein DB-Benutzer mit dem Namen «HSR» und dem Passwort «hsrweb_16»

Unten kann der Benutzer konfiguriert werden, welcher beim ersten starten der Applikation erstellt wird. Im Beispiel «manager» mit Passwort «12345» und email «test@webapp.url».

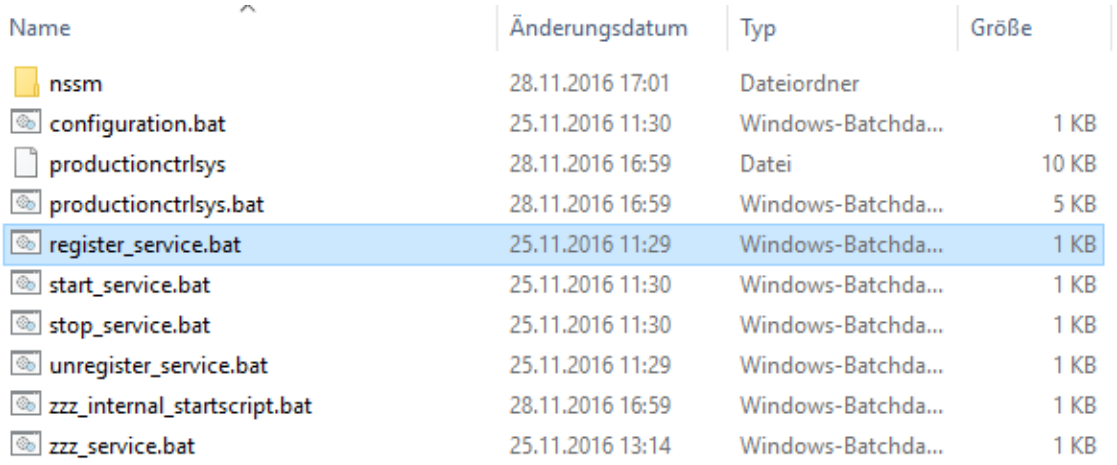
Unter «bico.logpath» kann man Angeben, in welchem Ordner Log-Dateien gespeichert werden.

Services starten

Nun wechselt man wieder zurück in das Hauptverzeichnis.

Im Beispiel unter dem Pfad C:\BicoServer\productionctrlsys-1.0\

Dann wechselt man in das Verzeichnis «bin» Hier sind die ganzen Verwaltungsskripts.



Name	Änderungsdatum	Typ	Größe
nssm	28.11.2016 17:01	Dateiordner	
configuration.bat	25.11.2016 11:30	Windows-Batchda...	1 KB
productionctrlsys	28.11.2016 16:59	Datei	10 KB
productionctrlsys.bat	28.11.2016 16:59	Windows-Batchda...	5 KB
register_service.bat	25.11.2016 11:29	Windows-Batchda...	1 KB
start_service.bat	25.11.2016 11:30	Windows-Batchda...	1 KB
stop_service.bat	25.11.2016 11:30	Windows-Batchda...	1 KB
unregister_service.bat	25.11.2016 11:29	Windows-Batchda...	1 KB
zzz_internal_startscript.bat	28.11.2016 16:59	Windows-Batchda...	1 KB
zzz_service.bat	25.11.2016 13:14	Windows-Batchda...	1 KB

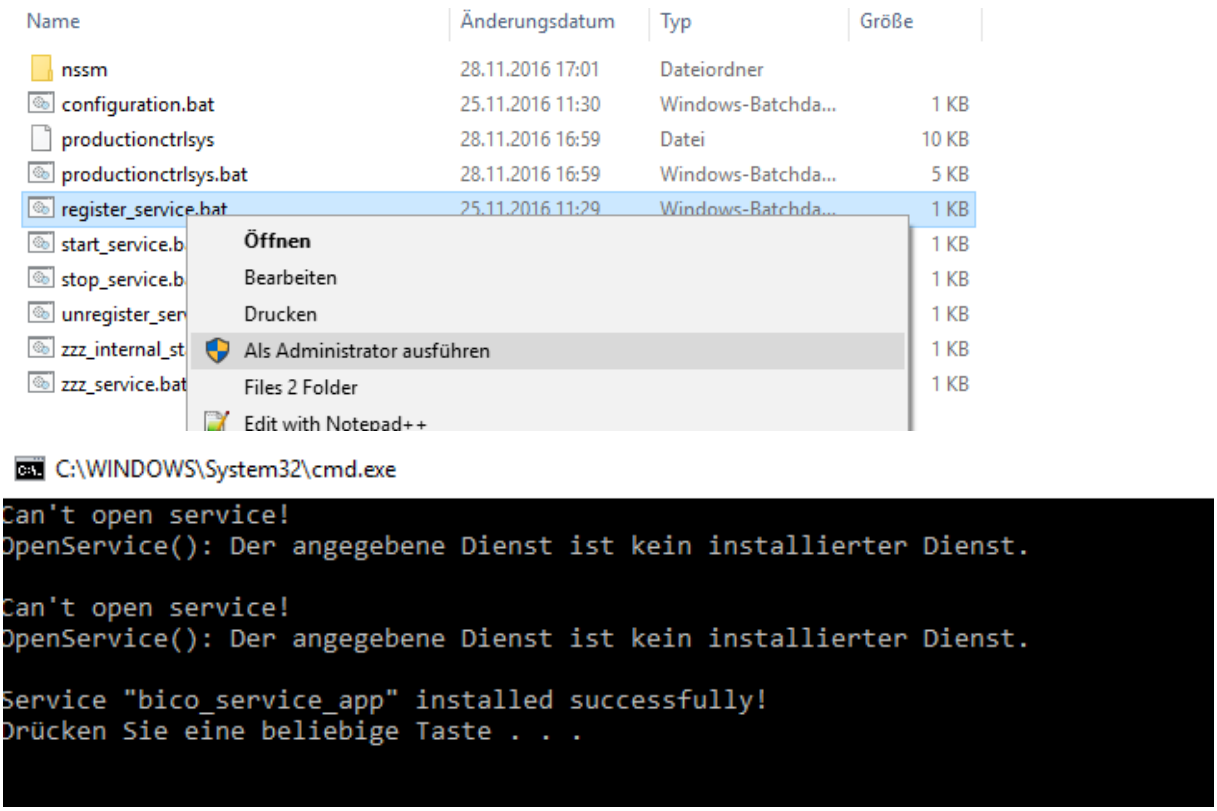
Wichtig sind hier vor allem die Windows Service Scripts:

- «start_service.bar» Startet den Registrierten Windows-Service
- «stop_service.bat» Stopt den Registrierten Windows Service
- «register_service.bat» Registriert die Applikation als Windows Service
- «unregister_service.bat» Deinstalliert den Registrierten Windows Service

WARNUNG! Diese Scripts müssen Alle als Administrator ausgeführt werden.

Zur Installation also:

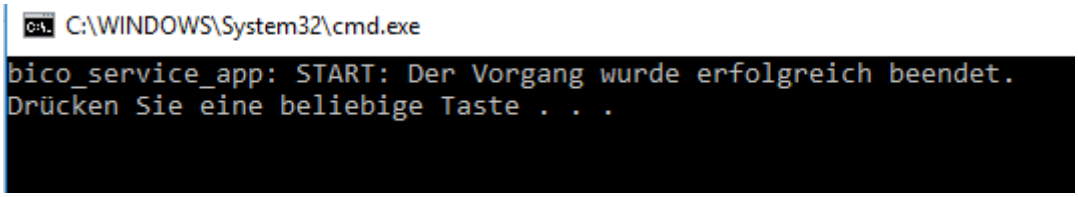
register_service.bat Als Adminsitrator ausführen:



Die Zwei ersten Fehlermeldungen werden angezeigt, da zuerst falls schon ein alter bestehender Service vorhanden ist, dieser gestoppt und entfernt werden muss.

Danach start_service.bat als Administrator ausführen:

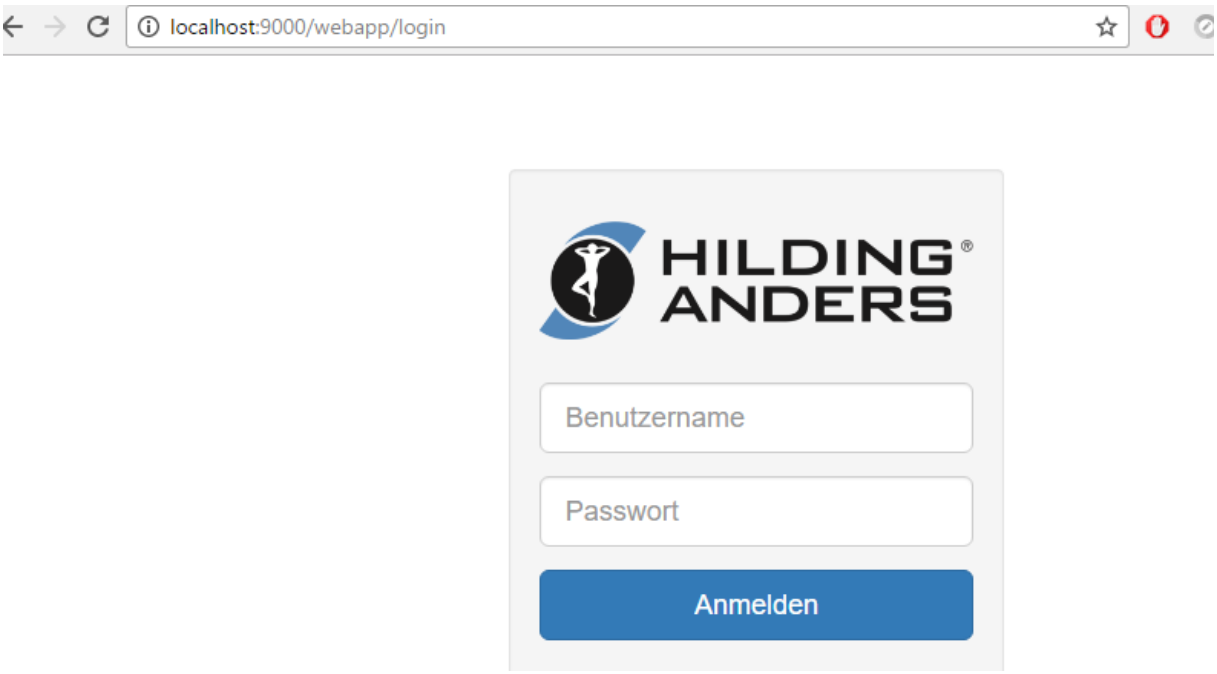
-Rechtsklick wie Oben gezeigt.



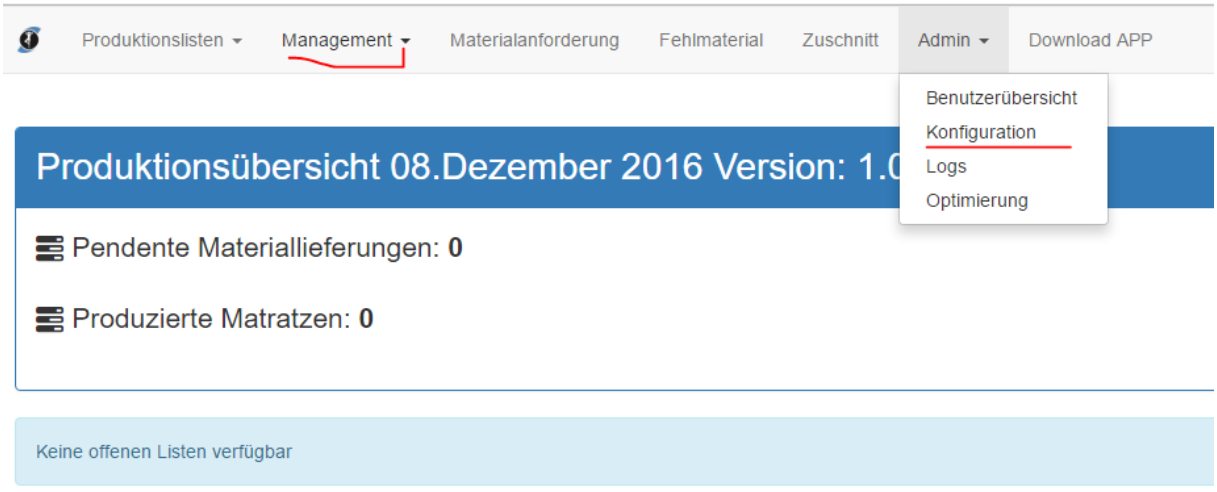
Nach dieser Meldung ist der Service gestartet.

Funktion überprüfen

Um zu testen, dass es läuft, die Seite in einem Browser öffnen:



Passwort und Benutzernamen im Beispiel aus der production.conf «manager» und «12345»



Wenn man angemeldet ist, kann unter Benutzerübersicht der Default Benutzer geändert werden. Dies sollte gemacht werden, damit das Passwort nicht auch in der Konfigurationsdatei sichtbar hinterlegt ist.

Unter Management-> Datenimport, im Tab «Bilder und Kerndaten laden», können die Basisdaten der Applikation in die Datenbank geladen werden. (Dauert einen Moment)

Die Ordnerpfade zu den Imports können unter Konfiguration gesetzt werden.

A.1.2 Wandel der Aufgabendefinition

Ausgeschriebene Aufgabe

Titel:	Produktionsoptimierungen bei BICO
Verantwortlicher:	Keller, Daniel
Betreuer:	Keller, Daniel
Gegenleser:	[nicht definiert]
Experte:	[nicht definiert]
Industriepartner:	Hilding Anders, Schänis (Bico)
Ausschreibung:	THEMA 1 "Simulation der Produktionskette bei BICO": Modellierung der Matratzen-Produktion bei Bico mit Losgrösse 1: Bestellungseingang, Vorräte (Matratzen an Lager), Produktionsstrassen, Halbfertigprodukte, Lagerplätze, Arbeitsplätze, Produktionspuffer/Zwischenlager Aufzeigen von Optimierungsmöglichkeiten mit Hilfe von Simulationen: wo kann man Zeit einsparen? z.B. dauert die Suche des Matratzenkerns im Lager zu lange? Verbesserung durch andere Lagerstrategien? wo laufen die Zwischenpuffer schnell voll? Bei dieser Arbeit gibt es viel Simulation/Modellierung mit Simio und wenig Programmierung. Es ist auch wichtig, den gesamten Produktionsprozess zusammen mit dem Kunden richtig zu erfassen und mit der Simulationsumgebung schlüssige Aussagen zu machen. THEMA 2 "Android App in der Produktionsplanung" Hier ist die Herausforderung, eine bestehende Android App (ex BA HSR, gut dokumentiert) zu erweitern. Zwei Punkte würden dem Kunden grossen Nutzen bringen: Materialabruf implementieren (Push Notifications); Wegoptimierung Gabelstapler (Algorithmen Java/Groovy); wenn Interesse vorhanden: HW/Beacons zur Positionsbestimmung innerhalb des Lagers einsetzen, damit die Wegoptimierung besser funktioniert.
Voraussetzungen:	THEMA 1: Flair für Simulationen, Abstraktionsvermögen, Freude am Kontakt mit den Bico Mitarbeitenden in der Produktion und in der Planung. THEMA 2: Programmierung in Java, Android, Play Framework, MySQL/SQLite, evtl. HW/Beacons zur Positionsbestimmung indoor.

Version 1

Scope vordefiniert

Support

Bestehende Web-App wieder zum Laufen bringen mit Hilfe von Philipp Nägeli und Logs, damit der normale Arbeitsbetrieb gewährleistet ist.

Erweiterung

Die Android App mit zwei Views erweitern, damit UC1 und UC2 erfüllt werden können.

UC1

Sieht ein Lagermitarbeiter ein Fehler am Material kann er davon, mit dem Tablet, ein Foto machen und ein Kommentar dazu eingeben. Diese werden dann entweder per Email an die zuständige Person geschickt oder auf dem Server eingetragen, damit die zuständige Person dies abrufen kann.

UC2

Nimmt ein Lagermitarbeiter von einem Kern den Letzten (oder zweitletzten, Kanban!) kann er mit dem Tablet den Barcode dieses Typs einlesen und im System wird dieser Typ dann als "knapp an Lager bezeichnet" und in der entsprechenden Ansicht angezeigt.

Entwicklung

Wir entwickeln ein statisches Simulationsprogramm in Java und versuchen mit den letztjährigen Bestelldaten nachzuweisen, dass im Bereich der Kernbereitstellung Optimierungspotential vorhanden ist.

Version 2

Scope vordefiniert

Support

Bestehende Web-App wieder zum Laufen bringen mit Hilfe von Philipp Nägeli und Logs, damit der normale Arbeitsbetrieb gewährleistet ist.

Erweiterung

Die Android App mit zwei Views erweitern, damit UC1 und UC2 erfüllt werden können.

UC1

Sieht ein Lagermitarbeiter ein Fehler am Material kann er davon, mit dem Tablet, ein Foto machen und ein Kommentar dazu eingeben. Diese werden dann entweder per Email an die zuständige Person geschickt oder auf dem Server eingetragen, damit die zuständige Person dies abrufen kann.

UC2

Nimmt ein Lagermitarbeiter von einem Kern den Letzten (oder zweitletzten, Kanban!) kann er mit dem Tablet den Barcode dieses Typs einlesen und im System wird dieser Typ dann als "knapp an Lager bezeichnet" und in der entsprechenden Ansicht angezeigt.

Entwicklung

Die grundlegende Ausgangslage für spätere Optimierungen an der Verwaltung des Matratzenkernlagers, ist die Berechnung und Annäherung der momentan gebrauchten Zeit aufgrund der alten Daten welche aus den letzten Jahren vorliegen.

Ziel dieses Arbeitsteiles ist es anhand der alten Daten, die Staplerfahrten nachzustellen und danach für jeden der jeweiligen aufgezeichneten Arbeitsschritte eine messbare Zeit zu definieren.

Hierfür müssen am Datenmodell einige Erweiterungen angebracht werden, welche dazu dienen Die Fahrten und die benötigten Zeiten zu modellieren. Dafür müssen die Alten Daten der Matratzenkernpalletten in die Applikation eingelesen werden und auf die Entsprechend Vorbereiteten Datenmodelle abgeglichen werden, um schlussendlich eine Vergleichbare Zeitberechnung aufstellen zu können.

(Mögliche weiter Schritte beinhalten dann, dass Nutzen dieses Zeitmodels und das Umrangieren der bestehenden Matratzenkernpalletten, um so eine signifikante Zeiteinsparung aufgrund dieses Modells fertigen zu können.)

Version 3

Scope vordefiniert

Support

Bestehende Web-App wieder zum Laufen bringen mit Hilfe von Philipp Nägeli und Logs, damit der normale Arbeitsbetrieb gewährleistet ist.

Es werden noch Erweiterungen am Server vorgenommen um Arbeitsprozesse zu erleichtern.

-Die Web-App soll den Import von .xls und .xlsx Dateien unterstützen

-Die Listen sollen automatisch geladen werden und können solange noch niemand daran gearbeitet hat auch aktualisiert werden. Wird versucht zu aktualisieren, während oder nachdem jemand daran arbeitete, wird eine Fehlermeldung ausgegeben und eine Email an admin.ch@hildinganders.com. Das automatische laden hat ein Intervall von Def=15 min. Ein File sollte jedoch niemals zwei Mal in den Server geladen werden.

-Die Freigabe der Listen von einem Administrativmitarbeiter kann entfernt werden, da dies den Prozess nur unnötig verlangsamt.

-Falls der Server neu gestartet wird muss unsere Web-App sich automatisch neu starten

-LogFile in konfigurierbarem ShareFolder führen

-Installationsinstruktion mit Testszenarien um sicher zu gehen, dass System läuft oder gezügelt werden kann => EXE für automatische Installation oder klare Weisungen

-KernenUpdate ebenfalls als Ergänzung mit neuen Date

-Bilder ergänzen austauschen => ID und FiledDatacode prüfen und updaten falls neuer job wird mit BilderAktualiseren ausgeführt

Erweiterung

-Beim gescheiterten Loginversuch sollte die Meldung sagen, ob die Credentials nicht stimmen oder Serverprobleme vorliegen.

Die Android App mit zwei Views erweitern, damit UC1 und UC2 erfüllt werden können.

UC1

Sieht ein Lagermitarbeiter ein Fehler am Material kann er davon, mit dem Tablet, ein Foto machen und ein Kommentar dazu eingeben. Diese werden dann per Email an eine vordefinierte Mailbox geschickt damit die zuständige Person diese abrufen kann.

UC2

Nimmt ein Lagermitarbeiter von einem Kern den Letzten (oder zweitletzten, Kanban!) kann er mit dem Tablet den Barcode dieses Typs einlesen und im System wird dieser Typ dann als "knapp an Lager bezeichnet" und in der entsprechenden Ansicht angezeigt. Zusätzlich sollte der Mitarbeiter die Möglichkeit haben, einen Kommentar einzufügen (als Failsave falls Barcode nicht erkannt wird) mit seinem Namen für Rückmeldungen und Fragen, ein Zahlenfeld (wie viele fehlen) und ein Datum bis wann man Nachschub braucht. Ebenfalls sollte man angeben können, ob gar nichts mehr vorhanden ist, oder ob die Meldegrenze (Kanban) erreicht wurde.

Entwicklung

Die grundlegende Ausgangslage für spätere Optimierungen an der Verwaltung des Matratzenkernlagers, ist die Berechnung und Annäherung der momentan gebrauchten Zeit aufgrund der alten Daten welche aus den letzten Jahren vorliegen.

Ziel dieses Arbeitsteiles ist es anhand der alten Daten, die Staplerfahrten nachzustellen und danach für jeden der jeweiligen aufgezeichneten Arbeitsschritte eine messbare Zeit zu definieren.

Hierfür müssen am Datenmodell einige Erweiterungen angebracht werden, welche dazu dienen Die Fahrten und die benötigten Zeiten zu modellieren. Dafür müssen die alten Daten der Matratzenkernpalletten in die Applikation eingelesen werden und auf die Entsprechend Vorbereiteten Datenmodelle abgeglichen werden, um schlussendlich eine Vergleichbare Zeitberechnung aufstellen zu können.

Das Nutzenpotenzial des Wegoptimierer Algorithmus versus optimiertes Lager sollte aufgezeigt werden, evtl. mit Empfehlung, wie das Lager umgestellt werden muss.

Version 4

Aufgabendefinition

1 Server

1.1 Web-App

Die Web-App muss wieder Laufen und sollte sich nach einem Beenden des Servers auch selbst wieder auf starten um den Arbeitsbetrieb zu gewährleisten.

Die Logs der Web-App sollten für berechtigte Benutzer leicht zugreifbar sein.

Die Matratzenkernbilder sollten sich automatisch aktualisieren, indem der

1.2 Android

Beim Login auf der Android App kommt zurzeit nur eine generelle Meldung, wenn das Login scheiterte. Diese Meldung sollte in Zukunft aussagen, ob das Problem bei den Benutzerdaten oder beim Server liegt.

2 Fehlmaterial

UC1: Ein Mitarbeiter sieht, dass an einem Lagerplatz kein Kern mehr vorhanden ist oder das Kanban-Minimum wurde erreicht. Mit der Android App scannt er den Barcode (Barcode an Säule auch wenn kein Material mehr vorhanden ist), wählt ob es eine Kanban-Nachbestellung oder Fehlmaterial ist und gibt seinen Namen ein. Diese Daten kann er absenden und sie werden in der Web-App unter Fehlmaterial angezeigt.

UC1.1: Ein Gabelstaplerfahrer kann angeben, wie viele Kerne ihm fehlen um seinen aktuellen Auftrag zu beenden.

UC1.2: Ein Gabelstaplerfahrer kann einen Termin eingeben, bis wann er eine Nachlieferung braucht.

UC1.3: Ein Mitarbeiter hat die Möglichkeit einen Kommentar zur Meldung zu schreiben für zusätzliche Informationen.

3 Q-Issue

UC2: Ein Mitarbeiter sieht defektes Material und kann mit der Android-App eine Meldung dazu erstellen. Damit möglichst gut auf das Problem reagiert werden kann, macht der Mitarbeiter ein Foto des Defekts, scannt den Barcode des Artikels und schreibt einen Kommentar dazu. Die Meldung geht per Email direkt an eine Mailbox für Materialfehler.

4 Rollenwechsel

UC3: Die Mitarbeiter beim Stoffzuschnitt haben die Möglichkeit in der Web-App verschiedene Sortierungen der aktuellen Aufträge anzusehen. So können sie nicht nur die Kernliste der Paletten, die gerade zusammengestellt werden, anschauen, sondern auch welche Stoffe an diesem Tag noch folgen werden. So können sie vorausschauen und evtl. schon Zuschnitte vorzeihen um überflüssige Rollenwechsel zu vermeiden.

5 Wegoptimierung

Bestehende Web-App wieder zum Laufen bringen mit Hilfe von Philipp Nägeli und Logs, damit der normale Arbeitsbetrieb gewährleistet ist.

Es werden noch Erweiterungen am Server vorgenommen um Arbeitsprozesse zu erleichtern.

-Die Web-App soll den Import von .xls und .xlsx Dateien unterstützen

-Die Listen sollen automatisch geladen werden und können solange noch niemand daran gearbeitet hat auch aktualisiert werden. Wird versucht zu aktualisieren, während oder nachdem jemand daran arbeitete, wird eine Fehlermeldung ausgegeben und eine Email an admin.ch@hildinganders.com. Das automatische laden hat ein Intervall von Def=15 min. Ein File sollte jedoch niemals zwei Mal in den Server geladen werden.

-Die Freigabe der Listen von einem Administrativmitarbeiter kann entfernt werden, da dies den Prozess nur unnötig verlangsamt.

-Falls der Server neu gestartet wird muss unsere Web-App sich automatisch neu starten

-LogFile in konfigurierbarem ShareFolder führen

-Installationsinstruktion mit Testszenarien um sicher zu gehen, dass System läuft oder gezügelt werden kann => EXE für automatische Installation oder klare Weisungen

-KernenUpdate ebenfalls als Ergänzung mit neuen Date

-Bilder ergänzen austauschen => ID und FiledDatacode prüfen und updaten falls neuer job wird mit BilderAktualiseren ausgeführt

Entwicklung

Die grundlegende Ausgangslage für spätere Optimierungen an der Verwaltung des Matratzenkernlagers, ist die Berechnung und Annäherung der momentan gebrauchten Zeit aufgrund der alten Daten welche aus den letzten Jahren vorliegen.

Ziel dieses Arbeitsteiles ist es anhand der alten Daten, die Staplerfahrten nachzustellen und danach für jeden der jeweiligen aufgezeichneten Arbeitsschritte eine messbare Zeit zu definieren.

Hierfür müssen am Datenmodell einige Erweiterungen angebracht werden, welche dazu dienen Die Fahrten und die benötigten Zeiten zu modellieren. Dafür müssen die alten Daten der Matratzenkernpalletten in die Applikation eingelesen werden und auf die Entsprechend Vorbereiteten Datenmodelle abgeglichen werden, um schlussendlich eine Vergleichbare Zeitberechnung aufstellen zu können.

Das Nutzenpotenzial des Wegoptimierer Algorithmus versus optimiertes Lager sollte aufgezeigt werden, evtl. mit Empfehlung, wie das Lager umgestellt werden muss.

Version 5

Aufgabendefinition

1 Server

1.1 Web-App

Die Web-App muss wieder Laufen und sollte sich nach einem Beenden des Servers auch selbst wieder auf starten um den Arbeitsbetrieb zu gewährleisten.

Die Logs der Web-App sollten für berechtigte Benutzer leicht zugreifbar sein. Evtl. Mit einer Remote Desktop Verbindung vom Verantwortlichen auf den Server.

Die Matratzenkernbilder sollten sich automatisch aktualisieren, indem der

1.2 Android

Beim Login auf der Android App kommt zurzeit nur eine generelle Meldung, wenn das Login scheiterte. Diese Meldung sollte in Zukunft aussagen, ob das Problem bei den Benutzerdaten oder beim Server liegt.

2 Fehlmaterial

UC1: Ein Mitarbeiter sieht, dass an einem Lagerplatz kein Kern mehr vorhanden ist oder das Kanban-Minimum wurde erreicht. Mit der Android App scannt er den Barcode (Barcode an Säule auch wenn kein Material mehr vorhanden ist), wählt ob es eine Kanban-Nachbestellung oder Fehlmaterial ist und gibt seinen Namen ein. Diese Daten kann er absenden und sie werden in der Web-App unter Fehlmaterial angezeigt.

UC1.1: Ein Gabelstaplerfahrer kann angeben, wie viele Kerne ihm fehlen um seinen aktuellen Auftrag zu beenden.

UC1.2: Ein Gabelstaplerfahrer kann einen Termin eingeben, bis wann er eine Nachlieferung braucht.

UC1.3: Ein Mitarbeiter hat die Möglichkeit einen Kommentar zur Meldung zu schreiben für zusätzliche Informationen.

3 Q-Issue

UC2: Ein Mitarbeiter sieht defektes Material und kann mit der Android-App eine Meldung dazu erstellen. Damit möglichst gut auf das Problem reagiert werden kann, macht der Mitarbeiter ein Foto des Defekts, scannt den Barcode des Artikels und schreibt einen Kommentar dazu. Die Meldung geht per Email direkt an eine Mailbox für Materialfehler.

4 Rollenwechsel

UC3: Die Mitarbeiter beim Stoffzuschnitt haben die Möglichkeit in der Web-App verschiedene Sortierungen der aktuellen Aufträge anzusehen. So können sie nicht nur die Kernliste der Paletten, die gerade zusammengestellt werden, anschauen, sondern auch welche Stoffe an diesem Tag noch folgen werden. So können sie vorausschauen und evtl. schon Zuschnitte vorzeihen um überflüssige Rollenwechsel zu vermeiden.

5 Wegoptimierung

Die grundlegende Ausgangslage für spätere Optimierungen an der Verwaltung des Matratzenkernlagers, ist die Berechnung und Annäherung des momentan gebrauchten Weges aufgrund der alten Daten welche aus den letzten Jahren vorliegen.

Ziel dieses Arbeitsteiles ist es anhand der alten Daten, die Staplerfahrten nachzustellen und danach für jeden der jeweiligen aufgezeichneten Arbeitsschritte eine messbare Distanz zu definieren.

Hierfür müssen am Datenmodell einige Erweiterungen angebracht werden, welche dazu dienen Die Fahrten und die benötigten Wege zu modellieren. Dafür müssen die alten Daten der Matratzenkernpalletten in die Applikation eingelesen werden und auf die Entsprechend Vorbereiteten Datenmodelle abgeglichen werden, um schlussendlich eine Vergleichbare Wegberechnung durchführen zu können.

Mögliche weitere Schritte:

Das Nutzenpotenzial eines Wegoptimierten Algorithmus versus optimiertes Lager sollte aufgezeigt werden, evtl. mit Empfehlung, wie das Lager umgestellt werden muss. →Lagerverwaltung evtl. notwendig

---

Bestehende Web-App wieder zum Laufen bringen mit Hilfe von Philipp Nägeli und Logs, damit der normale Arbeitsbetrieb gewährleistet ist.

Es werden noch Erweiterungen am Server vorgenommen um Arbeitsprozesse zu erleichtern.

-Die Web-App soll den Import von .xls und .xlsx Dateien unterstützen

-Die Listen sollen automatisch geladen werden und können solange noch niemand daran gearbeitet hat auch aktualisiert werden. Wird versucht zu aktualisieren, während oder nachdem jemand daran arbeitete, wird eine Fehlermeldung ausgegeben und eine Email an admin.ch@hildinganders.com. Das automatische laden hat ein Intervall von Def=15 min. Ein File sollte jedoch niemals zwei Mal in den Server geladen werden.

-Die Freigabe der Listen von einem Administrativmitarbeiter kann entfernt werden, da dies den Prozess nur unnötig verlangsamt.

-Falls der Server neu gestartet wird muss unsere Web-App sich automatisch neu starten

-LogFile in konfigurierbarem ShareFolder führen

-Installationsinstruktion mit Testszenarien um sicher zu gehen, dass System läuft oder gezügelt werden kann => EXE für automatische Installation oder klare Weisungen

-KernenUpdate ebenfalls als Ergänzung mit neuen Date

-Bilder ergänzen austauschen => ID und FiledDatacode prüfen und updaten falls neuer job wird mit BilderAktualiseren ausgeführt.

Version 6

Aufgabendefinition

1 Server

1.1 Web-App

Die Web-App muss wieder Laufen und sollte sich nach einem Beenden des Servers auch selbst wieder neu starten um den Arbeitsbetrieb zu gewährleisten.

Die Logs der Web-App sollten für berechtigte Benutzer leicht zugreifbar sein.

Die Matratzenkernbilder sollen aktualisiert werden können.

1.2 Android

Beim Login auf der Android App kommt zurzeit nur eine generelle Meldung, wenn das Login scheiterte. Diese Meldung sollte in Zukunft aussagen, ob das Problem bei den Benutzerdaten oder beim Server liegt.

2 Fehlmaterial

UC1: Ein Mitarbeiter sieht, dass an einem Lagerplatz kein Kern mehr vorhanden ist oder das Kanban-Minimum wurde erreicht. Mit der Android App scannt er den Barcode (Barcode an Säule auch wenn kein Material mehr vorhanden ist), wählt ob es eine Kanban-Nachbestellung oder Fehlmaterial ist und gibt seinen Namen ein. Diese Daten kann er absenden und sie werden in der Web-App unter Fehlmaterial angezeigt.

UC1.1: Ein Gabelstaplerfahrer kann angeben, wie viele Kerne ihm fehlen um seinen aktuellen Auftrag zu beenden.

UC1.2: Ein Gabelstaplerfahrer kann einen Termin eingeben, bis wann er eine Nachlieferung braucht.

UC1.3: Ein Mitarbeiter hat die Möglichkeit einen Kommentar zur Meldung zu schreiben für zusätzliche Informationen.

3 Q-Issue

UC2: Ein Mitarbeiter sieht defektes Material und kann mit der Android-App eine Meldung dazu erstellen. Damit möglichst gut auf das Problem reagiert werden kann, macht der Mitarbeiter ein Foto des Defekts, scannt den Barcode des Artikels und schreibt einen Kommentar dazu. Die Meldung geht per Email direkt an eine Mailbox für Materialfehler.

4 Rollenwechsel

UC3: Die Mitarbeiter beim Stoffzuschnitt haben die Möglichkeit in der Web-App verschiedene Sortierungen der aktuellen Aufträge anzusehen. So können sie nicht nur die Kernliste der Paletten, die gerade zusammengestellt werden, anschauen, sondern auch welche Stoffe an diesem Tag noch folgen

werden. So können sie vorausschauen und evtl. schon Zuschnitte vorzeihen um überflüssige Rollenwechsel zu vermeiden.

5 Wegoptimierung

Die grundlegende Ausgangslage für spätere Optimierungen an der Verwaltung des Matratzenkernlagers, ist die Berechnung und Annäherung des momentan gebrauchten Weges aufgrund der alten Daten welche aus den letzten Jahren vorliegen.

Ziel dieses Arbeitsteiles ist es anhand der alten Daten, die Staplerfahrten nachzustellen und danach für jeden der jeweiligen aufgezeichneten Arbeitsschritte eine messbare Distanz zu definieren.

Hierfür müssen am Datenmodell einige Erweiterungen angebracht werden, welche dazu dienen Die Fahrten und die benötigten Wege zu modellieren. Dafür müssen die alten Daten der Matratzenkernpalletten in die Applikation eingelesen werden und auf die Entsprechend Vorbereiteten Datenmodelle abgeglichen werden, um schlussendlich eine Vergleichbare Wegberechnung durchführen zu können.

Version

Aufgabendefinition

1 Server

1.1 Web-App

Die Web-App muss wieder Laufen und sollte sich nach einem Beenden des Servers auch selbst wieder neu starten um den Arbeitsbetrieb zu gewährleisten.

Die Logs der Web-App sollten für berechtigte Benutzer leicht zugreifbar sein.

Die Matratzenkernbilder sollen aktualisiert werden können.

1.2 Android

Beim Login auf der Android App kommt zurzeit nur eine generelle Meldung, wenn das Login scheiterte. Diese Meldung sollte in Zukunft aussagen, ob das Problem bei den Benutzerdaten oder beim Server liegt.

2 Fehlmaterial

UC1: Ein Mitarbeiter sieht, dass an einem Lagerplatz kein Kern mehr vorhanden ist oder das Kanban-Minimum wurde erreicht. Mit der Android App scannt er den Barcode (Barcode an Säule auch wenn kein Material mehr vorhanden ist), wählt ob es eine Kanban-Nachbestellung oder Fehlmaterial ist und gibt seinen Namen ein. Diese Daten kann er absenden und sie werden in der Web-App unter Fehlmaterial angezeigt.

UC1.1: Ein Gabelstaplerfahrer kann angeben, wie viele Kerne ihm fehlen um seinen aktuellen Auftrag zu beenden.

UC1.2: Ein Gabelstaplerfahrer kann einen Termin eingeben, bis wann er eine Nachlieferung braucht.

UC1.3: Ein Mitarbeiter hat die Möglichkeit einen Kommentar zur Meldung zu schreiben für zusätzliche Informationen.

3 Q-Issue

UC2: Ein Mitarbeiter sieht defektes Material und kann mit der Android-App eine Meldung dazu erstellen. Damit möglichst gut auf das Problem reagiert werden kann, macht der Mitarbeiter ein Foto des Defekts, scannt den Barcode des Artikels und schreibt einen Kommentar dazu. Die Meldung geht per Email direkt an eine Mailbox für Materialfehler.

4 Rollenwechsel

UC3: Die Mitarbeiter beim Stoffzuschnitt haben die Möglichkeit in der Web-App verschiedene Sortierungen der aktuellen Aufträge anzusehen. So können sie nicht nur die Kernliste der Paletten, die gerade zusammengestellt werden, anschauen, sondern auch welche Stoffe an diesem Tag noch folgen werden. So können sie vorausschauen und evtl. schon Zuschnitte vorzeihen um überflüssige Rollenwechsel zu vermeiden.

5 Wegoptimierung

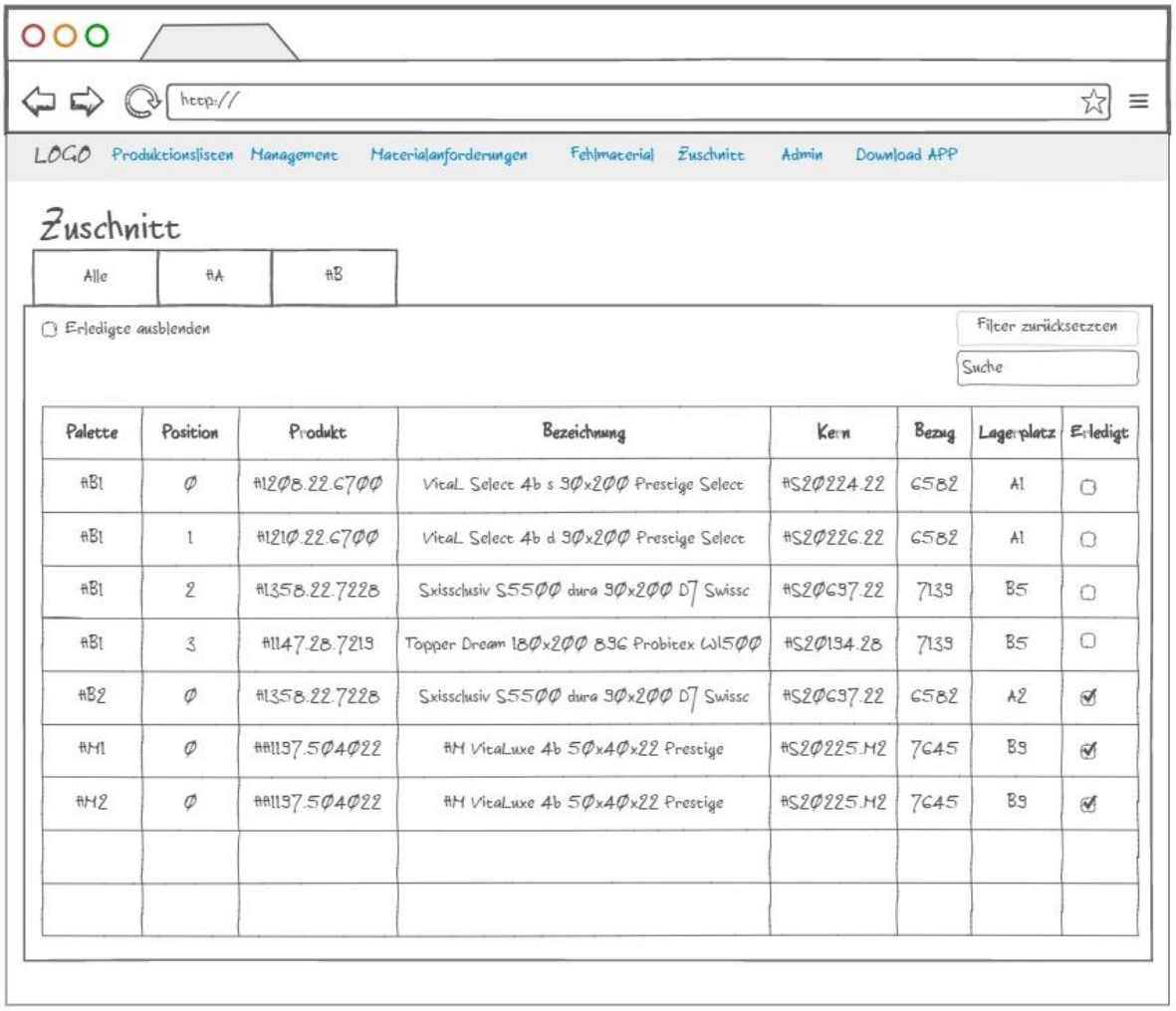
Die grundlegende Ausgangslage für spätere Optimierungen an der Verwaltung des Matratzenkernlagers, ist die Berechnung und Annäherung des momentan gebrauchten Weges aufgrund der alten Daten welche aus den letzten Jahren vorliegen.

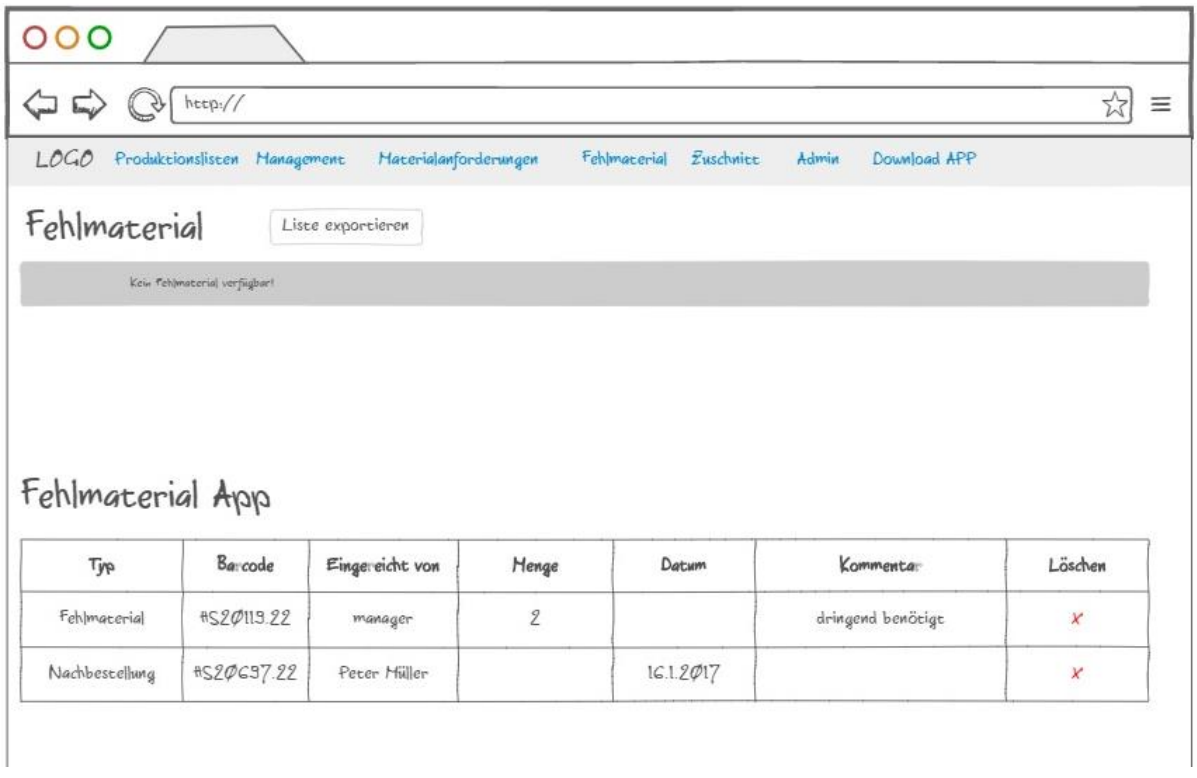
Ziel dieses Arbeitsteiles ist es anhand der alten Daten, die Staplerfahrten nachzustellen und danach für jeden der jeweiligen aufgezeichneten Arbeitsschritte eine messbare Distanz zu definieren.

Hierfür müssen am Datenmodell einige Erweiterungen angebracht werden, welche dazu dienen Die Fahrten und die benötigten Wege zu modellieren. Dafür müssen die alten Daten der Matratzenkernpalletten in die Applikation eingelesen werden und auf die Entsprechend Vorbereiteten Datenmodelle abgeglichen werden, um schlussendlich eine Vergleichbare Wegberechnung durchführen zu können.

A.1.3 Wireframes

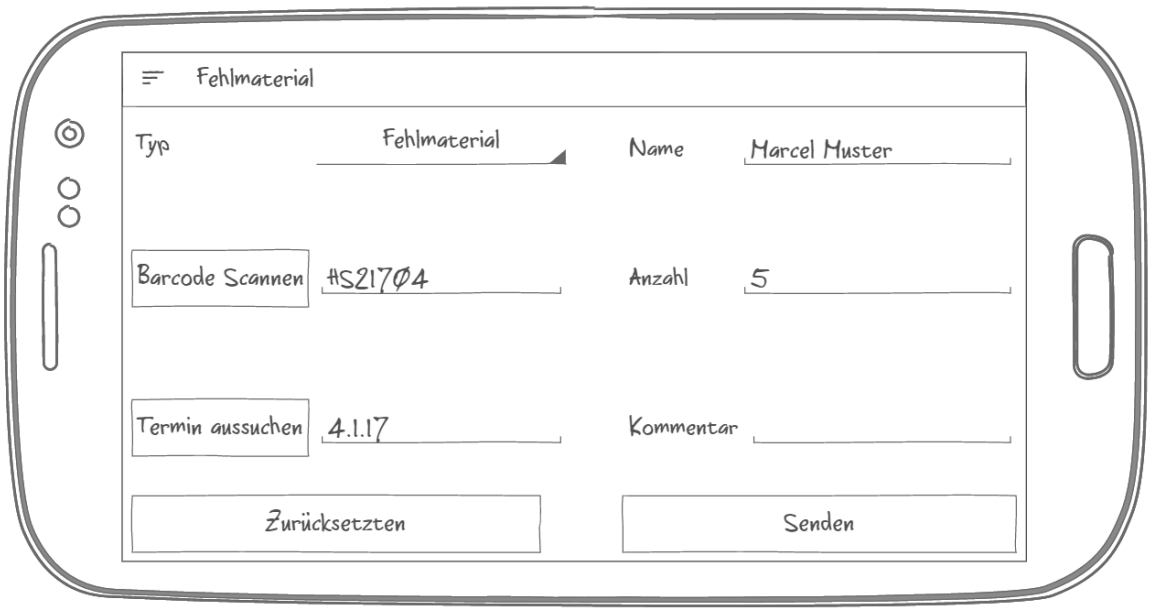
WebApp





Android App


#A-Mitarbeiter Meldungen
QIssue
Fehlmaterial
Einstellungen
Name
Marcel Muster
Speichern
QIssue Email
admin@product.optim
WebApp
WebApp öffnen
Update
Update herunterladen
#A-Mitarbeiter Meldungen
QIssue
Fehlmaterial
Einstellungen
Typ
Fehmaterial
Barcode scannen
#S217Ø4
Termin aussuchen
4.1.17
Name
Marcel Muster
Anzahl
3
Kommentar
Material verschmutzt
Zurücksetzten
Senden

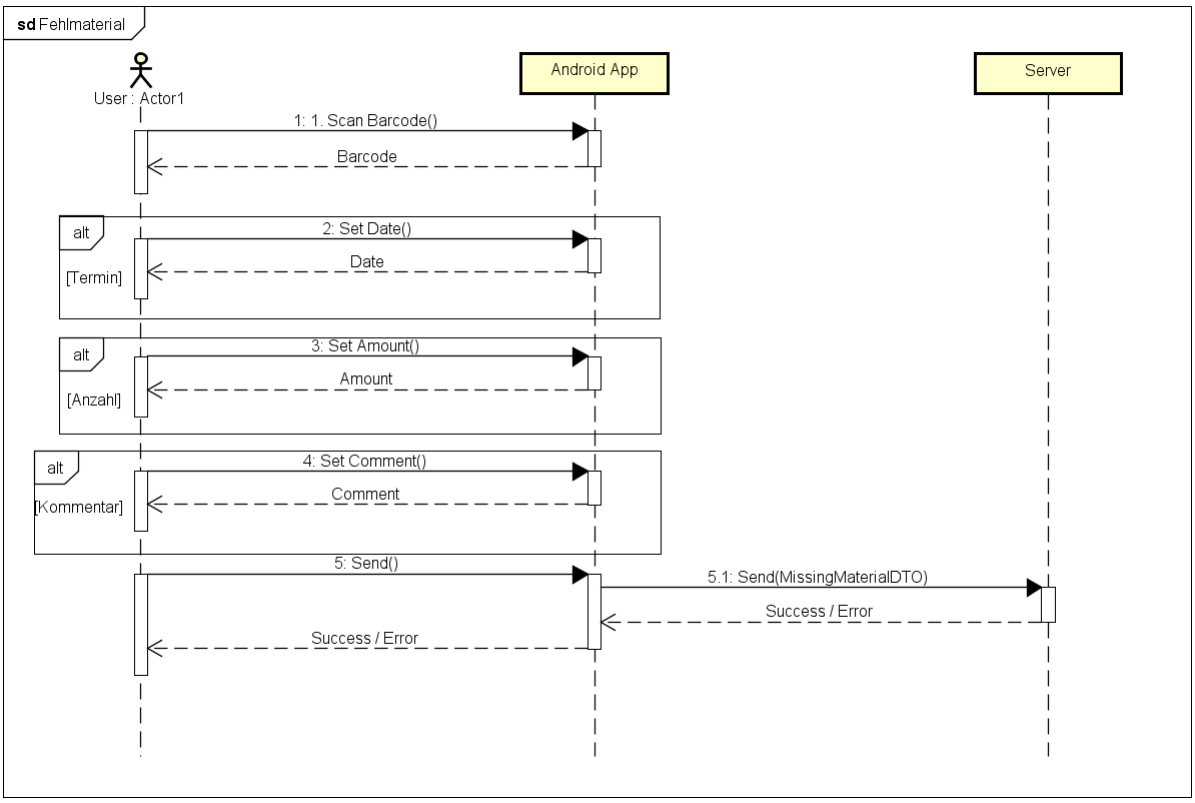

Fehlmaterial
Typ
Fehlmaterial
Name
Marcel Muster
Barcode Scannen
#S217Ø4
Anzahl
5
Termin aussuchen
4.1.17
Kommentar
Zurücksetzten
Senden


#A-Mitarbeiter Meldungen
QIssue
Fehlmaterial
Einstellungen
Barcode scannen
#S217Ø4
BetreFF:
Paket defekt
Zurücksetzten
Senden

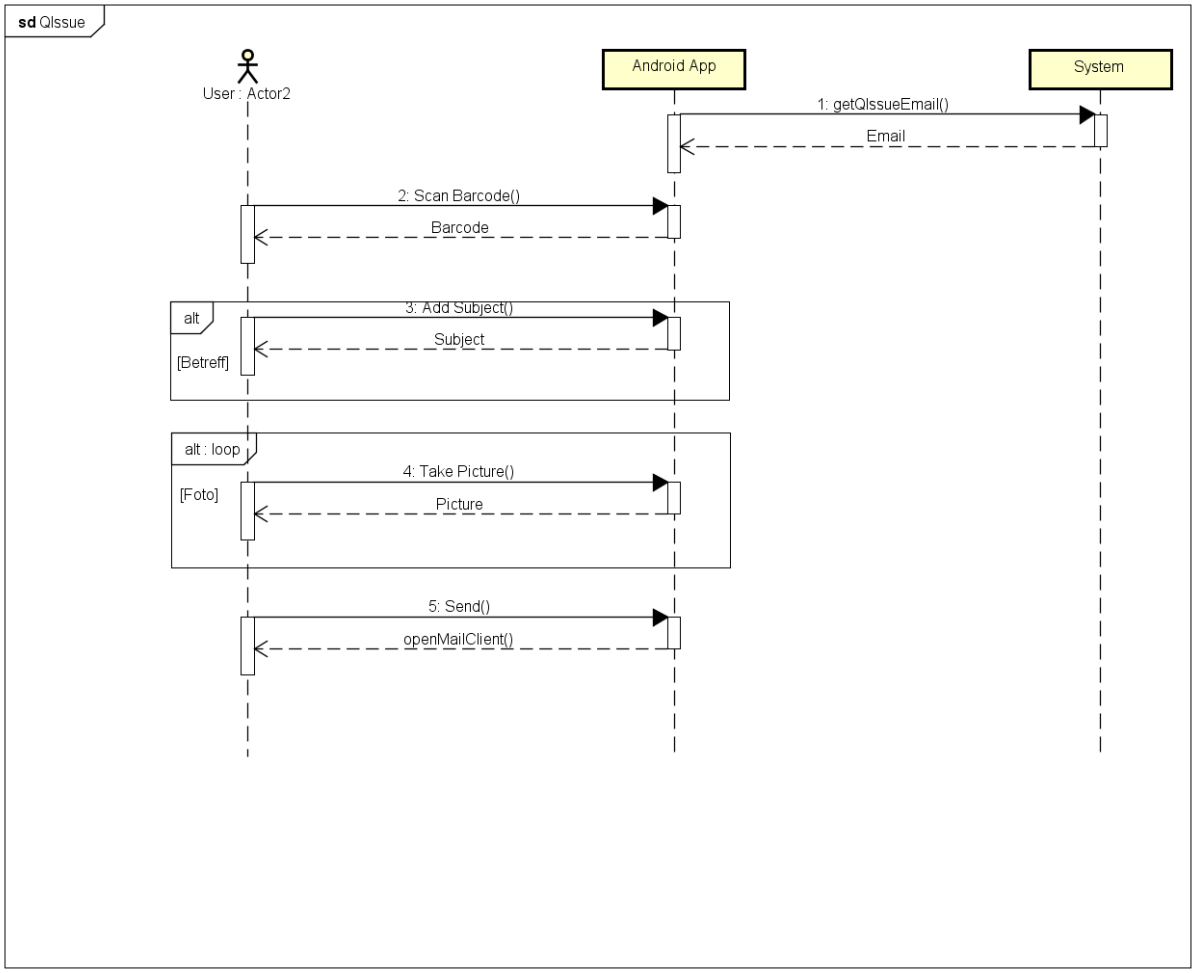

Qualitätsmangel
Barcode scannen
#S217Ø4
BetreFF
Kern schmutzig
Zurücksetzten
Senden

A.1.4 SSDs

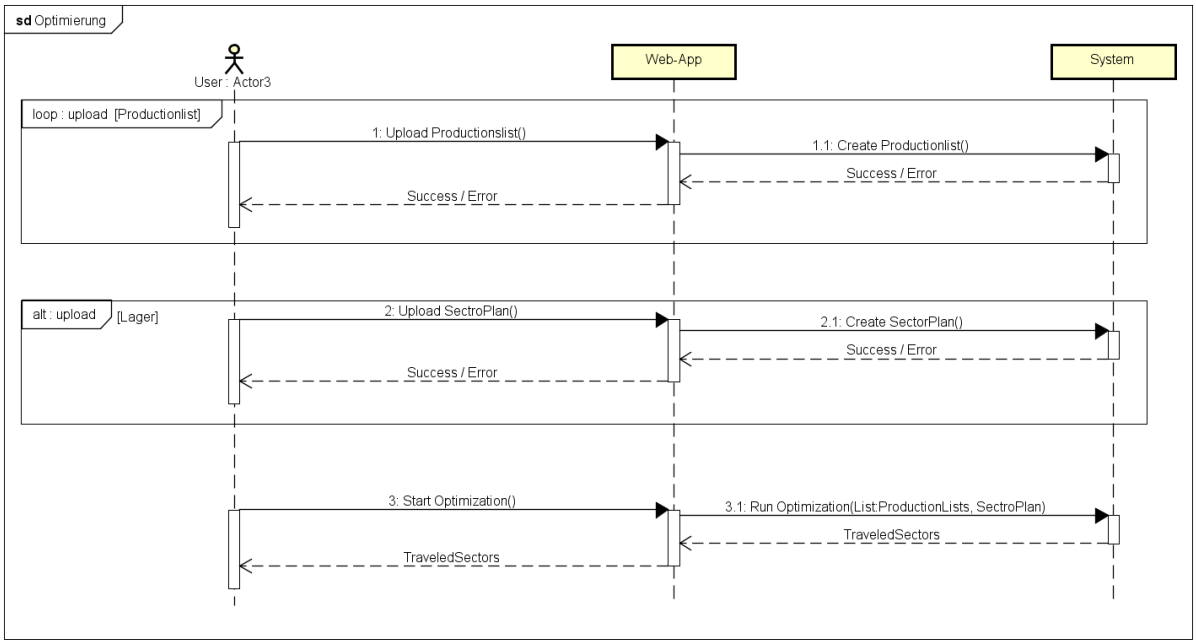
Fehlmaterial



Q-Issue



Optimierung



A.1.5 Contracts

sendMissingMaterial (UC1 Fehlmaterial)

Operation:	send(missingMaterialDTO)
Cross Reference:	Use Case 1 Fehlmaterial erfassen
Preconditions:	Typ der Meldung ist gewählt, Barcode wurde erfasst und es ist ein Name gesetzt.
Postconditions:	In der WebApp ist die Fehlmaterialmeldung zu sehen.

email (UC2 Q-Issue)

Operation:	email(context, emailTo, emailCC, subject, emailText, filePaths)
Cross Reference:	Use Case 2 Q-Issue erfassen
Preconditions:	Ein Barcode wurde erfasst und mindestens ein Foto des Defektes wurde aufgenommen.
Postconditions:	Das Gerät öffnet ein Email und fügt die gemachten Angaben automatisch in die entsprechenden Felder.

runOptimizationFor (UC4 Optimierung)

Operation:	runOptimizationFor (sectroPlan, selectedPathCoreLists)
Cross Reference:	UC4 Optimierung
Preconditions:	Es wurde ein Lagerplan ausgewählt und die Palettlisten, welche mit dem gewählten Lager simuliert werden sollen.
Postconditions:	Man sieht die Anzahl Sektoren, welche befahren wuden.