IFW
Institut für Fertigungstechnik
und Werkzeugmaschinen

HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
Fakultät 1
Elektro- und Informationstechnik

Entwicklung einer mobilen Anwendung zur Produktnachverfolgung im Fertigungsprozess

Bachelorarbeit

im Studiengang Elektro- und Informationstechnik der Fakultät I

Hochschule Hannover

Institut für Fertigungstechnik und Werkzeugmaschinen Hannover (IFW)

von

Marvin Sperling

Matrikelnummer: 1302260

Erstprüfer:	Prof. Dr.-Ing. Martin Mutz
Zweitprüfer:	M. Sc. Daniel Arnold
Abgabetermin:	25.08.2017

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach anderen gedruckten oder im Internet verfügbaren Werken entnommen sind, habe ich durch genaue Quellenangaben kenntlich gemacht.

Hannover, den 24.08.2017 _____

Marvin Sperling

Abstract

Die Digitalisierung birgt sowohl für den Unternehmer, als auch für den Kunden diverse Vorteile. Als Instrument zur Veranschaulichung dieser Vorteile dienen vor allem Service-Apps, welche bereits von großen Unternehmen beispielsweise zur Produktnachverfolgung etabliert wurden. Das Mittelstand 4.0 – Kompetenzzentrum Hannover legt den Fokus auf kleine und mittelständische Produktionsbetriebe und vermittelt das Potenzial der Digitalisierung im Rahmen von Industrie 4.0. Dies geschieht anhand einer Produktionslinie personalisierter Kugelschreiber und einer zu Demonstrationszwecken entwickelten App für mobile Endgeräte. Konzeptionierung, Entwicklung und Umsetzung dieser App werden genauestens aufgeführt und erläutert. Die App wurde unter dem Betriebssystem Android entwickelt und gliedert sich in sieben Phasen, beginnend mit der Anforderungsanalyse bis hin zur Veröffentlichung im Google Play Store. Während der Unternehmer die Möglichkeit geboten bekommt Rückschlüsse als Entscheidungshilfe zur effizienteren Gestaltung der Produktion zu nutzen, ist der Kunde in der Lage seine Aufträge echtzeitnah in der Produktion nachzuverfolgen und gegebenenfalls Änderungen vorzunehmen.

Inhaltsverzeichnis

Abbildungs- und Tabellenverzeichnis

Abbildungen

Tabellen

1. Einleitung

In Zusammenarbeit mit der Hochschule Hannover und dem Institut für Fertigungstechnik und Werkzeugmaschinen der Leibniz Universität Hannover wird in dieser Arbeit eine Service-App für die Demonstrationsfabriken des „Mittelstand 4.0 Kompetenzzentrum Hannover“ konzipiert. Die App bietet dem Kunden die Möglichkeit, den Produktionsprozess seiner Aufträge aktiv nachverfolgen und sogar nach Produktionsbeginn ändern zu können. Der Kunde soll den Status seiner Aufträge abrufen können. Dazu muss der Produktionsfortschritt in geeigneter Weise dargestellt werden. Des Weiteren soll der Kunde die Möglichkeit haben, einen neuen Auftrag zu erstellen.

Das Institut für Fertigungstechnik und Werkzeugmaschinen (IFW) ist ein Institut der Leibniz Universität Hannover, deren Forschungskern die zerspanende Fertigungstechnik ist. Seit Anfang 2016 arbeitet das IFW mit weiteren Instituten und Partnern aus Niedersachsen an dem Projekt: „Mittelstand 4.0 Kompetenzzentrum Hannover“, wobei die Leitung des Projekts hauptsächlich in den Händen der Institute liegt.

Das Mittelstand 4.0 Kompetenzzentrum Hannover wird vom Bundesministerium für Wirtschaft und Energie gefördert und wird unter dem Slogan „Mit uns Digital! Das Zentrum für Niedersachsen und Bremen“ firmiert. Es ist das erste von derzeit elf Zentren in Deutschland, welche dazu gegründet wurden, kleinen und mittelständischen Unternehmen und Handwerksbetrieben durch Informationsveranstaltungen und Anschauungsbeispielen bei der Integration der Digitalisierung im eigenen Unternehmen zu unterstützen. Das Kompetenzzentrum präsentiert auf dem Messegelände Hannover seine Generalfabrik, um Unternehmen an Hand einer Kugelschreiber-Produktion beispielhaft zu verdeutlichen, welche bezahlbaren Technologien hinsichtlich der Digitalisierung umsetzbar sind. Dazu gehören Kommissionierassistenzsysteme, Auftragserfassungstechnologien, sowie simulationsunterstützte Montageplätze und fahrerlose Transportsysteme. Künftig zählt unt eine App für mobile Endgeräte zu diesen Technologien. Für das Flächenland Niedersachsen hat das Kompetenzzentrum eine mobile Fabrik entwickelt, um den Unternehmen vor Ort zum Thema Digitalisierung zu informieren und anwendungsnahe Technologien zu demonstrieren. [DEN17]

Das Internet der Dinge („Internet of Things“, kurz: IoT) steht für die Entwicklung, Geräte und Gegenstände des Alltags zu vernetzen. Dazu werden diese mit Sensoren, Prozessoren und Netzwerktechnik ausgestattet. Hauptziel des IoT ist, den Alltag des Menschen zu vereinfachen, ob Privat oder im Unternehmen. Es ist maßgeblich für die Digitalisierung von Produktion und Logistik verantwortlich. [VOG17b]
Die Möglichkeiten der Vernetzung zwischen den Systemen bilden die Grundlage für vollkommen neue, angepasste Geschäftsmodelle. Ein wesentlicher Punkt dabei ist das Angebot zusätzlicher Dienstleistungen durch den Einsatz von Apps. In den Branchen Logistik und Personenverkehr, wie auch dem Onlinehandel haben sich solche Anwendungen bereits etabliert. Auch in der Industrie werden Apps verwendet, mit deren Hilfe Maschinen und Anlagen konfiguriert werden [LEN17], Aufträge nachverfolgt werden, oder aber Mengen und Bestände der Lager zurückmelden [SAP17].

Für die Demonstrationsfabriken des Kompetenzzentrums wurde eine App entwickelt, mit der kleinen und mittelständischen Unternehmen die Vorteile einer solchen App aufgezeigt werden. Hauptsächlich soll die App für Transparenz in der Produktion sorgen, um den Kunden des Kompetenzzentrums die Digitalisierung näher zu bringen. Die App zeigt, welchen Mehrnutzen eine App für ein produzierendes Unternehmen haben kann. Dazu gehört die Benachrichtigung von Fehlern und Ausfällen in der Produktion, als auch Maschinenauslastungen und der Analyse von Durchlaufzeiten. Unternehmer können rechtzeitig auf Ereignisse in der Produktion reagieren, sei es ein Defekt einer Maschine oder wiederholt auftretende Verzögerungen in den Durchlaufzeiten. Durch die Analyse von Maschinenauslastungen kann die Produktion effektiver gestaltet und Leerlaufzeiten vermieden werden.
Zu Beginn dieser Arbeit wird der gegenwärtige[1] Forschungsstand zu den Themen Digitalisierung und Service-Apps genauer referiert. Zudem wird in diesem Kapitel diskutiert, welche Chancen die Digitalisierung bietet und wie produzierende Unternehmen davon profitieren können. Aktuelle Zahlen und Fakten zu Betriebssystemen für mobile Endgeräte sind ebenfalls Gegenstand dieses Kapitels.

Im nächsten Kapitel wird die Herangehensweise zur Entwicklung einer mobilen Anwendung beschrieben. Die Entwicklung ist in sieben Phasen gegliedert, beginnend mit der Anforderungsanalyse, über die Implementierung bis hin zur Veröffentlichung. Hier werden alle Ergebnisse, die zur Entwicklung der App beigetragen haben, erwähnt. Dazu gehört nicht nur die App, sondern auch die Datenkommunikation zwischen der Fabrik und der App, welche die echtzeitnahe Produktionsnachverfolgung möglich macht. In den letzten beiden Phasen der Appentwicklung wird auf die Testphase und die Veröffentlichung der App eingegangen. Mit Hilfe einer Betaphase konnte die App vor der Veröffentlichung auf verschiedenen Endgeräten getestet werden. Durch Rückmeldungen der Tester konnten letzte Mängel und Fehler beseitigt und zusätzliche Wünsche der Tester umgesetzt werden. Am Ende dieser Arbeit sind Empfehlungen zur Frage der Marktreife einer App zu finden, die aus Erfahrungen durch die Entwicklung einer mobilen Anwendung erstellt wurden.

Unternehmer und Mitarbeiter können ihre Prozesse durch eine einfache Visualisierung, wie der Produktnachverfolgung, einfacher verstehen. Zudem wird gezeigt, dass Kunden durch die App eine zusätzliche Dienstleistung neben dem eigentlichen Produkt geboten werden kann. Mit der App kann der Kunde seine Aufträge in der Produktion nachverfolgen und nach Produktionsbeginn anpassen.

[1] Stand: 2017

2. Grundlagen

In diesem Kapitel werden grundlegende Themen behandelt, die zum Verständnis dieser Arbeit beitragen. Darunter gehört der allgemeine Aufbau einer Android-App, Grundlagen über Datenbanken und der ausführlichen Beschreibung der Demonstrationsfabriken, für deren Produktionssystem die App entwickelt wird.

2.1. Aufbau einer Android-App

Da im Rahmen dieser Arbeit eine mobile Anwendung für Android-betriebssysteme entwickelt wurde, werden in diesem Kapitel die Hauptbestandteile einer Android-App beschrieben.

Eine Android-App besteht hauptsächlich aus einer Manifest-Datei, Layout-Dateien und Activities. In der Manifest-Datei stehen wesentliche Informationen über die App, die Android benötigt, um die App ausführen zu können. Zu den wesentlichen Informationen gehören Zugriffsberechtigungen („uses-permission“), wie auch alle Activities und deren zusätzlichen Eigenschaften, wie z.B. der Bildschirmorientierung der jeweiligen Activity.

Zu einer Activity gehört eine Java-Datei, in der die Funktionalität der Activity in der Programmiersprache Java hinterlegt wird, sowie eine xml-basierte Layoutdatei, in der das Layout der Benutzeroberfläche erstellt wird [KÜN17]. Mit Hilfe eines Editors die Layouts designt werden (siehe Abbildung 1).

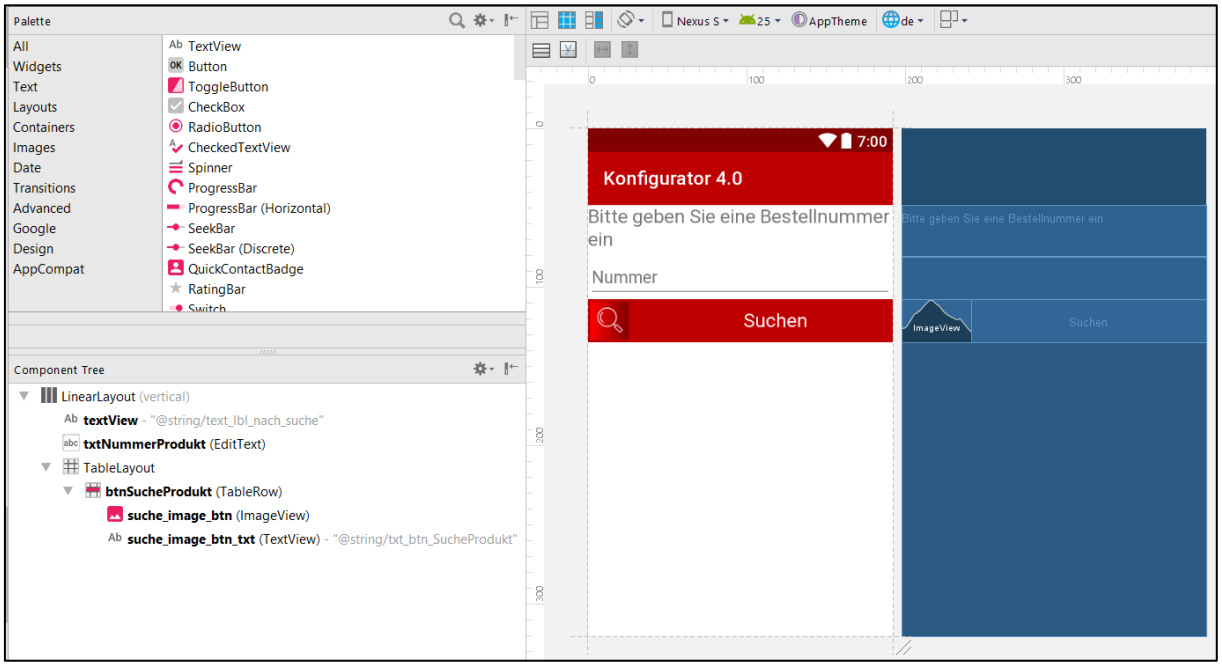


Abbildung 1: Abbildung Layout-Editor, Android Studio

In dem Layout-Editor sind oben links die verwendbaren und unten links die verwendeten Objekte aufgelistet. Rechts in der Abbildung ist eine Vorschau der designten Activity zu sehen.

Grundlage für die Entwicklung jeder Android-App ist der Lebenszyklus einer Activity („The Activity Lifecycle“, siehe Abbildung 2) [GOO17a]; [GOO17g].

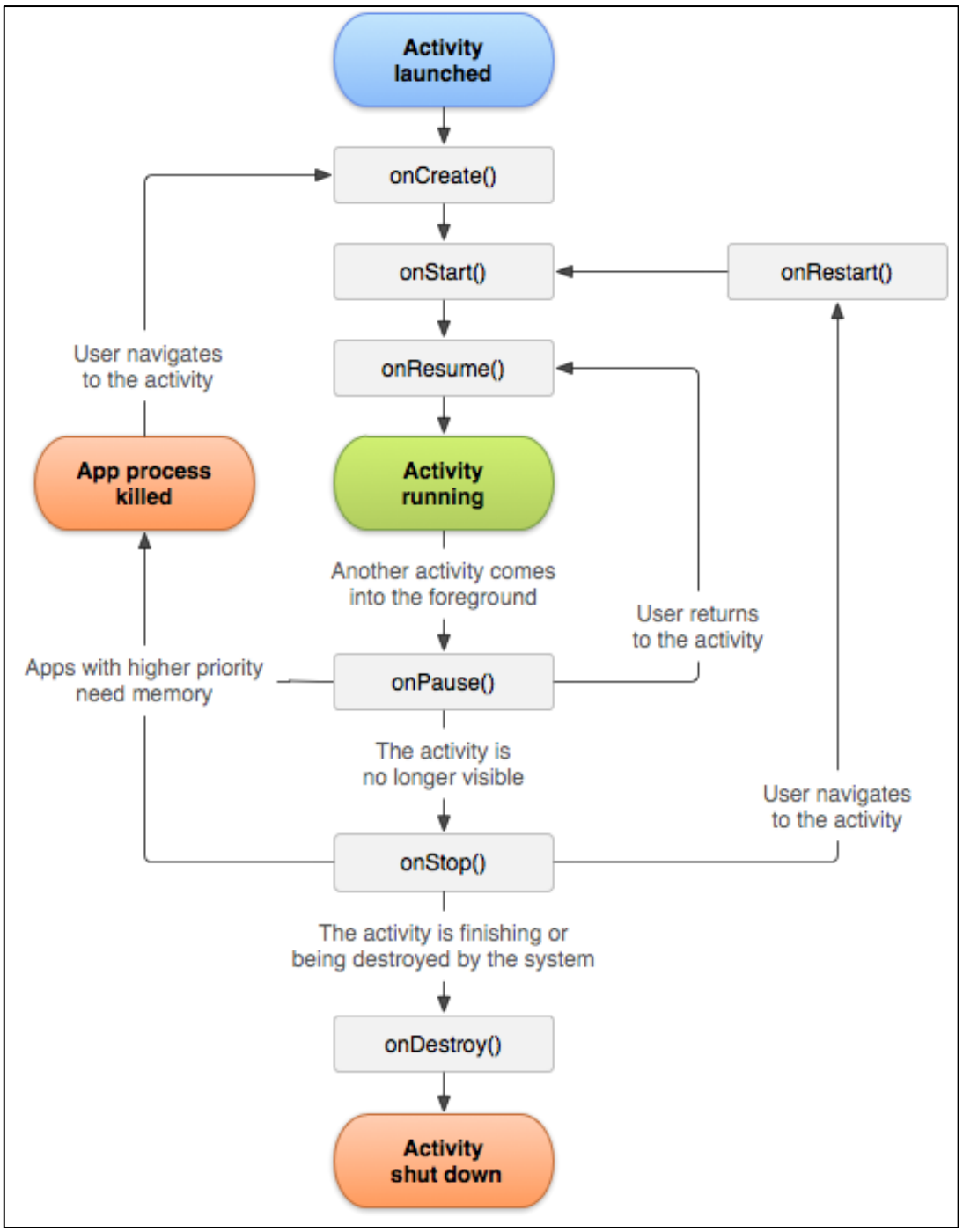


Abbildung 2: The Activity Lifecycle [GOO17g]

Der sogenannte Activity-Lebenszyklus beschreibt, welche Zustände eine Activity annehmen kann und wie diese miteinander zusammenhängen. Je nach Zustand werden die sogenannten Zustandswechselmethoden der jeweiligen Activity aufgerufen. Diese Methoden werden vererbt von der Oberklasse *AppCompatActivity* des Javapackages *android* und können in der Activity.java-Datei wie in Abbildung 3 überschrieben werden.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_start);

    //Code, was in der OnCreate()-Methode noch passieren soll,
    //zum Beispiel onClickListener für Buttons

}
```

Abbildung 3: Überschreiben der onCreate()-Methode

In folgender Tabelle (Tabelle 1) sind die Zustandswechselmethoden mit ihren Funktionen aufgelistet:

Tabelle 1: Zustandsänderungsmethoden und deren Funktion [GOO17a]

Name und Aufruf der Methode	**Funktion der Methode**
onCreate()	Die Methode wird beim Erstellen einer Activity aufgerufen und ist die einzige Zustandswechselmethode die zwingend implementiert werden muss.
onStart()	Methode wird aufgerufen, wenn Activity sichtbar ist.
onSaveInstanceState()	Die Methode wird unmittelbar nach der Methode onPause() aufgerufen. Sie speichert alle Zustände von Bedienelementen, die durch die Methode onRestoreInstanceState() wiederhergestellt werden können.
onRestoreInstanceState()	Die Methode wird nach onStart() aufgerufen. Sollte zuvor die Methode onSaveInstanceState() aufgerufen worden sein, so können gesicherte Zustände von Bedienelementen wiederhergestellt werden.
onResume()	Die Methode wird nach jeder Rückkehr aus einem pausierten Zustand aufgerufen.
onPause()	Diese Methode wird aufgerufen, wenn eine andere Activity oder ein Thread die Bedienung der aktuellen Activity verhindert.
onStop()	Die Methode wir aufgerufen, wenn eine Activity nicht mehr sichtbar ist.
onDestroy()	Die Methode wird aufgerufen, wenn eine Activity beendet wird. Hauptaufgabe dieser Methode ist das freigeben von arbeitsspeicher- und Prozessorressourcen, wie z.B. das Beenden von Threads.

2.2. Datenbanken

Grundlage für die Digitalisierung sind Daten (Informationen über den Produktionsprozess), die zu jeder Zeit abrufbar sind. Zur Verwaltung großer Datenmengen werden Datenbanken verwendet. In heutiger Zeit lässt sich so gut wie kein großes produzierendes Unternehmen finden, was keine Datenbank zur Datenverwaltung verwendet [KEM06]. Produktionsdaten können in zwei Kategorien, Stammdaten und Bewegungsdaten, eingeteilt werden.

Die Stammdaten (auch Bestandsdaten genannt) sind Daten, die Informationen über Eigenschaften eines Produktes oder Kundendaten beinhalten [HAN05]. Dazu gehören beispielsweise Kundendaten mit Vor- und Nachnamen, Kundennummer und deren bisherigen Aufträge, wie auch die Zusammensetzung des Produktes aus Einzelteilen.

Die Bewegungsdaten sind Daten, die im Laufe der Fertigung auftreten. Sie werden für die Aktualisierung von Stammdaten herangezogen [LAC17]. Ein Beispiel für Bewegungsdaten sind Zustandsdaten eines Produktes innerhalb einer Fertigung.

Um die Sicherheit der Daten in der Datenbank zu gewährleisten gibt es verschiedene Möglichkeiten. Dazu gehört die Identifikation und Authentisierung. Hierbei muss sich der Benutzer vor Nutzung des Datenbanksystems identifizieren. Dies kann zum Beispiel durch die Eingabe eines Benutzernamens erfolgen. Um die Authentisierung zu prüfen, wird meist neben dem Benutzernamen ein Passwort abgefragt, welches Gemeinsam zuvor als Nutzerkonto im Datenbanksystem registriert wurde. Neben der Identifikation und Authentisierung gibt es zusätzlich die Möglichkeit der Zugriffskontrolle. Durch die Vergabe von Rechten im Datenbanksystem können Nutzer oder Gruppen, bestehend aus Nutzern, ausschließlich auf die Daten zugreifen, welche für das Nutzerkonto freigegeben wurden. [KEM06]

2.3. Die Demonstrationsfabriken

Das Mittelstand 4.0 Kompetenzzentrum Hannover produziert in ihren Demonstrationsfabriken, der Generalfabrik und der mobilen Fabrik, hochwertige, personalisierte Aluminiumkugelschreiber (vgl. Abbildung 8) zur Veranschaulichung und Erprobung von Technologien des Industrie 4.0- und Digitalisierungszeitalters. Die Fabriken bestehen aus mehreren Bearbeitungsstationen, die den klassischen Betrieb einer Produktion von der Herstellung über die Kommissionierung bis hin zur Endmontage widerspiegeln. Die Bearbeitungsreihenfolge kann durch die Digitalisierung hinsichtlich der Maschinenauslastungen optimal festgelegt werden.

2.3.1. Die Generalfabrik

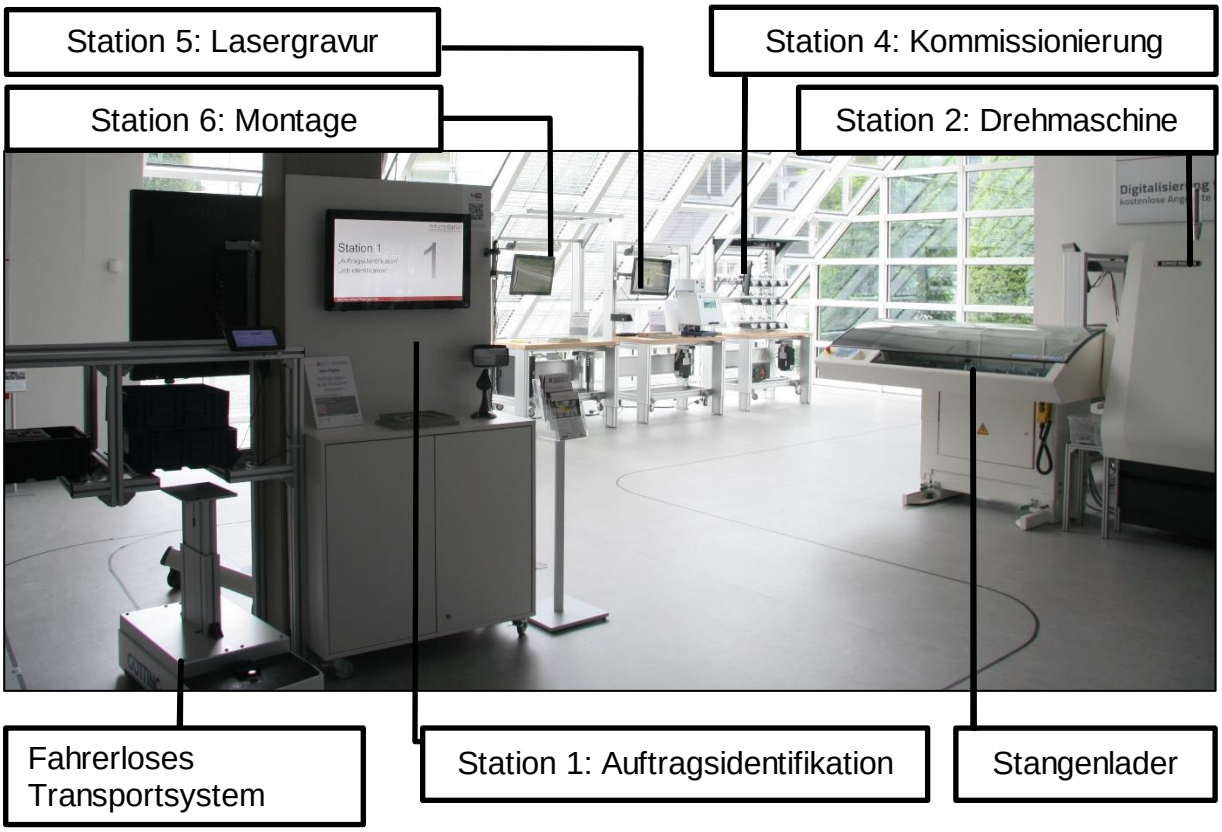


Abbildung 4: Generalfabrik

Die Generalfabrik (siehe Abbildung 4) umfasst sieben Stationen, an denen Unternehmer und Angestellte Technologien zur Digitalisierung erproben können. Unten links im Bild ist ein fahrerloses Transportsystem zu erkennen. Die schwarzen Markierungen auf dem Hallenboden werden als Fahrstraßen bezeichnet und dienen der Technologie als Wegweiser. Rechts neben dem fahrerlosen Transportsystem befindet sich die erste Station der Auftragsidentifizierung. Rechts im Bild ist ein Teil der CNC Drehmaschine des Typs „NEF 400“ mit dem links danebenstehenden Stangenlader zu erkennen. Im Hintergrund des Bildes kann man von rechts beginnend die Stationen vier, fünf und sechs erkennen. Die Bearbeitungsstationen teilen sich wie folgt auf:

Tabelle 2: Bearbeitungsstationen Generalfabrik

Station 1	Auftragsidentifikation
Station 2	Drehmaschine
Station 3	Nachbearbeitung
Station 4	Kommissionierung
Station 5	Gravieren/ Personalisieren
Station 6	Montage
Station 7	Qualitätsprüfung

2.3.2. Die mobile Fabrik

Abbildung 5: mobile Fabrik

Die mobile Fabrik (siehe Abbildung 5) ist in einem ausrangierten Linienbus integriert. Sie umfasst sechs Stationen. Eine umgerüstete Fräse fungiert als Drehmaschine zur Herstellung des Griffstückes (rechts im Bild). Eine Nachbearbeitungsstation wie in der Generalfabrik ist nicht notwendig. Die Bearbeitungsstationen der mobilen Fabrik unterscheiden sich nur geringfügig. Im Fokus dieses Bildes befindet sich ein Roboter der Firma „Universal Robots“, der in der Station 6 der Qualitätssicherung integriert ist. Die Stationen der mobilen Fabrik teilen sich wie folgt auf:

Tabelle 3: Bearbeitungsstationen mobile Fabrik

Station 1	Auftragsidentifikation
Station 2	Fräsmaschine
Station 3	Kommissionierung
Station 4	Gravieren/ Personalisieren
Station 5	Montage
Station 6	Qualitätsprüfung

2.3.3. Die Bearbeitungsstationen

Grundsätzlich besteht jede Station aus einem Monitor zur Interaktion mit dem Kunden (folgend Anwender genannt) und einer Vorrichtung, in die der Werkstückträger eingelassen werden kann, um durch den integrierten RFID-Chip im Werkstückträger die zugehörigen Auftragsparameter aus der Datenbank zu lesen und diese zu aktualisieren. Bei Beginn und bei Beendigung werden die Bestandsdaten des Auftrages in der Datenbank aktualisiert.

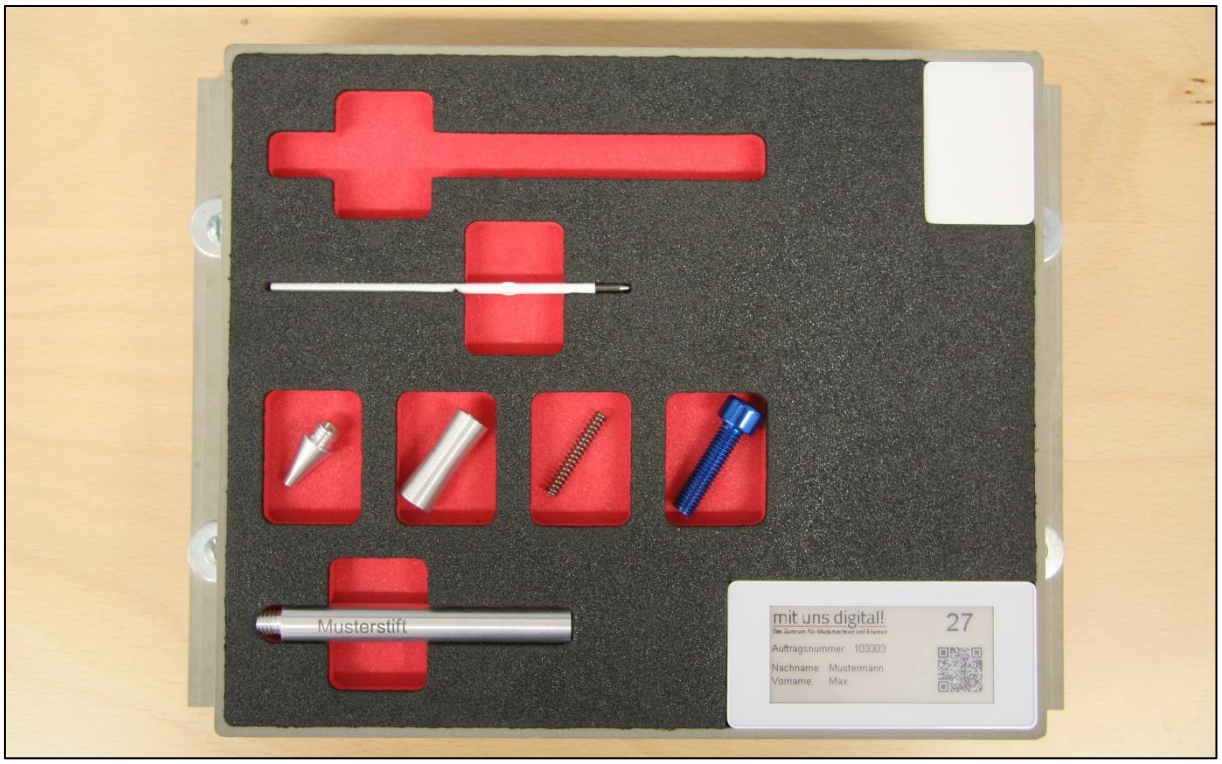


Abbildung 6: Abbildung eines Werkstückträgers

Der Werkstückträger (siehe Abbildung 6) ist konzipiert, um alle Teile, die zur Fertigung eines Produktes gehören, sortiert die Wertschöpfungskette durchlaufen können. Auf der Unterseite des Werkstückträgers befindet sich der RFID-Chip. Die Zuweisung eines Werkstückträgers zu einem Auftrag erfolgt in der Station 1, der Auftragsidentifikation (siehe Kapitel 2.3.3.1 „Bearbeitungsstationen der Generalfabrik“). Des Weiteren besitzt ein Werkstückträger ein Lokalisierungssystem (oben rechts in der Abbildung). Dies dient dazu, in dem Fabrikleitstand die aktuelle Position in der Fabrik, analog zu einem GPS-Ortungssystem, zu überwachen. Zusätzlich hat der Werkstückträger ein 2 Zoll großes elektronisches Preisschild (unten rechts in der Abbildung), um Informationen über die Bestellung, wie auch ein QR-Code der künftig die Auftragsnummer enthält, anzuzeigen.

2.3.3.1. Bearbeitungsstationen der Generalfabrik

Station 1, Auftragsidentifikation

Bei der ersten Station steht neben einem Monitor und der Vorrichtung für den Werkstückträger zusätzlich ein QR-Codescanner bereit. Ziel dieser Station ist es, einen zuvor erstellten Auftrag in das System zu bringen und einen Werkstückträger mit diesem Auftrag zu koppeln. Dafür muss der Anwender einen freien Werkstückträger auf die dazugehörige Vorrichtung legen und seinen zuvor erstellten Auftrag in Form eines QR-Codes mit dem Scanner einlesen.

Station 2, Drehmaschine

Die zweite Station besteht aus einer Drehmaschine des Typs „NEF 400“ zur Fertigung von Griffstücken nach Kundenwunsch. Dazu wird der passende Datensatz nach Auflegen des Werkstückträgers auf die Vorrichtung aus der Datenbank gelesen. Der Programmcode der Drehmaschine beinhaltet Variablen, die mit dem dazugehörigen Parameter der Auftragsdaten aus der Datenbank ersetzt werden.

Station 3, Nachbearbeitung

Bei der dritten Station handelt es sich um eine nachträglich digitalisierte Bohrmaschine, die zur Entgratung des zuvor produzierten Griffstückes dient. An dieser Station soll gezeigt werden, dass auch alte, nicht digitalisierte Maschinen und Anlagen in ein digitalisiertes System integriert werden können.

Station 4, Kommissionierung

Die Station der Kommissionierung ist mit einem Pick-by-Light-System ausgestattet (siehe Abbildung 7).

Abbildung 7: Pick-by-Light-System

Nach Auflegen des Werkstückträgers werden die Materialdaten aus der Datenbank gelesen. Die zu entnehmenden Teile werden durch blaue Leuchtmarkierungen angezeigt. Nach erfolgreicher Entnahme eines Teiles werden die Bestandsdaten in der Datenbank aktualisiert, welche durch einen Laserscanner detektiert wird. Falsche Entnahmen werden durch rote Leuchtmarkierungen angezeigt.

Station 5, Lasergravur

Bei der fünften Station wird der Kugelschreiber durch eine Lasergravur personalisiert. Nach Auflegen des Werkstückträgers wird der Anwender über den Monitor der Station angewiesen ein Stiftrohling in den Laser zu legen. Im Hintergrund hat die Recheneinheit des Lasers den zu gravierenden Text aus der Datenbank gelesen. Nach dem Starten des Lasers wird der Stiftrohling mit dem gewünschten Schriftzug graviert.

Station 6, Montage

Die sechste Station ist eine Simulationsunterstützte Montagestation. Der Anwender wird durch eine am Monitor erscheinende Simulation Schritt für Schritt durch die Montagekette geleitet. Die Geschwindigkeit der aufeinanderfolgenden Montageschritte passt sich an die Arbeitsgeschwindigkeit des Anwenders an.

Station 7, Qualitätskontrolle:

In der letzten Station wird überprüft, ob die Fertigung des Endproduktes ohne Fehler verlief. Ein Roboter führt den Stift dazu durch einen Laservorhang, mit dem beispielhaft der Durchmesser des gefertigten Griffstückes auf Richtigkeit geprüft wird. Dazu werden durch Auflegen des Werkstückträgers auf die vorgesehene Vorrichtung die Auftragsdaten aus der Datenbank ausgelesen und mit den aufgenommenen Messwerten verglichen. Nach erfolgreichem Vergleich werden die Bestandsdaten in der Datenbank aktualisiert.

2.3.3.2. Bearbeitungsstationen der mobilen Fabrik

Station 1, Auftragsidentifikation: Vgl. Station 1 (Generalfabrik)

Station 2, Fräsmaschine: Bei der zweiten Station der mobilen Fabrik wird das Griffstück in einer zu einer Drehmaschine umgerüsteten Fräse gefertigt. Die Besonderheit ist hierbei, dass anders als bei einer Fräse, das Werkstück rotiert und an einem Werkzeug vorbeigeführt wird.

Station 3, Kommissionierung: Vgl. Station 4 (Generalfabrik)

Station 4, Lasergravur: Vgl. Station 5 (Generalfabrik)

Station 5, Montage: Die Montagestation in der mobilen Fabrik leitet den „Monteur“ mit einer Laserprojizierten Montageanleitung Schritt für Schritt durch die Montagekette des Produktes. Wie auch bei der Montagestation der Generalfabrik, wird die Geschwindigkeit der aufeinanderfolgenden Montageschritte auf die Geschwindigkeit des Anwenders, durch die Interaktion mit der Station, angepasst.

Station 6, Qualitätskontrolle: Vgl. Station 7 (Generalfabrik)

2.3.4. Das Produkt

Abbildung 8: Personalisierter Kugelschreiber

Zur Demonstration von produktionstechnischen Technologien wird ein personalisierter Kugelschreiber hergestellt. Der Stift besteht im Inneren aus einer Mine und einer Feder. Die Schriftfarbe kann zuvor in der Konfiguration gewählt werden. Die sichtbaren Komponenten sind die Spitze, das Griffstück und der Rohling mit der Gravur, die jeweils aus Aluminium bestehen und die Verschlussschraube, die aus eloxiertem Aluminium besteht (von links nach rechts). Mit der Verschlussschraube lässt sich die Position der Mine einstellen. Neben der Schriftfarbe können Spitzform, die Form des Griffstückes, die Gravur und die Farbe der Verschlussschraube gewählt werden.

3. Stand der Technik

Industrie 4.0 steht für die vierte Generation der industriellen Revolution, deren Schwerpunkt auf der Digitalisierung der Produktion liegt. Digitalisierung bedeutet, dass Systeme intelligent, vernetzt und miteinander agieren können. Durch die Digitalisierung können individuelle Produkte zu gleicher Qualität und zu konkurrenzfähigen Preisen gegenüber der Massenproduktion produziert werden. Die folgende Abbildung 9 zeigt, wie sich die allgemeine Produktion im Laufe der Zeit, bezogen auf Stückzahl und Variation des Produktes, gewandelt hat.

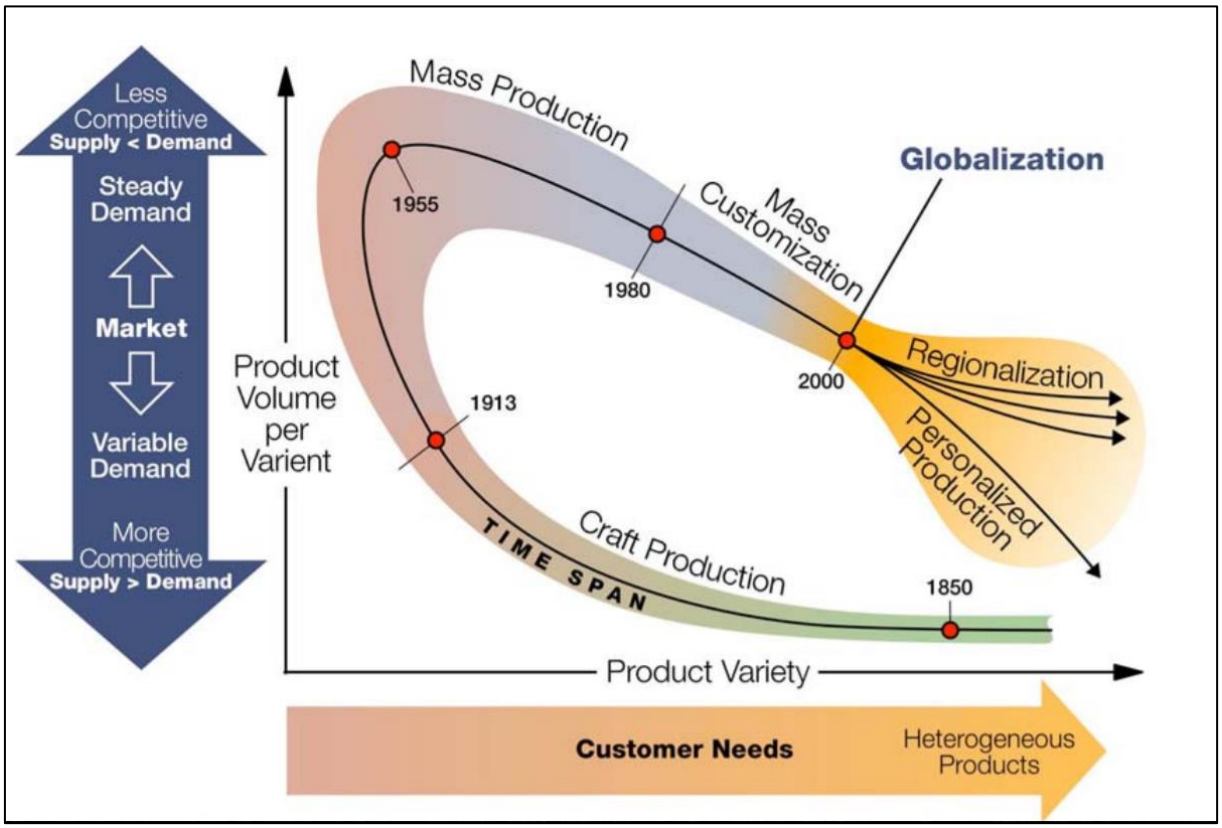


Abbildung 9: Relation Stückzahl und Variation eines Produktes [KOR13]

Zu Beginn der Industrialisierung (1850) wurde begonnen, größere Stückzahlen zu produzieren [MÜL88]. Im Jahre 1913 revolutionierte der US-Amerikaner Henry Ford mit seiner Automobil-Fabrik die Fließbandfertigung [CLA12]. Anfang der 1970er Jahre wurden zunehmend Computer in der Fertigung eingesetzt, die Rechenaufgaben in der Fertigung übernahmen. Das Produktionsvolumen pro Variante hat somit den höchstmöglichen Stand erreicht [KNA16]. Seitdem nimmt die Produktvielfalt zu, das Produktionsvolumen je Variante nimmt jedoch ab [KOE17]. Seit der Jahrhundertwende geht die Produktion intensiver auf die personalisierte Produktion ein. Durch die Digitalisierung der 4. Industriellen Revolution können Produkte in der vorhandenen Fertigung individuell gestaltet werden [BAU14].

Durch den hohen Automatisierungsgrad, den die Industrie durch die Digitalisierung erreicht hat, fallen große Datenmengen an. Diese Daten werden auch „Big Data“ genannt. Jedoch können die Datenmengen nicht ohne weiteres genutzt werden. Um die Wirtschaftlichkeit durch die Analyse der Produktionsdaten zu steigern, oder aber zusätzliche Servicedienstleistungen wie eine Produktnachverfolgung anbieten zu können, müssen diese Daten intelligent Verarbeitet werden. [SMA15]

„Smart Data“ sind Daten, die durch eine sinnvolle Analyse von „Big Data“ entstehen. Ein aktuelles Beispiel für „Smart Data“ ist die Sendungsverfolgung, die etliche Logistikunternehmen ihren Kunden zur Verfügung stellen. Mit der Sendungsverfolgung kann der Aufenthaltsort eines versendeten Paketes ermittelt werden. Das folgende Beispiel beschreibt eine Sendungsverfolgung bei DHL.

Das Tracking (die Nachverfolgung) der Sendung beginnt, wenn der Kunde die Sendung aufgibt. Bei der Aufgabe einer Sendung werden etliche Daten im System gespeichert. Dazu gehören Absender- und Empfängerdaten, sowie Ort und Datum der Sendungsaufgabe. Jedes Paket muss eine individuelle, eindeutige Identifikationsnummer bekommen, damit das sogenannte „elektronische Paketsystem“ (folgend System bezeichnet) jederzeit den Sendungsverlauf aktualisieren kann. Dazu hat das Paket einen eindeutigen Barcode, durch den der Postbote das Paket jederzeit identifizieren und der Status der Sendung entweder manuell oder automatisch im System aktualisiert werden kann. Das System stellt dem Kunden wenige Minuten später die Daten zur Verfügung, um die Sendung online nachzuverfolgen.

Die Sendungsverfolgung bietet Unternehmen wie auch für Privatkunden den entscheidenden Vorteil, sich über die fristgerechte Ankunft des Paketes zu informieren und bei Verzögerungen benachrichtigt zu werden. [BER17]

Ein weiteres Beispiel für „Smart Data“ in der Produktion sind Informationen über Produkte, die wichtige Konfigurationsparameter für die Produktion beinhalten (z.B. Farbton für die Produktlackierung). Diese Informationen laufen parallel zu den Prozessschritten der Produktion. Durch diese Informationen können ebenfalls Rückschlüsse auf den aktuellen Produktionsstand des Produktes gezogen werden. [VOG17b]

Durch „Smart Data“ können produzierende Unternehmen etliche Vorteile der Digitalisierung nutzen. Ein essentieller Vorteil ist die Transparenz der Produktion, wodurch Unternehmer und Mitarbeiter Rückschlüsse auf Effizienz und Produktivität ihrer Produktion ziehen können. Die Transparenz verhilft Unternehmern und Mitarbeitern, Prozesse besser zu verstehen. Auftretende Fehler können lokalisiert und durch Benachrichtigungen an die Mitarbeiter schneller behoben werden. Wiederkehrende Verzögerungen im Produktionsablauf werden sichtbar und können unterbunden werden. Auch der Kunde kann durch eine hohe Transparenz in der Produktion profitieren, zum Beispiel mit einer echtzeitnahen Auftragsnachverfolgung.
Funktionen, die erst durch „Smart Data“ ermöglicht werden, werden auch „Smart-Services“ genannt. Produkte, die selbst ihre Bearbeitungsschritte und Komponenten ken-

nen und sich somit selbst durch die Fertigung leiten, werden im Kontext der Digitalisierung „Smart Produkt“ genannt [HEI13]. Die BMW Group stellt Fahrzeuge (folgend Produkte genannt) nach Kundenwunsch her. Diese Produkte leiten sich selbst durch die Fertigung und besitzen alle notwendigen Produktionsdaten. Zudem hat die BMW Group im Jahr 2008 die Chance ergriffen, dem Kunden durch die Digitalisierung einen Mehrwert der immer stetig zunehmenden Komplexität der Fertigung bieten zu können. Dazu gehört, dass der Kunde bis zu sechs Tage vor Montagestart seine Bestellung ändern kann. [BAU08].

Die Chancen durch die Digitalisierung in der Produktion sind vielseitig. Dazu gehören auch die ökonomischen Chancen in produzierenden Unternehmen. Die ökonomischen Chancen sind nach dem Handbuch für Industrie 4.0 Bd.4 immens: *„Produkte sind eindeutig identifizierbar, jederzeit lokalisierbar und kennen ihre Historie, den aktuellen Zustand sowie alternative Wege zum Zielzustand. […] Die Produktion wird hochflexibel, hochproduktiv […]. Die Herstellung individualisierter Produkte zu den Kosten eines Massenprodukts wird damit Realität“. [VOG17c]*

3.1. Übersicht vorhandener Service-Apps

In diesem Kapitel wird gezielt auf Service-Apps eingegangen, da der Verbreitungsgrad der Technologie „Apps“ nicht nur im privaten Sektor, sondern auch in der Industrie stetig zunimmt [STA17b]; [VOG17b]. Die Service-Apps sollen Unternehmern und Mitarbeitern den Arbeitsalltag erleichtern und Kunden zusätzliche Dienstleistungen bieten.

In der Logistikbranche haben sich Apps etabliert, um Sender und Empfänger über den Verbleib eines versendeten Produktes zu informieren. Die bekannten Logistikunternehmen wie „DHL“, „Hermes“, „UPS“, und Andere stellen Apps zur Sendungsnachverfolgung zum Download in App-Stores bereit. [DHL17]; [HER17]; [UPS17]

Zudem gibt es Apps, die Sendungen dienstleisterunabhängig nachverfolgen können [PAR17]. Bei der App des Onlinehandels „Amazon“ wird die Funktion der unternehmensunabhängigen Sendungsverfolgung als zusätzliche Dienstleistung neben dem Onlineshop angeboten [AMA17]. Die deutsche Bahn bietet ebenfalls Apps für ihre Kunden an. Die am häufigsten[2] verwendete App der deutschen Bahn ist „DB Navigator“. In dieser App hat der Kunde die Möglichkeit, minutengenau aktualisierte Fahrpläne einzusehen [DB 17]. Auch Apps zur Produktionsnachverfolgung gehören zum Stand der Technik. Die Firma Bego ist Spezialist für Zahntechnik und bietet ihren Kunden eine App zur echtzeitnahen Produktnachverfolgung an. Nach einer Anmeldung mit einer Kundennummer und einem Passwort hat der Kunde die Möglichkeit seinen Auftrag nachzuverfolgen [BEG17]. Die Firma Lenze SE bietet zu ihren Maschinen eine App an, mit der über die Funkübertragungstechnologie „NFC“ Parameter angepasst und die Maschinen konfiguriert werden können [LEN17]. SAP bietet seinen Kunden eine App an, mit der die gesamte Ressourcenplanung und Organisation des Unternehmens gesteuert werden kann. Dazu gehört das Prüfen von Lagerbeständen, wie auch Alarme, die auf einen Fehler (z.B. in der Produktion) hinweisen [SAP17]. Voraussetzung für die Verwendung der App ist ein ERP-System von SAP.

Die aufgezählten Beispiele haben gezeigt, dass es schon etliche Apps gibt, die den Mitarbeiter in seiner Arbeit unterstützt (vgl. Maschinenkonfiguration Lenze SE) und Unternehmer seine Prozesse von Unterwegs steuern kann (vgl. SAP Business One App). Auch dem Kunden kann durch die Digitalisierung zusätzliche Vorteile geboten werden. Sei es die Sendungsnachverfolgung bei Logistikunternehmen oder die Echtzeitfahrpläne der deutschen Bahn.

[2] Gemessen an der Anzahl der Downloads

3.2. Betriebssysteme für mobile Anwendungen

Grundlage für eine App ist ein Betriebssystem, was eine mobile Anwendung (umgangssprachlich „App“) ausführen kann. In diesem Kapitel werden die beiden meist verbreitetsten Betriebssysteme für mobile Anwendungen, wie auch eine Entwicklungsvariante für betriebssystemunabhängige Apps aufgezeigt und verglichen.

3.2.1. Android

Android ist ein Betriebssystem des Unternehmens Google Inc., welches hauptsächlich für mobile Endgeräte entwickelt wurde. Zu mobilen Endgeräten gehören heute schon nicht mehr nur Smartphones und Tablets, sondern auch Smart-TVs [GOO17c] und Smart-Watches [GOO17e], welche in dieser Arbeit jedoch keine Bedeutung haben. Die erste Version des Betriebssystems Android kam im Oktober 2008 auf den Markt. Heute hat sich bereits die 7. Generation des Android-betriebssystems etabliert. Jede Generation von Android-Betriebssystemen ist zudem durch verschiedene Versionen gegliedert. Für Entwickler ist jedoch die API-Version von großem belangen. Sie gibt den Entwicklungsstand der unterstützten Methoden und Funktionen an. Werden Funktionen oder Methoden in einer App-Entwicklung genutzt, die z.B. erst in der API-Version 18 bereitgestellt werden, können die Android-Versionen 4.2 und kleiner diese Funktion nicht unterstützen, da sie nicht über die minimale API-Version (in diesem Fall 18) verfügen. In der folgenden Tabelle (Tabelle 4) sind die aktuellen Versionen des Android-betriebssystems mit ihrer aktuellen[3] prozentualen Verteilung aufgelistet.

Tabelle 4: Versionsübersicht Android [GOO17d]

Version	Codename	API	Distribution
< 4.1		< 16	1,4%
4.1	Jelly Bean	16	2,7%
4.2		17	3,8%
4.3		18	1,1%
4.4	KitKat	19	16,0%
5.0	Lollipop	21	7,4%
5.1		22	21,8%
6.0	Marshmallow	23	32,3%
7.0	Nougat	24	12,3%
7.1		25	1,2%

[3] Stand 08.08.2017

3.2.2. iOS

iOS ist ein Betriebssystem des Unternehmens Apple Inc. und ist für mobile Endgeräte ihrer eigens entwickelten Produkte vorgesehen. Seit der Markteinführung im Januar 2007 wurde bis heute ununterbrochen an der Weiterentwicklung des Betriebssystems gearbeitet. In der folgenden Tabelle (Tabelle 5) sind die aktuell verbreiteten iOS-Versionen mit ihrer aktuellen prozentualen Verteilung aufgelistet. [KOL11]

Tabelle 5: Versionsübersicht iOS [APP17]

Version	**Distribution**
IOS 8 und kleiner	3 %
iOS 9	10 %
IOS 10	87 %

Auffällig bei der Versionsübersicht von iOS ist, dass der größte Teil der Endgeräte die neueste Betriebssystem-Generation nutzen.

Bei einem Vergleich der Betriebssysteme Android und iOS fällt auf, dass beide seit der Markteinführung stetig an der Weiterentwicklung der Betriebssysteme arbeiten. Jährlich wird von beiden Unternehmen eine neue Generation des jeweiligen Betriebssystems auf den Markt gebracht. Deutlich sind auch die unterschiedlichen Verteilungen auf die einzelnen Betriebssystemversionen zu erkennen. Bei iOS sind 87 % der Endgeräte auf dem neuesten Stand. Bei Android sind es lediglich 13,5 %, die die neueste Generation des Betriebssystems nutzen.

Folgend werden Zahlen der Betriebssysteme für mobile Anwendungen ausgewertet. Die Auswertung soll zeigen, welchen Marktanteil die Betriebssysteme iOS und Android im Bereich der mobilen Endgeräte haben.

3.2.3. Marktanteile von Betriebssystemen für mobile Endgeräte

Die Zielgruppe der App beschränkt sich hauptsächlich auf Kunden aus Deutschland. Somit werden in diesem Kapitel hauptsächlich deutsche Statistiken ausgewertet.

Nach aktuellen Zahlen der Smartphone-Betriebssysteme ist das Betriebssystem Android sowohl in Deutschland mit 70 % (vgl. Abbildung 10), als auch Weltweit mit 73,4 % Marktanteil die wichtigste Plattform für Smartphone-Apps. Das zweithäufigste Betriebssystem für Smartphones, sowohl Weltweit mit einem Marktanteil von 18,8 % als auch in Deutschland mit 27 %, ist iOS (vgl. Abbildung 10) Gemeinsam machen diese Betriebssysteme rund 97 % der Smartphone-Betriebssysteme in Deutschland und 92,2 % Weltweit aus. [STA17a]

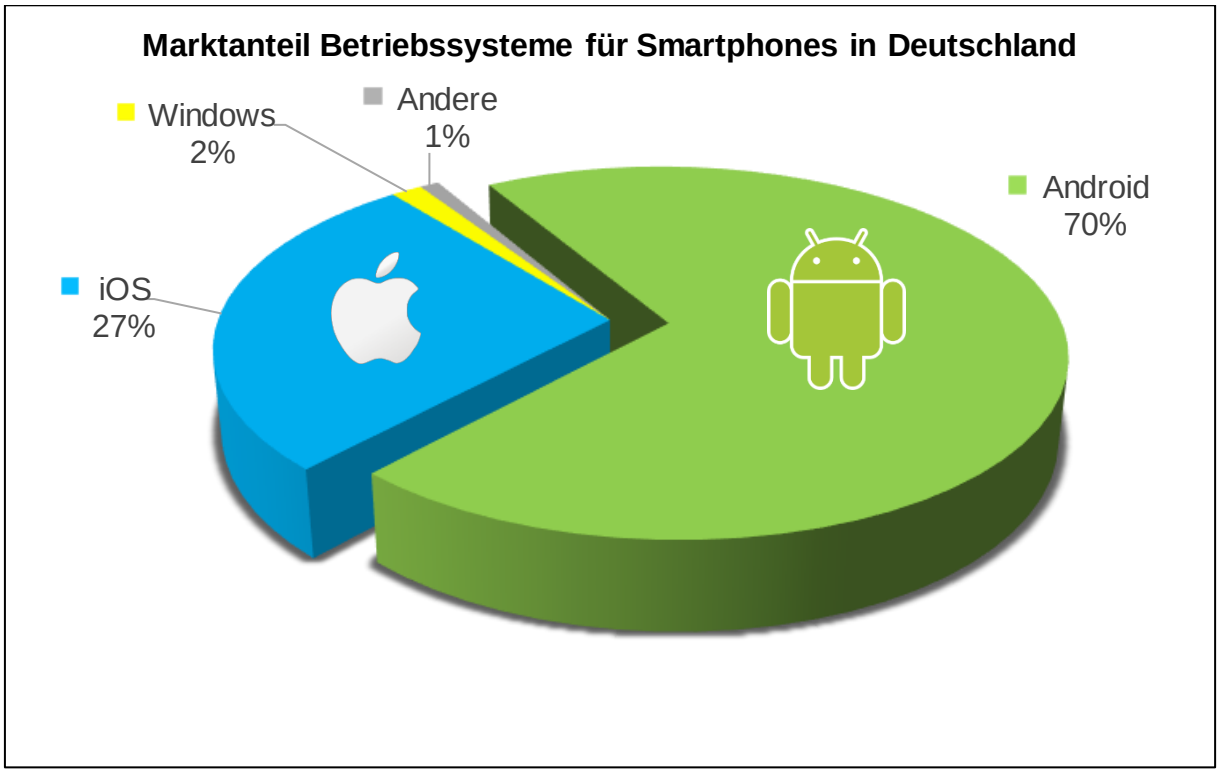


Abbildung 10: Marktanteil Betriebssysteme für Smartphones in Deutschland [STA17a]

Bei Tablet-Betriebssystemen fällt die Verteilung der Marktanteile mit deutlichem Unterschied zu den Marktanteilen von Betriebssystemen für Smartphones aus. Hierbei spielt das Betriebssystem Android mit 37% Marktanteil in Deutschland (vgl. Abbildung 11) und mit 35% Weltweit eine geringere Rolle, als bei den Smartphone-Betriebssystemen. Gleichwohl ist es das zweithäufigste Betriebssystem für Tablets, sowohl in Deutschland, als auch Weltweit. IOS spielt bei Tablets hingegen eine wichtigere Rolle. 62% aller Tablets in Deutschland (vgl. Abbildung 11), und 65% weltweit verfügen über ein IOS-Betriebssystem.

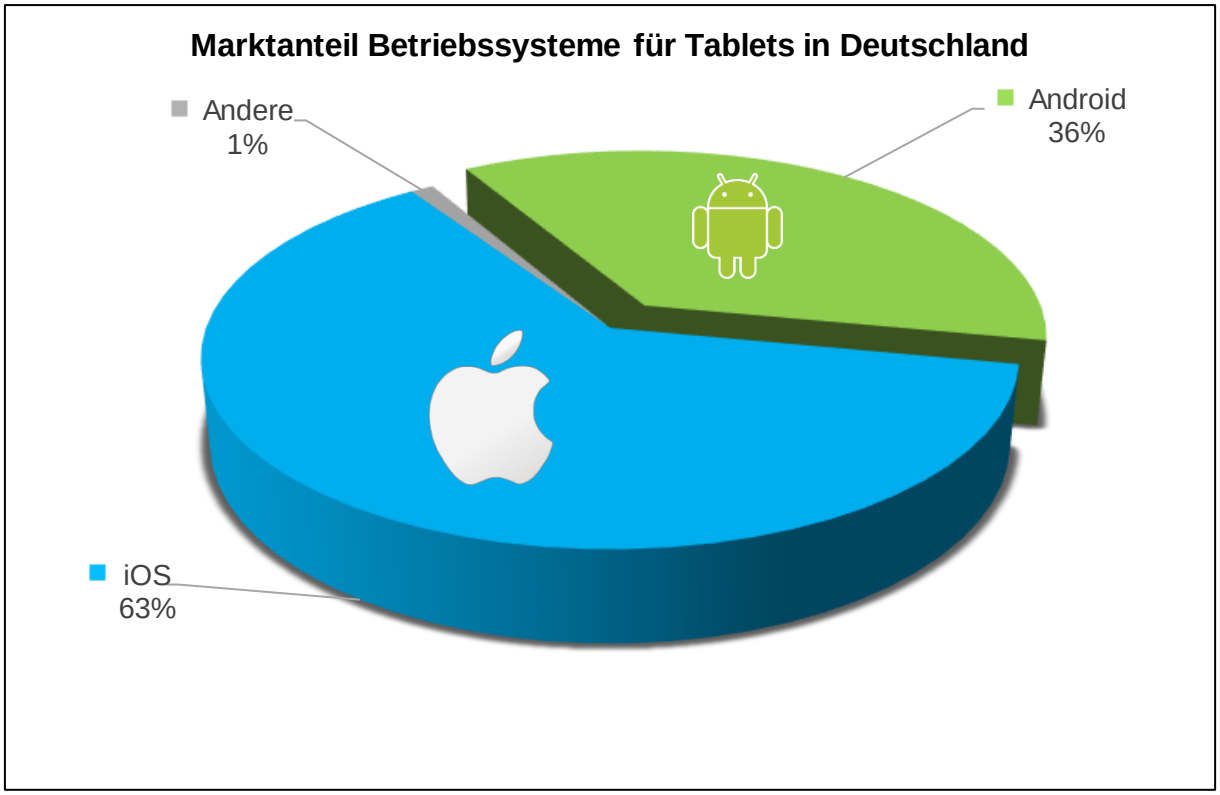


Abbildung 11: Marktanteil Betriebssysteme für Tablets in Deutschland [STA17a]

3.2.4. Progressive Web App

Seit einigen Jahren wird das Thema Progressive Web App immer stärker behandelt [STE17a]; [STE17b]. Hierbei handelt es sich um nicht native Apps, die vollständig plattform- und geräteunabhängig entwickelt und genutzt werden können. Progressive Web Apps sind Webanwendungen, die auf einem Webserver laufen und mit einem beliebigen mobilen Endgerät, oder anderen Geräten, die über einen Internetbrowser verfügen, aufgerufen werden können. Ein Icon als Schnellstart der Webanwendung kann auf dem Endgerät hinterlegt werden.

Zu den Nachteilen der Verwendung von Progressive Web Apps zählt einerseits die Performance von Internetbrowsern. Im Vergleich zu nativen Apps sind sie langsamer, da sich die gesamte App nicht auf dem Gerät befindet. Andererseits sind sie an eine dauerhafte Netzwerkverbindung gebunden. Zudem ist die Verwendung von Systemfunktionen, wie der Verwendung der Kamera, ist im Vergleich zu nativen Apps schwierig und zeitaufwendig. [STE17a]

4. Entwicklung der App

Das folgende Schaubild verdeutlicht die Entwicklungsphasen der App, beginnend mit der Analyse der Anforderungen, über die Implementierung bis hin zur Veröffentlichung der App (siehe Abbildung 12).

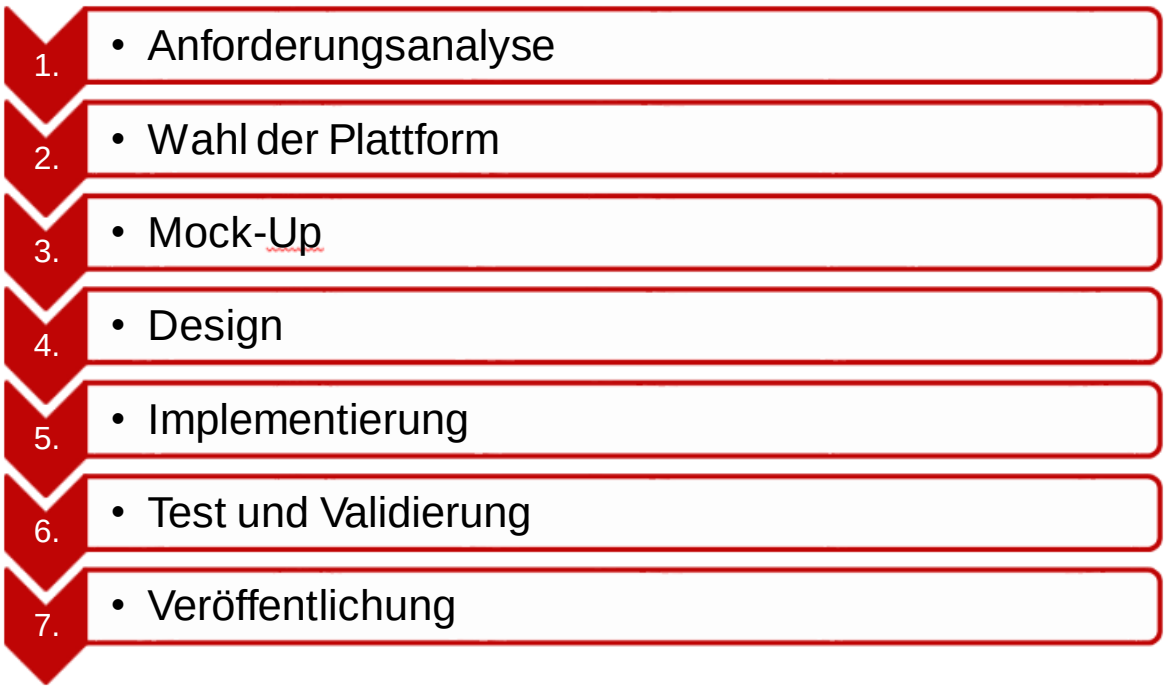


Abbildung 12: Phasen der App-Entwicklung

4.1. Anforderungsanalyse

Im ersten Band vom Handbuch Industrie 4.0 wird *„Anhand eines Beispiels […] aufgezeigt, wie in einem produzierenden Unternehmen eine Lösung zur lückenlosen digitalen und echtzeitnahen Nachverfolgung von Produktionsprozessen eingeführt werden kann.*“ [VOG17a] Nicht beachtet wird in dem Beispiel die Möglichkeit der Auftragsänderung während des Produktionsvorganges. Um neben der echtzeitnahen Produktionsnachverfolgung dem Kunden die Möglichkeit zu bieten, aktiv in den Fertigungsprozess einzugreifen, um seine Aufträge zu ändern, müssen folgende Punkte beachtet werden.

Als erstes ist eine Produktion vorausgesetzt, die ihre Daten zentral verwaltet. Produktions- bzw. Fertigungs- und Kundendaten müssen jederzeit, auch außerhalb der Fertigung erreichbar sein. Kundendaten müssen verschlüsselt übertragen werden, um nichtautorisierte Auftragsänderungen oder Produktionsdatenabfragen zu verhindern. Dazu ist eine Verschlüsselung der Daten notwendig. Hierfür gibt es verschiedenste Verschlüsselungsmethoden, wie z.B. AES[4], welche vom Bundesamt für Sicherheit in der Informationstechnik für Industrieanwendungen empfohlen wird [BUN13b]. Als weiteren Punkt wird die Flexibilität der Fertigung vorausgesetzt. Nicht jede Fertigung kann flexibel sein, sodass eine Auftragsänderung unmittelbar vor Beginn möglich ist. Damit die Produktion flexibel genug für die Auftragsänderung ist, müssen die Lagerbestände und somit auch die Kapitalbindung erhöht werden. Jedes Teil muss zu jeder Zeit vorrätig sein. Werden einzelne Komponenten für das Produkt von extern eingekauft, kann für diese Komponenten keine Auftragsänderung gewährleistet werden.

Funktionale Anforderungen

Neben den Anforderungen an das Produktionssystem, werden auch Anforderungen an das zu entwickelnde System gestellt. Diese funktionalen Anforderungen teilen sich auf zwei Zielgruppen auf. Die eine Zielgruppe setzt sich aus Kunden zusammen (siehe Abbildung 13, links). Durch Bereitstellen von Produktionsdaten können dem Kunden zusätzliche Dienstleistungen wie einer Produktnachverfolgung in der Fertigung und einer Auftragsänderung nach Bestelleingang anbieten. Um die nötigen Parameter der Fertigung ändern zu können, muss sichergestellt sein, dass der jeweilige Produktionsprozess noch nicht begonnen hat. Zudem muss sich der Anwender authentifizieren können, um nichtautorisierte Änderungen auszuschließen. Als dritte Aufgabe soll mit der App ein Auftrag erstellt werden können, um einen neuen Auftrag in Form eines QR-Codes zu erstellen. Eine weitere Möglichkeit soll darin bestehen, die Webseite des Kompetenzzentrums aus der App heraus öffnen zu können.
Die andere Zielgruppe setzt sich aus Unternehmern zusammen (siehe Abbildung 13, rechts). Der Unternehmer kann durch die Transparenz der Produktion in gleicher Weise Aufträge nachverfolgen. Zudem kann der Unternehmer Fehler, die in der Pro-

[4] Advanced Encryption Standard

duktion auftreten, identifizieren. Neben den Möglichkeiten kann auch eingesehen werden, welcher Auftrag an welcher Station bearbeitet wird und ob die durchschnittlichen Durchlaufzeiten eingehalten wurden.

Alle funktionalen Anforderungen sind zur Veranschaulichung, aufgeteilt auf die beiden Nutzergruppen, in folgendem Use-Case-Diagramm (Abbildung 13) dargestellt.

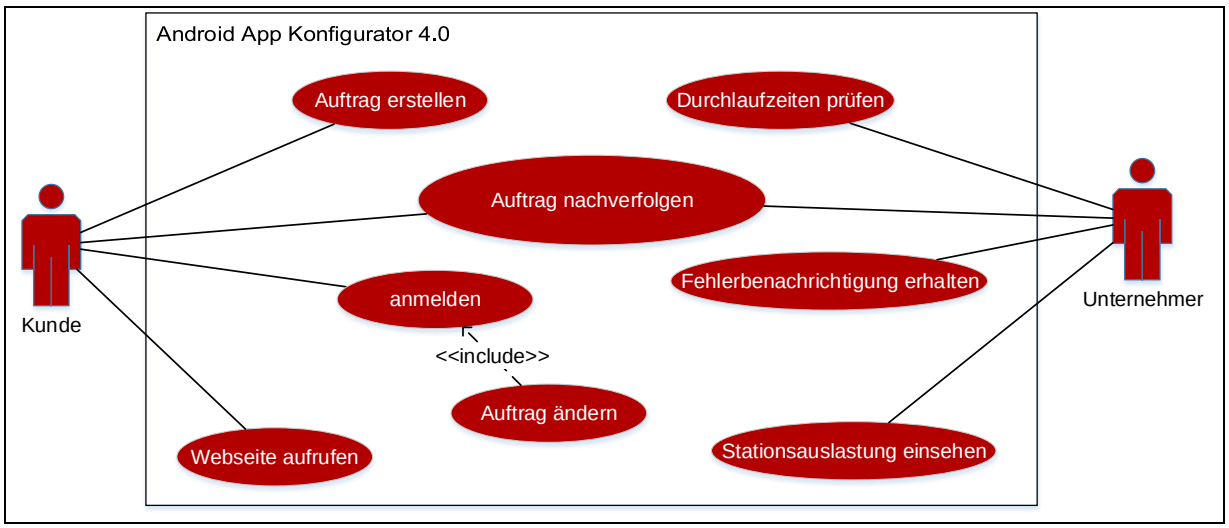


Abbildung 13: Use-Case-Diagramm der App

Nichtfunktionale Anforderungen

Zu den nichtfunktionalen Anforderungen der App gehört die Systemunterstützung von möglichst vielen Endgeräten. Hinzu kommt, dass die Bezugsquelle der App vertrauenswürdig sein muss. Dies bedeutet, dass die App nur über die offiziellen App-Stores anzubieten ist. Eine weitere nichtfunktionale Anforderung ist die Gewährleistung des Datenschutzes. Alle Produktionsdaten, die für die Nutzung dieser App notwendig sind, müssen verschlüsselt und gegen Mitlesen und Manipulation geschützt werden. Die App soll Ressourcensparend entwickelt werden. Damit ist die speicherplatzoptimierte Entwicklung, wie auch die Einsparung von großen Datenmengen, die über das Internet kommuniziert werden müssen und die effektive Nutzung von Systemressourcen (geringe CPU-Auslastung, Einsparung von Arbeitsspeichernutzung) gemeint.

4.2. Wahl der Plattform

In der folgenden Tabelle 6 sind Anforderungen gegenüber verschiedenen Appentwicklungsmethoden aufgelistet, die zur Wahl der Plattform beigetragen haben.

Tabelle 6: Anforderungen an die App

Methoden zur Appentwicklung / Anforderungen & Kriterien	Internetzugriff	Datenspeicher	Kamera	QR-Code Generierung	Effizienz	Zeitaufwand	Kosten Veröffentlichung[5]	Abdeckung Marktanteile DE[6]	Grundkenntnisse vorhanden
Progressive Web App	Ja	Ja	Ja	Ja	-	o	-	>99%	Nein
Multiplattform-Appentwicklung	Ja	Ja	Ja	Ja	o	++	~~€€€~~	>99%	Nein
Klassische Appentwicklung Android	Ja	Ja	Ja	Ja	++	o	€	~70%	Ja
Klassische Appentwicklung iOS	Ja	Ja	Ja	Ja	++	O	~~€€~~ bis ~~€€€~~	~27%	nein
- Schlecht, niedrig	++	Sehr gut, hoch			o	Okay, normal			
€ günstig	~~€€~~	erschwinglich			~~€€€~~	teuer			

[5] [GOO17f]; [APP17]

[6] [STA17a]; [STE17a]; [STE17b]

Eine Progressive Web App wurde aus dem Kreis der Möglichkeiten ausgeschlossen, da die App Anmeldedaten und Produktdaten dauerhaft gespeichert werden soll. Eine Speicherung in einer Progressive Web App ist möglich, jedoch geschieht dies im Speicher des installierten Internetbrowsers des mobilen Endgerätes.

Eine Progressive Web App benötigt zudem eine dauerhafte breitbandige Internetverbindung, um eine stabile Funktionalität bieten zu können. Mit einer nativen App kann im Vergleich zu webbasierten Apps die Anzahl der Netzwerkzugriffe minimiert werden, da die App auf dem Gerät installiert wurde. Alle notwendigen Funktionen, wie z.B. das Erstellen oder Scannen von QR-Codes können ohne Internetverbindung genutzt werden. Eine Progressive Web App kann nicht im Google Play Store angeboten werden. Somit müssten Installationen von Apps aus unsicheren Quellen zugelassen werden. [YEE17]

Für eine flächendeckende Betriebssystemunterstützung ist eine Cross Plattform Entwicklung möglich. Hierbei wird betriebssystemunabhängig auf Basis von HTML 5, CSS und JavaScript entwickelt [STE17b]. Ein großer Nachteil ist, dass die „*Umsetzung der grafischen Benutzungsoberfläche Schwächen [aufweist], sodass die GUI oftmals für jedes mobile Betriebssystem separat entwickelt werden muss*“ [VOL17]. Daher wäre ein erhöhter Zeitaufwand für das Design der verschiedenen Apps notwendig. Zudem kommen die hohen Kosten, die mit einer Veröffentlichung für verschiedene Betriebssysteme anfallen würden.

Obwohl eine Progressive Web App-Entwicklung oder eine Multiplattform-Appentwicklung 99 % der mobilen Endgeräte in Deutschland abdecken würden, ist die Wahl auf Grund der oben genannten Schwachstellen auf die Entwicklung einer klassischen nativen App gefallen. Nach den aktuellen Zahlen der Marktanteile ist die Entscheidung auf die Entwicklung einer Service-App für Android-Endgeräte gefallen, da über zweidrittel der Smartphone-nutzer in Deutschland Android verwenden (siehe Kapitel 3.2). Die App kann somit nach dem Installieren aus dem Google Play Store [GOO17f] jederzeit auf dem Handy oder auch auf dem Tablet mit dem Betriebssystem Android genutzt werden.

Für die Entwicklung der Android-App wird Android Studio mit der Version 2.3.2 verwendet. Android Studio ist die gängigste Entwicklungsumgebung für Androidanwendungen. Zudem wird es neben dem Betriebssystem direkt von Google kostenfrei zur Verfügung gestellt. [GOO17b]

4.3. Mock-Up

Eine gängige Methode bei der Entwicklung einer App ist die Erstellung eines Prototyps (auch Mock-Up genannt). Ein Mock-Up beinhaltet nur die grundlegendsten Funktionen. Dazu gehört hauptsächlich das Navigieren zu den verschiedenen Activities der App. Layout und Design sind hierbei nebensächlich.

Das Mock-Up wurde nach dem folgenden Struktogramm entwickelt, in dem alle grundlegenden Activities und ihrer Zusammenhänge aufgezeigt werden (siehe Abbildung 14).

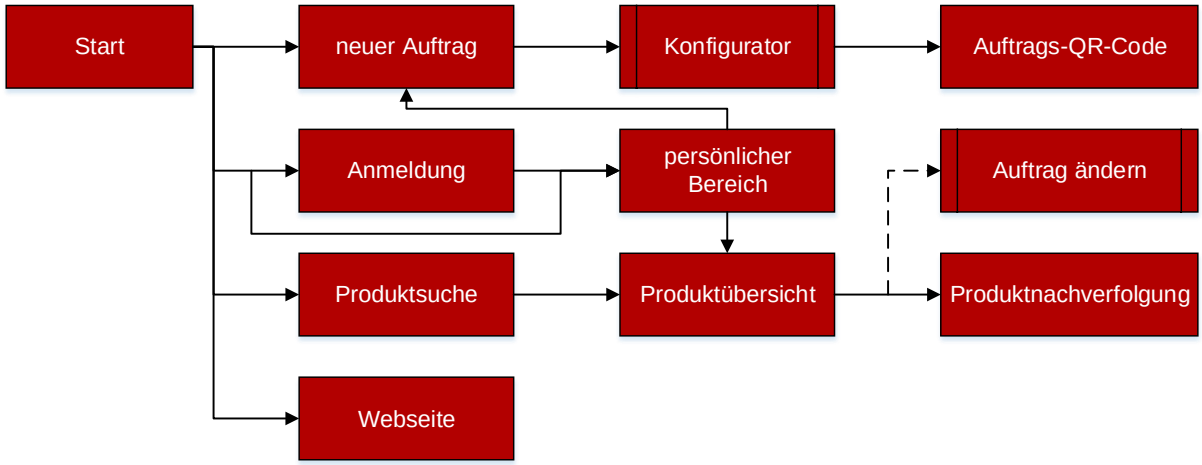


Abbildung 14: Struktur der App

Die Elemente *Konfigurator* und *Auftrag ändern* werden später aus mehreren Activities bestehen, die im Mock-Up vereinfacht als eine einzelne Activity dargestellt wurden.

Das Hauptmenü besteht aus vier Buttons, mit denen zu den Activities *Bestellung aufgeben, Anmeldung, Produktsuche* und *Webseite* weiternavigiert werden kann. In der Activity *Bestellung aufgeben* kann mit Hilfe zweier Buttons entschieden werden, für welche Fabrik der Auftrag erstellt werden soll. Durch klicken der Buttons wird man weiternavigiert zur Activity *Konfigurator*, der ebenfalls einen Button enthält. Klickt man diesen Button, wird die Activity *Konfigurator-QR-Code* gestartet. Beim Start der Activity wird ein QR-Code in der Mitte des Bildschirms angezeigt, der mit Hilfe der Open-Source-Bibliothek ZXing („zebra crossing“) [OPE17] aus einem String erzeugt wird. Der angezeigte QR-Code im Mock-Up beinhaltet lediglich Beispieldaten.

In der Activity *Anmeldung* kann durch Eingeben eines Nutzernamens und eines Passwortes Die Activity *persönlicher Bereich* gestartet werden. Die eingegebenen Daten werden mit einem im Quellcode hinterlegten Beispielbenutzer (bestehend aus Nutzernamen, Passwort und drei Beispielaufträgen) abgeglichen. Bei einmaliger erfolgreicher Anmeldung wird dieser Schritt übersprungen. Der Anmeldestatus wird im Hintergrund der App gespeichert.

Im Persönlichen Bereich werden die drei Beispielaufträge aufgelistet. Mit Klick auf den jeweiligen Auftrag wird man in die Activity *Produktübersicht* weitergeleitet.

In der Activity *Produktübersicht*, die ebenfalls über die Activity *Produktsuche* gestartet werden kann, werden alle Produktdaten (siehe Abbildung 31, vgl. Kapitel 4.5.1.2 „Objektklassen“) in Tabellenform aufgelistet. Unter der Auflistung sind zwei Buttons zu finden, mit denen zur Activity *Produktnachverfolgung* und *Auftragsänderung* weiternavigiert werden kann. Die Funktion der Auftragsänderung ist nur dann aktiviert, wenn die Activity *Produktübersicht* über den persönlichen Bereich gestartet wurde. Hintergrund hierbei ist, dass eine Auftragsänderung nur nach erfolgreicher Anmeldung möglich ist. Die Activities *Produktnachverfolgung* und *Auftragsänderung* bleiben im Mock-Up ohne Funktionen. Bei der Activity *Produktsuche* kann mit Hilfe einer numerischen Tastatur eine Produktnummer gesucht werden (in diesem Fall nur einer der drei Beispielaufträge). Bei erfolgreicher Suche wird die Activity *Produktübersicht* gestartet.

Die letzte Möglichkeit ist, dass durch das klicken auf den Button „Webseite“ eine Activity gestartet wird, in der die Webseite des Kompetenzzentrums angezeigt wird. Diese Vorgehensweise wurde im weiteren Verlauf der Appentwicklung effizienter gestaltet (vgl. Kapitel 4.5.4.2).

4.4. Design

4.4.1. Design der Activities

Die Layouts aller Activities basieren auf der gleichen Grundlage eines *Themes* (Designthema). Das *Theme* gibt das grobe Design, wie auch die verwendeten Farben, Schriftgrößen und Schriftarten vor. Die drei Navigationsbuttons im unteren Bereich des mobilen Endgerätes bleiben dauerhaft aktiv. So kann der Bediener jederzeit zurücknavigieren oder die App verlassen.

In der folgenden Abbildung ist der Theme Editor von Android Studio mit dem erstellten *Theme* zu sehen. Darin befindet sich die App-Bar oben links, die auf jeder Activity mit dem Titel der App zu sehen ist. Des Weiteren befinden sich im linken Teil der Abbildung Designs für Prozessbars, Texte oder Buttons. Im rechten Teil der Abbildung sind verschiedene Farben aufgelistet. Diese Farben können mit Hilfe ihrer eindeutigen ID in jedem Layout verwendet werden. Somit kann gewährleistet werden, dass alle Layouts die gleichen Farben verwenden.

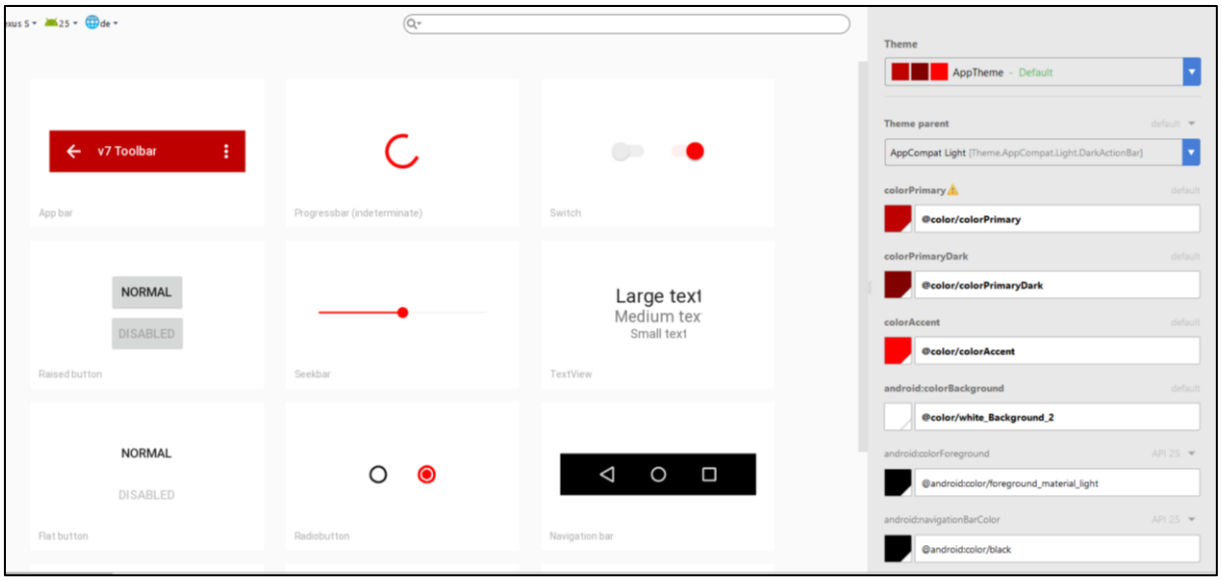


Abbildung 15: Android Studio Theme Editor

4.4.2. Design der Icons

Für die hohe Usability der App muss eine einfache Navigierbarkeit der App vorhanden sein. Zudem müssen die Komponenten der App übersichtlich und verständlich sein. Aus diesem Grund wurden beschriftete Felder mit Icons entwickelt (siehe Anhang 6). Die Vorlagen für die erstellten und verwendeten Icons sind aus der Onlinecommunity „Pixabay“ entnommen oder selbst erstellt. In dieser Community werden Grafiken zur kostenfreien Nutzung unter Creative Commons CC0 Lizenz veröffentlicht. [PIX17]

4.4.3. Design der Auftragsnachverfolgungskomponenten

Es wurden sechs verschiedene Anzeigevarianten für die Visualisierung der Auftragsnachverfolgung entwickelt. Die Visualisierungskomponenten sind der folgenden Tabelle 7 zu entnehmen.

Tabelle 7: Visualisierungskomponenten für die Auftragsnachverfolgung

Icon, grüner Haken	
Der Grüne Haken steht für das Bearbeitungsende einer Station.	
Icon, gelbe Uhr	
Die Gelbe Uhr bedeutet, dass die Bearbeitung an der jeweiligen Station noch nicht begonnen hat.	
Icon, rotes Kreuz	
Das rote Icon mit dem weißen Kreuz deutet auf einen Fehler im Produktionsablauf oder auf eine fehlerhafte Bedienung der jeweiligen Station hin.	
Anzeige, Uhr + „in Arbeit“	
Eine weitere einfache Anzeigevariante ist die Kombination aus dem gelben Icon mit der Uhr und dem Zusatztext „in Arbeit“, was darauf hinweist, dass sich die aktuelle Station in Bearbeitung befindet.	
Anzeige, Prozessbar	
Für vollautomatisierte Stationen, deren Durchlaufzeiten berechenbar oder konstant sind, werden mit der Prozessbar animiert. In der Prozessbar kann man den aktuellen Bearbeitungs- oder Produktionsprozess in Prozent ablesen. Die genaue Funktion wird in Kapitel 4.5.2 erläutert.	

Anzeige, Quotient	
Für die Station der Kommissionierung wurde eine Variante zur Visualisierung entwickelt, mit der man die Anzahl der entnommenen Teile gegenüber der Anzahl der zu entnehmenden Teile ablesen kann. Die genaue Funktion wird im Kapitel 4.5.2 erklärt.	

Aktuell bearbeitete Stationen werden mit einem roten Rahmen umrandet (vgl. Tabelle 7). Das Design der Visualisierungskomponenten wurden in drei Layoutdateien entwickelt (*stationrowimage.xml für die Visualisierung eines Icons*, *stationrowprogressbar.xml* für die Visualisierung einer Prozessbar und *stationrowteiler.xml für die Visualisierung einer Kommissionierstation*), wobei alle auf der unten abgebildeten Basis aufbauen (siehe Abbildung 16).

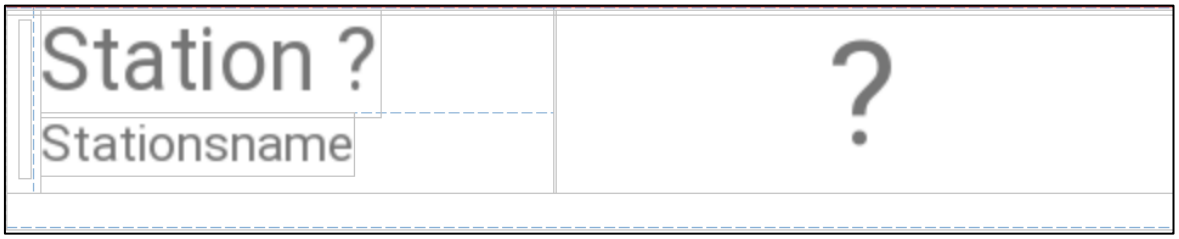


Abbildung 16: Basislayout für Stationslayouts

Das Basislayout besteht aus mehreren Layoutelementen. Die Layoutelemente haben eine eindeutige ID, um sie im funktionalen Teil der Activity anpassen zu können.

Die Stationsnummer und der Stationsname (oben links im Bild), die Stationsbeschreibung (unten links im Bild) und eine Anzeigevariante, sei es eine Prozessbar, ein Quotient oder ein Icon, können angepasst werden. Je nach Element wird eine andere Layoutdatei als Grundlage genommen, da in diesen die Spezifikationen der jeweiligen Elemente gespeichert sind.

4.5. Implementierung

In diesem Kapitel wird die finale Umsetzung zur „Entwicklung einer mobilen Anwendung zur Produktnachverfolgung im Fertigungsprozess“ behandelt.

Nach dem Entwickeln eines Mock-Ups und der Designphase wurde begonnen, die Funktionalität der App herzustellen. Dazu wurde ein Webserver entwickelt, der die Kommunikation zwischen der App und der Datenbank des Produktionssystems übernehmen soll. Eine Kopie der aktuellen Datenbank wurde während der Entwicklungsphase auf einem lokalen PC unter Microsoft SQL Server 2014 installiert. Der Webserver wurde auf demselben lokalen PC mit der Entwicklungsumgebung Eclipse entwickelt (siehe Kapitel 4). Der lokale PC spannte ein WLAN-Netz auf, mit dessen Hilfe Testgeräte mit dem Webserver kommunizieren konnten. Die Daten in der Datenbank wurden manuell überschrieben, um alle Funktionen der App (Anmelden, Produktsuche, Produktnachverfolgung, Auftragsänderung) zu testen. Nach erfolgreichem Testen mit dem Testaufbau wurde der Webserver in das Produktionssystem Integriert und bekam Zugriff auf die Produktionsdatenbank.

4.5.1. Datenkommunikation

Um die Sicherheit der Datenkommunikation zwischen der App und der Datenbank gewährleisten zu können, wurde ein Webserver entwickelt, der die Kommunikation zwischen App und Datenbank absichert, regelt, protokolliert und die Daten direkt in geeigneter Form zur Verfügung stellt. Die Produktionsdaten im Produktionssystem sind in einer SQL-Datenbank in Tabellen, welche nach Kundendaten und Auftragsdaten aufgeteilt sind, gespeichert. Aus Sicherheitsaspekten besitzt die Datenbank keinen Anschluss an das Internet, um sie vor Angriffen zu schützen.

Sowohl die Datenbank, als auch der Webserver, befinden sich auf einem Server, deren Recheneinheiten voneinander unabhängig auf Basis von virtuellen Maschinen laufen. Lediglich eine Schnittstelle ermöglicht die Kommunikation zwischen dem Webserver und der Datenbank.

Einen Datenbankzugriff aus der App heraus soll folgendes Schaubild (Abbildung 17) verdeutlichen.

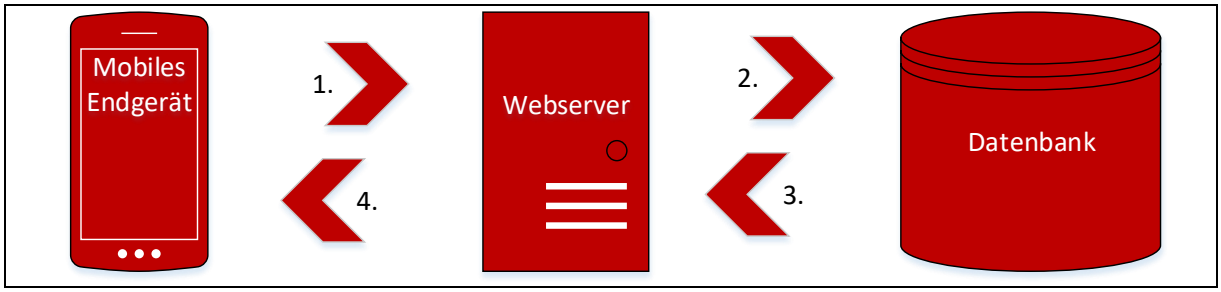


Abbildung 17: mobiler Datenbankzugriff

Der Datenbankzugriff über den Webserver geschieht in vier Schritten:

1. Die App sendet dem Webserver eine http-Anfrage. Die http-Anfrage beinhaltet die Information, was angefragt wird (Auftrag, Kunde, Events). Des Weiteren enthält die http-Anfrage Attribute, die für den Webserver zur Suche erforderlich sind (zum Beispiel eine Auftragsnummer bei der Anfrage eines Auftrages).
2. Der Webserver überprüft die Syntax der Anfrage auf Richtigkeit. Ist die Syntax in Ordnung, fragt der Webserver die gewünschten Daten per SQL-Statement in der Datenbank ab.
3. Gibt es einen Eintrag für das übergebene Attribut (wie zum Beispiel die Auftragsnummer), bekommt der Webserver die Daten zurück und erstellt daraus ein xml-Dokument. Sollte es für das Attribut keinen Eintrag geben, wird ein leeres Dokument erstellt (vgl. Kapitel 4.5.1.3).
4. Der Webserver antwortet auf die http-Anfrage mit dem erstellten Dokument, welches die gewünschten Informationen beinhaltet. Diese sind aus Sicherheitsaspekten verschlüsselt (vgl. Kapitel 4.5.1.3).

4.5.1.1. Webserver

Das folgende Use-Case-Diagramm (Abbildung 18) zeigt die grundsätzlichen Anfragen, die der Webserver verarbeiten kann. Neben dem Webserver ist die Datenbank als weiteres System abgebildet.

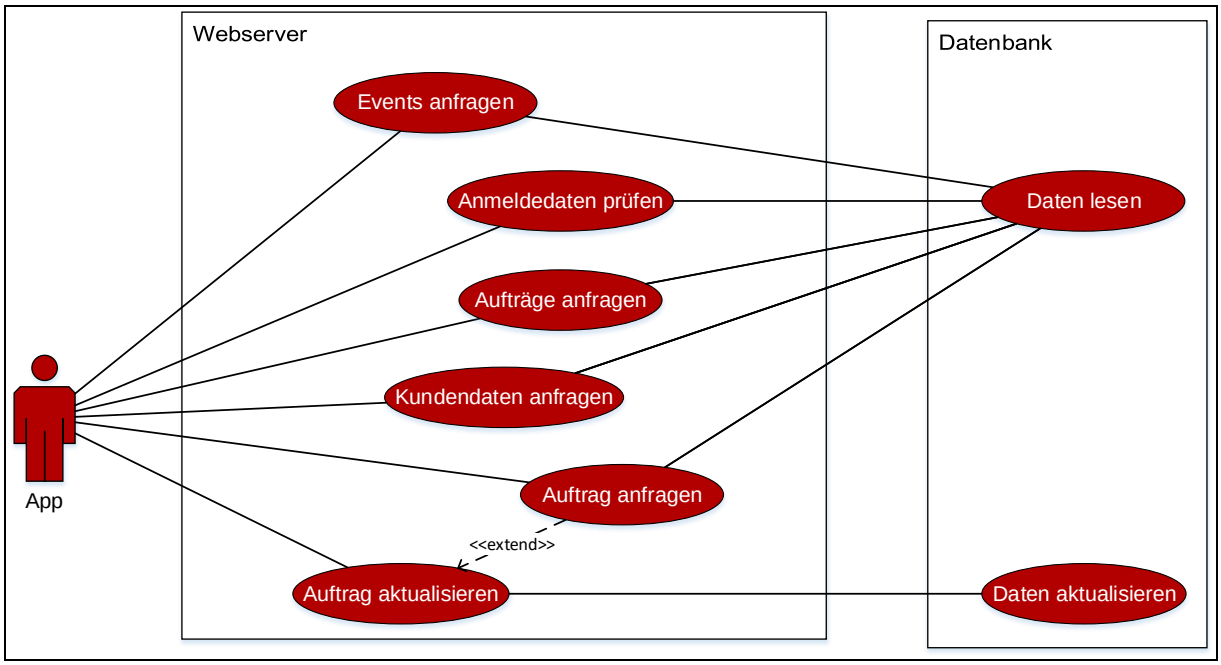


Abbildung 18: Use-Case-Diagramm des Webservers

Über eine http-Anfrage kann die App verschiedene Daten anfordern. Dazu gehören Events, Auftragsdaten oder Kundendaten. Zudem kann über http eine Anfrage zur Prüfung von Anmeldedaten gesendet werden. Zuletzt können durch das gleiche Prinzip Auftragsdaten aktualisiert werden. Jede Anfrage, Prüfung oder Aktualisierung wird mit einer xml-Datei vom Webserver beantwortet (vgl. Kapitel 4.5.1.3).

Der Webserver ist ein Java-Servlet-Projekt. Servlets sind Werkzeuge von Java, mit denen dynamische Webanwendungen entwickelt werden können. Als Entwicklungsumgebung für den Webserver wurde die Entwicklungsplattform „Eclipse“ mit der Version „Luna 4.4.2“ verwendet. Der Webserver wurde in einem „Tomcat-9.0“-Server im Produktionssystem integriert.

Die Anfrageparameter der http-Anfrage werden im Webserver zu einem SQL-Statement umgewandelt (siehe Anhang 5). Durch das SQL-Statement können Daten aus der Datenbank ausgelesen oder aktualisiert werden. Das folgende Klassendiagramm zeigt alle Methoden der erstellten HSQL-Klasse des Webservers, in denen die Methoden für die Datenbankkommunikation zu finden sind (siehe Abbildung 19).

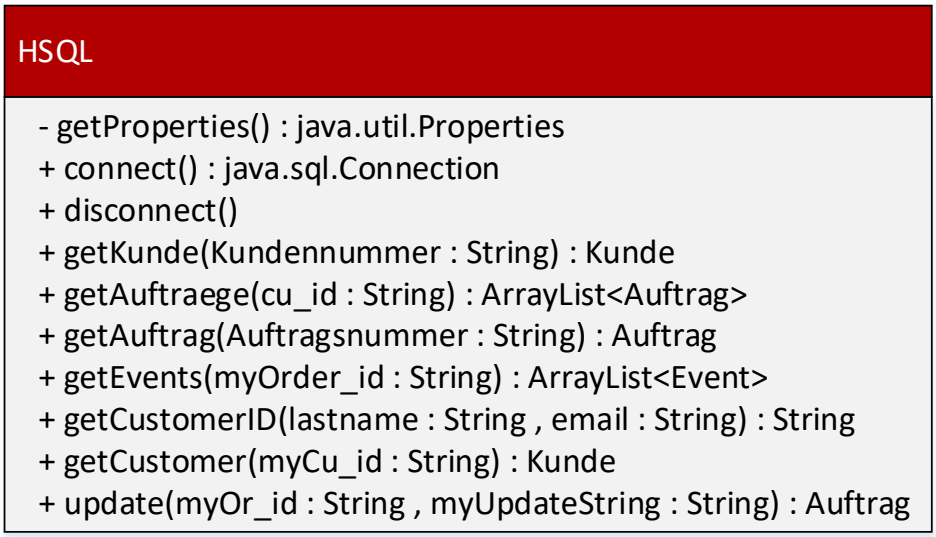


Abbildung 19: UML-Klassendiagramm Webserver, HSQL-Klasse

Die Methoden *getProperties()*, *connect()* und *disconnect()* sind für die Verbindung mit der Datenbank zuständig. In den anderen Methoden werden die http-Anfragen der App in ein SQL-Statement umgewandelt, mit deren Hilfe die passenden Daten aus der Datenbank ausgelesen oder aktualisiert werden (vgl. Anhang 7). Wenn eine http-Anfrage mit den Attributen *customer* und *cu_id* mit einer gültigen Kundennummer (sechsstellig) empfangen wird, wird die Methode getKunde() aufgerufen. In der Methode werden Kundendaten aus der Datenbank ausgelesen und in einer XML-Datei als Kundentags (vgl. Kapitel 4.5.1.3) gespeichert. Die Methode *getKunde()* wird aufgerufen, wenn eine http-Anfrage mit den Attributen *customer* und *cu_id* mit einer gültigen Kundennummer (sechsstellig) empfangen wird. Als Antwort wird eine XML-Datei erstellt, die einen Kundentag mit den ausgelesenen Parametern beinhaltet (vgl. Kapitel 4.5.1.3). Die Methoden *getAuftrag()* und *getAuftraege()* geben Auftragsdaten zurück. Sie werden aufgerufen, wenn die http-Anfrage die Attribute *order* oder *orders* beinhaltet. *getEvents()* wird aufgerufen, wenn eine http-Anfrage mit den Attributen *events* und *or_id* mit einer gültigen Auftagsnummer (sechsstellig) empfangen wird. Als Antwort sendet der Webserver eine XML-Datei, die alle Events der übergebenen Auftragsnummer zugehörig sind, beinhaltet. Die Methode *getCustomerID()* wird aufgerufen, wenn eine http-Anfrage mit den Attributen *customer*, *lastname* und *email* empfangen wird. Wird zu der Kombination aus Nachname und Email ein Kundeneintrag in der Datenbank gefunden, wird eine XML-Datei erstellt, die die Kundennummer beinhaltet (Bei einer Antwort mit Kundennummer bedeutet dies gleichzeitig, dass die Anmeldung für den persönlichen Bereich erfolgreich war). Wenn Kundendaten gefordert sind, beinhaltet die http-Anfrage die Attribute *customer* und *cu_id* und die *getCustomer()* wird aufgerufen. Die letzte Methode der Klasse heißt *update() und* wird aufgerufen, wenn Auftragsdaten geändert werden sollen. Die http-Anfrage kann die Attribute *pti_id*, *plc_id*, *or_shape*, *or_engraving* oder *pco_id* beinhalten, die jeweils für eine Komponente des zu fertigenden Produktes stehen.

4.5.1.2. Objektklassen

Da in der App, sowie im Webserver, Daten gleichen Typs verarbeitet werden, eignet sich eine objektorientierte Softwareentwicklung, die bei Android mit Java bereits gegeben ist.

Um die Daten in der App, als auch im Webserver gleichermaßen verarbeiten zu können, werden in jedem Softwareprojekt die Klassen *Event*, *Auftrag* und *Kunde* identisch entwickelt. Diese Klassen sind sogenannte Objektklassen (siehe Abbildung 20, oben), mit deren Hilfe Objekte mit hinterlegten Daten angelegt werden können. Neben den Attributen und dem Konstruktor besitzt jede Klasse für jedes Attribut get- und set-Methoden mit dem Modifikator *public*, um die initialisierten leeren Objekte mit Daten füllen und sie lesen zu können. In der folgenden Abbildung wurden diese Methoden vernachlässigt, da sie nicht zum Verständnis der Verwendung der Klassen beitragen. Das Androidprojekt besitzt zudem zwei weitere Objektklassen; die Klassen *Fabrik* und *Station* (siehe Abbildung 20, vgl. Anhang 1).

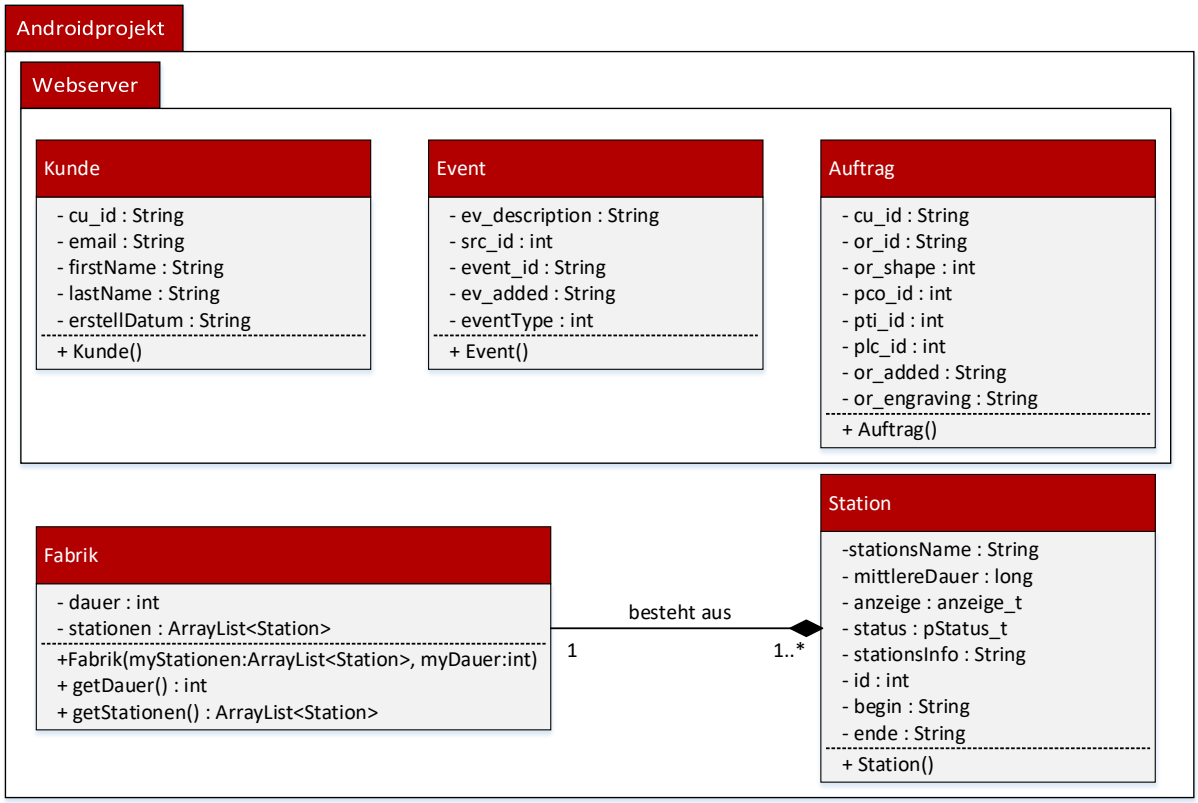


Abbildung 20: UML-Klassendiagramm der Objektklassen

Die Klassen Fabrik und Station werden für die Funktion der Produktnachverfolgung benötigt und in Kapitel 4.5.3.4 „Produktnachverfolgung“ genauer beschrieben._Die Klassen Kunde, Event und Auftrag stehen in keinem direkten Zusammenhang im objektorientierten Softwareprojekt. Die Klassen Fabrik und Station hängen mit einer Komposition zusammen, d.h. dass ein Objekt der Klasse Fabrik nur dann bestehen kann, wenn eine Station existiert. Dass eine Station ohne Fabrik existiert ist ebenfalls nicht möglich, da nur eine Fabrik Stationen beinhalten kann.

4.5.1.3. XML-Dateiformat als Schnittstelle

Für den plattformunabhängigen Austausch von Daten wird das Dateiformat XML verwendet. Dazu wurden Tags definiert, die alle Attribute eines Klassenobjektes der Objektklassen *Auftrag*, *Kunde* und *Event* beinhalten (Vgl. Kapitel 4.5.1.2).

Folgend sind drei beispielhafte Order-Tags abgebildet, die von einem äußeren Orders-Tag umgeben sind (siehe Abbildung 21).

```
<!-- Auftrags-Tag -->
<orders>
    <order or_id="123456" or_added="24.12.2016 12:00" or_shape="-12"
        pco_id="2" plc_id="2" pti_id="1" or_engraving="Musterstift"/>
    <order or_id="123457" or_added="24.12.2016 12:01" or_shape="-11"
        pco_id="3" plc_id="3" pti_id="3" or_engraving="Martins Stift"/>
    <order or_id="123458" or_added="24.12.2016 12:05" or_shape="1"
        pco_id="3" plc_id="1" pti_id="2" or_engraving="Daniels Stift"/>
</orders>
```

Abbildung 21: Schnittstelle Webserver - App / Auftragstag

Folgend kann die Struktur eines Event-Tags an Hand von fünf Beispieleinträgen entnommen werden (siehe Abbildung 22).

```
<!-- Event-Tag -->
<events>
    <event ev_id="300001" ev_added="24.12.2016 12:00:01.400"
        eventtype="1" src_id="2" ev_description="Fab_order_started"/>
    <event ev_id="300002" ev_added="24.12.2016 12:00:46.234"
        eventtype="1" src_id="2" ev_description="Fab_order_finished"/>
    <event ev_id="300004" ev_added="24.12.2016 12:01:43.144"
        eventtype="1" src_id="3" ev_description="Fab_turning_started"/>
    <event ev_id="300005" ev_added="24.12.2016 12:02:00.778"
        eventtype="1" src_id="3" ev_description="Fab_turning_moved"/>
    <event ev_id="300007" ev_added="24.12.2016 12:03:13.029"
        eventtype="1" src_id="3" ev_description="Fab_turning_finished"/>
</events>
```

Abbildung 22: Schnittstelle Webserver - App / Eventtag

In der folgenden Abbildung (Abbildung 23) kann die Spezifikation eines Kunden-Tags entnommen werden.

```
<!-- Kunden-Tag -->
<customers>
    <customer cu_id="123456" cu_email="Mustermann@email.de" cu_firstname="Max"
        cu_lastname="Mustermann" cu_added="24.12.2016 12:00"/>
</customers>
```

Abbildung 23: Schnittstelle Webserver - App / Kundentag

Zum Auslesen der XML-Datei des Webservers wurde eine Klasse *Parser* entwickelt. (vgl. Anhang 5). Die Attribute der Datei werden in der Klasse ausgelesen und in einem Objekt der jeweiligen Klasse gespeichert (vgl. Kapitel 4.5.1.2).

4.5.1.4. Sicherheit

Die Daten im XML-Dokument sind zudem nach der Methode AES[7] verschlüsselt, um das Mitlesen im Netzwerk zu erschweren. Die Wahl für die Verschlüsselungsmethode ist durch die Empfehlung des Bundesamtes für Sicherheit in der Informationstechnik getroffen worden [BUN13b].

Bei der Auftragsänderung muss für die Authentifizierung am Webserver ein Authentifizierungsschlüssel übergeben werden. Um potentiellen Angreifern keine Möglichkeit der Entschlüsselung und Manipulation der Auftragsdaten zu bieten, findet eine ständige Aktualisierung des Authentifizierungsschlüssels statt. Der zweite Punkt ist das Anlegen eines Nutzerkontos für den Web-Server. Damit der Webserver auf die Daten zugreifen kann, muss sich der Webserver in regelmäßigen Abständen am Datenbanksystem mit seinem Benutzerkonto und dem dazugehörigen Passwort Authentifizieren. Der dritte Punkt ist die Rechteverteilung der Benutzerkonten. Das Benutzerkonto des Web-Servers hat nur auf notwendige Daten Leserechte und ausschließlich für die aktuellen Stammdaten schreibrechte, um bei Auftragsänderung die Stammdaten anzupassen.

[7] Advanced Encryption Standard

4.5.2. Events

Für die reibungslose Produktionsnachverfolgung sind Events notwendig. Events sind Datenbankeinträge, die Vorgänge im Produktionsprozess, wie z.B. „Station Kommissionierung gestartet“ beschreibt. Für die Entwicklung der App wurden dazu diese Events für jede Bearbeitungsstation spezifiziert (vgl. Anhang 4). Jedes Event besteht aus einer eindeutigen Event-ID, einer dazugehörigen Auftragsnummer (Produkt-ID), einer Source-ID, die die Stations-ID wiederspiegelt, sowie einen Beschreibungstext und einem Zeitstempel, der das Hinzufügen des Events datiert. Stationen, die automatisiert ablaufen und dadurch vorhersehbare Durchlaufzeiten besitzen, benötigen neben dem Start-Event und dem Ende-Event ein weiteres Event. Das Start-Event wird bei jeder Station beim Positionieren des Werkstückträgers auf die vorgesehene Halterung geschrieben. Das Ende-Event wird geschrieben, wenn der Werkstückträger entnommen wurde. Voraussetzung hierfür ist aber die erfolgreiche Beendigung der Station.

Ein Problem der Darstellung in der Produktionsnachverfolgung der automatisierten Stationen ist, dass die Prozessbar (vgl. Kapitel 4.4.3) beim Start-Event beginnt, welches beim Auflegen des Werkstückträgers geschrieben wird. Der automatisierte Prozess beginnt jedoch erst nach Interaktion mit dem Bediener. Hierfür wurde für die Interaktion bei den Stationen der Qualitätssicherung und der Drehmaschine bzw. Fräsmaschine (sowohl mobile Fabrik, als auch Generalfabrik) ein weiteres Event, dem „move“-Event spezifiziert. Somit kann eine genauere Prozessnachverfolgung für automatisierte Prozesse gewährleistet werden.

Ausschlaggebend für die Visualisierung ist das zuletzt geschriebene Event zu der Station und dem Produkt. Handelt es sich um ein „finish“-Event, wird der Status der Station auf „done“ gesetzt, was im späteren Verlauf des Programmes zum grünen Haken als Statussymbol führt. Zudem wird die Endzeit der Station auf den Zeitstempel des Events gesetzt, damit neben dem Statussymbol die Produktionsdauer an dieser Station ermittelt und angezeigt werden kann. Ist das Event ein „move“-Event, wird der Status der Station auf „move“ gesetzt, welche im weiteren Verlauf des Programmes die Prozessbar aktiviert. Neben dem Status wird auch die Startzeit der Station auf die Zeit des Events gesetzt. Die Startzeit ist zum einen für die Animation der Prozessbar und zum anderen für die spätere Auswertung der Produktionsdauer notwendig. Wenn das Event ein „start“-Event ist, wird die Station in den Status „in“ versetzt. Zum Schluss gibt es noch das Event „cancel“. Bei dem Event „cancel“ wird die jeweilige Station in den Zustand „err“ gebracht. Im weiteren Verlauf des Programmes wird ein rotes Icon mit einem weißen Kreuz angezeigt, was auf einen Fehler bei der Bearbeitung an der Station steht.

Jedes Event hat in der Beschreibung noch eine Abkürzung, die für die jeweilige Fabrik steht. Ein beispielhaftes Event der Generalfabrik ist „Fab_robot_started“; ein Event der mobilen Fabrik als Beispiel „Bus_robot_started“ (vgl. Anhang 4).

In der folgenden Abbildung (Abbildung 24) ist ein Ausschnitt des Quellcodes zu sehen, der zeigt, wie die Stationsstatus durch die Events aktualisiert und beschrieben werden.

```
for (Station st : myFabrik.getStationen()) {
    for (Event ev : myEventList) {
        if (st.getstationID() == ev.getSrc_id()) {
            Log.i("EventList", ev.getSrc_id() + " " + ev.getEv_description());

                //ist das letzte Event "finish"
            if (ev.getEv_description().contains("finished")) {
                st.setStatus(ownDataTypes.pStatus_t.done);
                st.setEnde(ev.getEv_added());

                //ist das letzte Event "move"
            }else if(ev.getEv_description().contains("move") &&st.getStatus()!= ownDataTypes.pStatus_t.done){
                st.setStatus(ownDataTypes.pStatus_t.move);
                st.setBegin(ev.getEv_added());

                //ist das letzte Event "started"
            }else if (ev.getEv_description().contains("started") && st.getStatus() != ownDataTypes.pStatus_t.done) {
                st.setStatus(ownDataTypes.pStatus_t.in);
                //sollte die Station keine Progressbar als Anzeigetyp haben
                //wird die Startzeit geschrieben
                if(st.getAnzeige()!= ownDataTypes.anzeige_t.PROGRESSBAR){
                    st.setBegin(ev.getEv_added());
                }

                //ist das letzte Event "cancel"
            }else if (ev.getEv_description().contains("cancel") && st.getStatus() != ownDataTypes.pStatus_t.done) {
                st.setStatus(ownDataTypes.pStatus_t.err);
            }
        }
    }
```

Abbildung 24: Quellcode Event-Verwaltung

Die äußere for-Schleife durchläuft jede Station, die die aktuelle Fabrik besitzt und speichert das temporäre Objekt der Klasse Station unter dem Namen „st“. Die innere for-Schleife durchläuft alle Events, die zu dem aktuellen Auftrag in der Datenbank gespeichert sind. In der inneren Schleife wird zuerst geprüft, ob die aktuelle Stations-ID mit der Source-ID des Events übereinstimmt. Stimmen diese IDs überein, werden folgende if-clauses durchlaufen.

Wenn das aktuelle Event ein finish-event ist, wird die Endzeit der Station mit dem Zeitstempel des finish-Events überschrieben und der Status der Station wird auf „done“ gesetzt. Handelt es sich um ein „move“-Event, der Status aber noch nicht „done“, wird der Status auf „move“ gesetzt und die Startzeit der Station wird mit dem Zeitstempel des „move“-Events überschrieben. Bei den Events „start“ und „cancel“ funktioniert dies mit dem gleichen Prinzip. (vgl. Abbildung 24)

Die Stationen der Kommissionierung hat neben dem „finish“-Event ein spezifiziertes „start“-Event und weitere Events um die Visualisierung des Kommissionierprozesses optimal ausführen zu können (vgl. Abbildung 25). Abhängig von der Anzahl der zu entnehmenden Teile werden die Events in der Station erstellt. Die Spezifikationen der Events für die Kommissionierung können der folgenden Tabelle 8 entnommen werden.

Tabelle 8: Eventspezifikation Kommissionierstation

Startevent	com_startet_0/N
Aktualisierungsevent	com_update_M/N
Endevent	com_finished
Legende: *M = Anzahl der entnommenen Teile* *N=Anzahl der zu entnehmenden Teile*	

Mit den Events ist es möglich, unterschiedliche Teilmengen und Gesamtmengen zu viualisieren, da diese je nach produzierendem Produkt variieren können.

Im folgenden Ablaufplan (Abbildung 25) wird eine beispielhafte Visualisierung in Abhängigkeit der Events gezeigt. Die linke Spalte der Abbildung zeigt den Ablauf der Bearbeitungsschritte, die zufolge haben, dass das System Events schreibt (siehe mittlere Spalte der Abbildung), die eine Aktualisierung der Visualisierung zufolge haben. Insgesamt müssen in diesem Beispiel vier Teile entnommen werden.

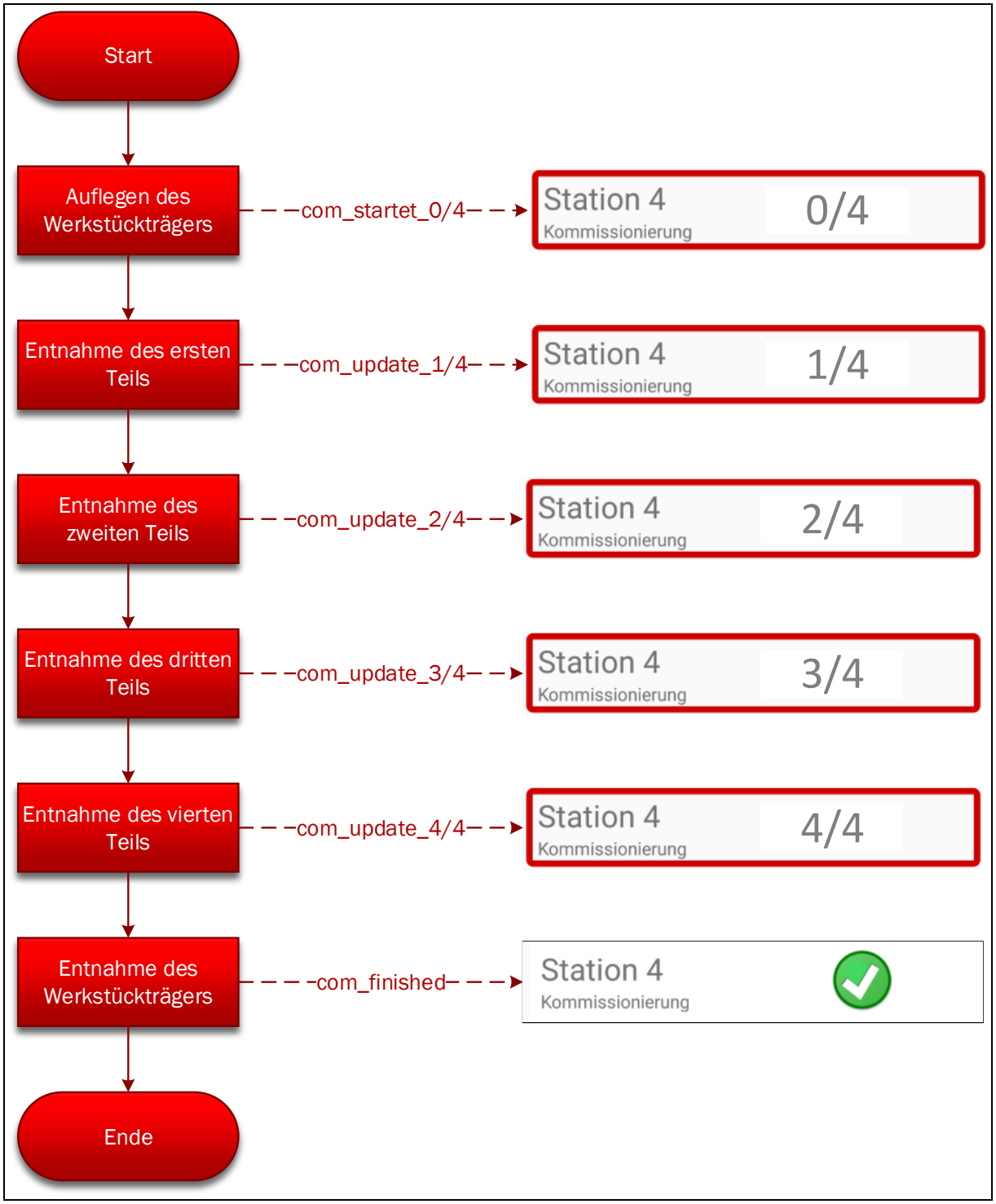


Abbildung 25: Beispiel Events & Visualisierung der Station Kommissionierung

4.5.3. Die App

In diesem Kapitel wird die App durch Abbildungen und Struktogramme erläutert. Die Grundlegende Struktur der App hat sich zu der des Mock-Ups (vgl. Abbildung 14) nicht geändert . Lediglich die Funktionalität und das Design der Activities sowie die Activities für den Konfigurator und der Bestelländerung sind hinzugekommen, welche Bestandteil dieses Kapitels sind.

4.5.3.1. Hauptmenü

Das Hauptmenü ist die erste Activity, die nach dem Start der App aufgerufen wird. Im Hauptmenü selbst kann man durch beschriftete Felder mit integrierten Icons weiter navigieren (siehe Abbildung 26).

Abbildung 26: Konfigurator 4.0 / Hauptmenü

Im Hauptmenü hat der Anwender die Möglichkeit eine Bestellung aufzugeben. Dafür muss der Konfigurator mit einem Klick auf das Feld mit dem Stiftsymbol gestartet werden. Aus dem Persönlichen Bereich (Siehe Kapitel 4.5.3.5) hat der Anwender die Möglichkeit, alle zu ihm gehörigen Aufträge einzusehen (vgl. Abbildung 36), sie zu ändern (vgl. Kapitel 4.5.3.6) oder nur den Fertigungsprozess des Auftrages nachzuverfolgen (vgl. Kapitel 4.5.3.4). Dieser kann mit einem Klick auf das Feld „persönlicher Bereich“ gestartet werden.

Über die Produktsuche kann der Anwender wählen, ob die Eingabe einer Auftragsnummer manuell oder per QR-Code erfolgen soll. Nach der Eingabe der Auftragsnummer können, die Auftragsparameter eingesehen und die Produktion nachverfolgt werden. Gestartet werden kann die Produktsuche durch einen Klick auf das Feld mit dem Lupensymbol. Die Webseite des Kompetenzzentrums kann mit einem Klick auf das Feld mit dem Haussymbol im Standardbrowser des Gerätes geöffnet.

4.5.3.2. Der Konfigurator

Der Konfigurator besteht aus mehreren, aufeinanderfolgenden Activities, die den Anwender Schritt für Schritt bis zum vollständig konfigurierten Produkt begleitet. Das folgende Schaubild (Abbildung 27) zeigt die Struktur des Konfigurators.

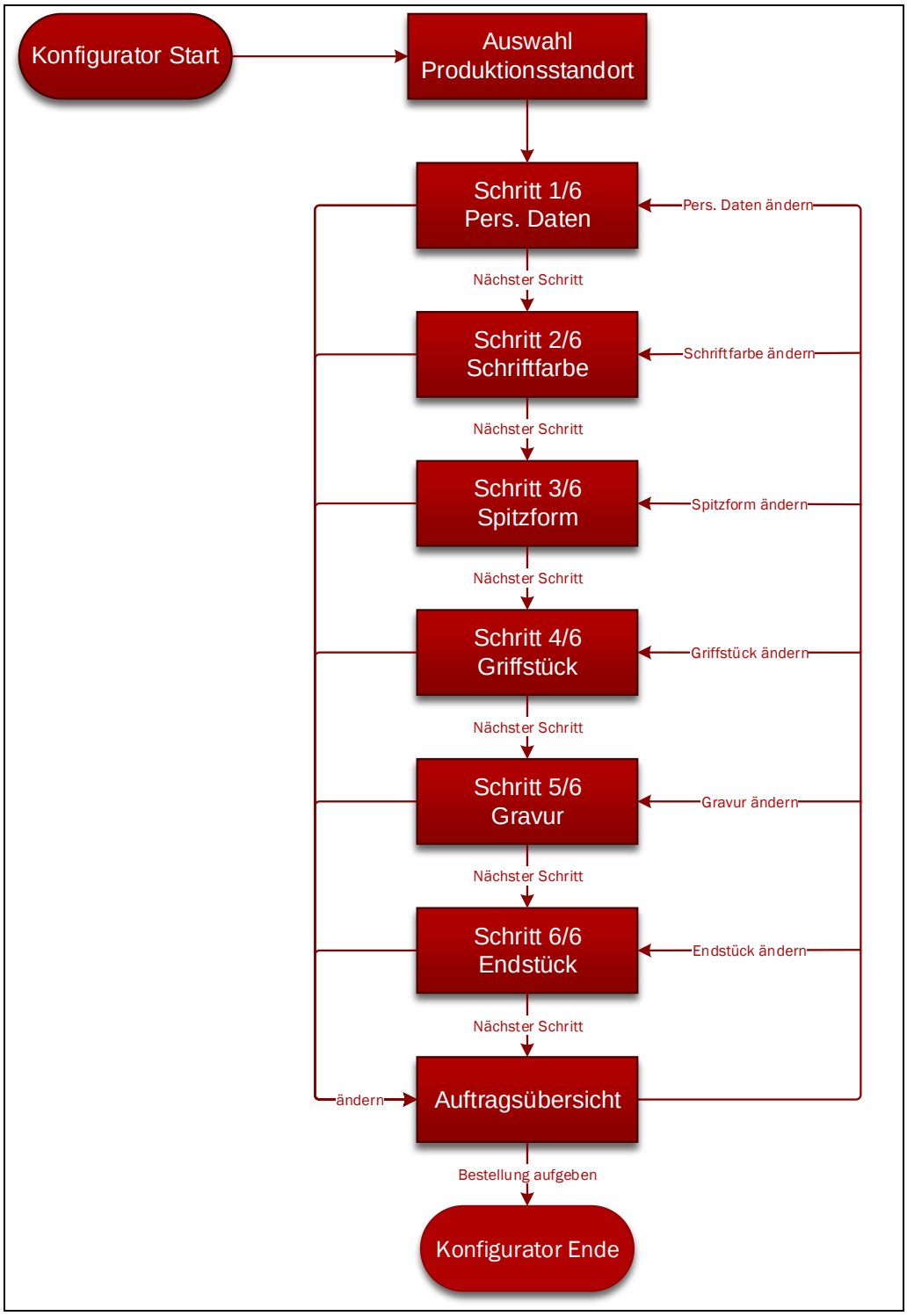


Abbildung 27: Struktur des Konfigurators

Der Struktur ist zu entnehmen, dass das Produkt schrittweise konfiguriert werden kann. Nach jedem Schritt kann zum nächsten Schritt weiternavigiert werden (vgl. Abbildung 27, Mittlerer Strang der Abbildung). In der Auftragsübersicht hat der Anwender die Möglichkeit die jeweiligen Einträge aus den Schritten zuvor zu ändern (vgl. Abbildung 27, Rückführungen auf der rechten Seite). Zur Änderung der Einträge wird die Activity des jeweiligen Schrittes gestartet, jedoch mit dem Text „ändern“ auf dem Feld, mit dem die Änderung übernommen wird und die Activity *Auftragsübersicht* wieder startet (vgl. Abbildung 27, Rückführung auf der linken Seite).

Nach dem Starten des „Konfigurators“ im Hauptmenü wird der Anwender in der darauffolgenden Activity *Auswahl Produktionsstandort* (vgl. Abbildung 27) aufgefordert, den Produktionsstandort zwischen der mobilen Fabrik und der Generalfabrik zu wählen (siehe Abbildung 28).

Abbildung 28: Konfigurator / Auswahl Produktionsstandort

Nach der Wahl des Standortes wird der Anwender Schritt für Schritt durch den Konfigurator geleitet (vgl. Abbildung 27).

Folgend werden die sechs Schritte des Konfigurators erläutert. Abbildungen zu den einzelnen Activities sind dem Anhang 2 zu entnehmen.

Im ersten Schritt werden persönliche Daten des Kunden gefragt, um ein neues Kundenregister in der Datenbank anzulegen, sofern noch kein Eintrag mit diesen Daten existiert. Wird der Konfigurator aus dem persönlichen Bereich der App gestartet, werden die persönlichen Daten automatisch eingetragen und sind nicht mehr editierbar. Um den nächsten Schritt der Konfiguration starten zu können, müssen folgende Kriterien erfüllt sein:

1. Vor- und Nachname müssen jeweils mindestens zwei Zeichen lang sein.
2. Die E-Mailadresse muss mindestens ein „@“-Zeichen und einen Punkt beinhalten und muss insgesamt mindestens 8 Zeichen lang sein.

Sollten die Kriterien dennoch nicht erfüllt worden sein, wird der Bediener mit einem Toast auf die fehlerhafte Eingabe hingewiesen und die Activity bleibt aktiv.
Im zweiten Schritt wird die Schriftfarbe des Stiftes gewählt. Dazu kann zwischen vier verschiedenen Farben in einem Drop-down-menü gewählt werden. Das Bild zur Veranschaulichung der Schriftfarbe in der Mitte der Activity passt sich automatisch beim Wechsel einer Farbe an.

Im dritten Schritt kann der Anwender zwischen drei verschiedenen Spitzformen wählen, die in Drop-down-menü zu wählen sind. Auch hier passt sich eine Abbildung beim Wechsel der Spitzform automatisch an.

Im vierten Schritt kann interaktiv das Griffstück gewählt werden. Je nach Stellung des Schiebers ändert sich die Form bzw. die Riffelung des Griffstückes. Ob die Form oder die Riffelung geändert wird, hängt davon ab, in welcher Fabrik gefertigt wird. Ob die Riffelung oder die Form des Griffstückes gewählt werden kann, hängt von dem Produktionsstandort ab.

Im fünften Schritt kann die Gravur eingegeben werden. Die Gravur wird automatisch auf dem unteren Stiftrohling angezeigt. Die Länge der Gravur ist auf 20 Zeichen beschränkt. Bis auf die Emoticons werden alle Sonderzeichen unterstützt.

Im letzten Schritt der Konfiguration kann ein Endstück gewählt werden. Dazu stehen alle zur Auswahl stehenden Produkte in einem Drop-down-menü zur Auswahl. Auch hier wird die Abbildung bei der Wahl des Endstückes automatisch angepasst.

Nach dem sechsten Schritt der Konfiguration werden alle eingegebenen Daten noch einmal aufgelistet, um den Auftrag zu überprüfen (vgl. Abbildung 29).

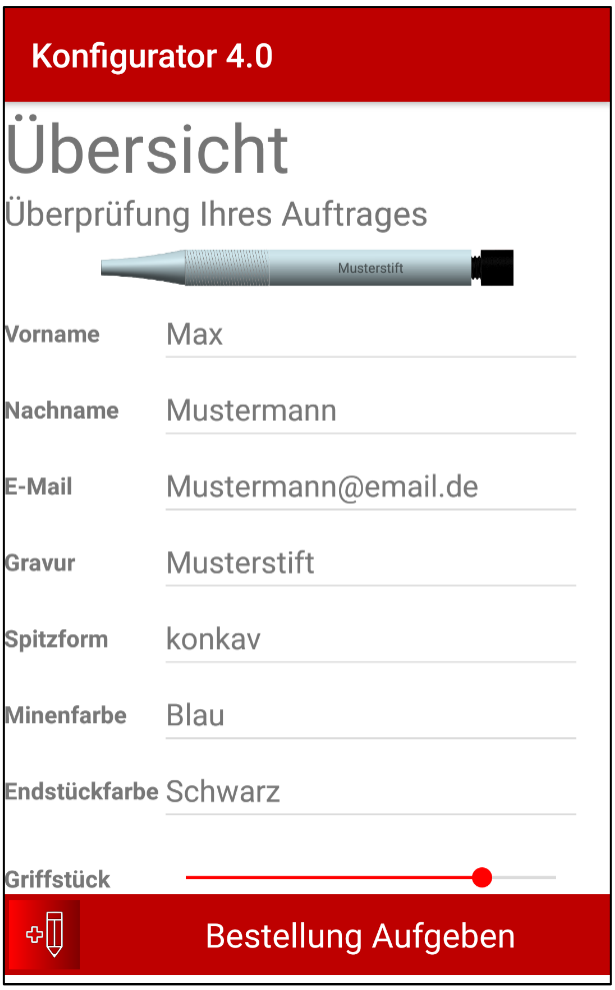


Abbildung 29: Konfigurator / Übersicht

Um dem Kunden einen ersten Gesamteindruck über sein zukünftiges Produkt zu verschaffen, wird aus den zuvor gewählten Teilen eine Gesamtansicht des Produktes erstellt. Sollte ein Fehler während der Konfiguration aufgetreten sein oder sich der Benutzer doch kurzfristig umentschieden haben, kann der jeweilige Eintrag durch einen Klick auf die Komponente geändert werden (vgl. Abbildung 27). Dazu öffnet sich nach dem Klick eine Activity zum Ändern des jeweiligen Parameters (vgl. Anhang 3.1).

Mit dem Klick auf das „Bestellung aufgeben“-Feld startet die Activity *QRCodeViewer* (Siehe Kapitel 4.5.3.3).

4.5.3.3. Auftrag in Form eines QR-Codes

Nachdem der Auftrag vollständig konfiguriert wurde, startet die Activity QR-Code-viewer (siehe Abbildung 30). In dieser Activity werden die Daten aus der Auftragsübersicht in einem String gespeichert und in einem QR-Code auf dem Bildschirm angezeigt (vgl. Abbildung 30, Hintergrund). Um den erstellten Auftrag zu produzieren, muss der erstellte QR-Code an der ersten Station der Auftragsidentifikation eingescannt werden.

Nach dem Betätigen des Beenden-Buttons (vgl. Abbildung 30, unten) öffnet sich ein Dialogfenster, in dem der Anwender gefragt wird, ob der Auftrags-QR-Code gelöscht werden soll (siehe Abbildung 30). Der Anwender hat die Möglichkeit mit der Auswahl „Code speichern“ den QR-Code als Bilddatei auf dem Mobilgerät zu speichern oder mit der Auswahl „Nein“ das Dialogfenster zu schließen und die Activity nicht zu beenden. Die dritte Auswahlmöglichkeit ist die Activity mit „Ja“ zu beenden und die Activity Hauptmenü zu starten.

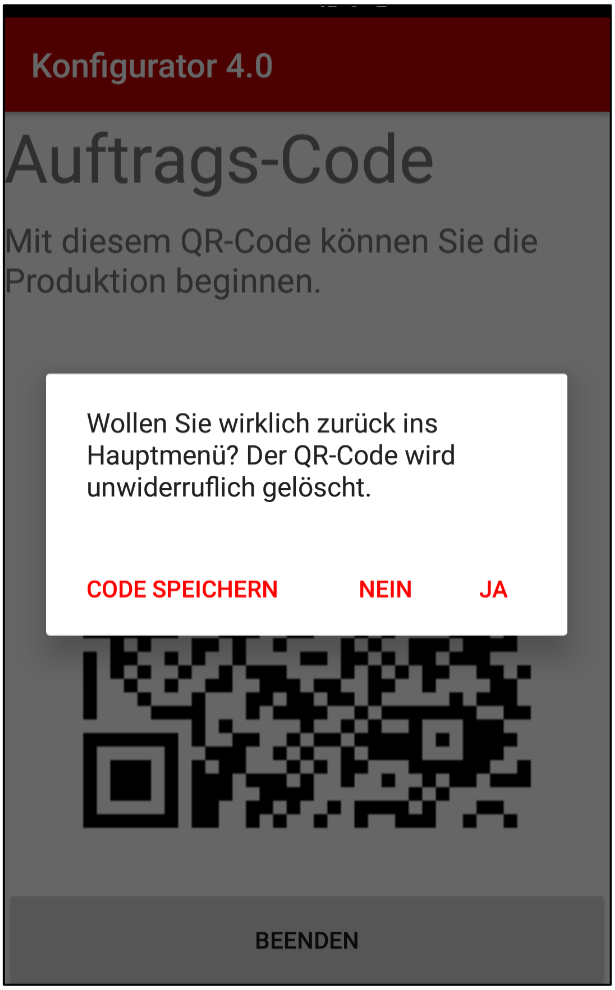


Abbildung 30: Konfigurator / QR-Code speichern

Die Entwicklung des QR-Codes wurde mit der Open-Source-Bibliothek ZXing [OPE17] umgesetzt.

4.5.3.4. Produktnachverfolgung

Die Produktnachverfolgung beginnt durch das Klicken auf das Feld „Auftragssuche“ mit dem Lupen-Symbol im Hauptmenü. In der darauffolgenden Activity wird der Anwender aufgefordert, die Eingabemethode für ein nachzuverfolgendes Produkt zu wählen (siehe Anhang 3.2). Hierbei kann zwischen der manuellen Eingabe und dem Scannen eines QR-Codes gewählt werden. Der QR-Code kann entweder nach Abschluss der Auftragsidentifikation am Bildschirm der Station oder nach Aktualisierung des elektronischen Preisschildes auf dem Werkstückträger eingescannt werden. Der QR-Codescanner ist auf der Grundlage der Open-Source-Bibliothek ZXing entwickelt worden. [OPE17]

Nach der Eingabe einer Auftragsnummer oder durch Scannen eines Nachverfolgungscodes öffnet sich eine Produktübersicht zu der jeweiligen Auftragsnummer (siehe Abbildung 31).

Abbildung 31: Produktsuche / Produktübersicht

In der Produktübersicht sind alle Produktionsdaten zum Produkt tabellarisch aufgelistet. Zudem wird der aktuelle Produktionsstatus (hier „Abgeschlossen“), wie auch die Auftragsnummer und der Zeitstempel, an dem der Auftrag aufgegeben wurde, angezeigt. Das Feld „Auftrag ändern“ ist grau hinterlegt, da die Produktion abgeschlossen

wurde und somit keine Änderungen mehr möglich sind. Wird dennoch auf das Feld gedrückt, erscheint ein Toast, der den Bediener mit folgendem Text informiert: „Sie können den Auftrag nicht mehr ändern, da dieser abgeschlossen wurde“. Durch das Klicken auf das Feld „Auftragsnachverfolgung“ wird die Activity Produktnachverfolgung gestartet. Dies ist möglich, wenn der Auftrag bereits abgeschlossen wurde.

Die erforderlichen Informationen für die Produktnachverfolgung sind in der Fabrikdatei im XML-Format zu finden. In dieser Datei stehen alle Informationen, die zur Visualisierung der einzelnen Stationen notwendig sind. Eine Fabrik beinhaltet mehrere Stationen (vgl. Abbildung 20).

```
<?xml version="1.0" encoding="utf-8"?>
<fabriken>
    <fabrik name="Generalfabrik" dauer="20">
        <station id="2" name="Station 1"  text="Auftragsidentifikation" anzeigeTyp="einfach"/>
        <station id="3" name="Station 2" dauer="100" text="Drehmaschine" anzeigeTyp="progressbar"/>
        <station id="4" name="Station 3" dauer="90" text="Nachbearbeitung" anzeigeTyp="progressbar"/>
        <station id="5" name="Station 4"  dauer="120" text="Kommissionierung" anzeigeTyp="teiler"/>
        <station id="6" name="Station 5" dauer="60" text="gravieren" anzeigeTyp="einfach"/>
        <station id="7" name="Station 6" dauer="120"  text="Montage" anzeigeTyp="einfach"/>
        <station id="8" name="Station 7" dauer="28" text="Qualitätsprüfung" anzeigeTyp="progressbar"/>
</fabrik>
    <fabrik name="Bus" dauer="16">
        <station id="2" name="Station 1"  text="Auftragsidentifikation" anzeigeTyp="einfach"/>
        <station id="5" name="Station 2"  dauer="120" text="Kommissionierung" anzeigeTyp="teiler"/>
        <station id="3" name="Station 3" dauer="100" text="fräsen" anzeigeTyp="progressbar"/>
        <station id="6" name="Station 4" dauer="60" text="gravieren" anzeigeTyp="einfach"/>
        <station id="7" name="Station 5"  dauer="120" text="Montage" anzeigeTyp="einfach"/>
        <station id="8" name="Station 6" dauer="82" text="Qualitätsprüfung" anzeigeTyp="progressbar"/>
    </fabrik>
</fabriken>
```

Abbildung 32: Fabrikdatei

Die Stationen besitzen neben dem Namen und einer ID einen Beschreibungstext, eine Dauer, welche die durchschnittliche Durchlaufzeit der Station angibt. Zudem gibt es einen Anzeigetypen, der für die Art der Stationsvisualisierung verantwortlich ist (vgl. Kapitel 4.5.4.4).

In der Auftragsnachverfolgung werden alle Stationen der Fabrik, in der das Produkt produziert wird, tabellarisch angezeigt (vgl. Abbildung 34). Dazu besitzt die Activity *Produktnachverfolgung* ein Tabellenlayout, welches je nach Eintrag in der Fabrik-Datei mit Stationselementen gefüllt wird (vgl. Kapitel 4.4.3). In der folgenden Abbildung 33 wird die Vorgehensweise zum Aufbau der tabellarischen Produktnachverfolgung gezeigt.

```
//Objekt myFabrik wird durch Auslesen der Fabrik.xml-Datei mit Stationsobjekten gefüllt
myFabrik = Parse_Fabrik.parseStationen(getResources(), myFabrikType);
//folgend werden Status der Stationen durch Auslesen der Events aus der DB aktualisiert
aktualisiere(myFabrik);

//stationList ist eine Liste, die Objekte der Klasse Stationen aus dem Objekt myFabrik beinhaltet
stationList = myFabrik.getStationen();
//in der for-Schleife wird jede Station der stationsList bearbeitet
for (Station station : myFabrik.getStationen()) {
    //folgend wird die aktuelle Station dem Tablelayout der Produktnachverfolgung hinzugefügt
    View currView = addElementsUtil.addStation(ProduktNachverfolgenActivity.this, station);
    stationViewList.add(currView);
    myTLayout.addView(currView);
    //in der folgenden Methode wird die anzeige des Prozesses aktualisiert, z.B. bei einer
    // Prozessbar der Prozess in %
    aktualisiere(currView, station, myAuftrag.getOr_id());
}
```

Abbildung 33: Quellcode, Hinzufügen von Stationen in der Produktnachverfolgung

In der ersten Programmzeile wird die Fabrikdatei ausgelesen (vgl. Abbildung 32). Die Informationen werden in Objekten der Klasse *Station* in einem Objekt der Klasse *Fabrik* gespeichert, die zuvor im Programm erstellt wurden (vgl. Kapitel 4.5.1.2, Abbildung 20). Im nächsten Schritt werden mit der Methode *aktualisiere()* die Stationsdaten der Fabrik aktualisiert. Dazu werden alle Events des nachzuverfolgenden Auftrages abgefragt (vgl. Kapitel 4.5.2) und wie in Abbildung 24 verarbeitet. Die Status der Stationen werden aktualisiert und in das Attribut *status* der Stationsobjekte überschrieben (vgl. Kapitel 4.5.2).

In der for-Schleife werden alle Stationen der Fabrik durch die Methode *addStation()* der Klasse *addElementsUtil* der Produktnachverfolgung hinzugefügt (vgl. Kapitel 4.5.4.4 „Unterstützungsklassen“). Der Methode ist das aktuelle Stationsobjekt zu übergeben, die Informationen über Status und Anzeigetyp, Start- und Endzeit verfügen. Je nach Status und Anzeigetyp wird das Tablelayout um das passende Layout (vgl. Kapitel 4.4.3) erweitert.

Neben dem Stationsstatus wird die Bearbeitungsdauer der Stationen wie auch der gesamten Produktionsdauer ausgewertet. Die Auswertung gibt die abweichende Zeit von der Durchschnittsbearbeitungszeit der Kunden in Minuten an. Eine grüne Zahl mit einem „-“ davor deutet auf eine kürzere Bearbeitungszeit hin. Eine rote Zahl mit nem „+“ davor steht für eine längere Bearbeitungszeit. Bei Abschluss der Produktion wird die gesamte Produktionsdauer ausgewertet und im unteren Bereich der Produktnachverfolgung angezeigt (vgl. Abbildung 34). Die Abbildung 34 zeigt eine abgeschlossene Produktion in der Generalfabrik.

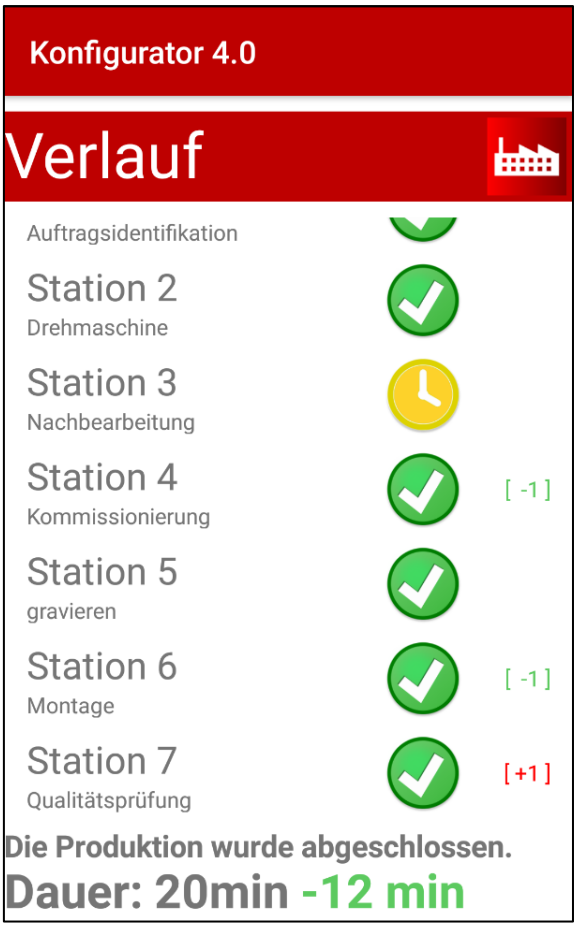


Abbildung 34: Produktnachverfolgung abgeschlossen

In einer nicht abgeschlossenen Produktnachverfolgung ist im unteren Teil der Activity ein voraussichtliches Produktionsende in Minuten angegeben (vgl. Abbildung 35).

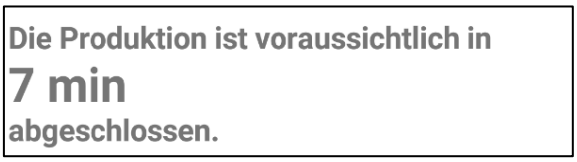


Abbildung 35: Nachverfolgung voraussichtliches Produktionsende

Wenn auf Grund von auftretenden Fehlern in der Fabrik keine genaue Angabe zum Produktionsende gegeben werden kann, wird folgender Text angezeigt: „Die Produktion verzögert sich um wenige Minuten“.

Die Werte für die Berechnung des voraussichtlichen Produktionsendes und der durchschnittlichen Bearbeitungszeiten an den Stationen wurden an Hand einer Kugelschreiberproduktion im Demonstrationsbetrieb bestimmt. Dazu wurden für jede Station die „start“- und „finish“-Events aufgezeichnet und deren Zeitstempel zur Berechnung der Stationszeiten genutzt. Zur Ermittlung der Stationsdauer wurde der Zeitstempel vom „start“-Event vom Zeitstempel des „finish“-Evens subtrahiert. Für die Gesamtdauer der Fabrik wurde der Zeitstempel des „start“-Events der ersten Station vom Zeitstempel des „finish“-Events der letzten Station subtrahiert. Die Durchschnittszeiten sind in der Fabrikdatei eingetragen (vgl. Abbildung 32).

4.5.3.5. Persönlicher Bereich

Durch das klicken auf das Personen-Symbol im Hauptmenü hat der Anwender die Möglichkeit sich in einem sogenannten persönlichen Bereich anzumelden (vgl. Anhang 3.2). Hierzu muss eine gültige Email-Adresse mit dem dazugehörigen Passwort eingegeben werden. Die Daten erhalten Gültigkeit, nachdem mit ihnen ein Produkt in Auftrag gegeben wurde. Ist eine Anmeldung nach Klick auf den Anmelden-Button erfolgreich, öffnet sich der persönliche Bereich. Die persönlichen Daten, sowie der Anmelde-Zustand werden lokal auf dem mobilen Endgerät bis zum Abmelden aus dem persönlichen Bereich gespeichert.

Der persönliche Bereich (siehe Abbildung 36) besteht aus zwei Tabellen. In der oberen Tabelle werden kundenspezifische Daten, wie Vor- und Nachname, Email, Kundennummer und Datum des angemeldeten Kunden angezeigt. In der unteren Tabelle werden alle zu dieser Kundennummer vorhandenen Aufträge aufgelistet.

Abbildung 36: Persönlicher Bereich

Dabei stehen die Icons für den Status des jeweiligen Auftrages (grüner Haken = Abgeschlossen, gelbe Uhr = wartet oder in Arbeit). Neben dem Icon ist der Zeitstempel der jeweiligen Auftragsidentifikation angegeben. Durch klicken auf eine beliebige Spalte in der unteren Tabelle kann eine Produktübersicht zu dem Auftrag geöffnet werden. In diesem hat der Kunde die Möglichkeit, den Auftrag nachzuverfolgen oder nach Belieben zu ändern. Durch einen Klick auf das Feld *Bestellung Aufgeben* kann ein neuer Auftrag konfiguriert werden. Der erste Schritt *Persönliche Daten* der Auftragskonfiguration wird hier übersprungen, da die Kundendaten durch die Anmeldung festgelegt sind.

4.5.3.6. Auftragsänderung

Die Auftragsänderung kann nur gestartet werden, wenn die Auftragsübersicht zuvor aus dem persönlichen Bereich gestartet wurde. Somit wird verhindert, dass Unberechtigte Auftragsänderungen vornehmen können. Des Weiteren kann die Auftragsänderung nur gestartet werden, wenn die Produktion noch nicht abgeschlossen wurde. Das folgende Diagramm (Abbildung 37) zeigt die Struktur der Auftragsänderung.

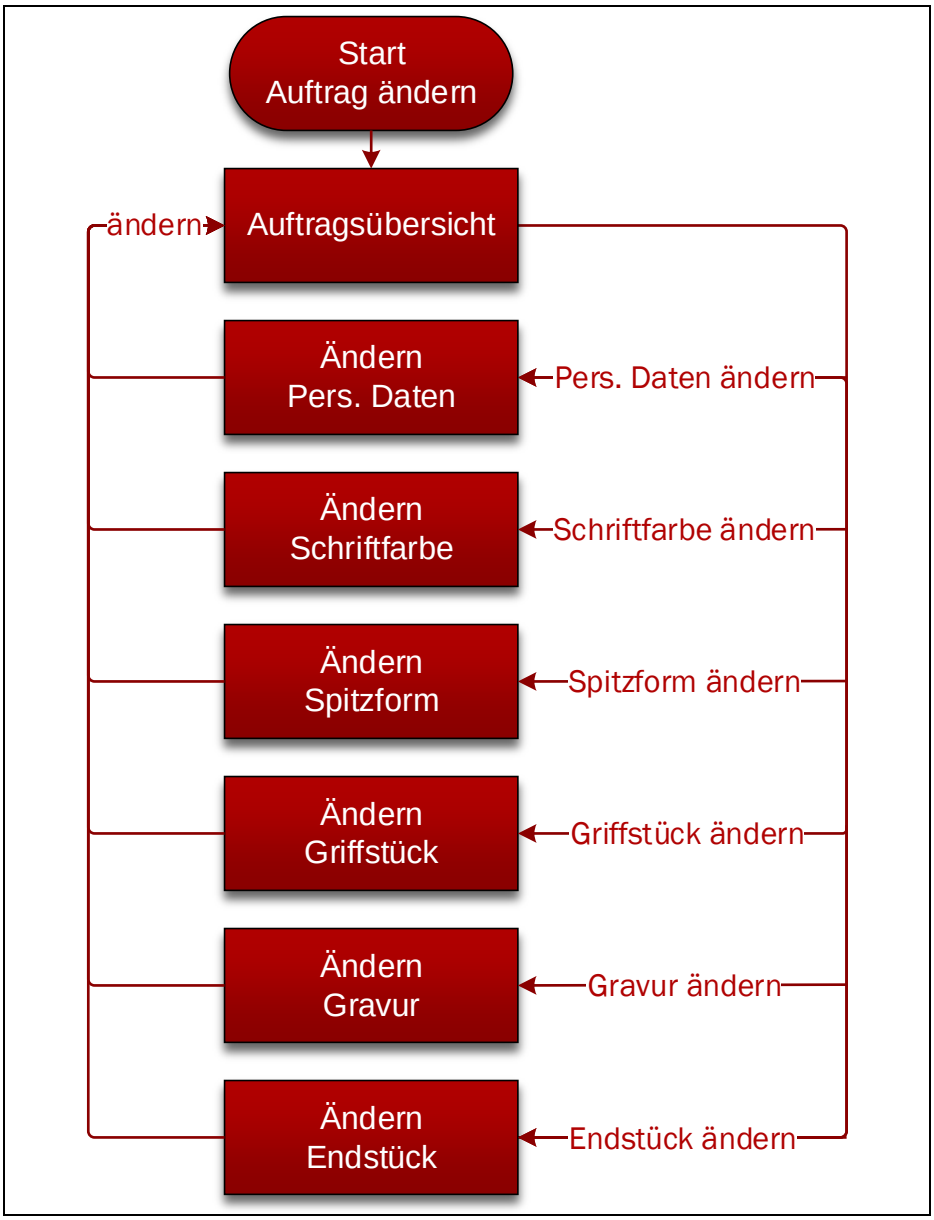


Abbildung 37: Struktur der Auftragsänderung

Aus der Struktur der Auftragsänderung lässt sich schließen, dass aus der Activity der Auftragsübersicht je nach Änderungswunsch (rechts im Bild) die jeweilige *Änderungsactivity* gestartet wird. In der *Änderungsactivity* hat der Anwender die Möglichkeit, den gewünschten Eintrag zu ändern (links im Bild). Nach einer Änderung wird die aktuelle Activity beendet und die *Auftragsübersichtactivity* wird gestartet.

Die Activity Auftragsänderung (siehe Abbildung 38) besteht aus einer Abbildung, welche das zu fertigende Produkt dargestellt wird. Des Weiteren ist im unteren Teil dieser Activity eine Auflistung aller veränderlichen Parameter des jeweiligen Produktes zu sehen. Durch Klicken auf einen gewünschten Parameter, kann dieser geändert werden und hat somit Auswirkungen auf die aktuelle Fertigung des Produktes.

Bevor sich eine weitere Activity öffnet, um eine Änderung vorzunehmen wird überprüft, ob der jeweilige Parameter in der Fertigung schon verwendet wurde. Dies geschieht, indem alle Events, die dem Auftrag zugeordnet sind, durchlaufen werden und nach Events der Station gesucht, die dem Auftragsparameter zugewiesen sind. Sollte die Station zum Ändern des gewünschten Parameters des Produktes schon verwendet worden sein, wird der Anwender mit einem „Toast“ benachrichtigt, dass keine Änderung mehr möglich ist (vgl. Abbildung 38 unten).

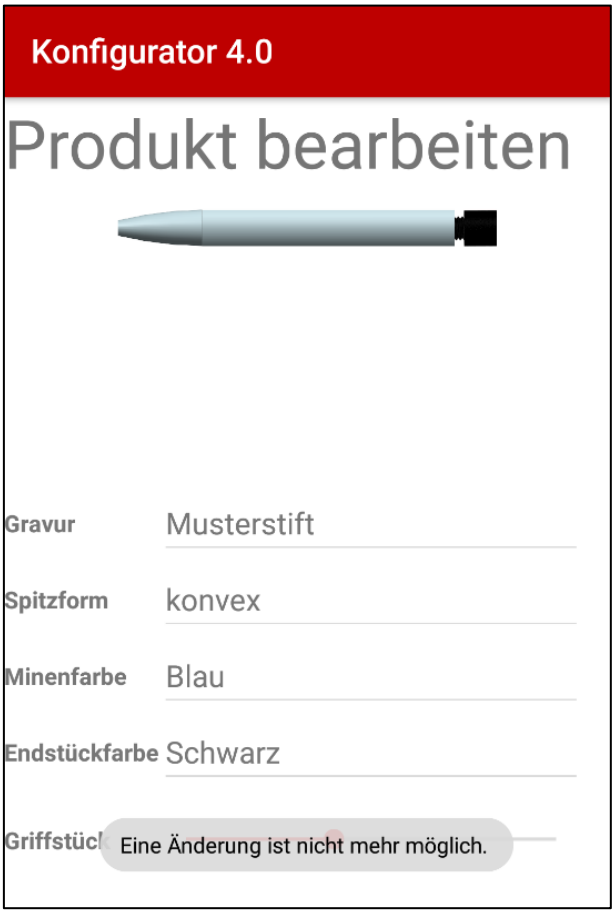


Abbildung 38: Auftragsänderung / Toast „Änderung nicht mehr möglich“

Nachdem eine Änderung nach erfolgreicher Prüfung des Produktionsstandes ausgeführt wurde, wird die jeweilige Änderungs-Activity geschlossen und der Anwender wird durch einen Toast benachrichtigt, dass der Auftrag geändert wurde (vgl. Abbildung 39). Die Parameter, die in der Activity Produkt bearbeiten angezeigt werden, sind immer die aktuellen Daten aus der Datenbank. Daraus lässt sich schließen, ob eine Parameteränderung tatsächlich stattgefunden hat.

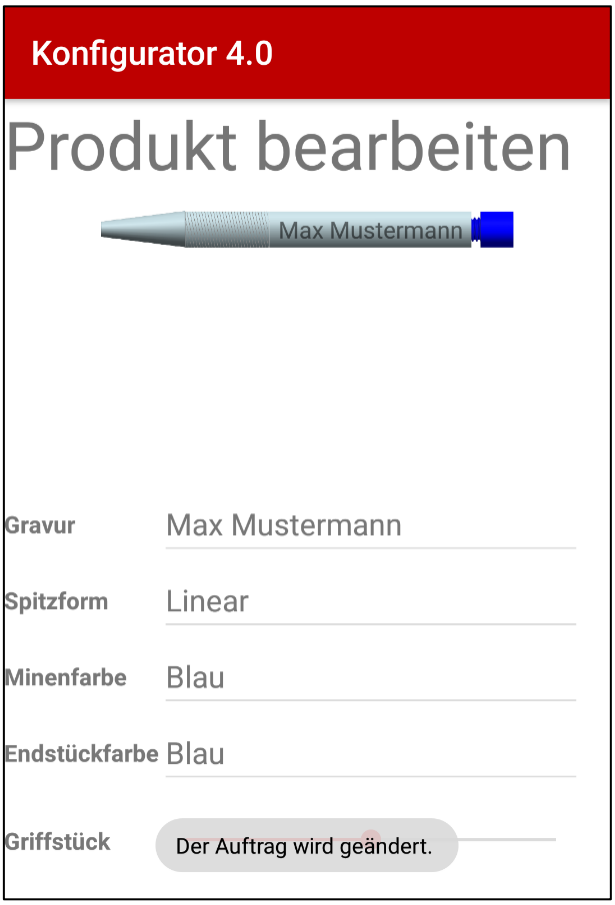


Abbildung 39: Auftragsänderung / Toast „Auftrag wird geändert“

4.5.3.7. Statusbenachrichtigungen

Die App benachrichtigt die Kunden durch sogenannte Push-Benachrichtigungen, wenn ein Auftrag begonnen oder abgeschlossen wurde. Bei aktiver Nutzung des Gerätes tauchen die Benachrichtigungen oben in der Statusleiste des Gerätes auf, bei ausgeschaltetem Bildschirm in der Mitte des Sperrbildschirms. Zieht man die Statusleiste nach unten, kann ebenfalls auf die Benachrichtigungen geklickt werden (vgl. Abbildung 40).

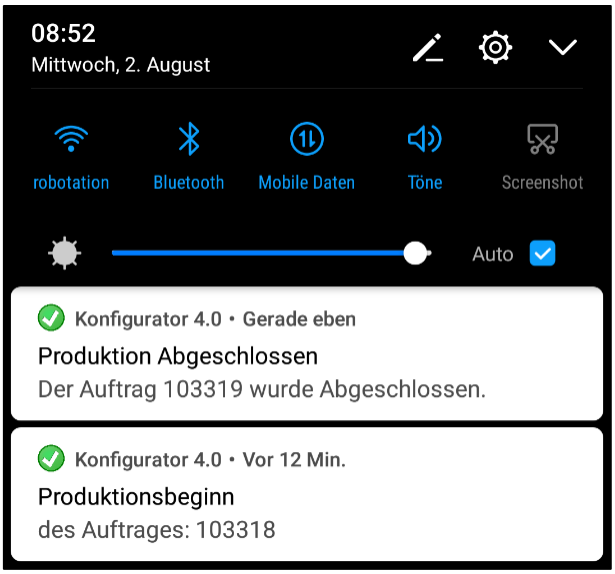


Abbildung 40: Benachrichtigungen Statusleiste

Durch das Klicken auf die jeweilige Benachrichtigung wird die Produktionsnachverfolgung zu dem jeweiligen Auftrag gestartet (siehe auch Kapitel 4.5.3.4 Produktnachverfolgung). Durch Benachrichtigungen kann der Anwender schnell auf wichtige Ereignisse reagieren. Durch die Benachrichtigung „Produktionsbeginn“ hat der Kunde die Möglichkeit, rechtzeitig seinen Auftrag anzupassen, sofern dies gewünscht ist. Benachrichtigungen kann Unternehmern oder Mitarbeitern einer Produktion helfen, schneller auf Ereignisse reagieren zu können. Ein Beispiel dafür ist eine Benachrichtigung für einen Defekt an einer Maschine. Der Anlagenbediener bekommt die Fehlermeldung der Maschine direkt auf das mobile Endgerät und kann ohne sich an der Anlage zu befinden, über den Defekt informieren.

Jeder Auftrag, den ein Anwender in der App nachverfolgt hat, wird im Hintergrund der App abgespeichert. Im Abstand von 30 Sekunden lädt die App aktuelle Produktionsdaten der jeweils hinterlegten Aufträge (vgl. Abbildung 41).

```
@Override
protected void onStop() {
    super.onStop();
    //Endlostimer
    backgroundTimer = new CountDownTimer(REFRESHRATEBACKGROUND, REFRESHRATEBACKGROUND) {
        public void onTick(long millisUntilFinished) {

            Log.i("Timer", "BackgroundTimer");
            Set<String> myAnfragen;
            myAnfragen = saveUtil.ladeSuchanfragen(StartActivity.this);
            if (myAnfragen != null) {
                //ladeStatusdaten für zuletzt gesuchte, um Push-Benachrihtigungen zu erstellen
                String[] myAnfragenArray = myAnfragen.toArray(new String[myAnfragen.size()]);
                for (int i = 0; i <= myAnfragenArray.length - 1; i++) {
                    //startet Hintergrundprozess für Netzwerkzugriff und
                    new LadeStatusDaten().execute(myAnfragenArray[i].toString());
            }   }   }
        //für Endlostimer, starte den Timer neu nach Ablauf der Zeit
        public void onFinish() {backgroundTimer.start();}
    }.start();
}
```

Abbildung 41: Benachrichtigungsstatus Speichern

Sollte sich der Zustand eines Auftrages verändert haben, wird der Anwender wie in den Abbildung 40 gezeigt, bei Produktionsbeginn und bei Produktionsende benachrichtigt. Um zu verhindern, dass die gleiche Benachrichtigung wiederholt auftritt, wird im Hintergrund gespeichert, ob es zu dem Auftrag zuvor bereits eine Benachrichtigung gegeben hat. Dies geschieht in der Methode *saveNotifyStatus()* der Utilklasse *saveUtil* (siehe Abbildung 42). Dort wird die Auftragsnummer mit dem Status der Benachrichtigungen (*pStatus_t.in* für Beginn und *pStatus_t.done* für Produktionsende) abgespeichert.

```
//überprüfe, ob der Produktionsstart zum Auftrag s[0] noch nicht benachrichtigt wurde
if (saveUtil.ladeNotifyStatus(StartActivity.this, s[0]) != ownDataTypes.pStatus_t.in) {
    //Benachrichtigung schreiben, Übergabeparameter sind Titel, Text, Icon und
    // die auszuführende Activity nach Klick auf die Benachrichtigung
    //sowie der Auftragsnummer(ProduktID), welche die Activity zur Anzeige benötigt
 NotificationUtil.setNotification(StartActivity.this, "Produktionsbeginn",
         "des Auftrages"+": "+ s[0], R.drawable.haken,s[0]);
 //im Hintergrund wird gespeichert, dass das Produkt mit der ProduktID s[0]
 //für Auftragsbeginn benachrichtigt wurde
 saveUtil.saveNotifyStatus(StartActivity.this, s[0], ownDataTypes.pStatus_t.in);

}
```

Abbildung 42: Quellcode Benachrichtigung

Eine echtzeitnähere Benachrichtigung wäre denkbar, zugunsten des Anwenders bleibt es jedoch bei der Wiederholrate von 30 Sekunden. Somit werden die Zugriffe über das Internet auf die Produktionsdaten, welche sich negativ auf das meist begrenzte Datenvolumen des Mobilfunkanbieters auswirken, beschränkt.

4.5.4. Weitere Punkte der Implementierung

In diesem Kapitel werden weitere Punkte beschrieben, die während der Implementierung der App bearbeitet wurden.

4.5.4.1. Sprachunterstützung

Die Generalfabrik steht auf dem Messegelände Hannovers. Sie ist somit auch ein Anlaufpunkt für internationale Messebesucher. Daher unterstützt die App zum Zeitpunkt der ersten Veröffentlichung am 03.08.2017 die Sprachen Deutsch und Englisch.

Für die Entwicklung einer mehrsprachigen App bietet Android Studio einen String-Editor, mit dem Strings auf verschiedenen Sprachen erstellt werden können. Jeder Zeichenkette wird eine eindeutige ID zugewiesen (vgl. Abbildung 43, *Key*), um mit dieser im späteren Programmcode die Strings nutzen kann. Je nach eingestellter Systemsprache im Gerät werden die Strings der jeweiligen Sprache geladen. Sollte die Sprache nicht unterstützt sein, werden „default"-Strings geladen (vgl. Abbildung 43, *Default Value*), die neben den Sprachen erstellt werden muss.

Key	Untransla...	Default Value	German (de)	English (en)	Spanish (es)
blau	☐	Blau	Blau	Blue	Azul
btn_beenden	☐	Beenden	Beenden	Exit	Salida
btn_weiter	☐	nächster Schritt	nächster Schritt	next Step	próximo paso
derAuftrag_pushUp	☐	Der Auftrag	Der Auftrag	The Order	El Pedido

Abbildung 43: String-Editor von Android Studio

4.5.4.2. Effektivitätssteigerung von Webseitenaufrufen

Die Variante, dass die Webseite des Kompetenzzentrums in der App geöffnet wird, hat sich als nichteffektive Lösung erwiesen. Die App hat als Webbrowser fungiert. Dadurch wurden Webseitendaten unter dem Speicherpfad der App gespeichert. Nach einem Aufruf der Webseite über die Web-Activity sind 700 KB Webseitendaten im Hintergrund der App gespeichert worden. Durch die Verwendung von Weiterleitungen in den vorinstallierten Browser des mobilen Gerätes fallen zwar noch immer Webseitendaten an, jedoch fallen diese nicht mehr in den Speicherbedarf der App und minimieren den langfristigen Speicherbedarf der App. Zudem ist ein Browser für Webseiten optimiert und kann eine stabile und schnelle Anzeige der Webseite garantieren.

4.5.4.3. Effizienzsteigerung

Die Effizienz ist gegeben, indem wiederholte Produktionsdatenzugriffe, durch Abspeichern von Produktionsdaten im Hintergrund der App, verhindert werden. Zudem erfolgen nur Produktionsdatenzugriffe, wenn der Anwender einen aktuellen Auftrag nachverfolgt. Abgeschlossene Aufträge werden in der App hinterlegt und werden von der Produktionsdatenaktualisierung ausgeschlossen.

Zudem werden Auftragsdaten bei einem Wechsel der Activity durch sogenannte Intents übergeben, sodass eine erneute Auftragsdatenabfrage nicht notwendig ist (vgl. Abbildung 44).

```
Intent i = new Intent(Konfigurator_4_Griffstueck_Activity.this, Konfigurator_5_Gravur_Activity.class);
i.putExtra("FabrikType", getIntent().getStringExtra("FabrikType"));
i.putExtra("Vorname", getIntent().getStringExtra("Vorname"));
i.putExtra("Nachname", getIntent().getStringExtra("Nachname"));
i.putExtra("Email", getIntent().getStringExtra("Email"));
i.putExtra("FabrikType", getIntent().getStringExtra("FabrikType"));
i.putExtra("Endstueck", getIntent().getIntExtra("Endstueck",0));
i.putExtra("Spitzform", getIntent().getIntExtra("Spitzform", 0));
i.putExtra("Minenfarbe", getIntent().getIntExtra("Minenfarbe", 0));
i.putExtra("Griffstueck", (Integer.parseInt(myGriffstueck)+""));
i.putExtra("LoginStatus", getIntent().getBooleanExtra("LoginStatus", false));
startActivity(i);
```

Abbildung 44: Beispiel eines Intent für den fünften Schritt des Konfigurators

Für die regelmäßige Aktualisierung der Auftragsdaten im Hintergrund der App ist der *mainTimer* verantwortlich. Der *mainTimer* ist ein Endlostimer, der bei Ablauf der Wiederholrate von 30 Sekunden die Produktionsdaten des aktuell angezeigten Auftrages und aller bereits hinterlegten Aufträge aktualisiert (für die Benachrichtigungen, vgl. Kapitel 4.5.3.7). Bei der Produktnachverfolgung wurde ein weiterer Timer integriert, der mit einer Wiederholrate von fünf Sekunden Events (vgl. Kapitel 4.5.2) aus der Datenbank liest und den Status des Auftrages aktualisiert. Die Wiederholrate von fünf Sekunden reicht aus, um den Kunden einen echtzeitnahen Produktionsverlauf visualisieren zu können.

4.5.4.4. Unterstützungsklassen

Funktionen, die wiederholt in verschiedenen Klassen auftreten, wurden in Unterstützungsklassen („Utils“) ausgelagert und vereinheitlicht (siehe Abbildung 45).

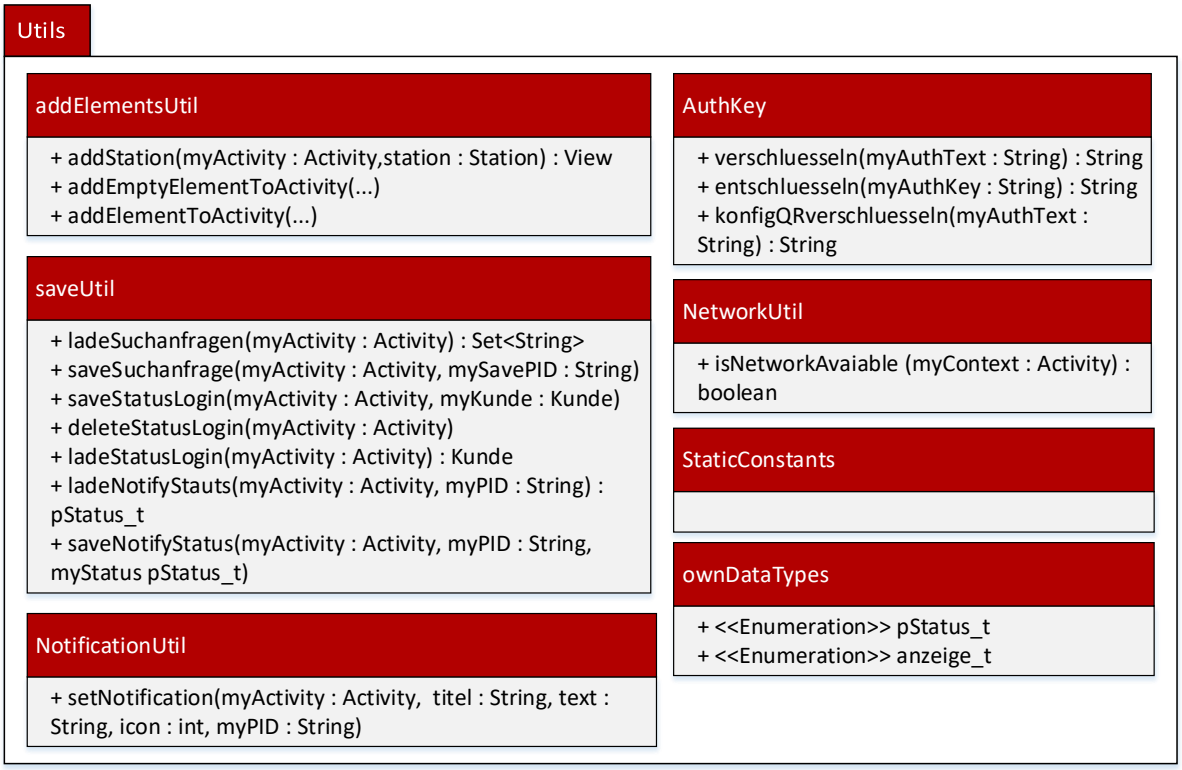


Abbildung 45: UML-Klassendiagramm, Utilklassen

Die Unterstützungsklassen können von jeder anderen Klasse im Projekt aufgerufen werden. Die Klasse *addElementsUtil* ist für die tabellarische Auflistung von Auftrags- und Kundendaten in den Activities, sowie der tabellarischen Auflistung der Stationen in der Produktnachverfolgung zuständig. In der Klasse *SaveUtil* werden Benachrichtigungen gespeichert, um zu verhindern, dass der Kunde mehrmals die gleiche Benachrichtigung erhält. Zudem werden mit der Methode *saveStatusLogin()* Kundendaten für den persönlichen Bereich und der Loginstatus gespeichert. Die Methode *deleteStatusLogin()* wird beim Abmelden aus dem persönlichen Bereich aufgerufen, die das Löschen der Kundendaten und dem aktualisieren des Loginstatus zufolge haben. Für die Ver- und Entschlüsselung von Daten ist die Klasse *AuthKey* verantwortlich. Der netzwerkstatus des mobilen Gerätes kann mit Hilfe der Methode *isNetworkAvaiable()* der Klasse *NetworkUtil* abgefragt werden. In der Klasse *StaticConstants* werden Passwörter und Wiederholraten für Netzwerkzugriffe gespeichert, die an verschiedenen Stellen im Softwareprojekt für Datenbankzugriffe und Verschlüsselungen benötigt werden. Die Klasse *NotificationUtil* beinhaltet die Methode setNotification(), die Benachrichtigungen auf dem mobilen Endgerät erstellt. Zum Methodenaufruf muss ein Titel, ein Text und ein Icon für die Benachrichtigung übergeben werden. Für die Visualisierung wurden zwei strukturierte Datentypen erstellt. Die Datentypen sind in der Klasse *ownDataTypes* zu finden. Der Datentyp *pStatus_t* spiegelt mögliche Zustände wieder. Die

Zustände sind *pstatus_t.in*, *pstatus_t.done, pstatus_t.err* und *pstatus_t.move*. Der Datentyp *anzeige_t* steht für die Anzeigevariante einer Station in der Produktnachverfolgung. Mögliche anzeigetypen sind *anzeige_t.progressbar*, *anzeige_t.teiler* und *anzeige_t.einfach*.

4.5.4.5. Erweiterbarkeit

Durch die objektorientierte Softwareentwicklung kann die Produktnachverfolgung, beispielsweise durch neue Produktionsschritte in den Fabriken einfach erweitert werden. Stationsdaten, die für die Auflistung und Visualisierung der Stationen notwendig sind, werden der Fabrikdatei (vgl. Abbildung 32) entnommen. Eine Erweiterung dieser Datei um eine Station passt automatisch die Stationsauflistung der Produktnachverfolgung an. Für die Visualisierung des Produktionsstatus an der neu eingebundenen Station sind, wie bei den anderen Stationen, ein Schreiben von Events in die Datenbank notwendig. Die Events müssen nach der Eventspezifikation erstellt werden (vgl. Kapitel 4.5.2).

4.6. Test und Validierung

Die Testphase ist ein elementarer Bestandteil der Appentwicklung. Das Ziel hinter dem Testen ist, dass die App zur Veröffentlichung fehlerfrei ihren Betrieb aufnehmen kann. Dazu wurde zum einen ein Betaphasentest und zum anderen ein Zuverlässigkeitstest der App durchgeführt.

4.6.1. Betaphase

Im Rahmen einer Betaphase im Zeitraum vom 17.07. bis zum 03.08.2017 konnten ausgewählte Nutzer die erste funktionsfähige Version 0.1 (Beta) der App vor Markteinführung testen. Die genutzte Hardware der Tester wurde in der Tabelle 9 mit Bezeichnung, Auflösung, Bildschirmdiagonale und Android-Version dokumentiert, um eventuelle Rückschlüsse auf Die Hardwareeigenschaften bei auftretenden Fehlern zu ziehen.

Zu Beginn der Betaphase bestanden die Felder zum Navigieren in der App ausschließlich aus den entwickelten Icons. Durch die Rückmeldungen der Tester stellte sich heraus, dass die Icons nicht aussagekräftig genug sind. Aus diesem Grund wurden Felder zum Navigieren entwickelt, die neben dem Icon einen Text beinhalten (vgl. 4.4.2, siehe auch Anhang 6), die dem Anwender die Nutzung der App erleichtern.

Tabelle 9: Hardwarekompatibilität, Auflistung getesteter Hardware

Bezeichnung des Gerätes	**Auflösung; Bildschirmdiagonale**	**Android-Version**	**API-Version**	**nichtbehobene Probleme[8]**
Huawei P9 lite	1920 x 1080; 5,2“	7.0	24	Keine
Sony Xperia Tablet Z	1920 X 1200; 10,1“	4.3	18	Layout nicht vollständig für Tablet geeignet
Samsung S3 mini	800 x 480; 4“	4.1.2	16	Text nicht zentriert
Samsung Galaxy Note 3 LTE	1920 x 1080; 5,7“	6.0.1	23	keine
Samsung S8	2960 x 1440; 5,8“	7.0	24	Layoutfehler
Samsung Galaxy Tab 2	1.024 x 600; 7“	4.2.2	17	keine
Samsung Galaxy Tab 10.1	1920 x 1200; 10,1“	4.2.2	17	Layout nicht vollständig für Tablet geeignet
Xiaomi Redmi Note 4	1920 x 1080; 5,5“	6.0.1	23	Keine
Huawei P9	1920 x 1080; 5,2“	7.0	24	Keine
HTC One mini	720 x 1280; 4,3“	4.4.2	17	Keine

[8] nach der Beta-Phase

Bezeichnung des Gerätes	Auflösung; Bildschirmdiagonale	Android-Version	API-Version	nichtbehobene Probleme
Samsung Galaxy Note 3	1920 x 1080; 5,7“	5.1.1	22	Layoutfehler
Samsung Galaxy Tab S2	2048 X 1536; 9,7“	7.0	24	Layout nicht vollständig für Tablet geeignet

Beobachtungen

Mit den getesteten Tablets „Sony Xperia Tablet Z“, „Samsung Galaxy Tab S2“ und „Samsung Galaxy Tab 10.1“ konnte die App Vollständig genutzt werden, jedoch ist das Layout hauptsächlich für Smartphones entwickelt. Dadurch werden die Icons unscharf und Texte zu klein dargestellt. Bei den Smartphones „Samsung S8“ und „Samsung Galaxy Note 3“ werden die Namen und Beschreibungen der Stationen in der Produktnachverfolgung (vgl. Abbildung 16) nicht vollständig angezeigt. Zudem werden bei dem „Samsung Galaxy S3 mini“ Texte nicht linksbündig, stattdessen vertikal zentriert angezeigt.

Der Anzeigefehler lässt sich auf die sehr hohe Auflösung des „Samsung S8“ und auf die sehr niedrige Auflösung des „Samsung Galaxy Note 3“ zurückführen (vgl. Tabelle 9). Die Funktion der App ist dadurch nicht weiter eingeschränkt.

Bei den API-Versionen 18 und darunter wurde festgestellt, dass bei der Produktnachverfolgung der rote Rahmen um die aktuelle Bearbeitungsstation (vgl. Kapitel 4.4.3) nicht sichtbar wird. Dies schränkt die Nutzung der App ebenfalls nicht ein.

Für den Anwender nicht sichtbare Fehler werden als Fehlermeldung an die Google Play Console für Appentwickler gesendet, die der Appentwickler zentral von allen Geräten, die die App installiert haben, einsehen und beheben kann.

Ergebnis

Durch die Rückmeldungen der Tester konnten Fehler vor der Markteinführung identifiziert und beseitigt werden. Dazu gehörten Layout-Fehler, die auf verschiedene Seitenverhältnisse und Bildschirmauflösungen zurückzuführen sind. Ebenfalls wurden durch die Tester ungültige Eingabemöglichkeiten aufgedeckt und auf fehlende Funktionen hingewiesen, wie der Beschränkung der Gravurlänge oder der Rücknavigierbarkeit in einigen Activities.

Die App funktioniert grundsätzlich auf allen getesteten Geräten. Auf Tablets traten Skalierungsfehler auf, die auf die Größe des Displays zurückzuführen sind. Obwohl bei dem „Samsung Galaxy S3 mini“ mehrere Anzeigefehler zu verzeichnen sind, jedoch aber keine Einschränkungen in der Nutzung der App auftreten, bleibt die API-Version 16 (vgl. Kapitel 3.2.3, Tabelle 4) unter den unterstützten API-Versionen der App.

Die Testpersonen betrachten die App aus einer anderen Perspektive, als der Entwickler. Dadurch konnten in der Betaphase Fehler schneller identifiziert werden. Neben der Fehleridentifizierung konnte die App auf verschiedenen Endgeräten getestet werden. Grundlegende Funktionen der App (wie Produktnachverfolgung, Konfiguration neuer Aufträge und Auftragsänderung) funktionieren auf den getesteten Geräten einwandfrei.

4.6.2. Zuverlässigkeitstest

Zur Ermittlung der Zuverlässigkeit der App, wurden neben dem Betatest weitere Tests durchgeführt. Getestet wird mit dem Smartphone „Huawei P9 lite“. In jeder Fabrik wurden zehn Kugelschreiber hergestellt, die zuvor als Auftrag mit dem Konfigurator der App generiert wurden. Die Kommunikation zwischen der App und der Datenbank funktioniert wie in Kapitel 4.5.1 „Datenkommunikation“ beschrieben für beide Fabriken identisch, da sie die gleiche Datenbank nutzen. Bei den Tests wird jeder Produktionsprozess mit der App nachverfolgt. Zudem werden stichprobenartig Änderungen an den Auftragsdaten durch die App während der Fertigung vorgenommen.

Beobachtungen

Der erste Test fand in der mobilen Fabrik statt. Bis auf die Station der Kommissionierung, konnten alle Stationen erfolgreich und auf fünf Sekunden genau nachverfolgt werden. Die fünf Sekunden sind der Aktualisierungsrate zu verschulden. Bei der Produktnachverfolgung werden die Events im Fünfsekundentakt aus der Datenbank gelesen (vgl. Kapitel 4.5.4.3). Die Station der Kommissionierung befindet sich gegenwärtig in einem Wartungszustand und kann aus diesem Grund nur mit dem „start“- und „finish“-Event getestet werden (vgl. Kapitel 4.5.2).

Der zweite Test fand in der Generalfabrik statt. Dort konnte ebenfalls die Station der Kommissionierung auf Grund von Wartungsarbeiten nicht vollständig getestet werden. Die Station Nachbearbeitung befindet sich noch in der Aufbauphase und wurde daher aus dem Test ausgeschlossen.

Die zeitliche Auswertung an jeder Station erfolgte Problemlos. Der Status Produktionsbeginn und Produktion Abgeschlossen wurden erfolgreich als Benachrichtigungen auf dem Smartphone angezeigt.

Ergebnis

Es wurden keine gravierenden Fehler in der App selbst festgestellt, die die Konfiguration, die Produktionsnachverfolgung oder die Auftragsänderung betreffen. Durch aktuelle Wartungs- und Umbauarbeiten konnten einzelne Stationen nur eingeschränkt getestet werden. Aus der Erkenntnis, dass die Dauer der Gravur nicht vorherbestimmt werden kann, wurde die Anzeigevariante der Gravurstation auf die Variante mit der Anzeige eines Icons geändert (vgl. Kapitel 4.4.3, Tabelle 7).

4.7. Veröffentlichung

Der Name der App muss ansprechend sein, gleichzeitig aber auch im Bezug zur Digitalisierung bzw. Industrie 4.0 und dem eigentlichen Zweck der App stehen. Zudem soll der Name auch dem Kompetenzzentrum zugeordnet werden können. Die Wahl ist auf den Namen „Konfigurator 4.0“ gefallen. Dabei steht der Begriff „Konfigurator“ für die Funktion der Auftragserstellung, der Konfiguration eines Stiftes. Die Funktion der Auftragserstellung ist unabdingbar und daher die wichtigste Funktion der App. Die Endung „4.0“ soll den Zusammenhang zu Industrie 4.0 verdeutlichen. Unter diesem Namen steht die App seit dem 03.08.2017 im Google Play Store [GOO17f] bereit.

Die App „Konfigurator 4.0“ ist seit dem 03.08.2017 kostenlos im Google Play Store [GOO17f] erhältlich.

5. Zusammenfassung

Die Digitalisierung bringt viele Vorteile mit sich. Das Ziel des Kompetenzzentrums ist es, seinen Kunden die Vorteile exemplarisch zu demonstrieren und einen Anstoß für eigene Projekte zu geben. Durch die Transparenz einer Produktion können Einblicke ins Produktionssystem geschaffen werden. Prozesse werden durch Visualisierung und Animation verständlicher. Der Unternehmer kann auf Ereignisse reagieren. Seien es wiederauftretende Verzögerungen, die in einzelnen Bearbeitungsstationen auftreten, können diese durch intelligente Auswertung der Produktionsdaten zum Alarmieren des Unternehmers führen, der daraufhin die Prozesse effektiver gestalten kann.

Der gegenwärtige Stand der Technik sind Apps zur Sendungsnachverfolgung bei Logistikunternehmen, Echtzeitfahrpläne von öffentlichen Verkehrsbetrieben, Konfigurations- und Benachrichtigungsapps von Maschinenherstellern oder Serviceapps, die die Unternehmenssteuerung erleichtert.

Um Kunden des Kompetenzzentrums die Vorteile der Digitalisierung zeigen zu können, wurde im Rahmen dieser Arbeit eine App entwickelt, die die vielfältigen Möglichkeiten im Produktionssystem aufzeigen. Dazu gehört hauptsächlich die Produktnachverfolgung. Mit der Produktnachverfolgung wird das Produktionssystem für die Kunden greifbarer. Daten, die im Hintergrund der Fertigung entstehen werden dafür mit einfachen Abbildungen und Animationen visualisiert. Zudem haben der Kunden des Kompetenzzentrums die Möglichkeit, neue Aufträge zu generieren, Aufträge in der Produktion nachzuverfolgen und Aufträge nach Produktionsbeginn zu ändern. Zudem hat der Anwender die Möglichkeit, Fehler in der Produktion zu erkennen, Benachrichtigungen bei besonderen Aktivitäten in der Produktion zu erhalten und Abweichungen der durchschnittlichen Bearbeitungszeit an jeder Station einzusehen.

Der hauptsächliche Einsatzort der App wird auf dem Messegelände Hannovers sein. Aus diesem Grund unterstützt die App zurzeit die sprachen deutsch und englisch und kann jederzeit um weitere Sprachen erweitert werden. Die App wurde effizient entwickelt, um Ressourcen, wie den verwendeten Arbeitsspeicher, die Prozessorleitung und mobiles Datenvolumen zu minimieren und den Akku des Gerätes zu schonen. Sofern es nötig ist, werden dazu Produktionsdaten aus dem Produktionssystem geladen und werden im Hintergrund der App gespeichert.

Zur Darstellung und Nachverfolgung der Aufträge benötigt die App Produktionsdaten. Diese Daten werden mit Hilfe eines Webservers, der als Schnittstelle zwischen der App und dem Produktionssystem agiert, ausgelesen, im XML-Dateiformat gespeichert und mit der Methode AES[9] verschlüsselt, bereitgestellt.

Durch eine ausführliche Testphase mit zwölf Testern konnte die App vor der Markteinführung verbessert werden. Der größte Kritikpunkt der Tester waren die Icons, die für die Navigierbarkeit der App verantwortlich waren. Die Icons waren nicht selbsterklärend genug. Durch die Rückmeldung der Tester konnten die Icons gegen beschriftete, verständlichere Felder ersetzt und die Navigierbarkeit der App erleichtert werden. Die App ist zurzeit ausschließlich für das Betriebssystem Android nutzbar und unterstützt alle Android-Versionen ab 4.1. Die App wurde am 3 August 2017 im Google Play Store unter dem Namen „Konfigurator 4.0“ veröffentlicht.

[9] Advanced Encryption Standard

6. Ausblick

Was die Digitalisierung für zusätzliche Dienstleistungen dem Kunden gegenüber mit sich bringen kann, zeigt die entwickelte App deutlich. Hinsichtlich der Vorteile für Unternehmer können mit Hilfe der App Fehler in der Produktion erkannt, durchschnittliche Durchlaufzeiten der Produktion eingesehen und die Auslastungen der Maschinen identifiziert werden. Zudem ist ein passives nachverfolgen von Aufträgen in der Fertigung, als auch ein aktives Ändern der Aufträge während der Fertigung möglich.

Durch unterschiedliche Programmiersprachen der Appentwicklung und des Produktionssystems, konnte noch keine Methode zum Ver- und Entschlüsseln der QR-Codedaten nach Standard des Bundesamtes für Sicherheit in der Informationstechnik [BUN13a] gefunden werden. Für große und hochauflösende Bildschirme von mobilen Geräten ist eine Auflösungsanpassung notwendig. Wie in der Betatestphase beschrieben (vgl. Kapitel 4.6.1), sind bei hohen Auflösungen und großen Bildschirmen Anzeigefehler zu verzeichnen. Die App unterstützt zum Zeitpunkt der Veröffentlichung (vgl. Kapitel 4.7) die Sprachen deutsch und englisch (vgl. Kapitel 4.5.4.1). Eine weitreichendere Sprachunterstützung der App kann internationalen Messegästen erleichtern, die Produktionsprozesse zu verstehen. An einer spanischen Übersetzung wird derzeit gearbeitet.

Eine Nutzung von Sonderzeichen bezüglich der Gravuränderung ist nicht möglich. Dies lässt sich auf die Kommunikation mit dem Webserver zurückführen, da http keine Sonderzeichen unterstützt. Eine Möglichkeit ist, die Gravur in einem geeigneten Verfahren zu kodieren (z.B. Base64). Um ein größeres Spektrum an Endgeräten zu unterstützen, kann neben der Appentwicklung für Android eine App für die Betriebssysteme iOS und Windows Phone entwickelt werden. Eine Einbindung der Produktnachverfolgung in der Webseite des Kompetenzzentrums ist denkbar, um Kunden ohne Mobilgerät den Service der Produktnachverfolgung bieten zu können.

Neben den oben genannten Punkten gibt es weitere Umsetzungsideen, welche die jetzigen Funktionen der App erweitern würden. Um für mehr Transparenz in der Auftragsänderung zu sorgen, wäre eine Erweiterung der Datenbank denkbar. Jede Auftragsänderung würde in die Datenbank eingetragen werden und mit einem Zeitstempel der Änderung versehen. Das System zur Ortung von Werkstückträgern (vgl. Kapitel 2.3.3) könnte ebenso in die App eingepflegt werden. Die Werkstückträger besitzen ein Lokalisierungssystem (vgl. Abbildung 6), welches jederzeit den aktuellen Standort an das Produktionssystem senden kann. Dies könnte wiederum in der App visuell dargestellt werden.

Um zum Schluss eine Entscheidungshilfe zu geben, sind folgend Punkte zusammengestellt, die die Entscheidung zur Marktreife erleichtern sollen.

Eine App ist marktreif, wenn:

1. Das Konzept erfolgreich umgesetzt wurde.

Zu Beginn der Markteinführung sollten alle grundlegenden Funktionen der App, die in dem zuvor erarbeiteten Konzept festgehalten wurden, umgesetzt worden sein (in diesem Fall die Sendungsnachverfolgung und Auftragsänderung). Zusätzliche Funktionen (weitere Umsetzungsideen), wie die automatische Benachrichtigung, wenn ein Auftrag fertig produziert wurde, ist für die erste Version nicht von belangen.

2. Das Design ansprechend gestaltet wurde.

Die App sollte zum Zeitpunkt der Markteinführung ein ansprechendes Design haben. Weitere Verbesserungen und Erweiterungen sind im Nachgang möglich.

3. Die Funktionalität ausreichend getestet wurde.

Dass eine App vor der Markteinführung getestet werden muss, ist unumgänglich. Ohne Tests auf verschiedensten Endgeräten (z.B. durch eine Beta-Testphase), kann nicht garantiert werden, dass die App beim Benutzer reibungslos funktioniert. Durch das Testen verschiedener Hardware kann nicht garantiert werden, dass die App auf jedem anderen Gerät funktioniert. Je mehr verschiedene Geräte getestet wurden, desto höher ist die Wahrscheinlichkeit, dass die App auch auf anderen Geräten funktioniert.

Eine App zur Produktionsnachverfolgung ist grundlegend sinnvoll ist, um sowohl dem Kunden, als auch dem Unternehmer entgegen zu kommen. Anhand dieser mobilen Anwendung werden sowohl Transparenz, zum allgemeinen Verständnis der Produktionsabläufe, als auch die Digitalisierung per se verdeutlicht und dem Nutzer nähergebracht. Industrie 4.0 steht für interaktives Gestalten der Produktion unter Einbezug sämtlicher Faktoren, die Kunden und Unternehmer betreffen und zeigt durch aktives Mitbestimmen des Nutzers den Fortschritt der Industrie.

7. Literatur

[ADA10]
Adam, Jessika; *Micromovie: Ein kreatives Medium für mobile Endgeräte,* Hamburg. Diplomica-Verl., 2010.

[ADV11]
Advanced Realtime Tracking GmbH, Weilheim i. OB.; Zürl, Konrad; *Angewandte Virtuelle Technologien im Produkt- und Produktionsmittellebenszyklus: Verbundprojekt AVILUS ; Schlussbericht ; Berichtszeitraum: 01.03.2008 - 30.04.2011,* 2011.

[AIC16]
Aichele, Christian; Schönberger, Marius; *App-Entwicklung - effizient und erfolgreich: Eine kompakte Darstellung von Konzepten, Methoden und Werkzeugen,* Wiesbaden. Springer Vieweg, 2016.

[AMA17]
Amazon Mobile LLC; *Amazon Shopping,* 2017, https://play.google.com/store/apps/details?hl=de&id=com.amazon.mShop.android.shopping.
Abgerufen am 26.07.2017.

[APP17]
Apple Inc.; *Apple Developer Support,* 2017, https://developer.apple.com/support/app-store/.
Abgerufen am 28.07.2017.

[BAU14]
Bauernhansl, Thomas; Hompel, Michael ten; Vogel-Heuser, Birgit (Hrsg.); *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung, Technologien, Migration,* Wiesbaden. Springer Vieweg, 2014.

[BAU08]
Baumgarten, Helmut (Hrsg.); *Das Beste der Logistik: Innovationen, Strategien, Umsetzungen,* Berlin. Springer, 2008.

[BEG17]
BEGO Bremer Goldschlägerei Wilh. Herbst GmbH & Co. KG; *BEGO Aktuell: Den Auftragsstatus bequem über App abrufen – die „BEGO CAD/CAM Tracking App“,* 2017, http://www.bego.com/de/aktuelles/bego-news/den-auftragsstatus-bequem-ueber-app-abrufen-die-bego-cadcam-tracking-app/.
Abgerufen am 26.07.2017.

[BER17]
Berger, Oliver; *DHL,* 2017, www.paket.net/dhl.
Abgerufen am 12.07.2017.

[BUN13a]
Bundesamt für Sicherheit in der Informationstechnik; *IT-Grundschutz: M 2.126 Erstellung eines Datenbanksicherheitskonzeptes,* 2013, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02126.html.
Abgerufen am 25.07.2017.

[BUN13b]
Bundesamt für Sicherheit in der Informationstechnik; *IT-Grundschutz: M 4.34 Einsatz von Verschlüsselung, Checksummen oder Digitalen Signaturen,* 2013, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04034.html.
Abgerufen am 25.07.2017.

[CLA12]
Claussen, Peter; *Die Fabrik als soziales System: Wandlungsfähigkeit durch systemische Fabrikplanung und Organisationsentwicklung - ein Beispiel aus der Automobilindustrie,* Wiesbaden. Springer, 2012.

[CRE17]
Creative Commons; *CC0 1.0 Universell (CC0 1.0): Public Domain Dedication,* 2017, https://creativecommons.org/publicdomain/zero/1.0/deed.de.
Abgerufen am 14.08.2017.

[DB 17]
DB Vertrieb GmbH; *Die App DB Navigator,* 2017, https://www.bahn.de/p/view/service/buchung/mobil/db-navigator.shtml.
Abgerufen am 12.07.2017.

[DEN17]
Denkena, Berend; *Mittelstand 4.0 Kompetenzzentrum Hannover: "Mit uns Digital!" Das Zentrum für Niedersachsen und Bremen,* Hannover, 2017, www.mitunsdigital.de.
Abgerufen am 14.07.2017.

[DHL17]
DHL; *DHL Paket App,* DHL, 2017, https://www.dhl.de/de/privatkunden/kampagnenseiten/dhl-app.html.
Abgerufen am 22.05.2017.

[FIN15]
Finkenzeller, Klaus; *RFID-Handbuch: Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC,* 7. Auflage, München. Hanser, 2015.

[GOO17a]
Google Inc.; *Activity,* Google Inc., 2017, https://developer.android.com/reference/android/app/Activity.html.
Abgerufen am 13.07.2017.

[GOO17b]
Google Inc.; *Android Studio: The Official IDE for Android,* Google Inc., 2017, https://developer.android.com/studio/index.html.
Abgerufen am 15.07.2017.

[GOO17c]
Google Inc.; *Android TV,* Google Inc., 2017, https://www.android.com/intl/de_de/tv/.
Abgerufen am 25.07.2017.

[GOO17d]
Google Inc.; *android version marked share,* Google Inc., 2017, https://developer.android.com/about/dashboards/index.html.
Abgerufen am 11.08.2017.

[GOO17e]
Google Inc.; *Android Wear,* Google Inc., 2017, https://www.android.com/intl/de_de/wear/.
Abgerufen am 25.07.2017.

[GOO17f]
Google Inc.; *Google Play Store: Apps,* Google Inc., 2017, https://play.google.com/store/apps.
Abgerufen am 11.08.2017.

[GOO17g]
Google Inc.; *The Activity Lifecycle,* Google Inc., 2017, https://developer.android.com/guide/components/activities/activity-lifecycle.html.
Abgerufen am 13.07.2017.

[GRA15]
Gralak, Michal; Stark, Thorsten; *Schnelleinstieg App Usability: Plattformübergreifendes Design: Android, Apple iOS und Windows Phone,* 1. Auflage, s.l. Franzis, 2015.

[HAN05]
Hansen, Hans Robert; Neumann, Gustaf; *Informationstechnik,* 9. Auflage, Stuttgart. Lucius & Lucius, 2005.

[HEI13]
Heilinger, Ariane; Stumpf, Veronika; *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0: Abschlussbericht des Arbeitskreises Industrie 4.0,* 2013, https://www.bmbf.de/files/Umsetzungsempfehlungen_Industrie4_0.pdf.

[HER17]
Hermes Germany GmbH; *Hermes Paket Versand & Empfang,* 2017, https://play.google.com/store/apps/details?id=de.myhermes.app&hl=de.
Abgerufen am 18.08.2017.

[KEM06]
Kemper, Alfons; Eickler, André; *Datenbanksysteme: Eine Einführung,* 6. Auflage, München. Oldenbourg, 2006.

[KNA16]
Knaut, Matthias (Hrsg.); *Digitalisierung: Menschen zählen: Beiträge und Positionen 2016,* 2016. Auflage, Berlin. BWV - Berliner Wissenschafts-Verlag GmbH, 2016.

[KOE17]
Koether, Reinhard; Meier, Klaus-Jürgen (Hrsg.); *Lean Production für die variantenreiche Einzelfertigung: Flexibilität wird zum neuen Standard,* Wiesbaden, s.l. Springer Fachmedien Wiesbaden, 2017.

[KOL11]
Koller, Dirk; *iPhone-Apps entwickeln: Applikationen für iPhone, iPad und iPod touch programmieren ; [so entwickeln Sie Apps mit dem iPhone-SDK und Objective-C ; Xcode, Datenbank-Framework und die wichtigsten Klassen des Betriebsystems iOS im Detail ; entwickeln ist nicht genug: Apps im App Store richtig vermarkten ; jetzt aktuell zu iOS 4.x],* 2. Auflage, Poing. Franzis, 2011.

[KOR13]
Koren, Yoram; *The global manufacturing revolution: Product-process-business integration and reconfigurable systems,* Hoboken, N.J. Wiley, 2013.

[KÜN17]
Künneth, Thomas; *Android 7: Das Praxisbuch für Entwickler. Inkl. Einstieg in Android Studio. 70 Projekte zu allen Android-Funktionen: Multimedia, Kamera, Organizer, Sensoren, Datenbanken, Android Wear u. v. m,* s.l. Rheinwerk Verlag, 2017.

[KUR05]
Kurbel, Karl; Endres, Albert; *Produktionsplanung und -steuerung: Methodische Grundlagen von PPS-Systemen und Erweiterungen,* 6. Auflage, München. Oldenbourg, 2005.

[LAC17]
Lackes, Richard; Siepermann, Markus; *Gabler Wirtschaftslexikon: Stichwort: Bewegungsdaten,* 2017, http://wirtschaftslexikon.gabler.de/Archiv/75175/bewegungsdaten-v8.html.
Abgerufen am 30.07.2017.

[LAS17]
Laserscanning Europe GmbH; *Laserscanning, so wird heute vermessen !,* 2017, http://www.laserscanning-europe.com.
Abgerufen am 28.07.2017.

[LEN17]
Lenze Automation; *Lenze Smart Motor: App,* 2017, https://play.google.com/store/apps/details?id=com.lenze.smartmotorapp&hl=de.
Abgerufen am 17.08.2017.

[LOU16]
Louis, Dirk; Müller, Peter; *Android: Der schnelle und einfache Einstieg in die Programmierung und Entwicklungsumgebung,* 2. Auflage, München. Hanser, 2016.

[MAN16]
Manzei, Christian; Schleupner, Linus; Heinze, Ronald (Hrsg.); *Industrie 4.0 im internationalen Kontext: Kernkonzepte, Ergebnisse, Trends,* Berlin, Offenbach, Berlin, Wien, Zürich. VDE Verlag GmbH; Beuth, 2016.

[MOS09]
Mosemann, Heiko; Kose, Matthias; *Android: Anwendungen für das Handy-Betriebssystem erfolgreich programmieren,* 1. Auflage, s.l. Carl Hanser Fachbuchverlag, 2009.

[MÜL88]
Müller, Helmut M.; *Schlaglichter der deutschen Geschichte,* Mannheim. Bibliograph. Inst, 1988.

[OPE17]
Open Source; *Official ZXing ("Zebra Crossing") project home: zxing,* github, 2017, https://github.com/zxing/zxing.
Abgerufen am 05.08.2017.

[PAR17]
ParcelTrack Technologies GmbH; *ParcelTrack - Sendungsverfolgung für DHL, UPS & co,* 2017, https://play.google.com/store/apps/details?id=com.uberblic.parceltrack&hl=de.
Abgerufen am 18.08.2017.

[PIX17]
Pixabay; *Pixabay: Kostenlose Bilder und Videos für Deine kreativen Projekte,* 2017, https://pixabay.com/.
Abgerufen am 14.08.2017.

[SAP17]
SAP SE; *SAP Business One: App,* 2017, https://play.google.com/store/apps/details?id=b1.mobile.android&hl=de.
Abgerufen am 17.08.2017.

[SIM16]
Simon, Hermann; Faßnacht, Martin; *Preismanagement: Strategie - Analyse - Entscheidung - Umsetzung,* 4. Auflage, Wiesbaden. Springer Gabler, 2016.

[SMA15]
Smart-Data-Begleitforschung c/o Loesch Hund Liepold Kommunikation GmbH; *Smart Data Newsletter,* 2015, http://www.digitale-technologien.de/DT/Redaktion/DE/Downloads/Publikation/SmartData_NL1.pdf?__blob=publicationFile&v=5.
Abgerufen am 12.07.2017.

[SPR17]
Springer Gabler Verlag; *Gabler Wirtschaftslexikon: Stichwort: Losgröße,* 2017, http://wirtschaftslexikon.gabler.de/Archiv/57174/losgroesse-v6.html.
Abgerufen am 19.07.2017.

[STA17a]
StatCounter Global Stats; *StatCounter Global Stats,* 2017, http://gs.statcounter.com/os-market-share/.
Abgerufen am 11.08.2017.

[STA17b]
Statista GmbH; *Anzahl der verfügbaren Apps im Google Play Store in ausgewählten Monaten von Dezember 2011 bis Juli 2017,* 2017, https://de.statista.com/statistik/daten/studie/74368/umfrage/anzahl-der-verfuegbaren-apps-im-google-play-store/.
Abgerufen am 30.07.2017.

[STE17a]
Steyer, Manfred; *Progressive Web-Apps: Offlinefähige Web-Anwendungen mit nativen Qualitäten,* Frankfurt am Main. entwickler.press, 2017.

[STE17b]
Steyer, Ralph; *Cordova: Entwicklung plattformneutraler Apps,* Wiesbaden. Springer Vieweg, 2017.

[TOR17]
Torsten Bartel, Gesine Quint; *Definition von Usability und UX.: Usability & User Experience,* Hannover, usability.de GmbH & Co. KG, 2017, https://www.usability.de/usability-user-experience.html.
Abgerufen am 15.07.2017.

[TTG08]
Ttgen, Markus R.; *Tracking und tracing,* [Place of publication not identified]. Grin Verlag, 2008.

[UPS17]
UPS; *UPS Mobile,* 2017, https://play.google.com/store/apps/details?id=com.ups.mobile.android&hl=de.
Abgerufen am 18.08.2017.

[VER12]
Verclas, Stephan; Linnhoff-Popien, Claudia; *Smart Mobile Apps: Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse,* Berlin, Heidelberg. Springer-Verlag Berlin Heidelberg, 2012.

[VOG17a]
Vogel-Heuser, Birgit; Bauernhansl, Thomas; Hompel, Michael ten (Hrsg.); *Handbuch Industrie 4.0: Bd. 1: Produktion,* 2. Auflage, Berlin. Springer Vieweg, 2017.

[VOG17b]
Vogel-Heuser, Birgit; Bauernhansl, Thomas; Hompel, Michael ten (Hrsg.); *Handbuch Industrie 4.0 Bd.2: Automatisierung,* 2. Auflage, Berlin, Heidelberg, s.l. Springer Berlin Heidelberg, 2017.

[VOG17c]
Vogel-Heuser, Birgit; Bauernhansl, Thomas; Hompel, Michael ten (Hrsg.); *Handbuch Industrie 4.0 Bd.4: Allgemeine Grundlagen,* 2. Auflage, Berlin, Heidelberg, s.l. Springer Berlin Heidelberg, 2017.

[VOL17]
Vollmer, Guy; *Mobile App Engineering: Eine systematische Einführung - von den Requirements zum Go Live,* 1. Auflage, Heidelberg. dpunkt.verlag, 2017.

[YEE17]
yeebase media GmbH; *Progressive Web Apps: So könnt ihr sie schon jetzt ausprobieren,* Hannover, 2017, http://t3n.de/news/progressive-web-apps-koennt-790923/. Abgerufen am 17.08.2017.

8. Anhang

8.1. Glossar

Activity

Eine Activity repräsentiert die einzelnen Bildschirmseiten einer App. Eine App besitzt mindestens eine Activity. [LOU16]; [MOS09]

Android

Android ist nicht nur ein Betriebssystem für mobile Endgeräte, sondern auch eine umfassende Softwareplattform, die über umfangreiche Bibliotheken und einer Laufzeitumgebung verfügt, als auch mobile Schlüsselapplikationen wie der Telefon- oder SMS-App, zur Verfügung stellt. Android ist in Java programmiert und kann daher herstellerunabhängig und kostenlos genutzt werden. Große Teile der Softwareplattform sind Open-Source. [MOS09]

API

„Application Programming Interface“ (engl.) bedeutet übersetzt „Programmierschnittstelle“. Eine Programmierschnittstelle ermöglicht zwei oder mehr Programmen miteinander zu kommunizieren. [MOS09]

App

Als eine App wird häufig eine Softwareanwendung für mobile Endgeräte wie Smartphones oder Tablets bezeichnet. Die Bezeichnung ist eine Abkürzung für den englischen Begriff „Application“ und bedeutet übersetzt Anwendung. Eine App kann daher auch eine Softwareanwendung für andere IT-Ressourcen, wie einer Speicherprogrammierbaren Steuerung (SPS) oder anderer Hardware stehen. [VER12]

Auto-ID-Technologie

Mit der Auto-ID-Technologie sind Verfahren zur automatischen Datenerfassung, wie Barcodes, RFID, als auch Chipkarten oder biometrische Verfahren wie dem Fingerabdruckverfahren gemeint. Die Hauptaufgabe einer Auto-ID-Technologie ist die Bereitstellung von Daten eines Produktes oder eines Gutes. [FIN15]

Big Data

Big Data werden große und komplexe Datenmengen genannt, die im Zuge der Digitalisierung von Technologien anfallen. Bezogen auf die Produktion beinhalten diese sowohl produktionsbezogene Daten als auch maschinenbezogene Daten, die durch intelligentes Analysieren der Daten Prognosen und Entscheidungshilfen für zukünftige Entwicklungen geben. [SMA15]; [MAN16]

Creative Commons CC0

Creative Commons CC0 ist eine Lizenz die besagt, dass kein Urheberrechtsschutz vorhanden ist. Dadurch ist es möglich, Grafiken, die diese Lizenz besitzen zu kopieren, zu verändern und zu verbreiten, ohne eine Quelle angegeben zu müssen. [CRE17]

Elektronische Preisschilder

Elektronische Preisschilder (engl. „Electronic Shelf Labels“, kurz: ESL) sind kleine Displays, in denen üblicherweise Produktdaten und Preise im Einzelhandel angezeigt werden. Per WLAN können die Displays mit neuen Daten beschrieben werden. In dieser Arbeit werden die elektronischen Preisschilder zur Anzeige von Auftragsdaten genutzt [SIM16]. Die verwendete Displaytechnologie heißt E-Paper. Jeder Pixel in dem Display kann seinen Zustand ohne Energiezufuhr halten. Lediglich beim Wechsel des Zustandes muss Energie hinzugeführt werden. Dies wirkt sich positiv auf eine lange Akkulaufzeit aus. [ADA10]

ERP

ERP (Enterprise-Resource-Planning (engl.)) ist ein System, mit dem funktionsübergreifend alle Geschäftsprozesse des Unternehmens unterstützt werden. [KUR05]

Intent

Mit Intents können Daten einer Activity in eine andere Activity oder ein anderes System gesendet werden. Die Nachrichten können neben Daten auch Aktivitäten beinhalten. aktivieren die Komponenten einer App oder des Betriebssystems. [MOS09]

iOS

iOS ist ein Betriebssystem des Unternehmens Apple Inc. und ist für mobile Endgeräte ihrer eigens entwickelten Produkte vorgesehen. [KOL11]

Laserscanner

Bei Laserscannern handelt es sich um Scanner, die eine zweidimensionale Flächen- oder dreidimensionale Raumüberwachung bietet. Ebenso kann ein Laserscanner Positionen von Objekten im Raum oder in einer Fläche ermitteln. [LAS17]

Losgröße

Mit der Losgröße ist die Menge einer Produktart gemeint, die in einer ununterbrochenen Wertschöpfungskette gefertigt wird. [SPR17]

Mock-Up

Ein Mock-Up ist ein früher Prototyp einer App, in der nur die grundlegendsten Funktionen umgesetzt werden. Dazu gehört hauptsächlich das Navigieren zu den verschiedenen Seiten (Activities) der App. Layout und Design sind hierbei nebensächlich. [AIC16]

Native App

Als eine native App wird eine App bezeichnet, die für ein bestimmtes Betriebssystem entwickelt und ausschließlich mit diesem Betriebssystem nutzbar ist. Bevor diese auf dem Gerät genutzt werden kann, muss diese installiert werden. Vorteil einer nativen App ist die gute Performance und die Möglichkeit durch plattformspezifische Programmierung Zugriff auf spezielle Hardwarebestandteile des Gerätes zu bekommen. Eine nicht native App hingegen ist eine App, die betriebssystemunabhängig und ohne vorherige Installation funktioniert. [STE17b]

QR-Code

Ein QR-Code ist ein 2-Dimensionaler Barcode, der große Mengen an Informationen im Vergleich zu 1-Dimensionalen Barcodes speichern kann. Ziel bei Barcodes und auch QR-Codes ist es, automatisch Produktinformationen zu erfassen, oder aber Daten durch Kamerasysteme optisch erfassen zu können. [FIN15]

RFID

Bei RFID-Systemen werden Daten auf einem elektronischen Datenträger gespeichert. Der Datenaustausch erfolgt unter Verwendung von elektromagnetischen Feldern. Die technischen Verfahren wurden aus der Funk- und Radartechnik übernommen und steht deshalb für Radio-Frequency-Identification. [FIN15]

Smart Data

Informationen, die durch die Analyse von Big Data entstehen und für künftige Handlungen und Entscheidungen in einer Wertschöpfungskette genutzt werden, wird auch Smart Data genannt. [VOG17c]; [SMA15]

Smart Factory

Eine Smart Factory (auch intelligente Fabrik genannt) ist eine Fabrik, in der das zu fertigende Produkt seinen Produktionsprozess selbst steuert. [MAN16]

Smart Produkt

Smarte Produkte (auch intelligente Produkte genannt) sind Produkte, die Informationen über ihren Herstellungsprozess und ihren künftigen Einsatz besitzen. Sie unterstützen aktiv ihren eigenen Fertigungsprozess. [HEI13]

Toast

Als Toasts werden Kurzbenachrichtigungen in einer Android-App bezeichnet, die für wenige Sekunden in den Vordergrund der App treten. Der Toast kann nicht angeklickt werden und fordert keine Interaktion mit dem Benutzer. [MOS09]

Tracking

Tracking (engl.) bedeutet übersetzt „Nachverfolgung“. Im Kontext dieser Arbeit steht Tracking für die Nachverfolgbarkeit eines Produktes in einer Wertschöpfungskette. Trackingdaten geben Rückschlüsse auf den Produktionsstatus und Aufenthaltsort des zu produzierenden Produktes. [TTG08]

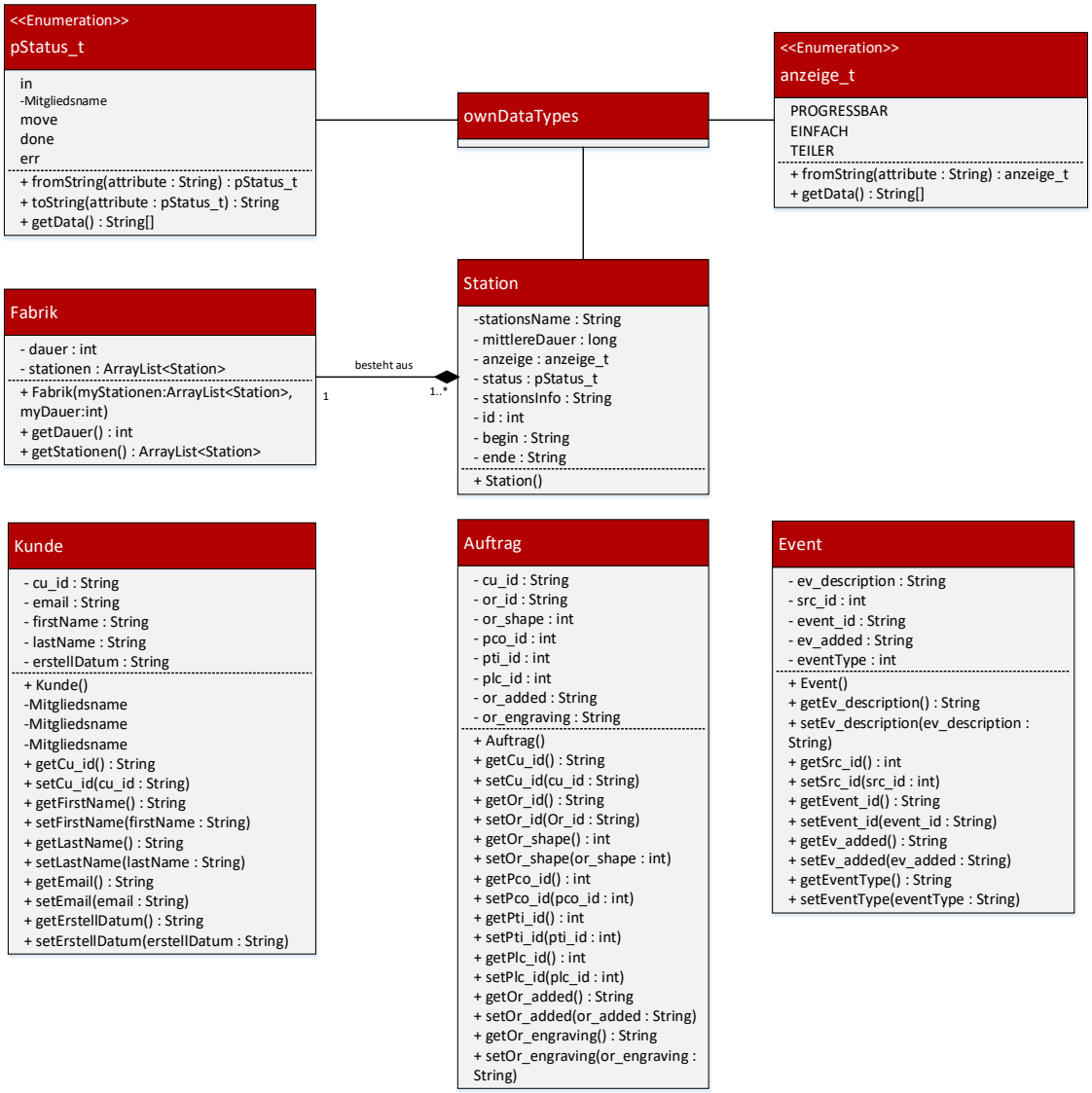
Tracking-System

Tracking-Systeme werden genutzt, um in einem Raum die Position eines gewünschten Objektes bestimmen zu können. [ADV11]

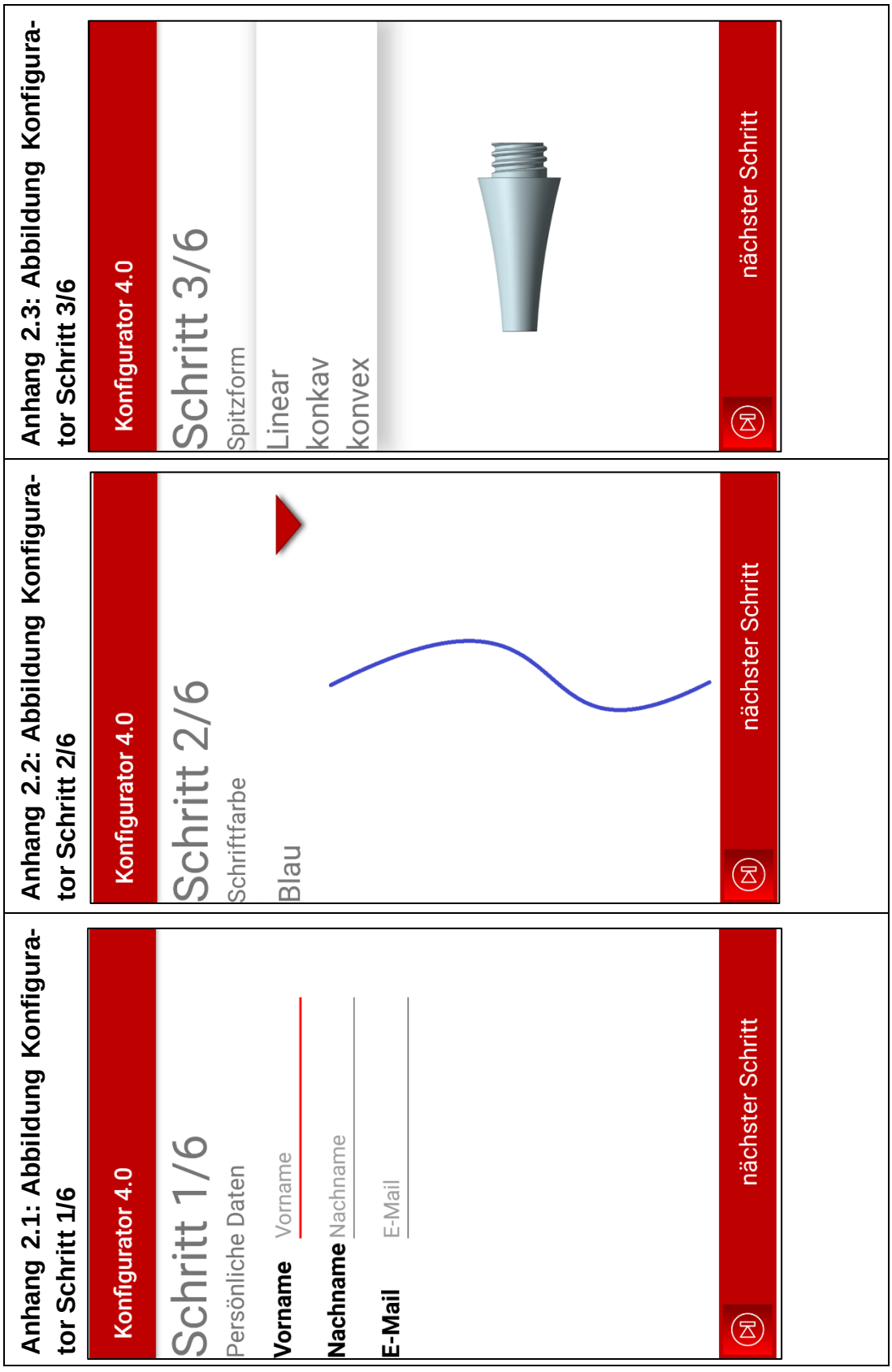
Usability

Usability (engl.) bedeutet übersetzt Gebrauchstauglichkeit oder Benutzerfreundlichkeit. Spricht man von einer hohen Usability, bedeutet dies, dass die Bedienung einen niedrigen Schwierigkeitsgrad aufweist. [TOR17]; [GRA15]

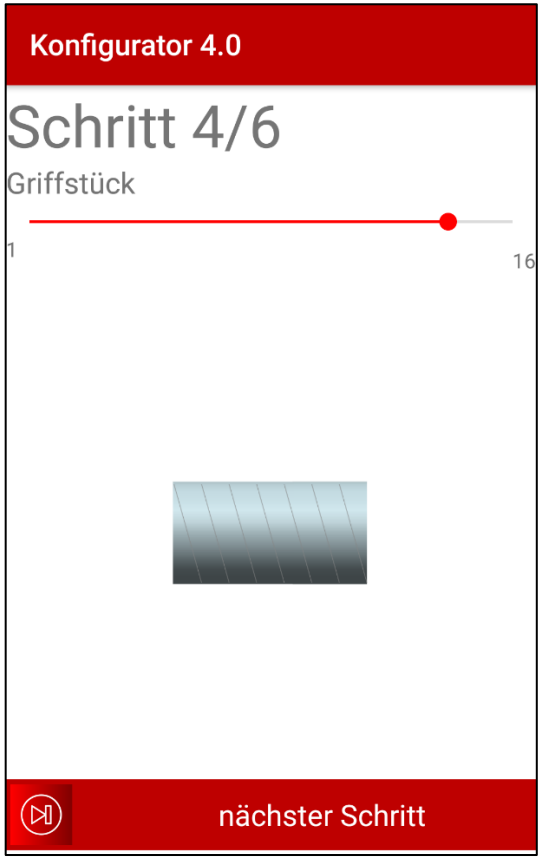
8.2. Anhang 1: Klassendiagramm Androidprojekt



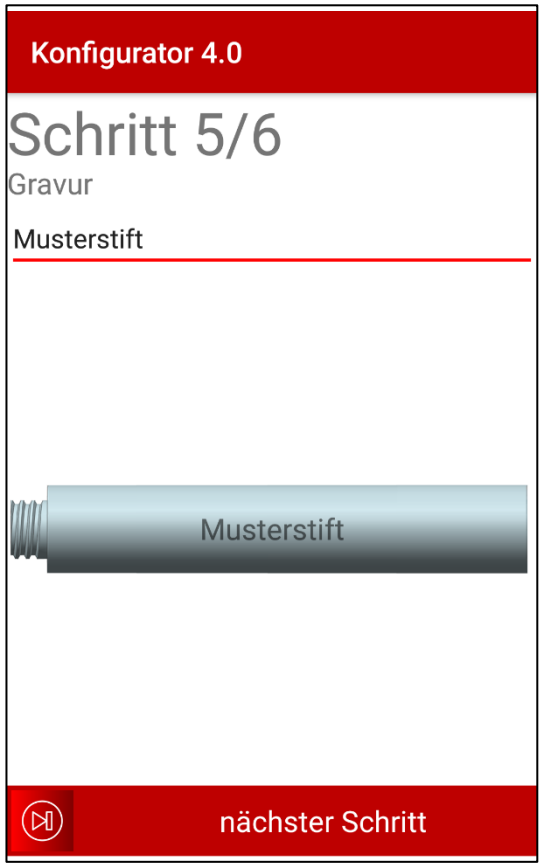
8.3. Anhang 2: Screenshots des Konfigurators der App



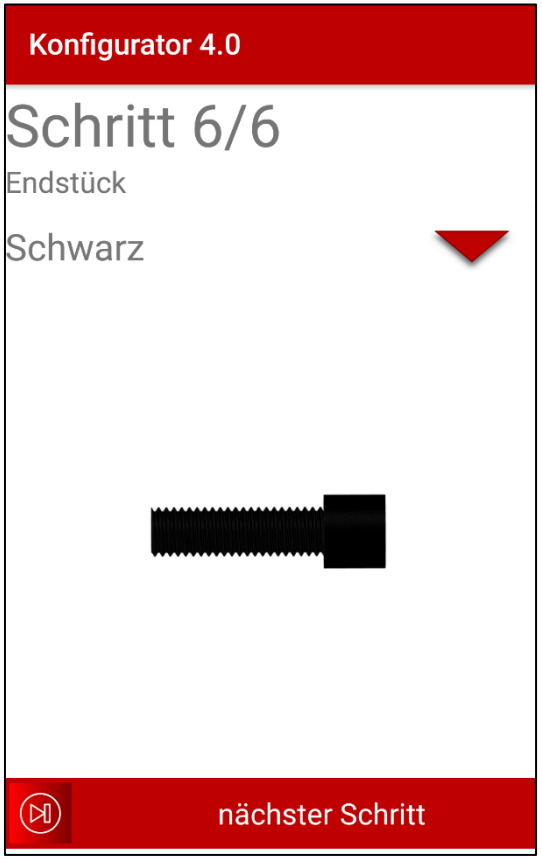
Anhang 2.4: Abbildung Konfigurator Schritt 4/6



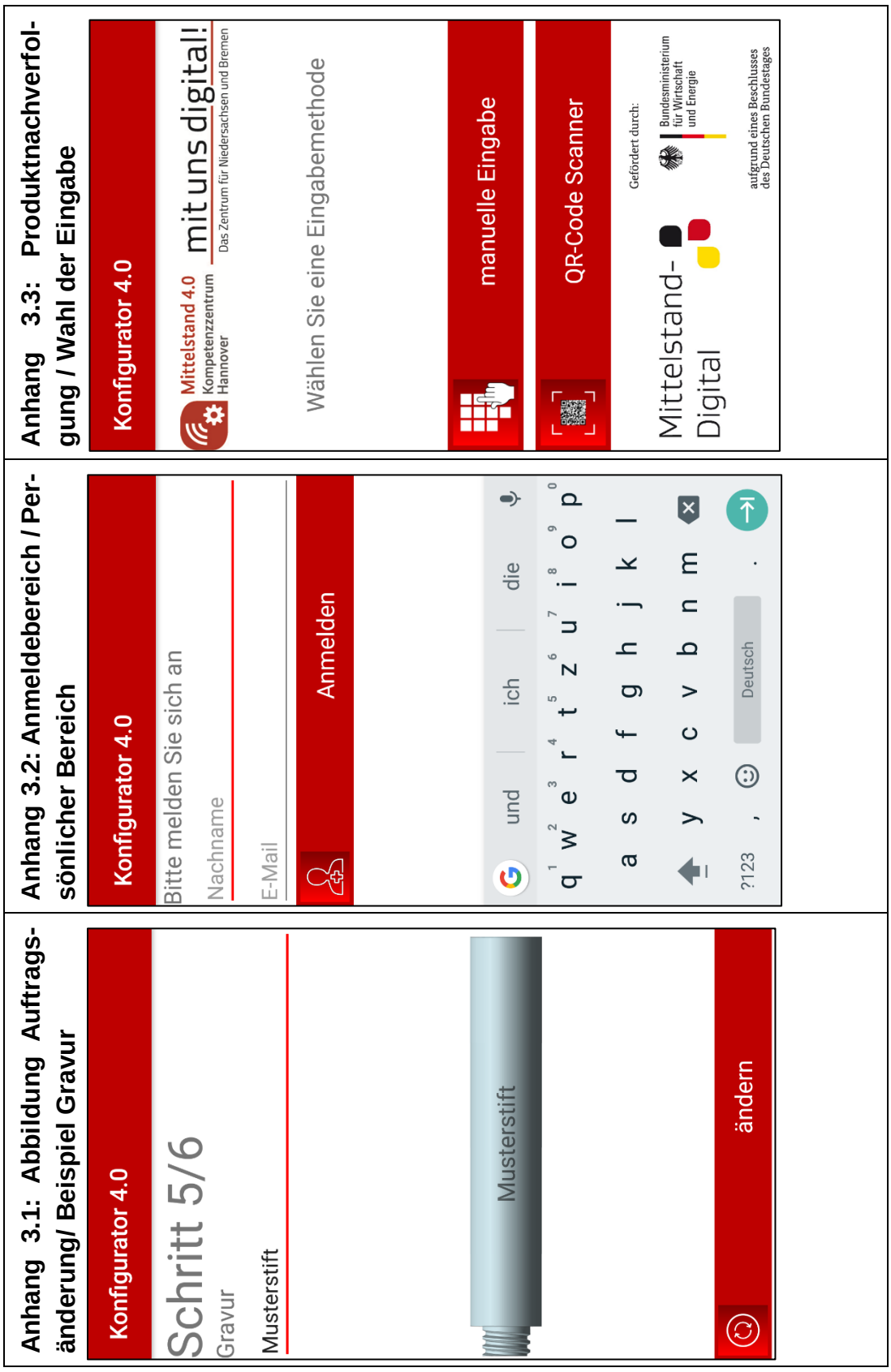
Anhang 2.5: Abbildung Konfigurator Schritt 5/6



Anhang 2.6: Abbildung Konfigurator Schritt 6/6



8.4. Anhang 3: Weitere Screenshots der App



8.5. Anhang 4: Spezifikation Events

Generalfabrik

Stationsnummer	Id	Name	event Eingang	event Ausgang	event Tastendruck
2	2	Hochzeit	Fab_order_started	Fab_order_finished	
3	3	Drehen	Fab_turning_started	Fab_turning_finished	Fab_turning_move
4	4	Nachbearbeiten	Fab_burr_started	Fab_burr_finished	Fab_burr_move
5	5	Kommissionierung	Fab_com_started_0/4		
5	5	Kommissionierung	Fab_com_update_1/4		
5	5	Kommissionierung	Fab_com_update_2/4		
5	5	Kommissionierung	Fab_com_update_3/4		
5	5	Kommissionierung	Fab_com_update_4/4		
5	5	Kommissionierung		Fab_com_finished	
6	6	Laser gravur	Fab_laser_started	Fab_laser_finished	Fab_laser_move
7	7	Montage	Fab_montage_started	Fab_montage_finished	
8	8	Qualitätsprüfung	Fab_robot_started	Fab_robot_finished	Fab_robot_move

mobile Fabrik

Stationsnummer	Id	Name	event Eingang	event Ausgang	event Tastendruck
2	2	Hochzeit	Bus_order_started	Bus_order_finished	
3	3	Drehen	Bus_turning_started	Bus_turning_finished	Bus_turning_move
5	5	Kommissionierung	Bus_com_started_0/4		
5	5	Kommissionierung	Bus_com_update_1/4		
5	5	Kommissionierung	Bus_com_update_2/4		
5	5	Kommissionierung	Bus_com_update_3/4		
5	5	Kommissionierung	Bus_com_update_4/4		
5	5	Kommissionierung		Bus_com_finished	
6	6	Laser gravur	Bus_laser_started	Bus_laser_finished	Bus_laser_move
7	7	Montage	Bus_montage_started	Bus_montage_finished	
8	8	Qualitätsprüfung	Bus_robot_started	Bus_robot_finished	Fab_robot_move

Legende		
	event Eingang	wenn Tray aufgelegt wird
	event Ausgang	wenn Tray abgezogen wird
	event Tastendruck	wenn Interaktion mit Bediener

8.6. Anhang 5: Klasse Parser

Beispiel einer Auftragsanfrage an den Webserver mit Erstellung von Auftragsobjekten für die Weiterverarbeitung in der App.

```
private static final String URL_EVENTS = "quest=events&or_id=";

public static ArrayList<Auftrag> Parse_Auftraege(String kundennummer) {
    ArrayList<Auftrag> myAuftraege = new ArrayList<~>();
        try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            //hier wird ein XML-Dokument über eine HTTP-Anfrage angefragt.
            //die http-Anfrage stellt sich aus einigen Url_Fragmenten zusammen, die aus Sicherheitsaspekten abgeändert wurden
            Document doc = dBuilder.parse(new URL(MAIN_URL + REQUEST_URL + URL_AUFTRAEGE + kundennummer).openStream());
            doc.getDocumentElement().normalize();
            //Alle Tags mit dem Namen "Order" werden in der NodeList nList gespeichert
            NodeList nList = doc.getElementsByTagName("order");
            //Über Jeden Tag in der NodeList werden die Attribute ausgelesen und in einem Objekt gespeichert
            for (int temp = 0; temp <= nList.getLength(); temp++) {
                //Aktueller Tag wird in nNode gespeichert
                Node nNode = nList.item(temp);
                if (nNode != null) {
                    if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                        Element eElement = (Element) nNode;
                        //neues Objekt der Klasse Auftrag wird erstellt
                        Auftrag myAuftrag = new Auftrag();
                        //folgend werden die Attribute des Auftragsobjektes mit den Attributen des Tags ersetzt.
                        //Die Attribute der Tags sind verschlüsselt und müssen daher mit der Methode
                        //AuthKey.entschluessel entschlüsselt werden
                        myAuftrag.setOr_id(AuthKey.entschluessel(eElement.getAttribute("or_id")));
                        myAuftrag.setOr_added(AuthKey.entschluessel(eElement.getAttribute("or_added")));
                        myAuftrag.setOr_engraving(AuthKey.entschluessel(eElement.getAttribute("or_engraving")));
                        myAuftrag.setOr_shape(Integer.parseInt(AuthKey.entschluessel(eElement.getAttribute("or_shape"))));
                        myAuftrag.setPco_id(Integer.parseInt(AuthKey.entschluessel(eElement.getAttribute("pco_id"))));
                        myAuftrag.setPlc_id(Integer.parseInt(AuthKey.entschluessel(eElement.getAttribute("plc_id"))));
                        myAuftrag.setPti_id(Integer.parseInt(AuthKey.entschluessel(eElement.getAttribute("pti_id"))));
                        myAuftraege.add(myAuftrag);
                    }
                }
            }
```

8.7. Anhang 6: Übersicht aller verwendeten Icons

8.8. Anhang 7: Webserver - Wandlung http-Anfrage in SQL-Statement

```
/**Anfrage Auftrag mit Auftragsnummer*/
public static Auftrag getAuftrag(String Auftragsnummer){
Statement stmt = null;
//Initialisiere neuen Auftrag, um Daten aus Datenbank in Form eines Objektes vorliegen zu haben
Auftrag currAuf = new Auftrag();

try {
    //Statement für Datenbankkommunikation
    stmt = connection.createStatement();
    //gibt die Einträge in der Tabelle DB_ORDERS für die Auftragsnummer or_id zurück
    ResultSet rs = stmt.executeQuery("SELECT * FROM "+DB_ORDERS+" WHERE or_id='"+ Auftragsnummer + "'");

    //Auslesen der Antwort des Statements
    while (rs.next()) {
        //Werte aus der Datenbank werden in Attribute eines Auftragsobjektes gespeichert
            currAuf.setOr_id(rs.getInt(1));
            currAuf.setCu_id(rs.getInt(2));
            currAuf.setOr_shape(rs.getInt(3));
            currAuf.setPco_id(rs.getInt(4));
            currAuf.setPti_id(rs.getInt(5));
            currAuf.setOr_added(rs.getString(6));
            currAuf.setOr_engraving(rs.getString(7));
            currAuf.setPlc_id(rs.getInt(8));
            }

    if (stmt != null) {
        stmt.close();
    }
} catch (SQLException e1) {
    e1.printStackTrace();
}
//gibt einen Auftrag zurück
return currAuf;
```