Nutzerzentrierte Entwicklung eines digitalen Bestellprozesses für Rohmaterialien

Bachelorarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Herbstsemester 2022

Autoren:	Abdullah Almaz & Ursin Zimmermann
Betreuer:	Prof. Dr.-Ing. Frieder Loch
Projektpartner:	Adrian Scherrer (SFS Group AG, Heerbrugg)
Experte:	Dr.-Ing. Marius Orfgen
Gegenleser:	Prof. Dr. Markus Stolze

Abstract

Die SFS Group AG ist ein weltweit führendes Unternehmen für applikationskritische Präzisionskomponenten und Baugruppen, mechanische Befestigungssysteme, Qualitätswerkzeuge und Logistiksysteme. Am Standort Heerbrugg existiert ein individueller interner Bestellprozess zwischen dem Lager und der Produktion, welcher über den Austausch von Papierformularen erfolgt. Er ist träge, ineffizient und fehleranfällig und weist unnötige Mehraufwände auf, weshalb er optimiert werden sollte.

In dieser Arbeit wird die Digitalisierung und Optimierung dieses Bestellprozesses mit dem Fokus auf der Anwendung von nutzerzentrierten Entwicklungsmethoden behandelt. Um die Umstellung auf einen digitalen Prozess möglichst angenehm für die Nutzer zu gestalten, wurde eine Literaturrecherche zum Thema User-centered Design und Dashboard-Design für das Industrieumfeld durchgeführt. Das neu angeeignete Wissen wurde im Rahmen der Arbeit direkt angewendet. Zudem wurde bei der SFS Group AG ein Contextual Inquiry nach dem Master-Apprentice-Model durchgeführt, um die Ziele und die Aufgaben der Nutzer zu verstehen. Basierend auf der Nutzeranalyse wurden Personas erstellt und Wireframes designt, welche mit den Anwendern nochmals diskutiert und optimiert wurden.

Im Verlauf der Arbeit wurde ein Prototyp umgesetzt, Usability-Tests unterzogen und weiter optimiert. Er erfüllt alle definierten User-Storys und entspricht weiteren Anforderungen, beispielsweise berücksichtigt er Cloud-native Standards. Das Feedback seitens der SFS Group AG war sowohl beim Usability-Test als auch bei der Evaluation positiv. Der Prototyp konnte den technischen Verantwortlichen des Unternehmens übergeben werden und wird in Zukunft im produktiven Umfeld eingesetzt werden.

Management-Summary

Nutzerzentrierte Entwicklung eines digitalen Bestellprozesses für Rohmaterialien

Diplomanden	Abdullah Almaz und Ursin Zimmermann
Betreuer	Prof. Dr.-Ing. Frieder Loch
Experte	Dr.-Ing. Marius Orfgen
Gegenleser	Prof. Dr. Markus Stolze
Themengebiet	Internet-Technologien und -Anwendungen
Industriepartner	SFS Group AG

Ausgangslage

Die SFS Group AG verarbeitet interne Bestellungen von Rohmaterial zwischen der Produktion und dem Lager über einen Papierprozess. Dabei werden Bestellformulare von der Produktion in Behältern deponiert und vom Lager abgeholt, das dann die Bestellung bearbeitet und an die Produktion liefert. Dieser Prozess soll in Zukunft durch eine Software abgelöst werden, da er aktuell träge, wenig effizient und fehleranfällig ist. Im Rahmen dieser Arbeit sollte ein nutzerzentrierter Prototyp geplant, entwickelt und der SFS Group AG zur Weiterentwicklung übergeben werden.

Vorgehen

Gemäss dem *User-centered Design*-Prozess wurde als vorgängige Analyse bei der SFS Group AG ein *Contextual Inquiry* durchgeführt, im Rahmen derer die Mitarbeiter und zukünftigen Nutzer der Software bei der Arbeit beobachtet und befragt wurden. Basierend auf den gesammelten Informationen wurden erste konkrete Anforderungen festgehalten und *Wireframes* erstellt. Diese wurden im Austausch mit der SFS Group AG und den zukünftigen Nutzern optimiert und wo nötig erweitert. Anschliessend wurde der Prototyp mit den vereinbarten Technologien entwickelt, getestet und auf einem Testserver zur Verfügung gestellt. Mit *Usability-Tests* bei der SFS Group AG wurde er von den Endnutzern auf Tauglichkeit geprüft. Das Feedback war positiv und die daraus entstandenen Optimierungen wurden umgesetzt oder zur Weiterentwicklung dokumentiert.

Ergebnis

Der entwickelte Prototyp entspricht dem geplanten Umfang und es werden weitere Anforderungen der SFS Group AG eingehalten, beispielsweise die Erfüllung der *Twelve Factors* und das Schaffen einer Basis für das *Deployment* mit *Docker-Images*. Bei einer gemeinsamen Evaluation mit der SFS Group AG wurde sichergestellt, dass die technischen Fragen und Unklarheiten geklärt sind und der Prototyp vom Unternehmen weiterentwickelt werden kann. Die Integration mit den Umsystemen der SFS Group AG sowie das *Deployment* der Applikation auf die Cloudumgebung werden auf Wunsch der Firma von dieser selbst durchgeführt.

Danksagung

Wir möchten uns an dieser Stelle bei allen Personen bedanken, welche uns während dieser Bachelorarbeit unterstützt haben. Als Erstes möchten wir uns bei unserem Betreuer Prof. Dr.-Ing. Frieder Loch für seine Hilfe und Betreuung durch die ganze Arbeit bedanken.

Weiter möchten wir uns auch bei unserem Experten Dr.-Ing. Marius Orfgen für seine Unterstützung und Inputs bedanken.

Des Weiteren bedanken wir uns bei unserem Projektpartner, der SFS Group AG– namentlich bei Adrian Scherrer, Oliver Gächter, Beat Hofstetter und Gottfried Frei –, für die gute Zusammenarbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Aufgabenstellung

Entwicklung eines Bestellsystems für Rohmaterial

Beteiligte Personen

Diese Bachelorarbeit wird verfasst von

Abdullah Almaz
Ursin Zimmermann

Betreuer dieser Arbeit ist

Prof. Dr.-Ing. Frieder Loch

Betreuer der Arbeit bei SFS ist

Adrian Scherrer

Problembeschreibung

Die SFS Group AG ist ein Unternehmen für applikationskritische Präzisionskomponenten und Baugruppen, mechanische Befestigungssysteme, Qualitätswerkzeuge und Logistiksysteme.

Die Bestellung von Rohmaterialien (z.B. Draht) erfolgt derzeit manuell auf Papier. Dabei wird ein Zettel ausgefüllt und an einer vorgegebenen Position platziert. Dieser wird dann von einer weiteren Person eingesammelt und abgearbeitet. Dieser Prozess ist träge und wenig effizient. Neben den inhärenten Nachteilen der Bearbeitung auf Papier fehlt dem Personal ein aktueller Überblick über den Status von Bestellungen. Weiterhin können Bestellungen nicht priorisiert werden, da die Bestellungen in einem festen Intervall abgeholt werden.

Dieser Prozess soll nun digitalisiert werden. Es soll ein Front- und ein Backend konzipiert, prototypisch umgesetzt und evaluiert werden. Die Ergebnisse der Arbeit sollen die Grundlage für die Entwickelung einer produktiven Anwendung liefern.

Formulierung eines konkreten Auftrags

1. **Anforderungserhebung.** Der bestehende Bestellprozess und die Anforderungen der Produktionsumgebung werden untersucht. Hierzu werden Besuche vor Ort, die Gespräche mit dem beteiligten Personal umfassen, durchgeführt werden.
2. **Systementwurf.** Das System wird auf Grundlage der Anforderungserhebung entworfen. Nach Möglichkeit werden Anforderungen von SFS berücksichtigt. Bei der Entwicklung der Anwendung zum Bearbeiten der Bestellungen im Lager werden nutzerzentrierten Entwicklungsansätze angewendet. Die Anwendung wird nach Cloud Native Standard entwickelt.
3. **Implementierung.** Die Anwendung wird prototypisch implementiert. Eine Integration in die produktiv eingesetzten Umsysteme ist nicht notwendig.
4. **Evaluation.** Das System wird mit Personal von SFS evaluiert. Hierbei wird die Wahrnehmung des Systems durch das Personal mit Beobachtungen und Interviews erhoben und die technische Eignung des gewählten Ansatzes untersucht.
5. **Dokumentation.** Die Entwurfsentscheidungen und Teilergebnisse werden dokumentiert. Es erfolgt eine kritische Reflektion und es werden Handreichungen für die weitere Realisierung des Systems formuliert.

Umfang und Form der erwarteten Resultate

Als Resultate der Arbeit entsteht eine schriftliche Dokumentation der Projektergebnisse und des Vorgehens und der erstellte Programmcode. Diese werden den Projektbeteiligten spätestens zum Ende der Bearbeitungszeit zur Verfügung gestellt.

Anfangs- und Abgabetermin

- Start der Bearbeitung: **19. September 2022**
- Abgabe: **13. Januar 2022 (17:00 Uhr)**

Zulässige Hilfsmittel und weitere Betreuung

Alle verwendeten Hilfsmittel werden in der Arbeit aufgeführt.

Die Betreuung erfolgt durch die genannte Betreuungsperson. Es werden wöchentliche Besprechungen mit dem Betreuer an der Hochschule durchgeführt. Mit den Betreuungspersonen bei SFS werden nach Bedarf Besprechungen durchgeführt und von diesen bei Bedarf weitere Personen hinzugezogen.

1. Einführung

1.1 Problemstellung und Vision

Die SFS Group AG (im Weiteren als SFS abgekürzt) ist ein weltweit führendes Unternehmen für applikationskritische Präzisionskomponenten und Baugruppen, mechanische Befestigungssysteme, Qualitätswerkzeuge und Logistiksysteme. (SFS Group, 2022)

Am Standort Heerbrugg existiert ein individueller interner Bestellprozess zwischen dem Lager und der Produktion. Bei diesem Bestellprozess verarbeitet die Firma ihre internen Bestellungen von Rohmaterial aktuell über einen Papierprozess. Dabei werden die Bestellinformationen auf einem Formular (siehe *Abbildung 2*) festgehalten, das Formular in einem Behälter (siehe *Abbildung 1*) deponiert und von dort abgeholt. Danach wird die Bestellung weiterverarbeitet und schliesslich ausgeliefert. Dieser Prozess ist träge, ineffizient und hat weitere Nachteile. Daher soll er nun digitalisiert werden. Dadurch werden beispielsweise die Priorisierung, die Überschaubarkeit und die Nachverfolgbarkeit der Bestellungen ermöglicht. Im Rahmen dieser Arbeit wird ein Prototyp geplant und entwickelt, welcher die Basis für eine produktive Anwendung darstellt.

Abbildung 1: Sammelbehälter für Bestellungen bei der SFS Group AG
Quelle: Foto von Abdullah Almaz

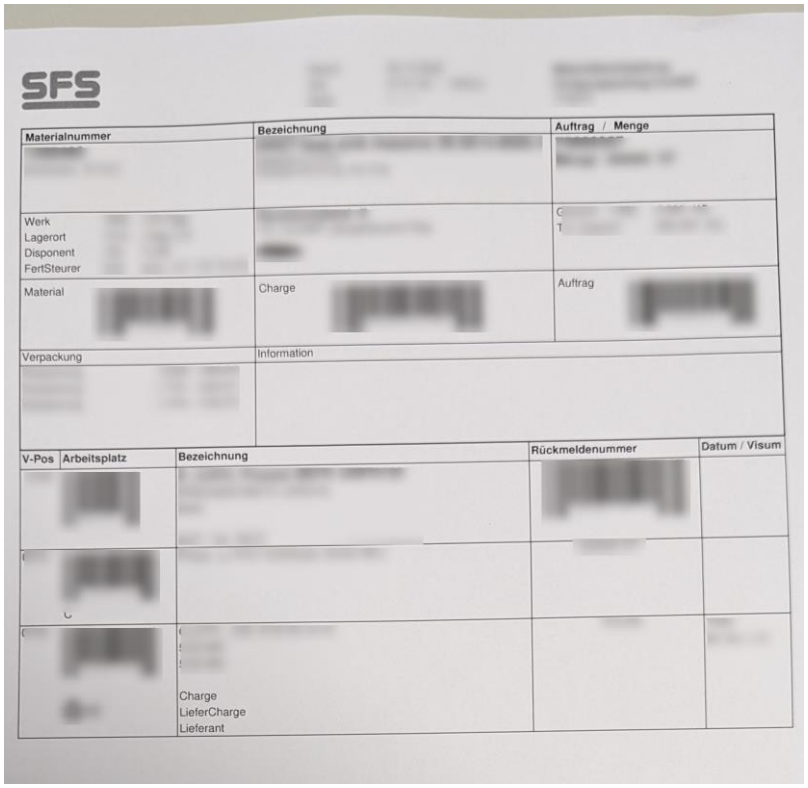


Abbildung 2: Bestellschein hinsichtlich Material, z. B. Draht
Quelle: Foto von Abdullah Almaz
Verschleierung aus Datenschutzgründen

Zurzeit existieren digitale Umsysteme bei der SFS, in welchen Bestellungen erfasst werden. Sie werden nach der Durchführung dieser Arbeit vom Unternehmen selbst erweitert. Im Kontext dieser Arbeit werden lediglich Schnittstellen für die bestehenden Umsysteme angeboten, mit welchen Bestellungen erfasst und priorisiert werden können. In der neuen Applikation werden jegliche Bestellungen erfasst, bearbeitet und überwacht.

1.2 Vorgehen und Aufbau der Arbeit

Als erster Schritt wird die SFS besucht. Dabei sollen alle Anforderungen und Fragen geklärt werden, zusätzlich soll eine Kontextanalyse und eine Nutzeranalyse durchgeführt werden. Nachdem die Anforderungen aufgenommen wurden, werden im nächsten Schritt Entwürfe für die Benutzeroberfläche erstellt. Diese dienen als Grundlage für die Diskussion mit der SFS, um die *User-Experience* zu verbessern.

Darauffolgend wird die Software nach dem vereinbarten Rahmen entwickelt und getestet. Priorität werden auch die nichtfunktionalen Anforderungen (NFRs) haben, da diese z. B. die erfolgreiche Übernahme und Weiterentwicklung seitens der SFS garantieren.

Gegen Ende der Arbeit werden *Usability-Tests* in Heerbrugg mit den Endbenutzern der SFS durchgeführt und allfällige Mängel entweder korrigiert oder in der Softwaredokumentation vermerkt. Bei der Evaluation mit der SFS erfolgt eine Demonstration der Software. Dies ermöglicht ein letztes Feedback vom Unternehmen zur umgesetzten Lösung und dient als Projektabschluss.

Wir haben uns entschieden ein Augenmerk auf die *Usability* zu legen und die Applikation für den Industriekontext zu optimieren. Des Weiteren haben wir festgestellt, dass die grösste geplante Herausforderung darin besteht, die Komplexität der Applikation und deren Benutzeroberfläche gering zu halten. Für die Literaturrecherche fiel die Entscheidung folglich auf das Themengebiet *User-centered Design*.

Abbildung 3 veranschaulicht den beschriebenen Prozess auf einer Zeitachse von fünfzehn Wochen.

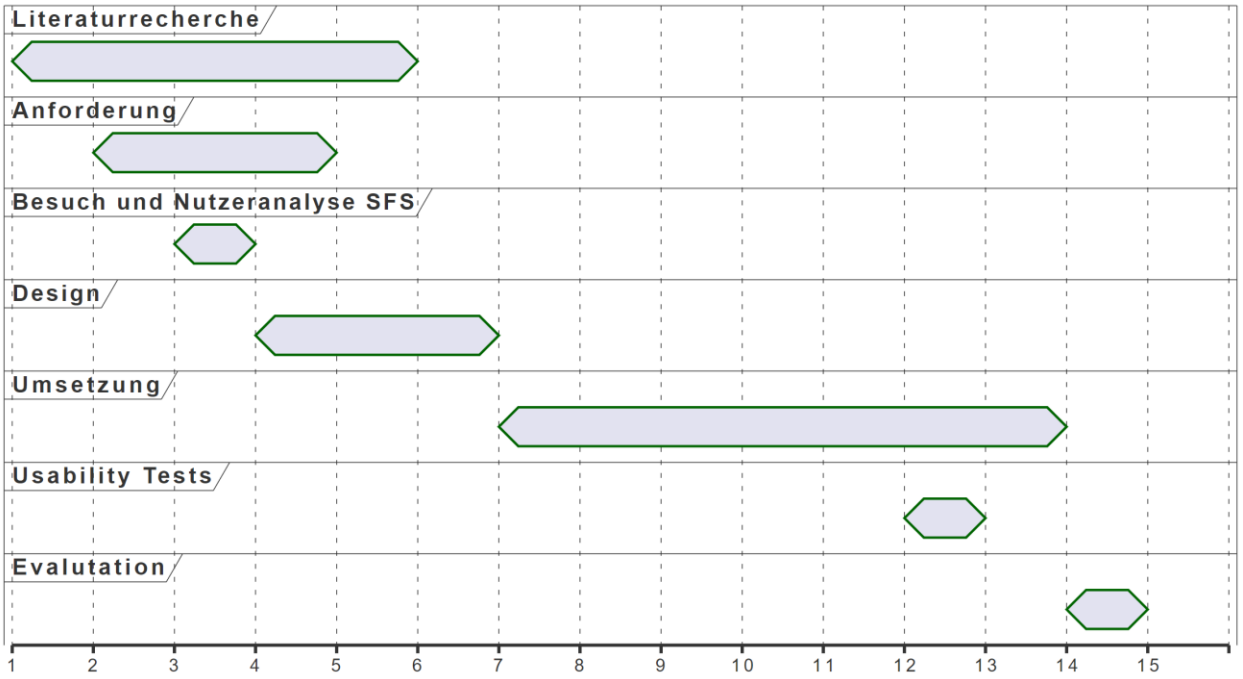


Abbildung 3: Geplanter Prozess
Quelle: Erstellt mit PlantUML

2. Literaturrecherche

Im Rahmen der Literaturrecherche wurden Bücher und wissenschaftliche Dokumente zum Thema *User-centered Design* evaluiert. Das Ziel dabei war eine theoretische Grundlage für die Arbeit zu haben.

2.1 Contextual Inquiry

Beim *Contextual Inquiry* handelt es sich um ein Konzept, mit welchem anhand von Beobachtungen und Interviews mit der Zielgruppe z. B. die Anforderungen an eine Applikation benutzerzentriert erhoben werden können. Wesentlich ist, dass der aktuelle Prozess (Ist-Zustand), welcher später abgebildet oder abgelöst wird, nicht nur von einer Person erklärt wird, sondern dass er auch direkt beobachtet oder daran teilgenommen wird. (Cooper et al., 2014, S. 44)

Es fällt vielen Menschen schwer ohne Kontext ihren Arbeitsprozess zu erklären. Dabei wird der Beschäftigte in eine Situation gebracht, in welcher er ohne Zusammenhang versuchen muss, einen komplexen Arbeitsprozess zu erklären. Das ist vergleichbar mit der Situation, wenn versucht wird, einem Fahrschüler in einem Sitzungszimmer das Autofahren beizubringen. Zum einen fehlt ihm die Strasse, welche ihm nur erklärt wird, zum anderen hat er kein Steuerrad, kein Gas- und kein Bremspedal, um zu üben. (Beyer & Holtzblatt, 2009, S. 37)

Daher sollte versucht werden, sich in die Position eines Lernenden zu versetzen, welcher vom Mitarbeiter in die Arbeit eingeführt wird. Dieses Vorgehen entspricht dem *Master-Apprentice-Model*. Es erfolgt am besten im gewohnten Arbeitsumfeld. Ein Lehrmeister wird einem Lehrling seine Tätigkeit auch nicht in einem Klassenzimmer erklären, sondern direkt bei der Arbeit. So muss er sich keine Gedanken machen, wie er dem Lernenden die Tätigkeit erklärt, sondern kann dies direkt an einem Praxisbeispiel demonstrieren. (Beyer & Holtzblatt, 2009, S. 42–43)

Die Vorbereitung auf ein *Contextual Inquiry* ist relevant. Vorgängige Fragen sollten aber minimiert werden; vielmehr sollte versucht werden, Schlüsse aus dem Beobachten des Arbeitsprozesses zu ziehen. Bedeutsam ist auch, dass während des Zuschauens versucht wird, die Erklärungen dahingehend zu lenken, dass eventuelle Unklarheiten direkt ausgeführt werden. Falls während der Durchführung des *Contextual Inquirys* Personen mit zu vielen Fragen eingeschränkt werden, kann dies zu verfälschten Antworten führen. (Cooper et al., 2014, S. 43–45)

Auf Fragen, die während der Beobachtung aufkommen resultieren bessere Antworten. Zusätzlich besteht die Möglichkeit, Fragen zu stellen, die sich während des Zuschauens ergaben. Es ist aber auch bedeutsam, Beobachtungen und Annahmen mit der interviewten Person zu besprechen, um falsche Erkenntnisse zu verhindern. (Cooper et al., 2014, S. 44–45)

2.2 Personas

Personas sind ein Modell, aus dem Bereich *Interaction-Design*. Sie ermöglichen es, nachzuvollziehen und darüber zu kommunizieren, wie sich bestimmte Nutzergruppen verhalten, was diese denken sowie was und weshalb sie etwas erreichen möchten. Die Personas sind ein aggregiertes Modell, welches die Beobachtungen und die Erkenntnisse aus einer qualitativen Nutzeranalyse, in diesem Fall aus dem *Contextual Inquiry*, zusammenfasst. (Cooper et al., 2014, S. 62)

Personas können mit einem Steckbrief einer Person verglichen werden (siehe *Abbildung 4*). Sie beinhalten das Verhaltensmuster, die Aufgabe, die Ziele und die Ängste einer Nutzergruppe bei der Verwendung eines Produkts. Diese Angaben werden mit Daten so ergänzt, dass auf Basis einer Persona die Vorstellung einer fiktiven Person ermöglicht wird. (Cooper et al., 2014, S. 67)

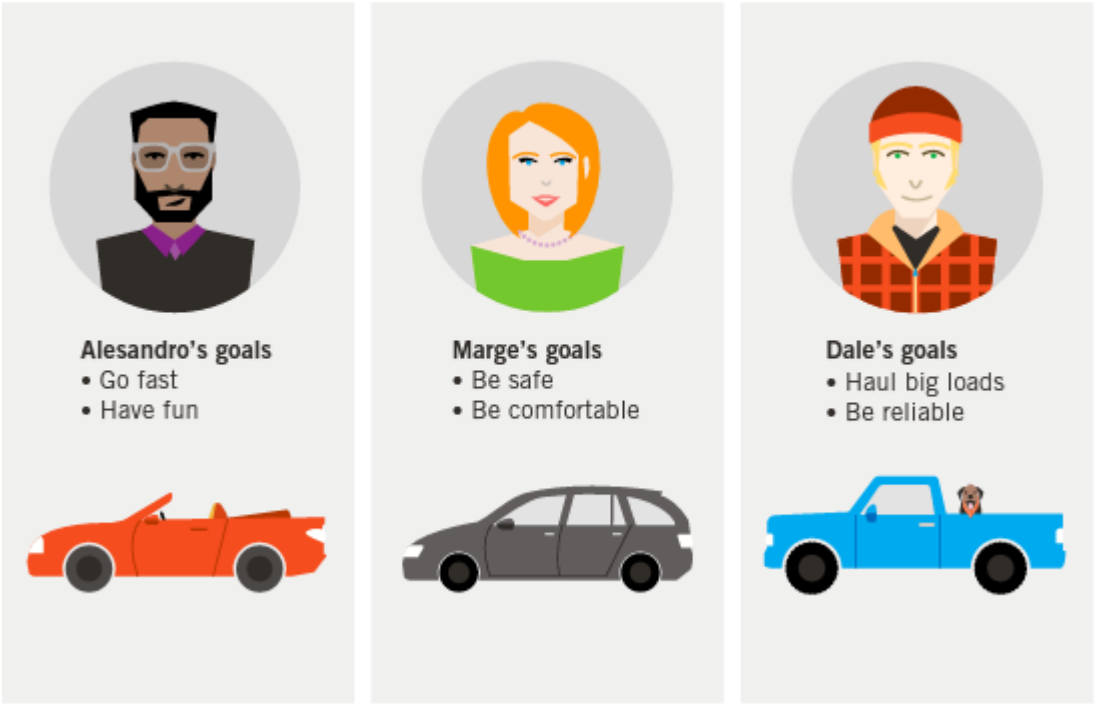


Abbildung 4: Beispiel-Personas für die Planung eines Autos
Quelle: (Cooper et al., 2014, S. 63)

> *Personas are more appropriate design tools than other user models.*
> *(Cooper et al., 2014, S. 69)*

Um dieses Zitat zu bestätigen, werden in der Literatur Personas mit anderen Modellen anhand ihrer Vor- und Nachteile verglichen. Was solche im Vergleich zu Personas limitiert, sind folgende Probleme:

- Die Kommunikation über das menschliche Verhalten und Beziehungen ist erschwert. Es fällt nicht leicht, Empathie bei einer starren Liste von Rollen oder Funktionalitäten einzubringen.
- Bei anderen Modellen liegt der Fokus auf den *Tasks*, es wird aber nicht oder nicht genug auf die *Goals* eingegangen.
  - *Tasks* sind Aufgaben, die ein Benutzer oder eine Gruppe ausführen muss oder möchte.
  - *Goals* sind Ziele oder die Motivation eines Benutzers oder einer Gruppe bei der Bedienung der Applikation.
- Es besteht kein einheitliches Modell für die Entwicklung, die Kommunikation und die Messung von Design-Entscheidungen.

(Cooper et al., 2014, S. 69–70)

Es gibt keine eindeutige oder perfekte Methode zur Herleitung von Personas. Daher wird in der Literatur ein Beispiel für einen guten Prozess zur Entwicklung einer Persona in achten Schritten beschrieben (siehe *Abbildung 5*). Der Beispielprozess behandelt das Interviewen von Nutzergruppen und das Erkennen von zusammenhängenden Verhaltensweisen bis hin zur Erstellung und Verfeinerung der Personas.

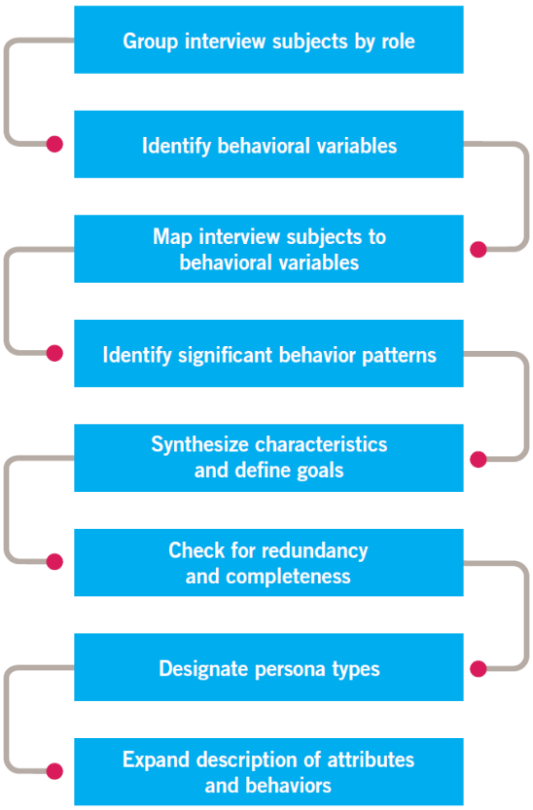


Abbildung 5: Persona-Herleitung in acht Schritten
Quelle: (Cooper et al., 2014, S. 82)

Dieser Prozess von Cooper wurde in dieser Arbeit nicht angewendet. Einerseits ist dieser darauf ausgelegt, dass entweder eine grosse Datenmenge vorliegt oder viele Endnutzer zum Interviewen zur Verfügung stehen und seitens der SFS konnten nicht viele Personen interviewt werden. Andererseits war ein so detaillierter Prozess für die vorliegende Arbeit aufgrund der zeitlichen Ressourcen nicht passend. Aus diesen Gründen wurde im Rahmen dieser Arbeit ein adaptierter Prozess herangezogen, wobei der Fokus auf der Kollaboration und den Erwartungen der verschiedenen Bereiche lag. Dieser adaptierte Prozess wird in *Abbildung 6* veranschaulicht.

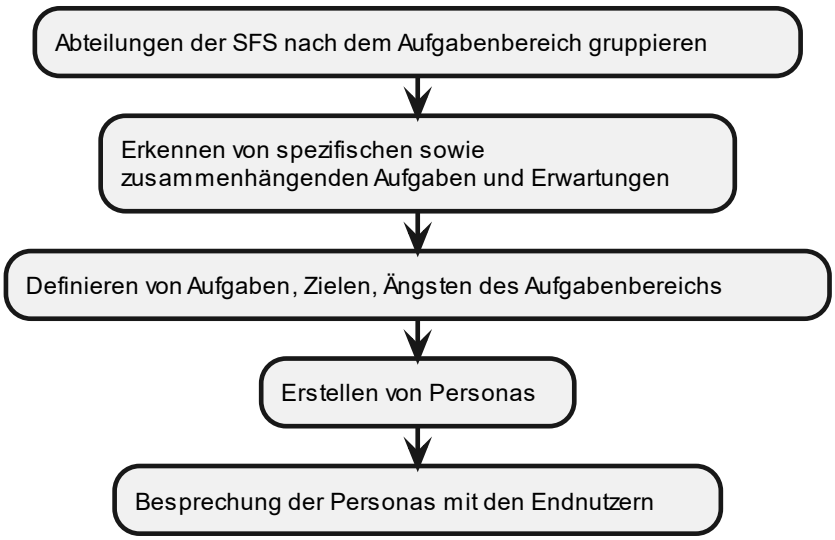


Abbildung 6: Adaptierter Prozess für Persona-Herleitung
Quelle: Erstellt mit PlantUML

Der adaptierte Prozess war wie folgt gestaltet:

- Die Punkte eins bis drei von Cooper wurden zusammengefasst, um die verschiedenen Abteilungen der SFS nach ihrem Aufgabenbereich zu gruppieren (eine Abteilung kann mehrere Aufgabenbereiche haben).
- Die Punkte vier und fünf von Cooper wurden ebenfalls kombiniert, damit spezifische und zusammenhängende Aufgaben und Erwartungen erkannt werden können.
- Der Schritt sechs und sieben von Cooper wurde praktisch übernommen. Bei diesen wurden die Aufgaben, die Ziele und die Ängste eines Aufgabenbereichs definiert und als Personas designt.
- Als letzter Schritt wurden die Personas mit den Endnutzern besprochen, um herauszufinden, ob deren Ansicht repräsentiert wird.

2.3 Usability-Testing

Nachdem das Design geplant und umgesetzt ist, sollte es verifiziert werden. Dieses Ziel wird durch *Usability-Tests* verfolgt.

Bei einem moderierten *Usability-Test* führt ein Benutzer vordefinierte Szenarien auf einer Applikation oder einem Prototyp aus. Durch die Beobachtung und das ‹laute Denken› des Testers können Erkenntnisse resultieren, an die zuvor nicht gedacht wurde. Mittels mehrfacher Durchführung mit verschiedenen Nutzern lassen sich generelle Probleme und einzelne Ausreisser unterscheiden. *Usability-Tests* mit fünf Personen reichen aus, um viele Probleme zu erkennen und Erkenntnisse zu gewinnen. (Marsh, 2022)

In der Literatur werden folgende Punkte als essenzielle Bestandteile eines *Usability-Tests* angesehen:

- Der Test sollte so spät durchgeführt werden, dass genug ‹Design› vorhanden ist, aber so früh, dass Änderungen möglich sind.
- Es sollen reale Szenarien getestet werden.
- Die Teilnehmer sollen die Hauptnutzer sein und am besten einer Persona entsprechen.
- Während der Ausführung der Szenarien sollen die Teilnehmer ‹laut denken›.

- Der Fokus für die Resultate liegt auf dem Verhalten der Tester und deren Gründen dafür.
- Unklarheiten zum beobachteten Verhalten sollen am Ende des Tests geklärt werden.

(Cooper et al., 2014, S. 142)

2.4 Entwurfsprinzipien für Dashboards

Im Rahmen der vorliegenden Arbeit wird ein *Dashboard* für eine Lagerhalle entwickelt. Daher lag ein weiterer Fokus bei der Literaturrecherche auf dem Designen eines möglichst nutzerfreundlichen *Dashboards* im Industrieumfeld.

Das Ziel eines *Dashboards* ist es, dem Nutzer eines Systems schnell einen Gesamtüberblick zu verschaffen. Dabei werden die bedeutsamsten Punkte und Daten zusammengefasst, sodass Anwender rasch und unkompliziert Trends, Muster oder nicht normale Situationen erkennen können. (Richard Brath & Michael Peters, 2004)

Ein wesentliches Element zur Informationsdarstellung auf einem *Dashboard* sind Farben. Auf deren Wahl hat das menschliche Auge Einfluss. Hierbei gilt zu beachten, dass dieses Licht aufnimmt und je nach Wellenlänge eine andere Farbe interpretiert. Dies hat Auswirkungen darauf, wie anstrengend es für Menschen ist, Farben zu unterscheiden oder zu erkennen. Daraus lassen sich zentrale Regeln ableiten, welche beim Gestalten mit Farben beachtet werden müssen:

- Dunkle Schrift auf hellem Hintergrund lässt sich leichter lesen als helle Schrift auf dunklem Hintergrund, da das Auge länger braucht, um von Hell auf Dunkel umzustellen.
- Bedeutsame Objekte am Bildschirmrand sollten nicht in grüner oder roter Farbe dargestellt werden, da die Netzhaut eine geringe Empfindlichkeit für diese Farben hat.
- Zu viele Farben auf einmal können das Auge ermüden und sollten daher vermieden werden.
- Nur grössere Objekte sollten eingefärbt werden, da sonst zu schmale Objekte schlecht wahrnehmbar sind.
- Da in Produktionsstätten Blende- und Verschmutzungsprobleme bestehen, sollte die Hintergrundfarbe Grau verwendet werden.

(Zühlke, 2012, S. 8–12)

Um ein möglichst einheitliches Bild zu schaffen, sollte beim Designen eine Farbpalette verwendet werden. Bei deren Zusammenstellung sollten folgende Punkte beachtet werden:

- Falls in Firmen Styleguides vorhanden sind, sollten diese herangezogen werden. Gibt es keine, kann anhand bisheriger Produkte versucht werden, ein Farbmuster zu erkennen.
- Es sollten nicht zu viele gesättigte Farben verwendet werden. Diese ziehen den Fokus schnell auf sich, weshalb bei gehäuftem Vorkommen die Gefahr besteht, dass der Nutzer überreizt ist und bedeutsame Elemente übersieht.

(Shannon Brown, 2018)

Es sollten aber nicht nur Farben verwendet werden, um Informationen anzuzeigen, vielmehr sollten immer ergänzende Mittel genutzt werden. Beispielsweise sollte ein Systemstatus im Falle eines Fehlers nicht nur in Rot angezeigt werden, sondern es sollte zusätzlich einen Signalton, eine Fehlermeldung oder einen weiteren visuellen Hinweis geben. (Human Factors and Ergonomics Society, 2008)

Bei einem *Dashboard* ist nicht nur eine schöne Darstellung bedeutsam. Vielmehr sollte der Prozess so abgebildet werden, wie er ausgeführt wird. Das heisst, dass bei einem Ablauf aus vier Schritten dieser auf

dem *Dashboard* in der korrekten Reihenfolge dargestellt wird, statt etwa zuerst den dritten Schritt und danach den ersten Schritt abzubilden. (Zühlke, 2012, S. 184–185)

2.5 Diskussion

Zur Einleitung der Diskussion verwenden wir folgendes Zitat, welches den Vorteil des *User-centered Design* zusammenfasst:

> *Research Is Critical to Good Design.*
> *(Cooper et al., 2014, S. 59)*

Nach der Literaturrecherche wurde das *User-centered Design* als essenziell für Softwareprojekte erachtet. Einem Mehraufwand zu Beginn aufgrund der Analyse, der Auswertung und der Diskussionen steht die Einsparung unnötiger Funktionalitäten in der Umsetzung gegenüber. Es wird eine Software angefertigt, welche auf die Bedürfnisse und die Ziele des Benutzers abgestimmt ist.

Nach Meinung der Verfasser sollte in der heutigen Zeit der agilen Prozesse das *User-centered Design* genauso bedeutsam und verbreitet sein wie z. B. die Entwicklung nach *Scrum*. Denn auch hierbei würde rasches Feedback vom Kunden oder von Benutzern einen Mehrwert bieten und unerwarteten Überraschungen im späteren Verlauf des Projekts vorbeugen.

Ein bekanntes Beispiel eines gefloppten Projekts ist das des *Segways*. Heute sind *E-Scooter* und ähnliche elektrische Transportmittel stark vertreten, weshalb sich die Frage stellt, warum der *Segway* nicht erfolgreich war. Einer der Gründe dafür ist, dass das Produkt nicht auf die konkreten Ziele eines Benutzers ausgelegt wurde.

> *Segway failed because it did not focus on any one application, and develop that market as it enhanced and improved the product. Selling 100 Segways to 20 different uses was an inherently bad decision. What Segway needed to do was sell 100 units to a single, or at most 2, applications.*
> *(Hartung, 2015)*

Im Weiteren erfolgt eine Auseinandersetzung darüber, ob und wann der *User-centered Design*-Aspekt verkürzt oder weggelassen werden könnte. Die Verfasser kamen zum Schluss, dass es bei kleinen oder klaren Projekten sinnvoll sein kann, diesen Aspekt nicht ausführlich zu betreiben. Beispielsweise sind sie der Meinung, dass, wenn ein Student eine Software für Studenten entwickelt, ein langer und aufwändiger Design-Prozess zu viel Zeit in Anspruch nehmen könnte. Der *User-centered Design*-Aspekt sollte entsprechend der Grösse oder der Dauer des Projekts und dem Vorwissen des Studenten verkürzt oder weglassen werden. Dennoch sollte dies eine bewusste Entscheidung sein, welche vorgängig evaluiert wird. Das *User-centered Design* zu ignorieren, sollte nicht gängige Praxis sein.

Aus der Literaturrecherche generierten wir Wissen zum *Contextual Inquiry* und zu Personas. Beispielsweise hatten wir das *Master-Apprentice-Model* zuvor nicht gekannt. Hinsichtlich der Personas war uns der Unterschied zwischen Aufgaben und Zielen nicht klar gewesen und wir wissen nun, dass die Erreichung der Ziele das ist, was den Nutzer zufrieden stellt. Basierend auf der Persona-Herleitung von Cooper, haben wir eine eigene Persona-Herleitung erschaffen. Diese wurde für diese Arbeit optimiert und angewendet. Wir konnten etwas über relevante Normen für die Arbeit lernen, und zwar dass neben Farben immer weitere Mittel, wie Text oder Bilder, verwendet werden sollten, um einen Benutzer auf etwas hinzuweisen. *Usability-Tests* waren uns nicht fremd, dennoch konnten wir auch in diesem Bereich Neues lernen, z. B. hinsichtlich des Zusammenhangs des Tests zu einer Persona.

3. Umsetzung von User-centered Design

In diesem Kapitel wird auf die Aktionen und Massnahmen des *User-centered Design*-Aspekts im Rahmen dieser Arbeit eingegangen. Das Ziel ist es, die Applikation auf die Ansprüche und die Ziele des Nutzers angepasst zu entwickeln. Dafür wurde die Theorie aus der Literaturrecherche in der Praxis angewendet. Mit einem *Contextual Inquiry* sollten hinsichtlich der Sicht des Nutzers Daten ermittelt werden. Daraus wurden dann die Personas und erste *Wireframes* entwickelt. Während der Design- und Entwicklungsphase wurde das neue Wissen bezüglich der *Dashboard*-spezifischen Literatur eingebracht und so wurden die *Usability* und die *User-Experience* verbessert. Am Ende wurden *Usability-Tests* durchgeführt, um zu verifizieren, ob das Design den Anforderungen entspricht.

3.1 Contextual Inquiry bei der SFS Group AG

Im Rahmen der Anforderungsanalyse wurde ein *Contextual Inquiry* am 04.10.2022 bei der SFS in Heerbrugg durchgeführt. An diesem Tag wurden die zukünftigen Endnutzer der Applikation beim Arbeiten beobachtet und befragt.

3.1.1 Vorgehen

Anhand der Aufgabenstellung und des ersten Remoteaustauschs mit der SFS wurden die *Contextual Inquirys* vorbereitet für das Lager und die Produktion. Dafür wurden Unklarheiten in Fragen umgewandelt, beispielsweise ‹Was für Zustände hat eine Bestellung?›, ‹Ob und wie können Bestellungen abgebrochen werden?› und ‹Wie werden Bestellungen priorisiert?›. Die Fragen wurden in abteilungsspezifische Kategorien unterteilt, um bedeutsame Themen oder Unklarheiten während des Besuchs nicht zu vergessen. Beispielsweise wurde von jeder Abteilung in Erfahrung gebracht, was die jeweiligen *Pain-Points* beim aktuellen Prozess sind, während die Frage, wozu ein *Scanner*, welcher im Remoteaustausch erwähnt wurde, benötigt wird, nur Mitarbeitern im Lager gestellt wurde.

3.1.2 Durchführung

Für die Nutzeranalyse wurden zwei Abteilungen der Firma SFS beim Arbeitsablauf beobachtet. Dabei handelte es sich um das Drahtlager und um die Produktion, in welcher der Draht weiterverarbeitet wird.

In der Produktion wurde den Verfassern gezeigt, wie mit den Maschinen gearbeitet wird, wie Bestellungen ans Drahtlager geschickt werden, was ein Bestellschein enthält, wie die existierenden Umsysteme funktionieren und wie diese in Zukunft funktionieren könnten. Die verbleibenden Fragen zur Produktion konnten gegen Ende mit einem Mitarbeiter und Adrian Scherrer geklärt werden.

Danach durften die Verfasser im Drahtlager am Prozess einer echten Bestellung teilnehmen. Angelehnt an das *Master-Apprentice-Model* wurde ihnen wie einem Auszubildenden gezeigt und erklärt, welche Schritte durchgeführt werden, worüber sie sich Gedanken machen sollten und was relevant für sie ist. Während des *Contextual Inquirys* war es den Verfassern möglich, Fragen und Unklarheiten sofort zu klären, was zu einer besseren Nachvollziehbarkeit geführt hat. Für eine Diskussion bezüglich des Lagers standen den Verfassern zwei Mitarbeiter zur Verfügung. Als Beispiel wurde in der Diskussion eine Schwäche des digitalen Bestellprozesses thematisiert. Aktuell sind nämlich Bestellungen, welche für die Auslieferung bereitstehen, mit dem Bestellformular identifizierbar, welches angehängt wird. In Zukunft ist dies nicht mehr der Fall, da sich die Informationen auf einem Tablet befinden. In der Diskussion wurde daher eine gemeinsame Lösung erarbeitet, um dieser Schwäche entgegenzuwirken: Der Lagermitarbeiter soll dem Staplerfahrer als Hilfestellung die Anzahl der *Coils* mitteilen. Dadurch können z. T. ähnliche Bestellungen besser erkannt werden.
Zusätzlich wurden mit Adrian Scherrer und seinem Team technische Diskussionen geführt und letzte offene

Fragen geklärt. Als Beispiel forderten die Verfasser konkrete Angaben zu den Endgeräten, auf welchen die Software bedient wird. Daraufhin wurde ihnen ein Tablet als Testgerät zur Verfügung gestellt und sie erhielten Angaben zu anderen Geräten bezüglich des Modells und der Dokumentation.

3.1.3 Ablauf einer Bestellung

Der Fokus des Besuchs lag auf dem Lager, da dort das Produkt der Bachelorarbeit am stärksten eingesetzt wird. Die Verfasser erhielten Informationen zu diesem Bereich und der Ablauf einer Bestellung wurde anhand eines realen Vorgangs demonstriert.

Das Lager ist in drei Zuständigkeitsbereiche aufgeteilt. Ein Mitarbeiter ist immer zwei Wochen für einen Bereich verantwortlich und wechselt dann zum nächsten. Nach sechs Wochen ist er dann zwei Wochen als Staplerfahrer unterwegs und liefert die im Lager vorbereiteten Bestellungen aus. Somit wiederholt sich dieser Zyklus alle acht Wochen.

Generell ist wesentlich, dass bei einer Bestellung immer nur Draht bestellt wird. Die Menge wird anhand des Gewichts spezifiziert.

Der gesamte Ablauf einer Bestellung von der Produktion bis hin zum Lager wird in *Abbildung 7* auf Seite 22 demonstriert.

Im Folgenden wird der Prozess einer Bestellung innerhalb des Lagers in Textform beschrieben:

1. Ein Bestellformular wird von einem Staplerfahrer vor dem Lager in einem Fach deponiert.
2. Ein Lagermitarbeiter entnimmt die Formulare und verteilt diese in die jeweiligen Fächer der Standorte (EG, UG, Lager 2).
3. Der zuständige Lagermitarbeiter des Standorts entnimmt die Zettel und geht damit in seinen Lagerbereich.
4. Er sucht anhand des Drahttyps auf dem Lageplan den Standort des Drahtes und begibt sich mit einem Stapler dorthin.
5. Er entnimmt das Material und führt dabei folgende Teilschritte durch:
    a. Er überprüft, ob es sich dabei um den richtigen Draht handelt.
    b. Im Falle eines Wareneingangendes, Chargenendes oder Drahtendes erfasst er den Draht in einer weiteren Liste.
    c. Er entnimmt die entsprechende Anzahl an *Coils* und hängt einem davon das Bestellformular an.
6. Er stellt die *Coils* bei der Abholstelle bereit.
7. Der Staplerfahrer holt eine Bestellung ab und liefert diese zur auf dem Bestellformular beschriebenen Maschine.

Durch die Begleitung des ganzen Prozesses von der Bestellung bis hin zur Auslieferung war es den Verfassern möglich, einen Einblick in den Alltag der zukünftigen Benutzer der Applikation zu erhalten. Für den *User-centered Design*-Prozess war es von Bedeutung, die Gedanken und Hintergründe der Arbeit zu verstehen (*Goals*), damit diese bei den Anforderungen und im Design berücksichtigt werden können.

Auf Basis dieser Nutzeranalyse und des *Contextual Inquirys* wurden Personas und die Anforderungsanalyse erstellt, welche im Kapitel *3.2 Personas* und im Anhang *A.1 User Stories* dokumentiert sind.

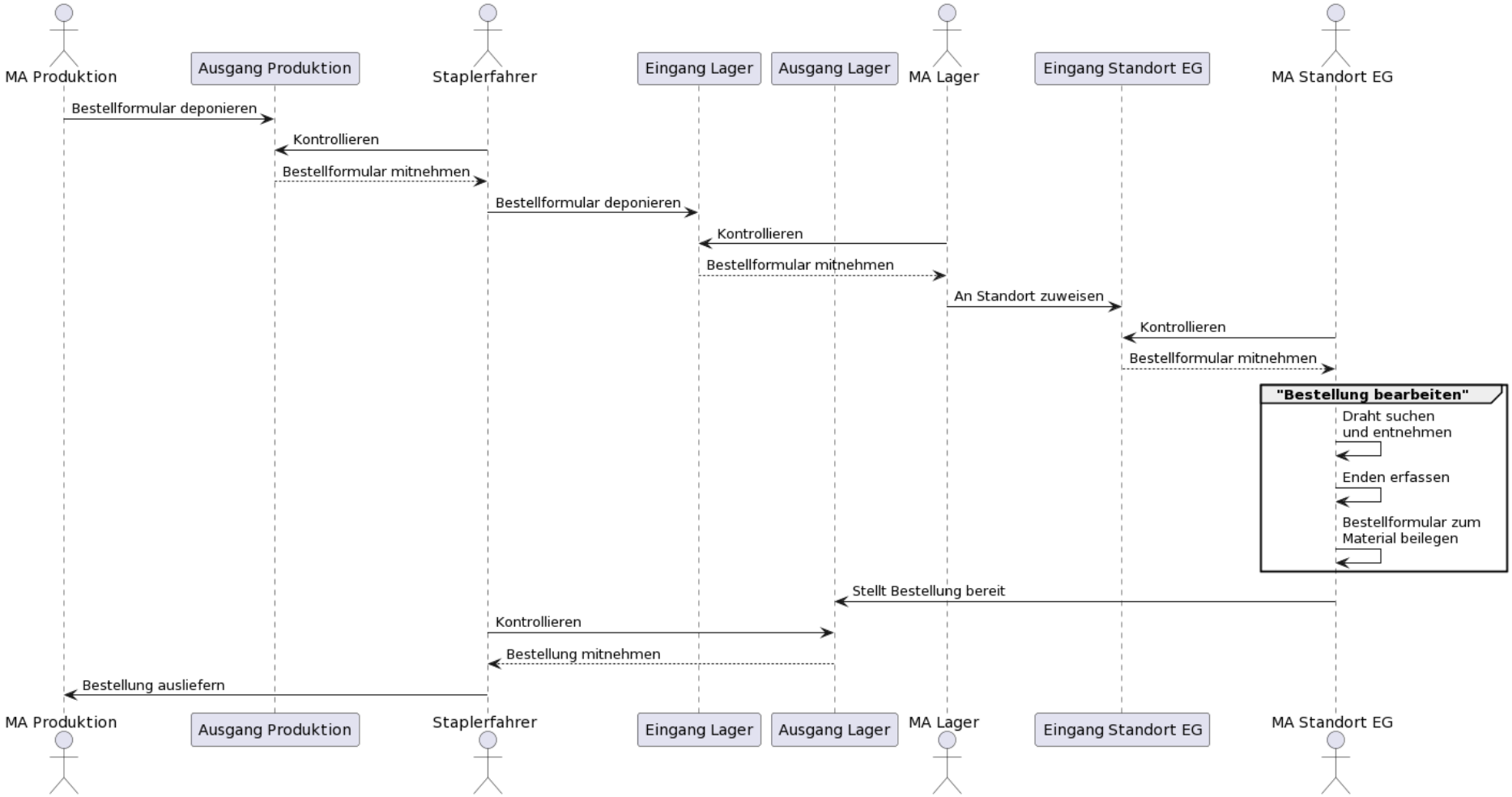


Abbildung 7: Ablauf einer Bestellung (Ist-Zustand)
Quelle: Erstellt mit PlantUML

3.2 Personas

Anhand der Theorie aus der Literaturrecherche und der Daten aus der Nutzeranalyse aus Kapitel *3.1 Contextual Inquiry* wurden drei Personas erarbeitet, welche die identifizierten Nutzergruppen und deren Anforderungen repräsentieren. Es handelt sich hierbei um fiktive Personen und Bilder.

3.2.1 Hans Graf (Produktionsmitarbeiter)

Ich möchte einfach und schnell Bestellungen tätigen, ohne meinen Arbeitsplatz zu verlassen.

Was motiviert mich, die Anwendung zu nutzen?

- Bestellungen können direkt von der Maschine aus getätigt werden.
- Bei einer falschen Bestellung kann diese noch früh genug storniert oder korrigiert werden.
- Bestellungen werden schneller ausgeliefert.
- Es besteht ein besserer Überblick über alle getätigten Bestellungen und deren Stand.

Ängste und Befürchtungen

- Es werden falsche Bestellungen oder solche mit falschen Angaben getätigt.

Anwendungsfälle

- Erfassen von neuen Bestellungen
- Bearbeitung oder Stornierung von bestehenden Bestellungen
- Expressbestellung auslösen
- Bestellungsstatus einsehen

Abbildung 8: Persona Hans Graf

Quelle: Erstellt mit https://this-person-does-not-exist.com

3.2.2 Noah Huber (Lagermitarbeiter)

Ich möchte neu getätigte Bestellungen schnell abarbeiten.	 *Abbildung 9: Persona Noah Huber* *Quelle: Erstellt mit https://this-person-does-not-exist.com*
Was motiviert mich, die Anwendung zu nutzen? • Es besteht eine verständliche grafische Übersicht über alle oder einzelne Bestellungen. • Das Aushelfen in anderen Lagern wird ermöglicht, wenn die eigenen Bestellungen abgearbeitet sind. • Bei keinen offenen Bestellungen kann der Staplerfahrer unterstützt werden. • Es besteht eine Zeitersparnis durch den digitalen Prozess.	
Ängste und Befürchtungen • Die Applikation ist schwer zu bedienen.	
Anwendungsfälle • Einsehen aller offenen Bestellungen • Annahme und Abarbeitung offener Bestellungen • Markieren der Bestellung als abholbereit	

3.2.3 Nico Meier (Staplerfahrer)

Ich möchte Bestellungen kontrollieren und wissen, wohin diese müssen.	 *Abbildung 10: Persona Nico Meier* *Quelle: Erstellt mit https://this-person-does-not-exist.com*
Was motiviert mich, die Anwendung zu nutzen? • Der Lageplan ermöglicht ein einfaches Finden der Maschinen.	
Ängste und Befürchtungen • Bestellungen werden nicht mehr direkt angeschrieben (da papierlos).	
Anwendungsfälle • Durchführung der letzten Kontrolle der Bestellung • Einsehen aller abholbereiten Bestellungen und deren Details • Einsehen des Zustellortes	

3.3 Wireframes

3.3.1 Vorgehen

Um mit den Benutzern nochmals Rücksprache über das Design zu halten, wurden *Wireframes* entwickelt. Zuvor wurden *Moodboards* erstellt, die in diesem Rahmen als Diskussionsgrundlage und zur Ideensammlung verwendet wurden. (Wikipedia, 2021b) Mit ihrer Hilfe wurde thematisiert, wie andere *Dashboards* aussehen, welche Informationen diese anzeigen und welche Farbpaletten verwendet werden. Die erstellten *Moodboard*s für verschiedene Ansichten befinden sich im Anhang *E Moodboards.*

Auf Basis der Literaturrecherche (Kapitel *2.4 Entwurfsprinzipien für* Dashboards) erfolgten Überlegungen zur Farbwahl auf dem *Dashboard*. Die Entscheidung fiel dabei auf die Farbpalette in *Abbildung 11*, welche von *Power BI*[1] verwendet wird.

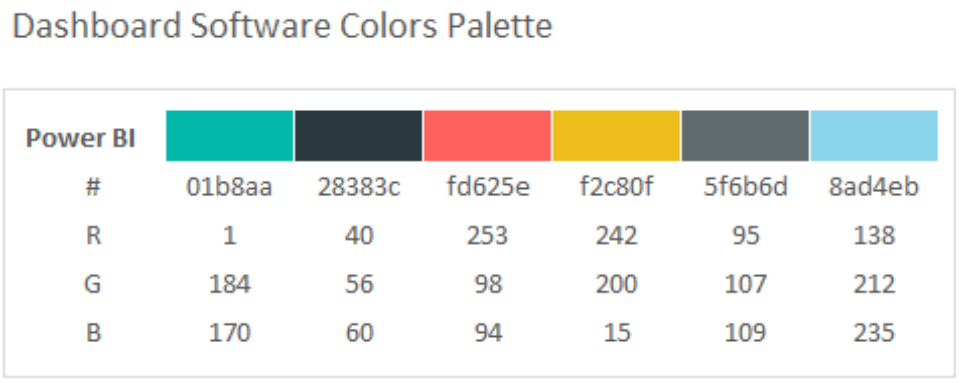


Abbildung 11: Farbpalette von Power BI
Quelle: https://adniasolutions.com/dashboard-design-principles/colors-palettes-for-dashboards/color-palette/

Diese weist eine gute Farbenblindheitsabdeckung (siehe *Abbildung 12*) auf und wurde in den Besprechungen mit der SFS und Frieder Loch als gut und passend empfunden. Obwohl in der Literatur darauf hingewiesen wurde, am Rand eines Bildschirmes kein Rot zu verwenden, wurde entschieden, dort ein rotes Element auszuführen (siehe *Abbildung 13*), weil dadurch der Arbeitsablauf der SFS besser dargestellt wird. Um dem roten Element gegenzusteuern, wurde zwischen diesem und dem Rand ein leerer Raum gelassen, sodass das Objekt nicht direkt am Bildschirmrand haftet.

[1] https://powerbi.microsoft.com/de-de/

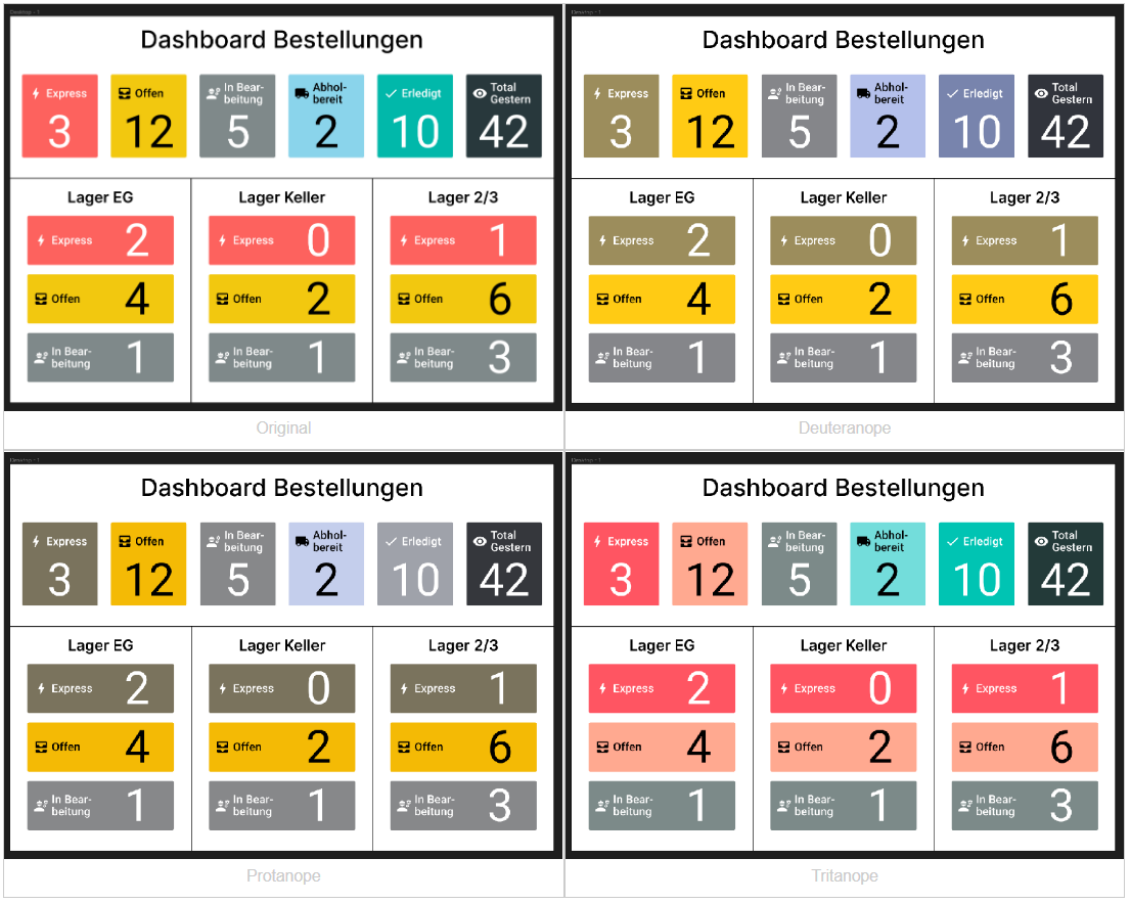


Abbildung 12: Dashboard für verschiedene Farbenblindheiten
Quelle: Generiert mit http://hclwizard.org:3000/cvdemulator/

Aus den Personas, den Anforderungen und den technischen Einschränkungen aus den Kapiteln *3.2 Personas* und dem Anhang *A Software-Dokumentation* wurden drei Ansichten für die *Wireframes* hergeleitet. Eine davon ist die *Dashboard*-Ansicht (siehe *Abbildung 13*), welche im Lager an einem Monitor an der Wand angezeigt wird. Sie soll einen Überblick über die aktuellen Bestellungen und deren Status verschaffen.

Der erste Entwurf des Designs entstand auf Basis des *Moodboards*. Dieser Entwurf wurde schliesslich mit Frieder Loch und Marius Orfgen besprochen und zu diesem aktuellen Wireframe aufgearbeitet.

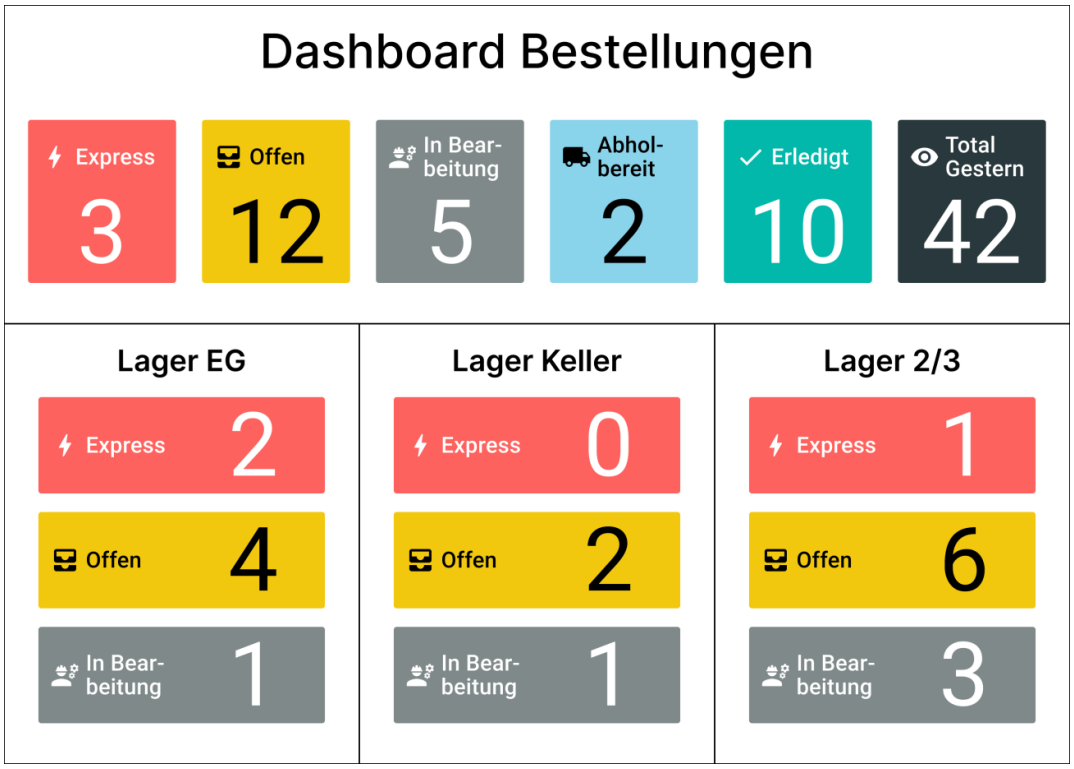


Abbildung 13: Wireframe der Dashboard-Ansicht
Quelle: Erstellt mit Figma

Die zweite Ansicht (siehe *Abbildung 14*) wird für den Staplerfahrer eine für ein Tablet optimierte Ansicht benötigt. Darauf soll dieser nochmals die wesentlichsten Informationen sehen, um die Bestellung zu identifizieren, zu kontrollieren und schliesslich auszuliefern. Zusätzlich soll ein Lageplan als Hilfestellung für neue Mitarbeiter zur Verfügung stehen, welche sich bei den vielen internen Standorten noch nicht zurechtfinden.

Das Design basiert auf der *Master-Detail-View.* Hierzu gibt es auch ein Beispiel auf dem *Moodboards.*

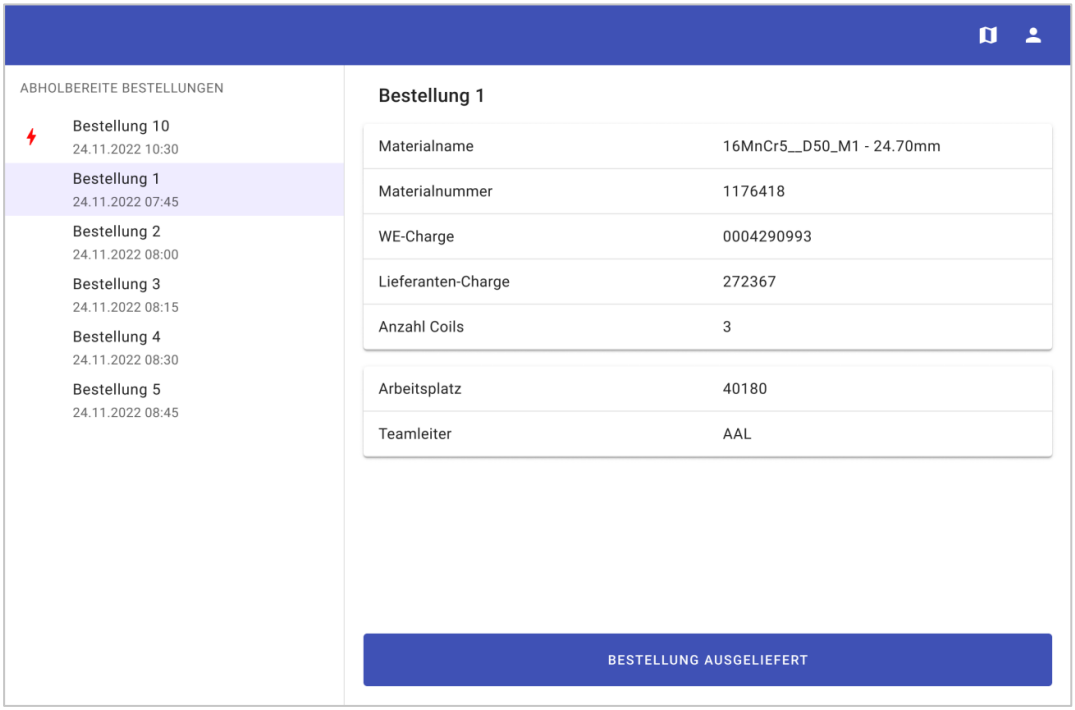


Abbildung 14: Wireframe der Staplerfahrer-Ansicht
Quelle: Erstellt mit Figma

Als dritte und letzte Ansicht (siehe *Abbildung 15*) wird primär über ein Smartphone der SFS bedient. Darin wird der Bestellprozess für den Lagermitarbeiter abgebildet, d. h. der Ablauf von der Annahme der Bestellung über deren Verarbeitung bis zu deren Bereitstellung für den Staplerfahrer.

Dieses Design basiert ebenfalls auf der *Master-Detail-View* für eine Mobile-Ansicht. Hierzu gibt es auch ein Beispiel auf dem *Moodboards.*

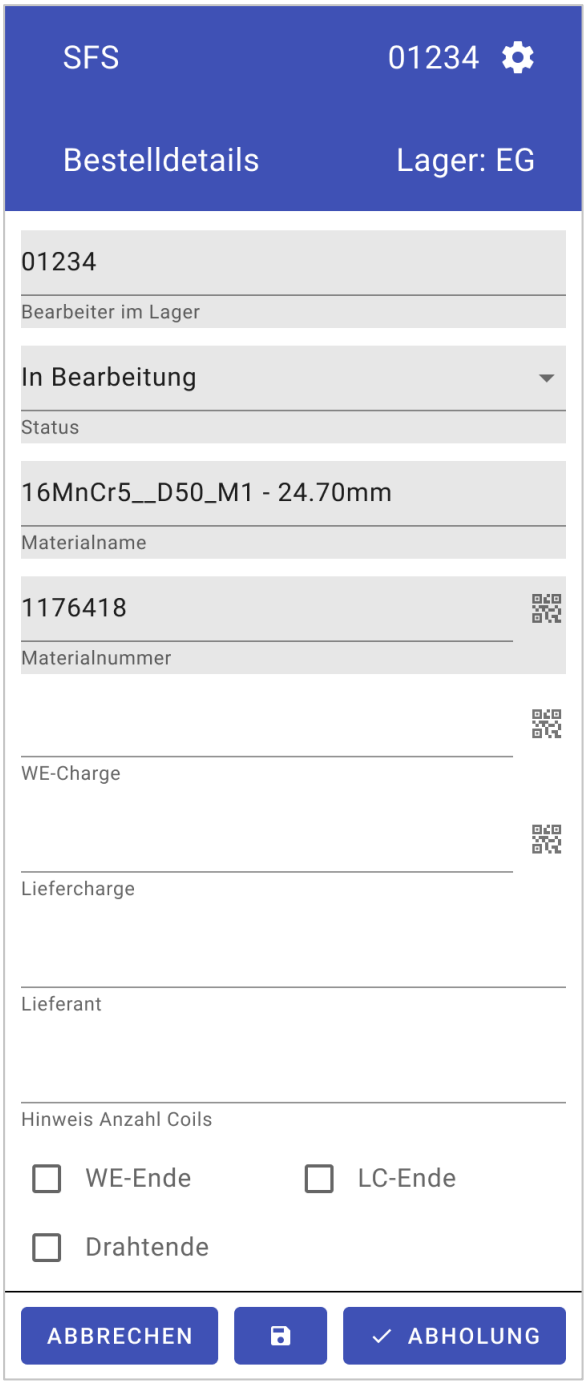


Abbildung 15: Wireframe der Lagermitarbeiter-Ansicht
Quelle: Erstellt mit Figma

Die komplette *Wireframe*-Sammlung der drei Ansichten befindet sich im Anhang *F Wireframes*.

In einem Workshop wurden die erstellten *Wireframes* zusammen mit den zukünftigen Nutzern der SFS diskutiert. Jede Ansicht wurde dabei zuerst von den Verfassern erläutert und es wurde dargestellt, welche Gedanken sich diese gemacht haben. Daraufhin wurden die offenen Fragen der Verfasser geklärt. Danach erfolgte pro Ansicht eine offene Diskussion für Inputs der SFS. (siehe Kapitel *3.3.2 Feedback der SFS Group AG zu den Wireframes*).

3.3.2 Feedback der SFS Group AG zu den Wireframes

Die gesammelten Erkenntnisse aus dem Design-Walkthrough mit der SFS werden in der folgenden Tabelle aufgeführt. Sie wurden direkt in der Umsetzung berücksichtigt.

Ansicht	Anpassung
Dashboard	Das Lastwagen-Icon sollte gegen ein Staplerfahrer-Icon getauscht werden.
Lageransicht	Es sollte möglich sein, mehrere Lager-Zuständigkeitsbereiche anzugeben.
	In den Einstellungen sollten zusätzlich zur Personalnummer der Vor- und der Nachname konfigurierbar sein, damit andere Benutzer eine Person einfacher identifizieren können.
	Die Materialnummer und die Wareneingang-Charge sollten immer gescannt werden. Auch wenn diese bereits vorgegeben sind, sollten sie hierdurch verifiziert werden.
	Das Statusfeld sollte kein Auswahlfeld sein, da dieses immer schreibgeschützt ist.
	Der Lagerstandort des Materials fehlt auf den *Wireframes*. Er sollte auf der Listen- sowie der Detailansicht angezeigt werden.
	Auf der Listenansicht sollten die Materialnummer, der Lagerstandort und die Eingangszeit der Bestellung angezeigt werden.
Stapleransicht	Der Staplerfahrer sollte sich nicht mit der Personalnummer identifizieren, sondern mit der Bezeichnung des Staplers, mit welchem er die Bestellungen ausliefert.
	Auf der Listenansicht sollte die Materialnummer der Bestellung angezeigt werden.
	Es sollte ein Suchfeld geben, in welchem Bestellungen anhand der Materialnummer und der Wareneingang-Charge gesucht werden können. Dadurch können sie auch via Barcodescan gefunden werden.

3.3.3 Beschlüsse zu Wireframes

Im Folgenden sind die Entscheidungen und die Begründungen für die Beschlüsse bezüglich den *Wireframes* festgehalten.

3.3.3.1 Design-Walkthroughs mit Wireframes

Die Wireframes wurden, wie im Kapitel *3.3.2 Feedback der SFS Group AG zu den Wireframes* beschrieben, in einem *Design-Walkthrough*, mit der SFS besprochen. Bei diesem konnten die Endnutzer Feedback und Inputs zu den *Wireframes* geben. Die *Usability-Tests* erfolgten stattdessen anhand des entwickelten Softwareprototyps gemeinsam mit der SFS (siehe Kapitel *3.4 Usability-Tests*).

3.3.3.2 Entscheid zur Anzeige der Bestellungen auf dem Dashboard

Bei einer ersten Feedbackrunde zu den Anforderungen aus der Nutzeranalyse (siehe Kapitel *3.1 Contextual Inquiry*) ergab sich eine ergänzende Anforderung seitens des Projektpartners, wonach auf dem *Dashboard* auch einzelne Bestellungen aufgelistet werden sollten. Auch im Vorfeld war darüber nachgedacht, jedoch entschieden worden, auf dem *Dashboard* keine konkreten Bestellungen anzuzeigen. Die Hintergründe dazu waren:

- Die Überschaubarkeit des *Dashboards* würde beeinträchtigt. Ein solches soll schnell einen Überblick verschaffen. (Wikipedia, 2021a)
- Aus der Nutzeranalyse liess sich herleiten, dass Bestellungen kurzlebig sind und daher mit ihrer einzelnen Darstellung kein Mehrwert entsteht.

Dennoch wurde mit dem Projektpartner vereinbart, dass die Notwendigkeit dieser Anforderung in *Usability-Tests* verifiziert wird. Mit diesen sollte überprüft werden, ob und wie viele Benutzer diese Informationen auf dem *Dashboard* benötigen.

3.3.3.3 Entscheid zur Identifizierung von Mitarbeitern in der Lagermitarbeiter-Ansicht

Beim Design-Walkthrough der *Wireframes* (siehe Kapitel *3.3.2 Feedback der SFS Group AG zu den Wireframes*) gab es eine ergänzende Anforderung seitens des Projektpartners, wonach sich Lagermitarbeiter in der Lagermitarbeiter-Ansicht mit ihrer Personalnummer und ihrem Namen identifizieren sollten. Die Hintergründe dazu waren:

- So erkennen andere Mitarbeiter, wer welche Bestellung bearbeitet.

Nachträglich wurde von den Verfassern entschieden, dass das Erfassen des Namens des Mitarbeiters keinen Nutzen bringt, da Bestellungen, welche einem Mitarbeiter zugewiesen sind, anderen Mitarbeitern nicht angezeigt werden.

3.4 Usability-Tests

Um den entwickelten Prototyp im produktiven Umfeld zu testen, wurde am 06.12.2022 bei der SFS ein *Usability-Test* durchgeführt. Die Tests wurden auf dem aktuellen Stand des Prototyps durchgeführt (siehe Designs vom Kapitel *4.3 Ergebnis des Prototyps*). In diesem Kapitel werden die Vorbereitungen, der Ablauf und die Resultate beschrieben.

3.4.1 Vorgehen

Als Vorbereitung wurden vier Szenarien erstellt. Diese sollten möglichst alle *User-Story*-Funktionalitäten abdecken (siehe Anhang *A.1 User Stories*). Sie sind zum einen generalisiert geschrieben, damit der Tester selbst das gewünschte Ziel erreicht, zum anderen sind sie aber auch möglichst realitätsnah gestaltet. Da neben den *User-Storys* auch die *NFRs* des Anhangs *A.2.1 Usability* getestet werden mussten, wurden in einem zusätzlichen Fragebogen Fragen erfasst, um die Zufriedenheit der Tester zu messen. Alle Fragen und Szenarien befinden sich im Anhang *G Usability-Tests*.

In einem Büro der SFS wurde ein Lager simuliert, damit für die Tester der Arbeitskontext nicht vollständig verloren geht. Dabei durchliefen die Teilnehmer die Szenarien und Aufgaben, als ob sie real eine Bestellung bearbeiten würden. Die Verfasser forderten die Tester dazu auf, ‹laut zu denken›, damit auch ihre Gedankengänge nachvollzogen werden können und allfällige Probleme mit der Software besser erkannt werden.

Um ein realistischeres Szenario zu erhalten, wurde auf einem Testserver, welcher von der OST zur Verfügung gestellt wurde, die gesamte Applikation gehostet. Dadurch stand diese den Teilnehmern so zur Verfügung, wie es auch im produktiven Umfeld der Fall wäre. Das heisst, dass die Tests für den Lagermitarbeiter auf einem Handy und jene für den Lieferanten auf einem Tablet durchgeführt wurden. Zusätzlich wurde das *Dashboard* auf einem Laptop geöffnet. Da nicht alle Geräte, auf welchen die Applikation am Ende verwendet wird, zur Verfügung standen, wurden ein normales Handy und ein Laptop der Verfasser sowie ein Test-Tablet der SFS verwendet. Die Daten, mit welchen die *Usability-Tests* simuliert werden, basierten auf realen Daten, welche ebenfalls von der SFS zur Verfügung gestellt wurden. Sie wurden zum Zweck des Tests statisch in das *Backend* geschrieben und danach wieder gelöscht.

Die *Usability-Tests* wurden mit zwei Personen der SFS durchgeführt. Beide arbeiten für die Logistik und sind Teil des Drahtlagers und der Auslieferungen. Sie wurden separat getestet, um die Resultate nicht zu verfälschen. Eine Person hatte bis dahin die Software nicht gekannt, die andere war dagegen am Planungsprozess beteiligt. Zusätzlich wurde die Applikation dem Abteilungsleiter des Lagers und dem Leiter der Entwicklung präsentiert. Während des *Usability-Tests* wurden alle Probleme und Anmerkungen der Benutzer protokolliert. Sie sind im anschliessenden Abschnitt dokumentiert.

3.4.2 Resultate

In der nachfolgenden Tabelle sind alle Probleme und Erkenntnisse festgehalten, welche während des *Usability-Tests* auftraten oder bemerkt wurden. Dazu wurden Massnahmen erfasst: Jene, bei denen der Text in der Zelle mit ‹fix› startet, wurden im Rahmen der Arbeit umgesetzt; die anderen, bei denen am Anfang der Zelle ‹Weiterentwicklung SFS› steht, wurden im Kapitel *5.2 Weiterentwicklung* erfasst, damit sie von der SFS gelöst werden können.

Die Probleme sind in drei Kategorien aufgeteilt:

Inputs Benutzer	Probleme, welche von einer Testperson während oder nach den Tests gemeldet wurden.
Inputs aus Erkenntnissen	Erkenntnisse, welche bei der Beobachtung des Testers aufgefallen sind; hierbei handelt es sich entweder um Fehlverhalten des Benutzers oder um Aktionen, welche falsch interpretiert wurden, die der Tester aber nicht als Fehler meldete
Fehler	Fehler in der Applikation, welche während der Tests aufgetreten sind.

Nach den *Usability-Tests* entstand keine Notwendigkeit, das Design des Prototyps anzupassen. Es wurden primär Fehler gefixt und kleinere Anpassungen vorgenommen, welche das Design nicht beeinflussen. Zusätzlich bestätigte sich in den *Usability-Tests* mit den Teilnehmern, dass es nicht nötig ist, Details von Bestellungen auf dem *Dashboard* anzuzeigen (siehe Kapitel *3.3.3.2 Entscheid zur Anzeige der Bestellungen*).

Typ	Problem	Massnahme
Inputs Benutzer	Es wird nicht angezeigt, wie viele Draht-*Coils* bestellt wurden.	**fix:** Einbauen der Anzeige der Menge an bestellten Draht-*Coils*; falls es sich dabei um mehr als einen handelt, wird der Wert angezeigt, damit der Benutzer die Menge besser erkennt.
	Das Feld ‹Legierungscode› wird nicht benötigt, da es nicht mehr verwendet wird.	**fix:** Entfernen des Felds ‹Legierungscode›
	Falls mehrere Lager selektiert sind, wäre es praktisch, wenn die Bestellungen in der Übersicht nach Lager sortiert werden könnten.	**Weiterentwicklung SFS:** Einbauen der Möglichkeit, Bestellungen entweder nach Erstellungsdatum oder nach Lager zu sortieren
	Wenn eine Bestellung als abholbereit markiert wurde, kann sie nicht mehr bearbeitet werden, falls vergessen wurde, etwas einzutragen (beispielsweise die Enden).	**Weiterentwicklung SFS:** Einbauen der Möglichkeit, kürzlich als abholbereit markierte Bestellungen wieder zu bearbeiten
	Die Abkürzungen sind verwirrend, sie sollten ausgeschrieben werden (beispielsweise ‹Lieferchargenende› statt ‹LC-Ende›).	**fix:** Einbauen der Anzeige des ganzen Textes anstelle von Abkürzungen
	Die Karte der Maschinenstandorte in der Lieferanten-Ansicht ist nicht nötig, da es in jedem Stapler bereits eine laminierte Karte gibt.	Beibehaltung aufgrund des Bereichsleiter-Feedbacks

	Da Retouren am selben Ort wie abholbereite Bestellungen deponiert werden, könnte zukünftig Verwechslungsgefahr zwischen abholbereiten und retournierten Bestellungen vorliegen.	administratives Problem, welches von der SFS gelöst werden muss
	Abholbereite Expressbestellungen können nicht unkompliziert von normalen Bestellungen unterschieden werden.	administratives Problem, welches von der SFS gelöst werden muss.
	Die Maschinenkarte sollte nicht heruntergeladen werden müssen, sondern direkt in der Applikation angezeigt werden.	**fix:** Einbauen des Öffnens der Maschinenkarte in der Applikation (Dialog)
	Auf dem *Dashboard* sollte das Lastwagen-Symbol durch ein Stapler-Icon ersetzt werden.	**fix:** Ersetzen des Symbols
	Die Buttontexte sollten in der Applikation konsistent sein.	**fix:** Anpassung der Buttontexte
Inputs aus Erkenntnissen	Es ist nicht sofort klar, dass der Button ‹Speichern› dazu dient, Bestellungen zwischenzuspeichern, und nicht dazu, diese abzuschliessen.	**fix:** Verfassen eines User-Manuals
	Die Elemente im Tab ‹Meine› werden zuerst als abgearbeitete Bestellungen interpretiert.	**fix:** Verfassen eines User-Manuals
	Es ist nicht sofort klar, dass für die Bearbeitung von Bestellungen zuerst auf ‹Bearbeiten› geklickt werden muss.	**fix:** Verfassen eines User-Manuals
	Benutzern war der Prozess des Speicherns einer Bestellung nicht klar.	**fix:** Verfassen eines User-Manuals
Fehler	Es ist möglich, dass dieselbe Bestellung von zwei oder mehr verschiedenen Geräten bearbeitet wird.	**fix:** Behebung des Fehlers
	Das automatische Springen zum nächsten Feld (von der Materialnummer zur Wareneingang-Charge) nach einer korrekten Eingabe funktioniert nicht immer.	**fix:** Behebung des Fehlers

3.5 Diskussion

Um eine möglichst nutzerfreundliche Applikation zu entwickeln, wurde diese anhand des *User-centered Designs* geplant und umgesetzt. So konnte sichergestellt werden, dass der aktuelle Prozess verbessert wird.

Um den Kontext der Arbeit zu verstehen, wurden im Zuge eines *Contextual Inquirys* die Arbeitsprozesse der Abteilungen beobachtet und durchlaufen. Ein spezielles Augenmerk lag dabei auf dem Arbeitsprozess des Lagers, da dieser im Fokus der neuen Applikation ist. Er wurde daher anhand des *Master-Apprentice-Models* analysiert, was dabei half, den Prozess zu verstehen. Anhand der Resultate des *Contextual Inquirys* wurden Personas erstellt. Diese spiegeln die Bedürfnisse und die Anforderungen der betroffenen Mitarbeiter an die neue Applikation wieder.

Basierend auf dem bis dahin gesammelten Wissen wurden *Wireframes* der geplanten Ansichten erstellt. In Rücksprache mit der SFS wurde verifiziert, ob die Designentwürfe den Anforderungen der Mitarbeiter entsprechen. Die *Wireframes* wurden nicht wie gewöhnlich mittels *Usability-Tests* geprüft, sondern mit einem *Design-Walkthrough*. Stattdessen wurde dann die Applikation – die auf den *Wireframes* aufbaute – *Usability-Tests* unterzogen.

Die *Usability-Tests* dienten zum einen dazu, Fehler in der Applikation zu finden; bedeutsamer war aber, Unklarheiten und Probleme in der Anwendung zu erkennen. Neben den entdeckten Fehlern und mangelnden Funktionalitäten wurde festgestellt, dass Funktionalitäten, welche die Tester als nützlich empfanden, zu Beginn schwer zu verstehen sind. Zum Beispiel war der Prozess des Speicherns eingebaut worden, mit welchem Bestellungen zwischengespeichert werden können. Es wurde daher entschieden, die Bedienung in einer Benutzeranleitung zu dokumentieren, sodass die Anwender von Anfang an wissen, welche Funktionalitäten verfügbar sind und was diese bewirken.

4. Technische Umsetzung

4.1 Vorgehen

Als Vorbereitung für die Umsetzung wurde die Architektur laut *Abbildung 16* diskutiert und bestimmt. Die hier entwickelten *Microservices* werden in die *Microservice*-Architektur der SFS integriert. Die *Microservice*-Architektur der SFS wird in der Abbildung als Umsysteme SFS zusammengefasst.

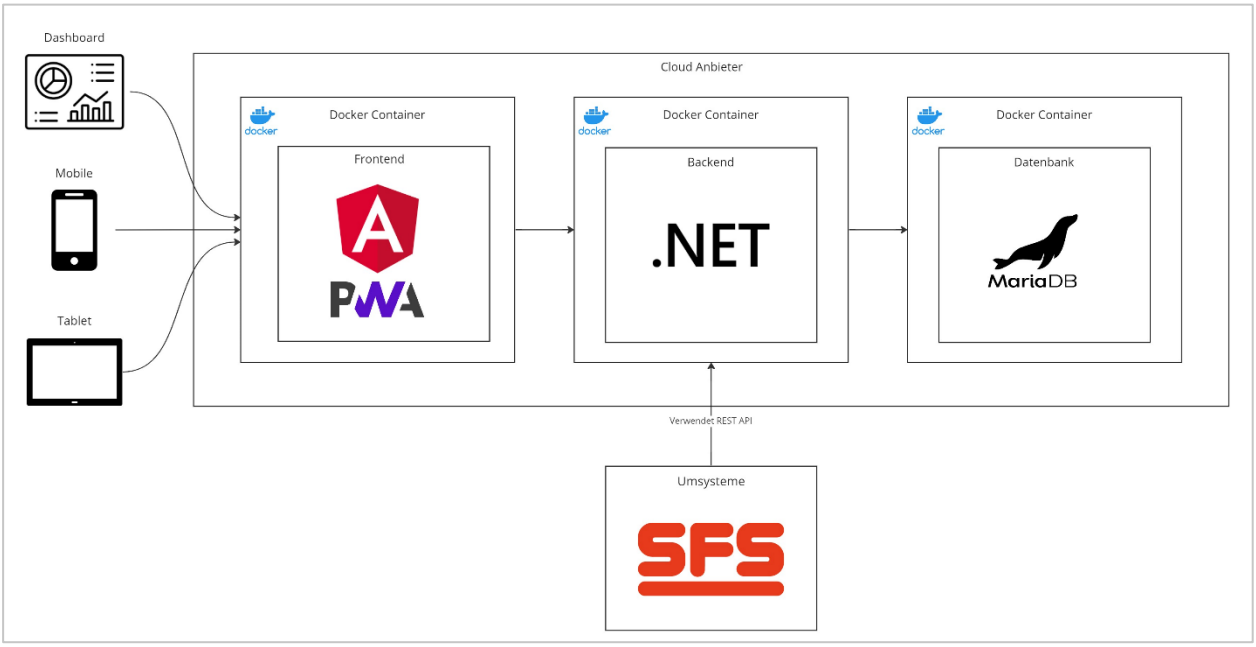


Abbildung 16: Architektur Übersicht
Quelle: Erstellt mit draw.io

In der Designphase waren erste Datenbank-Schemas entworfen worden. Aufbauend auf diesen wurden weitere Schnittstellen zwischen *Front-* und *Backend* definiert, damit die beiden *Microservices* vorerst unabhängig voneinander entwickelt werden können.

Die Verantwortlichkeiten hinsichtlich der beiden *Microservices* wurden aufgeteilt. So konnte sich Ursin Zimmermann vorerst auf die Entwicklung des *.NET 6*-Backends und Abdullah Almaz auf die Entwicklung des *Angular*-Frontends fokussieren.

4.1.1 Analyse und Entscheidung bezüglich WebSockets

In der Aufgabenstellung wurde verlangt, dass beim umgesetzten Prototyp die *Twelve Factors* berücksichtigt werden. Um frühzeitig Widersprüche gegenüber diesen zu erkennen, wurden die Anforderungen und die NFRs nochmals durchgegangen.

Aus der Analyse wurde festgestellt, dass die NFR *A.2.3,* die *Performance Efficiency*, welche eine Implementation mit *WebSockets* voraussetzt, im Widerspruch zur Nebenläufigkeit der *Twelve Factors* steht. Denn eine *WebSocket*-Verbindung stellt eine dauerhafte Verbindung zu einem Server her und erzeugt somit auch einen *State*. Daher würde ein herkömmlicher *WebSocket* im konkreten Fall die Skalierbarkeit beeinträchtigen und das *Stateless*-Prinzip verletzen. Eine mögliche Lösung für dieses Problem wäre ein

cloudbasierter *WebSocket*[2]. Da ein solcher komplex ist und die Verfasser keinen Zugriff auf die Infrastruktur der SFS haben, entschieden sie sich dazu, ein *Polling* einzuführen. Dessen Ersetzung durch einen cloudbasierten *WebSocket* wird als mögliche Weiterentwicklung im Kapitel *5.2.1 Cloudbasierte WebSockets* aufgeführt.

4.2 Realisierung des Prototyps

In den beiden folgenden Unterkapiteln wird auf die wesentlichsten technischen Komponenten des *Back-* und des *Frontends* eingegangen. Die komplette Softwaredokumentation – von den *User-Storys* über die NFRs bis hin zum Datenbankmodell und zur detaillierten Architektur – befindet sich im Anhang *A Software-Dokumentation*.

4.2.1 Detailumsetzung Frontend

Die *Angular*-Applikation ist in sieben Module unterteilt und wird in *Abbildung 17* veranschaulicht.

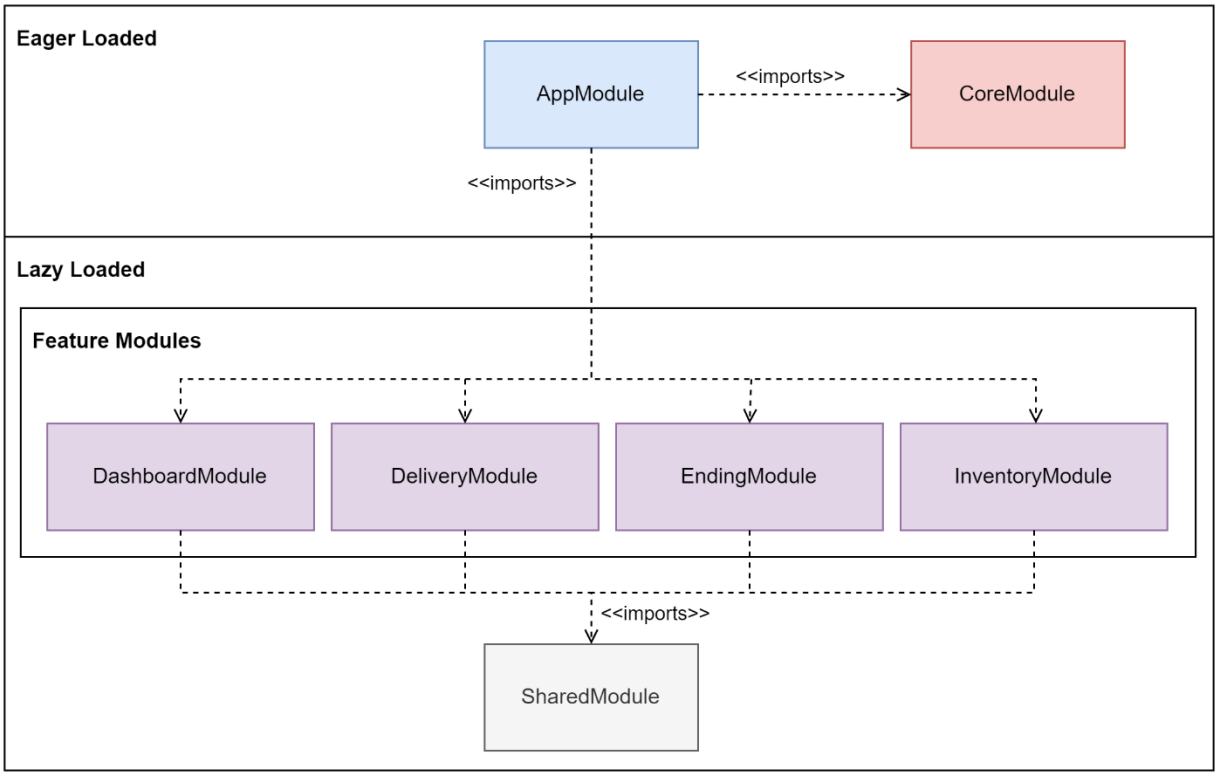


Abbildung 17: Module der Frontend-Architektur
Quelle: Erstellt mit draw.io

Das ‹App-Modul› ist das Hauptstück der Applikation und importiert alle anderen Module. Hier sind die initialen *Routes* definiert. Je nach *URL* bzw. aufgerufener *Route* wird das einzelne Feature-Modul nachgelagert geladen (*Lazy Loading*).

Das ‹Core-Modul› beinhaltet die *Services* und die Logik, welche über die ganze Applikation einmal instanziiert und zur Verfügung gestellt werden. Es ‹entschlackt› somit das App-Modul und beinhaltet die globale

[2] https://learn.microsoft.com/en-us/aspnet/core/signalr/scale?view=aspnetcore-7.0#scale-out

fachliche und technische Logik und Funktionalität. Hier befindet sich z. B. der *TranslationService*, welcher die Übersetzungen von technischen Namen wie ‹Open› in ausgeschriebene deutsche Wörter wie ‹Offen› vornimmt. Aktuell wird in Abstimmung mit der SFS nur die deutsche Sprache unterstützt.

Das ‹Shared-Modul› ist eine Sammlung von wiederverwendbaren Komponenten, *Pipes* usw. Hier werden alle genutzten *Third Party*-Module von *Angular Material* importiert und exportiert, damit diese Komplexität bei den anderen Feature-Modulen nicht erforderlich ist und eine Codeduplizierung vermieden wird. Beispielsweise befindet sich hier eine *Text-Input-Component*, welche die Darstellung und die Logik eines Texteingabefelds in einem Formular beinhaltet. Diese Komponente wird in mehreren Feature-Modulen, wo eine Texteingabe benötigt wird, wiederverwendet.

Die nächsten vier Module sind die Feature-Module. Durch die Trennung und das nachgelagerte Laden von Modulen verringern sich die Ladezeit und die Grösse der Applikation. Die vier Feature-Module ‹*Dashboard*›, ‹*Delivery*›, ‹*Inventory*› und ‹*Ending*› beinhalten die Komponenten und die *Services* für die jeweilige Nutzergruppe. Beispielweise befinden sich im ***Dashboard*-Modul** die *Services*, welche die Endpunkte des *Dashboard-Controllers* im *Backend* ansprechen, und die Komponenten, welche die Daten basierend auf den *Wireframes* auf der Benutzeroberfläche anzeigen.

Observables

Gemäss den Best Practices von *Angular* werden im umgesetzten *Frontend* hauptsächlich *Observables* verwendet. Diese basieren auf dem bekannten *Publish-subscribe-Pattern*[3]. Durch die Verwendung von *Observables* wird die *State*-Verwaltung innerhalb einer Applikation vereinfacht. Als Beispiel gibt es im *Frontend* einen *SettingsService*, welcher die aktuellen Einstellungen eines Benutzers in einem *Observable* namens ‹settings$› veröffentlicht, d. h. sobald eine Komponente die Einstellungen benötigt, um diese z. B. darzustellen, können diese im Markup wie in *Abbildung 18* gezeigt, verwendet werden.

```
<span *ngIf="settings$ | async as settings">{{settings.personNumber}}</span>
```

Abbildung 18: Beispiel eines Codes mit Verwendung von ‹Observables› und ‹Async-Pipe›
Quelle: Eigener Code

Durch diesen Code wird die Personalnummer des Benutzers ganz oben im Header angezeigt. Sollte nun der Anwender seine Einstellungen in einer anderen Komponente ändern, wird ein neuer Wert publiziert und alle *Subscriber* dieses *Observables* werden darüber notifiziert. Das heisst, dass im obigen Code durch die *Async-Pipe* ‹| async› eine *Subscription* erstellt wird und diese automatisch alle Änderungen erhält.

Proxy und Cross-Origin Resource Sharing

Damit das *Front-* mit dem *Backend* verwendet werden kann, wird ein *Proxy* vorausgesetzt. Für die Entwicklung gibt es einen ‹proxy.conf.json›, in welchem die *URL* des *Backends* hinterlegt werden muss. Bei einem produktiven *Build* wird verlangt, dass ein *Proxy* entsprechend aufgesetzt und konfiguriert wird.

Der *Proxy* wird wie folgt verwendet: Jegliche Requests startend mit dem Pfad ‹/api› werden an eine andere *URL* umgeleitet. Zum Beispiel werden in der Entwicklung alle Requests an ‹http:localhost:**4200**/api/dashboard› zu ‹http:localhost:**5039** /api/dashboard› umgeleitet.

[3] https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern

Der Umstand mit dem *Proxy* ist folgendermassen zu begründen: Erstens entfallen dadurch jegliche *CORS*-Probleme und Konfigurationen in der produktiven Anwendung und zweitens wird dadurch das *Frontend* konfigurationslos. Dies entspricht den Prinzipien der *Twelve Factors*, welche in der Aufgabenstellung vorausgesetzt werden.

4.2.2 Detailumsetzung Backend

Die Architektur des *Backends* basiert auf der *Clean Architecture*. Bei einer mit dieser umgesetzten Applikation sind die Businesslogik und das Datenmodell das Herzstück. Alle weiteren Schichten wie die Infrastruktur basieren auf diesen beiden Komponenten. Dazu werden *Interfaces* oder Abstraktionen definiert, welche dann beispielsweise von einer Infrastruktur implementiert werden. So ist es möglich, Teile von Letzterer anzupassen oder zu ersetzen, ohne dass die Grundlogik der Applikation verändert wird. Eine solche Architektur lässt sich mit einem Ringdiagramm veranschaulichen, in welchem die äusseren Ringe auf dem nächstinneren Ring aufbauen (siehe *Abbildung 19*). (Microsoft, 2022a)

Da in dieser Arbeit ein Prototyp entwickelt wurde, wurde das *Backend* mit dieser Architektur umgesetzt, damit zukünftig Teile der Applikation möglichst unkompliziert angepasst werden können. Zusätzlich ist das Testen einer solchen Anwendung ‹sauber›, da die Abhängigkeiten und die Schichten jeweils durch einen *Mock* ersetzt werden können. Zum Teil wurden aber Entscheidungen getroffen, welche nicht den Prinzipien von *Clean Architecture* entsprechen, damit die Komplexität und der Aufwand geringgehalten werden. Folgende Punkte werden nicht eingehalten:

- Es fehlen *Repository*-Klassen, welche Befehle auf der Datenbank ausführen.
- *Entitys* sind im Datenbank-Projekt definiert.
- Es wurde kein separates Infrastruktur-Projekt für *Controller*, *WebApi*-Konfigurationen, ein Setup und *Middlewares* erstellt.

Diese Entscheidungen können aber mit geringem Aufwand wieder rückgängig gemacht werden, wenn seitens der SFS Bedarf besteht. (siehe Kapitel 5.2.4 Testing der Service-Klassen im Backend für weitere Details)

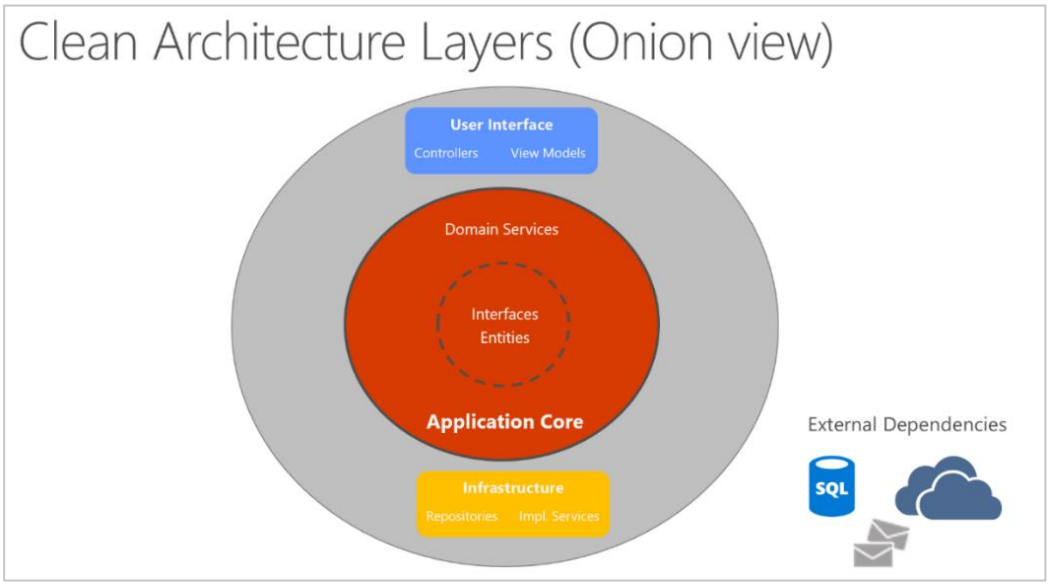


Abbildung 19: Clean Architecture Überblick
Quelle: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/media/image5-7.png

Aufbau

Um eine gute Trennung von Code und Logik innerhalb des Projekts zu gewährleisten, wurde die Applikation in vier Projekte aufgeteilt (siehe auch *Abbildung 20*):

- ‹OrderingSystemBackend.WebApi›
- ‹OrderingSystemBackend.Data›
- ‹OrderingSystemBackend.Shared›
- ‹OrderingSystemBackend.Test›

Nachfolgend wird auf diese Projekte eingegangen und die wesentlichsten Funktionalitäten werden beschrieben. Für eine bessere Lesbarkeit wird dabei der Teil ‹OrderingSystemBackend› im Namen der Projekte weggelassen.

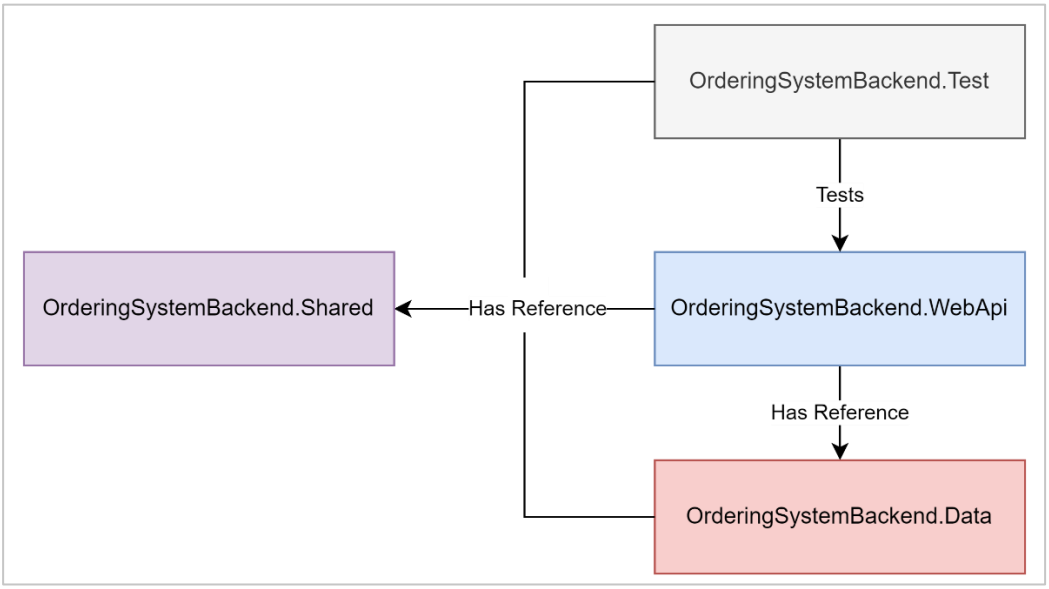


Abbildung 20: Projektstruktur Backend
Quelle: Erstellt mit draw.io

WebApi

Im *WebApi*-Projekt wurden die Schnittstelle zur Benutzeroberfläche und die Businesslogik entwickelt. Wesentlich ist hierbei, dass Letztere innerhalb des Projekts von den *Controllern* abgekapselt wurde. So kann, wie erwähnt, der Code einfacher getestet und angepasst werden. Zusätzlich wurden in diesem Projekt der Startpunkt der *WebApi* und somit auch die *Dependency-Injection* entwickelt. Es wird weiter unterschieden, ob die Applikation produktiv oder in einer Entwicklungsumgebung gestartet wird. Im letztgenannten Fall wird zusätzlich eine *Swagger-API* gestartet, um den Überblick über die gesamte *API* zu vereinfachen.

Da die Applikation den Prinzipien der *Twelve Factors* entsprechen soll, werden die Fehler der *REST*-Aufrufe in einer entwickelten *Middleware* geloggt. Diese werden dann in nicht öffentlichen Logs festgehalten, um die Fehlerfindung im produktiven Umfeld zu vereinfachen. Zusätzlich wird in der Middleware anhand des aufgetretenen Fehlers der korrekte *HTTP*-Statuscode zurückgegeben. Damit dies möglich ist, wurden für alle bekannten Fehler in der Applikation eigene *Exceptions* erstellt. Anhand dieser kann dann in der

Middleware entschieden werden, welcher *HTTP*-Statuscode zurückgegeben wird. Im Falle einer nicht definierten *Exception* wird ein 500er Fehler mit einem statischen Fehler zurückgegeben und der echte Fehler geloggt, damit keine vertraulichen Informationen das System verlassen. (siehe *Abbildung 21*)

```
context.Response.StatusCode = exception switch
{
    NoEntityWithIdException => StatusCodes.Status404NotFound,
    ModelValidationException or OrderNotEditableException => StatusCodes.Status400BadRequest,
    _ => StatusCodes.Status500InternalServerError
};
```

Abbildung 21: HTTP-Statuscode zu Exception-Mapping
Quelle: Eigener Code

Data

Das Data-Projekt enthält alle bedeutsamen Datenbank-Komponenten. Diese Vorgehensweise wurde gewählt, da die Datenbankanbindung komplett vom restlichen Code getrennt ist. Hier ist auch eine Abhängigkeit zu einem externen Framework gegeben, nämlich dem *OR-Mapper ‹EF Core›*.

EF Core bietet zum einen die Migrationsfunktionalität an, welche anhand vorhandener Klassen eine Datenbank erstellen oder migrieren kann. Dazu generiert der *OR-Mapper* aus einem im Code definierten Datenmodell und -schema einen *Snapshot*, welcher den Zustand der aktuellen Konfiguration enthält, und eine Migrations-Klasse, welche die Anpassung für die Datenbank beschreibt. Falls nun im Code Änderungen an der Datenstruktur erfolgen, kann wieder eine neue Migration generiert werden, welche dann den *Snapshot* anpasst und eine Migrationsklasse erstellt. Nun kann die Datenbank entweder via Konsolenaufruf erstellt bzw. upgedatet werden, oder dies erfolgt automatisch, wenn die Applikation ausgeführt wird. Für die produktive Anwendung kann und sollte die automatische Migration ausgeschaltet werden und sollten stattdessen über *EF Core SQL*-Scripts generiert werden, damit diese direkt in der Datenbank ausgeführt werden können.

Shared

Im Shared-Projekt befinden sich alle Klassen, welche von mehreren Projekten verwendet werden. Zum einen beinhaltet es die Ressourcen mit allen Übersetzungen und zum anderen alle selbst erstellten *Exceptions*.

Test

Im Test-Projekt befinden sich alle *Unit*- und Integrationstests des Systems. Diese verwenden *NUnit* und *Moq*. Getestet wurden die *Controller*, die *Helper* und die *Services*. Da Letztere auf *Interfaces* basieren, konnten die *Controller* ohne Datenbank getestet werden. Weil die *Services* direkt auf die Datenbank zugreifen, waren die *Unit*-Tests kompliziert. Daher wurde entschieden (siehe Kapitel *4.3.3 Entscheid bezüglich der Tests der* Services), die *Services* mit Integrationstests zu prüfen.

4.2.3 Feststellung bezüglich Patch-Requests im Backend

Bestellungen werden im Verlauf des Bearbeitungs-Workflows mehrmals angepasst, d. h. entweder werden neue Daten erfasst oder bestehende Daten geändert. Dazu werden mehrheitlich *Patch-Requests* verwendet, welche im *Frontend* abgesetzt werden und im *Backend* direkt auf den Datenobjekten ausgeführt werden. Während der Entwicklung und des Testens wurde festgestellt, dass so theoretisch Daten inkorrekt

angepasst werden könnten und es somit möglich wäre, sie in einen ungültigen Zustand zu bringen. Bei einer Analyse wurde festgestellt, dass die Anpassungen nicht direkt auf den Datenentitäten erfolgen sollten, sondern auf speziell erstellten *Model*-Klassen, in welchen nur die erlaubten Daten enthalten sind. (Piotr Zieliński, 2021)

Im Kapitel *5.2.3 Restriktivere Patch-Requests* wird darauf eingegangen, wie restriktivere *Patch-Requests* in der Zukunft umgesetzt werden können.

4.2.4 Einschränkung GitLab und Docker

Da die Applikation in Zukunft in der Cloud, auf einem *Kubernetes-Cluster,* gehostet wird, sollte sie in *Docker-Images* zur Verfügung gestellt werden. Es stellte sich jedoch heraus, dass dies mit der vorgegebenen Infrastruktur nicht möglich war. Zuerst war angedacht, die *Docker-Images* in einer *GitLab-Pipeline* zu erstellen. Da das aber ohne einen speziellen *Runner* im *GitLab* nicht möglich war, musste dies umgangen werden, indem die benötigten Dateien für ein *Docker-Image* innerhalb der *Pipeline* erstellt und mit einem *Dockerfile* ausgeliefert wurden, aber das Erstellen des *Docker-Images* manuell erfolgte.

4.3 Ergebnis des Prototyps

Im Folgenden werden Screenshots (*Abbildung 22, Abbildung 23, Abbildung 24* und *Abbildung 25*) aus Teilen der umgesetzten Applikation dargestellt, welche den *Wireframes* ähnelt und bei der die Verbesserungen aufgrund der *Usability-Tests* berücksichtigt worden sind.

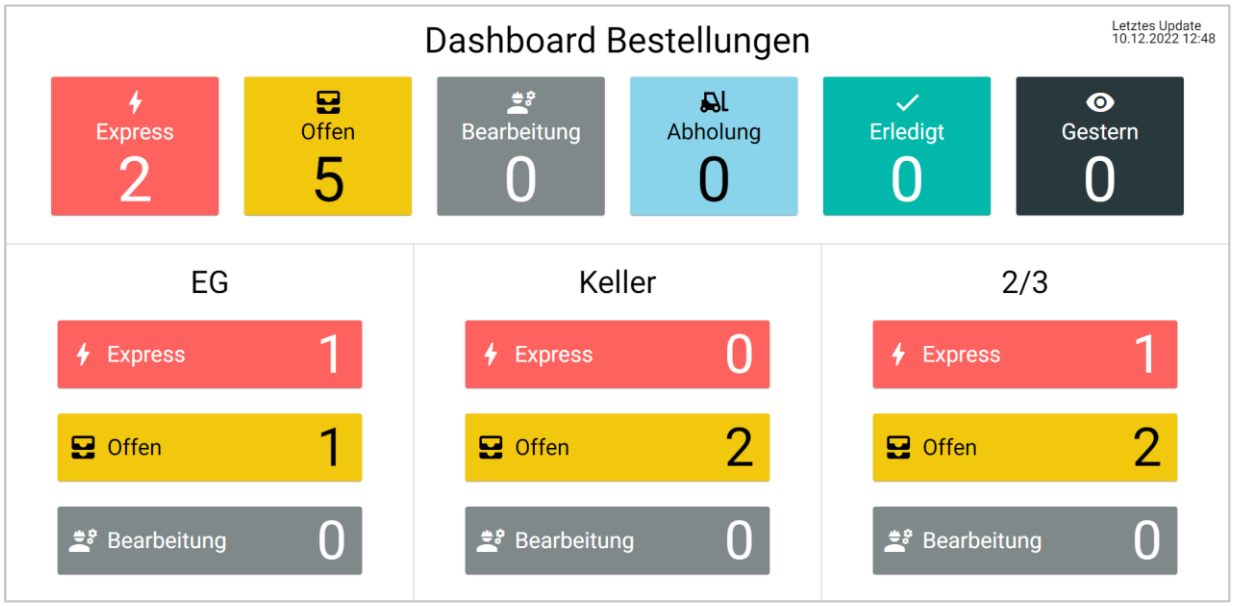


Abbildung 22: Ergebnis Dashboard-Ansicht
Quelle: Applikation

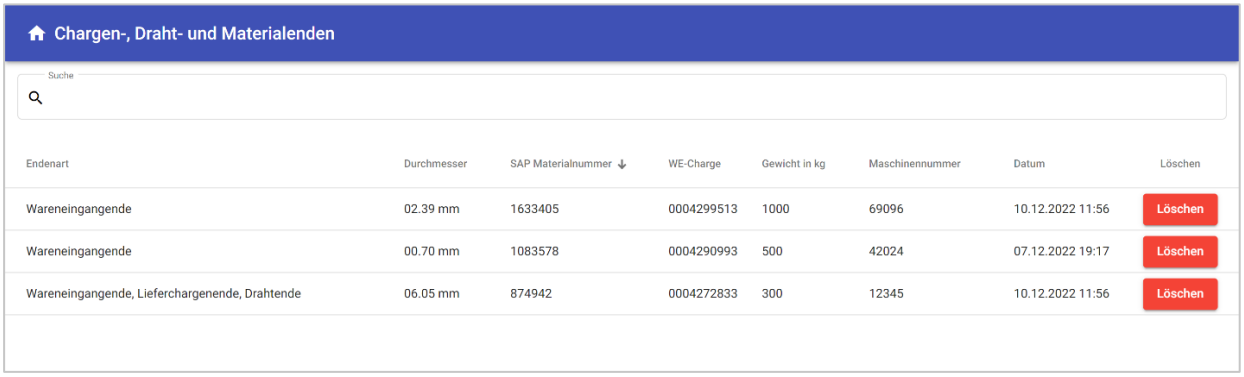


Abbildung 23: Ergebnis Enden-Ansicht
Quelle: Applikation

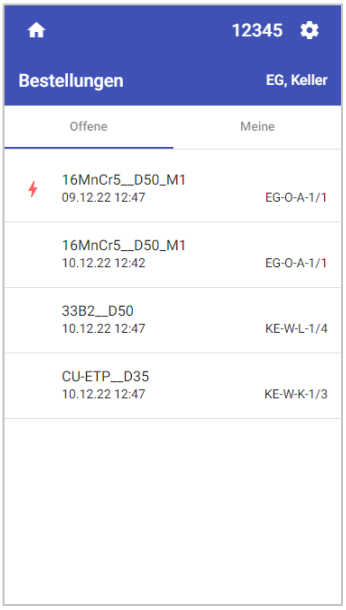


Abbildung 24: Ergebnis Lagermitarbeiter-Ansicht
Quelle: Applikation

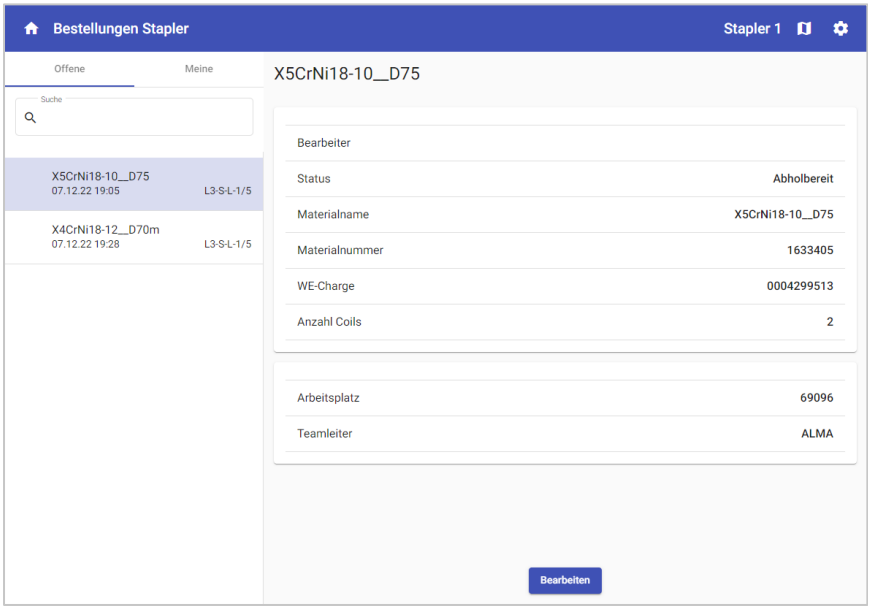


Abbildung 25: Ergebnis Staplerfahrer-Ansicht
Quelle: Applikation

4.3.1 Testabdeckung

Im *Frontend* wurde eine *Statement*-Testabdeckung von 97,47 % erreicht, mit insgesamt 61 *Unit*-Tests (siehe *Abbildung 26*). Diese prüfen und validieren auch das Zusammenspiel und die Logik mehrerer Abhängigkeiten, statt diese immer komplett zu *mocken*. Zum Beispiel wird beim Test einer Komponente, welche drei *Services* verwendet, keiner davon *gemockt*. Stattdessen wird das *HttpTestingModule*[4] verwendet, mit welchem während eines *Unit*-Tests bestimmte *HTTP-Requests* zu erwarten sind, die mit einer definierten Antwort versehen werden können. Dadurch können auch ganze Datenflüsse und die Businesslogik in den *Services* in einem Test der Komponente validiert werden.

[4] https://angular.io/api/common/http/testing/HttpClientTestingModule

```
------------------------------ Coverage summary ------------------------------
Statements   : 97.48% ( 658/675 )
Branches     : 82% ( 246/300 )
Functions    : 94.3% ( 182/193 )
Lines        : 97.44% ( 610/626 )
------------------------------------------------------------------------------

Test Suites: 26 passed, 26 total
Tests:       61 passed, 61 total
Snapshots:   0 total
Time:        71.585 s
Ran all test suites.
```

Abbildung 26: Testabdeckung Frontend
Quelle: Erstellt mit Konsole (npm)

Im *Backend* wurden die Klassen getestet, bei welchen *Unit*-Tests einen Mehrwert bringen, d. h. alle Klassen, in welchen sich die Businesslogik befindet. Generierte Klassen und Codestellen wurden dabei bewusst ausgeschlossen.
Die Businesslogik ist Teil des **WebApi**-Projekts. Daher wurden folgende Klassen getestet:

- *Controller*
- *Services*
- *Helper*

Es wurden 32 Tests geschrieben und die Testabdeckung betrug 93 % (siehe *Abbildung 27*). Bei den *Helper*- und den *Controller*-Tests handelt es sich um reine *Unit*-Tests, bei den *Service*-Tests dagegen um Integrationstests. Somit wurde zusätzlich geprüft, ob die Schnittstelle zwischen *Backend* und Datenbank korrekt funktioniert (für eine detaillierte Dokumentation siehe Kapitel *4.3.3 Entscheid bezüglich der Tests der* Services).

Symbol	Coverage (%)	Uncovered/Total Stmts.
Total	93%	16/227
OrderingSystemBackend.WebApi	93%	16/227
OrderingSystemBackend	93%	16/227
Helpers	100%	0/36
DashboardHelper	100%	0/21
OrderHelper	100%	0/15
Controllers	97%	3/91
DashboardController	100%	0/10
InventoryEndingController	100%	0/11
InventoryController	96%	1/28
OrderController	96%	1/26
DeliveryController	94%	1/16
Services	87%	13/100
InventoryEndingService	100%	0/19
OrderService	84%	13/81

Abbildung 27: Testabdeckung Backend
Quelle: Erstellt im Rider

4.3.2 Automatische Testverfahren

Im *Repository* des *Backends* und des *Frontends* wurden *Pipelines* erstellt, welche jeweils die Codebasis bei jedem *Merge-Request* testen und validieren.

Im *Frontend* werden bei jedem *Merge-Request* mehrere Jobs in der *Pipeline* gestartet. Zum einen wird vorausgesetzt, dass alle *Unit*- und Integrationstests erfolgreich verlaufen, zum anderen wird überprüft, ob alle *Linter*- und *Prettier*-Richtlinien eingehalten werden. Anschliessend wird verifiziert, ob das Projekt erfolgreich *gebuildet* werden kann.

Sobald dann eine Änderung erfolgreich in den *Develop-Branch gemerged* wurde, wird die gleiche *Pipeline* wieder ausgeführt, jedoch erzeugt der *Build-Job* im *Develop*- und im *Main-Branch* ein *Build*-Resultat, welches das ‹kompilierte› *Frontend* mit dem *Dockerfile* und dem *nginx.conf* beinhaltet. Somit kann danach ein *Release* oder ein *Prerelease* des *Frontends* erzeugt werden.

Um sicherzugehen, dass im *Backend* bei zukünftigen Codeanpassungen die Gesamtlogik weiterhin funktioniert, werden bei einem *Merge-Request* die *Unit*- und die Integrationstests im *GitLab* in einer *Pipeline* ausgeführt, welche nur erfolgreich durchläuft, wenn alle Tests erfolgreich sind. Anschliessend wird in derselben *Pipeline* geprüft, ob sich das Projekt auch *builden* lässt. Sobald ein *Branch* auf den *Main*- oder den *Develop-Branch gemerged* wurde, wird eine weitere *Pipeline* ausgeführt, in welcher nochmals die Tests erfolgen, und im Anschluss wird das Projekt veröffentlicht. Das resultierende Verzeichnis, welches während des *Publish*-Befehls erstellt wird, wird mit dem *Dockerfile* im *Build*-Resultat gespeichert. Somit kann dann mit diesen beiden Komponenten ein *Backend-Docker-Image* erstellt werden.

In den beiden *Repositorys* war es technisch nicht möglich, ein *Docker-Image* direkt in der *Pipeline* zu erzeugen, da zum einen bei *GitLab* ein spezieller *Runner* konfiguriert und zur Verfügung gestellt werden muss und zum anderen kein Zugriff auf ein *Image-Repository* der SFS besteht. Bei der Evaluation mit dem Unternehmen wurde dies angesprochen und entschieden, dass die Firma selbst auf diesen *Pipelines* aufbauen wird.

4.3.3 Entscheid bezüglich der Tests der Services

Während des Schreibens der *Unit*-Tests wurde festgestellt, dass deren Entwicklung für die *Services* der vorliegenden Architektur nicht so unkompliziert ist wie erwartet, da in den *Service*-Klassen direkt der Datenbankkontext von *EF Core* verwendet wird. Nach einer Analyse standen folgende Optionen zur Auswahl, um dieses Problem zu lösen:

- Erstellen einer *In-Memory*-Testdatenbank
- Verwenden einer *SQLite*-Testdatenbank
- *Mocking* oder *Stubbing* des Datenbankkontexts und der Daten
- Auslagern der Datenbankzugriffe in *Repository*-Klassen

Die ersten beiden Varianten sind ähnlich und entsprechen eher Integrations- als *Unit*-Tests. Da es sich bei der Applikation um einen Prototyp handelt und die Komplexität der Architektur nicht in einem *Over-Engineering* resultiert, fiel die Entscheidung auf eine *In-Memory*-Testdatenbank. Zusätzlich wurde mit Integrationstests sichergestellt, dass die Schnittstelle zwischen *Service* und Datenbank funktioniert. (Microsoft, 2022c)
Im Kapitel *5.2.4 Testing der Service-Klassen im Backend* wird darauf eingegangen, wie diese Tests weiter aufgeteilt und spezifischer durchgeführt werden könnten.

4.4 Diskussion

Während der Entwicklung des Bestellsystems traten Herausforderungen (siehe Kapitel *4.2.3 Feststellung bezüglich Patch-Requests im Backend* und Kapitel *4.2.4 Einschränkung GitLab und Docker*) auf. Diese konnten vorwiegend bewältigt werden oder wurden für die Weiterentwicklung durch die SFS dokumentiert. Da es sich bei der Applikation um einen Prototyp handelt, wurde darauf geachtet, dass nicht zu viel Zeit für Details verloren ging. So konnte bis zur Abgabe der Arbeit eine Applikation geschaffen werden, welche eine gute Grundlage für die Weiterentwicklung bietet. Um sicherzustellen, dass zukünftige Anpassungen im Code die wesentlichen Funktionalitäten nicht unbeabsichtigterweise ändern oder entfernen, wurden Komponenten mit *Unit*- oder Integrationstests versehen.

Für die zukünftigen Entwickler und Betreiber der Applikation wurde die Applikation so gestaltet, dass sie unkompliziert erweitert werden kann oder Teile davon ausgetauscht werden können. Es ist, wie erwähnt, auch möglich, die Applikation in *Docker-Containern* zu *deployen* und diese in einem produktiven Umfeld zu betreiben.

5. Resultate und Ausblick

In den folgenden Kapiteln werden die Resultate und der Ausblick dieser Arbeit ausgeführt.

5.1 Resultate hinsichtlich der funktionalen und der nichtfunktionalen Anforderungen

Im Rahmen der Arbeit und in der Entwicklungsphase konnten alle Akzeptanzkriterien der *User-Storys*, welche im Anhang *A.1 User Stories* definiert wurden, umgesetzt, getestet und verifiziert werden.

In den *Usability-Tests* wurden alle Funktionalitäten und Benutzeroberflächen von den Endanwendern getestet und entweder von diesen abgenommen oder zur Weiterentwicklung im Kapitel *5.2 Weiterentwicklung* definiert.

Im Folgenden wird die Einhaltung der NFAs aus dem Anhang *A.2 Nicht-funktionale Anforderungen (NFR) & Qualitätsattribute* ausgeführt.

Nummer	Akzeptanzkriterien	Tests	Erreichung/Verifikation
NFR1	Ein Nutzer soll sich in der Benutzeroberfläche selbständig zurechtfinden und die Bedienung erlernen können.	Es sollen *Usability-Tests* mit zwei Benutzern durchgeführt werden, welche selbständig den Workflow einer Bestellung in der Applikation durchspielen.	Im Rahmen des *Usability-Tests* wurde die Applikation von zwei Mitarbeitern des Lagers getestet. Diese erhielten keine bis wenige Instruktionen und konnten den ganzen Workflow einer Bestellung durchspielen.
NFR2	Die Applikation soll den Arbeitsalltag sowie den Bestellprozess erleichtern.	Beim *Usability-Test* soll mittels einer Frage, ob das Kriterium erfüllt ist, eine Zustimmung von mindestens 70% erreicht werden.	Im Rahmen der *Usability-Tests* wurden die Teilnehmer aufgefordert, Feedback zur Erfüllung der Anforderung zu geben. Da noch einige administrativen Probleme, die durch die Ablösung des Papierprozesses entstehen, zu klären sind erhielten wir im Durchschnitt eine Bewertung von 9 auf einer Skala von 1 bis 10.
NFR3	Die erstellten Softwareartefakte können ohne Umstände in die weitere Infrastruktur von SFS integriert werden.	Es sollen eine Codeanalyse und eine Umfrage mit zwei technischen Ansprechpersonen bei der SFS durchgeführt werden. Die Umfrage beinhaltet die Ermittlung, zu wie viel Prozent das Kriterium für die Person erfüllt ist. Das Ziel für diesen Test ist mindestens 75%.	Gemäss dem Feedback der SFS lassen sich die Softwareartefakte mit den vordefinierten Umständen zu 100% in die Infrastruktur des Unternehmens integrieren. Die vordefinierten Umstände sind: 1. Die Schnittstelle vom bestehenden System zur neuen Software muss von der SFS geschaffen werden. 2. Die *GitLab-Pipeline* und das *Deployment* müssen erweitert werden, da die Zugriffsrechte der

			Verfasser für weitere Arbeiten nicht ausreichen.
NFR4	Die Datenabfragen von Dashboards sollen nur bei Bedarf erfolgen und die Anzahl «http-Requests» auf ein Minimum reduzieren.	Durch die Implementation und Tests von *WebSockets* o. ä. Technologien wird garantiert, dass die Datenabfrage nicht unnötig oft erfolgt (Verifikation erfolgt durch Architektur und Implementation).	Wie im Kapitel *4.1.1 Analyse und Entscheidung bezüglich WebSockets* erläutert, erfolgte aus technischen Gründen die Implementation von *Polling* statt von *WebSocket*. Eine mögliche Weiterentwicklung für die Erfüllung dieses Tests wird im Kapitel *5.2.1 Cloudbasierte WebSockets* beschrieben.
NFR5	Die Weiterentwicklung der Software soll durch sauberen und verständlichen Code sowie weitere Dokumentation im Repository gewährleistet sein.	Es sollen eine Codeanalyse und eine Umfrage mit zwei technischen Ansprechpersonen bei der SFS durchgeführt werden. Die Umfrage beinhaltet die Ermittlung, zu wie viel Prozent das Kriterium für die Person erfüllt ist. Das Ziel für diesen Test ist mindestens 75%.	Im Rahmen der Evaluation gab es Feedback von zwei Entwicklern, dass der Code sowie die Dokumentation verständlich sind. Es kamen einzelne Fragen auf, welche während der Evaluation geklärt wurden; somit ist das Kriterium zu 90% erfüllt.

5.2 Weiterentwicklungen

In diesem Kapitel werden die in dieser Arbeit festgestellten potenziellen Weiterentwicklungen definiert und beschrieben.

5.2.1 Cloudbasierte WebSockets

Im Kapitel *4.1.1 Analyse und Entscheidung bezüglich WebSockets* wurde festgehalten, weshalb im Rahmen dieser Arbeit keine *WebSockets* verwendet werden konnten. Das Hauptgegenargument waren die *Sticky Sessions*, d. h., dass ein Client über einen *WebSocket* fix mit einem Server verbunden bleibt. Dies würde bedeuten: Sollte bei einem skalierbaren System eine Server-Instanz abgeschaltet werden, geht die *Session* verloren und es muss eine neue aufgebaut werden, welche potenziell nicht den gleichen *State* hat.

Jedoch erachten die Verfasser einen cloudbasierten *WebSocket-Service* als Lösung, um eine *WebSocket*-Verbindung in einem skalierbaren Cloudumfeld zu ermöglichen. Beispielsweise verwenden die Autoren den *SignalR-Service* von *Azure* mit der offiziellen Dokumentation[5]. Hierbei wird ein Client, welcher eine *WebSocket*-Verbindung mit dem Server aufbaut, auf den *SignalR-Service* umgeleitet, damit der Client und der Server danach nur hierüber kommunizieren und damit der *State* zentral gehalten wird (siehe *Abbildung 28* und *Abbildung 29*). Dadurch fungiert der *SignalR-Service* wie ein Mittelsmann, der die Skalierbarkeit der Server ermöglicht, ohne dabei Rücksicht auf die *WebSocket*-Verbindungen zu nehmen.

[5] https://learn.microsoft.com/en-us/aspnet/core/signalr/scale

Durch die Umsetzung dieser Weiterentwicklung kann die NFA aus dem Anhang *A.2.3 Performance Efficiency* erfüllt werden, indem über *WebSockets* dann alle relevanten Datenänderungen an die Endgeräte gesendet werden und das bestehende *Polling* abgelöst wird.

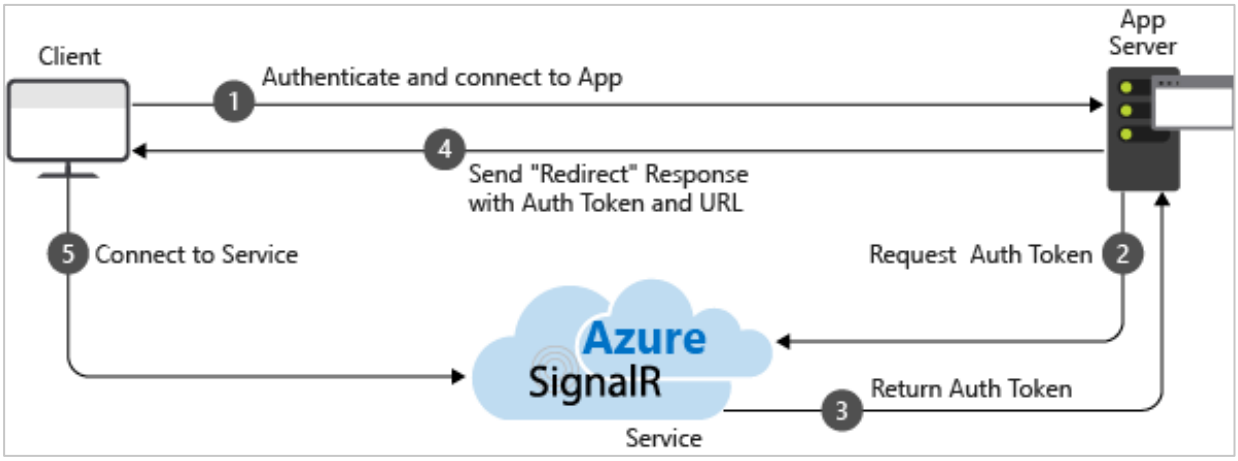


Abbildung 28: Prozess der Client-Umleitung auf den SignalR-Service von Azure
Quelle: https://learn.microsoft.com/en-us/aspnet/core/signalr/scale/_static/azure-signalr-service-one-connection.png

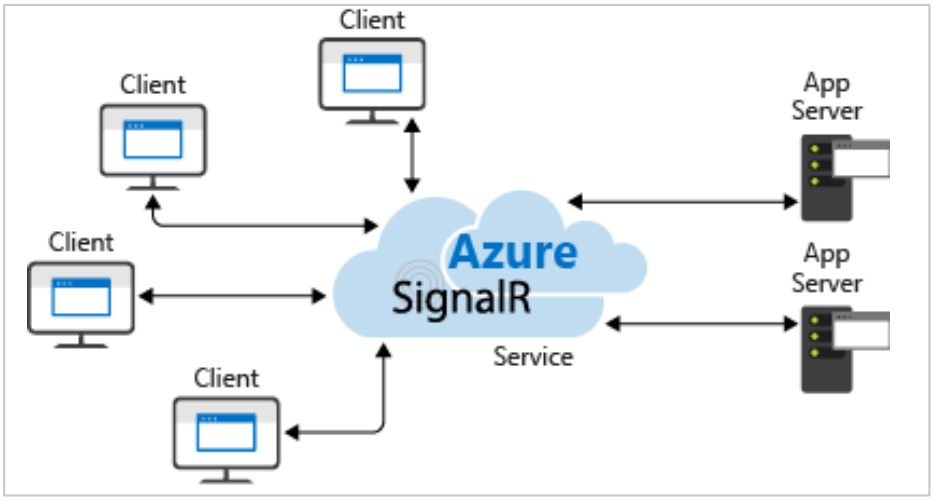


Abbildung 29: WebSocket-Kommunikation über den zentralen SignalR-Service von Azure
Quelle: https://learn.microsoft.com/en-us/aspnet/core/signalr/scale/_static/azure-signalr-service-multiple-connections.png

5.2.2 Offene Anforderungen aus den Usability-Tests

Im Rahmen der *Usability-Tests* wurden von den Nutzern Anforderungen und Änderungen vorgeschlagen. Diese sind im Kapitel *3.4.2 Resultate* mit ‹Weiterentwicklung SFS› oder ‹administratives Problem, …› gekennzeichnet. Sie sollten nochmals verifiziert und mit den Endnutzern abgesprochen werden und vorzugsweise nochmals einen *User-centered Design*-Prozess durchlaufen, bevor sie umgesetzt werden.

5.2.3 Restriktivere Patch-Requests

Im Kapitel *4.2.3 Feststellung bezüglich Patch-Requests* wurde festgehalten, weshalb die *Patch-Requests* im *Backend* nicht validiert werden. Das Hauptargument war, dass so Zeit beim Entwickeln gespart wird und es sich aktuell bei der Software nur um einen Prototyp handelt.

Die Verfasser erachten jedoch folgende Anpassung als essenziell, wenn die Applikation weiterverwendet werden soll: Es sollten validierbare *Model*-Klassen verwendet werden. Aktuell werden die *Patch-Requests* direkt auf den Entitäten ausgeführt; wenn aber die Adaptierung hinsichtlich *Model*-Klassen erfolgt, besteht die Möglichkeit, mit einer *Model*-Validierung die geänderten Werte zu prüfen. Dadurch ist sichergestellt, dass nur gültige Anpassungen erfolgen. Der Ablauf wäre dann wie folgt:

1. Abfüllen der *Model*-Klasse mit den Originaldaten
2. Ausführen der *Patch*-Befehle auf der *Model*-Klasse
3. Validieren der *Model*-Klasse
4. Übertagen der Daten der *Model*-Klassen auf Originaldaten

Ein weiterer Vorteil einer solchen Lösung wäre, dass die Validierung selbst angepasst werden kann und je nach Fehler genaue Fehlermeldungen ausgegeben werden können, um das Benutzen der *API* zu vereinfachen. (Piotr Zieliński, 2021)

Diese Validierung könnte mit Hilfe von Validierungsattributen auf den jeweiligen *Model*-Klassen realisiert werden, welche dann in einer Validationsklasse validiert werden, siehe dazu das Beispiel in *Abbildung 30*. (Microsoft, 2022b)

Weitere Informationen zur Validierung von *Model*-Klassen enthält die offizielle *Microsoft*-Dokumentation.[6]

```
public int Id { get; set; }
[Required]
public string Name { get; set; }
public decimal Price { get; set; }
[Range(0, 999)]
public double Weight { get; set; }
```

Abbildung 30: Beispiel für Validierungsattribute

Quelle: https://learn.microsoft.com/de-de/aspnet/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api

5.2.4 Testing der Service-Klassen im Backend

Im Kapitel *4.3.3 Entscheid bezüglich der Tests der* Services wurde festgehalten, weshalb die *Service*-Klassen im *Backend* nur mit Integrationstests geprüft wurden und dafür eine *In-Memory*-Datenbank verwendet wurde. Die Hauptargumente waren, dass Zeit eingespart wird und direkt Integrationstests für die Schnittstelle zwischen *Backend* und Datenbank vorliegen.

Die Verfasser erachten jedoch folgende Adaptierungen als passender, wenn die Applikation weiterverwendet werden soll:

- Erstellen von *Repository*-Klassen für Datenbankzugriffe
- Dediziertes Testen der fachlichen Logik in den *Services* und der Datenzugriffslogik im *Repository*
- zusätzliche *Unit*-Tests mit *Mocking*

Die bedeutsamste Anpassung wäre das Erstellen von *Repository*-Klassen. In diesen würden alle Datenbankzugriffe implementiert werden; dies würde das Testen der *Service*-Klassen vereinfachen, da dann für

[6] https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api

die *Repository*-Klassen nur ein *Mock* erstellt werden muss, da aktuell das *Mocken* der Datenbankzugriffe schwer möglich ist (siehe *Abbildung 31*). Zusätzlich wäre die Datenbankschnittstelle von der fachlichen Logik abgetrennt, was ein einfacheres Austauschen des *OR-Mappers* in Zukunft ermöglichen würde. Das Verwenden von *SQLite*-Datenbanken ist kein Muss, es würde aber Nachteile einer *In-Memory*-Datenbank beheben.
Weitere Informationen sind in der offiziellen *Microsoft*-Dokumentation enthalten.[7]

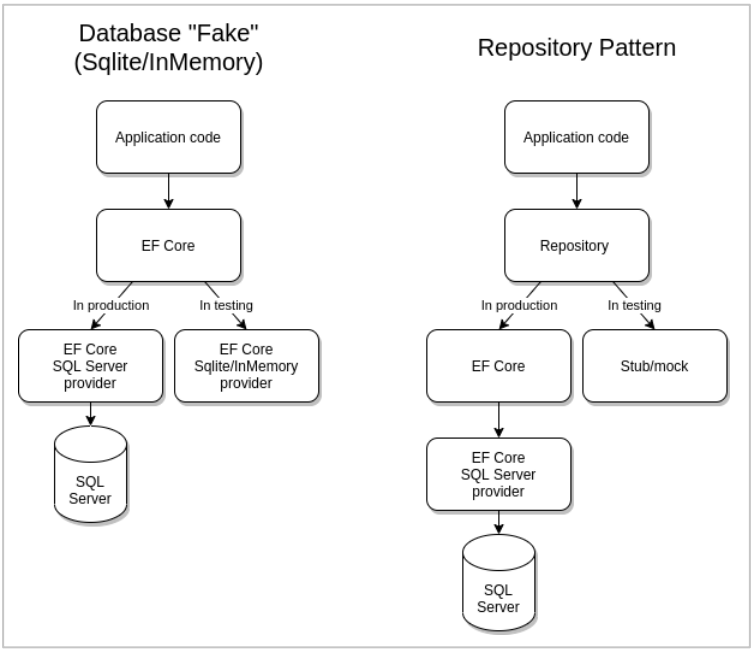


Abbildung 31: Aufbau beim Verwenden des Repository-Patterns
Quelle: https://learn.microsoft.com/de-de/ef/core/testing/choosing-a-testing-strategy

[7] https://learn.microsoft.com/de-de/ef/core/testing/choosing-a-testing-strategy

5.3 Evaluation mit der SFS Group AG

Am 19.12.2022 wurde zusammen mit der SFS eine Evaluation durchgeführt. Vor dieser wurde das Unternehmen aufgefordert, die gesamte Software durchzusehen, lokal eine Entwicklungsumgebung einzurichten und die Anwendung zu starten.

Bei der Evaluation wurden alle offenen Fragen, Unklarheiten und Probleme besprochen. Ein Entwickler der SFS hatte gemäss seiner Aussage die Software problemlos starten können, da diese den erwarteten Standards und Best Practices entsprochen habe. Es wurde zudem nochmals besprochen und bestätigt, dass das Unternehmen die bestehenden *GitLab-Pipelines* ausbauen wird und das *Deployment* der Applikation übernimmt, da den Verfassern nicht mehr Zugriffsrechte für die Systeme der SFS gewährt werden dürften.

Folgende Beschlüsse wurden während der Evaluation festgehalten:

- Die Studierenden schicken der SFS einen aktuellen Stand der Dokumentation sowie den finalen Stand nach der Abgabe.
- Die Studierenden fixen einen Fehler mit *Linter* im *Frontend*.
- Die Studierenden benennen eine Bezeichnung im Code um (‹WareEntryCharge› zu ‹Batch›).
- Die SFS baut die Schnittstelle von ihrem bestehenden System zur neu erstellen *REST-API*.
- Die SFS erweitert die *GitLab-Pipeline* und kümmert sich um das *Deployment* der Applikation.

6. Fazit

Wir sind mit der Arbeit zufrieden und konnten durch die Literaturrecherche Neues zu *User-centered Design*, *Usability* und *Dashboard*-Design lernen. Auf Basis dessen konnten wir das neue Wissen anwenden und ein Produkt schaffen, mit welchem die Nutzer in Zukunft gern arbeiten. Für uns war dies die erste Anwendung des *User-centered Design*-Prozesses, wobei wir viel lernen und unser Wissen vertiefen konnten.

Als Beispiel würden wir ein Abgleichen der Personas mit den Betroffenen im Nachhinein gesehen so nicht mehr durchführen. Wir erhielten Feedback, welches zum einen nicht mit den Vorgaben der Personas gemäss der Literatur übereinstimmte. Zum anderen mussten wir unserer Meinung nach relevante Informationen aus den Personas entfernen, da die Nutzer der Meinung waren, dass diese Angaben nicht ihrem Sinnbild entsprechen. Daher würden wir bei der nächsten Verwendung von Personas diese nicht mehr mit den Nutzern besprechen. Wir vermuten, dass die Nutzer sich zu stark mit den Personas identifizieren wollen, obwohl diese eine Nutzergruppe präsentieren soll und ein Kommunikationshilfsmittel ist.

User-centered Design brachte uns über die ganze Zeit einen Mehrwert und war das Fundament für die Entwicklung einer guten Applikation, mit der die Endnutzer zufrieden waren. Somit sind wir der Meinung, dass wir den Prozess, angepasst auf die verfügbaren Ressourcen, optimal angewendet haben. Dies wurde durch das positive Feedback der SFS nach den *Usability-Tests* bestätigt. Zudem legten wir so einen Grundstein für eine mögliche Weiterentwicklung der SFS. Wir sind der Meinung, dass wir die Ziele der Arbeit erreicht haben, und konnten dem Projektpartner das geforderte Produkt in guter Qualität und vollumfänglich bereitstellen. Zu den dokumentierten Weiterentwicklungen haben wir unsere Meinung und Empfehlungen eingebracht und hoffen, dass die SFS dadurch das Produkt weiter verbessern kann.

Zudem haben wir beide während der Arbeit spannende Diskussionen geführt, *Pair-Programming* betrieben und konnten einerseits voneinander lernen und andererseits uns gegenseitig stets unterstützen und fördern. Wir sind überaus zufrieden mit unserer selbständigen Organisation, Planung und Durchführung des Projekts, da wir stets unsere nächsten Ziele im Blick hatten und uns auch Zeit genommen haben für das Projektmanagement.

Als abschliessendes Feedback haben wir von Adrian Scherrer folgende Worte erhalten:

> *Abdullah und Ursin haben sehr selbständig gearbeitet. Dabei haben sie Informationen von verschiedenen Anspruchsgruppen eigenständig eingeholt und daraus Anforderungen abgeleitet. Aus den Anforderungen haben sie eine einfach und direkt deploybare Anwendung kreiert, die vorgängig definierten Rahmenbedingungen eingehalten und die Anwendung mit den Anwendern auf Usability geprüft. Wir sind mit dem Ergebnis aus User Sicht, vom Code, und aus Sicht Betrieb sehr zufrieden. Nach aktuellem Stand kann die Anwendung produktiv eingesetzt werden.*
>
> *Wir würden jederzeit gerne wieder eine Arbeit mit Abdullah und Ursin machen und wünschen Ihnen das Beste.*

7. Glossar & Abkürzungsverzeichnis

.

.NET 6
Ein plattformunabhängiges Software Framework von Microsoft 37

A

Angular
Ein TypeScript-basiertes Front-End-Webapplikationsframework. 37, 38, 39, 64
API
Schnittstelle zur Kommunikation mit anderen Applikationen 40, 41, 51, 53

B

Backend
Als Backend wird der Teil eines IT-Systems bezeichnet, der sich mit der Datenverarbeitung im Hintergrund beschäftigt 33, 37, 38, 39, 40, 42, 45, 46, 50, 51
Branch
Separate Kopie des Git Repositories innerhalb des Projekts 46
Build
Ausführbare Applikation, welche aus Code erzeugt wurde 39, 46
Businesslogik
Teile der Software, welche fachliche Logik abbilden 40, 41, 44, 45

C

Clean Architecture
Softwaredesign Pattern 40
Coil
Fachbegriff für eine Drahtspule 20, 21, 34, 62, 63, 65
Contextual Inquiry
Eine nutzerzentrierte Designforschungsmethode 3, 14, 19, 20, 21, 36
Controller
Klasse in welcher REST Schnittstellen definiert sind 39, 40, 41, 42, 45
CORS
Cross-Origin Resource Sharing 40

D

Dependency-Injection
Injiziert Abhängigkeiten während der Ausführung einer Applikation 41
Design-Walkthrough
Eine Besprechung bei der Wireframes besprochen werden. 31, 32, 36

Docker
Open-Source Software zur Isolation von Anwendungen in virtuellen Containern. 3, 43, 46, 47

E

EF Core
OR-Mapper für das .NET Framework 42, 46
Entities
Objekte zur Abbildung von Datenbank Tabellen 40
Exception
Unerwarteter Fehler welcher während der Ausführung einer Applikation auftritt 42

F

Frontend
Als Frontend bezeichnet man die Präsentationsebene, d.h. den Teil einer Applikation, den der Nutzer sehen kann 37, 38, 39, 40, 42, 44, 46, 53

G

Git Repository
Speichert alle Änderungen in einem Git Projekt 46
GitLab
Webanwendung zur Versionisierung mit Git 43, 46
GitLab Pipeline
Sammlung von mehreren Kommandozeileneingaben, welche nacheinander aufgerufen werden 46
GitLab Runner
Führt Jobs von GitLab Pipelines aus 43, 46
GitLab-Pipeline
Sammlung von mehreren Kommandozeileneingaben, welche nacheinander aufgerufen werden 43, 48, 53

H

Helper
Klasse in welcher Logik definiert wurde zur Unterstützung anderer Klassen 42, 45
HTTP
Hypertext Transfer Protocol 41, 44

I

Image Repository
Speicherort für Docker Images 46
In-Memory Database
Temporäre Testdatenbank für lokale Tests 46, 51
Interface
Definition einer Schnittstelle, welche implementiert wird 40, 42

K

Kubernetes Cluster
Ein Set von Nodes, um virtuelle Container auszuführen 43

L

L**azy Loading**
Wenn Teile einer Applikation erst dann geladen werden, wenn sie benötigt werden 38
Legierungscode
Fachbegriff für eine alte Materialnummer bei der SFS 34
Linter
Software zur statischen Codeanalyse 46, 53

M

Merge
Zusammenführen von zwei Branches 46
Merge-Request
Anfrage, ob ein Branch in einen anderen gemerged werden darf 46
Microservice
Ein Service eines Architekturpatterns, welches Software in unabhängige Services aufteilt 37
Middleware
Softwarekomponente zwischen zwei oder mehr weiteren Softwarekomponenten 40, 41, 42
Mock
Platzhalter eines Objekts für Softwaretests 40, 44, 46, 52
Moodboard
Das Moodboard ist ein oft genutztes Arbeits- und Präsentationsmittel in Kommunikations- und Designberufen 25
Moq
Library zur Erstellung von Mock Tests in .NET 42

N

nginx.conf
Konfiguration eines nginx-Webservers 46
NUnit
Unit Tests Framework für .NET 42

O

Observable
Eine nachgelagerte Berechnung oder Evaluation, welche synchron oder asynchron erfolgen kann 39
OR-Mapper
Bildet Software Objekte auf Datenbank Objekte ab 42, 52
Over-Engineering
Wenn eine Software viel detaillierter als nötig entwickelt wurde. 46

P

Pain-Point
Eine Art Schmerzpunkt, welche den Nutzer bei der Verwendung einer Applikation o.ä. stört oder behindert 20
Patch
HTTP-Request für Teiländerungen 42, 43, 50, 51
Pipe
Eine Klasse zur Transformation von Daten 39
Polling
Regelmässige Abfrage einer Ressource 38, 49, 50
Prettier
Software zur Codeformatierung 46
Proxy
Vermittler zwischen Netzwerk-Komponenten 39

R

Release
Veröffentlichte Version einer Software 46
Repository Klasse
Klasse für Datenbankzugriffe 40, 46, 51
REST
Representational State Transfer 41, 53
Route
Eine Konfiguration, welche z.B. den Pfad einer Webressource definiert 38

S

Scrum
Ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung 19
Service
Eine Klasse in welcher Businesslogik definiert ist 42, 45, 46, 51
Snapshot
Abbildung des aktuellen Zustands einer Datenbank 42
SQL
Structured Query Language 42
SQLite
Relationale Datenbank welche alle Daten in ein File schreibt 46, 52
State
Zustand einer Webressource 37, 49
Stubbing
Ein Stub bezeichnet einen Stellvertreter/Platzhalter bei Softwaretests 46
Subscription
Repräsentiert die Ressource des Observables 39
Swagger-API
API zur Dokumentation eines Web Services 41

T

Third-Party

U

W

8. Literatur- & Quellenverzeichnis

Beyer, H. & Holtzblatt, K. (2009). *Contextual design: Defining customer-centered systems*. *Morgan Kaufmann series in interactive technologies*. Morgan Kaufmann.

Cooper, A., Reimann, R., Cronin, D., Noessel, C., Csizmadi, J. & LeMoine, D. (2014). *About face: The essentials of interaction design / Alan Cooper* (Fourth edition / Robert Reimann, David Cronin, Christopher Noessel with Jason Csizmadi and Doug LeMoine). Wiley.

Hartung, A. (2. Dezember 2015). The Reason Why Google Glass, Amazon Fire Phone and Segway All Failed. *Forbes*. https://www.forbes.com/sites/adamhartung/2015/02/12/the-reason-why-google-glass-amazon-firephone-and-segway-all-failed/

Human Factors and Ergonomics Society. (2008). *ANSI/HFES 200 ANS I-HFES 200.5*. https://law.resource.org/pub/us/cfr/ibr/006/hfes.200.2.html

ISO 25010. (2022, 21. Oktober). https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Marsh, S. (2022). *User research: Improve product and service design and enhance your UX research / Stephanie Marsh* (Second edition). KoganPage.

Microsoft. *NET*. https://dotnet.microsoft.com/en-us/

Microsoft. (2022a). *Common web application architectures: Clean architecture*.

Microsoft. (2022b). *Model Validation in Web API*.

Microsoft. (2022c). *Testing without your production database system*.

Piotr Zieliński. (2021). *When (not) use JSON Patch in ASP.NET Core. A real-life testimony*.

Richard Brath & Michael Peters. (2004). *Dashboard Design: Why Design is Important*. http://cs.furman.edu/~pbatchelor/csc105/articles/TUN_DM_ONLINE.pdf

SFS Group. (2022, 25. Oktober). *Inventing success together.* SFS Group. https://www.sfs.com/ch/de/

Shannon Brown. (2018). *Designer Tips for Choosing Dashboard Colors*. https://www.linkedin.com/pulse/designer-tips-choosing-dashboard-colors-shannon-brown

Wikipedia (Hrsg.). (2021a). *Dashboard (Informationsmanagement)*. https://de.wikipedia.org/w/index.php?title=Dashboard_(Informationsmanagement)&oldid=217008130

Wikipedia (Hrsg.). (2021b). *Moodboard*. https://de.wikipedia.org/w/index.php?title=Moodboard&oldid=216061999

Zühlke, D. (2012). *Nutzergerechte Entwicklung von Mensch-Maschine-Systemen*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-22074-6

Anhang

A Software-Dokumentation

A.1 User Stories

Alle User Stories sind auf eine einzelne Persona (siehe *3.2 Personas*) ausgelegt.

Ein Teil der Integration wird von der SFS entwickelt. Diese werden im Rahmen dieser Arbeit nur im Backend umgesetzt und sind in den Akzeptanzkriterien mit einem roten Stern (*) markiert.

A.1.1 User Story-1

Als Lagermitarbeiter (Noah Huber) möchte ich mir einen Überblick über alles Bestellungen verschaffen können.

Akzeptanzkriterien:

1. Auf dem Dashboard sind die Kennzahlen für Bestellungen und deren Status ersichtlich.
    a. Express-Bestellungen werden erkenntlich angezeigt.
2. Die Bestellungen sind auf dem Dashboard nach dem Lagerstandort gruppiert.
3. Änderungen an Bestellungen werden auf dem Dashboard automatisch aktualisiert.

A.1.2 User Story-2

Als Lagermitarbeiter (Noah Huber) möchte ich die offenen Bestellungen abarbeiten können.

Akzeptanzkriterien:

1. Bestellungen können sich selbst zugewiesen werden.
2. Beim Abarbeiten einer Bestellung fordert die Applikation einen auf:
    a. Die Materialnummer und Wareneingangcharge einzuscannen. (Verifikation & Nachweisbarkeit)
    b. Die Anzahl Coils anzugeben (Hilfestellung für den Staplerfahrer)
    c. Zu kontrollieren und anzugeben, ob ein Wareneingangende, Chargenende oder Drahtende erreicht wurde.
4. Nachdem die Bestellung abgearbeitet wurde bzw. einen konfigurierten Status erreicht hat, wird die Bestellung für den Lagermitarbeiter (Noah Huber) als erledigt betrachtet.

A.1.3 User Story-3

Als Lagermitarbeiter (Noah Huber) möchte ich auf verschiedenen Geräten alle Wareneingangenden, Chargenenden und Drahtenden (im Weiteren als «Enden» bezeichnet) einsehen und bearbeiten können.

Akzeptanzkriterien:

1. Für die Übersicht von den «Enden» steht eine responsive Ansicht zur Verfügung.
2. Die «Enden» können eingesehen und gelöscht werden.

A.1.4 User Story-4

Als Staplerfahrer (Nico Meier) möchte ich die Bestellungen, welche bereit zur Lieferung sind, einsehen können.

Akzeptanzkriterien:

1. Die kontrollrelevanten Informationen, wie Materialnummer, Wareneingangcharge und Anzahl Coils werden angezeigt.
2. Die lieferungsrelevanten Informationen, wie Teamleiter und Maschinennummer werden angezeigt.
    a. Ein Lageplan (PDF oder Bild) steht als Hilfestellung bereit und kann geöffnet werden.
3. Die Bestellung kann als ausgeliefert/abgeschlossen markiert werden.

A.1.5 User Story-5

Als Produktionsmitarbeiter (Hans Graf) möchte ich Bestellungen erstellen und beobachten können.

Akzeptanzkriterien:

1. *Bestellungen können erstellt werden.
    a. *WE-Charge kann optional angegeben werden.
    b. *Die Bestellung kann als Express definiert werden.
2. *Bestellungen können storniert oder verändert werden.
    a. *Änderungen an der Bestellung (stornieren oder Lieferort ändern) können nur bis zu einem definierbaren Status durchgeführt werden.

A.2 Nicht-funktionale Anforderungen (NFR) & Qualitätsattribute

Als Grundlage für die NFRs verwenden wir den ISO 25010 Standard. (*ISO 25010*, 2022) Wir gehen nur auf die, für unser Projekt relevanten Bereiche, des ISO-Standards ein.

Die Ergebnisse der Tests sind im Kapitel *5.1 der funktionalen und der nichtfunktionalen* Anforderungen aufgeführt.

A.2.1 Usability

Nummer	Akzeptanzkriterien	Tests
NFR1	Ein Nutzer soll sich in der Benutzer-oberfläche selbständig zurechtfinden und die Bedienung erlernen können.	Usability-Tests mit 2 Benutzern, welche selbstän-dig den Workflow einer Bestellung in der Applika-tion durchspielen können.
NFR2	Die Applikation soll den Arbeitsalltag sowie den Bestellprozess erleichtern.	Beim Usability-Test soll mittels einer Umfrage, ob das Kriterium erfüllt ist, eine Zustimmung von min-destens 70% erreicht werden.

A.2.2 Compatibility

Nummer	Akzeptanzkriterien	Tests
NFR3	Die erstellten Softwareartefakte können ohne Umstände in die weitere Infrastruktur von SFS integriert werden.	Codeanalyse und Umfrage mit zwei technischen Ansprechpersonen bei der SFS. Umfrage beinhaltet eine Frage, zu wie viel Prozent das Kriterium für die Person erfüllt ist. Das Ziel für diesen Test ist min. 75%.

A.2.3 Performance Efficiency

Nummer	Akzeptanzkriterien	Tests
NFR4	Die Datenabfragen von Dashboards sollen nur bei Bedarf erfolgen und die Anzahl «http-Requests» auf ein Minimum reduzieren.	Durch die Implementation und Tests von WebSockets o.ä. Technologien wird garantiert, dass die Datenabfrage nicht unnötig oft erfolgt. (Verifikation erfolgt durch Architektur und Implementation)

A.2.4 Maintainability

Nummer	Akzeptanzkriterien	Tests
NFR5	Die Weiterentwicklung der Software soll durch sauberen und verständlichen Code sowie weitere Dokumentation im Repository gewährleistet sein.	Codeanalyse und Umfrage mit zwei technischen Ansprechpersonen bei der SFS. Umfrage beinhaltet eine Frage, zu wie viel Prozent das Kriterium für die Person erfüllt ist. Das Ziel für diesen Test ist min. 75%.

A.3 Technische Beschränkungen

Es werden folgende Technologien/Frameworks für die Entwicklung der jeweiligen Schichten verwendet:

Frontend	Angular
Backend	.NET 6.0
Datenbank	MariaDB

Des Weiteren sollen alle Teilsysteme in Docker-Containern laufen.

Folgende technische Beschränkungen beziehen sich auf die einzelnen User Stories:

User Story	Beschränkung
User Story-1	Im Lager läuft das Dashboard auf einem Monitor an der Wand.
User Story-2	Der Lagermitarbeiter verwendet ein portables Gerät wie z.B. ein Smartphone.
User Story-4	Der Staplerfahrer verwendet ein Tablet oder Tablet-ähnliches Gerät.

A.4 Weitere Anforderungsentscheidungen

Nach Austausch mit dem SFS wurden folgende Entscheidungen bezüglich der Anforderungen festgelegt:

1. Es soll keine Authentifizierung implementiert werden. Durch die Zutrittsregelungen und Firewalls wird gewährleistet, dass niemand unerlaubt Zugriff auf Anwendungen erhält. Zur Identifikation von Personen, wird und soll weiterhin eine Identifikationsnummer eingegeben werden können.

A.5 Domain Model vom 08.10.2022

Basierend auf den Anforderungen aus dem Kapitel *A.1 User Stories* und im Austausch mit dem Projektpartner wurde folgendes Domain Model erstellt.

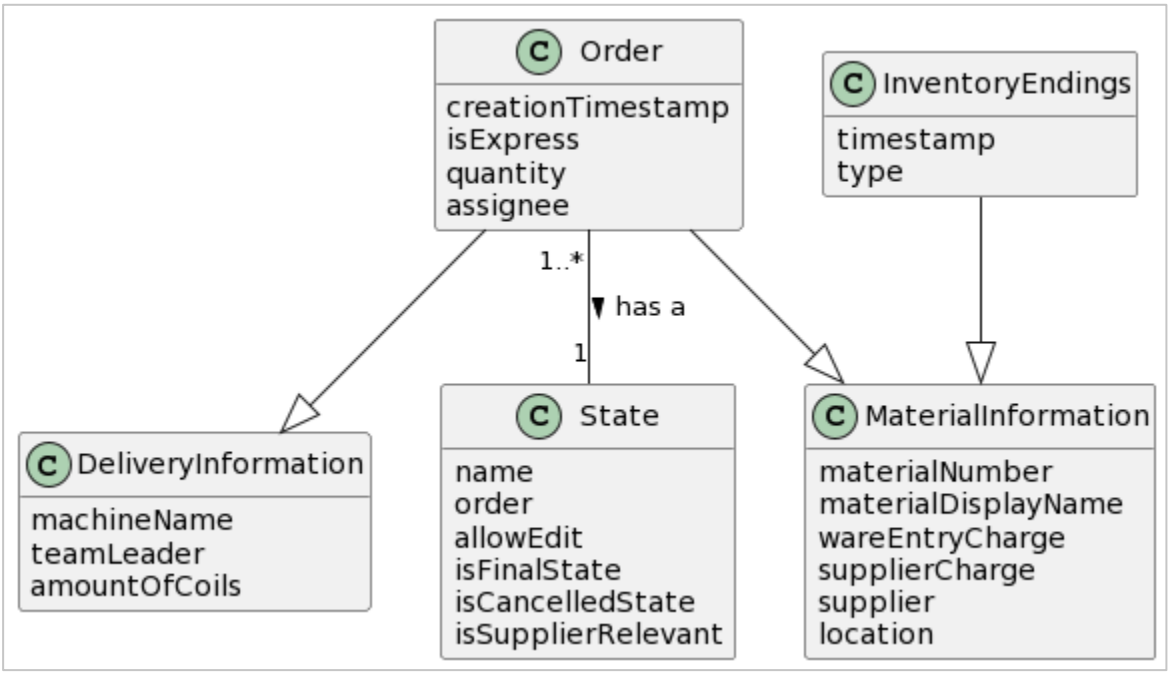


Abbildung 32: Domain Model V1
Quelle: Erstellt mit PlantUML

Weitere Ausführungen zum Domain Model:

Lieferungsinformationen (DeliveryInformation)

Die Lieferungsinformation abstrahieren die Informationen, welche für den Staplerfahrer relevant sind. Der Maschinenname und Teamleiter sind Informationen, welche dem Staplerfahrer mitteilen, wohin die Bestellung ausgeliefert werden muss. Die Anzahl Coils (amountOfCoils) sind eine Hilfestellung für den Staplerfahrer, welche der Lagermitarbeiter angibt, damit der Staplerfahrer bereits weiss wie viele Coils zu einer Bestellung gehören und diese so leichter identifizieren kann und auch nichts vergisst.

Status (State)

Hier sollen die verschiedenen Status konfigurierbar sein, welche eine Bestellung haben kann. Die weiteren Eigenschaften wie z.B. «allowEdit», «isFinalState» erlauben den Workflow zu steuern.

Für den Status wurde eine grundlegende Entscheidung getroffen, welche im Kapitel *A.5.1 Entscheidung vom 10.10.2022 bezüglich Status (State)* dokumentiert ist.

Materialinformation (MaterialInformation)

Hier werden die konkreten Informationen zum Material abstrahiert, welche nicht direkt bestellungsrelevant sind. Ein Material hat immer eine Identifikationsnummer, einen Anzeigenamen und Lagerstandort. Zusätzlich beinhaltet jedes Material die Informationen zum Lieferanten, der Liefercharge und der Wareneingangcharge.

Bestellung (Order)

Die Bestellung ist hier das Kernelement. Relevant für die Bestellung sind das Eingangsdatum, die Menge, der Bearbeiter und der Identifikator, ob es sich um eine Expressbestellung handelt. Des Weiteren beinhaltet die Bestellung Informationen zum Material und zur Lieferung. Der Bearbeiter der Bestellung wird nicht zu Beginn festgelegt, sondern im Prozess, wenn der Lagermitarbeiter sich eine Bestellung zuweist.

Inventar-Enden (InventoryEndings)

Wie im Kapitel *3.1.3 Ablauf einer Bestellung* nachzuvollziehen ist, wird in einer Liste festgehalten, ob es sich bei einer Herausgabe des Materials um ein Wareneingang-, Chargen oder Drahtende handelt. Diese statische Liste und die administrative Arbeit erübrigten sich durch die Erfassung und späteren Darstellung dieser Daten. Relevant sind hier, um welchen Typ von Ende es sich handelt, sowie wann dieser erfasst wurde.

A.5.1 Entscheidung vom 10.10.2022 bezüglich Status (State)

Im Rahmen eines Austausches mit dem Betreuer der Arbeit, wurde das Prinzip von konfigurierbaren Status und intuitivem Workflow thematisiert. Bei konfigurierbaren Status muss ein Benutzer immer den Status selbst setzen, da durch die Flexibilität kein sauberer Workflow garantiert werden kann. Beim statischen Status bzw. intuitivem Workflow ist der Workflow vorgegeben, d.h. die Applikation kann anhand der Arbeitsweise des Benutzers den Status automatisch setzen und der Benutzer muss nur im Falle eines Bedienungsfehlers den Status manuell ändern.

Aufgrund der Komplexität der konfigurierbaren Status und dem Fokus auf User Centered Design haben wir uns für die statischen Status entschieden.

A.5.2 Entscheidung vom 11.10.2022 bezüglich Bestellungsbearbeiter

Aus einer Diskussion bzgl. Usability und Design wurde entschieden, für die Bestellung zwei Bearbeiter zu erfassen, nämlich zum einen für den Bearbeiter im Lager und zum anderen für den Staplerfahrer. Nämlich sollen Bestellungen in den Ansichten nach «Alle» und «Meine Bestellungen» gefiltert werden, damit zum einen ersichtlich ist, ob und wer allenfalls bereits an einer Bestellung arbeitet und zum anderen soll der Mitarbeiter während der Abarbeitung einer oder mehrerer Bestellungen nicht durch neue hinzukommende Bestellungen gestört werden, welche in real-time dazukommen.

A.5.3 Weitere Inputs von der SFS vom 17.10.2022

Die SFS brachte konkrete Inputs zu den Angaben von Inventar-Enden ein, welche im ersten «Domain Model» noch nicht berücksichtig wurden.

A.6 Domain Model vom 22.10.2022

Anhand der im Kapitel *A.5 Domain Model vom 08.10.2022* getroffenen Entscheidungen und Inputs wurde das bisherige Domain Model überarbeitet.

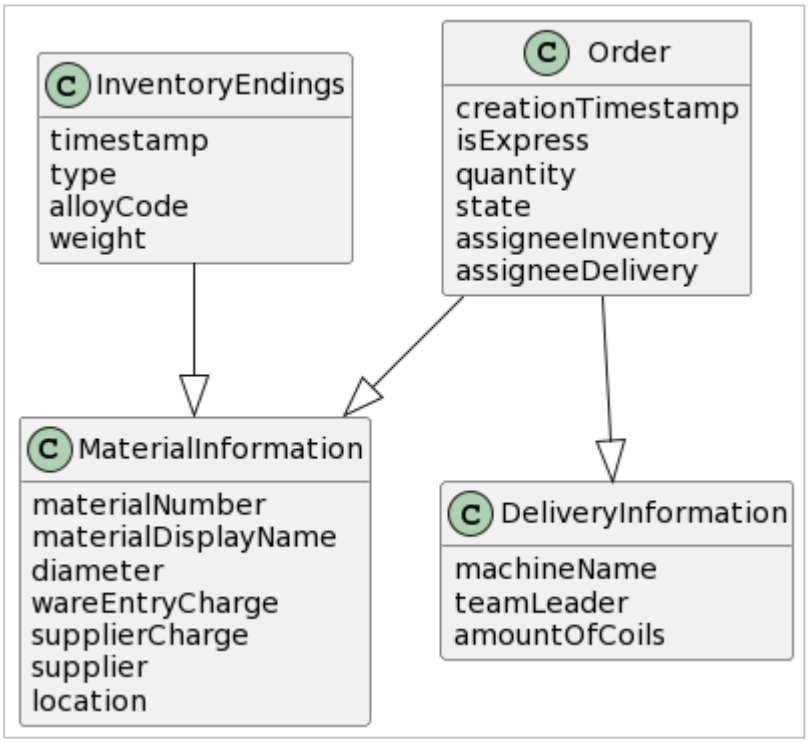


Abbildung 33: Domain Model V2
Quelle: Erstellt mit PlantUML

A.7 Datenbank Model

Basierend auf den Anforderungen und dem Domain-Modell *A.6 Domain Model vom 22.10.2022* wurde folgendes Datenbank Model erstellt.

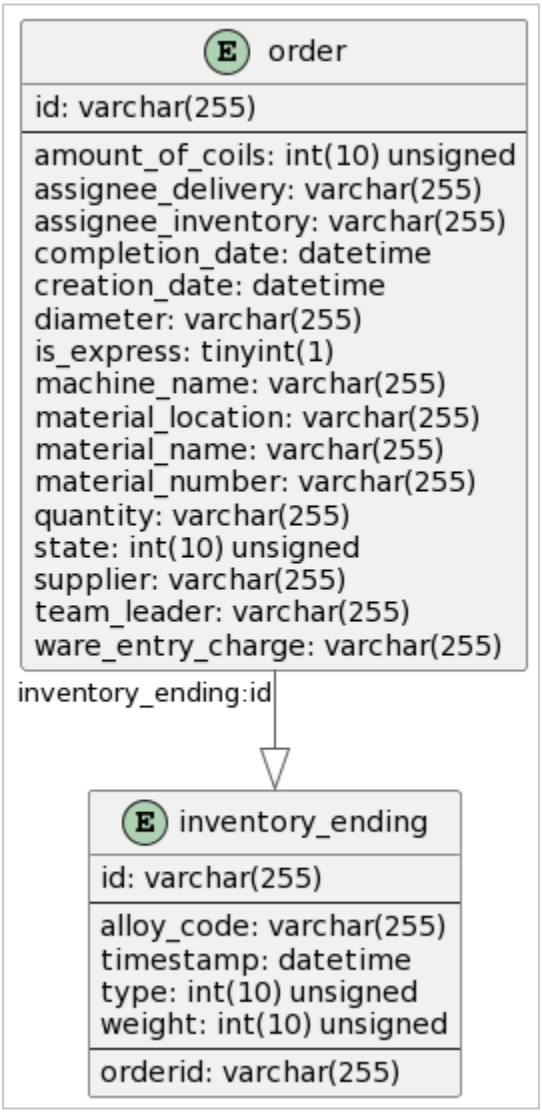


Abbildung 34: Datenbank Model

Quelle: Erstellt mit Rider und PlantUML

Die Datenbank wurde in die zwei Tabellen, *order* und *inventory_ending*, aufgeteilt, welche nachfolgend beschrieben sind.

A.7.1 Order

In der *order*-Tabelle werden alle Daten zu einer Bestellung persistiert.

Die Kerndaten einer Bestellung (alle Pflichtfelder) werden vom Umsystem der SFS an das Backend geschickt und dann in die Datenbank geschrieben. Gewisse Datenfelder werden erst während der Bearbeitung des Bestellprozesses ergänzt. Wenn eine Bestellung abgeschlossen wird, ist der Datensatz komplett.

Da das erstellte System in dieser Arbeit keinen Zugriff auf die Primärsysteme der SFS hat, ist es auch nicht mögliche die Tabelle *order* in mehrere Tabellen mit Relationen aufzubrechen. Daher führt dies dazu, dass z.T. doppelte Daten in derselben Spalte vorkommen können. Beispielsweise könnte bei einem Zugriff auf ein Primärsystem die Materialnummer als Fremdschlüssel dienen. Dann würden sich Felder wie Materialname, Materialstandort und Durchmesser in der Tabelle erübrigen, da man diese anhand des Fremdschlüssel im Primärsystem der SFS auslesen könnte.

A.7.2 Inventory_Ending

In der *inventory_ending* Tabelle wird ein Eintrag erfasst, falls eine Bestellung ein Wareneingangende, Chargenende und/oder Drahtende bewirkt. Dadurch wird die aktuelle händische Liste im Lager der SFS abgebildet bzw. soll diese ablösen. Die aktuellen «Enden» können daher nun eingesehen und entfernt werden, äquivalent zu der bestehenden händischen Liste.

A.7.3 Begrüdung zu den Varchar-Grössen

Wir haben uns entschieden alle «varchar» Spalten auf die Grösse 255 zu setzen. Dies hat den Grund, dass alle Textdaten vom Primärsystem der SFS bereits validiert wurden und an diese Datenbank weitergereicht werden.

Des Weiteren spielt es die Länge des «varchar» keine Rolle für den Speicherverbrauch. Dieser verbraucht genau so viel Speicher wie sein Inhalt beträgt. D.h. der Text «Hello World» verbraucht bei einem «varchar(20)» und «varchar(255)» genau gleich viel Speicher.

A.8 Frontend-Architektur

A.8.1 Verzeichnisstruktur

Die komplette Verzeichnisstruktur des Frontend wird hier nicht als Bild aufgeführt, da diese stark verschachtelt ist und daher keinen Überblick bieten würde. Die Verzeichnisstruktur basiert aber auf den Konzepten von den beiden verwendeten Frameworks Nx und Angular.

- Nx: https://nx.dev/more-concepts/folder-structure
- Angular: https://angular.io/guide/file-structure

A.8.2 Libraries & Frameworks

Die Informationen zu den 3rd-Party-Lizenzen des gesamten Frontend befinden sich im Repository im «3rdpartylicenses.txt»

Angular	Angular ist ein Framework, um moderne Webapplikationen und PWA zu erstellen. https://angular.io/ Alternativen: • React (https://reactjs.org/) • Vue.js (https://vuejs.org/) Angular wurde verwendet, da dies einerseits eine technische Beschränkung vom der SFS ist, aber andererseits wäre es auch das Framework unserer Wahl für eine solche Webapplikation gewesen.
Angular Material	Angular Material ist eine Library mit fix fertigen Komponenten, Services, usw., welche bei der Umsetzung Zeit erspart und ein einheitliches sowie angenehmes Benutzererlebnis ermöglicht. https://material.angular.io/ Alternativen: • Microsoft Fluent Design System (https://www.microsoft.com/design/fluent/#/) Die Entscheidung fiel auf «Angular Material», da diese Library im Rahmen der Arbeit viel Zeit spart und bereits Komponenten oder Logik mitbringt, welche hier benötigt werden.
Cypress	Cypress ist eine Library für Integrations- und UI-Test im Web. https://www.cypress.io/

	Wir haben uns für Cypress entschieden, da wir bereits in vergangenen Projekten mit Cypress komplexe Integrationstest erstellen konnten und diese bei jedem Build automatisiert ausführen konnten.
Jest	Jest ist die bekannteste und meistverbreitete Library für Unittests im Webumfeld. https://jestjs.io/ Alternativen: • Angular Standard (Jasmine + Karma https://angular.io/guide/testing) Die Entscheidung fiel auf Jest und da wir die Erfahrung gemacht haben, dass Jest gewisse Probleme beim Testen eleganter lösen als andere Testing-Libraries
Nx	Nx ist ein Entwicklungsframework «on-top» von Angular. Nx bringt moderne Tools für die Webentwicklung mit, wie z.B. Jest, Cypress und einige ESLint Regeln. https://nx.dev/ Wir haben diese Library bereits in unsere Studienarbeit evaluiert und als durchaus nützlich erlebt und haben uns entschieden, diese im Rahmen dieser Arbeit erneut zu verwenden.
ng2-pdf-viewer	ng2-pdf-viewer ist eine Library, welche die Verwendung von «pdfjs-dist» (https://www.npmjs.com/package/pdfjs-dist) vereinfacht und es ermöglicht PDFs im Web direkt darzustellen. https://www.npmjs.com/package/ng2-pdf-viewer Auf Wunsch der SFS sollte das PDFs des Lageplans direkt in der Applikation und nicht im neuen Tab angezeigt werden. Um dies zu ermöglichen haben wir uns für diese Library entschieden.
ngx-pull-to-refresh	Dies ist eine kleine und aktuelle Library, welche auf Touch-Geräten die intuitive Funktion «Pull to Refresh» zur Verfügung stellt. D.h. auf Listen kann man nach unten ziehen, damit ein Ladeindikator angezeigt wird, dass gerade neue Daten geladen werden. https://www.npmjs.com/package/ngx-pull-to-refresh Wir haben diese Library verwendet, da wir sonst für einen kleinen Use-Case viel Zeit und Code sparen können. Theoretisch ist es möglich die Library zu entfernen und die Komponente selbst zu programmieren.

A.9 Backend-Architektur

A.9.1 Verzeichnisstruktur

Die Verzeichnisstruktur des Backend basiert auf dem Pattern Clean Architecture wurde aber etwas angepasst, um dem Umfang der Arbeit nicht zu sprengen. (siehe Kapitel *4.2.2 Detailumsetzung Backend*)

- Clean Architecture Dokumentation: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
- Clean Architecture mit C#: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures#clean-architecture

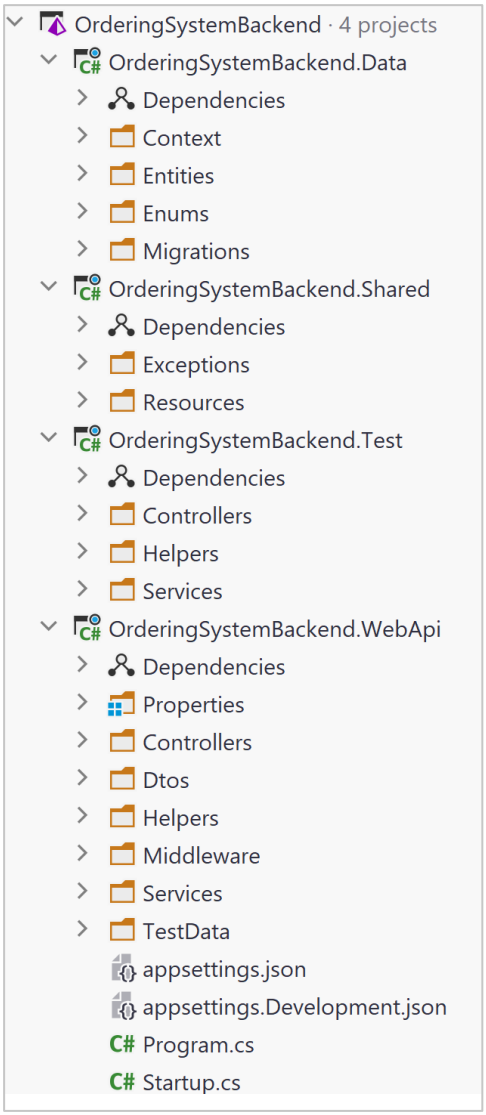


Abbildung 35: Verzeichnisstruktur Backend
Quelle: Erstellt mit Rider

A.9.2 Libraries & Frameworks

.NET 6.0	.NET 6.0 ist ein Entwickler Plattform für diverse Applikationen. (Microsoft) https://dotnet.microsoft.com/en-us/ Alternativen: • Spring Boot (https://spring.io/projects/spring-boot) Da wir beide mehr Erfahrung mit .NET 6.0 haben, haben wir uns nach Absprache mit der SFS dazu entschieden .NET 6.0 zu verwenden anstelle von Spring Boot.
Entity Framework	Das Entity Framework ist ein von Microsoft entwickeltes ORM. https://learn.microsoft.com/de-de/ef/ Alternativen: • Hibernate (https://hibernate.org/) • DevExpress XPO (https://www.devexpress.com/products/net/orm/) Da das Entity Framework eine Standard Library von Microsoft ist und diese am besten dokumentiert ist, fiel die Entscheidung auf das Entity Framework.
Swagger	Swagger wird verwendet, um APIs zu dokumentieren und deren Entwicklung zu vereinfachen. https://swagger.io/ Alternativen: • Postman (https://www.postman.com/) Wir haben bereits in mehreren privaten und firmen Projekten Swagger verwendet und sind daher sehr vertraut mit der Anwendung und sind der Meinung, dass diese einem solchen Projekt einen Mehrwert liefert.

B Projektmanagement

B.1 Meilensteine

Meilensteine	Termin	Quality Gates	Bemerkungen
Anforderungs-analyse	18.10.22	1. Nutzeranalyse durchgeführt und dokumentiert 2. Personas erstellt und dokumentiert 3. Funktionale sowie nicht-funktionale Anforderungen erstellt und vom Projektpartner abgenommen 4. Domain Model erstellt und dokumentiert	Meilenstein abgeschlossen am 19.10.2022.
Entwurf	01.11.22	1. Wireframes gezeichnet und abgelegt 2. Datenbank geplant und erstellt 3. Analyse von Libraries 4. Entwicklungsrelevante Tools und Systeme aufgesetzt und konfiguriert 5. Architektur-Prototyp erstellt.	Meilenstein abgeschlossen am 01.11.2022
Implementation & Test	29.11.22	1. Backend und Frontend fertig implementiert und getestet	Meilenstein abgeschlossen am 29.11.2022
Evaluation	20.12.22	1. Usability-Tests 2. Evaluation durchgeführt	Meilenstein abgeschlossen am 20.12.2022
Abgabe	13.01.23	1. Technischer Bericht abgeschlossen 2. Software-Dokumentation abgeschlossen 3. AVT relevante Abgabe erstellt und abgegeben 4. Alle abgaberelevanten Dokumente abgegeben	Meilenstein abgeschlossen am 08.01.2023

B.2 Team, Rollen und Verantwortlichkeiten

Teammitglied	Rolle	Verantwortungen
Abdullah Almaz	Software-Entwickler	- Entwicklung & Testing Frontend - Unterstützung Backend - Projektmanagement - Dokumentation / Bericht
Ursin Zimmermann	Software-Entwickler	- Entwicklung & Testing Backend - Unterstützung Frontend - Projektmanagement - Dokumentation / Bericht

B.3 Entwicklungswerkzeuge & eingesetzte Software

Tools & Software	Verwendung
Cypress & Jest	Frontend Integration-Testing
Docker Desktop	Docker-Verwaltung
ESLint	Frontend Code Quality
Figma	Paper Prototype
GitLab	Code-Versionierung, Code-Reviews, CI, CD, Release Management
JetBrains Rider	Entwicklung Backend (.NET 6.0)
JetBrains WebStorm	Entwicklung Frontend (Angular)
Miro	Moodboard, Architekturübersicht
MS Office Produkte	Dokumentation
PlantUML, draw.io	Diagramme erstellen für Dokumentation
Resharper	Backend Code Quality
YouTrack	Projektplanung, Zeiterfassung, Issue-Management, Sprint-Planung

B.4 Aufwandschätzung, Zeitplan, Projektplan

Der Projektplan, die Zeitplanung und ein Export all unserer Tickets mit den dazugehörigen Aufwandschätzungen befinden sich im Anhang *C Projektplan*, *I Zeitprotokolle* und *J Export Tickets*.

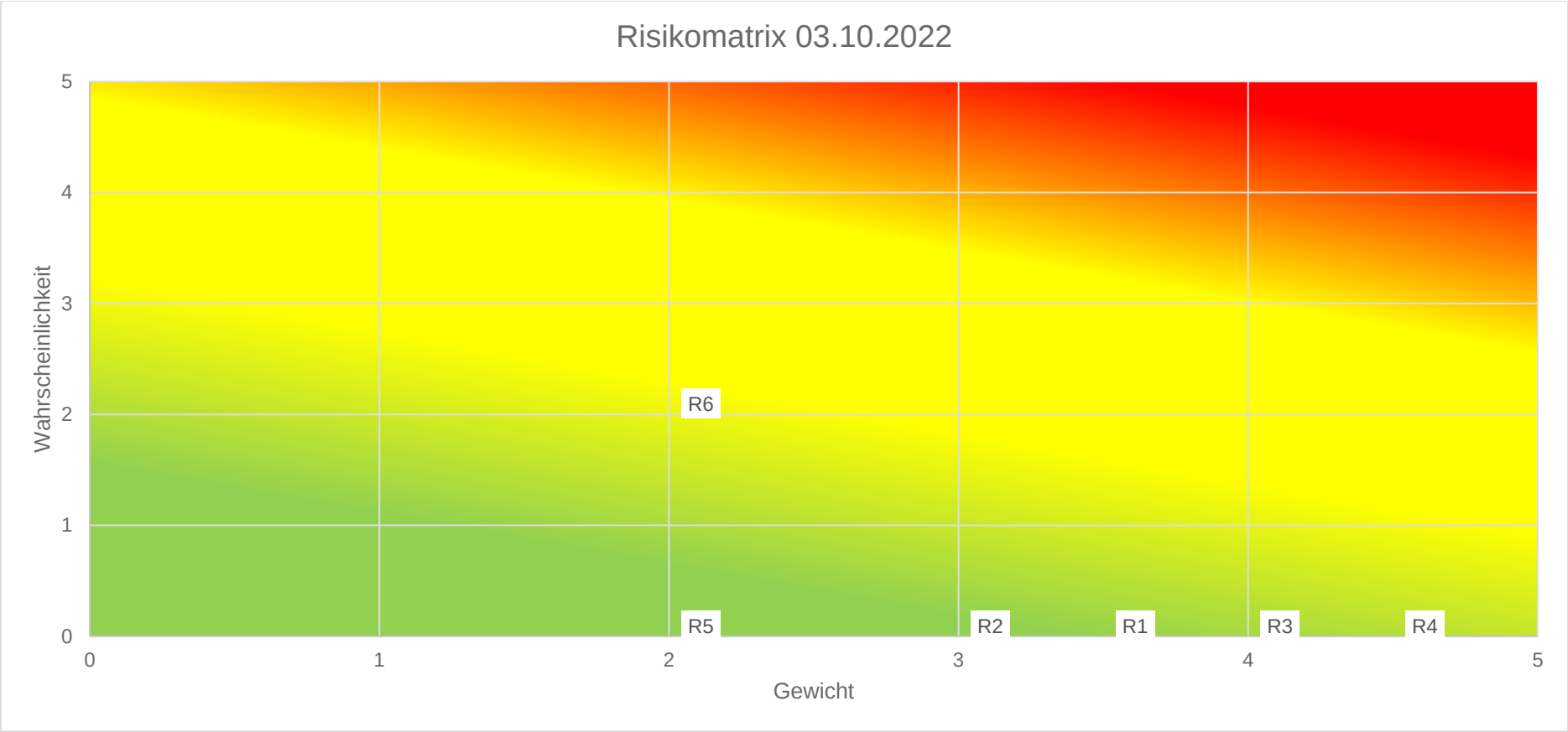
B.5 Risiken

Unsere Risikoanalyse soll uns eine Übersicht über mögliche Risiken geben. Wir ergänzen während des Projekts fortlaufend unsere Risiken.

Wir versuchen die meisten unserer Risiken durch die definierte Vorbeugung zu vermeiden. Falls jedoch ein Risiko trotzdem eintritt, haben wir in der Projektplanung zu jedem Meilenstein einen Reserveplatzhalter eingeplant.

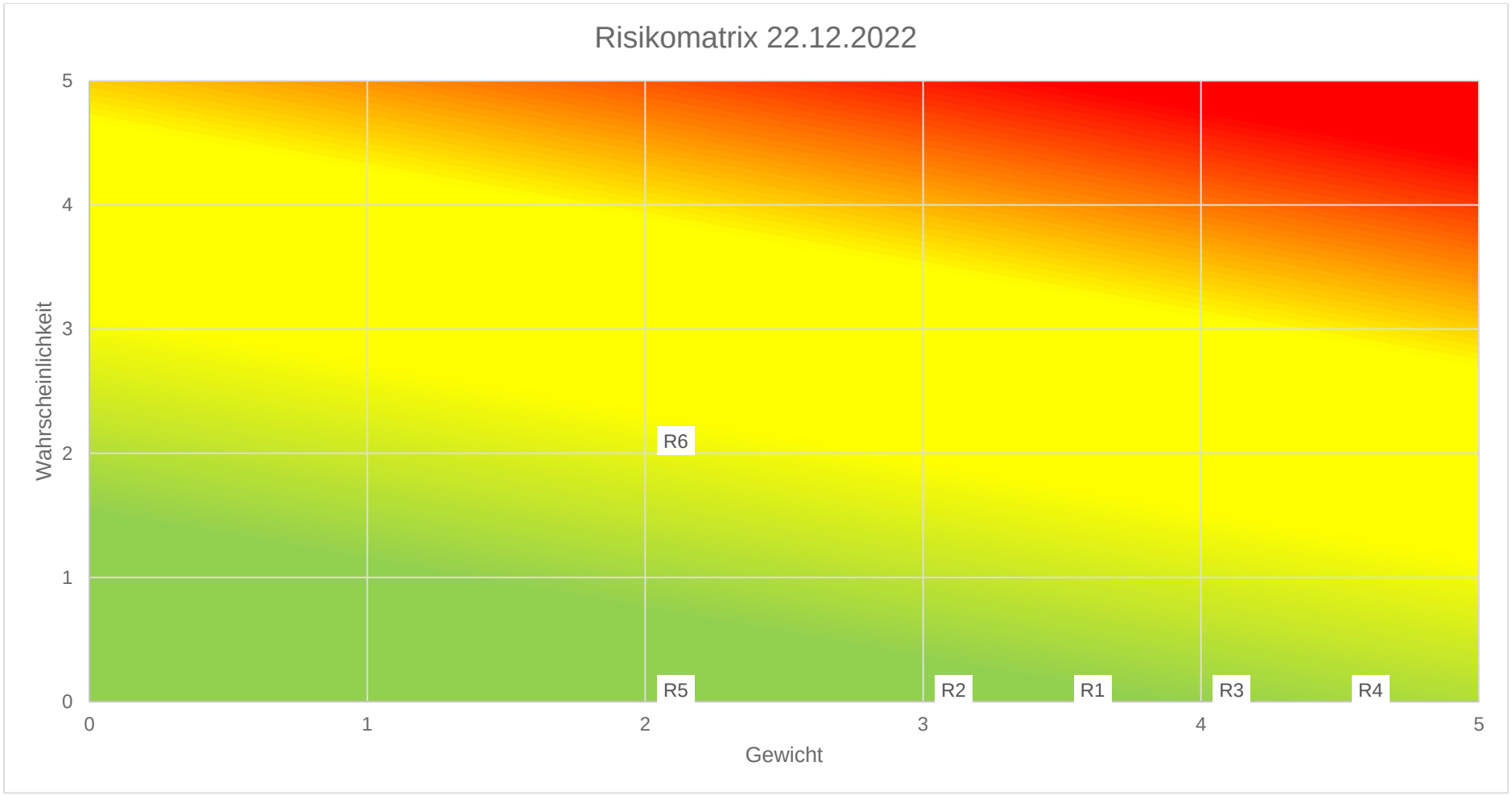
B.5.1 Risiko Auswertung vom 03.10.2022

Nr	Titel	Beschreibung	Wahrscheinlichkeit	Gewicht	Vorbeugung	Verhalten beim Eintreten	Bemerkung
R1	Java & Framework	Das Backend mit Java oder vorgegebenen Frameworks nimmt länger in Anspruch als geplant.	0	3.5	Vertiefung in Java und Frameworks	Absprache mit Betreuer, Backend auf bekannten Technologien erstellen	Framework konnte auf .Net 6 festgelegt werden
R2	Stand bestehende Applikation	Die bestehende Angular-Applikation ist in einem Zustand bei der die Weiterentwicklung mehr Zeit bedarf.	0	3	Absprachen mit Projektpartner	Absprache mit Betreuer, Unabhängige Entwicklung	Die erweiterung der bestehnder Applikation und alle Integrationen werden von der SFS Group übernommen
R3	Komplexität Cassandra DB	Als mögliche Datenbank Anbieter wurden MariaDB und Cassandra vorausgesetzt. Von den beiden ist Cassandra sehr komplex in der Anwendung.	0	4	Einarbeitung in Cassandra	Wechsel der Datenbank	In Absprache mit der SFS Group wurde festgelegt, dass die Vorteile von Cassandra keinen Mehrwert für diese Anwendung mitbringen.
R4	Unerwartete Änderungen seitens SFS Group	Abbruch des Arbeit seitens der SFS. Negatives Feedback von den Usability Tests oder von der Evaluation.	1	4.5	Zusammenarbeit fördern und ständiges Feedback fordern	Abklärung zur weiteren Durchführung mit Betreuer	-
R5	Anforderungsanalyse zögert sich hinaus	Das Abklären der Anforderungen dauert länger da SFS immer neue Anpassungs- und Featurewünsche hat.	0	2	Stetigen Austausch bis zum Abschluss des Meilensteins betreiben	Abgrenzung der Features für den Rahmen der Arbeit in Absprache mit dem Betreuer	-


Risikomatrix 03.10.2022
Wahrscheinlichkeit
Gewicht
0
1
2
3
4
5
R6
R5
R2
R1
R3
R4

B.5.2 Risiko Auswertung vom 22.12.2022

Nr	Titel	Beschreibung	Wahrscheinlichkeit	Gewicht	Vorbeugung	Verhalten beim Eintreten	Bemerkung
R1	Java & Framework	Das Backend mit Java oder vorgegebenen Frameworks nimmt länger in Anspruch als geplant.	0	3.5	Vertiefung in Java und Frameworks	Absprache mit Betreuer, Backend auf bekannten Technologien erstellen	Framework konnte auf .Net 6 festgelegt werden
R2	Stand bestehende Applikation	Die bestehende Angular-Applikation ist in einem Zustand bei der die Weiterentwicklung mehr Zeit bedarf.	0	3	Absprachen mit Projektpartner	Absprache mit Betreuer, Unabhängige Entwicklung	Die erweiterung der bestehnder Applikation und alle Integrationen werden von der SFS Group übernommen
R3	Komplexität Cassandra DB	Als mögliche Datenbank Anbieter wurden MariaDB und Cassandra vorausgesetzt. Von den beiden ist Cassandra sehr komplex in der Anwendung.	0	4	Einarbeitung in Cassandra	Wechsel der Datenbank	In Absprache mit der SFS Group wurde festgelegt, dass die Vorteile von Cassandra keinen Mehrwert für diese Anwendung mitbringen.
R4	Unerwartete Änderungen seitens SFS Group	Abbruch des Arbeit seitens der SFS. Negatives Feedback von den Usability Tests oder von der Evaluation.	0	4.5	Zusammenarbeit fördern und ständiges Feedback fordern	Abklärung zur weiteren Durchführung mit Betreuer	-
R5	Anforderungsanalyse zögert sich hinaus	Das Abklären der Anforderungen dauert länger da SFS immer neue Anpassungs- und Featurewünsche hat.	0	2	Stetigen Austausch bis zum Abschluss des Meilensteins betreiben	Abgrenzung der Features für den Rahmen der Arbeit in Absprache mit dem Betreuer	-
R6	GitLab Pipeline	Bei der Infrastruktur der SFS gab es vereinzelte Ausfälle, welche gewisse Arbeiten verhindern	2	2	Keine	SFS kontaktieren	Eine 3 tätige Störung trat auf, aber wurde dann von der SFS behoben. Da die Störung folge einer Migration war, wird ein weitere Ausfall als unwahrscheinlich betrachet.


Risikomatrix 22.12.2022
Wahrscheinlichkeit
Gewicht
0
1
2
3
4
5
R6
R5
R2
R1
R3
R4

C Projektplan

Projektplan vom 01.10.2022:

	SOLL-Aufwand in Stunden	Termine Meilensteine	SW 1 KW 38	SW 2 KW 39	SW 3 KW 40	SW 4 KW 41	SW 5 KW 42	SW 6 KW 43	SW 7 KW 44	SW 8 KW 45	SW 9 KW 46	SW 10 KW 47	SW 11 KW 48	SW 12 KW 49	SW 13 KW 50	SW 14 KW 51	(SW15) KW52	(SW16) KW1	(SW17) KW2
Anfoderungsanalyse	80																		
Nicht-Funktionale Anforderungen	10																		
Technology Constraints	6																		
Funktionale Anforderungen	12																		
Domain Model	8																		
User Research	20																		
Austausch mit SFS	20																		
Meilenstein Anforderungsanalyse	76	11.10.2022				★													
Entwurf	100																		
MVP definieren	10																		
Paper Prototype	24																		
Datenbank planen & erstellen	8																		
Analyse sinnvoller Libraries und Frameworks	12																		
Analyse Persona UCD	12																		
GitRepo und CI aufsetzen	8																		
Architektur-Prototyp	24																		
Meilenstein Entwurf	98	01.11.2022							★										
Implementation und Test	280																		
Backend Entwickeln	48																		
Frontend Entwickeln	120																		
Usability Tests	20	06.12.2022																	
Evaluation & Nachbesserung	52																		
Reserve	40																		
Meilenstein Implementation	0	29.11.2022											★						
Meilenstein Optimierungen (nach Evaluation)	280	20.12.2022														★			
Dokumentation	260																		
Projektplan erstellen & pflegen	8																		
Beschreibung der Ausgangssituation & Anforderungsanalyse	20																		
Dokumentation des Entwurfs	12																		
Fundierte Quellenrecherche	40																		
Dokumentation der Implementation	40																		
Developer und User-Dokumentation	8																		
Technische Dokumentation	40																		
Plakat erstellen	16																		
Dokumentation gegenlesen & verbessern	40																		
AVT relevante Abgaben (abstract, summary, usw.)	20																		
Reserve	16																		
Meilenstein Abgabe	260	13.01.2023																	★
Projektende																			
Gesamtstunden geplant	**720**																		

Projektplan vom 22.12.2022:

	SOLL-Aufwand in Stunden	Termine Meilensteine	SW 1 KW 38	SW 2 KW 39	SW 3 KW 40	SW 4 KW 41	SW 5 KW 42	SW 6 KW 43	SW 7 KW 44	SW 8 KW 45	SW 9 KW 46	SW 10 KW 47	SW 11 KW 48	SW 12 KW 49	SW 13 KW 50	SW 14 KW 51	(SW15) KW52	(SW16) KW1	(SW17) KW2
Anfoderungsanalyse	80																		
Nicht-Funktionale Anforderungen	10																		
Technology Constraints	6																		
Funktionale Anforderungen	12																		
Domain Model	8																		
User Research	20																		
Austausch mit SFS	20																		
Meilenstein Anforderungsanalyse	76	18.10.2022					★												
Entwurf	100																		
Paper Prototype	30																		
Analyse sinnvoller Libraries und Frameworks	12																		
Analyse Persona UCD	16																		
GitRepo und CI aufsetzen	8																		
Architektur-Prototyp	24																		
Reserve	10																		
Meilenstein Entwurf	100	01.11.2022							★										
Implementation und Test	280																		
Backend Entwickeln	68																		
Frontend Entwickeln	100																		
Usability Tests	20	06.12.2022																	
Evaluation & Nachbesserung	52																		
Reserve	40																		
Meilenstein Implementation	0	29.11.2022											★						
Meilenstein Optimierungen (nach Evaluation)	280	20.12.2022														★			
Dokumentation	260																		
Projektplan erstellen & pflegen	8																		
Beschreibung der Ausgangssituation & Anforderungsanalyse	20																		
Dokumentation des Entwurfs	12																		
Fundierte Quellenrecherche	40																		
Dokumentation der Implementation	40																		
Zwischenpräsentation	12	08.11.2022																	
Developer und User-Dokumentation	8																		
Technische Dokumentation	40																		
Plakat erstellen	16																		
Dokumentation gegenlesen & verbessern	40																		
AVT relevante Abgaben (abstract, summary, usw.)	16																		
Reserve	8																		
Meilenstein Abgabe	260	13.01.2023																	★
Projektende																			
Gesamtstunden geplant	**720**																		

D Projektmonitoring

D.1 Burndown-Diagramm

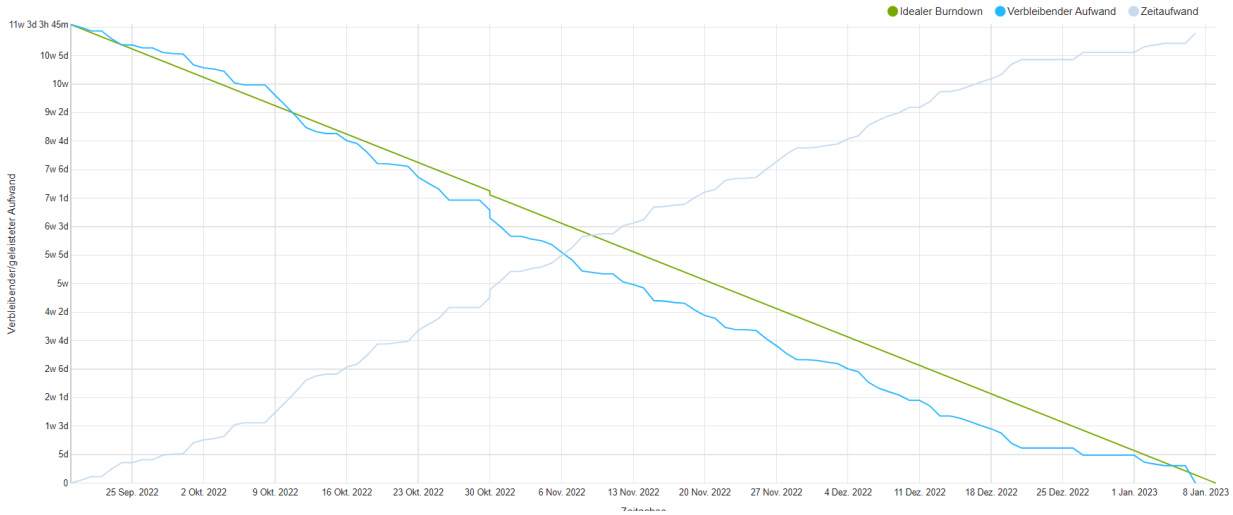


Abbildung 36: Burndown-Diagramm
Quelle: Screenshot aus YouTrack

D.2 Aufwandverteilung

Benutzer	Zeitschätzung	Zeitaufwand	Analyse	Anforderungen	Design & Architektur	Dokumentation	Entwicklung	Projektmanagment	Review & Testing
Gesamtzeit	**753h** 15m	**643h** 45m	**63h** 45m	**22h** 45m	**39h** 45m	**169h** 15m	**121h** 15m	**175h** 00m	**52h** 00m
Abdullah Almaz		330h 15m	33h 15m	12h 30m	19h 30m	78h 45m	54h 15m	97h 00m	35h 00m
Ursin Zimmermann		313h 30m	30h 30m	10h 15m	20h 15m	90h 30m	67h 00m	78h 00m	17h 00m

Abbildung 37: Aufwandverteilung
Quelle: Screenshot aus YouTrack

D.3 Code-Metriken

In den folgenden Kapiteln sind die Metriken des Source-Codes aufgeführt. Die Werte zeigen die Anzahl Zeilen ohne Kommentare und ohne Leerzeilen.

D.3.1 Frontend

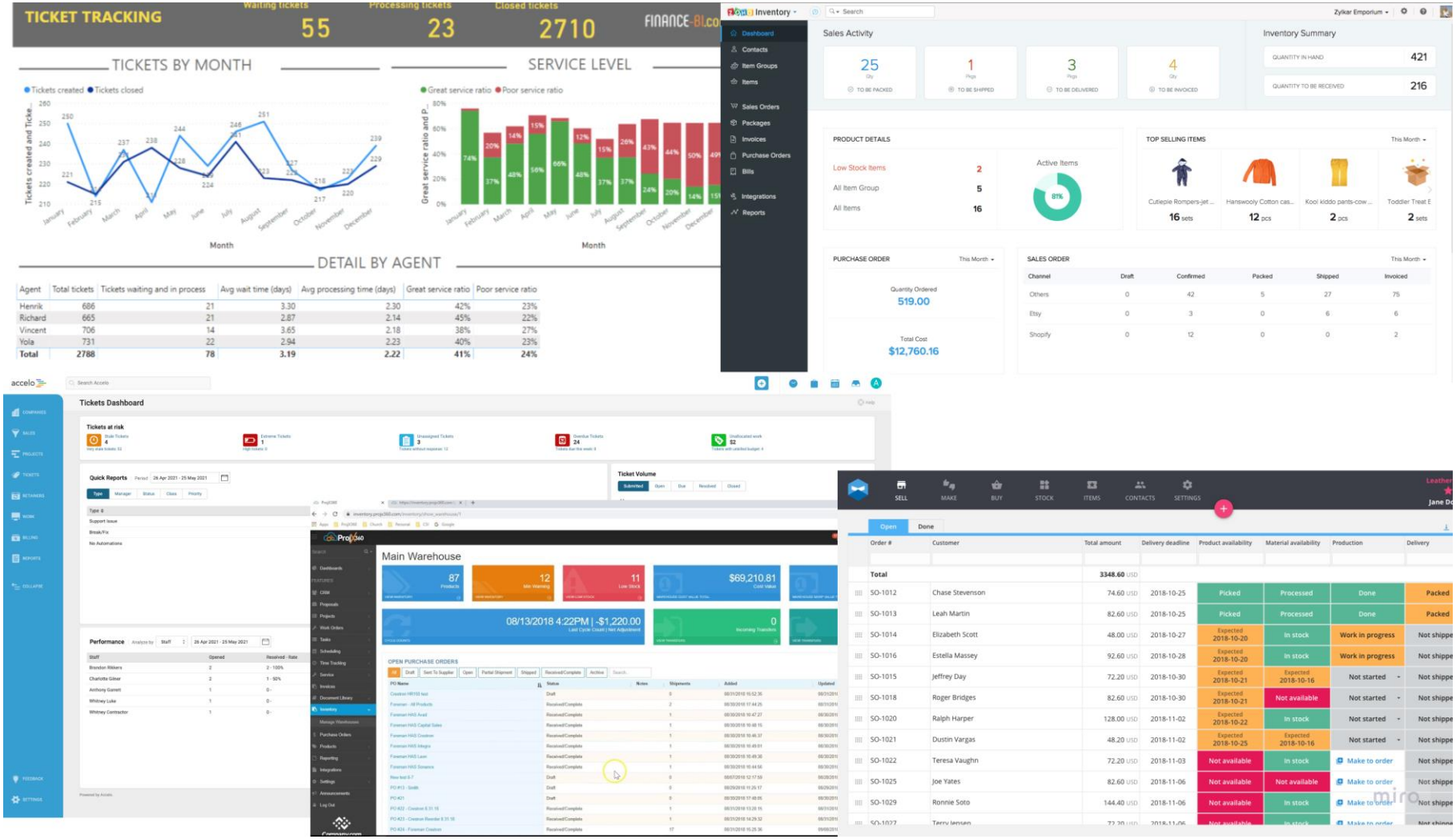
	Gesamt	Davon Tests	Davon Fremdleistung oder adaptiert
TypeScript	3’205	1’063	4
SCSS	361	0	0
HTML	519	0	0

D.3.2 Backend

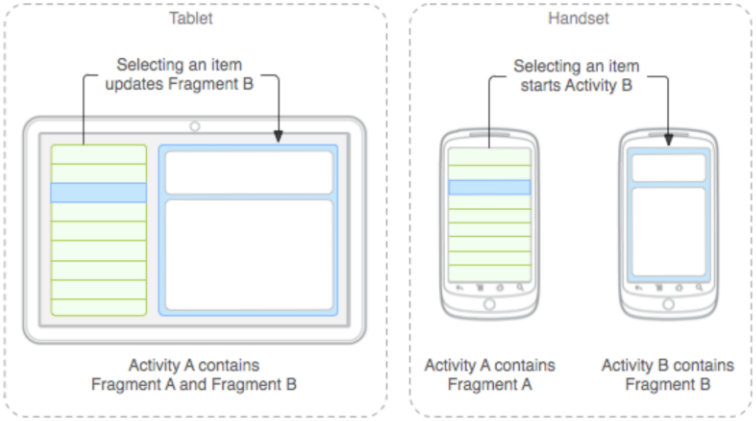
	Gesamt	Davon Tests	Davon Fremd-leistung oder adaptiert	Davon generiert
C#	3’689	1’048	20	1’838

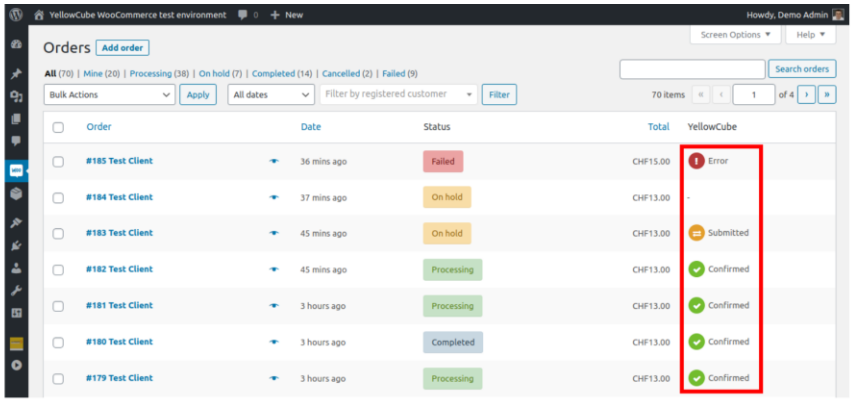
E Moodboards

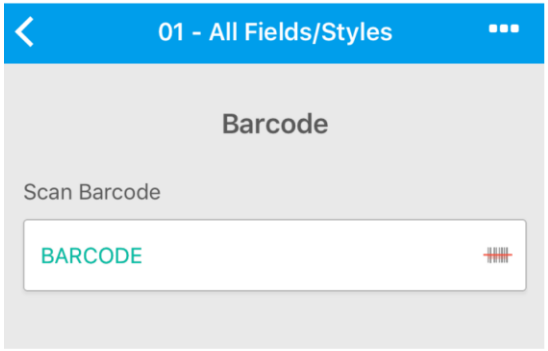
Moodboard fürs Dashboard:

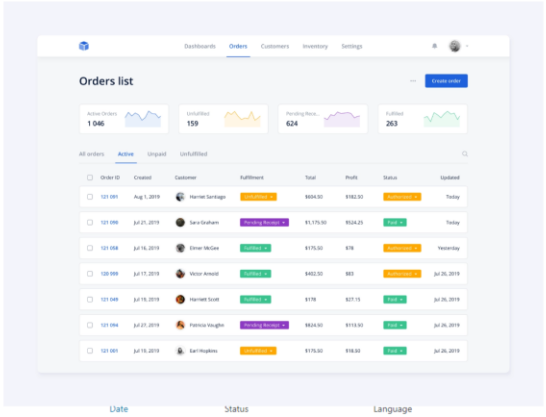


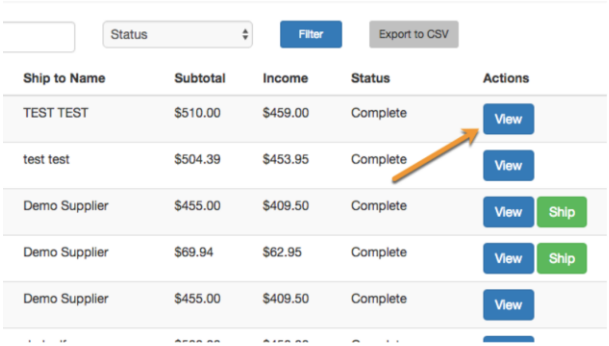
Moodboard für Staplerfahrer und Lagermitarbeiter

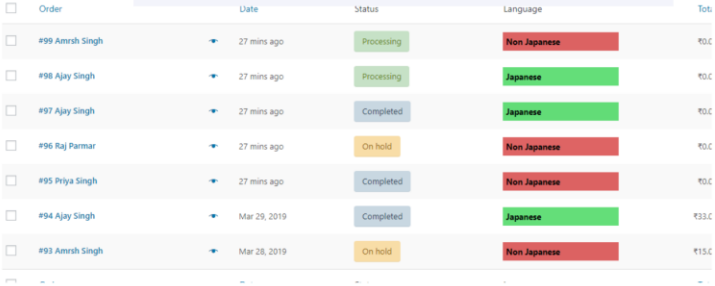












E.1 Alle Bildquellen der Moodboards:

https://www.accelo.com/assets/UI-Screenshots/Tickets-Dashboard.png

https://finance-bi.com/blog/wp-content/uploads/2019/11/Tickets-tracking-final.png

https://www.gocanvas.com/content/images/file-uploads/Barcode3-3.PNG

https://www.zoho.com/inventory/help/images/getting-started/dashboard.png

https://i.ytimg.com/vi/xsJAfS9kVG4/maxresdefault.jpg

https://comparecamp.com/media/uploads/2019/11/Katana-dashboard.png

https://i.stack.imgur.com/DKBMq.png

https://i.stack.imgur.com/cpmuX.png

https://cdn.dribbble.com/users/1169587/screenshots/6938860/media/764ac059a63f3823eaf2f2cb87466e69.png?compress=1&resize=1600x1200&vertical=top

https://d33v4339jhl8k0.cloudfront.net/docs/assets/558f9e89e4b01a224b42f278/images/58ac3f04dd8c8e56bfa7df05/file-mtqxEy1Vup.png

https://swisspost-yellowcube.github.io/wooyellowcube-docs/assets/order_list_v2_marker.png

F Wireframes

F.1 Entwurf der Dashboard-Ansicht

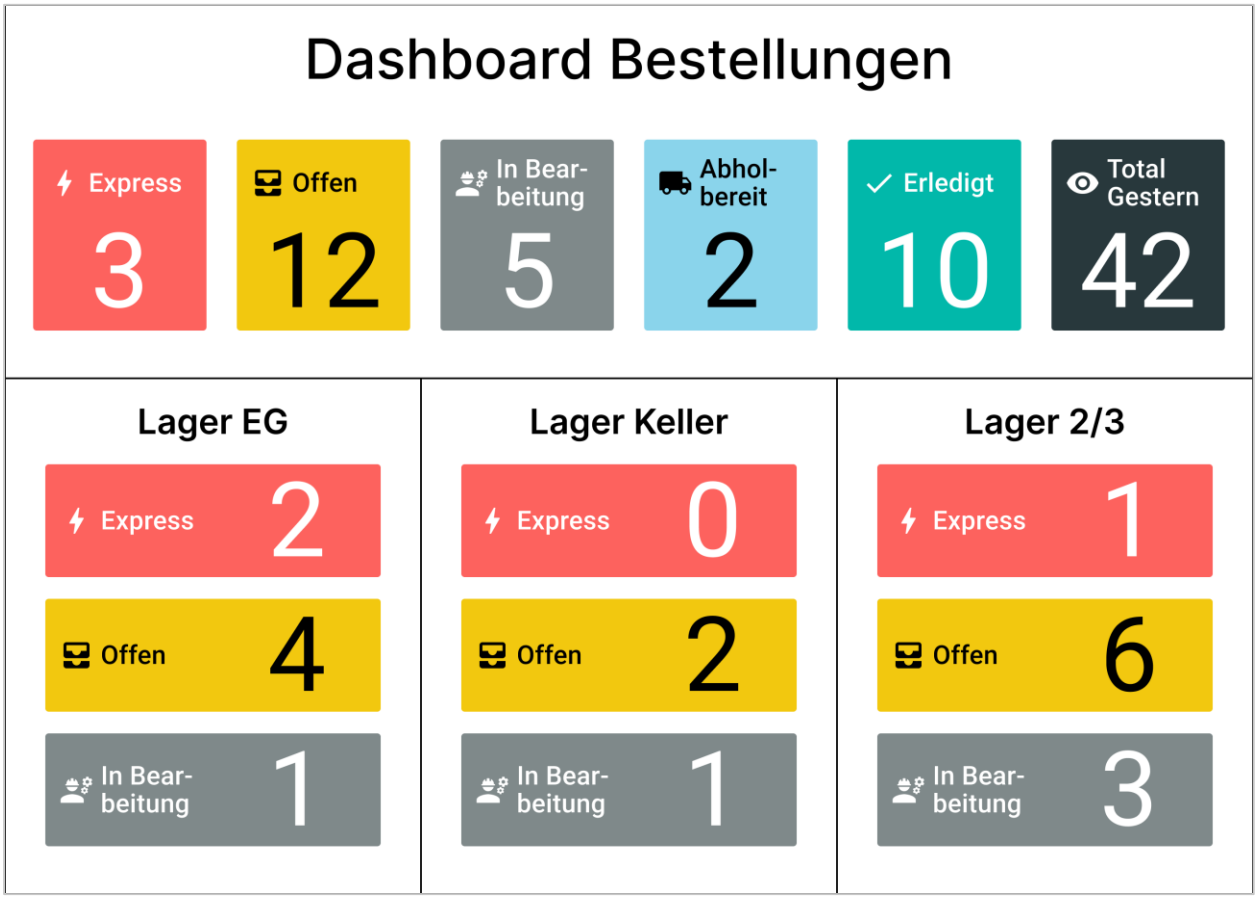


Abbildung 38: Wireframe Dashboard
Quelle: Erstellt mit Figma

F.2 Entwürfe der Lagermitarbeiter-Ansicht

Abbildung 39: Wireframe 1 Lagermitarbeiter
Quelle: Erstellt mit Figma

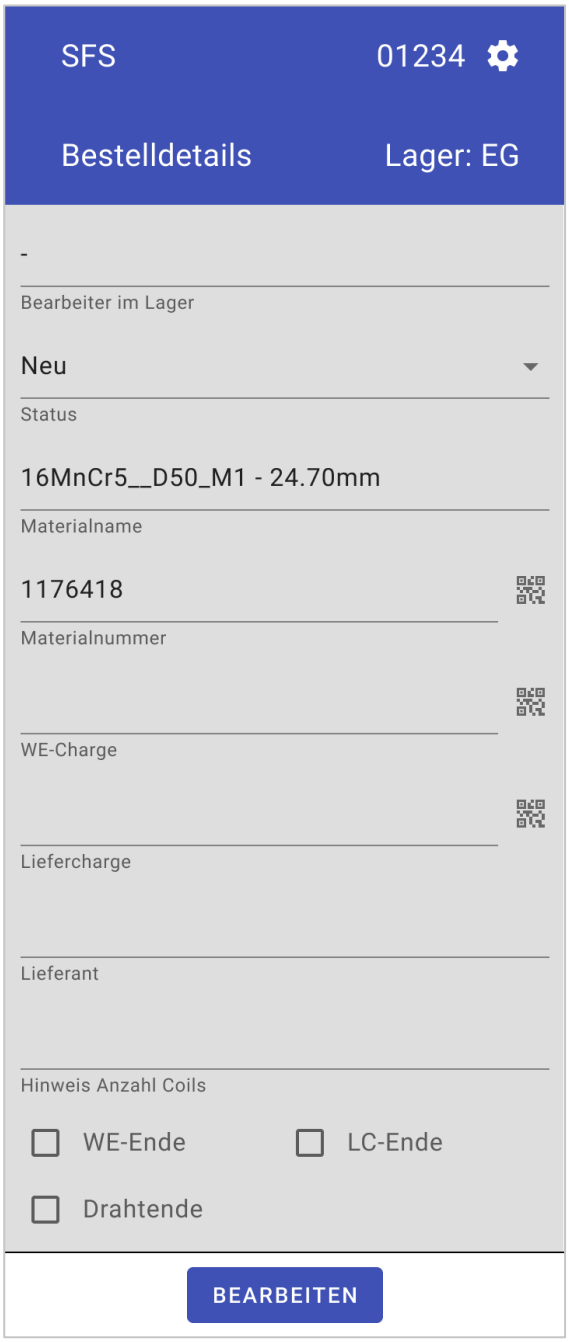


Abbildung 40: Wireframe 2 Lagermitarbeiter
Quelle: Erstellt mit Figma

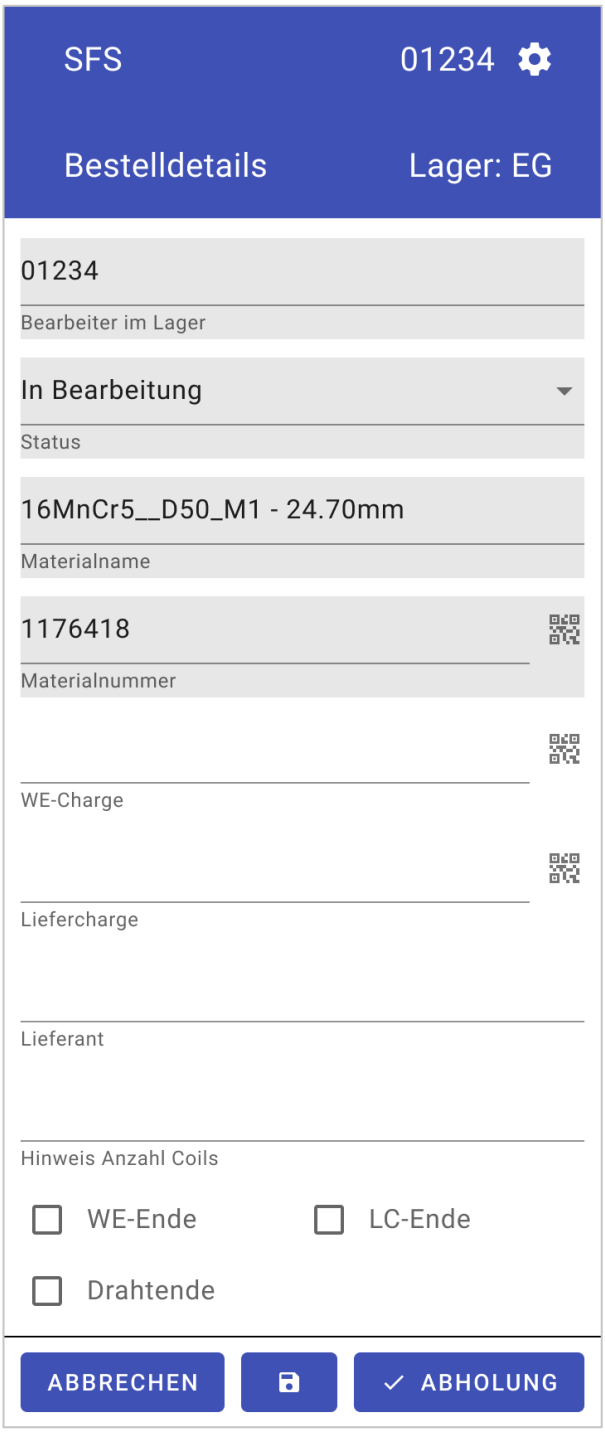


Abbildung 41: Wireframe 3 Lagermitarbeiter
Quelle: Erstellt mit Figma

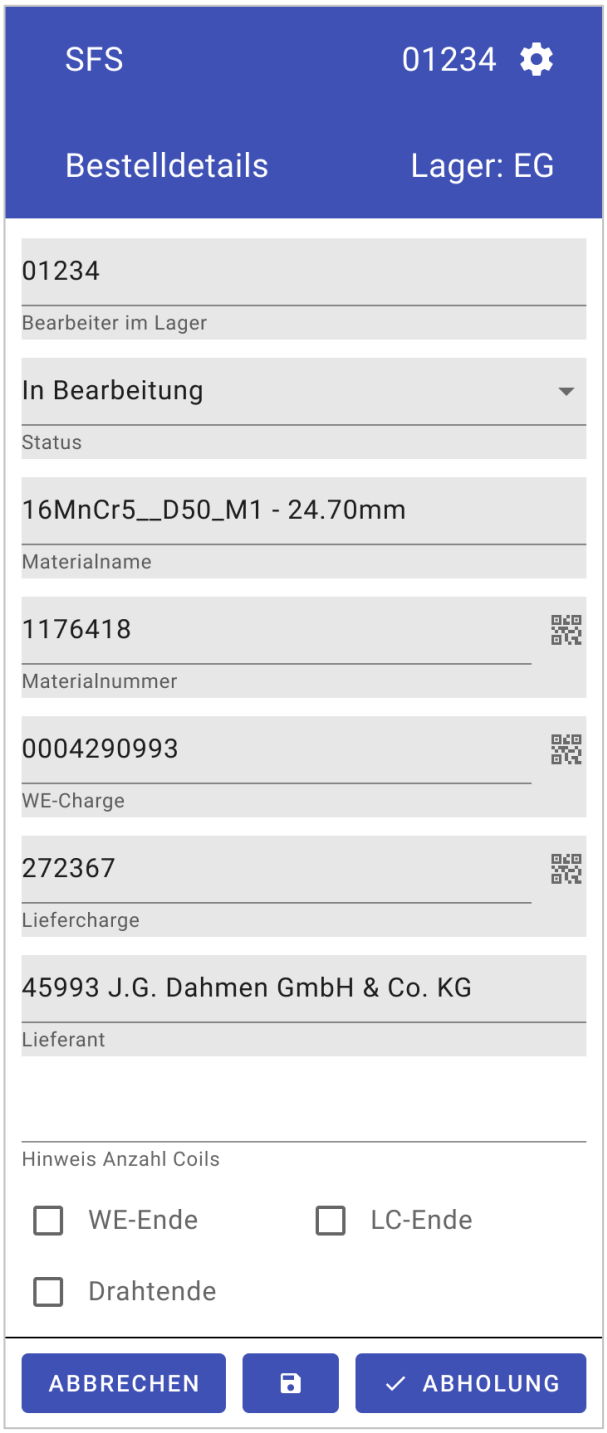


Abbildung 42: Wireframe 4 Lagermitarbeiter
Quelle: Erstellt mit Figma

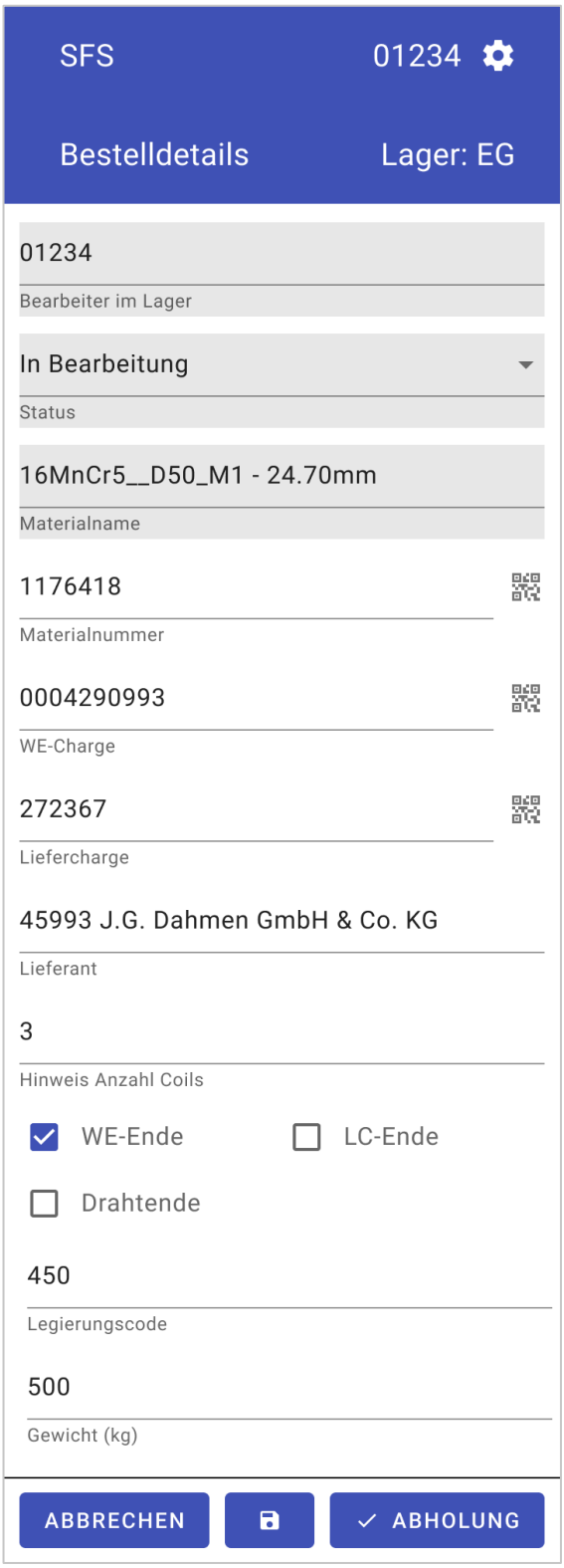


Abbildung 43: Wireframe 5 Lagermitarbeiter
Quelle: Erstellt mit Figma

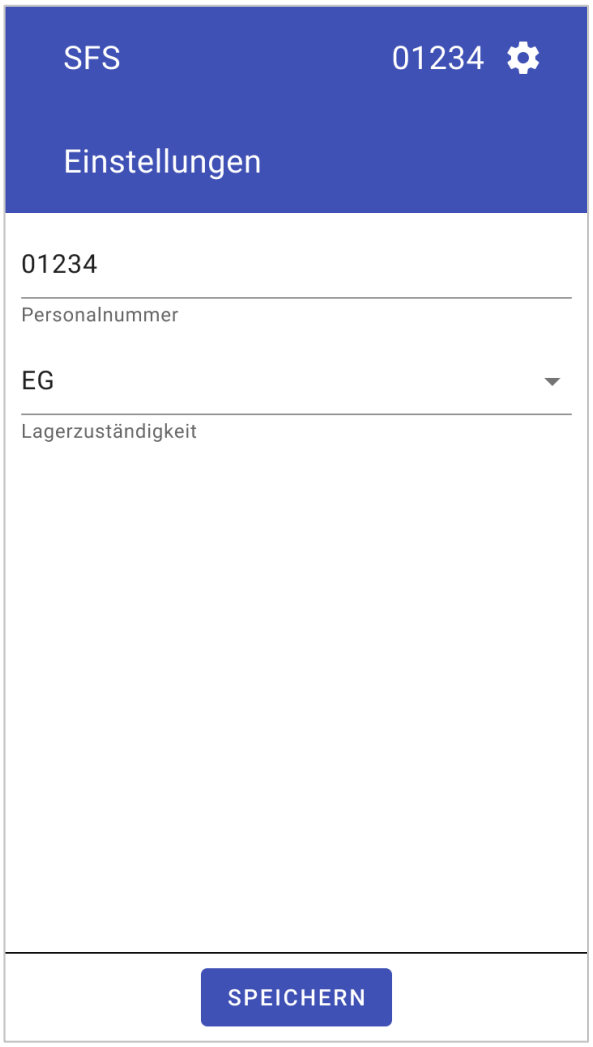


Abbildung 44: Wireframe 6 Lagermitarbeiter
Quelle: Erstellt mit Figma

F.3 Entwurf der Staplerfahrer-Ansicht

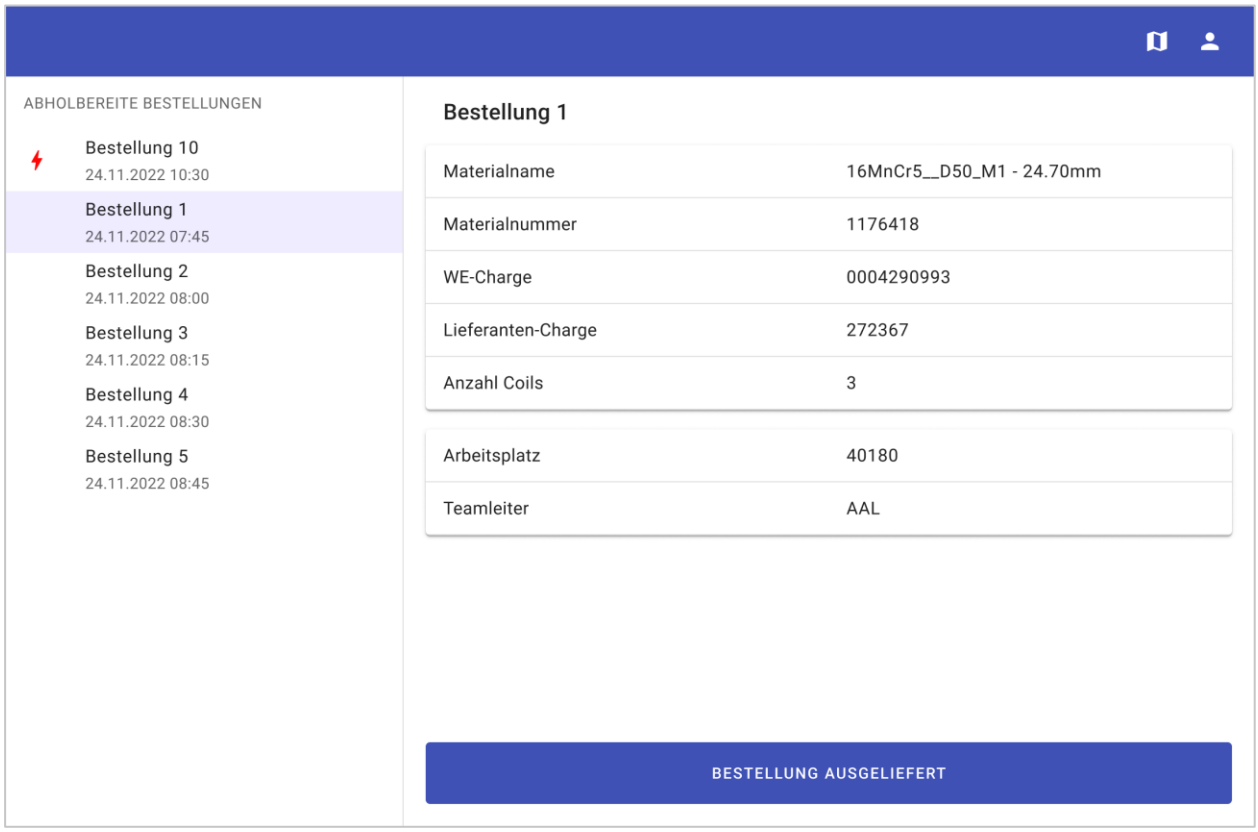


Abbildung 45: Wireframe Staplerfahrer
Quelle: Erstellt mit Figma

G Usability-Tests

G.1 Szenario 1

Sie sind heute für das Keller Lager verantwortlich.

G.1.1 Aufgabe 1

Prüfen Sie wie viele Bestellungen in ihrem Verantwortungsbereich offen sind und arbeiten sie die dringendste Bestellung davon ab.

G.1.2 Aufgabe 2

Sie sind gerade dabei Ihre Kaffee Pause zu machen als man Ihnen mitteilt, dass ein Mitarbeiter leider krankheitsbedingt nachhause musste. Daher übernehmen Sie zusätzlich noch das Lager 2 / 3.

Nach der Pause kommen Sie zurück in das Lager und wollen nun weiterarbeiten. Prüfen Sie wie viele Bestellungen in ihrem Verantwortungsbereich offen sind und arbeiten sie die dringendste Bestellung davon ab.

Beim Auslagern des Materials fällt Ihnen auf, dass es sich um ein Drahtende handelt bei dem bestellten Draht.

G.1.3 Aufgabe 3

Sie beginnen die nächstwichtige Bestellung zu bearbeiten. Sie bekommen während dem einen dringenden privaten Anruf und speichern daher den aktuellen Stand der Bestellung, ohne ihn abzuschliessen.

Nach dem Telefon wollen sie zurück an die Arbeit als ein anderer Mitarbeiter ihnen mitteilt das vor einigen Minuten ein Mitarbeiter aus der Produktion angerufen habe und er die Bestellung löschen wollte da er diese aus Versehen erfasst habe. Löschen Sie daher die Bestellung.

G.2 Szenario 2

Sie sind heute für das Ausliefern der Bestellungen verantwortlich. Und arbeiten dabei mit dem Stapler mit der Identifikation «Stapler 4».

G.2.1 Aufgabe 1

Prüfen Sie wie viele Bestellungen abholbereit sind und liefern sie die dringendste Bestellung davon aus.

G.3 Umfrage

G.3.1 Frage 1

Denken Sie die Applikation wird den Bestellprozess und den Arbeitsalltag erleichtern? (Beispiel und Begründungen)

G.3.2 Frage 2

Was war während der Nutzung unklar? Was kann an der Applikation noch verbessert werden?

G.3.3 Frage 3

Gibt es Funktionalitäten, welche in der Applikation noch fehlen oder verbessert werden müssen?

G.4 Wissensziele

Im Folgenden werden die Wissensziele hinter den einzelnen Aufgaben und Szenarien dokumentiert. Diese werden den Testern nicht ausgehändigt und dienen zum Nachvollziehen der Aufgabenstellung.

Da einige Ziele nicht quantifizierbar sind, werden die Tester dazu aufgefordert laut zu denken, damit anhand ihrer Gedankengänge die Erreichung der Ziele so gut wie möglich gemessen werden kann.

Szenario	Aufgabe	Wissensziele
1	**1**	• Wissen die Benutzer, wo sie sich informieren können über aktuelle offene Bestellungen? (Dashboard und Mobile) • Werden die Einstellungen auf dem Mobile korrekt interpretiert und erfasst? • Kann die Bestellung wie gewohnt bearbeitet werden? • Ist ersichtlich welche Bestellung am dringendsten ist? • Sind alle Informationen der Bestellung dargestellt? Braucht es mehr Informationen oder fehlen welche?
1	**2**	• Findet man die manuelle Änderung der Einstellungen? • Weiss der Benutzer, dass er die Einstellungen anpassen muss, wenn er das Lager wechseln will, oder mehrere Lager einsehen will? • Findet sich der Benutzer auf der Listenansicht zurecht? • Ist der angepasste Enden-Prozess verständlich oder wirft er Fragen auf?
1	**3**	• Kann der Benutzer nachvollziehen, was beim Speichern passiert und wohin die Applikation wechselt? • Findet man nach dem Speichern die Bestellung wieder? (Filter=Meine) • Findet man die Löschfunktion hinter dem erweiterten Menu?
2	**1**	• Sind alle relevanten Informationen zur Lieferung übersichtlich dargestellt und gruppiert? Fehlen Informationen? • Ist das Konzept mit den Filtern auch hier nützlich oder eher verwirren? • Kann eine Bestellung gemäss dem bisherigen Wissensstand des Nutzers abgearbeitet werden? • Sind alle Informationen der Bestellung dargestellt? Braucht es mehr Informationen oder fehlen welche?

H Benutzeranleitung

H.1 Dashboard

Das Dashboard dient dazu einen möglichst schnellen Überblick über die Gesamtsituation zu erlangen.

Der Bildschirm ist aufgeteilt in vier Bereiche. Im **orange** Markierten Bereich wird angezeigt wie viele Bestellungen sich in welchem Zustand befinden.

In den **rot** markierten Bereichen wird der Zustand der Bestellungen eines spezifischen Lagers angezeigt.

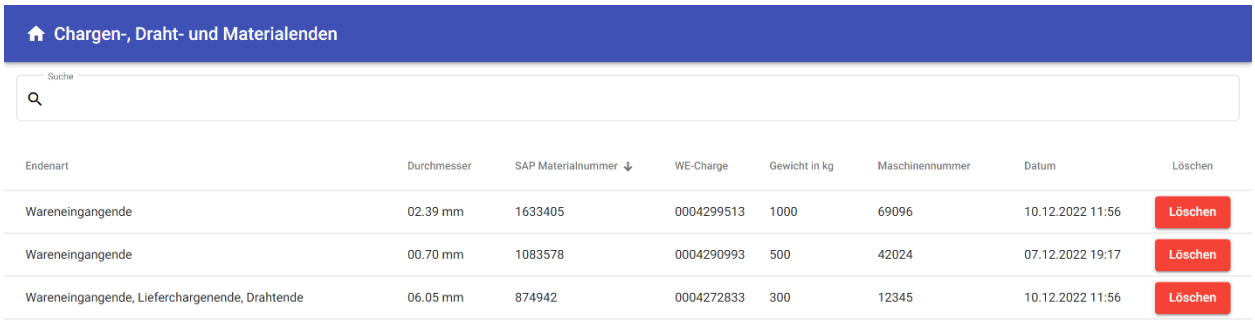
Im **blau** markierten Bereich wird angezeigt, wann die Daten des Dashboards das letzte Mal aktualisiert wurden.

Hier die Erklärungen für die jeweiligen Zustände:

1	Express	Die Anzahl an offenen Express Bestellungen im jeweiligen Lager
2	Offen	Die Anzahl an offenen Bestellungen im jeweiligen Lager Express Bestellungen sind hier nicht mit einbezogen
3	Bearbeitung	Die Anzahl an Bestellungen welche gerade in Bearbeitung sind
4	Abholung	Die Anzahl an Bestellungen welche abholbereit sind für den Staplerfahrer
5	Erledigt	Die Anzahl ausgelieferter Bestellungen am heutigen Tag
6	Gestern	Die Anzahl ausgelieferter Bestellungen vom Tag zuvor.

H.2 Chargen-, Draht- und Materialenden

Die Chargen-, Draht- und Materialenden Ansicht zeigt alle Bestellungen an, bei welchen Enden beteiligt waren.



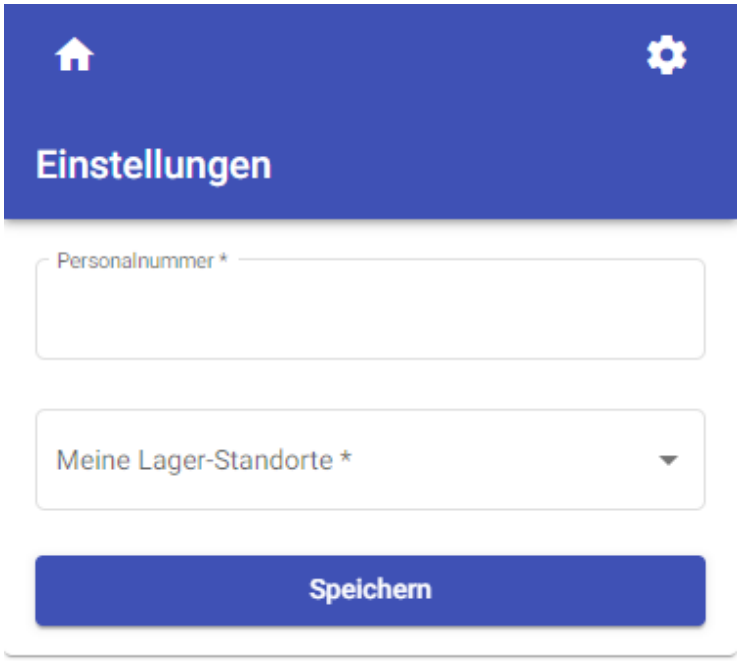
Einträge in der Liste können mit der **Löschen** Schaltfläche gelöscht werden.

Im **Suche** Feld kann nach allen Werten bis auf das Datum gesucht werden. (Das Datum ist für die Sortierung optimiert und wird anders dargestellt, als es eigentlich in den Daten vorhanden ist)

Alle Felder können sortiert werden in dem auf die jeweiligen Überschriften gedrückt wird, danach wird neben der Überschrift ein Pfeil angezeigt, um anzuzeigen wie das Feld sortiert ist.

H.3 Lager

H.3.1 Einstellungen



Beim ersten Öffnen der Lager Ansicht, wird man direkt auf die **Einstellungen** weiteregeleitet. Die Einstellungen können aber auch von überall aufgerufen in dem man das **Zahnrad** oben rechts anklickt.

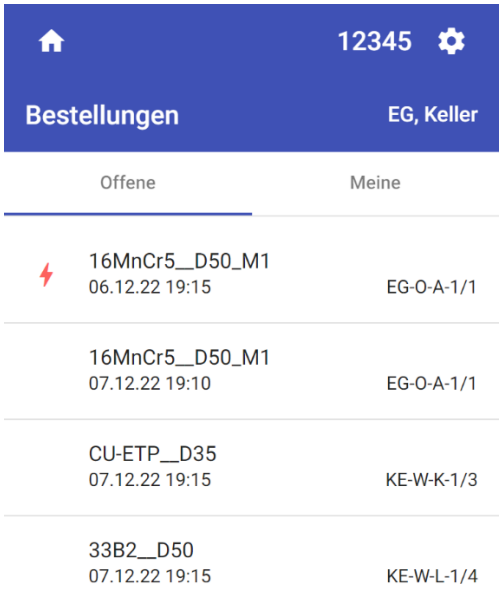
Im Feld **Personalnummer** trägt man seine Personalnummer ein.

Im Feld **Meine Lager-Standorte** kann man eins oder mehrere Lager auswählen.
Mit dieser Auswahl bestimmt man welche Bestellungen einem angezeigt werden, wenn man also nur das Lager Keller auswählt, werden nur die Bestellungen des Kellers angezeigt.
Falls ich mehrere Lager ausgewählt habe, werden mir alle Bestellungen dieses Lagers angezeigt Das Auswahlfenster schliesst sich erst nach einem Klick ausserhalb des Auswahlbereichs.

Mit **Speichern** kann man die getätigten Einstellungen speichern und wird weitergeleitet auf die Bestellungen Ansicht.

Sobald die Einstellungen gespeichert wurden, werden in der Kopfzeile der Applikation die getätigten Einstellungen angezeigt.

H.3.2 Bestellungen Ansicht



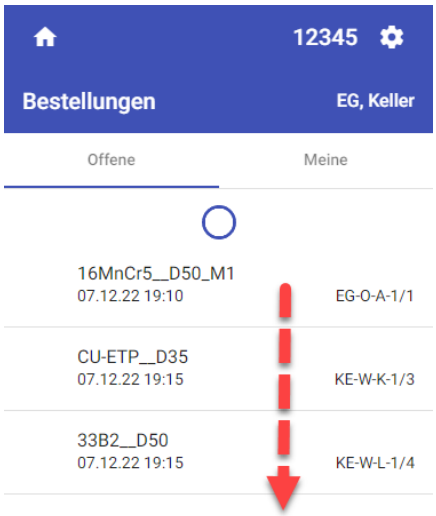
In der Bestellungen Ansicht werden alle Bestellungen angezeigt der ausgewählten Lager.
Diese sind sortiert nach Dringlichkeit und Erstelldatum.
Durch einen Klick auf eine Bestellung kann diese geöffnet werden.

Anhand des **roten Blitzes** erkennt man, ob es sich um eine Expressbestellung handelt.

Mit dem **Lagerstandort** einer Bestellung erkennt man, welche Bestellung zu welchem Lager gehört. Zusätzlich werden weitere wichtige Informationen einer Bestellung angezeigt, das sind Materialname und Erstelldatum.

Im Tab **Offene** sind alle offenen Bestellungen der selektierten Lager sichtbar.

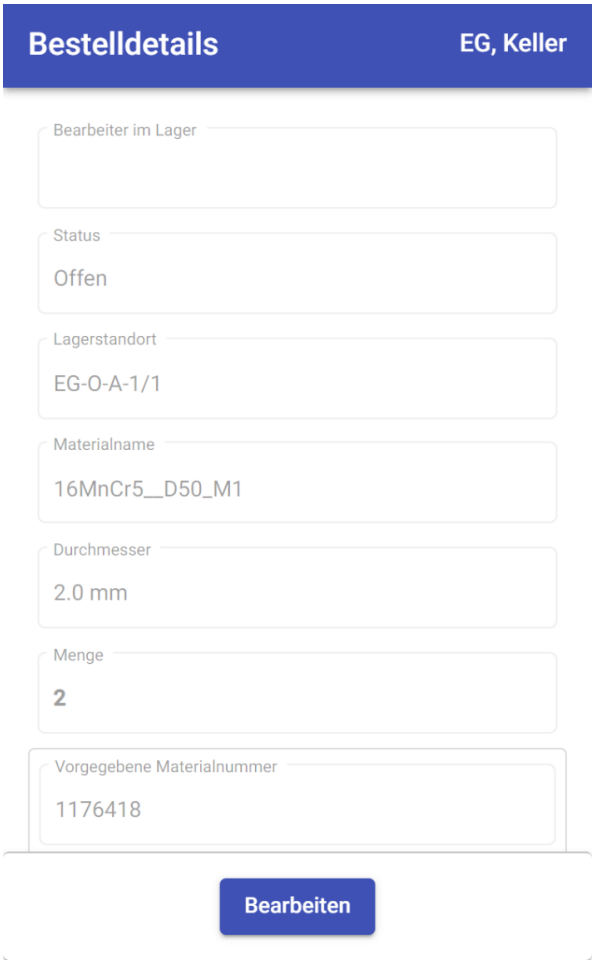
Im Tab **Meine** werden alle Bestellungen angezeigt welche mir zugewiesen sind.



Die Bestellungen werden automatisch im Minuten-Takt **aktualisiert**.

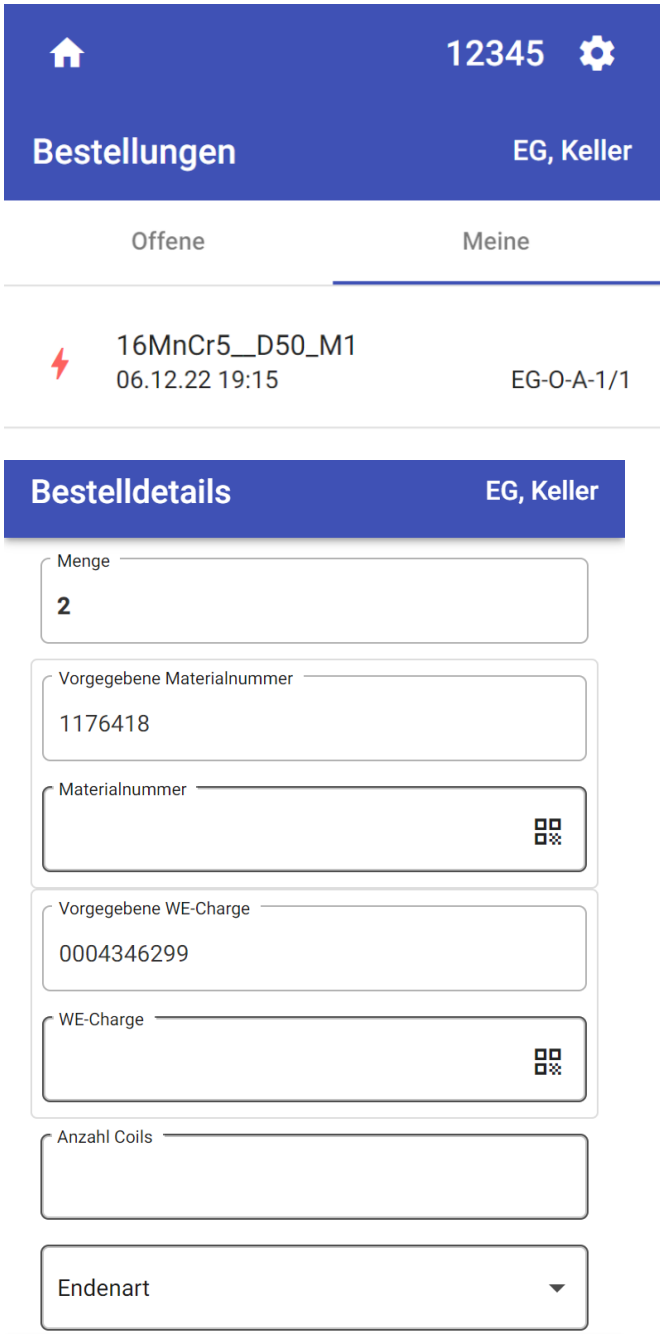
Falls man die Bestellungen manuell aktualisieren will, kann dies mit einem nach unten ziehen in der Bestellungen Liste gemacht werden.

H.3.3 Bestelldetails Ansicht



Beim Öffnen einer Offenen Bestellung wird diese zuerst schreibgeschützt angezeigt, in dieser kann man alle Informationen einer Bestellung betrachten aber nicht bearbeiten.

Mit der **Bearbeiten** Schaltfläche kann man eine Bestellung einem selbst zuweisen und beginnen diese zu bearbeiten.
Diese Bestellung ist ab diesem Zeitpunkt nur noch im Tab **Meine** der Bestellungen-Ansicht sichtbar, bis sie Abholbereit ist, die Bearbeitung abgebrochen wird oder sie gelöscht wird.



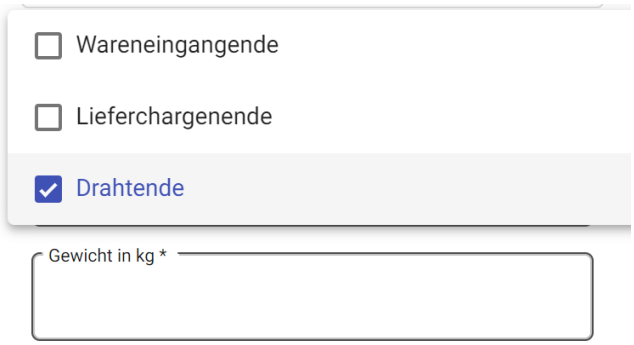
Alle Felder, welche bearbeitet werden können, sind mit einem dicken und dunkleren Rahmen gekennzeichnet.

Im Feld **Menge** wird angezeigt was für eine Menge der Presser bestellt hat. Sobald diese grösser als 1 ist, wird diese fett markiert.

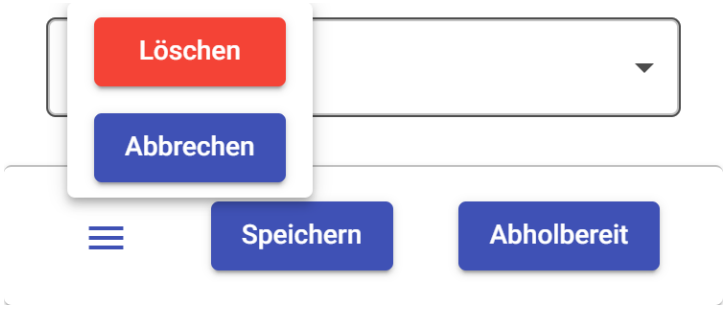
Das gewünschte Material des Pressers wird im Feld **Vorgegebene Materialnummer** angezeigt.
Falls der Presser eine gewünschte Wareneingangcharge angegeben hat, wird diese im Feld **Vorgegebene WE-Charge** angezeigt.
Dies dient zur Orientation welches Material benötigt wird.

Zum Erfassen der **Materialnummer** und **WE-Charge** kann in das jeweilige Feld geklickt werden und dann der Barcode gescannt werden. Dieser sollte dann direkt im Feld eingetragen werden. Falls im Feld **Materialnummer** die korrekte Materialnummer eingetragen wurde, springt der Fokus automatisch weiter zum Feld WE-Charge, damit gleich beide Scans hintereinander ausgeführt werden können.

Im Feld **Anzahl Coils** wird festgehalten wie viele Draht Coils für die Bestellung vorbereitet wurden. Dies dient später dem Staplerfahrer als Hilfestellung.



Falls es sich bei einem Material um ein Ende handelt, kann dies in der Auswahl **Endenart** erfasst werden. Das Auswahlfenster schliesst sich erst nach einem Klick ausserhalb des Auswahlbereichs.
Wenn eine Bestellung ein Ende hat, muss zusätzlich das **Gewicht** angegeben werden.



Mit der **Löschen** Schaltfläche kann eine Bestellung gelöscht werden.

Mit der **Abbrechen** Schaltfläche kann die Bearbeitung der Bestellung abgebrochen werden bzw. wieder auf offen gesetzt werden und ist danach wieder für anderen Mitarbeiter des Lagers sichtbar.

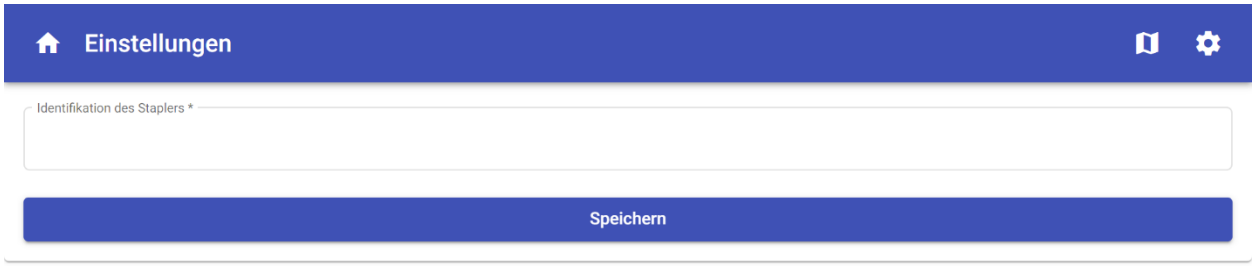
Mit der **Speichern** Schaltfläche kann der aktuelle Stand einer Bestellung zwischengespeichert werden. Dies ist dann nützlich, wenn man mehrere Bestellungen aufs mal bearbeiten will, wie zum Beispiel im Keller. So können Bestellungen, die im Lift bereit sind, gespeichert werden und man kann weitere Bestellungen abarbeiten.

Mit der **Abholbereit** Schaltfläche wird der Status einer Bestellung auf abholbereit gestellt und ist ab diesem Zeitpunkt nur noch in der Staplerfahrer Ansicht sichtbar.

Bestellungen können nur gespeichert oder als abholbereit markiert werden, wenn alle Pflichtfelder korrekt ausgefüllt sind.

H.4 Staplerfahrer

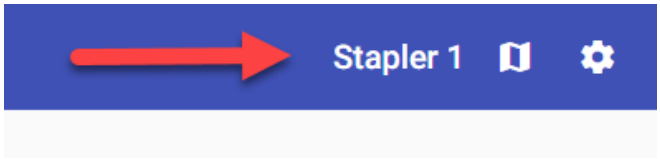
H.4.1 Einstellungen



Beim ersten Öffnen der Staplerfahrer Ansicht, wird man direkt auf die **Einstellungen** weiteregeleitet. Die Einstellungen können aber auch von überall aufgerufen in dem man das **Zahnrad** oben rechts anklickt.

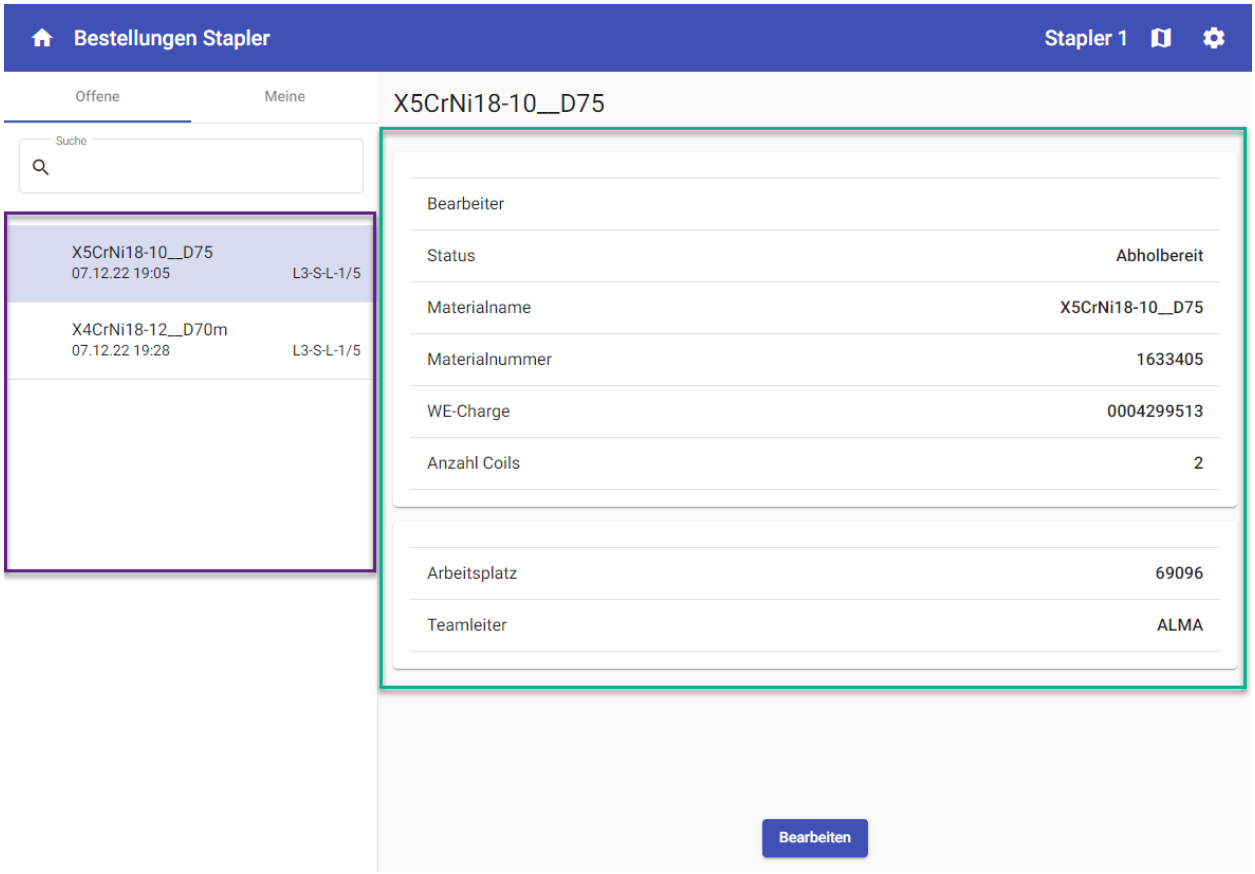
Im Feld **Identifikation des Staplers** trägt man die Identifikation des Staplers ein.

Mit **Speichern** kann man die getätigten Einstellungen speichern und wird weitergeleitet auf die Bestellungen Ansicht.



Sobald die Einstellungen gespeichert wurden, werden in der Kopfzeile der Applikation die getätigten Einstellungen angezeigt.

H.4.2 Bestellungen Ansicht



In der Bestellungen Ansicht werde alle abholbereiten Bestellungen angezeigt.

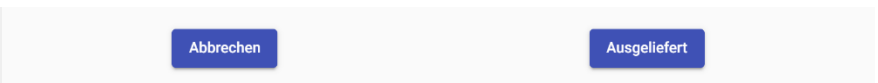
Im Tab **Offene** sind alle abholbereiten Bestellungen sichtbar.

Im Tab **Meine** werden alle Bestellungen angezeigt die von diesem Stapler auf in Bearbeiten markiert sind.

Im **grünen** Bereich werden alle wichtigen Informationen der selektierten Bestellung angezeigt.

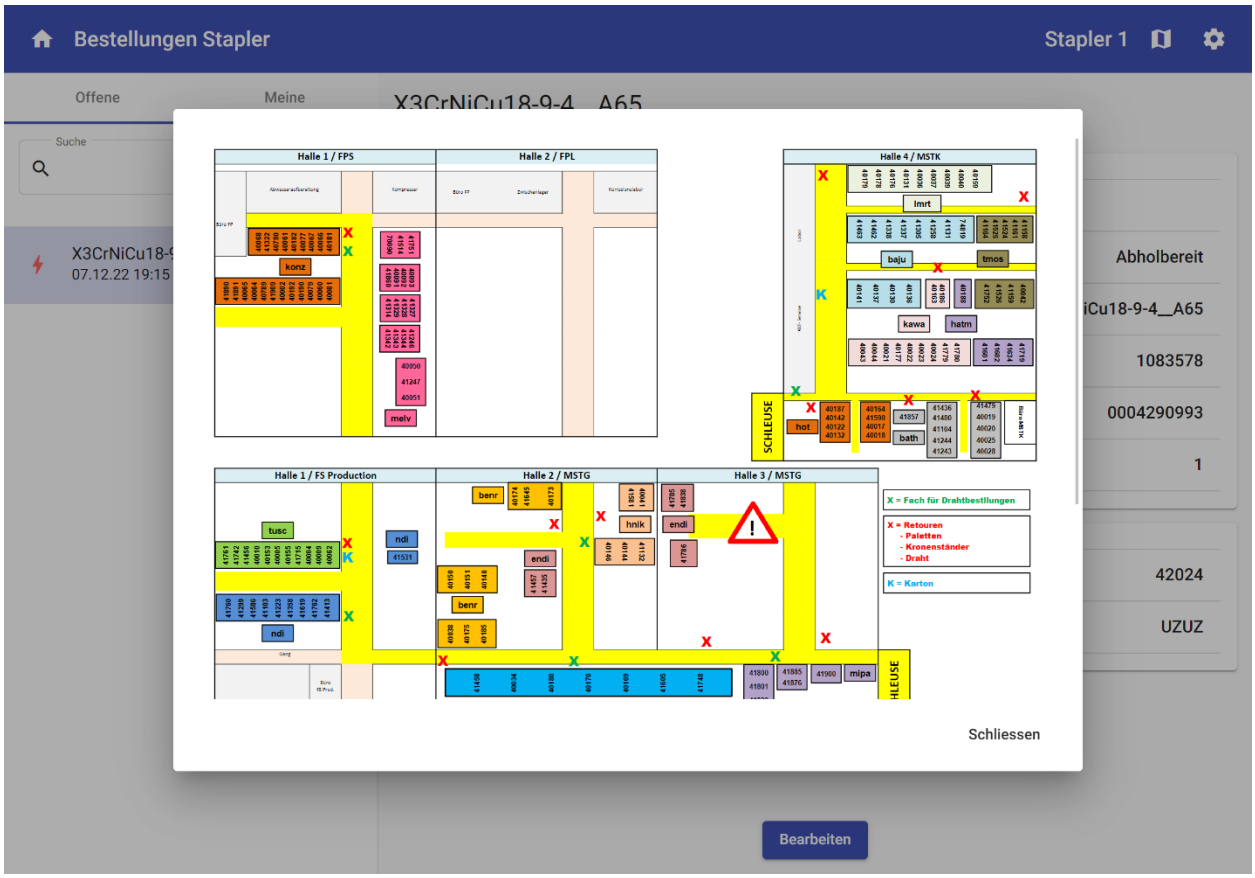
Mit dem **Suche** Feld kann nach Bestellungen durch Eingabe einer Materialnummer oder WE-Charge gesucht werden.

Mit der **Bearbeiten** Schaltfläche wird eine Bestellung auf in Bearbeitung markiert, diese Bestellung ist ab diesem Zeitpunkt für andere Stapler nicht mehr sichtbar.



Mit der **Abbrechen** Schaltfläche kann eine Bestellung, welche in Bearbeitung ist, abgebrochen werden und ist dann wieder für alle anderen Stapler sichtbar.

Mit der **Ausgeliefert** Schaltfläche wird markiert das eine Bestellung ausgeliefert wurde, diese Bestellung verschwindet dann aus der Staplerfahrer Ansicht.



Wenn auf das **Karten** Symbol in der Kopfzeile geklickt wird, wird die Maschinenstandort Karte angezeigt. Dies soll helfen beim Suchen des Standorts der Maschine, an welche man die Bestellung liefern muss.

I Zeitprotokolle

Tickets, gruppiert nach Autor der Arbeit			Sep. 2022						
			19 Mo.	20 Di.	21 Mi.	22 Do.	23 Fr.	24 Sa.	25 So.
Gesamtzeit		**32h 45m**	**4h 00m**	**5h 00m**	**00m**	**11h 00m**	**8h 45m**	**00m**	**4h 00m**
Abdullah Almaz		**19h 30m**	4h 00m	3h 00m	00m	6h 00m	4h 15m	00m	2h 15m
BA-1	Administration	2h 00m		2h 00m					
BA-2	Projektplan erstellen	3h 00m				3h 00m			
BA-3	Risikoanalyse vorbereiten	45m							45m
BA-4	BA Dokument vorbereiten	1h 00m					1h 00m		
BA-5	Fragen an Kunden vorbereiten	45m					45m		
BA-7	Reports Zeiterfassung erstellen	1h 30m				1h 30m			
BA-8	Requirementvorstellungen dokumentieren	1h 00m					1h 00m		
BA-12	Einlesen BA Leitfaden und BA Tipps	1h 00m		1h 00m					
BA-15	Aufgabestellung lesen und Feedback geben	1h 00m					1h 00m		
BA-17	Themen für fundierte Quellenrecherche bestim...	30m							30m
BA-18	Austausch	2h 00m				1h 30m	30m		
BA-19	Aufsetzen und Konfiguration Youtrack	4h 00m	4h 00m						
BA-34	Tickets Folgesprint erstellen	1h 00m							1h 00m
Ursin Zimmermann		**13h 15m**	00m	2h 00m	00m	5h 00m	4h 30m	00m	1h 45m
BA-1	Administration	2h 00m		2h 00m					
BA-2	Projektplan erstellen	15m					15m		
BA-3	Risikoanalyse vorbereiten	15m							15m
BA-4	BA Dokument vorbereiten	3h 00m				1h 00m	2h 00m		
BA-6	Meetingprotokoll erstellen	1h 30m				1h 30m			
BA-9	Requirementvorstellungen dokumentieren	1h 00m							1h 00m
BA-13	Einlesen BA Leitfaden und BA Tipps	1h 00m				1h 00m			
BA-14	Fragen an Kunden vorbereiten	45m					45m		
BA-16	Aufgabestellung lesen und Feedback geben	1h 00m					1h 00m		
BA-17	Themen für fundierte Quellenrecherche bestim...	30m							30m
BA-18	Austausch	2h 00m				1h 30m	30m		

Tickets, gruppiert nach Autor der Arbeit			Sep. 2022						
			26 Mo.	27 Di.	28 Mi.	29 Do.	30 Fr.	1 Sa.	2 So.
Gesamtzeit		**29h 45m**	**00m**	**6h 30m**	**1h 30m**	**45m**	**15h 00m**	**4h 15m**	**1h 45m**
Abdullah Almaz		**15h 00m**	00m	3h 30m	00m	00m	7h 30m	3h 15m	45m
BA-21	Themen und Bücher für fundierte Quellenrecher...	2h 30m		30m				2h 00m	
BA-23	Administration	2h 00m		1h 00m			1h 00m		
BA-24	Meeting Betreuer und Projektpartner	1h 00m		1h 00m					
BA-26	Nicht funktionale Anforderungen abklären und d...	3h 00m					3h 00m		
BA-27	Technology Constrains abklären und dokumenti...	1h 00m					1h 00m		
BA-30	User Research vorbereiten	2h 00m					1h 00m	1h 00m	
BA-31	Beschreibung Ausgangssituation	1h 45m		1h 00m					45m
BA-36	Termine für Usability Test und Evaluation definie...	30m					30m		
BA-37	Projektplan anpassen	45m					30m	15m	
BA-40	Oli & Frieder schreiben	30m					30m		
Ursin Zimmermann		**14h 45m**	00m	3h 00m	1h 30m	45m	7h 30m	1h 00m	1h 00m
BA-21	Themen und Bücher für fundierte Quellenrecher...	1h 00m		1h 00m					
BA-22	Meetingprotokoll erstellen	2h 00m			1h 30m	30m			
BA-23	Administration	2h 15m		1h 00m		15m	1h 00m		
BA-24	Meeting Betreuer und Projektpartner	1h 00m		1h 00m					
BA-26	Nicht funktionale Anforderungen abklären und d...	1h 00m					1h 00m		
BA-27	Technology Constrains abklären und dokumenti...	3h 00m					3h 00m		
BA-30	User Research vorbereiten	1h 30m					1h 00m	30m	
BA-31	Beschreibung Ausgangssituation	45m							45m
BA-36	Termine für Usability Test und Evaluation definie...	30m					30m		
BA-37	Projektplan anpassen	1h 00m					30m	30m	
BA-40	Oli & Frieder schreiben	30m					30m		
BA-45	Dokumente drucken für Dienstag	15m							15m

Okt. 2022

Tickets, gruppiert nach Autor der Arbeit			3 Mo.	4 Di.	5 Mi.	6 Do.	7 Fr.	8 Sa.	9 So.
Gesamtzeit		**51h 00m**	**3h 15m**	**16h 00m**	**3h 00m**	**00m**	**00m**	**14h 45m**	**14h 00m**
Abdullah Almaz		**26h 45m**	1h 30m	8h 00m	3h 00m	00m	00m	7h 00m	7h 15m
BA-28	Funktionale Anforderungen abklären und dokum...	3h 30m			2h 00m			1h 30m	
BA-29	Domain Model skizzieren und dokumentieren	3h 00m						3h 00m	
BA-38	Git Projekte aufsetzen & Rechte testen	15m						15m	
BA-42	User Research dokumentieren	2h 00m							2h 00m
BA-43	Administratives	2h 30m	1h 00m					1h 00m	30m
BA-44	Personas erstellen	1h 00m						1h 00m	
BA-46	Besuch SFS	8h 00m		8h 00m					
BA-47	Moodboard & Paper Prototype besprechen	3h 00m							3h 00m
BA-48	Meeting mit Frieder	30m	30m						
BA-50	Entscheidungsprotokoll Dienstag	15m						15m	
BA-51	Risikomatrix anpassen	1h 00m			1h 00m				
BA-52	Projektplan anpassen	15m							15m
BA-57	Anforderung schicken	1h 30m							1h 30m
Ursin Zimmermann		**24h 15m**	1h 45m	8h 00m	00m	00m	00m	7h 45m	6h 45m
BA-28	Funktionale Anforderungen abklären und dokum...	1h 00m						1h 00m	
BA-29	Domain Model skizzieren und dokumentieren	1h 00m						1h 00m	
BA-38	Git Projekte aufsetzen & Rechte testen	45m	45m						
BA-42	User Research dokumentieren	3h 00m							3h 00m
BA-43	Administratives	1h 30m						1h 00m	30m
BA-44	Personas erstellen	3h 00m						3h 00m	
BA-46	Besuch SFS	8h 00m		8h 00m					
BA-47	Moodboard & Paper Prototype besprechen	3h 00m							3h 00m
BA-48	Meeting mit Frieder	30m	30m						
BA-49	Entscheidungsprotokoll Montag	30m	30m						
BA-50	Entscheidungsprotokoll Dienstag	45m						45m	
BA-51	Risikomatrix anpassen	1h 00m						1h 00m	
BA-52	Projektplan anpassen	15m							15m

Tickets, gruppiert nach Autor der Arbeit			Okt. 2022						
			10 Mo.	11 Di.	12 Mi.	13 Do.	14 Fr.	15 Sa.	16 So.
Gesamtzeit		**53h 30m**	**14h 45m**	**16h 15m**	**6h 00m**	**2h 30m**	**00m**	**10h 00m**	**4h 00m**
Abdullah Almaz		**26h 00m**	5h 00m	8h 00m	6h 00m	1h 00m	00m	6h 00m	00m
BA-61	Analyse Sinnvoller Libraries Frontend	2h 00m		2h 00m					
BA-62	Architektur planen	1h 00m						1h 00m	
BA-63	Anpassungen Anforderungsanalyse	2h 00m		1h 00m	1h 00m				
BA-64	Dokumentation finalisieren von Einführung Bericht	1h 00m		1h 00m					
BA-65	Administration	2h 30m	30m			1h 00m		1h 00m	
BA-67	Server aufsetzen	3h 00m			3h 00m				
BA-53	Paper Prototype erstellen	3h 00m		3h 00m					
BA-54	Meeting mit Frieder	30m	30m						
BA-55	Meeting mit SFS	45m		45m					
BA-56	Entscheidungsprotokolle	15m		15m					
BA-59	Literatur Recherche	10h 00m	4h 00m		2h 00m			4h 00m	
Ursin Zimmermann		**27h 30m**	9h 45m	8h 15m	00m	1h 30m	00m	4h 00m	4h 00m
BA-62	Architektur planen	3h 00m						3h 00m	
BA-63	Anpassungen Anforderungsanalyse	2h 00m		1h 00m		1h 00m			
BA-65	Administration	2h 30m	30m	30m		30m		1h 00m	
BA-66	Moodboard dokumentieren	2h 00m		2h 00m					
BA-53	Paper Prototype erstellen	6h 00m	4h 00m	2h 00m					
BA-54	Meeting mit Frieder	30m	30m						
BA-55	Meeting mit SFS	45m		45m					
BA-56	Entscheidungsprotokolle	45m	45m						
BA-59	Literatur Recherche	8h 00m	4h 00m						4h 00m
BA-60	Analyse Sinnvoller Libraries Backend	2h 00m		2h 00m					

Tickets, gruppiert nach Autor der Arbeit			Okt. 2022 17 Mo.	18 Di.	19 Mi.	20 Do.	21 Fr.	22 Sa.	23 So.
Gesamtzeit		**56h 00m**	**12h 30m**	**15h 45m**	**15m**	**1h 45m**	**2h 00m**	**15h 15m**	**8h 30m**
Abdullah Almaz		**29h 30m**	5h 30m	7h 45m	15m	1h 45m	2h 00m	6h 30m	5h 45m
BA-68	Dokumentieren Design	3h 30m						3h 30m	
BA-69	Dokumentation Literatur Recherche	6h 00m		6h 00m					
BA-70	Administration	2h 15m	1h 00m	30m	15m			30m	
BA-71	Meeting mit Frieder	1h 15m		1h 15m					
BA-72	Entscheidungsprotokolle	15m						15m	
BA-73	Frontend vorbereiten	3h 30m	3h 30m						
BA-75	Anforderungen abschliessen	2h 00m						2h 00m	
BA-76	Architekturprototyp Backend	2h 00m							2h 00m
BA-77	Datenbank planen & anlegen	1h 00m	1h 00m						
BA-79	Design Refining	45m							45m
BA-80	Figma Prototyp anpassen anhand Feedback	4h 00m				1h 00m			3h 00m
BA-82	Doku für gegenlesen vorbereiten und schicken	2h 00m					2h 00m		
BA-83	Hardware Scanner Doku lesen	45m				45m			
BA-41	Backend vorbereiten	15m						15m	
Ursin Zimmermann		**26h 30m**	7h 00m	8h 00m	00m	00m	00m	8h 45m	2h 45m
BA-68	Dokumentieren Design	30m						30m	
BA-69	Dokumentation Literatur Recherche	6h 00m		6h 00m					
BA-70	Administration	2h 00m	1h 00m	30m				30m	
BA-71	Meeting mit Frieder	1h 15m		1h 15m					
BA-72	Entscheidungsprotokolle	45m						45m	
BA-73	Frontend vorbereiten	15m		15m					
BA-76	Architekturprototyp Backend	7h 00m	2h 00m					3h 00m	2h 00m
BA-77	Datenbank planen & anlegen	3h 00m	3h 00m						
BA-79	Design Refining	45m							45m
BA-80	Figma Prototyp anpassen anhand Feedback	1h 00m						1h 00m	
BA-83	Hardware Scanner Doku lesen	15m						15m	
BA-41	Backend vorbereiten	3h 45m	1h 00m					2h 45m	

Tickets, gruppiert nach Autor der Arbeit			Okt. 2022						
			24 Mo.	25 Di.	26 Mi.	27 Do.	28 Fr.	29 Sa.	30 So.
Gesamtzeit		**48h 30m**	**8h 00m**	**15h 30m**	**00m**	**00m**	**00m**	**14h 00m**	**11h 00m**
Abdullah Almaz		**24h 45m**	4h 15m	7h 45m	00m	00m	00m	8h 30m	4h 15m
BA-78	Architekturprototyp Frontend	3h 00m	3h 00m						
BA-85	Feedback von Review einbauen	9h 30m		1h 00m				6h 00m	2h 30m
BA-86	Meeting mit Frieder	1h 00m		1h 00m					
BA-87	Meeting mit SFS	1h 00m		1h 00m					
BA-88	Administration	3h 30m	1h 15m	1h 00m				30m	45m
BA-89	Entscheidungsprotokolle	15m		15m					
BA-90	Dokumentation Wireframes	2h 00m		2h 00m					
BA-93	Datenbank anpassen	30m		30m					
BA-94	Platzhalter Backendentwicklung	1h 00m		1h 00m					
BA-96	Literaturrecherche V2	3h 00m						2h 00m	1h 00m
Ursin Zimmermann		**23h 45m**	3h 45m	7h 45m	00m	00m	00m	5h 30m	6h 45m
BA-85	Feedback von Review einbauen	1h 30m							1h 30m
BA-86	Meeting mit Frieder	1h 00m		1h 00m					
BA-87	Meeting mit SFS	1h 00m		1h 00m					
BA-88	Administration	3h 30m	1h 15m	1h 00m				30m	45m
BA-89	Entscheidungsprotokolle	1h 15m		1h 15m					
BA-90	Dokumentation Wireframes	2h 00m		2h 00m					
BA-93	Datenbank anpassen	1h 00m	30m	30m					
BA-94	Platzhalter Backendentwicklung	3h 00m	2h 00m	1h 00m					
BA-96	Literaturrecherche V2	9h 30m						5h 00m	4h 30m

Okt. 2022								
Tickets, gruppiert nach Autor der Arbeit				31 Mo.	1 Di.	2 Mi.	3 Do.	4 Fr.	5 Sa.	6 So.
Gesamtzeit			**48h 30m**	**12h 15m**	**13h 30m**	**00m**	**4h 00m**	**2h 15m**	**5h 15m**	**11h 15m**
Abdullah Almaz			**22h 45m**	4h 15m	6h 45m	00m	4h 00m	1h 00m	00m	6h 45m
BA-84	Vorbereitungen Zwischenpräsi		4h 00m	1h 00m	2h 00m					1h 00m
BA-91	Risikomatrix anpassen		15m							15m
BA-95	Frontend Entwicklung		13h 00m	3h 00m	2h 00m		4h 00m	1h 00m		3h 00m
BA-97	Projektplan pflegen		15m		15m					
BA-99	Meeting mit Frieder		30m		30m					
BA-100	Dokumentation Todos abarbeiten		1h 00m							1h 00m
BA-102	Administration		2h 15m	15m	1h 00m					1h 00m
BA-103	Backend - REST Endpoints		1h 30m		1h 00m					30m
Ursin Zimmermann			**25h 45m**	8h 00m	6h 45m	00m	00m	1h 15m	5h 15m	4h 30m
BA-84	Vorbereitungen Zwischenpräsi		3h 00m		2h 00m					1h 00m
BA-95	Frontend Entwicklung		15m					15m		
BA-97	Projektplan pflegen		15m	15m						
BA-99	Meeting mit Frieder		30m		30m					
BA-100	Dokumentation Todos abarbeiten		2h 45m						15m	2h 30m
BA-101	Entscheidungsprotokoll		45m		45m					
BA-102	Administration		2h 15m	15m	1h 00m					1h 00m
BA-103	Backend - REST Endpoints		16h 00m	7h 30m	2h 30m			1h 00m	5h 00m	

Tickets, gruppiert nach Autor der Arbeit			Nov. 2022 7 Mo.	8 Di.	9 Mi.	10 Do.	11 Fr.	12 Sa.	13 So.
Gesamtzeit		**45h 00m**	**10h 30m**	**15h 30m**	**2h 00m**	**2h 00m**	**00m**	**11h 30m**	**3h 30m**
Abdullah Almaz		**24h 45m**	5h 15m	8h 30m	2h 00m	2h 00m	00m	5h 00m	2h 00m
BA-104	Administration	2h 30m	30m	1h 00m				1h 00m	
BA-105	Zwischenpräsentation	1h 30m		1h 30m					
BA-106	Dokumentation implementation	4h 00m	1h 00m					1h 00m	2h 00m
BA-107	Refactoring & Testing Backend	2h 00m		1h 00m				1h 00m	
BA-108	Frontend: Entwicklung Lager-Ansicht	14h 00m	3h 00m	5h 00m	2h 00m	2h 00m		2h 00m	
BA-109	Korrekturen Review für BA-103	30m	30m						
BA-110	Beispiel Daten Backend	15m	15m						
Ursin Zimmermann		**20h 15m**	5h 15m	7h 00m	00m	00m	00m	6h 30m	1h 30m
BA-104	Administration	2h 00m	30m	1h 00m				30m	
BA-105	Zwischenpräsentation	1h 30m		1h 30m					
BA-106	Dokumentation implementation	1h 30m							1h 30m
BA-107	Refactoring & Testing Backend	9h 00m		4h 00m				5h 00m	
BA-108	Frontend: Entwicklung Lager-Ansicht	1h 00m						1h 00m	
BA-109	Korrekturen Review für BA-103	1h 00m	1h 00m						
BA-110	Beispiel Daten Backend	3h 00m	3h 00m						
BA-111	Update Database - Diameter	1h 15m	45m	30m					

Tickets, gruppiert nach Autor der Arbeit			Nov. 2022 14 Mo.	15 Di.	16 Mi.	17 Do.	18 Fr.	19 Sa.	20 So.
Gesamtzeit		**43h 30m**	**4h 45m**	**17h 45m**	**45m**	**2h 00m**	**1h 00m**	**9h 30m**	**7h 45m**
Abdullah Almaz		**19h 45m**	30m	8h 45m	15m	2h 00m	00m	5h 15m	3h 00m
BA-112	Termine für Abgaben im Projektplan und Sprintf...	1h 00m		1h 00m					
BA-113	Readme Frontend	2h 00m		2h 00m					
BA-115	Frontend Delivery	7h 30m		3h 00m		2h 00m		2h 30m	
BA-116	Backend Tests & Refactoring	30m		30m					
BA-119	Administration	3h 15m	30m	1h 00m				45m	1h 00m
BA-120	Meeting Frieder	30m		30m					
BA-121	Entscheidungsprotokoll	15m			15m				
BA-122	Backend: InventoryEnding REST Endpoints	30m		30m					
BA-123	Adrian Video zukommen lassen	15m		15m					
BA-136	Frontend Ending	2h 00m							2h 00m
BA-137	Dockerfile Frontend für Produktion	2h 00m						2h 00m	
Ursin Zimmermann		**23h 45m**	4h 15m	9h 00m	30m	00m	1h 00m	4h 15m	4h 45m
BA-112	Termine für Abgaben im Projektplan und Sprintf...	1h 00m		1h 00m					
BA-113	Readme Frontend	15m		15m					
BA-114	Readme Backend	2h 15m	2h 00m	15m					
BA-115	Frontend Delivery	30m						30m	
BA-116	Backend Tests & Refactoring	6h 00m		6h 00m					
BA-117	Dokumentation Implementation	3h 30m	45m					1h 00m	1h 45m
BA-119	Administration	2h 15m	30m	1h 00m				45m	
BA-120	Meeting Frieder	30m		30m					
BA-121	Entscheidungsprotokoll	30m			30m				
BA-122	Backend: InventoryEnding REST Endpoints	1h 00m	1h 00m						
BA-136	Frontend Ending	3h 00m							3h 00m
BA-138	Dockerfile Backend für Produktion	3h 00m					1h 00m	2h 00m	

Tickets, gruppiert nach Autor der Arbeit				Nov. 2022						
				21 Mo.	22 Di.	23 Mi.	24 Do.	25 Fr.	26 Sa.	27 So.
Gesamtzeit			**42h 30m**	**3h 45m**	**12h 30m**	**3h 00m**	**15m**	**1h 00m**	**11h 30m**	**10h 30m**
Abdullah Almaz			**27h 45m**	3h 30m	7h 15m	3h 00m	00m	00m	7h 00m	7h 00m
BA-139	Adrian Map verlangen und einbauen		1h 30m	30m	1h 00m					
BA-140	Frontend: Scan Logik einbauen		3h 15m		3h 15m					
BA-141	Administration		1h 00m						1h 00m	
BA-142	Dokumentation		3h 15m		2h 00m					1h 15m
BA-143	Meeting Frieder		45m		45m					
BA-144	Meetingprotokoll erstellen		30m							30m
BA-145	SFS CI Pipeline anschauen		45m	30m	15m					
BA-146	Usability Tests Vorbereitungen		1h 30m							1h 30m
BA-148	Release Build abschliessen		1h 00m			1h 00m				
BA-150	Dockerfile Frontend für Produktion		2h 00m			2h 00m				
BA-152	Swipe Refresh		1h 30m	1h 30m						
BA-153	PWA Erstellen für Test erstellen		1h 30m	1h 00m						30m
BA-159	Testing Frontend		7h 00m						4h 00m	3h 00m
BA-161	Release auf dem Server deployen		2h 00m						2h 00m	
BA-164	Readme anpassen - Backend		15m							15m
Ursin Zimmermann			**14h 45m**	15m	5h 15m	00m	15m	1h 00m	4h 30m	3h 30m
BA-139	Adrian Map verlangen und einbauen		15m		15m					
BA-140	Frontend: Scan Logik einbauen		1h 30m		1h 30m					
BA-141	Administration		1h 15m						1h 15m	
BA-142	Dokumentation		15m							15m
BA-143	Meeting Frieder		45m		45m					
BA-144	Meetingprotokoll erstellen		30m					30m		
BA-145	SFS CI Pipeline anschauen		15m		15m					
BA-146	Usability Tests Vorbereitungen		2h 00m		1h 00m					1h 00m
BA-147	Testdaten für Usabilitytest		1h 15m		1h 15m					
BA-148	Release Build abschliessen		2h 15m				15m	30m	1h 30m	
BA-151	Dockerfile Backend für Produktion		1h 00m						1h 00m	
BA-152	Swipe Refresh		15m	15m						
BA-153	PWA Erstellen für Test erstellen		15m		15m					
BA-159	Testing Frontend		15m							15m
BA-164	Readme anpassen - Backend		45m						45m	
BA-165	Testing Backend		2h 00m							2h 00m

Nov. 2022

Tickets, gruppiert nach Autor der Arbeit			28 Mo.	29 Di.	30 Mi.	1 Do.	2 Fr.	3 Sa.	4 So.
Gesamtzeit		**32h 30m**	**11h 00m**	**8h 30m**	**00m**	**1h 00m**	**2h 30m**	**2h 00m**	**7h 30m**
Abdullah Almaz		**16h 45m**	3h 00m	8h 30m	00m	15m	30m	00m	4h 30m
BA-129	Usability Tests Vorbereitung	1h 30m		30m					1h 00m
BA-149	Testing & Optimierungen Platzhalter	1h 00m		1h 00m					
BA-155	Administration	2h 15m	1h 00m			15m			1h 00m
BA-156	Meeting mit Frieder	30m		30m					
BA-157	Entscheidungsprotokolle	30m		30m					
BA-158	Sammelticket für Fehler & Optimierungen	3h 00m		2h 00m					1h 00m
BA-161	Release auf dem Server deployen	3h 30m	2h 00m	1h 30m					
BA-162	Dokumentation	2h 30m		1h 00m					1h 30m
BA-167	Testing Backend	30m					30m		
BA-168	Readme Frontend	30m		30m					
BA-169	Doku aufbereiten und schicken für Feedback	1h 00m		1h 00m					
Ursin Zimmermann		**15h 45m**	8h 00m	00m	00m	45m	2h 00m	2h 00m	3h 00m
BA-129	Usability Tests Vorbereitung	1h 15m					15m		1h 00m
BA-149	Testing & Optimierungen Platzhalter	15m					15m		
BA-155	Administration	2h 15m	1h 00m			15m			1h 00m
BA-157	Entscheidungsprotokolle	15m				15m			
BA-161	Release auf dem Server deployen	1h 30m					1h 30m		
BA-162	Dokumentation	4h 30m	1h 30m					2h 00m	1h 00m
BA-167	Testing Backend	5h 30m	5h 30m						
BA-168	Readme Frontend	15m				15m			

Dez. 2022

Tickets, gruppiert nach Autor der Arbeit			5 Mo.	6 Di.	7 Mi.	8 Do.	9 Fr.	10 Sa.	11 So.
Gesamtzeit		**44h 00m**	**3h 45m**	**15h 00m**	**7h 15m**	**6h 00m**	**4h 30m**	**7h 30m**	**00m**
Abdullah Almaz		**20h 15m**	1h 30m	7h 30m	6h 15m	1h 30m	00m	3h 30m	00m
BA-128	Usability Tests Durchführung	7h 00m		7h 00m					
BA-130	Usability Tests dokumentieren	30m						30m	
BA-170	Administration	1h 30m	30m			30m		30m	
BA-171	Meeting mit Frieder	30m		30m					
BA-172	Entscheidungsprotokol	15m			15m				
BA-173	Benutzeranleitung	30m						30m	
BA-174	Fehlerbehebungen im Code	7h 00m			6h 00m	1h 00m			
BA-175	Quantity dokumentieren	1h 00m	1h 00m						
BA-176	Testing dokumentieren	30m						30m	
BA-177	Evaluation Vorbereitung und SFS schicken	30m						30m	
BA-184	Ergebnis & Weiterentwicklung dokumentieren	1h 00m						1h 00m	
Ursin Zimmermann		**23h 45m**	2h 15m	7h 30m	1h 00m	4h 30m	4h 30m	4h 00m	00m
BA-128	Usability Tests Durchführung	7h 00m		7h 00m					
BA-130	Usability Tests dokumentieren	6h 00m				2h 00m	4h 00m		
BA-170	Administration	1h 45m	45m			30m		30m	
BA-171	Meeting mit Frieder	30m		30m					
BA-172	Entscheidungsprotokol	30m			30m				
BA-173	Benutzeranleitung	2h 30m						2h 30m	
BA-174	Fehlerbehebungen im Code	30m			30m				
BA-175	Quantity dokumentieren	15m	15m						
BA-176	Testing dokumentieren	3h 45m	1h 15m			2h 00m	30m		
BA-177	Evaluation Vorbereitung und SFS schicken	30m						30m	
BA-184	Ergebnis & Weiterentwicklung dokumentieren	30m						30m	

Dez. 2022

Tickets, gruppiert nach Autor der Arbeit			12 Mo.	13 Di.	14 Mi.	15 Do.	16 Fr.	17 Sa.	18 So.
Gesamtzeit		**40h 00m**	**7h 45m**	**14h 15m**	**00m**	**3h 15m**	**5h 00m**	**5h 15m**	**4h 30m**
Abdullah Almaz		**19h 30m**	3h 30m	7h 15m	00m	3h 15m	2h 45m	1h 30m	1h 15m
BA-125	Plakat/Poster	2h 00m		2h 00m					
BA-132	Evaluation vorbereiten	1h 00m							1h 00m
BA-163	Rückbau und Doku von Testing Workarounds	15m						15m	
BA-180	Broschürenabstract / Management Summary sc...	3h 00m		3h 00m					
BA-181	Anhang soweit möglich abschliessen	2h 30m	2h 30m						
BA-182	Dokumentation	3h 00m		1h 00m		2h 00m			
BA-185	Docker Environment Variablen	30m		30m					
BA-186	Mail Adrian bzgl. Zitat	30m	15m	15m					
BA-187	Administration	1h 45m	45m			15m	15m	15m	15m
BA-188	Meeting mit Frieder	30m		30m					
BA-190	Backend Kontrolle von Fremdleistung/kopiert/ad...	30m					30m		
BA-191	Glossar	4h 00m				1h 00m	2h 00m	1h 00m	
Ursin Zimmermann		**20h 30m**	4h 15m	7h 00m	00m	00m	2h 15m	3h 45m	3h 15m
BA-125	Plakat/Poster	2h 00m		2h 00m					
BA-132	Evaluation vorbereiten	1h 00m							1h 00m
BA-163	Rückbau und Doku von Testing Workarounds	30m						30m	
BA-180	Broschürenabstract / Management Summary sc...	3h 00m		3h 00m					
BA-181	Anhang soweit möglich abschliessen	15m		15m					
BA-182	Dokumentation	2h 30m	30m						2h 00m
BA-185	Docker Environment Variablen	3h 00m	3h 00m						
BA-186	Mail Adrian bzgl. Zitat	15m		15m					
BA-187	Administration	1h 30m	45m				15m	15m	15m
BA-188	Meeting mit Frieder	30m		30m					
BA-189	Entscheidungsprotokoll	30m		30m					
BA-190	Backend Kontrolle von Fremdleistung/kopiert/ad...	30m		30m					
BA-191	Glossar	5h 00m					2h 00m	3h 00m	

Dez. 2022

Tickets, gruppiert nach Autor der Arbeit			19 Mo.	20 Di.	21 Mi.	22 Do.	23 Fr.	24 Sa.	25 So.
Gesamtzeit		**27h 00m**	**6h 00m**	**14h 30m**	**6h 30m**	**00m**	**00m**	**00m**	**00m**
Abdullah Almaz		**13h 45m**	3h 00m	7h 15m	3h 30m	00m	00m	00m	00m
BA-124	Abstract	1h 30m		1h 30m					
BA-131	Evaluation Meeting	1h 00m	1h 00m						
BA-192	Administration	2h 00m		1h 00m	1h 00m				
BA-193	Meeting mit Frieder	30m		30m					
BA-195	Dokumentation Platzhalter	2h 00m			2h 00m				
BA-196	Präsentation vorbereiten	2h 00m		2h 00m					
BA-197	Plakat überarbeiten	45m		45m					
BA-198	Umbenennen von WareEntryCharge zu Batch/W...	2h 00m	2h 00m						
BA-199	Dokumentation Evaluation	1h 30m		1h 30m					
BA-201	Gegenlesen lassen	30m			30m				
Ursin Zimmermann		**13h 15m**	3h 00m	7h 15m	3h 00m	00m	00m	00m	00m
BA-124	Abstract	1h 30m		1h 30m					
BA-131	Evaluation Meeting	1h 00m	1h 00m						
BA-192	Administration	1h 00m		1h 00m					
BA-193	Meeting mit Frieder	30m		30m					
BA-194	Entscheidungsprotokolle	30m			30m				
BA-195	Dokumentation Platzhalter	2h 00m			2h 00m				
BA-196	Präsentation vorbereiten	2h 00m		2h 00m					
BA-197	Plakat überarbeiten	45m		45m					
BA-198	Umbenennen von WareEntryCharge zu Batch/W...	2h 00m	2h 00m						
BA-200	Bilderquellen durchgehen	1h 30m		1h 30m					
BA-201	Gegenlesen lassen	30m			30m				

Dez. 2022

Tickets, gruppiert nach Autor der Arbeit			26 Mo.	27 Di.	28 Mi.	29 Do.	30 Fr.	31 Sa.	1 So.
Gesamtzeit		**10h 00m**	**00m**	**10h 00m**	**00m**	**00m**	**00m**	**00m**	**00m**
Abdullah Almaz		**5h 00m**	00m	5h 00m	00m	00m	00m	00m	00m
BA-135	Gegenlesen Korrekturen	5h 00m		5h 00m					
Ursin Zimmermann		**5h 00m**	00m	5h 00m	00m	00m	00m	00m	00m
BA-135	Gegenlesen Korrekturen	5h 00m		5h 00m					

Jan. 2023								
Tickets, gruppiert nach Autor der Arbeit			2 Mo.	3 Di.	4 Mi.	5 Do.	6 Fr.	7 Sa.	8 So.
Gesamtzeit		**39h 15m**	**8h 00m**	**2h 30m**	**2h 15m**	**00m**	**00m**	**14h 30m**	**12h 00m**
Abdullah Almaz		**18h 30m**	4h 00m	1h 30m	2h 00m	00m	00m	5h 00m	6h 00m
BA-133	Präsentation abschliessen und üben	4h 00m							4h 00m
BA-183	Weiteres Feedback Frieder & Scribbr	8h 00m	4h 00m		2h 00m			2h 00m	
BA-202	Eigenständigkeitserklärung und Einverständnise...	1h 00m						1h 00m	
BA-203	Arbeit abschliessen, Dokumente hochladen, usw.	4h 00m						2h 00m	2h 00m
BA-204	Meeting Frieder	1h 00m		1h 00m					
BA-205	Entscheidungsprotokoll	30m		30m					
Ursin Zimmermann		**20h 45m**	4h 00m	1h 00m	15m	00m	00m	9h 30m	6h 00m
BA-133	Präsentation abschliessen und üben	4h 00m							4h 00m
BA-183	Weiteres Feedback Frieder & Scribbr	10h 30m	4h 00m					6h 30m	
BA-202	Eigenständigkeitserklärung und Einverständnise...	1h 00m						1h 00m	
BA-203	Arbeit abschliessen, Dokumente hochladen, usw.	4h 00m						2h 00m	2h 00m
BA-204	Meeting Frieder	1h 00m		1h 00m					
BA-205	Entscheidungsprotokoll	15m			15m				

J Export Tickets

«Zeitschätzung» und «Geleistete Zeit» sind in Minuten angegeben.

Ticket-ID	Zusammenfassung	Erstellt	Gelöst	Verantwortlicher	Zeitschätzung	Geleistete Zeit	Sprints
BA-1	Administration	20 Sep 2022	20 Sep 2022	ursin.zimmermann	240	240	SW 1 - KW 38
BA-2	Projektplan erstellen	20 Sep 2022	23 Sep 2022	abdullah.almaz	240	195	SW 1 - KW 38
BA-3	Risikoanalyse vorbereiten	20 Sep 2022	27 Sep 2022	abdullah.almaz	60	60	SW 1 - KW 38
BA-4	BA Dokument vorbereiten	20 Sep 2022	23 Sep 2022	ursin.zimmermann	360	240	SW 1 - KW 38
BA-5	Fragen an Kunden vorbereiten	20 Sep 2022	23 Sep 2022	abdullah.almaz	60	45	SW 1 - KW 38
BA-6	Meetingprotokoll erstellen	20 Sep 2022	22 Sep 2022	ursin.zimmermann	120	90	SW 1 - KW 38
BA-7	Reports Zeiterfassung erstellen	20 Sep 2022	23 Sep 2022	abdullah.almaz	120	90	SW 1 - KW 38
BA-8	Requirementvorstellungen dokumentieren	20 Sep 2022	23 Sep 2022	abdullah.almaz	60	60	SW 1 - KW 38
BA-9	Requirementvorstellungen dokumentieren	20 Sep 2022	26 Sep 2022	ursin.zimmermann	60	60	SW 1 - KW 38
BA-12	Einlesen BA Leitfaden und BA Tipps	20 Sep 2022	20 Sep 2022	abdullah.almaz	60	60	SW 1 - KW 38
BA-13	Einlesen BA Leitfaden und BA Tipps	20 Sep 2022	22 Sep 2022	ursin.zimmermann	60	60	SW 1 - KW 38
BA-14	Fragen an Kunden vorbereiten	20 Sep 2022	23 Sep 2022	ursin.zimmermann	60	45	SW 1 - KW 38
BA-15	Aufgabestellung lesen und Feedback geben	20 Sep 2022	23 Sep 2022	abdullah.almaz	60	60	SW 1 - KW 38
BA-16	Aufgabestellung lesen und Feedback geben	20 Sep 2022	23 Sep 2022	ursin.zimmermann	60	60	SW 1 - KW 38
BA-17	Themen für fundierte Quellenrecherche bestimmen	20 Sep 2022	25 Sep 2022	abdullah.almaz	60	60	SW 1 - KW 38
BA-18	Austausch	20 Sep 2022	23 Sep 2022	ursin.zimmermann	240	240	SW 1 - KW 38
BA-19	Aufsetzen und Konfiguration Youtrack	20 Sep 2022	20 Sep 2022	abdullah.almaz	240	240	SW 1 - KW 38
BA-34	Tickets Folgesprint erstellen	25 Sep 2022	25 Sep 2022	abdullah.almaz	60	60	SW 1 - KW 38
BA-21	Themen und Bücher für fundierte Quellenrecherche bestimmen	25 Sep 2022	2 Okt 2022	abdullah.almaz	240	210	SW 2 - KW 39
BA-22	Meetingprotokoll erstellen	25 Sep 2022	29 Sep 2022	ursin.zimmermann	120	120	SW 2 - KW 39

BA-23	Administration	25 Sep 2022	2 Okt 2022	ursin.zimmermann	360	255	SW 2 - KW 39
BA-24	Meeting Betreuer und Projektpartner	25 Sep 2022	27 Sep 2022	abdullah.almaz	120	120	SW 2 - KW 39
BA-26	Nicht funktionale Anforderungen abklären und dokumentieren	25 Sep 2022	30 Sep 2022	abdullah.almaz	240	240	SW 2 - KW 39
BA-27	Technology Constrains abklären und dokumentieren	25 Sep 2022	30 Sep 2022	ursin.zimmermann	240	240	SW 2 - KW 39
BA-30	User Research vorbereiten	25 Sep 2022	2 Okt 2022	abdullah.almaz	240	210	SW 2 - KW 39
BA-31	Beschreibung Ausgangssituation	25 Sep 2022	2 Okt 2022	ursin.zimmermann	120	150	SW 2 - KW 39
BA-36	Termine für Usability Test und Evaluation definieren und mit SFS besprechen	27 Sep 2022	30 Sep 2022	abdullah.almaz	60	60	SW 2 - KW 39
BA-37	Projektplan anpassen	27 Sep 2022	1 Okt 2022	ursin.zimmermann	120	105	SW 2 - KW 39
BA-40	Oli & Frieder schreiben	30 Sep 2022	30 Sep 2022	abdullah.almaz	60	60	SW 2 - KW 39
BA-45	Dokumente drucken für Dienstag	30 Sep 2022	2 Okt 2022	ursin.zimmermann	15	15	SW 2 - KW 39
BA-28	Funktionale Anforderungen abklären und dokumentieren	25 Sep 2022	9 Okt 2022	abdullah.almaz	360	270	SW 3 - KW 40
BA-29	Domain Model skizzieren und dokumentieren	25 Sep 2022	8 Okt 2022	abdullah.almaz	240	240	SW 3 - KW 40
BA-38	Git Projekte aufsetzen & Rechte testen	28 Sep 2022	8 Okt 2022	ursin.zimmermann	60	60	SW 3 - KW 40
BA-42	User Research dokumentieren	30 Sep 2022	10 Okt 2022	Nicht zugewiesen	480	300	SW 3 - KW 40
BA-43	Administratives	30 Sep 2022	9 Okt 2022	ursin.zimmermann	240	240	SW 3 - KW 40
BA-44	Personas erstellen	30 Sep 2022	8 Okt 2022	ursin.zimmermann	240	240	SW 3 - KW 40
BA-46	Besuch SFS	2 Okt 2022	4 Okt 2022	abdullah.almaz	720	960	SW 3 - KW 40
BA-47	Moodboard & Paper Prototype besprechen	2 Okt 2022	9 Okt 2022	abdullah.almaz	480	360	SW 3 - KW 40
BA-48	Meeting mit Frieder	3 Okt 2022	3 Okt 2022	abdullah.almaz	60	60	SW 3 - KW 40
BA-49	Entscheidungsprotokoll Montag	3 Okt 2022	5 Okt 2022	ursin.zimmermann	30	30	SW 3 - KW 40
BA-50	Entscheidungsprotokoll Dienstag	3 Okt 2022	8 Okt 2022	ursin.zimmermann	60	60	SW 3 - KW 40
BA-51	Risikomatrix anpassen	3 Okt 2022	9 Okt 2022	abdullah.almaz	120	120	SW 3 - KW 40
BA-52	Projektplan anpassen	8 Okt 2022	9 Okt 2022	ursin.zimmermann	30	30	SW 3 - KW 40

BA-57	Anforderung schicken	9 Okt 2022	9 Okt 2022	abdullah.almaz	120	90	SW 3 - KW 40
BA-53	Paper Prototype erstellen	9 Okt 2022	11 Okt 2022	ursin.zimmer-mann	480	540	SW 4 - KW 41
BA-54	Meeting mit Frieder	9 Okt 2022	10 Okt 2022	Nicht zugewiesen	120	60	SW 4 - KW 41
BA-55	Meeting mit SFS	9 Okt 2022	11 Okt 2022	ursin.zimmer-mann	120	90	SW 4 - KW 41
BA-56	Entscheidungsprotokolle	9 Okt 2022	11 Okt 2022	ursin.zimmer-mann	120	60	SW 4 - KW 41
BA-59	Literatur Recherche	9 Okt 2022	17 Okt 2022	abdullah.almaz	1200	1080	SW 4 - KW 41
BA-60	Analyse Sinnvoller Libraries Backend	9 Okt 2022	11 Okt 2022	ursin.zimmer-mann	120	120	SW 4 - KW 41
BA-61	Analyse Sinnvoller Libraries Frontend	9 Okt 2022	11 Okt 2022	abdullah.almaz	120	120	SW 4 - KW 41
BA-62	Architektur planen	9 Okt 2022	15 Okt 2022	ursin.zimmer-mann	240	240	SW 4 - KW 41
BA-63	Anpassungen Anforderungs-analyse	9 Okt 2022	13 Okt 2022	abdullah.almaz	240	240	SW 4 - KW 41
BA-64	Dokumentation finalisieren von Einführung Bericht	9 Okt 2022	15 Okt 2022	abdullah.almaz	240	60	SW 4 - KW 41
BA-65	Administration	10 Okt 2022	15 Okt 2022	ursin.zimmer-mann	240	300	SW 4 - KW 41
BA-66	Moodboard dokumentieren	11 Okt 2022	11 Okt 2022	ursin.zimmer-mann	120	120	SW 4 - KW 41
BA-67	Server aufsetzen	11 Okt 2022	15 Okt 2022	abdullah.almaz	240	180	SW 4 - KW 41
BA-41	Backend vorbereiten	30 Sep 2022	22 Okt 2022	ursin.zimmer-mann	240	240	SW 5 - KW 42
BA-68	Dokumentieren Design	15 Okt 2022	22 Okt 2022	abdullah.almaz	240	240	SW 5 - KW 42
BA-69	Dokumentation Literatur Recherche	15 Okt 2022	22 Okt 2022	abdullah.almaz	720	720	SW 5 - KW 42
BA-70	Administration	15 Okt 2022	23 Okt 2022	ursin.zimmer-mann	240	255	SW 5 - KW 42
BA-71	Meeting mit Frieder	15 Okt 2022	18 Okt 2022	abdullah.almaz	120	150	SW 5 - KW 42
BA-72	Entscheidungsprotokolle	15 Okt 2022	23 Okt 2022	ursin.zimmer-mann	60	60	SW 5 - KW 42
BA-73	Frontend vorbereiten	15 Okt 2022	18 Okt 2022	abdullah.almaz	240	225	SW 5 - KW 42
BA-75	Anforderungen abschliessen	15 Okt 2022	22 Okt 2022	abdullah.almaz	120	120	SW 5 - KW 42
BA-76	Architekturprototyp Backend	15 Okt 2022	24 Okt 2022	ursin.zimmer-mann	480	540	SW 5 - KW 42
BA-77	Datenbank planen & anlegen	15 Okt 2022	17 Okt 2022	ursin.zimmer-mann	240	240	SW 5 - KW 42

BA-79	Design Refining	18 Okt 2022	23 Okt 2022	abdullah.almaz	120	90	SW 5 - KW 42
BA-80	Figma Prototyp anpassen anhand Feedback	18 Okt 2022	23 Okt 2022	abdullah.almaz	360	300	SW 5 - KW 42
BA-82	Doku für gegenlesen vorbereiten und schicken	18 Okt 2022	21 Okt 2022	abdullah.almaz	120	120	SW 5 - KW 42
BA-83	Hardware Scanner Doku lesen	18 Okt 2022	22 Okt 2022	abdullah.almaz	60	60	SW 5 - KW 42
BA-84	Vorbereitungen Zwischenpräsi	18 Okt 2022	6 Nov 2022	abdullah.almaz	480	420	SW 7 - KW 44
BA-91	Risikomatrix anpassen	23 Okt 2022	6 Nov 2022	abdullah.almaz	15	15	SW 7 - KW 44
BA-95	Frontend Entwicklung	23 Okt 2022	6 Nov 2022	abdullah.almaz	960	795	SW 7 - KW 44
BA-97	Projektplan pflegen	30 Okt 2022	1 Nov 2022	ursin.zimmermann	30	30	SW 7 - KW 44
BA-99	Meeting mit Frieder	30 Okt 2022	1 Nov 2022	ursin.zimmermann	120	60	SW 7 - KW 44
BA-100	Dokumentation Todos abarbeiten	30 Okt 2022	6 Nov 2022	ursin.zimmermann	240	225	SW 7 - KW 44
BA-101	Entscheidungsprotokoll	30 Okt 2022	1 Nov 2022	ursin.zimmermann	60	45	SW 7 - KW 44
BA-102	Administration	30 Okt 2022	6 Nov 2022	ursin.zimmermann	300	270	SW 7 - KW 44
BA-103	Backend - REST Endpoints	31 Okt 2022	6 Nov 2022	ursin.zimmermann	960	1050	SW 7 - KW 44
BA-78	Architekturprototyp Frontend	15 Okt 2022	30 Okt 2022	abdullah.almaz	240	180	SW 6 - KW 43
BA-85	Feedback von Review einbauen	23 Okt 2022	30 Okt 2022	abdullah.almaz	720	660	SW 6 - KW 43
BA-86	Meeting mit Frieder	23 Okt 2022	25 Okt 2022	Nicht zugewiesen	180	120	SW 6 - KW 43
BA-87	Meeting mit SFS	23 Okt 2022	25 Okt 2022	Nicht zugewiesen	120	120	SW 6 - KW 43
BA-88	Administration	23 Okt 2022	30 Okt 2022	ursin.zimmermann	360	420	SW 6 - KW 43
BA-89	Entscheidungsprotokolle	23 Okt 2022	30 Okt 2022	ursin.zimmermann	120	90	SW 6 - KW 43
BA-90	Dokumentation Wireframes	23 Okt 2022	25 Okt 2022	abdullah.almaz	240	240	SW 6 - KW 43
BA-93	Datenbank anpassen	23 Okt 2022	25 Okt 2022	ursin.zimmermann	90	90	SW 6 - KW 43
BA-94	Platzhalter Backendentwicklung	23 Okt 2022	30 Okt 2022	ursin.zimmermann	480	240	SW 6 - KW 43
BA-96	Literaturrecherche V2	25 Okt 2022	30 Okt 2022	ursin.zimmermann	720	750	SW 6 - KW 43
BA-104	Administration	6 Nov 2022	13 Nov 2022	abdullah.almaz	360	270	SW 8 - KW 45

BA-105	Zwischenpräsentation	6 Nov 2022	8 Nov 2022	ursin.zimmermann	240	180	SW 8 - KW 45
BA-106	Dokumentation implementation	6 Nov 2022	13 Nov 2022	abdullah.almaz	480	330	SW 8 - KW 45
BA-107	Refactoring & Testing Backend	6 Nov 2022	13 Nov 2022	ursin.zimmermann	720	660	SW 8 - KW 45
BA-108	Frontend: Entwicklung Lager-Ansicht	6 Nov 2022	12 Nov 2022	abdullah.almaz	960	900	SW 8 - KW 45
BA-109	Korrekturen Review für BA-103	6 Nov 2022	7 Nov 2022	ursin.zimmermann	120	90	SW 8 - KW 45
BA-110	Beispiel Daten Backend	6 Nov 2022	7 Nov 2022	ursin.zimmermann	240	195	SW 8 - KW 45
BA-111	Update Database - Diameter	7 Nov 2022	8 Nov 2022	ursin.zimmermann	90	75	SW 8 - KW 45
BA-112	Termine für Abgaben im Projektplan und Sprintfesthalten und Ressourcen reservieren	12 Nov 2022	15 Nov 2022	abdullah.almaz	120	120	SW 9 - KW 46
BA-113	Readme Frontend	12 Nov 2022	15 Nov 2022	abdullah.almaz	180	135	SW 9 - KW 46
BA-114	Readme Backend	12 Nov 2022	15 Nov 2022	ursin.zimmermann	180	135	SW 9 - KW 46
BA-115	Frontend Delivery	12 Nov 2022	19 Nov 2022	abdullah.almaz	480	480	SW 9 - KW 46
BA-116	Backend Tests & Refactoring	12 Nov 2022	15 Nov 2022	ursin.zimmermann	480	390	SW 9 - KW 46
BA-117	Dokumentation Implementation	12 Nov 2022	21 Nov 2022	ursin.zimmermann	270	210	SW 9 - KW 46
BA-119	Administration	12 Nov 2022	20 Nov 2022	abdullah.almaz	360	330	SW 9 - KW 46
BA-120	Meeting Frieder	12 Nov 2022	16 Nov 2022	ursin.zimmermann	120	60	SW 9 - KW 46
BA-121	Entscheidungsprotokoll	13 Nov 2022	20 Nov 2022	ursin.zimmermann	60	45	SW 9 - KW 46
BA-122	Backend: InventoryEnding REST Endpoints	14 Nov 2022	15 Nov 2022	ursin.zimmermann	120	90	SW 9 - KW 46
BA-123	Adrian Video zukommen lassen	14 Nov 2022	15 Nov 2022	abdullah.almaz	15	15	SW 9 - KW 46
BA-136	Frontend Ending	15 Nov 2022	20 Nov 2022	abdullah.almaz	240	300	SW 9 - KW 46
BA-137	Dockerfile Frontend für Produktion	16 Nov 2022	20 Nov 2022	abdullah.almaz	120	120	SW 9 - KW 46
BA-138	Dockerfile Backend für Produktion	16 Nov 2022	20 Nov 2022	ursin.zimmermann	180	180	SW 9 - KW 46
BA-139	Adrian Map verlangen und einbauen	19 Nov 2022	22 Nov 2022	abdullah.almaz	120	105	SW 10 - KW 47
BA-140	Frontend: Scan Logik einbauen	20 Nov 2022	22 Nov 2022	ursin.zimmermann	360	285	SW 10 - KW 47

BA-141	Administration	20 Nov 2022	27 Nov 2022	ursin.zimmermann	360	135	SW 10 - KW 47
BA-142	Dokumentation	20 Nov 2022	27 Nov 2022	abdullah.almaz	360	210	SW 10 - KW 47
BA-143	Meeting Frieder	20 Nov 2022	23 Nov 2022	ursin.zimmermann	120	90	SW 10 - KW 47
BA-144	Meetingprotokoll erstellen	20 Nov 2022	27 Nov 2022	ursin.zimmermann	60	60	SW 10 - KW 47
BA-145	SFS CI Pipeline anschauen	20 Nov 2022	23 Nov 2022	abdullah.almaz	120	60	SW 10 - KW 47
BA-146	Usability Tests Vorbereitungen	20 Nov 2022	27 Nov 2022	abdullah.almaz	240	210	SW 10 - KW 47
BA-147	Testdaten für Usabilitytest	20 Nov 2022	25 Nov 2022	ursin.zimmermann	120	75	SW 10 - KW 47
BA-148	Release Build abschliessen	20 Nov 2022	26 Nov 2022	ursin.zimmermann	240	195	SW 10 - KW 47
BA-150	Dockerfile Frontend für Produktion	20 Nov 2022	23 Nov 2022	abdullah.almaz	120	120	SW 10 - KW 47
BA-151	Dockerfile Backend für Produktion	20 Nov 2022	26 Nov 2022	ursin.zimmermann	60	60	SW 10 - KW 47
BA-152	Swipe Refresh	21 Nov 2022	21 Nov 2022	abdullah.almaz	120	105	SW 10 - KW 47
BA-153	PWA erstellen für Test erstellen	21 Nov 2022	22 Nov 2022	abdullah.almaz	120	105	SW 10 - KW 47
BA-159	Testing Frontend	22 Nov 2022	27 Nov 2022	abdullah.almaz	480	435	SW 10 - KW 47
BA-164	Readme anpassen - Backend	26 Nov 2022	27 Nov 2022	ursin.zimmermann	60	60	SW 10 - KW 47
BA-165	Testing Backend	27 Nov 2022	27 Nov 2022	ursin.zimmermann	180	120	SW 10 - KW 47
BA-129	Usability Tests Vorbereitung	15 Nov 2022	4 Dez 2022	abdullah.almaz	240	165	SW 11 - KW 48

BA-149	Testing & Optimierungen Platzhalter	20 Nov 2022	4 Dez 2022	ursin.zimmermann	240	75	SW 11 - KW 48
BA-155	Administration	22 Nov 2022	4 Dez 2022	abdullah.almaz	360	270	SW 11 - KW 48
BA-156	Meeting mit Frieder	22 Nov 2022	29 Nov 2022	abdullah.almaz	60	30	SW 11 - KW 48
BA-157	Entscheidungsprotokolle	22 Nov 2022	1 Dez 2022	abdullah.almaz	60	45	SW 11 - KW 48
BA-158	Sammelticket für Fehler & Optimierungen	22 Nov 2022	4 Dez 2022	abdullah.almaz	360	180	SW 11 - KW 48
BA-161	Release auf dem Server deployen	23 Nov 2022	4 Dez 2022	abdullah.almaz	480	420	SW 11 - KW 48
BA-162	Dokumentation	23 Nov 2022	4 Dez 2022	ursin.zimmermann	480	420	SW 11 - KW 48
BA-167	Testing Backend	27 Nov 2022	2 Dez 2022	ursin.zimmermann	360	360	SW 11 - KW 48
BA-168	Readme Frontend	28 Nov 2022	1 Dez 2022	abdullah.almaz	60	45	SW 11 - KW 48
BA-169	Doku aufbereiten und schicken für Feedback	29 Nov 2022	29 Nov 2022	abdullah.almaz	60	60	SW 11 - KW 48
BA-128	Usability Tests Durchführung	15 Nov 2022	6 Dez 2022	ursin.zimmermann	960	840	SW 12 - KW 49
BA-130	Usability Tests dokumentieren	15 Nov 2022	10 Dez 2022	ursin.zimmermann	480	390	SW 12 - KW 49
BA-170	Administration	3 Dez 2022	10 Dez 2022	ursin.zimmermann	240	195	SW 12 - KW 49
BA-171	Meeting mit Frieder	3 Dez 2022	7 Dez 2022	ursin.zimmermann	120	60	SW 12 - KW 49
BA-172	Entscheidungsprotokol	3 Dez 2022	7 Dez 2022	ursin.zimmermann	60	45	SW 12 - KW 49
BA-173	Benutzeranleitung	3 Dez 2022	10 Dez 2022	ursin.zimmermann	240	180	SW 12 - KW 49

BA-174	Fehlerbehebungen im Code	3 Dez 2022	7 Dez 2022	abdullah.almaz	480	450	SW 12 - KW 49
BA-175	Quantity dokumentieren	4 Dez 2022	6 Dez 2022	abdullah.almaz	180	75	SW 12 - KW 49
BA-176	Testing dokumentieren	4 Dez 2022	10 Dez 2022	ursin.zimmer-mann	240	255	SW 12 - KW 49
BA-177	Evaluation Vorbereitung und SFS schicken	4 Dez 2022	10 Dez 2022	ursin.zimmer-mann	120	60	SW 12 - KW 49
BA-184	Ergebnis & Weiterentwicklung dokumentieren	8 Dez 2022	10 Dez 2022	abdullah.almaz	240	90	SW 12 - KW 49
BA-125	Plakat/Poster	15 Nov 2022	13 Dez 2022	abdullah.almaz	360	240	SW 13 - KW 50
BA-132	Evaluation vorbereiten	15 Nov 2022	18 Dez 2022	abdullah.almaz	240	120	SW 13 - KW 50
BA-163	Rückbau und Doku von Testing Workarounds	26 Nov 2022	17 Dez 2022	ursin.zimmer-mann	120	45	SW 13 - KW 50
BA-180	Broschürenabstract / Manage-ment Summary schreiben	5 Dez 2022	13 Dez 2022	abdullah.almaz	480	360	SW 13 - KW 50
BA-181	Anhang soweit möglich ab-schliessen	5 Dez 2022	13 Dez 2022	abdullah.almaz	240	165	SW 13 - KW 50
BA-182	Dokumentation	5 Dez 2022	18 Dez 2022	abdullah.almaz	360	330	SW 13 - KW 50
BA-185	Docker Environment Variablen	8 Dez 2022	13 Dez 2022	ursin.zimmer-mann	240	210	SW 13 - KW 50
BA-186	Mail Adrian bzgl. Zitat	10 Dez 2022	13 Dez 2022	abdullah.almaz	60	45	SW 13 - KW 50
BA-187	Administration	11 Dez 2022	18 Dez 2022	ursin.zimmer-mann	240	195	SW 13 - KW 50
BA-188	Meeting mit Frieder	11 Dez 2022	13 Dez 2022	ursin.zimmer-mann	120	60	SW 13 - KW 50
BA-189	Entscheidungsprotokoll	11 Dez 2022	16 Dez 2022	ursin.zimmer-mann	45	30	SW 13 - KW 50

BA-190	Backend Kontrolle von Fremd-leistung/kopiert/adaptiert	12 Dez 2022	16 Dez 2022	ursin.zimmer-mann	120	60	SW 13 - KW 50
BA-191	Glossar	13 Dez 2022	18 Dez 2022	abdullah.almaz	600	540	SW 13 - KW 50
BA-124	Abstract	15 Nov 2022	21 Dez 2022	abdullah.almaz	240	180	SW 14 - KW 51
BA-131	Evaluation Meeting	15 Nov 2022	19 Dez 2022	ursin.zimmer-mann	120	120	SW 14 - KW 51
BA-192	Administration	18 Dez 2022	21 Dez 2022	abdullah.almaz	240	180	SW 14 - KW 51
BA-193	Meeting mit Frieder	18 Dez 2022	21 Dez 2022	ursin.zimmer-mann	60	60	SW 14 - KW 51
BA-194	Entscheidungsprotokolle	18 Dez 2022	21 Dez 2022	ursin.zimmer-mann	45	30	SW 14 - KW 51
BA-195	Dokumentation Platzhalter	18 Dez 2022	21 Dez 2022	abdullah.almaz	480	240	SW 14 - KW 51
BA-196	Präsentation vorbereiten	18 Dez 2022	21 Dez 2022	ursin.zimmer-mann	360	240	SW 14 - KW 51
BA-197	Plakat überarbeiten	19 Dez 2022	21 Dez 2022	abdullah.almaz	120	90	SW 14 - KW 51
BA-198	Umbenennen von WareEntry-Charge zu Batch/WareEntry-Batch	19 Dez 2022	19 Dez 2022	ursin.zimmer-mann	240	240	SW 14 - KW 51
BA-199	Dokumentation Evaluation	20 Dez 2022	21 Dez 2022	ursin.zimmer-mann	120	90	SW 14 - KW 51
BA-200	Bilderquellen durchgehen	20 Dez 2022	21 Dez 2022	ursin.zimmer-mann	120	90	SW 14 - KW 51
BA-201	Gegenlesen lassen	20 Dez 2022	21 Dez 2022	abdullah.almaz	120	60	SW 14 - KW 51
BA-135	Gegenlesen Korrekturen	15 Nov 2022	1 Jan 2023	ursin.zimmer-mann	960	600	SW 15 - KW 52
BA-133	Präsentation abschliessen und üben	15 Nov 2022	7 Jan 2023	abdullah.almaz	480	480	SW 16 - KW 01

BA-183	Weiteres Feedback Frieder & Scribbr	7 Dez 2022	7 Jan 2023	abdullah.almaz	1200	1110	SW 16 - KW 01
BA-202	Eigenständigkeitserklärung und Einverständniserklärung	20 Dez 2022	2 Jan 2023	ursin.zimmer-mann	120	120	SW 16 - KW 01
BA-203	Arbeit abschliessen, Dokumente hochladen, usw.	1 Jan 2023	7 Jan 2023	abdullah.almaz	480	480	SW 16 - KW 01
BA-204	Meeting Frieder	3 Jan 2023	3 Jan 2023	ursin.zimmer-mann	120	120	SW 16 - KW 01
BA-205	Entscheidungsprotokoll	3 Jan 2023	7 Jan 2023	abdullah.almaz	60	45	SW 16 - KW 01