PROJEKTDOKUMENTATION SOMMER 2022

STAPELPLANUNG - WEBAPPLIKATION ZUR VERWALTUNG VON ZU KOMMISIONIERENDEN WAREN MIT ANBINDUNG AN ABAS ERP

STAND: 29.04.2022

Prüfungsbewerber:

Arthur Schimpf

Bernhard-Lichtenberg-Straße. 74

76189 Karlsruhe

Prüflings-Nr.: 30790

Ausbildungsbetrieb:

abas Software GmbH

Gartenstraße 67

76135 Karlsruhe

INHALTSVERZEICHNIS

ABBILDUNGSVERSZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

CSS *Cascading Style Sheets*

EDP *Ausschreibung ergänzung*

ERP *Enterprise Resource Planning, Enterprise Resource Planning*

FTPS *File Transfer Protocol Secure*

HTML *Hypertext Markup Language*

IHK *Industrie- und Handelskammer*

JVM *Java Virtual Machine*

HINWEISE ZUM DATENSCHUTZ

Um den firmeninternen Datenschutzrichtlinien zu entsprechen, wurden sensible Daten durch imaginäre Daten ersetzt. Das betrifft Daten, sowie Geschäftsprozesse des Kunden. Kostenrechnungen werden ausschließlich mit erfundenen Beträgen durchgeführt.

GENDER-HINWEIS

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers(m/w/d) verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

1. EINLEITUNG

1.1 PROJEKTBESCHREIBUNG

In der folgenden Dokumentation schildert der Autor den Ablauf des IHK-Abschlussprojektes, welches im Rahmen der Ausbildung zum Fachinformatiker in Anwendungsentwicklung durgeführt wurde. Der leitende Ausbildungsbetrieb ist die ABAS Software GmbH, ein Softwareunternehmen mit mehr als 65 internationalen Standorten. Der Hauptsitz des Unternehmens befindet sich in Karlsruhe. Das Kernprodukt des Unternehmens ist ein Enterprise Resource Planning System (ERP) für die Verwaltung von unternehmerischen Aufgaben, Personal und Ressourcen wie Kapital, Betriebsmittel und Material. Zusätzlich bietet ABAS seinen Kunden Software-Dienstleistungen rund um das ERP System. Beispielsweise bietet ABAS Business Itelligence Software oder Webapplikationen zu Lagerverwaltung an.

Im Rahmen der Standardentwicklung werden bestehende ABAS-Produkte weiterentwickelt und verfeinert. Im Bereich des Professional Services werden hauptsächlich die kundennahe Betreuung geleistet und Umsetzungen spezifischer Anpassungen vorgenommen. Im Sinne des IHK-Abschlussprojektes plant, entwickelt und implementiert der Autor ein Stapelplanungssystem aus dem, den Professional Services zugehörigen Team Web.

1.2. PROJEKTZIEL

Wie bereits im vorgehenden Abschnitt erwähnt, ist das Ziel des Projektes, dass der Autor eine Webanwendung entwickelt. Diese Anwendung soll von einem Kunden verwendet werden, um die Stapelplanung im Versandbereich zu ermöglichen. Der Kunde handelt mit mittleren bis großen Autoanhängern, weshalb die Beladung eines Transporters vor der Kommissionierung organisiert werden sollte. Den Artikeln soll dafür eine Koordinate zugewiesen werden, um die Position für die Einlagerung festzulegen. Dabei beachtet der Mitarbeiter die Höhe, Länge und Breite, sowie das Gewicht der Anhänger. Diese Informationen sollen dem Anwender in der Web-Applikation zur Verfügung stehen. Zusätzlich

soll die zuvor erwähnte Planung stattfinden können. Um den Anwendern eine möglichst einfache Verwendung zu ermöglichen, soll es möglich sein nach dem drag-and-drop-Verfahren vorzugehen. In einem zweidimensionalen Raster wird die Koordinate von Artikeln definiert.

1.3. PROJEKTBEGRÜNDUNG

Der Kunde ist derzeit dabei, sein altes ERP System durch ABAS abzulösen. Dieses Projekt beinhaltet die Erneuerung aller durch das alte System gehandhabten Geschäftsprozesse. In diesem Sinne wird auch die alte Stapelplanung des Kunden ersetzt, welches die IHK-Abschlussprüfung des Autors darstellt. Der Vorteil einer neuen Stapelplanung für den Kunden ist es, dass Daten aus dem ERP System direkt verwaltet werden können. Um einen möglichst einfachen Umstieg zu ermöglichen, soll die gewohnte Funktionalität beibehalten werden. Theoretisch könnte die Position von Artikeln im ERP System eingetragen werden. Dem Benutzer wird jedoch durch die Webapplikation eine visuelle Planung ermöglicht. Dadurch kann zukünftig garantiert werden, dass Fehler während des Planungsprozesses vermieden werden. Dies ist umso wichtiger, da der Kund mit schweren Gütern handelt.

1.4. PROJEKTSCHNITTSTELLEN

Mehrere Schnittstellen werden durch die Stapelplanung erzeugt und verwendet. Eine Kommunikation zwischen dem ERP System und der Weboberfläche wird durch das Produkt „Shopfloor“ ermöglicht. Das ist ein von ABAS entwickeltes Full-Stack-Framework für die Erweiterung des ERP Systems. Dieses Produkt besteht aus einer Oberfläche und einen Geschäftslogikanteil. Im weiteren Verlauf der Dokumentation werden diese Bereiche als frontend und backend bezeichnet. Über das backend wird der Datenaustausch zum ERP ermöglicht. Dieser Anteil von Shopfloor wurde in der Programmiersprache Java entwickelt und verwendet das Framework Spring Boot. Über EDP wird der Datenaustausch zu ABAS ERP ermöglicht. Dies ist ein von ABAS entwickeltes Protokoll, welches auf HTTP basiert. Der allgemeine Datenfluss ist im Anhang unter Abbildung 6: Datenaustausch zwischen ABAS und Shopfloor abgebildet. Über EDP wird eine Anfrage an das ERP System gesendet. Nach erfolgreicher Anfrage antwortet

dieses mit den entsprechenden Daten aus dem System. Diese Daten werden im backend für das frontend vorbereitet bzw. verarbeitet und im Anschluss über HTTP an dieses übermittelt. Diese Daten können im frontend als Informationen ausgegeben werden bzw. ermöglicht diese Bereitstellung dem Anwender das Arbeiten mit den Daten aus dem ERP System. Die Weboberfläche basiert darauf auf NodeJS und verwendet für die Darstellung der Web-Komponenten das Framework VueJS. Für die Kommunikation an und vom frontend, wird Axios verwendet, ein in JavaScript geschriebener HTTP-Client.

Auch im Entwicklungsprozess lassen sich Schnittstellen auslesen. Der Projektleiter des Gesamtprojektes ist die leitende Person für die Kommunikation mit dem Kunden. Er übermittelt Anforderungen an den Entwickler und steht in direkter Kommunikation mit dem Kunden. Für die Entwicklung der Stapelplanung müssen Anpassungen im ERP System vorgenommen werden, weshalb der Autor sich über Entwicklungsprozesse mit dem ERP-Entwickler austauscht und Anforderungen übermittelt.

1.5. PROJEKTABGRENZUNG

Die Entwicklung der Stapelplanung, ist ein Teilprojekt bei der Einführung von ABAS ERP. Wie bereits im Abschnitt 1.3 Projektbegründung befindet sich der Kunde derzeit in einer Umstellungsphase. Das Alte ERP System soll durch ABAS ERP ersetzt werden. Der Autor beteiligt sich nur an der Entwicklung der Weboberfläche; nur einen Prozess in der Verkaufs-Abteilung des Kunden. Vom Autor werden keine Anpassungen im ERP System verlangt. Wie bereits in 1.4 Projektschnittstellen erläutert, wird diese Aufgabe vom jeweiligen ERP-Entwickler übernommen.

Für die technischen Entwicklungsdetails hält der Autor sich an das Lastenheft. Zu finden ist dies im Anhang unter A.6. Lastenheft (Auszug). Nicht im Lastenheft festgehalten ist die nötige Funktionsweise der Weboberfläche. Vom Kunden ist nicht gewünscht, dass einige sonst logische Eckdaten berechnet werden. Beispielsweise ist nicht verlangt eine maximale Höhe für eine Raster Zeile zu definieren, damit bei der Einlagerung die Ware auch in den Transporter passt. Auch die Gewichtsverteilung soll nicht durch die Applikation, sondern

durch den Verkaufsmitarbeiter bzw. durch den Anwender ermittelt werden. Dafür hat sich der Kunde entschieden, da sich Anhänger mit verschiedenen Formfaktoren auch teilweise ineinander Stapeln lassen und die Abbildung in der Weboberfläche in einem hohen Aufwand resultieren. Das Design der Applikation spielt im Projekt eine Nebensächliche Rolle und ist kein Teil des Lastenheftes. Die Oberfläche wird für den Gebrauch auf einem Monitor konzipiert. Die Stapelplanung könnte in Zukunft auch auf einem Tablett stattfinden.

2. PROJEKTPLANUNG

2.1. PROJEKTPHASEN

Nach Verordnung der IHK Karlsruhe, standen dem Autor für die Umsetzung des Projektes 70 Stunden zur Verfügung. Das Projekt wurde in verschiedene Phasen eingeteilt, um den Prozess der Softwareentwicklung abzudecken. Die Zeitplanung lässt sich der Tabelle 1: Grobe Zeitplanung entnehmen. Die jeweiligen Hauptphasenpunkte wurden in mehrere Unterphasen unterteilt und im Anhang in der Tabelle 4: Detaillierte Zeitplanung zusammengefasst.

Projektphase	Geplante Zeit
1. Planung	6h
2. Entwurf	3h
3. Implementierung	40h
a. Frontend	
b. Backend	
4. Test	8h
5. Einführung	2h
6. Dokumentation	11h
Summe	70h

Tabelle 1: Grobe Zeitplanung

2.2. RESSOURCEN

Die verwendeten Ressourcen für die Entwicklung der Stapelplanung lassen sich in verschiedene Varianten aufteilen. Neben den Hard- und Softwareressourcen wurden auch Personalressourcen verwendet. Eine Übersicht der benutzten Ressourcen lassen sich im Anhang unter A.2. Verwendete Ressourcen nachschlagen. Alle aufgelisteten wurden von dem Ausbildungsbetrieb gestellt und sind zum Teil kostenfrei. Bei der Verwendung von zusätzlichen Ressourcen wurde darauf geachtet, dass diese kostenfrei sind, beispielsweise ist die Verwendung der drag-and-drop-Library dem Projekt hinzugefügt wurden ist

Open-Source. Dadurch konnten die Projektkosten auf ein Minimum reduziert werden. Für Hilfestellungen während des Entwicklungsprozesses stand mir zusätzlich ein Web-Team Mitarbeiter zur Seite.

2.3. ENTWICKLUNGSPROZESS

Für das zu realisierende Projekt wurde der klassische Entwicklungsprozess gewählt. Dies definiert die Vorgehensweise, nach der die Umsetzung erfolgen soll. Der Autor hielt sich bei der Umsetzung zusätzlich an das Wasserfallmodell. Bei diesem Vorgehen wurden die Entwicklungsphasen schrittweise ausgeführt und die Phasen rückwirkend verifiziert. Während der Analyse der Projektanforderungen wurden Einzelheiten mit dem Projektleiter besprochen und das Konzept für die Umsetzung ausgearbeitet. Da der Projektleiter alle Anforderungen bereits im Lastenheft mit dem Kunden ausgearbeitet hatte, war eine Kommunikation mit diesem für die Einhaltung des magischen Dreiecks[1] nötig. Es sollten möglichst alle Anforderungen des Kunden umgesetzt werden, ohne dabei unnötige Funktionen hinzuzufügen. Letzteres musste auch beachtet werden, da es ein vom Kunden festgelegtes Budget für die Anwendung gibt. Anforderungen außerhalb des Lastenheftes (A.6. Lastenheft (Auszug)), sind nicht Teil der zu erbringenden Leistungen. Beispielsweise wurde für das Design der Weboberfläche kein Budget vereinbart und steht dem Autor während des Entwicklungsprozesses frei. Deshalb konnte in Abschnitt 2.1 Projektphasen verhältnismäßig wenig Zeit für die Analyse und Entwurfsphase eingeplant werden. Um die Entwicklungsphase zu bereichern, wurden Präsentationen mit dem Kunden und dem Projektleiter veranstaltet. Durch das resultierende Feedback konnten Anpassungen vorgenommen und das Projektziel erreicht werden.

[1] Terminus aus dem Projektmanagement: Verhältnis zwischen Kosten, Zeit und Leistung einhalten, um das Ziel zu erreichen.

3. ANALYSEPHASE

3.1. IST-ANALYSE

Wie bereits in Abschnitt 1.2 (Projektziel) erwähnt wurde, handelt es sich bei dem betreffenden Kunden um ein mittelständiges Unternehmen[2], der mit Autoanhängern handelt. Bevor der Kunde sich für ABAS entschieden hat, verwendete der Kunde bereits ein kleineres ERP System zur Verwaltung seiner Geschäftsprozesse. Auch die Stapelplanung wurde über dieses System realisiert. Die Verkaufsabteilung verwendet dieses System, um die Position der Autoanhänger in einem Transporter festzulegen. Dabei ist gewährleistet das Höhe und Gewicht durch den Mitarbeiter organisiert werden kann.

Für die Umsetzung einer neuen Stapelplanung wurden detaillierte Anforderungen im Lastenheft (A.6. Lastenheft (Auszug)) festgehalten. Dies ist insofern auch wichtig, um den Mitarbeitern einen möglichst nahtlosen Umstieg von der alten auf die neue Stapelplanung zu ermöglichen.

3.2. WIRSCHAFTLICHKEIT

3.2.1. MAKE-OR-BUY-ENTSCHEIDUNG

Als Alternative zur Entwicklung einer Stapelplanungs-Weboberfläche über ABAS, könnte der Kunde auf eine Lösung von einem Drittanbieter zurückgreifen. Da der Kunde jedoch spezielle Anforderungen für die Funktionsweise dieser Applikation hat, gilt zu vermuten das eine Drittlösung entweder zu viel oder zu wenig Funktionalitäten mit sich bringt. Eine Lösung von ABAS ist hingegen maßgeschneidert und basierend auf den Kundenanforderungen. Selbst wenn ein Drittanbieter kundenspezifische Software anbieten würde, wäre die Kommunikation zum ERP System nicht möglich. Zugrunde dessen kann vermutet werden das die Entwicklung seitens ABAS die beste Lösung für das Unternehmen wäre.

Alternativ könnte der Kunde auch auf die Funktion einer Stapelplanung verzichten. Der Verkaufs-Mitarbeiter müsste dann eine Planung während der

[2] Auch: KMU – Klein und mittlere Unternehmen

Einlagerung durchführen. Dadurch würde dieser Geschäftsprozess anfällig für Fehler gemacht werden. Nach jetziger Einschätzung gibt es keine alternative Lösung, um die Daten für den Verkauf zusammenzufassen und eine Planung durchführen zu können.

Neben den offensichtlichen Vorteilen einer Entwicklung seitens ABAS, wird dem Kunden für die Zukunft auch eine Erweiterbarkeit und Wartbarkeit gewehrleistet.

3.2.2. PROJEKTKOSTEN

Um die Kosten für die Entwicklung der Weboberfläche zu berechnen, setzen wir einen internen Stundensatz von 140 € an. Für die Umsetzung des Projektes wird nur die reine Arbeitsleistung berechnet. Ressourcen wie Strom, Rechner und Lizenzen werden in die Kostenrechnung nicht mit aufgenommen. Dokumentation und Code-Review werden nicht als Kosten aufgefasst.

Entwickler	Stunden	Tätigkeit	Lohn	Kosten
Arthur Schimpf	56	Entwicklung	140 €	7840 €
Florian Cords	10	Projektarbeit	140 €	1400 €
			Summe	**8240 €**

Tabelle 2: Kostenkalkulation

3.3. ANWENDUNGSFÄLLE

Im Allgemeinen dient die Applikation für die Anwendung im Verkaufsbereich des Kunden. Es werden Prozesse für die Kommissionierung abgenommen. Datensätze müssen nicht manuell gepflegt werden, sondern werden durch die Weboberfläche intuitiv gepflegt.

Die Applikation deckt zusätzlich einige spezifizierte Anwendungsfälle für den Benutzer ab. Die Verwendung soll für Fehlereingaben vom Benutzer nicht anfällig sein. In der Stapelplanung gilt deshalb von Anfang an zu beachten, dass es eine Möglichkeit gibt einsortierte Produkte in ihrer Position anzupassen bzw. Fehler zu korrigieren.

Alternativ ist es eingebaut das sich ein Produkt im Planungsraster, auch einfach in eine andere Zelle verschoben werden kann. Damit zwei oder mehr Produkte sich nicht an derselben Position befinden, ist bei diesem Prinzip auch zu beachten, falls sich bereits ein Produkt in der Zelle befindet, dies zurück in die Liste geschoben wird.

Für die Software wäre auch möglich gewesen, das Gewicht der Ladung zu beachten und bei der Lagerung Möglichst ein gleichmäßiges Gewicht im Anhänger zu gewährleisten. Schwere Ladung wäre dabei nach Empfehlungssystem nach unten zu lagern. Dementsprechend leichte Ladung nach oben. Die könnte als zukünftige Anpassung des Kunden beachtete werden.

Eine weitere mögliche Anpassung für die Stapelplanung wäre es gewesen, die Höhe der Ladung zu beachten, so dass diese auch Ohne Probleme mit der Größe des Anhängers übereinstimmen kann. Auch dies könnte eine mögliche Kundenanforderung für die Zukunft sein.

3.4. QUALITÄTSSICHERUNG

Um die Qualität der Anwendung zu wahren, werden Maßnahmen ergriffen. Während der Entwicklung wird darauf geachtet, dass der Code für zukünftige Anpassungen gut lesbar ist. Dafür werden verschiedene Methoden, welche sich am Clean Code orientieren angewendet. Durch die modulare Struktur von Shopfloor wird die vertikale Erweiterung gewährleistet.

4. ENTWURFSPHASE

4.1. ZIELPLATTFORM

Die Zielplattform für die Shopfloor Applikation ist der Browser, über welchen die Stapelplanung aufgerufen wird. Das Benutzte Medium werden Computer sein. In Zukunft soll die Stapelplanung eventuell auch für den Gebrauch auf einem Tablet möglich sein.

4.2. ARCHITEKTURDESIGN

Shopware hat eine bereits vordefinierte Full-Stack-Architektur. Es wird zwischen front- und backend unterschieden, wobei beide Bereiche innerhalb der Applikation miteinander Daten austauschen. Das backend ist für die Kommunikation mit dem ERP System und die Verarbeitung von Daten zuständig. Wie bereits im Projektschnittstellen-Abschnitt der Dokumentation erwähnt, werden diese Daten aufbereitet, manipuliert und im Anschluss an das frontend für die weitere Verarbeitung gesendet. Der frontend Teil dient als Datenaustausch zwischen Benutzer und Applikation. Daten werden vom backend empfangen und in HTML Code für den Anwender sichtbar gemacht. Der Benutzer hat die Möglichkeit diese Daten interaktiv zu manipulieren. Sollten sich Daten ändern, Verläuft die Kommunikationskette der Daten Rückwärts. Um den Datenfluss nachvollziehen zu können befindet sich im Angang Abbildung 6: Datenaustausch zwischen ABAS und Shopfloor) auf Seite iv.

4.3. BENUTZEROBERFLÄCHE

Für die Benutzeroberfläche wird das Framework VueJS verwendet, ein JavaScript Framework zur Entwicklung von Benutzeroberflächen. Bei VueJS werden sogenannte Komponenten verwendet. Das sind Dateien, welche eine bestimmte Syntax verwenden um HTML, JavaScript und CSS in einem vereinen. Sie ermöglichen während der Entwicklungsphase die modulare Programmierung. Der frontend Bereich von Shopfloor ermöglicht es, neben der Verwendung von VueJS auch das Routing von Komponenten zu URLs in der Applikation. In einer vorgesehenen Datei können Routen definiert und modifiziert werden. Für die Planung der Benutzeroberfläche wurde sich am

Lastenheft und dem zugehörigen Mockup orientiert (Abbildung 7: Ursprüngliches Mockup). Während der Planung, der zu entwickelten Benutzeroberfläche wurde zusätzlich eine Layout-Planung (Abbildung 8: Layout-Planung) entwickelt und vom Projektleiter abgesegnet. Das ursprüngliche Mockup wurde überarbeitet und anstatt zwei Gitter, wurde sich mit dem Projektleiter auf ein Hauptraster im linken Teil und eine Liste im rechten Teil der Oberflächendarstellung geeinigt.

4.4. GESCHÄFTSLOGIK

Für die Geschäftslogik verwendet Shopfloor Java Spring Boot. Dies ermöglicht es unter Verwendung von JVM Applikationen zu erstellen. Durch die Eigenschaften dieses Frameworks wird es ermöglicht Web-Applikationen und Micro Services zu entwickeln. In Shopfloor wird Spring Boot für die Kommunikation zwischen ABAS ERP und Applikation, sowie back- und frotnend verwendet. Wie bereits im Abschnitt 1.4 (Projektschnittstellen) erwähnt, wird dies Hauptsächlich über EDP und HTTP ermöglicht.

4.5. MAßNAHME ZUR QUALITÄTSSICHERUNG

Während der Durchführung des Projektes werden Maßnahmen zur Qualitätssicherung durchgeführt. Um die Sicherheit für den Echtmandanten beim Kunden zu gewährleisten, gibt es für die Entwicklung einen speziellen Mandanten. Dies versichert das keine echten Daten manipuliert werden können.

Für die lokale Entwicklung wurde ein Repository über Git erstellt und über Bitbucket verwaltet. Für Neuerungen am Repository wurde ein Feature-Branch erstellt. Diese werden über Pull-Requests in den aktuellen Stand gemerged.

Für die Arbeit mit Git wurden Conventional-Commits [3]verwendet.

[3] Conventional-Commits Homepage: https://www.conventionalcommits.org/en, Stand: 24.04.2022

5.1. IMPLEMENTIERUNGSPHASE

5.1. VORBEREITUNG UND INSTALLATION

Bevor die Entwicklung beginnen kann, muss zunächst das Framework installiert, die Entwicklungsumgebung angepasst und eine Konfigurationsdatei angeglichen werden. Zunächst wird sich die aktuelle Version von Shopfloor heruntergeladen. Für die Sicherung solcher Programmstrukturen verwendet ABAS die Versionsverwaltung BitBucket. Aus diesem Dienst kann die neuste Version als ZIP-Datei herunterladen und lokal auf dem System entpackt werden. Im nächsten Schritt wird in der verwendeten Entwicklungsumgebung das Projekt aufgesetzt. Die Projekt-Struktur wird dafür auf eine mit Shopfloor kompatible JDK-Version gesetzt. Anschließend wird der Maven-Task für die Installierung ausgeführt. Dieser Prozess installiert die Anwendung und ladet sich die benötigten Java-Librarys herunter. Zusätzlich werden die im frontend definierten Abhängigkeiten heruntergeladen. Um die Installation nun abzuschließen, werden in der Shopfloor-Konfigurationsdatei die Verbindungsdaten zum ABAS ERP System eingetragen. Für die Entwicklung wird die IP-Adresse und der jeweilige Port des ERP-Mandanten verwendet. Um die Applikation für die Entwicklung zu starten, führen wir die Main-Funktion im backend Bereich aus. Parallel kann im frontend Bereich über Node VueJS gestartet werden. Front- sowie backend geben dem Entwickler während der Entwicklung Informationen bzw. Logs in der Konsole aus. Für die Entwicklung wird die Applikation über *localhost:8080* erreichbar sein.

5.2. BENUTZEROBERFLÄCHE

Für Anpassungen in der Benutzeroberfläche, werden Vue-Komponenten im frontend Bereich erstellt, angepasst und verwendet. Für die Stapelplanung wird eine Route benötigt, die für den Benutzer aufrufbar ist. Über die Tourenplanung soll die Stapelplanung erreichbar gemacht werden. Um den jeweiligen Touren eine eindeutige Zuordnung zu verschaffen, legen wir in der entsprechenden Datei eine dynamische Route an. Die Datei router.js (Abbildung 1: Router Konfigurationsdatei (Auszug)) ermöglicht dem Entwickler die Router der Applikation bekannt zu machen. Zusätzlich wird der definierten Route eine Vue-

Komponente zugewiesen. Diese Komponente wird beim Aufrufen der jeweiligen ULR für den Anwender gerendered. Der dynamische Anteil hierbei wird über die „tour_id“ ermöglicht, welche durch eine Tourenplanungs-ID aus dem ERP System ersetzt werden kann und als Konstante in der Vue-Komponente aufgerufen werden kann.

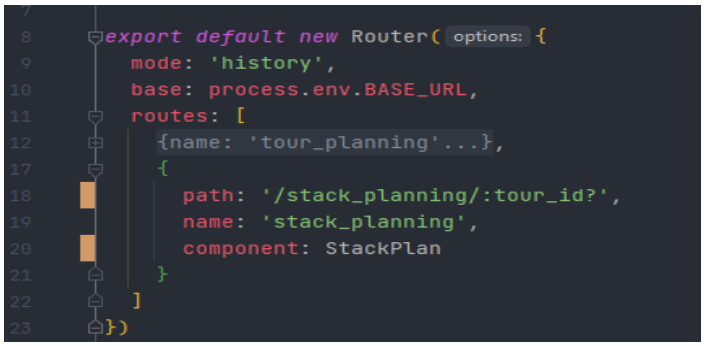


Abbildung 1: Router Konfigurationsdatei (Auszug)

Die in der „stack_planning“-Route definierten Vue-Komponente muss im Anschluss erstellt werden und dem Namen aus dem router.js Eintrag gleichen. Hier können wir über VueJS Anpassungen vornehmen (Abbildung 2: Ansicht leerer Vue-Komponente): Im „template“-Tag wird definiert, wie die Oberfläche für den Anwender strukturiert ist, im „script“-Tag wird die Funktion der Applikation festgelegt und das Aussehen einzelner HTML-Elemente wird im „style“-Tag vorgenommen.

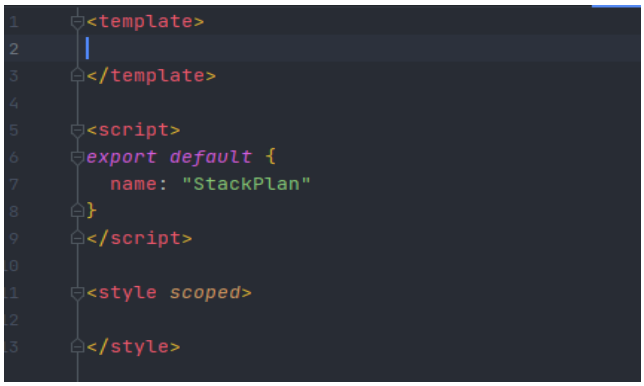


Abbildung 2: Ansicht leerer Vue-Komponente

Es besteht auch die Möglichkeit bereits bestehende Komponenten zu verwenden. Shopfloor stellt hierfür bereits einige nützliche Komponenten zur Verfügung, jedoch nur grundlegende Komponenten wie beispielsweise eine Login-Funktion zur Authentifizierung bei ABAS. Auch Komponenten von Anderen

Entwicklern können über Node.Js in die VueJS Umgebung hinzugefügt werden. Für die drag-and-drop-Funktion der Anwendung, verwenden wir Drag- und Drop-Komponenten. Drag-Komponenten ermöglichen es dem Benutzer gezogen zu werden. Als Pendant wird die Drop-Komponente dafür verwendet Drag-Komponenten, nach dem Loslassen dieser in sich zu speichern. Für die Oberfläche wird eine Rasteransicht auf der linken Seite der Gesamtübersicht erstellt. Jedes Feld besteht dabei aus Drop-Komponenten. Auf der rechten Seite wird eine Liste aus Drop-Komponenten für die Artikel erstellt. Die aus dem backend erhaltenen Artikeldaten werden dafür im Listenbereich iteriert und ausgegeben. Um den Anforderungen im A.6. Lastenheft (Auszug)gerecht zu werden, muss während dieser Iteration geprüft werden, ob ein Artikel bereits eine Koordinate zugeordnet bekommen hat. Sollte dies der Fall sein, wird er automatisch ins Planungsraster einsortiert. Für diese Funktion verwenden wir im VueJS-Hook „mounted“ welcher uns ermöglicht nach dem Rendering der Oberfläche Funktionen auszuführen. In diesem Teil der Komponente wird der Artikel im Falle einer bestehenden Koordinate dem jeweiligen Feld im Planungsraster zugewiesen.

Für die optische Anpassung der Oberfläche verwendet der Autor die CI-Richtlinien von ABAS. Dadurch werden optische Details definiert. Für die Bereichseinteilung der beiden Hauptbestandteile wird flexbox verwendet. Des Weiteren wird für das Planungsraster gridbox verwendet.

Für die Funktionsweise der Applikation, werden Benutzer-Events abgefangen und in Programmlogik verarbeitet. Sollte ein Artikel in das Planungsraster gezogen und losgelassen werden, wird die Funktion „dropCart“ ausgeführt. Hier werden alle in 3.3 (Anwendungsfälle) definierten Fälle aufgefasst und verarbeitet. Im Anhang unter A.5. Benutzerhandbuch) findet sich eine Abbildung der Benutzeroberfläche.

Für die Hauptkomponente findet sich ein Auszug des programmierten Codes im Anhang (A.7. Benutzeroberflächen Code (Auszug)) auf Seite ix.

5.3. GESCHÄFTSLOGIK

Anpassungen im backend werden benötigt, um zu definieren, welche Daten von ABAS ERP angefragt werden sollen und wie diese Daten verarbeitet und an das frontend übertragen werden. Für die Anfrage an das ERP System wird ein neues Bean in der Spring Boot Umgebung definiert. In diesem werden Parameter angegeben, die über EDP vermittelt werden. Es wird festgelegt, welche Kopf- und Tabellenfelder zurückgegeben werden sollen.

```
@Bean
InfosystemService stackPlanningInfosystem() throws EdpServiceException {
    List<String> headFields = Arrays.asList("ytoun", "bstart");
    List<String> tableFields = Arrays.asList("ytartnum", "ytauftrnum", "ytartbez", "ytverw", "ytgewicht", "ytlaenge",
    AliveBeanInfo aliveBeanInfo = new AliveBeanInfo(headFields, tableFields, Collections.emptyList());
    aliveBeanInfo.setLazyLoad(false);

    return edpInfosystemServiceFactory.create( infosystem: "STAPELPLAN", aliveBeanInfo);
}
```

Abbildung 3: Bean Konfiguration

Um die Daten zu empfangen wird ein Controller angelegt. In diesem wird festgelegt, wie mit den Empfangenen Daten verfahren werden soll. Da die Daten lediglich an das frontend übermittelt werden soll, werden keine weiteren Anpassungen benötigt. Shopfloor übernimmt den Rest der Kommunikation an das frontend. Die Daten werden in der Variable „stackPlanning“ für die Vue-Komponente erreichbar gemacht. Wir verwenden diese beispielsweise für die Artikel-Liste, um über die Tabellenfelder zu iterieren und die Artikel zu rendern, zu sehen unter A.7. Benutzeroberflächen Code (Auszug)) im „template“-Tag.

```
@RequestMapping("/api/stack_planning")
@RestController
public class StapelplanController extends SingleInfosystemRestControllerTemplate {

    public StapelplanController(@Autowired InfosystemService stackPlanningInfosystem) {
        super(stackPlanningInfosystem);
    }
}
```

Abbildung 4: Controller Konfiguration

6. ABNAHMEPHASEN

Nachdem die Anwendung fertig gestellt war, konnte diese dem Fachbereich zur Endabnahme vorgelegt werden. Aufgrund der klassischen Softwareentwicklungsmethode wurde den Fachbereichen nach jeder größeren Änderung die aktuelle Version der Anwendung präsentiert. Für den Kunden wurden separate Termine vereinbart, in welchen die Details der Applikation besprochen wurden. Dadurch konnten sich die Projektleitung und der Kunde im Vorfeld bereits mit der Oberfläche und der Funktionsweise der Applikation vertraut machen. Auf Anregungen und Kritik konnten, während diesem Vorgehen bereits frühzeitig reagiert werden (2.3 Entwicklungsprozess), so dass sich bei der Endabnahme keine Probleme oder Hindernisse ergaben. Der Einführung stand somit nichts mehr im Wege. Vor dem Deployment wurde zur Qualitätssicherung zusätzlich zur Abnahme durch den Kunden ein Code Review durch einen anderen Entwickler durchgeführt.

Um den Zugriff auf die Anwendung für den Kunden zu ermöglichen, wurde zunächst eine kompilierte Version der Anwendung erstellt. Diese wurde im Anschluss per FTPS auf den Server des Kunden, in eine dafür vorgesehene Umgebung übertragen. Über ein Bash-Skript startet die Applikation und wird dem Kunden Lokal über die vorher eingetragene IP-Adresse zugänglich gemacht. Wie bereits erwähnt wurde, waren die Kunden bereits mit der Anwendung vertraut, weshalb keine zusätzliche Benutzerschulung erforderlich war.

7. DOKUMENTATION

Für die Dokumentation der Stapelplanungs-Applikation werden zwei Bestandteile benötigt: die Projektdokumentation und das Benutzerhandbuch. In der Projektdokumentation beschreib der Autor die einzelnen Phasen, die während der Umsetzung des Projektes durchlaufen wurden.

Das Benutzerhandbuch enthält Informationen über den Aufbau und die Funktionsweise der Anwendung. Es soll den Fachbereichen als Anhaltspunkt für Nachfragen zur Verfügung stehen und zur Einarbeitung neuer Mitarbeiter in die Anwendung dienen. Ein Auszug aus dieser Dokumentation befindet sich im Anhang 0 A.5. Benutzerhandbuch auf Seite v.

8. FAZIT

8.1. SOLL-/IST-VERGLEICH

Rückblickend konnten alle zuvor festgelegten Anforderungen erfüllt werden. Der zu Beginn des Projektes im Abschnitt 2.1(Projektphasen) erstellte Projektplan konnte eingehalten werden. In der Tabelle 3: Soll-/Ist-Vergleich werden die Zeiten, welche tatsächlich für die einzelnen Phasen benötigt wurden, der zuvor eingeplanten Zeit gegenübergestellt. Es ist zu erkennen, dass nur geringfügig von der Zeitplanung abgewichen wurde. Die sich daraus ergebenen Differenzen konnten untereinander kompensiert werden, sodass das Projekt in dem von der IHK festgelegten Zeitrahmen von 70 Stunden umgesetzt werden konnte.

Projektphasen	Soll	Ist	Differenz
Planung	6h	6h	0
Entwurf	3h	4h	+1h
Implementierung	40h	40h	0h
a. Frontend	25h	26h	+1h
b. Backend	15h	13h	-2h
Test	8h	8h	0
Einführung	2h	2h	0
Dokumentation	11h	11h	0
Gesamt	**70h**	**70h**	**0h**

Tabelle 3: Soll-/Ist-Vergleich

8.2. RETROPERSPEKTIVE

Im Zuge des Projektes konnte der Autor wertvolle Erfahrungen bzgl. Der Planung und Durchführung von Projekten sammeln. Dabei wurde besonders deutlich, von welcher großen Bedeutung stetige Kommunikation untereinander und Rücksprachen mit den Fachbereichen für eine erfolgreiche Projektumsetzung sind. Außerdem konnten neue Erkenntnisse in Bezug auf das Einbinden und Nutzen von Frameworks und Libraries gewonnen werden. Die drag-and-drop-library erwies sich beispielsweise als sehr hilfreich, da die Entwicklung der Benutzeroberfläche vereinfachte und der Autor sich somit auf

die wesentlichen Funktionalitäten konzentrieren konnte. Abschließend kann man sagen, dass die Realisierung des Projektes nicht nur einen Mehrwert für die Fachbereiche bietet, sondern auch für den Autor.

8.3. AUSBLICK

Obwohl alle definierten Anforderungen realisiert werden konnten, können in Zukunft dennoch neue Anforderungen definiert bzw. Erweiterungsvorschläfe entwickelt werden. Für die Übersichtlichkeit könnte beispielsweise eine Suchfunktion in der Artikelliste programmiert werden, um dadurch einen noch schnelleren Zugriff zu ermöglichen. Denkbar wäre es auch, in Zukunft weitere Prozesse über die Web-Applikation zu verwalten. Aufgrund des im Abschnitt 4.2 (Architekturdesign) beschriebenen modularen Aufbaus des Projektes können solche Anpassungen bzw. Erweiterungen sehr einfach vorgenommen werden. Die Modularität der Anwendung ermöglicht somit eine gut Wartbarkeit und Erweiterbarkeit.

A ANHANGVERZEICHNIS

A.1. DETAILLIERTE ZEITPLANUNG

1. Planung			6h
a. Einarbeitung ins Einführungskonzept	3h		
b. Absprache mit dem Projektleiter	3h		
2. Entwurf			3h
a. Installation und Anpassung Shopfloor	2h		
b. Aufsetzen Git-Repository	1h		
3. Implementierung			40h
a. Frontend		25h	
i. Installation der benötigten Libraries	2h		
ii. Routing der benötigten Seiten	4h		
iii. Benutzeroberfläche	8h		
iv. Anpassungen Schnittstelle zum Backend	8h		
v. CSS-Anpassungen / Layout	3h		
b. Backend		15h	
vi. Request Handling / Kommunikation zu ABAS ERP	10h		
vii. Kommunikation zum Frontend	5h		
4. Test			8h
a. Code Review	1h		
b. Präsentation und Einführung	3h		
c. Funktion	4h		
5. Einführung			2h
a. Installation auf dem Firmennetzwerk	2h		

6. Dokumentation			11h
a. Anwenderdokumentation	1h		
b. Projektdokumentation	10h		
Summe			70h

Tabelle 4: Detaillierte Zeitplanung

A.2. VERWENDETE RESSOURCEN

Hardware

- Notebook
- Arbeitsplatz im Standort Karlsruhe

Software

- Betriebssystem: Windows 10 Enterprise
- VPN: Open VPN
- Java/JavaScript Entwicklungsumgebung: IntelliJ von Jet Brains
- Full Stack App: Shopfloor
    - Java Framework: Spring Boot
    - Benutzeroberfläche: VueJS Framework
        - Drag-and-drop-Library

Personal

- Umsetzung des Projektes: Entwickler (Autor der Projektdokumentation)
- Projektarbeiten: Projektleiter von ABAS
- ERP-Anpassungen: ABAS ERP-Entwickle

A.3. DIAGRAMME

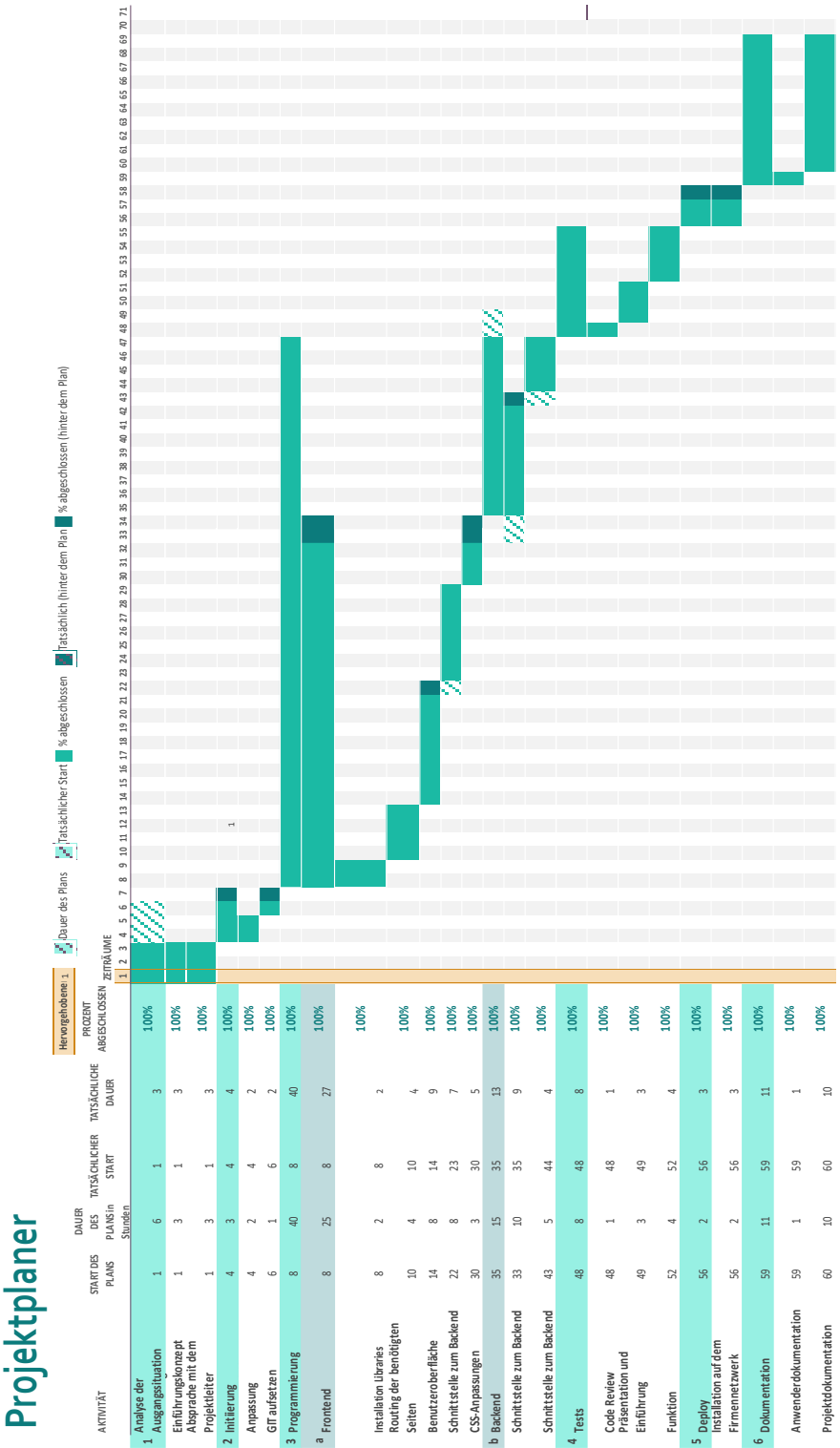


Abbildung 5: Gantt-Diagramm

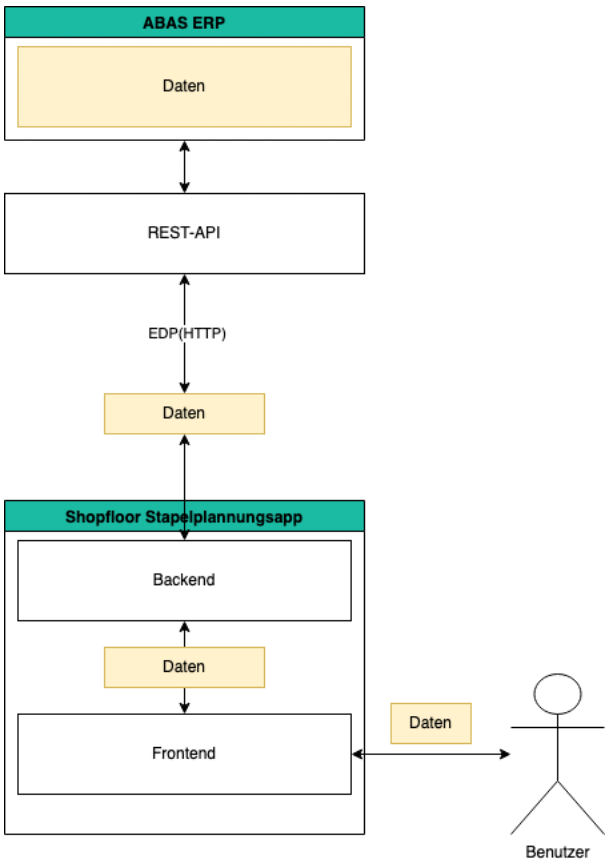


Abbildung 6: Datenaustausch zwischen ABAS und Shopfloor

A.4. MODELLE

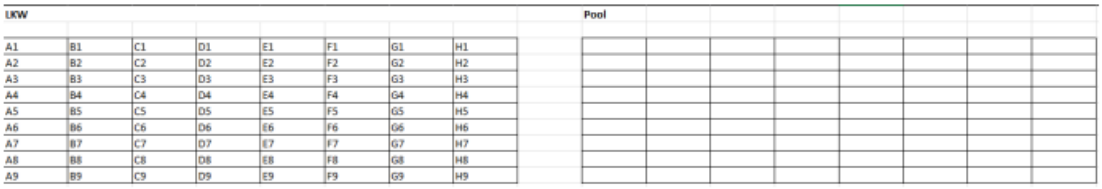


Abbildung 7: Ursprüngliches Mockup

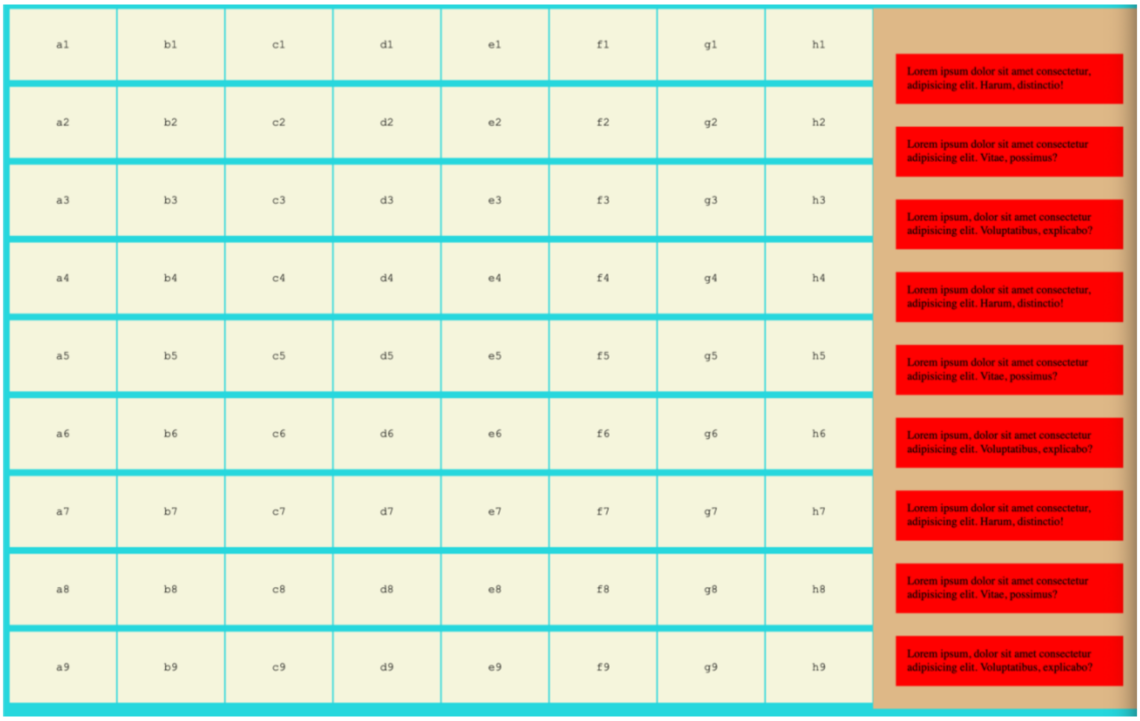


Abbildung 8: Layout-Planung

A.5. BENUTZERHANDBUCH

Ausschnitt aus der Benutzerdokumentation:

Die Verwendung der Stapelplanung beginnt, nachdem man sich über die ABAS Tourenplanung eine Tour ausgesucht und über das Fenster „Stapelplanung" geöffnet hat. Nachdem die Daten geladen sind, öffnet sich die Stapelplanübersicht.

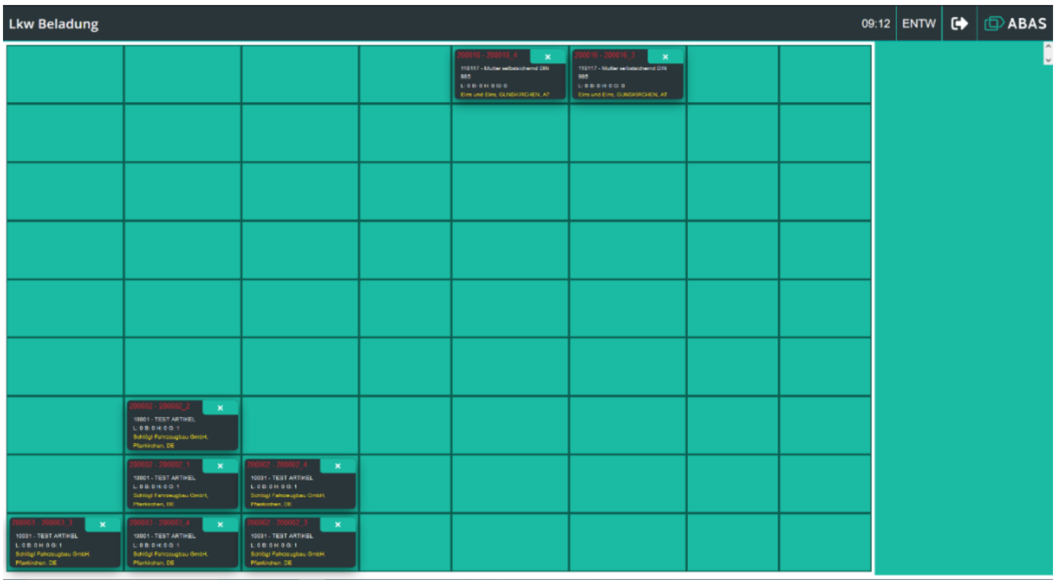


Abbildung 9: Screenshot der Webanwendung

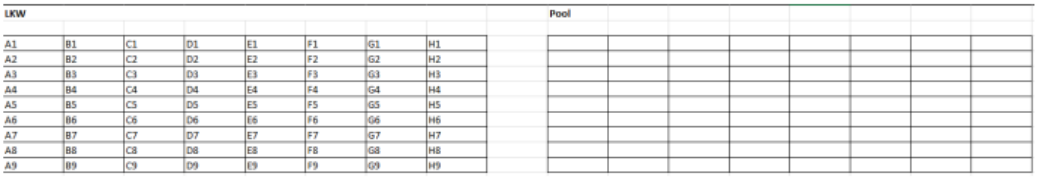
Zu erkennen ist die Ladezeit durch eine bildschirm-einnehmende Animation. Während dieses Prozesses werden die Artikel-Koordinaten von der Applikation ausgelesen und automatisch dem zugehörigen Feld im Planungsraster zugewiesen. Alle Artikel, denen keine Koordinate zugewiesen wurde, sind in der Listenansicht zu sehen. Der Zuordnungs-Prozess erfolgt über das drag-and-drop-Verfahren. Der gewählte Artikel wird aus der Liste in das Planungsraster gezogen. Beim Loslassen wird der Artikel de, jeweiligen Feld eingeordnet. Um zu erkennen, in welches Feld der gezogene Artikel während dem Loslassen einsortiert wird, leuchtet das jeweilige Feld weiß auf. Nachdem die Sortierung abgeschlossen ist, kann die Stapelplanung geschlossen werden. Die Koordinaten konnten erfolgreich zugeordnet werden und die Stapelplanung ist somit vollendet. Wird ein Artikel aus der Liste, auf ein bereits mit einem Artikel befüllten Feld gezogen, wird der ursprüngliche Artikel zurück in die Liste sortiert und der gewählte in das Feld sortiert. Dieses Prinzip findet auch statt, wenn ein Artikel aus dem Planungsraster auf ein bereits befülltes Feld gezogen wird.

A.6. LASTENHEFT (AUSZUG)

In der Stapelplanung wird das Stapeln auf LKWs vorgeplant. Eine Tourenplanung entspricht dabei einem LKW.

1 Stapelbild aus dem Altsystem

Die Webseite besteht aus 2 Bereichen.



Im rechten Bereich „Pool“ werden alle Auftragspositionen als Kästchen gesammelt und dargestellt. Eine Auftragsposition entspricht dabei einem Kästchen. - Im linken Bereich befindet sich der „LKW“ hier werden die Kästchen auf einen bestimmten Platz verschoben.

- Die Anzahl der Kästchen orientiert sich an den Auftragspositionen (und damit automatisch auch an verschiedenen Chargen – da bei Chargenpflicht die Menge = 1 im Auftrag ist.).
- Das LKW-Koordinatensystem ist 9 Kästchen hoch und 8 breit. Damit dürfen einer Tour nur maximal 72 Auftragspositionen zugeordnet werden. (Dies muss per FOP sichergestellt werden)
- Die Koordinaten werden dann in die Auftragsposition geschrieben (z.B. A1, C8 – maximal jedoch H9)

A1	B1	C1	D1	E1	F1	G1	H1
A2	B2	C2	D2	E2	F2	G2	H2
A3	B3	C3	D3	E3	F3	G3	H3
A4	B4	C4	D4	E4	F4	G4	H4
A5	B5	C5	D5	E5	F5	G5	H5
A6	B6	C6	D6	E6	F6	G6	H6
A7	B7	C7	D7	E7	F7	G7	H7
A8	B8	C8	D8	E8	F8	G8	H8
A9	B9	C9	D9	E9	F9	G9	H9

- Die Kästchen können jederzeit vom LKW zum Pool und zurück verschoben werden. Wird auf den LKW verschoben werden die Koordinaten in die Auftragsposition geschrieben, zurück werden sie wieder entfernt.
- Ein Kasten (also eine Auftragspositionsnummer) enthält folgende lesbare Informationen:
    - Auftragsnummer
    - Verwendung
    - Artikelnummer + Bezeichnung
    - Kundennamen des Warenempfängers
    - Gewicht aus Chargendatensatz (COC) - Wenn leer dann aus dem Artikelstamm Sachmerkmale
    - Länge, Breite und Höhe aus COC- Wenn leer dann aus dem Artikelstamm Sachmerkmale

- Wird der Tourendatensatz geladen wird das Stapelbild automatisch anhand der Koordinaten aufgebaut.

A.7. BENUTZEROBERFLÄCHEN CODE (AUSZUG)

```
<template>
  <div>
    <ShopfloorHeader></ShopfloorHeader>
    <div id="App" v-if="!isEmptyTour">
      <div id="drop-container">
        <img v-if="isLoading" id="loading-img" alt="Loading icon" src="../assets/loading.gif">
        <div v-else v-for="(arrValue, arrIndex) in contract_storage" :key="arrIndex" class="row">
          <drop v-for="(value, index) in arrValue" :key="index" class="cell"
                @drop="dropCart($event,value)">
            <div v-if="value.data.zn" class="readjustable-item">
              <div class="readjustable-item-header-bar">
                <span class="item-name">{{ value.data.ytauftrnum }} - {{
                    value.data.ytverw
                  }}</span>
                <font-awesome-icon @click="rejoinItemToList(value)" icon="times" class="times"></font-
awesome-icon>
              </div>
              <drag go-back class="redrag-item" :data="value">
                <div class="redrag-item-content">
                  <span>{{ value.data.ytartnum }} - {{ value.data.ytartbez }}</span>
                  <span>L: {{ value.data.ytlaenge }} B: {{ value.data.ytbreite }} H: {{
                      value.data.ythoehe
                    }} G: {{ value.data.ytgewicht }}</span>
                  <span class="destination">{{ value.data.ytkundname }}</span>
                </div>
              </drag>
            </div>
          </drop>
        </div>
      </div>
      <div v-if="!isLoading" id="drag-list-container">
        <div class="drag-container-items">
          <div v-for="(item,status) in stackPlanning.table" :key="status.index">
            <drag v-if="!isOutOfList.includes(item.zn)" go-back :data="item" class="drag-item">
              <div class="drag-item-content">
                <span class="item-name">{{ item.ytauftrnum }} - {{ item.ytverw }}</span>
                <span>{{ item.ytartnum }} - {{ item.ytartbez }}</span>
                <span>Länge: {{ item.ytlaenge }} Breite: {{ item.ytbreite }} Höhe: {{
                    item.ythoehe
                  }} Gewicht: {{ item.ytgewicht }}</span>
                <span class="destination">{{ item.ytkundname }}</span>
              </div>
            </drag>
          </div>
        </div>
      </div>
    </div>
    <div v-else>
      <div class="screen-center">
        Die Tour wurde ohne Daten geladen. <br>
        Ohne Daten kann keine Stapelplanung vollzogen werden.
      </div>
    </div>
  </div>
</template>
```

```
<script>

import ShopfloorHeader from "@/modules/shopfloor/components/ShopfloorHeader";
import {Drag, Drop} from "vue-easy-dnd";
import {mapActions, mapState} from "vuex";

export default {
  name: "LoadingApplication",
  components: {
    ShopfloorHeader,
    Drag,
    Drop
  },
  computed: {
    ...mapState({
      stackPlanning: state => state.shopfloor.stackPlanning
    })

  },
  created() {
    this.getMandant();
  },
  data: function () {
    return {
      contract_storage: [...],
      isOutOfList: [],
      coordinate_helper_vector: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'],
      infoItem: '',
      isLoading: false,
      isEmptyTour: false
    }
  },
  async mounted() {
    if (this.$route.params.tour_id !== undefined) {
      this.isLoading = true;
      await this.updateStackPlanning({
        'head': {
          'ytour': this.$route.params.tour_id,
          'bstart': '1'
        }
      }).then(() => {
        if (this.stackPlanning.table.length > 0) {
          this.stackPlanning.table.map((item) => {
            if (item.ytkoordinate !== '') {
              this.isOutOfList.push(item.zn);
              this.contract_storage[(item.ytkoordinate[1] -
1)][this.coordinate_helper_vector.indexOf(item.ytkoordinate.toLowerCase()[0])].data = item;
            }
          })
        } else {
          this.isEmptyTour = true;
        }
        this.isLoading = false;
      })
    } else {
      this.isEmptyTour = true;
    }
  },
  methods: {
    ...mapActions([
      'updateStackPlanning',
      'getMandant'
    ]),
    dropCart(draggedEntity, field) {
      if (this.droppedIntoSameField(draggedEntity, field)) return;

      if (this.foundInGrid(draggedEntity)) {
        if (this.droppedOnContainingField(field)) this.kickItemBackToList(field);

        this.contract_storage[field.y][field.x].data = this.contract_storage[draggedEntity.data.y]
[draggedEntity.data.x].data;
        this.contract_storage[draggedEntity.data.y][draggedEntity.data.x].data = {};
      } else {
        if (this.droppedOnContainingField(field)) this.kickItemBackToList(field);

        this.contract_storage[field.y][field.x].data = draggedEntity.data;
        this.isOutOfList.push(draggedEntity.data.zn);
      }
      this.$forceUpdate();
    },
    droppedIntoSameField(draggedEntity, field) {
      return draggedEntity.data.x === field.x && draggedEntity.data.y === field.y
    },
    foundInGrid(draggedEntity) {
      return draggedEntity.data.data;
    },
    droppedOnContainingField(field) {
      return this.contract_storage[field.y][field.x].data.zn;
    },
    kickItemBackToList(field) {
      this.isOutOfList.splice(this.isOutOfList.indexOf(this.contract_storage[field.y][field.x].data.zn),
1);
    },
    rejoinItemToList(fieldEntity) {
      this.isOutOfList.splice(this.isOutOfList.indexOf(fieldEntity.data.zn), 1);
      this.contract_storage[fieldEntity.y][fieldEntity.x].data = {};
    }
  }
}
</script>

<style scoped lang="scss">...</style>
```