

**Bachelorarbeit**  
**HS15 Studiengang Informatik**

Verbesserung der Benutzererfahrung der Kund-  
schaft eines internationalen Velokuriers

---

<b>Autor</b>	Nicoals Roos
<b>Hauptbetreuung</b>	Beat Seeliger
<b>Experte</b>	Jaime Oberle
<b>Industriepartner</b>	ImagineCargo GmbH
<b>Datum</b>	21. Oktober 2016

---

## **Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering**

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

**Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.**

Ort, Datum:

Unterschriften:

.....

.....

.....

.....

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Abstract bzw. dem Management Summary mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>5</b>
1.1. Zusammenfassung . . . . .	5
1.2. Ausgangslage . . . . .	5
1.3. Ziel der Arbeit . . . . .	5
1.4. Aufgabenstellung . . . . .	5
1.5. Erwartete Resultate . . . . .	6
<b>2. Recherche</b>	<b>7</b>
2.1. Kurier-, Express- und Paketdienstbranche . . . . .	7
2.1.1. Schweiz . . . . .	7
2.1.2. Europa . . . . .	8
2.2. Standards und Software . . . . .	9
2.2.1. EDIFACT . . . . .	9
2.3. LoBo . . . . .	10
2.3.1. API . . . . .	10
<b>3. Ist Analyse</b>	<b>14</b>
3.1. ImagineCargo . . . . .	14
3.1.1. Prozesse . . . . .	14
3.2. Marktanalyse . . . . .	16
3.2.1. Deliveo . . . . .	16
3.2.2. eCourier . . . . .	17
3.2.3. Fazit . . . . .	17
<b>4. User Centered Design</b>	<b>19</b>
4.1. Anwendung . . . . .	19
4.1.1. Team . . . . .	20
4.1.2. Grundlegende Recherche der Domäne . . . . .	20
4.1.3. Interviews mit Stakeholders . . . . .	20
4.1.4. Interviews mit Endbenutzer . . . . .	20
4.1.5. Personas . . . . .	21
4.1.6. Stories erstellen und Anforderungen definieren . . . . .	21
4.1.7. Wireframes und Interaktionsabläufe . . . . .	21
4.1.8. Prototyp . . . . .	21
4.2. Interviews . . . . .	22
<b>5. Anforderungsanalyse</b>	<b>23</b>
5.1. Einleitung . . . . .	23
5.2. Stakeholders . . . . .	23
5.3. Personas . . . . .	23
5.3.1. Einleitung . . . . .	23
5.3.2. Mara Hürlimann . . . . .	23
5.3.3. Peter Elsener . . . . .	24
5.3.4. Laura Energie . . . . .	25
5.4. Persona Stories . . . . .	27
5.4.1. Mara Hürlimann . . . . .	27
PS-001 Abholadresse eingeben . . . . .	27
PS-002 Lieferadresse eingeben . . . . .	27
PS-003 Pakete Dimensionen eingeben . . . . .	28

PS-004 Abholuhrzeit eingeben . . . . .	28
PS-005 Kontaktinformationen eingeben . . . . .	29
PS-006 Zollinformationen eingeben . . . . .	29
PS-007 Dienstleistung kaufen . . . . .	30
PS-008 Lieferung verfolgen . . . . .	30
PS-009 Unterstützung anfordern . . . . .	31
5.4.2. Peter Elsener . . . . .	32
PS-010 Versandangaben eingeben . . . . .	32
PS-011 Zollinformationen eingeben . . . . .	32
PS-012 Abholadressen speichern . . . . .	33
PS-013 Zieladressen speichern . . . . .	33
PS-014 Auftrag abbrechen . . . . .	34
PS-015 Aufträge auflisten . . . . .	34
5.4.3. Laura Energie . . . . .	35
PS-016 One-To-Many Auftrag erfassen . . . . .	35
PS-017 Auftragsstatus überprüfen . . . . .	35
PS-018 Routen auswählen . . . . .	35
5.5. Priorisierung . . . . .	36
5.6. Funktionale Anforderungen . . . . .	37
FREQ-001 Adresse automatisch vervollständigen . . . . .	37
FREQ-002 Adresse verifizieren . . . . .	37
FREQ-003 Auftrag im Backend-System erstellen . . . . .	37
FREQ-004 Auftrag im Backend-System aktualisieren . . . . .	37
FREQ-005 Auftrag im Backend-System starten . . . . .	37
FREQ-006 Auftragskosten in Rechnung stellen . . . . .	37
FREQ-007 Auftrag aus dem Backend-System laden . . . . .	38
FREQ-008 Auftragsstatus synchronisieren . . . . .	38
FREQ-009 Benutzer registrieren . . . . .	38
FREQ-010 Benutzer anmelden . . . . .	38
FREQ-011 Benutzer verwalten . . . . .	38
FREQ-012 Daten speichern . . . . .	38
FREQ-013 Daten laden . . . . .	38
5.7. Nicht-Funktionale Anforderungen . . . . .	39
NFREQ-001 Browser Kompatibel . . . . .	39
NFREQ-002 LoBo Kompatibel . . . . .	39
NFREQ-003 SSL Zertifikat . . . . .	39
<b>6. Konzept</b>	<b>40</b>
6.1. Web Modell . . . . .	40
6.1.1. Single-page application . . . . .	40
6.1.2. Mini-Backend . . . . .	41
6.1.3. Fazit . . . . .	42
6.2. JavaScript . . . . .	42
6.2.1. ES2015 . . . . .	43
6.2.2. Backend Framework . . . . .	43
6.2.3. Frontend Framework . . . . .	43
6.2.4. Fazit . . . . .	44
6.3. Prozess . . . . .	45
6.4. Architektur . . . . .	45
6.4.1. Mini-Backend . . . . .	45
6.4.2. SPA . . . . .	47
<b>7. Prototyp</b>	<b>50</b>
7.1. Tools . . . . .	50
7.1.1. Entwicklungsumgebung . . . . .	50
7.1.2. Browser . . . . .	50

7.1.3. Test Server . . . . .	52
7.2. Codebasis . . . . .	52
7.2.1. Webpack . . . . .	52
7.2.2. Babel . . . . .	53
7.2.3. React-Redux . . . . .	53
7.2.4. ESLint . . . . .	54
7.3. Entwicklungsprozess . . . . .	54
7.3.1. Mini-Backend . . . . .	54
7.3.2. SPA . . . . .	55
<b>8. Fazit</b>	<b>58</b>
<b>9. Verzeichnisse</b>	<b>59</b>
Literaturverzeichnis . . . . .	59
Abbildungsverzeichnis . . . . .	61
Tabellenverzeichnis . . . . .	62
Listingverzeichnis . . . . .	63
<b>A. Anhang</b>	<b>I</b>
A.1. LoBo API . . . . .	I
A.2. Kano Modell . . . . .	III
A.3. Interview Fragen . . . . .	IV
A.3.1. Stakeholder Interview Fragen . . . . .	IV
A.3.2. Stakeholder Interview Zusammenfassung . . . . .	V
A.4. Wireframes . . . . .	XI

# 1. Einleitung

## 1.1. Zusammenfassung

Webapplikationen werden heute von viele Menschen täglich genutzt. Emailverwaltung, Textbearbeitung, Routenplanung und Hotelreservierung sind einige Aufgaben, welche mit einer Applikation im Web erledigt werden. Die Benutzeroberflächen dieser Webapplikationen müssen einem grossen und diversen Publikum dienen. Dieses breite Anforderungsspektrum kann dazu führen, dass die Oberflächen zu kompliziert oder sogar unbrauchbar werden. Das Entwickeln einer Web Anwendung mit einer positiven Benutzererfahrung benötigt deshalb eine strukturierte und umfassende Analyse des zu bewältigenden Prozesses und der zukünftigen Benutzer.

## 1.2. Ausgangslage

Das Startup ImagineCargo revolutioniert die Art und Weise, wie Pakete versendet werden. Anstelle von Lastwagen-Flugzeug-Lastwagen setzt das Unternehmen auf Fahrrad-Zug-Fahrrad. Die Mission: Gleicher Preis, gleiche Geschwindigkeit, aber 99% Reduktion an CO2. Dazu ist es auf die neueste Technologie angewiesen. Für die Dispo und Backoffice Software LoBo sucht ImagineCargo nach einer Webapplikation, die Eingabe von Abholort, Abholzeit und Lieferort für den Kunden ermöglicht. Dazu sollen alle möglichen Zugverbindungen nach Abholzeit angegeben werden. Die Lieferzeit soll automatisch je nach gewählter Abholzeit bestimmt und die Preise für alle möglichen Zugverbindungen angezeigt werden. Zusätzlich soll der CO2-Ausstosses je nach gewählter Verbindung berechnet werden.

## 1.3. Ziel der Arbeit

Die bisherigen Prozesse und Strukturen des Unternehmens sollen kritisch betrachtet und analysiert werden. Es sollen konkrete Massnahmen zur Verbesserung konzipiert und umgesetzt werden. Mittels geeigneten Prozessen sollen die Anforderungen an eine Webapplikation von der Auftraggeberin und deren Endkunden erfasst werden. Es soll ein Konzept zur Optimierung der Benutzererfahrung im Web erstellt werden. Die vorgeschlagenen Massnahmen sollen mittels einem oder mehreren Prototypen strukturiert überprüft werden.

## 1.4. Aufgabenstellung

- Recherche: Nachforschung über das Velokurier- und das Expresskuriergeschäft im Allgemeinen anstellen. Herausfinden, welches die üblichen Prozesse und Praktiken in der Branche sind. Recherchieren, welche Software Produkte im Bereich Routenplanung bereits existieren.
- Ist-Analyse: Eine Analyse der vorhandenen Prozesse sowie der bereits eingesetzten Software durchführen und die gewonnenen Ergebnisse dokumentieren.
- Anforderungsanalyse: Herausfinden, welche Stakeholder existieren und mit persönlichen Interviews die Anforderungen aller Beteiligten aufnehmen. Alle Anforderungen an eine geeignete Webapplikation analysieren, priorisieren und strukturiert dokumentieren.

- Konzept: Anhand der Anforderungsanalyse ein Konzept für eine geeignete Webapplikation erstellen, welche das Ziel der Arbeit am besten unterstützt.
- Prototyp: Einen oder mehrere Prototypen zur Überprüfung des Konzepts implementieren.
- Testen: Umgesetztes Konzept mit den Stakeholdern auf ihre Richtigkeit und die tatsächliche Verbesserung des Prozesses testen und protokollieren. Prototypen werden mit Akzeptanztests auf ihre Fähigkeiten und Leistungen getestet.

## 1.5. Erwartete Resultate

- Recherche: Recherchebericht mit den gewonnenen Erkenntnissen. Übersicht über bereits existierende Softwarelösungen.
- Ist-Analyse: Dokumentation der vorhanden Prozesse, deren Zusammenhänge und Abhängigkeiten. Beschreibung der bereits genutzten Software.
- Anforderungsanalyse: Strukturierte Darstellung aller Anforderungen der Stakeholder.
- Konzept: Dokumentiertes Konzept mit Vorschlägen für Verbesserungen bzw. die Implementation einer Webapplikation.
- Prototyp: Einer oder mehrere präsentierbare Prototypen.
- Testen: Testprotokolle.

## 2. Recherche

### 2.1. Kurier-, Express- und Paketdienstbranche

#### 2.1.1. Schweiz

Die Kurier-, Express- und Paketdienstbranche (kurz KEP-Branche) ist ein altes und unscheinbares Gewerbe. In der Schweiz konnte sich der KEP-Markt erst ab 1997 entwickeln, weil zuvor die schweizerische Post (Post AG) ein Monopol auf den gesamten Inland-KEP-Markt hatte. Das Postgesetz vom 30. April 1997 [3] öffnete diesen Markt für private Anbieter. Ab dem ersten 1. Januar 1998 war es per Gesetz für private KEP-Anbieter erlaubt, Pakete mit einem Gewicht über 2 Kg und alle Expresssendungen zu transportieren. Am 1. Januar 2004 wurde das Gewichtslimit für Pakete auf 1 Kg gesenkt und per 1. April 2009 auf 50 Gramm womit die schweizerische Post nur noch ein Teilmopol auf den inländischen Briefmarkt hat. Seit der Öffnung des KEP-Marktes haben sich in der Schweiz einige private KEP-Dienstleistungsanbieter etabliert und unter dem Dachverband KEP&Mail<sup>1</sup> organisiert. Der Dach-

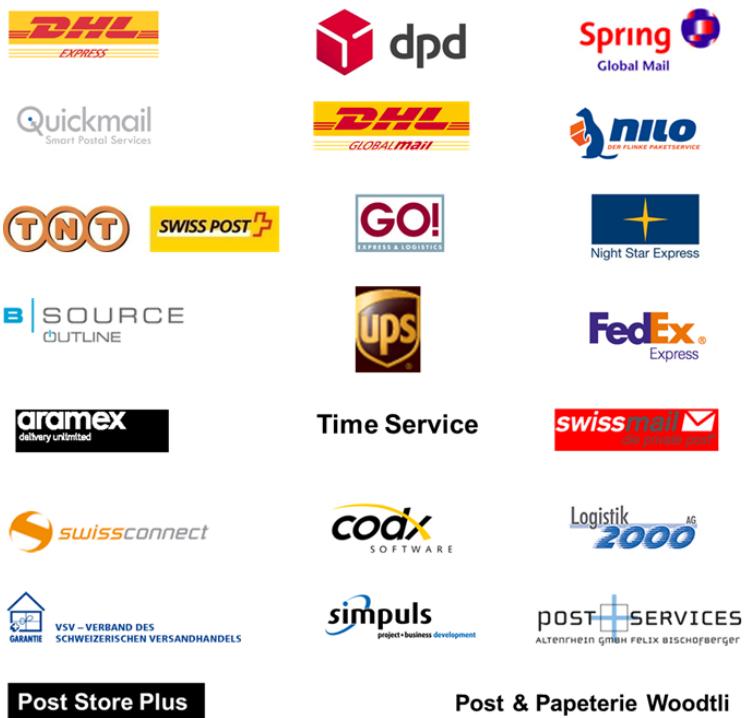


Abbildung 2.1.: Mitglieder von KEP&Mail

verband publiziert keine Zahlen bezüglich Marktanteilen der einzelnen Mitglieder. Im Jahresbericht 2015 [9, S.32] der PostCom, einer unabhängigen Regulierungsbehörde des schweizerischen Postmarktes, welche durch das neue Postgesetz im Oktober 2012 ins Leben gerufen wurde, werden aber Zahlen zum gesamten KEP-Markt publiziert. Der KEP-Markt (Pakete bis 30Kg, National, Import, Export) mit einem Volumen von 150 Millionen Paketen wird zu 25% von Kurier- und Expressdienstleistern bestritten.

<sup>1</sup>Siehe Abbildung 2.1

Beim Import und Export von Paketen wird ca. 80% des Volumens von privaten Anbietern abgearbeitet. Beim inländischen Versand von Paketen kommen die privaten Anbieter nur auf 20% Umsatzanteil<sup>2</sup>. Der Jahresbericht zählt die Post, DPD und DHL Express zu den grössten Anbieter von Paketdienstleis-

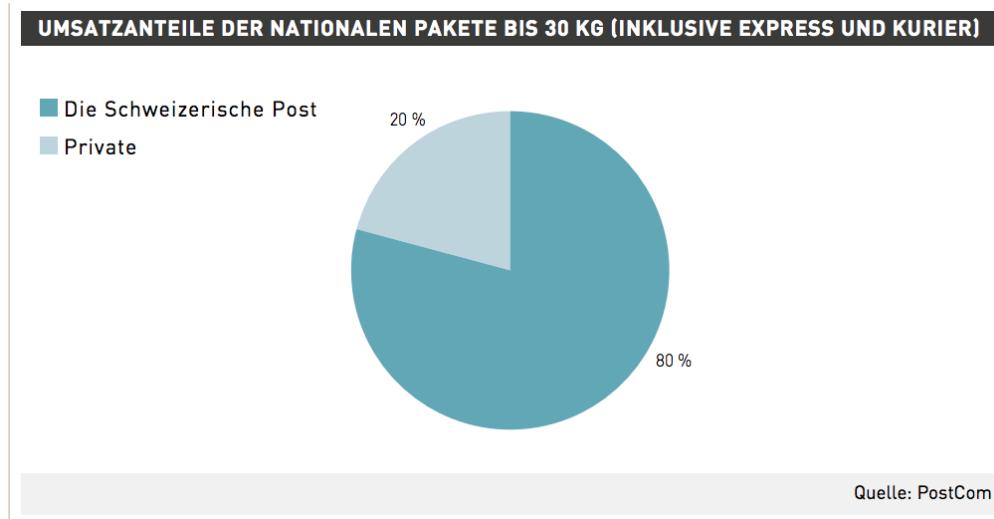


Abbildung 2.2.: Umsatzanteil der Nationalen Pakete

tungen in der Schweiz. FedEx, UPS, GO und TNT sind die grösseren internationalen Mitanbieter von Paketdienstleistungen in der Schweiz.

## 2.1.2. Europa

Informationen und Zahlen zum KEP-Markt in Europa sind rar und schwierig zu finden. Im Jahre 1999 haben sich sieben KEP-Dienstleistungsanbieter (DHL, DPD, General Parcel Austria, Österreichische Post, TNT, trans-o-flex und UPS) zu einem sogenannten KEP-Forum zusammen geschlossen, jedoch bereits 2005 wieder aufgelöst wurde, weil 2004 interne Preisabsprache vermutet wurde und die gemeinsame Plattform sich nicht in die Richtung entwickelt habe, wie ursprünglich angenommen [7, S. 14]. Allerdings ist anzunehmen, dass sich KEP-Märkte in Mitteleuropa gleichen. Der grösste KEP-Markt in Europa ist in Deutschland zu finden. 2011 machte dieser 36.9% aus<sup>3</sup>. Europäische KEP-Märkte weisen

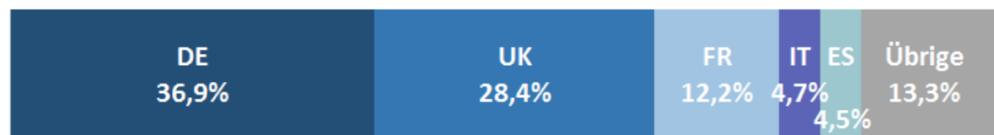


Abbildung 2.3.: Europäischer Kurier-, Express- und Paketmarkt im Jahre 2011

oligopolistische Tendenzen auf, weil ein Grossteil des Umsatzes von einigen wenigen Anbietern erzielt wird (DHL, DPD, FedEx, TNT, UPS)[2, S.14]. Jedoch bieten diese Marktstrukturen ein grosses Potenzial für innovative Unternehmen, welche nicht mit der Philosophie der zuvor genannten Anbietern arbeiten.

<sup>2</sup>Siehe Abbildung 2.2 Quelle: PostCom Jahresbericht 2015

<sup>3</sup>Siehe Abbildung 2.3 Quelle: WIK-Consult, Main Developments in the Postal Sector (2010-2013), Study for the European Commission, DG Internal Market and Services

## 2.2. Standards und Software

Obwohl es gewisse Standards für die Kurier-, Express- und Paketdienstbranche gibt, sind die meisten Systeme, welche bei den Anbietern im Einsatz sind eigene Entwicklungen. Diese Entwicklungen sind proprietär und bieten meistens ihre eigenen Benutzeroberflächen beziehungsweise Weboberflächen für Endkunden an. Größere Anbieter stellen Schnittstellen zu ihren Systemen zur Verfügung, folgen dabei jedoch keinem Standard. So brauchen Module wie z.B. XSI (eXpress Shipper Interface) als Teil von SAP<sup>4</sup>, welche auf einen elektronischen Datenaustausch vorbereitet sind, immer noch zusätzliche Komponenten, welche zwischen den Systemen übersetzen.

Standards, welche seit langer Zeit in der Branche genutzt werden, sind EDIFACT beziehungsweise EDITRANS, welche zu Zeiten ihrer Einführung über Telex System kommunizierten. Mit dem Aufkommen von Fax und Email verloren Telex Systeme und auch teilweise die Standards an Bedeutung. Telex und die Standards sind aber immer noch bei den ganz grossen KEP-Anbietern, welche eine eigene Flugzeugflotte betreiben, (z.B. Fedex) zu finden.

### 2.2.1. EDIFACT

UN/EDIFACT steht für *United Nations Electronic Data Interchange For Administration, Commerce and Transport* und ist Teil der internationalen EDI-Standards. EDI (Electronic Data Interchange)-Standards wurden für den elektronischen Datenaustausch zwischen Unternehmen oder durch Normierungsvorschläge von Branchenverbänden entwickelt. Unter den EDI-Standards findet sich auch SWIFT, welcher für den Datenaustausch zwischen Banken genutzt wird. Das UN zu Beginn des offiziellen Namens des Standards steht für *United Nations* (Vereinte Nationen), welche mit der Einrichtung CEFAC (Centre for Trade Facilitation and E-Business) für den Standard verantwortlich ist. Aufgrund der Grösse und Komplexität von EDIFACT wurden für die verschiedenen Branchen sogenannte *Subsets* erstellt. Im Folgenden eine nicht vollständige Liste mit Namen und Zugehörigkeit verschiedener Subsets [8, S.1].

**EANCOM** Konsumgüterindustrie

**EDIFOR** Speditionsbranche

**EDIFURN** Möbelbranche

**EDILIBE** Buchhandel

**EDITRANS** Transportwirtschaft

**EDIWHEEL** Reifen- und Räderhersteller

Im EDIFACT Standard sind ca. 200 verschiedene Nachrichtentypen für die unterschiedlichsten Anwendungszwecke definiert. Jede Nachricht wird durch einen sechsstelligen Namen eindeutig gekennzeichnet. Im Folgenden eine nicht vollständige List mit einigen Beispielen.

**IFTMBF** Buchungsanfrage (transport booking request)

**IFTMBC** Buchungsbestätigung (transport booking confirmation)

**IFTMIN** Transport-/Speditionsauftrag (instructions of transport)

**ORDERS** Bestellung (purchase order message)

**PRICAT** Preisliste/Katalog (price catalogue message)

Im Folgenden ist eine EDIFACT Nachricht, welche auf eine Verfügbarkeitsanfrage für einen Flug zwischen Frankfurt und Miami zurückkommt.

<sup>4</sup>SAP ist ein ERP-System

```

UNA:+.?
UNB+IATB:1+6XPPC+LHPPC+940101:0950+1'
UNH+1+PAORES:93:1:IA'
MSG+1:45'
IFT+3+XYZCOMPANY AVAILABILITY'
ERC+A7V:1:AMD'
IFT+3+NO MORE FLIGHTS'
ODI'
TVL+240493:1000::1220+FRA+JFK+DL+400+C'
PDI++C:3+Y::3+F::1'
APD+74C:0::6++++++6X'
TVL+240493:1740::2030+JFK+MIA+DL+081+C'
PDI++C:4'
APD+EM2:0:1630::6++++++DA'
UNT+13+1'
UNZ+1+1'

```

Die erste Zeile mit *UNA* wird in allen EDIFACT-Nachrichten benötigt. Sie definiert die Trennzeichen der Nachricht fest. Darauf folgt mit *UNB* ein *Interchange Header* und danach mit *UNH* die eigentliche Nachricht, welche bei *UNT* wieder endet. Mit *UNZ* endet die Nachricht.

Obwohl der EDIFACT Standard viele Geschäftsprozesse abdecken könnte, wird er nur von den grössten Anbietern der Branche eingesetzt. Ein finanzieller Gewinn durch den Standard entsteht erst ab einem gewissen Transportvolumen.

## 2.3. LoBo

Lobo ist eine SaaS Applikation, welche auf die Bedürfnisse von Kurier-, Express- und Paketdienstleistern zugeschnitten ist. Der Service wurde zu Beginn hauptsächlich von Fahrradkurieren eingesetzt, aber wird vermehrt auch von anderen Dienstleistern eingesetzt. Lobo wurde in PHP entwickelt, wird von einem Apache Server ausgeliefert und speichert die Daten in einer Datenbank. Zusätzlich bietet Lobo eine REST (*Representational State Transfer*) API (*Application Programming Interface*) an, welche mit einem Schlüssel und einer Client Identifikation genutzt werden kann. Die Abbildungen 2.4 und 2.5 geben einen visuellen Überblick über die Benutzeroberfläche von Lobo. In Lobo können Zeitmodelle erstellt werden, welche definieren, wie eine Zustellung von Start bis Ende zeitlich abläuft<sup>5</sup>. Diese Zeitmodelle sind die Produkte, welche dem Kunden zur Verfügung stehen. Zusätzlich wird eine Preisstaffel erstellt, welche den Preis für die Pakete und Route berechnet<sup>6</sup>.

### 2.3.1. API

Die API von Lovo kann mit jedem REST-kompatiblen System angesprochen werden. Jede Instanz von LoBo hat einen eigenen API-Endpunkt, welcher über eine URL (Uniform Resource Locator) erreichbar ist. Bei jeder Anfrage werden folgende Parameter zwingend benötigt.

**action** Die auszuführende Aktion

**clientIp** Die IP-Adresse des Anfragenden

Zusätzlich kann mit *responseFormat* die Enkodierung der Antwort angegeben werden. Standard ist JSON (JavaScript Object Notation). Bei gewissen Aktionen müssen oder können zusätzliche Parameter angegeben werden. Diese Parameter werden als URL-Enkodierte Forms versendet (*application/x-www-form-urlencoded*). Bevor die Anfrage gesendet werden kann, wird ein Hash über die Parameter berechnet. Die Berechnung findet nach *SHA-256* statt und der API Schlüssel wird dafür verwendet. Der

<sup>5</sup>Siehe Abbildung 2.6

<sup>6</sup>Siehe Abbildung 2.7

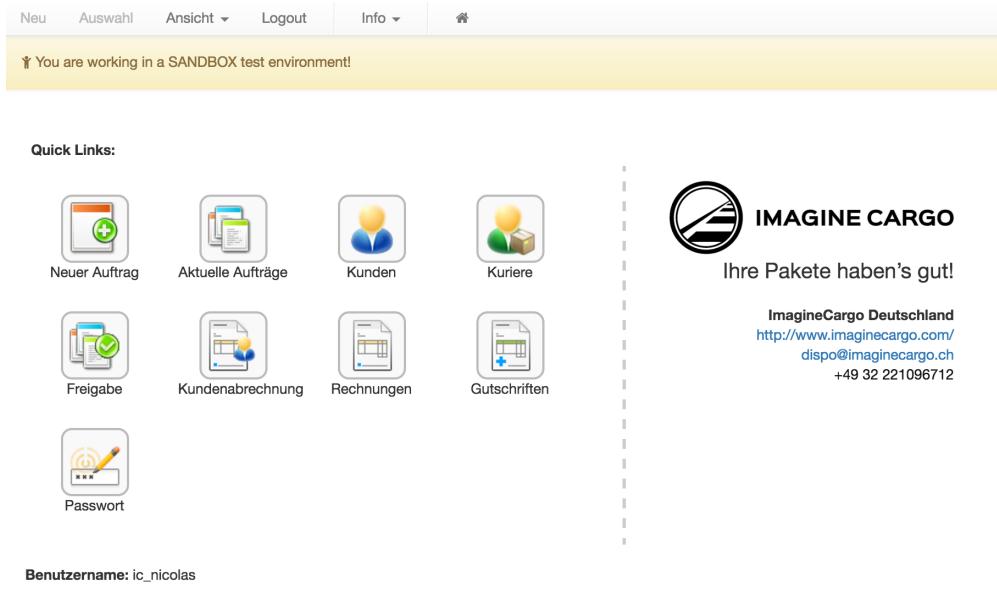


Abbildung 2.4.: Einstiegsmaske von Lobo

berechnete Hash wird als Hexadezimal im Header der Anfrage mitgeschickt. Zusätzlich wird die Client Identifikation im Header mitgeschickt. Dadurch lässt sich die Authentizität und Integrität der Daten auf dem Server verifizieren. Die API antwortet im gewünschten Format mit numerischen Stati, welche den Erfolg bzw. nicht Erfolg beschreiben und im Falle einer erfolgreichen Anfrage die gewünschten Daten zurückliefert. Im Folgenden eine nicht vollständige Liste mit einigen Beispiel-Anfragen. Die vollständige Liste ist im Anhang A.1 zu finden.

**getProductList** Benötigt keine Parameter und liefert die oben erwähnten Produkte zurück.

**createTask** Benötigt eine Produkt-Id, eine Zahlungs-Id und eine Kundennummer. Die Anfrage liefert einen Tasktoken zurück, welcher für die Weiterführung des Auftrags benötigt wird.

**addStop** Benötigt den oben erwähnten Tasktoken, einen dreistelligen Ländercode, eine Postleitzahl, eine Strasse und eine Hausnummer. Die Anfrage liefert die eingaben Daten zurück und eine eindeutige Id.

**calculateTask** Benötigt den oben erwähnten Tasktoken. Liefert bei erfolgreicher Verarbeitung alle Informationen, Zeiten und Kosten des Auftrages zurück.

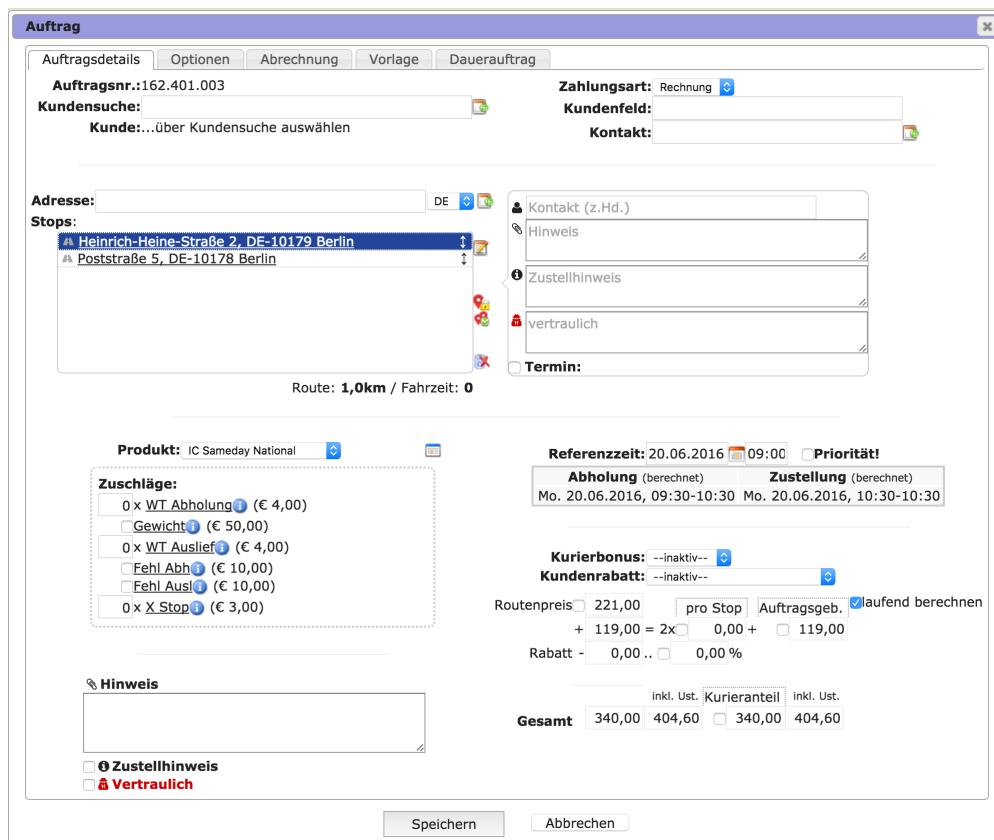


Abbildung 2.5.: Neuen Auftrag erstellen in Lobo

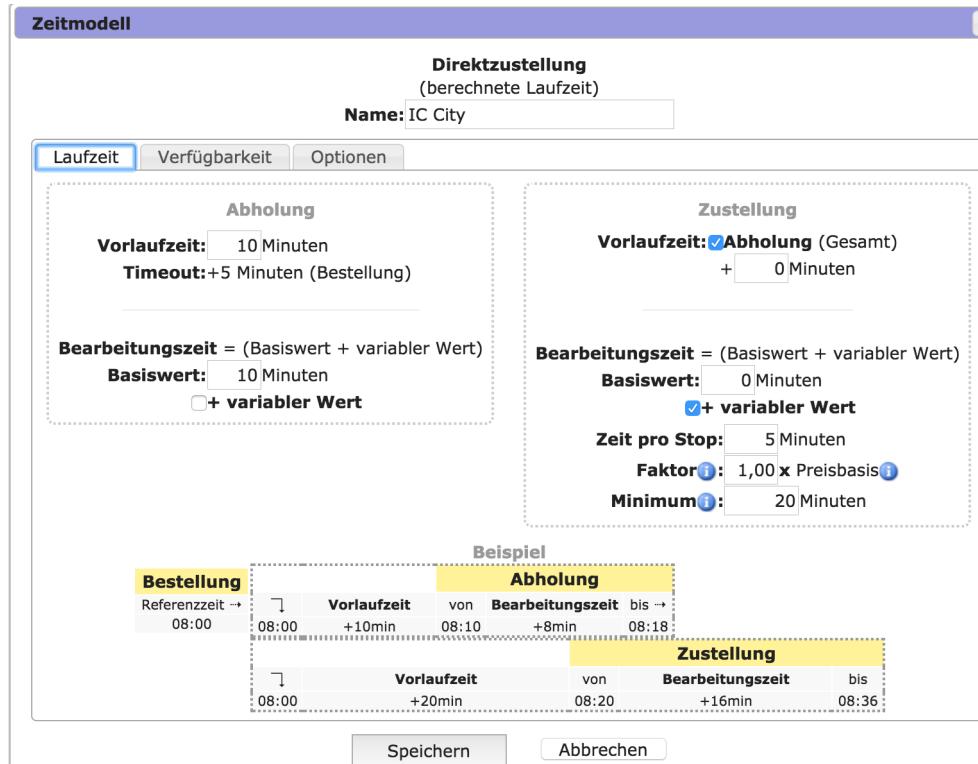


Abbildung 2.6.: Zeitmodell in Lobo

**Preisstaffel**

**Name:** Express

**Einheit:** kg (zum Beispiel: kg)

**Mengenangabe:**  freie Angabe  Auswahlliste

<b>Schwelle</b>		<b>Zuschlag</b>			<b>Firmenanteil</b>	
Name	Menge (q)	Pauschal [€]	Betrag x Menge [€]	% des Routenpreises <small>i</small>	%	
	≥		+	· q	+	

<b>Schwelle</b>		<b>Zuschlag</b>			<b>Firmenanteil</b>	
Name	Menge (q)	Pauschal	x Menge	% des Routenpreises	%	
package	≥0,000	55,000€	+0,000€ · q	+0,00%	100,00%	
package	≥5,000	60,000€	+0,000€ · q	+0,00%	100,00%	
package	≥10,000	60,000€	+0,000€ · q	+0,00%	100,00%	
> 20 pack	≥20,000	10,000€	+5,000€ · q	+0,00%	100,00%	
> 50 pack	≥50,000	10,000€	+4,500€ · q	+0,00%	100,00%	

**Preisstaffel**

**Schwelle**

**Zuschlag**

**Firmenanteil**

**Speichern** **Abbrechen**

Abbildung 2.7.: Zeitmodell in Lobo

# 3. Ist Analyse

## 3.1. ImagineCargo

ImagineCargo operiert im DACH-Raum (Deutschland, Österreich und Schweiz) und bietet die nachhaltige Expresskurier-Dienstleistung in 15 Städten an. Das Startup wurde im Oktober 2014 von Nick Blake gegründet und beschäftigt zur Zeit acht bis neun Mitarbeiter. Für das Unternehmen wurde ein Business Model Canvas erstellt, welches in der Abbildung 3.1 zu sehen ist. Business Model Canvas sind Vorlagen, um neue Prozesse zu entwickeln bzw. bestehende Prozesse zu dokumentieren. Die Mitarbeiter

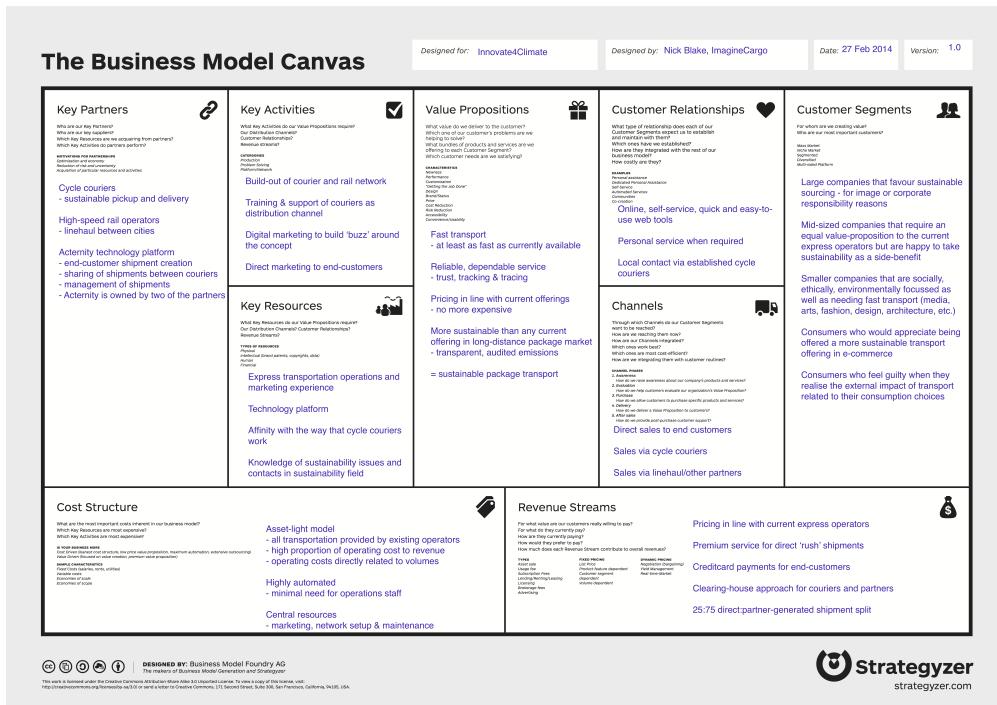


Abbildung 3.1.: Business Model Canvas von Imagine Cargo

von ImagineCargo haben rotierend die Rolle des Disponenten, welcher Aufträge per Mail oder Telefon annimmt und alle notwendigen Schritte für eine erfolgreiche Lieferung ausführt. Auf der Webseite von Imagine Cargo existiert eine *wufoo-Form* mit welchem über HTML 5 Form Elemente die benötigten Informationen eingegeben werden können und danach per Mail an den zuständigen Disponenten geschickt werden. Der Prozess wird im Kapitel 3.1.1 genauer beschrieben.

### 3.1.1. Prozesse

LoBo wurde ursprünglich für die Verwaltung und Koordinierung von Fahrradkurieren in **einer** Stadt entwickelt. In LoBo wird mit einem Polygon auf einer Karte das Versorgungsgebiet markiert. Damit wird überprüft ob eine Start- beziehungsweise Zieladresse zugelassen ist. Obwohl in einer LoBo-Instanz mehrere Polygone respektive Versorgungsgebiete konfiguriert werden können, bietet LoBo nicht sonderlich viele Funktionen an, um eine Dienstleistung zwischen diesen Gebieten zu ermöglichen. Das

grundständliche Problem besteht in der Berechnung der Dauer des Auftrages. Für das bessere Verständnis sei hier ein Beispiel beschrieben. Eine Zugfahrt von Zürich HB nach Berlin Hbf dauert ca. acht Stunden und elf Minuten. Ein Auftrag für ein Paket aus dem Zürcher Kreis 4 nach Wedding in Berlin dauert inkl. Bahntransport ca. zehn Stunden. Wenn besagter Auftrag um acht Uhr morgens gestartet wird, trifft das Paket um ca. 18 Uhr an seinem Bestimmungsort ein. Die Öffnungszeiten des Berliner Kuriers sind von montags bis freitags von 07:30 Uhr bis 20:00 Uhr und dementsprechend kein Hindernis für das Paket nach Berlin Wedding. Wenn der gleiche Auftrag um 13:00 Uhr gestartet werden soll, wird das Paket in Berlin aber nicht mehr vom Hauptbahnhof abgeholt. Um diese Berechnungen möglich zu machen, wurden in LoBo sogenannte Kreuztabellen implementiert. In diesen Kreuztabellen werden die Reisezeiten zwischen den verschiedenen Versorgungsgebieten eingetragen, wodurch die Berechnung der Gesamtdauer eines Auftrages möglich wird. Zusätzlich zu den Kreuztabellen sind einige weitere Schritte notwendig, um einen Auftrag korrekt im System einzutragen. In der Abbildung 3.2 ist der Prozess schematisch dargestellt.

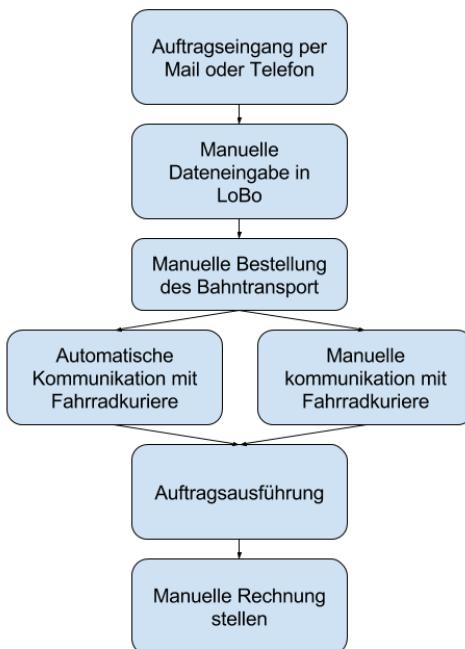


Abbildung 3.2.: Aktueller Prozess für die Erstellung eines Auftrages

Bei der manuellen Dateneingabe in LoBo muss der korrekte Bahnhof der Stadt, in der eine Lieferung startet und in der sie ankommen soll, als zusätzliche Stopps eingetragen werden. Dadurch weiss der zuständige Fahrradkurier, wohin das Paket soll. Für die Start- und Zielstadt sind in den meisten Fällen unterschiedliche Fahrradkurierunternehmen zuständig. Deshalb muss dem Auftrag, sobald der Fahrradkurier der Startstadt das Paket an den Bahnhof gebracht hat, der zuständige Fahrradkurier der Zielstadt zugewiesen werden. Dieser Umstand wird in Kauf genommen, damit am Ende des Auftrages die Abrechnung immer noch korrekt funktioniert. Bevor der Auftrag überhaupt startet, muss für das Paket ein Ticket bei der zuständigen Bahngesellschaft bestellt werden. In Deutschland wird der Verkauf von diesen Tickets von time:matters<sup>1</sup> angeboten, aber nicht über eine API. Dies wird vom zuständigen Disponenten manuell ausgeführt und dann den beauftragten Fahrradkurieren übermittelt.

<sup>1</sup>time:matters ist eine Tochtergesellschaft von Lufthansa Cargo, welche das Monopol auf das Transportieren von Paketen in Deutschland besitzt <http://www.time-matters.com/>

## 3.2. Marktanalyse

Beim Versuch einen Überblick über die vorhandenen Softwarelösungen auf dem Markt zu bekommen, sind viele Softwarelösungen für Speditionen in den Suchresultaten aufgetaucht. Viele dieser Softwarelösungen sind nur schon unbrauchbar, weil sie auf einer klassischen Client-Server-Architektur mit ausführbaren Programmen aufbauen. Die wenigen Angebote, welche eine webbasierte Benutzeroberfläche anbieten, sind aber durch den starken Fokus auf Spedition unbrauchbar. Diese Tools sind bessem geeignet, eine möglichst effiziente Traveling-Salesman-Route zu berechnen, als für Kunden, welche zum ersten mal versuche, ein Paket mit einem Expresskurier zu verschicken. Die zwei vielversprechendsten Lösungen werden im Folgenden kurz vorgestellt.

### 3.2.1. Deliveo

Deliveo ist eine Software für Kurierdienste und KEP-Dienstleister mit dem Fokus auf motorisierte Fahrzeuge und Fahrräder. Die Benutzeroberfläche von Deliveo macht einen vielversprechenden Eindruck<sup>2</sup> und bietet eine grosse Anzahl an Funktionen. Im Folgende eine nicht abgeschlossene Liste mit Funktionen von Deliveo<sup>3</sup>.

- Das ganze System ist mehrsprachig.
- Abrechnung mit dem Kurier (Nachnahme-Preise, Servicepreise und zurückgeholte Dokumente).
- Positionsanzeige des Kuriers
- Gut entwickeltes API System für die Kommunikation mit anderen Systemen, oder zur Unterstützung der unabhängigen Entwicklungen des Kurierdienstes.
- Paket-Betrieb (Aufnahme und Auslieferung) mittels Barcode-Leser mit Smartphone-Kamera

Die Benutzeroberfläche welche die Erstellung eines neuen Auftrags ermöglicht, scheint den Bedürfnissen welche in Kapitel 5.4 beschrieben sind, zu einem grossen Teil gerecht zu werden<sup>4</sup>. Die Handhabung macht einen einfachen Eindruck und aus Sicht des Benutzer wirkt es angenehmer, als die Eingabemaske von LoBo.

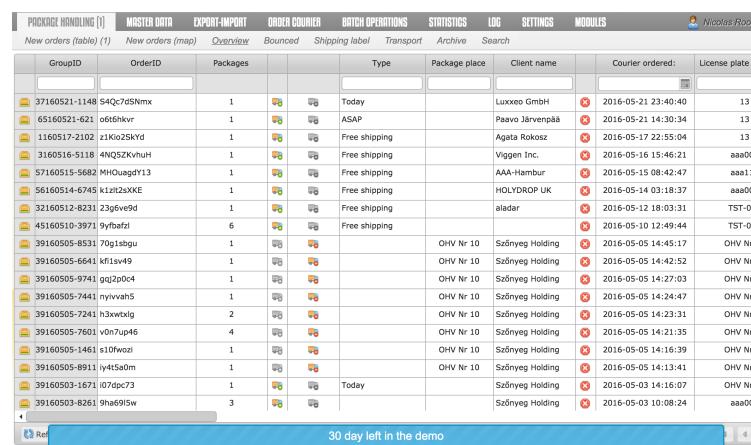


Abbildung 3.3.: Auftragsmaske von Deliveo

Deliveo offeriert eine eigene API. Obwohl die Dokumentation dieser Schnittstelle in Ungarisch geschrieben ist, ist es möglich, damit einen Auftrag zu erstellen. Das System bietet eine Benutzerverwaltung an, welche aber nur eine Authentifizierung anbietet und keine Autorisierung betreibt.

<sup>2</sup> Siehe Abbildung 3.3

<sup>3</sup> Informationen direkt von der Webseite des Herstellers. <http://deliveo.eu/>

<sup>4</sup> Siehe Abbildung 3.4

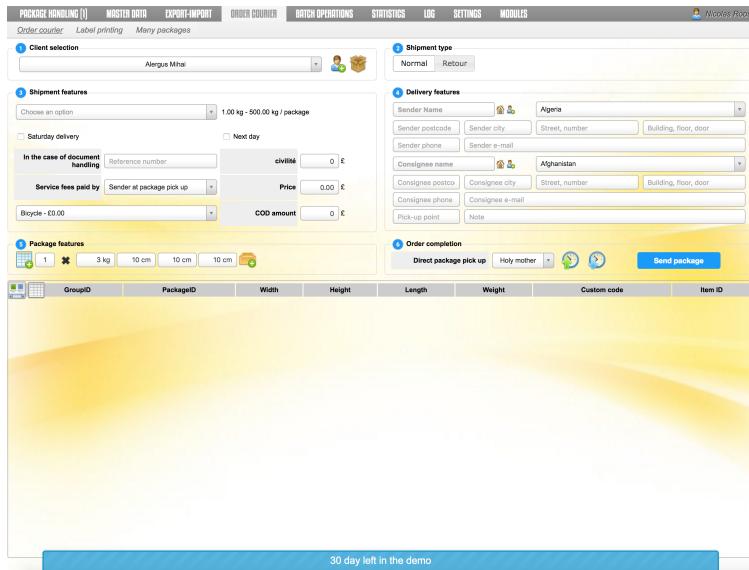


Abbildung 3.4.: Auftragsmaske für das Erstellen eines neuen Auftrages

### 3.2.2. eCourier

$$m^2 = f_t * r$$

Die Webapplikation eCourier bietet eine Softwarelösung für KEP-Dienstleister an. Sie wurde von Bamboo-Software, einem Entwicklungsunternehmen in Berlin, hergestellt. Die Software bietet ähnliche Funktionen wie LoBo und Deliveo an. Im Folgenden eine nicht abgeschlossene Liste mit Funktionen von eCourier<sup>5</sup>

- Geeignet für Stadtcurier / Direktcurier / nationaler und internationaler Expressversand
- Anbindungen an Tom Tom und AIS vorhanden
- Schnittstellen unter anderem zu UPS, Der Kurier, KEP AG, FedEx, Ilonexs, GEL, GLS, time:matters

Aufgrund mangelnder Testmöglichkeiten - Bamboo-Software bietet für Evaluationszwecken keine Testinstallation an - konnte eCourier im Rahmen dieser Bachelorarbeit nicht getestet werden. Die Benutzeroberfläche<sup>6</sup> jedoch sieht vielversprechend aus und die direkte Anbindung an time:matters ist äusserst interessant.

### 3.2.3. Fazit

Obschon die beiden analysierten Softwarelösungen, die aktuelle auf dem Markt zu finden sind, einen positiven Eindruck hinterlassen und die gleiche Funktionalität, welche die im Rahmen dieser Bachelorarbeit zu entwickelnde, Benutzeroberfläche benötigt, wie LoBo anbieten, sind beide Lösungen ungeeignet. Die Lösung von Bamboo-Software wurde bei der Gründung von ImagineCago zusammen mit LoBo evaluiert und hat den Anforderungen des Start-Ups nicht genügt. Die ganze Backoffice-Funktionalität wird zwar nicht für die zu entwickelnde Benutzeroberfläche benötigt, ist aber ein grosser Bestandteil für die Wahl eines Backend. Die Benutzeroberfläche von Deliveo ist Menschen mit weniger Erfahrung zuzumuten. Aber aufgrund der fehlenden Möglichkeit Benutzer zu autorisieren, benötigt auch diese Lösung eine eigene Webapplikation, bei welcher der Benutzer nur Aufträge aufgeben kann und keine anderen Berechtigungen hat. Zusätzlich unterhält ImagineCargo enge Beziehungen zum Entwickler von LoBo und hat dadurch die Möglichkeit, zukünftige Entwicklungen zu beeinflussen. Der Entwickler von LoBo hat

<sup>5</sup> Informationen direkt von der Webseite des Herstellers. <https://bamboo-software.de/ecourier>

<sup>6</sup> Siehe Abbildung 3.5

The screenshot shows a software interface for creating a new order. The main window is titled 'Auftrag bearbeiten' and displays a grid for entering shipping details. The grid includes columns for 'Auftrag d.', 'Rechnung an', 'Rechnung an', 'Nachnahme', 'Auslage', 'Zustellung', and 'Artikel'. The sidebar on the right contains transport and payment settings, including 'STADTKURIER DEKU', 'ROUTE', 'Interne Belegnr.', 'Sendungsnummer', 'KURIER', 'RekFak Kunde', 'Rückkommentar', and various buttons for 'PREISE', 'DOCS', 'NOTIZ', 'HISTORIE', 'SPEICHERN', and 'TRANSPORT'.

Abbildung 3.5.: Auftragsmaske für das Erstellen eines neuen Auftrages

auch Interesse an einer *White Label*<sup>7</sup> Version der im Rahmen dieser Bachelorarbeit zu entwickelnden Webapplikation bekundet. Diese Situation bietet viele Möglichkeiten für Anpassungen in LoBo, was ein bedeutend grösserer Vorteil ist als die zuvor erwähnten von Deliveo und eCourier. Dieses Fazit ist in der nicht funktionalen Anforderung NFREQ-002 festgehalten.

<sup>7</sup> Eine *White Label* Lösung ist ohne Marke und kann vom Kunden selbst angepasst werden.

# 4. User Centered Design

Als *User Centered Design* werden Prinzipien und Werkzeuge verstanden, welche den Designern, Entwicklern und allen anderen, die an der Entstehung eines Produktes, Services oder Prozesses beteiligt sind, helfen, die Bedürfnisse und Wünsche des Endbenutzer während der gesamten Entwicklungsphase mit hoher Priorität in den Vordergrund zu stellen. Dieses Framework steht in starkem Kontrast zu herkömmlichen Entwicklungsmethoden, bei welchen die Stakeholders, zu einem grossen Teil die Ausrichtung, das Aussehen und Verhalten einer neuen Erschaffung bestimmen. Bei diesen älteren Entwicklungsmethoden kam es vermehrt dazu, dass ein Produkt erst bei der Veröffentlichung zum ersten Mal die Hände eines Endbenutzer berührte und sich dann als komplett unnütz herausstellte. Der *User Centered Design*-Ansatz involviert mögliche Endbenutzer bereits ab den ersten Entwicklungsphasen und fordert das Zielpublikum zur Mitgestaltung auf. Dadurch werden viele Probleme, die bei einer Markteinführung auftreten können, bereits früh entdeckt und behoben. Die Anwendung dieses Frameworks kostet im Gegensatz zu anderen Methoden mehr Zeit und dadurch auch mehr Geld. *User Centered Design* ist eine iterative Methode, welche nach der Durchführung ihrer Teilschritte wieder von vorne beginnt. Abbildung 4.1 zeigt eine mögliche schematische Darstellung des Ablaufs von *User Centered Design*.

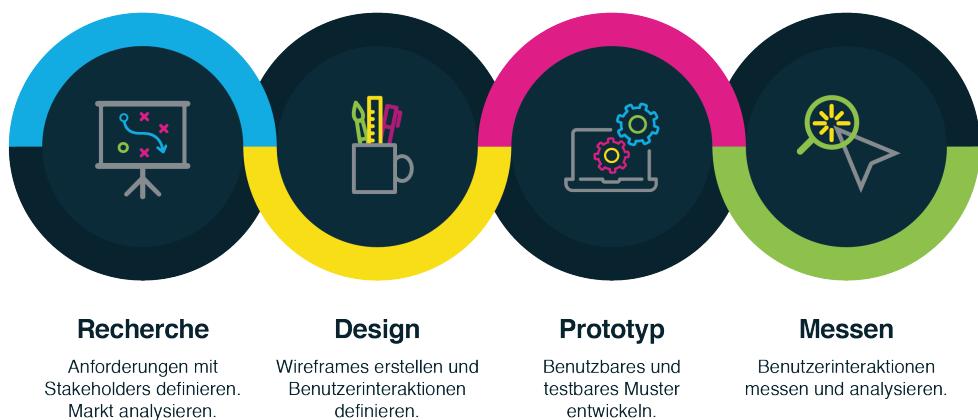


Abbildung 4.1.: Prozessablauf von User Centered Design

Während all diesen Teilschritten werden die Ergebnisse und Erkenntnisse den möglichen Endbenutzern vorgestellt. Dabei wird überprüft, ob z.B. der Interaktionsfluss oder die Benutzeroberflächen einer Applikation verständlich sind oder alle relevanten Spezialfälle abdecken. *User Centered Design* kann auf viele verschiedene Arten angewendet werden und zu den gewünschten Resultaten führen. Diese Bachelorarbeit orientiert sich am User Centered Design-Prozess, welcher im Buch von Kim Goodwin mit dem Namen *Designing for the Digital Age* [5] beschrieben wird.

## 4.1. Anwendung

In den ersten Gesprächen mit den Stakeholdern wurde schnell klar, dass das Ziel des Produktes eine Webapplikation sein soll, die von Benutzern jeglicher Generation und Demographie benutzt werden kann. Die Entscheidung für eine Entwicklung unter *User Centered Design* war naheliegend. Im Folgenden ist der geplante Ablauf mit dem Fokus auf den Endbenutzer beschrieben.

### 4.1.1. Team

Bei der Zusammenstellung des Design Teams ist es wichtig die verschiedenen benötigten Fähigkeiten in einer kleinen Gruppe zusammenzufassen. Ein Interaktionsdesigner hat einen ganz anderen Fokus als der grafische Designer. Im Rahmen dieser Bachelorarbeit fielen alle Rollen des Designs auf eine Person. Diese Konstellation ist nicht per se nachteilig. Die einzelne Person muss aber in der Lage sein, während den verschiedenen Phasen die Probleme, Ideen, Vorschläge, Lösungen und vieles andere aus den jeweiligen Perspektiven zu betrachten und beurteilen. In der idealen Zusammensetzung nach Goodwin sind folgende Rollen zu besetzen [5, Kapitel 2].

**Teamleiter\_in** Verantwortlich für das Team und die Koordination mit allen anderen Involvierten.

**Interaction Designer\_in (Generator)** Verantwortlich für die Visualisierung des Systemverhaltens.

**Interaction Designer\_in (Synthesizer)** Verantwortlich für die Analyse und Kommunikation des Designs.

**Grafischer Designer\_in** Verantwortlich für die visuelle Umsetzung des Designs.

**Industrie Designer\_in** Verantwortlich für das Design der Hardware.

### 4.1.2. Grundlegende Recherche der Domäne

Während der Recherche wird die Domäne, in der sich das zu erschaffende Produkt befindet, analysiert. Dabei werden Marktanalysen getätigt sowie falls vorhanden die Produkte der Konkurrenz untersucht. Zusätzlich werden Experten und Menschen, welche bereits lange in dem Bereich tätig sind, befragt. Diese Arbeit wird vom gesamten Design-Team bewältigt, da bereits in dieser Phase die verschiedenen Perspektiven wichtig sind. Mit Nick Blake als Gründer von Imagine Cargo und Experte der Kurier-, Express- und Paketdienstindustrie hat diese Bachelorarbeit auf sehr viel Erfahrung und Insiderwissen zurückgreifen können.

### 4.1.3. Interviews mit Stakeholders

Bevor die Interviews beginnen können, müssen die Stakeholder definiert werden. Zu den Stakeholdern gehören grundsätzlich alle Menschen, die ein Produkt mitbestimmen können. Jedoch zählt nicht jede Meinung gleich viel. Bei der Analyse ist es wichtig, mit allen involvierten Menschen zu reden und auch jene Personen aufzufinden, welche sich aus dem ganzen Design-Prozess raushalten, aber die Macht haben das ganze Projekt zu beerdigen. Üblicherweise finden sich solche Stakeholder im Management. Es sind Personen, die das Gefühl haben ihre Meinung sei noch nicht notwendig beziehungsweise er/sie werde erst aktiv wenn das Projekt ein Fundament hat. Meist ist es dann aber zu spät für grundlegende Designänderungen. Für die Bachelorthesis wurden David Emmerth und Nick Blake von Imagine Cargo interviewt. Die Fragen für die Interviews sind vom Buch von Kim Goodwin inspiriert worden [5, Kapitel 5] und sind im Anhang A.3.1 zu finden.

### 4.1.4. Interviews mit Endbenutzer

In den Interviews mit den Stakeholdern sollte sich eine grobe Definition des zu erstellenden Produktes heraus kristallisieren. Mit dieser gewonnenen Definition werden mit möglichen Endbenutzern Interviews geführt. Das soll weitere Einblicke in die Anforderungen und Unzufriedenheiten mit bestehenden Produkten zu Tage bringen. Aufgrund der beschränkten Ressourcen im Rahmen dieser Bachelorarbeit, wurde auf diesen Schritt verzichtet. Dafür sollen in der Visualisierungsphase die erstellten Wireframes und die Interaktionsabläufe mit möglichen Endbenutzern überprüft werden.

### 4.1.5. Personas

Aus den Interviews mit Stakeholdern und Endbenutzern, der Recherche und Gesprächen mit Experten werden sogenannte Personas erstellt. Eine Persona ist eine mit Fliesstext und Legende beschriebene Persönlichkeit. Im Text wird beschrieben wie sich die Person in Bezug auf das zu entwickelnde Produkt verhält und was ihre Voraussetzungen sind. Auch ihr Grundwissen wird definiert. Die Persona soll dem Design-Team ermöglichen, die Perspektive der beschriebenen Person einzunehmen und wie sie zu denken. Aufgrund der fehlenden Interviews mit Endbenutzer wurden für die Bachelorarbeit die Personas aufgrund der Interviews mit den Stakeholdern sowie der Recherche erstellt. Persönliche Erfahrungen mit Geschäftsbranchen, welche Kuriere nutzten, sowie Kontakte zu Fahrradkurieren haben zur Erstellung der Personas beigetragen. Die Personas sind im Kapitel 5.3 definiert.

### 4.1.6. Stories erstellen und Anforderungen definieren

Aus den definierten Personas und den Interviews mit den Stakeholdern werden Persona Stories erstellt, welche die Anforderungen an das zu entwickelnde Produkt beschreiben. Die Persona Stories werden aus Sicht der Personas geschrieben. Diese grammatischen Bedingungen hilft den Autoren sich in die Gedanken, Gefühle und Denkweisen der Persona zu versetzen. Die Persona Stories für die Bachelorthesis sind in Kapitel 5.4 nach zu lesen.

### 4.1.7. Wireframes und Interaktionsabläufe

Die Personas, ihre Stories und die Anforderungen dienen als Grundlage für die Erstellung erster Wireframes<sup>1</sup> und Definitionen der Interaktionsabläufe. Wireframes verbunden mit Interaktionsabläufen sind schneller zu erzeugen als ein Prototyp zu entwickeln. Dadurch sind sie geeignet für die *User Centered Design* Methode, weil der Benutzer dadurch in der Lage ist, das bisher theoretisch Besprochene praktisch zu sehen und es sich besser vorzustellen. Dadurch können erneut viele Missverständnisse zwischen dem Team und dem Endbenutzer beseitigt werden, dies alles vor der Entwicklung eines ersten Prototypen. Im Rahmen dieser Bachelorthesis wurden auf der Suche nach möglichen Endbenutzern 40 Zürcher Unternehmen angeschrieben. Weil die Kundschaft von ImagineCargo zum Zeitpunkt dieses Schrittes noch nicht existierte, wurden Architekturbüros, Werbeagenturen (mit Fokus auf Print), Anwaltskanzleien und Medizinlabors angeschrieben. Die Annahme dass diese Firmen regelmässig Kurier-, Express- und Paketdienstleistungen in Anspruch nehmen würden, lag nahe. Von den knapp 40 angeschriebenen Unternehmen antwortete jedoch nur acht mit der jeweils gleichen Antwort, sie sagten dass Sie zu unregelmässig KEP-Dienstleistungen benutzt würden und daher keine relevante und signifikante Hilfe sein können. Aufgrund der beschränkten Zeitdauer dieser Bachelorthesis wurden für die Überprüfung der Wireframes erneut die Stakeholder kontaktiert. Zusätzlich wurde vereinbart, dass nach der Veröffentlichung der ersten Version die effektiven Benutzer der Webapplikation für eine Überprüfung des Designs und der Interaktionsabläufe kontaktiert werden. Die Wireframes sind im Anhang A.4 zu finden.

### 4.1.8. Prototyp

Die Entwicklung des Prototyps ermöglicht es, die Machbarkeit zu überprüfen. Wie die restlichen Schritte findet auch die Entwicklung des Prototypen iterativ statt. Dadurch kann der Fortschritt regelmässig überprüft und den möglichen Endbenutzern zum ausprobieren vorgelegt werden. Dabei können schlecht oder nicht funktionierende Aktionen oder Interaktionsabläufe angepasst werden. Die Entwicklung des Prototypen im Rahmen dieser Bachelorthesis wurde in regelmässigen Abständen mit den Stakeholdern überprüft.

<sup>1</sup> Wireframe (Drahtmodell) bezeichnet eine visuelle Darstellung einer Benutzeroberfläche wobei der Fokus auf der Anordnung der Elemente und nicht auf der Gestaltung liegt

## 4.2. Interviews

Die Interviews mit den Stakeholdern wurden in entspannter und informeller Atmosphäre einzeln geführt. Das Interview wurde mit einem Mobiltelefon aufgenommen, damit der Interviewer während des Interviews zu 100% auf die Gesprächspartner konzentriert war. Diese Art von Interview lässt auch Abschweifungen zu, was nützliche Informationen zu Tage brachte, welche sonst nicht zur Sprache gekommen wären. Die Interviews sind im Anhang A.3.2 zusammengefasst.

# 5. Anforderungsanalyse

## 5.1. Einleitung

Mit dem User Centered Design Prozess konzentriert sich die Anforderungsanalyse sehr stark auf den Benutzer. Die Interviews zeigten was für Aufgaben die Webapplikation für die Firma lösen soll.

## 5.2. Stakeholders

**David Emmerth** Radkuriert und Social Entrepreneurship-Experte, Community Organiser für die Schweiz und Deutschland.

**Nick Blake** Gründer von Imagine Cargo.

## 5.3. Personas

### 5.3.1. Einleitung

Personas sind Urbilder, welche die verschiedenen Ziele und Verhaltensmuster der typischen Benutzer beschreiben sollen. Sie unterscheiden sich von anderen Methoden hauptsächlich durch ihre geschichterzählende Art, welche bei Entwicklern und Designern besonders die soziale und emotionale Intelligenz fördert. Die Wichtigkeit von Personas wird im Buch *Designing for the Digital Age* [5, chapter 11] wie folgt beschrieben: „A persona encapsulates and explains the most critical behavioral data in a way that designers and stakeholders can understand, remember, and relate to“.

Es gibt verschiedene Möglichkeiten, die Beschreibung und Charakterisierung einer Persona festzuhalten. Im Rahmen dieser Arbeit wird die Methode angewendet, welche im Artikel [6] „User stories don't help users: introducing persona stories“ beschrieben ist. Die Beschreibung ist in eine Vorder- und Rückseite aufgeteilt. Die Vorderseite soll den Hintergrund und die Motivation einer Persona beschreiben und damit erklären, warum ein gewisses Merkmal auf diese Art und Weise implementiert werden soll. Die Rückseite beschreibt praktische Details, welche helfen, Personas von diesem Typ zu erkennen. Diese Informationen werden auch beim rekrutieren von Testpersonen benötigt.

### 5.3.2. Mara Hürlimann

#### Hintergrund & Motivation

Mara nutzt die Dienstleistungen der KEP-Branche nicht regelmässig, was sie aber nicht zu einer atypischen Nutzerin macht. Mara ist alleinerziehende Mutter und arbeitet als Lektorin von zuhause aus. In ihrer Freizeit engagiert sie sich im Quartierverein, organisiert kleinere Veranstaltungen für Flüchtlinge oder liest in der Sonne auf dem Balkon. Mara hat ein starkes Umweltbewusstsein und setzt für die Fortbewegung in der Stadt voll und ganz auf das Fahrrad. Für grössere Be- bzw. Entsorgungen mietet Mara ein Fahrzeug bei Mobility. Maras Umweltbewusstsein zeigt sich auch bei den wöchentlichen Einkäufen. Sie kauft bewusst saisonal und regional ein und versucht dadurch ihren, CO2 Abdruck möglichst klein zu halten.

Photo	Steckbrief
	<ul style="list-style-type: none"> <li>▪ Mara Hürlimann</li> <li>▪ 49-Jährig</li> <li>▪ aus Zürich (Stadt)</li> <li>▪ arbeitet als Lektorin</li> </ul>

Tabelle 5.1.: Persona Steckbrief für Mara Hürlimann

Alter	20 - 60 Jahre
Wohnort	Stadtzentrum
Ausbildung	Matura oder besser
Nutzung einer KEP-Dienstleistung / Jahr	1 - 2 mal

Tabelle 5.2.: Charaktermerkmal für Persona Mara Hürlimann

Maras älteste Tochter studiert Mediale Künste in Berlin und hat Bei ihrem letzten Besuch in Zürich ihre Unterlagen und ein Raspberry Pi, welches sie für die Abschlusspräsentation benötigt, vergessen. Die Präsentation findet am Folgetag um 16 Uhr statt und die junge Frau hat keine Möglichkeit, ein neues Gerät aufzutreiben. Mara will die Unterlagen inklusive dem Raspberry Pi mit einer Same-Day-Lieferung an ihre Tochter schicken und dabei ihre Umweltsphilosophie nicht verraten.

### Charaktermerkmal

Die Personen brauchen eine Expresskurier-Dienstleistung nur für Notfälle oder spezielle Ausnahmen. Sie haben keine Erfahrungen mit anderen KEP-Dienstleistern und verlassen sich auf das im Markt etablierte Image eines Anbieters. Ihnen ist Transparenz und Zuverlässigkeit wichtig. Sie wissen nicht, wie der Prozess abläuft oder welche Informationen benötigt werden und sind beim Einkauf einer Dienstleistung auf Hilfe angewiesen.

### 5.3.3. Peter Elsener

#### Hintergrund & Motivation

Peter arbeitet in einer kreativen Agentur als Projektleiter. Peter treibt in seiner Freizeit viel Sport und spielt einmal in der Woche mit seinen Arbeitskollegen über Mittag Fussball. Peter hat seine Prinzipien, welche auch der Umwelt zugute kommen, kann aber auch sehr opportun sein. Effizientes Arbeiten ist ihm sehr wichtig. Das Gegenteil akzeptiert er weder bei Mitarbeitern noch bei Dienstleistern. Peter verwaltet viele Kunden und ist sehr darum Bemüht all deren Bedürfnisse zur vollsten Zufriedenheit zu erfüllen.

Trotz des digitalen Zeitalters involvieren viele Projekte von Peter immer noch Drucksachen. Diese werden vor der Massenproduktion dem Kunden ausgedruckt vorgelegt, damit dieser ein „Gut zum Druck“geben kann. Üblicherweise werden diese Drucksachen erst kurz vor der Deadline fertig. Damit der Kunde schnellstmöglich das Okay geben kann, werden diese Ausdrucke per Expresskurier verschickt.

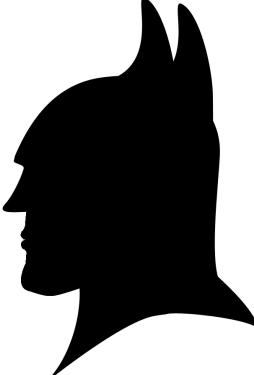
Photo	Steckbrief
	<ul style="list-style-type: none"> <li>▪ Peter Elsener</li> <li>▪ 28-Jährig</li> <li>▪ aus Zürich (Stadt)</li> <li>▪ arbeitet als Projektleiter</li> </ul>

Tabelle 5.3.: Persona Steckbrief für Peter Elsener

Alter	20 - 40 Jahre
Wohnort	Stadt oder Land
Ausbildung	Bachelor oder besser
Nutzung einer KEP-Dienstleistung / Jahr	50 - 100 mal

Tabelle 5.4.: Charaktermerkmal für Persona Peter Elsener

Viele Kunden von Peter haben ihre Räumlichkeiten ausserkantonal und sind deshalb nicht über einen lokalen Fahrradkurierdienst erreichbar.

Peter benötigt dafür eine Same-Day-Lieferung und will einen verlässlichen und transparenten Expresskurier. Pflichtbewusst wie Peter ist, hat er den Expresskurier bei seinen Kunden schon budgetiert und will nicht mit unvorhergesehenen Kosten überrascht werden.

### Charaktermerkmal

Die Personen benötigen regelmässig Expresskurierdienstleistungen und können gut mit webbasierten Plattformen umgehen. Sie wissen, welche Informationen für eine erfolgreiche Lieferung benötigt werden und brauchen bei der Eingabe keinen Assistenten. Sie wollen alle sich wiederholenden Tasks z. B. „Abholadresse eingeben“ automatisieren beziehungsweise speichern können.

### 5.3.4. Laura Energie

#### Hintergrund & Motivation

Laura ist Geschäftsführerin eines Verlages und publiziert ein monatliches Stadt/Land Magazin, welches sich stark mit den Themen Nachhaltigkeit, Familie und Ernährung auseinandersetzt. Laura ist eine starke Persönlichkeit und trennt ihr Arbeits- und Privatleben strikt. Ihr Geschäft wird rein wirtschaftlich betrieben, was keinen Gewinn abwirft, wird nicht weiterverfolgt.

Trotz der Mehrkosten die ein nachhaltiger KEP-Dienstleister mit sich bringt, wittert Laura die grosse Marketingstrategie. Sie hat einen sehr guten Geschäftssinn und schon früh bemerkt, dass mit Grün- und Biolabels sehr einfach zu werben ist. Als Geschäftsfrau ist ihr die Umwelt nicht gerade an erster Stelle, aber auch sie will etwas zum Umweltschutz bei tragen.

Photo	Steckbrief
	<ul style="list-style-type: none"> <li>▪ Laura Energie</li> <li>▪ 37-Jährig</li> <li>▪ aus Aarau</li> <li>▪ Inhaberin eines Verlags</li> </ul>

Tabelle 5.5.: Persona Steckbrief für Laura Energie

Alter	30 - 50 Jahre
Wohnort	Stadt oder Land
Ausbildung	N/A
Nutzung einer KEP-Dienstleistung / Jahr	> 100 mal

Tabelle 5.6.: Charaktermerkmal für Persona Laura Energie

Lauras Verlag macht wöchentlich Aboabschlüsse im zweistelligen Bereich und will die Ausgabe mit einem Expresskurier verschicken. Laura ist mehr an einer API-Anbindung an ihre Abonnementsverwaltungssoftware interessiert, als an einer Maske, wo die neuen Adressen eingetragen werden. Für eine begrenzte Zeit wäre ein Workaround möglich. Sie interessiert sich sowieso viel mehr für die Zahlen. Eine solche Webapplikation könnte diese liefern.

### Charaktermerkmal

Die Personen sind meist Inhaber oder verantwortlich für die Bereiche Logistik und/oder Versand ihres Unternehmens. Sie brauchen die KEP-Dienstleistung als ein Teil ihres Businessprozesses und sind an einer totalen Automatisierung interessiert. Ihr Interesse an einer Webapplikation liegt alleine im Auswerten von Zahlen und Statistiken.

## 5.4. Persona Stories

Anforderungen werden mithilfe von Persona Stories in der Form „<Persona[:Rolle]> macht <Aufgabe>[, um <Ziel> zu erreichen]“ festgehalten. Die Platzhalter in den eckigen Klammern sind optional und sollen ausgefüllt werden, wenn dabei der Sinn der Story klarer wird. Zusätzlich werden Akzeptanzkriterien für die Stories definiert, welche in einem späteren Schritt als Akzeptanztests dienen. Die Priorität wird mit dem Kano Model bestimmt, welches im Kapitel 5.5 beschrieben und auf die folgenden Persona Stories angewendet worden ist. Die Priorität wird nur der Schönheits halber auch in der Persona Storie Karte aufgeführt.

### 5.4.1. Mara Hürlimann

---

#### PS-001 Abholadresse eingeben

---

*Story*

1. Mara gibt eine Abholadresse ein.

---

*Akzeptanzkriterien*

1. Mara sieht während der Eingabe Vorschläge von Adressen, welche im Versorgungsgebiet liegen.
2. Mara sieht nach Abschluss der Eingabe, ob die Adresse im Versorgungsgebiet liegt.
3. Mara wird nach der Eingabe zum nächsten Eingabeschritt weitergeleitet.

---

*Priorität* Hoch

---

---

#### PS-002 Lieferadresse eingeben

---

*Story*

1. Mara gibt eine Lieferadresse ein.

---

*Akzeptanzkriterien*

1. Mara sieht während der Eingabe Vorschläge von Adressen, welche im Versorgungsgebiet liegen.
2. Mara sieht nach Abschluss der Eingabe, ob die Adresse im Versorgungsgebiet liegt.
3. Mara wird nach der Eingabe zum nächsten Eingabeschritt weitergeleitet.

---

*Priorität* Hoch

---

---

**PS-003 Pakete Dimensionen eingeben**

---

*Story*

1. Mara gibt die Pakete Dimensionen (Gewicht, Länge, Breite und Grösse) ein.
- 

*Akzeptanzkriterien*

1. Mara sieht, ob die Dimension versendet werden können.
  2. Mara wird nach der Eingabe zum nächsten Eingabeschritt weitergeleitet.
- 

*Priorität* Hoch

---

**PS-004 Abholuhrzeit eingeben**

---

*Story*

1. Mara gibt eine Abholuhrzeit ein, um die verfügbaren Routen und deren Preis zu sehen.
- 

*Akzeptanzkriterien*

1. Mara kann eine verfügbare Route auswählen.
  2. Mara wird nach der Auswahl zum nächsten Eingabeschritt weitergeleitet.
- 

*Priorität* Hoch

---

**PS-005 Kontaktinformationen eingeben**

---

*Story*

1. Mara gibt Kontaktinformation (Name, Telefonnummer, evt. Stockwerk) für den Abhol- sowie den Zielort ein.

---

*Akzeptanzkriterien*

1. Mara wird nach der Eingabe zum nächsten Eingabeschritt weitergeleitet.

---

*Priorität* Hoch

---

**PS-006 Zollinformationen eingeben**

---

*Story*

1. Mara gibt bei grenzüberschreitenden Paketen zusätzliche Informationen für den reibungslosen Ablauf beim Zoll ein.

---

*Akzeptanzkriterien*

1. Mara sieht allfällige Dokumente, welche ausgedruckt und mitgeschickt werden müssen.
2. Mara wird nach der Eingabe zum nächsten Eingabeschritt weitergeleitet.

---

*Priorität* Hoch

---

**PS-007 Dienstleistung kaufen**

---

*Story*

1. Mara kauft die Dienstleistung, um den Auftrag abzuschliessen.
- 

*Akzeptanzkriterien*

1. Mara bekommt eine Bestätigung (Mail/Drucken) für ihren Auftragsabschluss.
  2. Mara bekommt eine Nummer, mit welcher der Lieferprozess verfolgt werden kann.
- 

*Priorität* Mittel

---

---

**PS-008 Lieferung verfolgen**

---

*Story*

1. Mara gibt die Trackingnummer ein, um den Status der Lieferung zu überprüfen.
- 

*Akzeptanzkriterien*

1. Mara sieht den Fortschritt ihrer Lieferung.
- 

*Priorität* Niedrig

---

---

**PS-009    Unterstützung Anfordern**

---

*Story*

1. Mara will mit einer Person telefonieren, um Hilfe beim Kauf der Dienstleistung zu bekommen.
- 

*Akzeptanzkriterien*

1. Mara sieht die Telefonnummer oder Emailadresse.
  2. Mara sieht die Auftragsnummer.
- 

*Priorität    Niedrig*

---

### 5.4.2. Peter Elsener

---

#### PS-010 Versandangaben Eingeben

---

*Story*

1. Peter gibt alle Informationen für den Versand ein, um die verfügbaren Routen und deren Preis zu sehen.

---

*Akzeptanzkriterien*

1. Peter kann eine verfügbare Route auswählen.
2. Peter wird zum Schritt „Zoll Informationen“ weitergeleitet.

---

*Priorität* Hoch

---

---

#### PS-011 Zollinformationen Eingeben

---

*Story*

1. Peter gibt alle relevanten Informationen für den Zoll ein.

---

*Akzeptanzkriterien*

1. Peter sieht allfällige Dokumente, welche ausgedruckt und mitgeschickt werden müssen.
2. Peter wird nach der Eingabe zum Schritt „Bezahlung“ weitergeleitet.

---

*Priorität* Hoch

---

---

**PS-012 Abholadressen Speichern**

---

*Story*

1. Peter speichert die Abholadresse und Kontaktinformationen seines Büros, um sie für den späteren Gebrauch wiederverwenden zu können.

---

*Akzeptanzkriterien*

1. Peter sieht seine gespeicherten Adressen und kann sie bearbeiten.
2. Peter kann die gespeicherte Adresse beim PS-009 auswählen.

---

*Priorität* Mittel

---

---

**PS-013 Zieladressen Speichern**

---

*Story*

1. Peter speichert die Zieladressen und Kontaktinformationen seiner Kunden, um sie für den späteren Gebrauch wiederverwenden zu können.

---

*Akzeptanzkriterien*

1. Peter sieht seine gespeicherten Adressen und kann sie bearbeiten.
2. Peter kann die gespeicherte Adresse beim PS-009 auswählen.

---

*Priorität* Mittel

---

---

**PS-014 Auftrag abbrechen**

---

*Story*

1. Peter bricht einen noch nicht gestarteten Auftrag ab.
- 

*Akzeptanzkriterien*

1. Peter sieht, dass der Auftrag abgebrochen ist.
  2. Peter bekommt eine Bestätigung, dass der Auftrag nicht ausgeführt wird.
- 

*Priorität* Mittel

---

---

**PS-015 Aufträge auflisten**

---

*Story*

1. Peter listet die im System erfassten Aufträge auf.
- 

*Akzeptanzkriterien*

1. Peter sieht alle bereits erfassten Aufträge.
  2. Peter kann einzelne Aufträge anklicken und sieht deren Details.
- 

*Priorität* Niedrig

---

### 5.4.3. Laura Energie

---

#### PS-016 One-To-Many Auftrag erfassen

---

*Story*

1. Laura möchte einen Auftrag erfassen, welcher von einer Abholadresse an mehrere Lieferadressen versendet wird.

---

*Akzeptanzkriterien*

1. Laura sieht den Auftrag und die dafür berechnete Route.
2. Laura sieht, wann der Auftrag gestartet wird und wann er beendet sein soll.

---

*Priorität* Niedrig

---

---

#### PS-017 Auftragsstatus überprüfen

---

*Story*

1. Laura kann den Status ihrer Aufträge überprüfen.

---

*Akzeptanzkriterien*

1. Laura sieht den Status ihrer abgeschlossenen und aktuellen Aufträge.

---

*Priorität* Niedrig

---

---

#### PS-018 Routen auswählen

---

*Story*

1. Laura kann für ihren Auftrag zwischen verschiedenen Routen auswählen.

---

*Akzeptanzkriterien*

1. Laura sieht Routen, welche sich im Preis und/oder der Dauer unterscheiden können.

---

*Priorität* Niedrig

---

## 5.5. Priorisierung

Die im Kapitel 5.4 aufgeführten Persona Story Karten werden in der folgenden Tabelle nach dem Kano Modell priorisiert. Das Kano Modell ist im Anhang A.2 ausführlich beschrieben. Aufgrund der fehlenden Endbenutzern, wurde die Priorisierung der Anforderungen mit den beiden Stakeholdern durchgeführt. Der besseren Übersicht halber werden die möglichen Antworten auf die funktionale sowie die dysfunktionale Frage mit Grossbuchstaben wie folgt dargestellt. **A)** Das würde mich sehr freuen **B)** Das setzt ich voraus **C)** Das ist mir egal **D)** Das nehme ich gerade noch hin **E)** Das würde mich sehr stören. Die Persona Stories werden nur durch ihre Nummer PS-XXX identifiziert.

Persona Story	Funktionale Antwort	Dysfunktionale Antwort	Ergebnis
PS-001	B	+	E = Basis-Merkmal
PS-002	B	+	E = Basis-Merkmal
PS-003	B	+	E = Basis-Merkmal
PS-004	B	+	E = Basis-Merkmal
PS-005	B	+	E = Basis-Merkmal
PS-006	B	+	E = Basis-Merkmal
PS-007	A	+	E = Leistungs-Merkmal
PS-008	A	+	C = Begeisterungs-Merkmal
PS-009	A	+	C = Begeisterungs-Merkmal
PS-010	B	+	E = Basis-Merkmal
PS-011	B	+	E = Basis-Merkmal
PS-012	A	+	E = Leistungs-Merkmal
PS-013	A	+	E = Leistungs-Merkmal
PS-014	A	+	E = Leistungs-Merkmal
PS-015	A	+	C = Begeisterungs-Merkmal
PS-016	A	+	C = Begeisterungs-Merkmal
PS-017	A	+	C = Begeisterungs-Merkmal
PS-018	A	+	C = Begeisterungs-Merkmal

Tabelle 5.25.: Priorisierung der Persona Stories nach Kano

Im Anhang A.2 sind die Basis-Merkmale beschrieben als Eingenschaften bzw. Funktionen, welche dem Benutzer auffallen, wenn sie nicht vorhanden sind. Im Rahmen dieser Bachelorthesis werden deshalb alle Basis-Merkmale im Prototypen umgesetzt.

## 5.6. Funktionale Anforderungen

---

**FREQ-001      Adresse automatisch vervollständigen**

---

*PS-Referenz*    PS-001,PS-002,PS-010

*Beschreibung*    Die Applikation soll mit Hilfe einer geeigneten Bibliothek/API nach Eingabe von wenigen Buchstaben Vorschläge zur vollständigen Adresse machen.

---

---

**FREQ-002      Adresse verifizieren**

---

*PS-Referenz*    PS-001,PS-002,PS-010

*Beschreibung*    Die Applikation soll überprüfen, ob die eingegebene Adresse in einem der Versorgungsgebiete liegt, welche bewirtschaftet werden.

---

---

**FREQ-003      Auftrag in Backend-System erstellen**

---

*PS-Referenz*    PS-001,PS-002,PS-004,PS-010

*Beschreibung*    Die Applikation soll im Backend-System einen Auftrag mit einer Start- und Zieladresse sowie einer Start-Uhrzeit erstellen.

---

---

**FREQ-004      Auftrag im Backend-System aktualisieren**

---

*PS-Referenz*    PS-003,PS-005,PS-006,PS-010,PS-011,PS-014

*Beschreibung*    Die Applikation soll die zusätzlichen Angaben (z.B. Kontakt-person, Pakete Dimensionen, Gewicht) im Backend-System aktualisieren.

---

---

**FREQ-005      Auftrag im Backend-System starten**

---

*PS-Referenz*    PS-007

*Beschreibung*    Die Applikation soll den Auftrag im Backend-System starten.

---

---

**FREQ-006      Auftragskosten in Rechnung stellen**

---

*PS-Referenz*    PS-007

*Beschreibung* Die Applikation soll den Auftrag dem Benutzer in Rechnung stellen.

---

**FREQ-007 Auftrag aus dem Backend-System laden**

---

*PS-Referenz* PS-008,PS-014,PS-015

---

*Beschreibung* Die Applikation soll Aufträge aus dem Backend-System laden.

---

**FREQ-008 Auftrag Status synchronisieren**

---

*PS-Referenz* PS-009

---

*Beschreibung* Die Applikation soll den Auftragsstatus bei jeder Änderung synchronisieren.

---

**FREQ-009 Benutzer registrieren**

---

*PS-Referenz* PS-012,PS-013,PS-014,PS-015

---

*Beschreibung* Die Applikation soll einen neuen Benutzer persistent erstellen.

---

**FREQ-010 Benutzer anmelden**

---

*PS-Referenz* PS-012,PS-013,PS-014,PS-015

---

*Beschreibung* Die Applikation soll einen Benutzer mit dem korrekten Passwort anmelden.

---

**FREQ-011 Benutzer verwalten**

---

*PS-Referenz* PS-012,PS-013,PS-014,PS-015

---

*Beschreibung* Die Applikation soll die Benutzer verwalten.

---

**FREQ-012 Daten speichern**

---

*PS-Referenz* PS-012,PS-013,PS-014,PS-015

---

*Beschreibung* Die Applikation soll Daten persistent speichern können.

---

---

**FREQ-013 Daten laden**

---

*PS-Referenz* PS-012,PS-013,PS-014,PS-015

---

*Beschreibung* Die Applikation soll abgespeicherte Daten laden können.

---

## 5.7. Nicht-Funktionale Anforderungen

---

**NFREQ-001 Browser Kompatibel**

---

*Beschreibung* Die Applikation muss in einem Browser funktionieren.

---

---

**NFREQ-002 LoBo Kompatibel**

---

*Beschreibung* Die Applikation muss mit dem Backend-System LoBo funktionieren.

---

---

**NFREQ-003 SSL Zertifikat**

---

*Beschreibung* Die Applikation soll über eine sichere HTTPS Verbindung mit dem Benutzer und dem Backend-System kommunizieren.

---

# 6. Konzept

Aus den Anforderungen ist klar ersichtlich, dass eine eigene Benutzeroberfläche entwickelt werden soll, die über die Programmierschnittstelle von LoBo kommunizieren soll. Die nicht funktionelle Anforderung NFREQ-001 verlangt eine Web Browser-kompatible Benutzeroberfläche und disqualifiziert eine zu installierende Desktop Client Anwendung. Trotz der Limitierung der Lösungsmöglichkeiten bleiben bei der Entwicklung einer Benutzeroberfläche im Web viele Fragen offen. Im Folgenden werden die Technologie- sowie Architekturentscheidungen vorgestellt und begründet.

## 6.1. Web Modell

Traditionelle Websites bestehen aus mehreren *Hyper Text Markup Language* (HTML) Seiten und liefern diese, wenn der Client sie anfordert, aus. Es gibt verschiedene Programmiersprachen, die es erlauben zum Zeitpunkt des Aufrufs den Code, der in diesen HTML Seiten vorhanden ist, zu interpretieren und das entsprechende Resultat auszuliefern. *PHP Hypertext Preprocessor*<sup>1</sup> (PHP) ist eine der bekanntesten Programmiersprachen. Aber auch Python bietet mit dem Framework *Django*<sup>2</sup> eine ähnliche Funktionalität. Mit der zunehmende Möglichkeiten im Web sowie der immer grösseren Komplexität wurden andere Modelle für das Web entwickelt. Mit der offiziellen Spezifikation des *XMLHttpRequest Object* am 5. April 2006 [1] und dessen Einsatz von Google in Webapplikationen wie Gmail und Google Maps wurde die Entwicklung von *Single-page application* (SPA) ermöglicht. Mit *XMLHttpRequest* ist es möglich nach dem Laden der Website eine Anfrage an den Server zu schicken ohne dabei die gesamte HTML Seite neu zu laden. Für die Ausführung dieser *XMLHttpRequest* Anfragen wird eine clientseitige Skriptsprache benötigt. Trotz vereinzelten Alternativen ist JavaScript der Standard bei den browserfähigen clientseitigen Skriptsprachen. Dies ist mitunter auch ein Grund warum SPA Webseiten mit JavaScript erstellt werden. In den Abbildungen 6.1 und 6.2<sup>3</sup> ist der Unterschied zwischen einer konventionellen Webseite und einer SPA veranschaulicht.

### 6.1.1. Single-page application

*Single-page application* (SPA) haben viele Vorteile gegenüber traditionellen Webseiten. Die gesamte Webseite kann mit wenigen Anfragen an den Server geladen werden. Im Wesentlichen werden nur drei Anfragen (1. HTML Seite, 2. Cascading Style Sheets (CSS), 3. JavaScript) benötigt. Alle weiteren Daten wie z.B. Benutzerdaten aus der Datenbank werden asynchron mit einer *XMLHttpRequest* Anfrage, während die Webseite im Browser aufgebaut wird, geladen. Dies hat den Vorteil, dass die Webseite bedeutend schneller geladen wird und trägt damit zur Benutzerfreundlichkeit bei. Im Folgende ist eine nicht abgeschlossene Liste mit den Vorteilen von SPA.

**Entlastung** Die Benutzeroberfläche wird komplett im Browser erstellt und benötigt dafür keine Ressourcen vom Server.

**Entkopplung** Daten und Businesslogik sind nicht mehr an die Visuellen Elemente gebunden.

**Performant** Single-page Applikationen verwalten ihren eigenen Status und benötigten dafür weniger Informationen vom Server.

---

<sup>1</sup>Offizielle Website von PHP: <http://php.net/>

<sup>2</sup>Offizielle Website von Django: <https://www.djangoproject.com>

<sup>3</sup>Bilder von <https://blog.4psa.com/an-intro-into-single-page-applications-spa/>

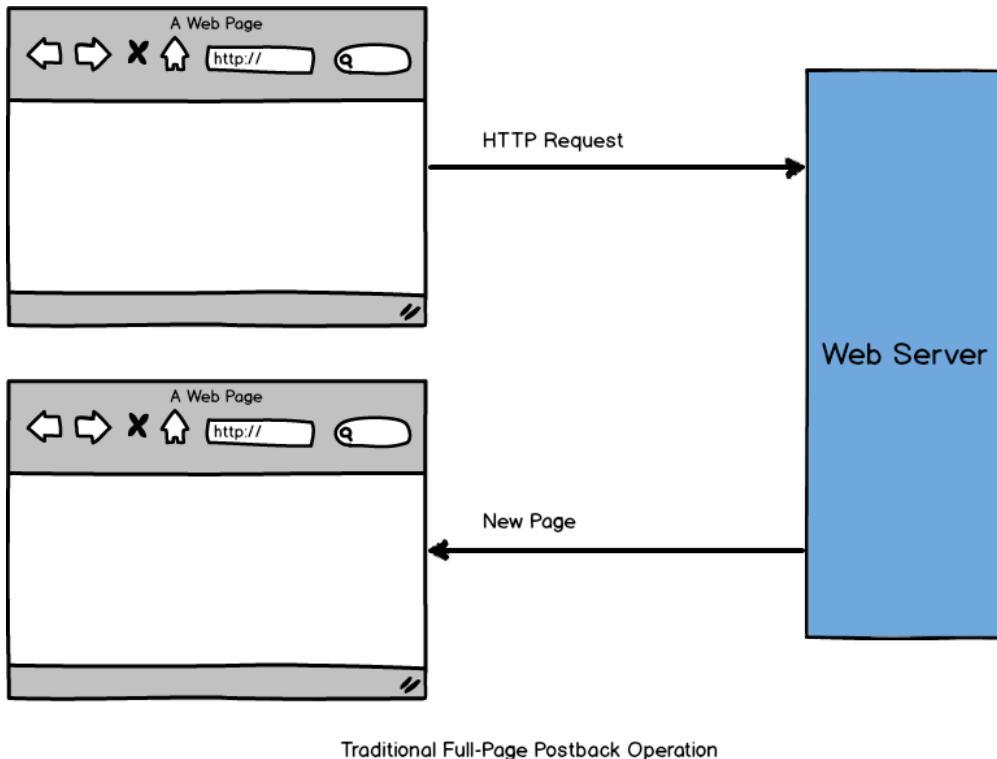


Abbildung 6.1.: Visualisierung einer Anfrage in einer konventionellen Webseite

SPA setzen voraus, dass der Server eine definierte Schnittstelle besitzt, über welche Daten geladen bzw. geschrieben werden können. Dies hat zusätzlich den Vorteil, dass auch Applikationen anderer Plattformen wie z.B. Android oder iOS auf diese Schnittstellen zugreifen können.

Ein grosser Nachteil von Single-Page Applikationen ist die Lesbarkeit des auszuführenden Codes. JavaScript ist eine Skriptsprache, die zur Laufzeit interpretiert und ausgeführt wird. Dies bedeutet dass der gesamte Code im Browser vorhanden ist und für den Benutzer lesbar ist. Es gibt Werkzeuge, die versuchen diesen Code unleserlich zu machen indem Variablen umbenannt und Leerzeichen entfernt werden, aber mit Energie und Geduld kann auch dies wieder lesbar gemacht werden.

### 6.1.2. Mini-Backend

Für die Auslieferung der *Single-page application* wird nur ein Webserver wie z.B. Apache<sup>4</sup> oder Nginx benötigt. Die SPA kann aber auch von einem Webserver ausgeliefert werden, welcher eigene Logik besitzt. Im Rahmen dieser Bachelorarbeit nenne ich die Benutzung eines Webservers mit eigener Logik Mini-Backend. Es existieren mehrere Plattformen in verschiedenen Programmiersprachen, mit welchen die Entwicklung eines Mini-Backend möglich wäre. Node.js<sup>5</sup> ist eine serverseitige Plattform welche auf JavaScript basiert und die Möglichkeit bietet, Webseiten auszuliefern und eigene Logik auszuführen. Diese Logik bzw. dieser Code befindet sich immer auf dem Server und kann von den Benutzer nicht gelesen werden. Node.js kann die gesamte Businesslogik eines Projektes beinhalten oder aber nur Webanfragen weiterleiten.

<sup>4</sup>Apache ist ein weitverbreiteter opensource Webserver <https://httpd.apache.org/>

<sup>5</sup>Node.js ist eine Serverseitige Plattform <https://nodejs.org>

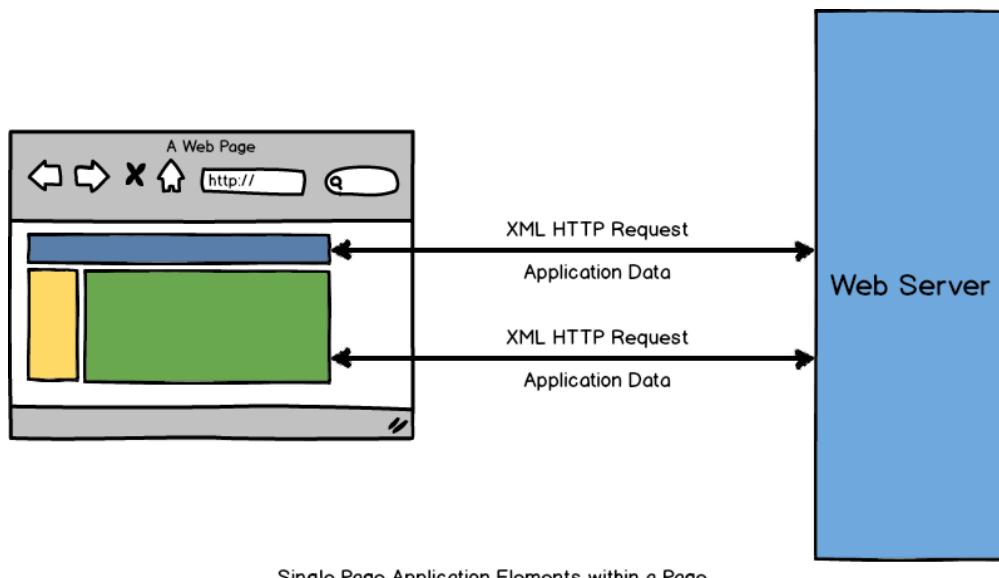


Abbildung 6.2.: Visualisierung einer Anfrage in einer Single-page Applikation

### 6.1.3. Fazit

Obwohl die Logik für die zu entwickelnde Benutzeroberfläche ausschliesslich in der SPA ausgeführt werden könnte, hat der Einsatz eines sogenannten Mini Backends erhebliche Vorteile. LoBo benutzt für die Authentifizierung einen Schlüssel, mit welchem alle Anfragen *gehasht* werden müssen. Der entstandene Hash wird im Header jeder Anfrage mitgeschickt. Dadurch kann LoBo die Authentizität und Integrität der Anfragen sicher stellen. Dieses hashing könnte in der SPA geschehen, hat jedoch den Nachteil, dass der Schlüssel von den Benutzer ausgelesen werden kann. Zusätzlich vereinfacht der Einsatz eines Mini-Backend die Skalierung. Weil die Mini-Backends nicht auf eine Datenbank zugreifen, können einfach mehrere Instanzen gestartet werden und ein zusätzlicher Webserver verteilt die Anfragen auf die verfügbaren Instanzen. Die Mini-Backends können auch mit verschiedenen LoBo Systemen kommunizieren. Dies verhindert einen Flaschenhals bei einem Anstieg der Benutzerzahlen. Im Rahmen dieser Bachelorarbeit wird eine Single-page Applikation mit JavaScript entwickelt und ein Mini-Backend welches auf Node.js basiert. Node.js wurde ausgewählt, weil es in der gleichen Sprache programmiert wird wie die SPA und dadurch einige Vorteile für den Entwickler bietet. Der Entwickler braucht während der Entwicklung nicht den Kontext zu wechseln und kann unter Umständen die gleichen Ressourcen für die SPA wie auch das Mini-Backend verwenden.

## 6.2. JavaScript

JavaScript ist eine zurzeit populäre Programmiersprache, die zu einem grossen Teil für dynamischen Aspekte einer Webseite verantwortlich ist, aber auch Serverseitig zum Einsatz kommt. Obwohl während den Anfängen von JavaScript mehrere Interpretationen der Sprache existierten, ist JavaScript heute der einzige Dialekt welcher auf der EcmaScript Spezifikation basiert und interpretiert wird. JavaScript wird von allen grossen Browsern mit ihren jeweiligen Implementationen unterstützt. Im Folgenden eine Liste der JavaScript Engines der bekanntesten Browsers<sup>6</sup>.

**SpiderMonkey** wird von Mozilla für den Browser Firefox entwickelt.

**V8** wird von Google für den Browser Chrome entwickelt.

**JavaScriptCore** wird von Apple für den Browser Safari entwickelt.

<sup>6</sup>Liste von Wikipedia <https://en.wikipedia.org/wiki/ECMAScript>

**Chakra** wird von Microsoft für den Browser Edge entwickelt.

V8 von Google wird unter anderem auch für Node.js verwendet, welches im Rahmen dieser Bachelorarbeit für das Mini-Backend verwendet wird.

### 6.2.1. ES2015

Die Sprache hat im Juni 2015 eine lang erwartet neue Version bekommen. Ecma International<sup>7</sup> welche die Spezifikationen für die Programmiersprache JavaScript schrieb, hat die Version EcmaScript 2015 veröffentlicht und damit der Sprache einige neue Funktionen verliehen. Mit EcmaScript 2015 (ES2015) können Klassen erstellt werden und mit einem Module System Klassen, Objekte, Funktionen und Variablen exportiert wie auch importiert werden. Dank dieser Erneuerung ist es bedeutend einfacher JavaScript Code zu unterhalten und zu pflegen. Er kann in verschiedene Module unterteilt werden und ist dadurch auch besser lesbar. Einige der Erneuerungen machen auch Helfer Bibliotheken wie *underscore.js* oder *jquery* überflüssig, was wiederum zur Leistung beiträgt.

### 6.2.2. Backend Framework

Für Node.js existieren mehrere Webserver Frameworks, die für den Einsatz im Rahmen dieser Bachelorarbeit eingesetzt werden können. 3 populäre Frameworks sind Express.js<sup>8</sup>, Koa.js<sup>9</sup> und hapi.js<sup>10</sup>. Jedoch Express.js ist das populärste Framework und findet auf Github die grösste Unterstützung.[4]. Die Entwicklung der Mini-Backends benötigt keine aussergewöhnlichen Anforderungen und hat mit Express.js alle Ansprüche abgedeckt.

### 6.2.3. Frontend Framework

In JavaScript existieren unzählige Frameworks, die etliche Anwendungsfälle unterstützen. Obwohl eine Single-Page Applikation auch ohne Framework und Helfer-Bibliothek entwickelt werden kann, können Frameworks zur Stabilität und zum Erfolg der Webseite beitragen. Frameworks wie Angular<sup>11</sup> und Ember.js<sup>12</sup> unterstützen die Entwicklung einer *Model View Controller* (MVC) SPA erheblich. Angular stellt z.B. einen Service (*\$resource*) zur Verfügung, mit welchem alle XMLHttpRequest-Anfragen ausgeführt werden können. Die Handhabung im Falle eines Fehlers kann zentral implementiert werden und auch die Transformation von empfangenen sowie zu versendenden Daten kann mit dem Service verwaltet werden. In Ember.js wird eine Datenverwaltungs-Komponente zur Verfügung gestellt, die die gesamte Synchronisation der Daten mit einem Backend übernimmt. Sofern das Backend eine standardisierte JSON REST<sup>13</sup> Programmierschnittstelle besitzt, kann diese Synchronisation, die Daten erstellt, bearbeitet, lädt und löscht, mit wenigen Zeilen Code implementiert werden. Diese Beispiele sind nur vereinzelte Einblicke in die Möglichkeiten der Frameworks. Frameworks bringen aber auch Nachteile mit sich. Sobald ein Anwendungsfall implementiert werden soll, der nicht dem Framework entspricht, kann es kompliziert und fehleranfällig werden.

Ein relativ neues JavaScript Framework ist *React*<sup>14</sup>, welches von Facebook entwickelt wird. React setzt im Gegensatz zu MVC Frameworks eine andere Philosophie um. Zum einen wird in React mit Komponenten, die wiederum Subkomponenten beinhalten, gearbeitet. Zum anderen wird das Programmierparadigma *Data Down, Actions up* umgesetzt. In diesem Programmierparadigma wird den Komponenten die benötigten Daten übergeben und die Komponenten rufen eine Methode auf, die die bearbeiteten

<sup>7</sup> European Computer Manufacturers Association <http://www.ecma-international.org/>

<sup>8</sup> Web Framework für Node.js <http://expressjs.com/>

<sup>9</sup> Web Framework für Node.js <http://koajs.com/>

<sup>10</sup> Web Framework für Node.js <http://hapijs.com/>

<sup>11</sup> Javascript Framework von Google <https://angularjs.org/>

<sup>12</sup> Javascript Framework aus der OpenSource Gemeinschaft <http://emberjs.com/>

<sup>13</sup> REST steht für *Representational State Transfer* und bezeichnet ein Programmierparadigma

<sup>14</sup> JavaScript Framework von Facebook <https://facebook.github.io/react/>

Daten zurück gibt. Zusätzlich werden die *Templates*, die für die visuelle Repräsentation der Komponenten benötigt werden nicht wie üblich in HTML geschrieben sondern in JSX<sup>15</sup>, womit der gesamte Code einer Komponente in der gleichen Datei geschrieben werden kann. Auch von Facebook stammt die Applikations Architektur *Flux*. *Flux* besteht aus hauptsächlich aus einem *Dispatcher*, *Store* und den *Views* und baut auf einigen wenigen Prinzipien auf, die in der folgenden Liste erklärt werden.

**Unidirektonaler Datenfluss** Daten fließen immer in eine Richtung. Jede Änderung der Daten findet über eine Aktion statt.

**Klare Trennung der Kontrolle** Die Kontrolle über die Daten befindet sich in den *Stores* und diese können von Außen nicht verändert werden.

**Zentralisierter Dispatcher** Es existiert nur ein *Dispatcher*, welcher alle *Stores* über eine neu Aktion informiert. Dadurch wird garantiert, dass immer nur eine Aktion aufs mal ausgeführt wird.

Flux und die Prinzipien sind in der Abbildung 6.3 dargestellt.

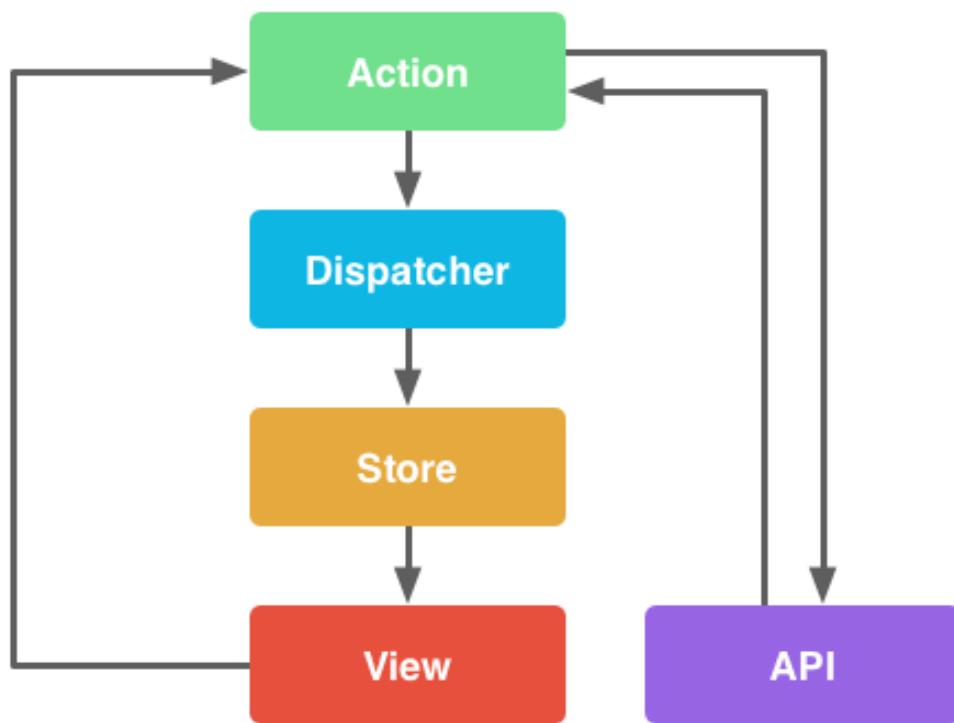


Abbildung 6.3.: Darstellung des Programmierparadigma Flux

Flux und React können kombiniert werden und ergeben dadurch ein stabiles, modernes und fehlerarmes JavaScript Framework.

#### 6.2.4. Fazit

Aus den Anforderungen in Kapitel 5 wird klar, dass die gleichen Eingabemasken an verschiedenen Stellen zum Einsatz kommen. Die Eingabe einer Adresse wird in mehreren Anforderungen benötigt. Für diese Anforderung spricht ein Framework wie React, welches diese Eingabemasken als Komponenten einmal implementiert und an verschiedene Stellen zu Verfügung stellt. Auch der Aufbau einer Schritt für Schritt Eingabe lässt sich in Komponenten und Subkomponenten praktisch umsetzen. Weil für die Erstellung eines Auftrags in LoBo einem klaren Prozess gefolgt werden muss, eignet sich der Einsatz von Flux. Mit Flux ist die gesamte Single-page Applikation weniger störungsanfällig und hat zu jedem Zeitpunkt

<sup>15</sup> JSX steht für *JavaScript Syntax extension* und erlaubt das Schreiben von HTML in JavaScript

einen klaren Zustand. Im Rahmen dieser Bachelorarbeit wird die zu erstellende Benutzeroberfläche mit React und Flux entwickelt.

## 6.3. Prozess

Die Erstellung eines Auftrages hat nur zwei Szenarien. Die Start und Zieladresse sind entweder im gleichen Versorgungsgebiet oder befinden sich in unterschiedlichen. Die Single-Page-Applikation weiss dies nicht und es ist Aufgabe des Mini-Backends dies herauszufinden. Davor muss aber zuerst ein Auftrag erstellt werden. Mit dem gewonnenen *Tasktoken* werden die verifizierten Adressen als Stops hinzugefügt. Danach wird im Mini-Backend die Methode *completetask* aufgerufen, welche herausfinden soll, ob und welche Stops hinzugefügt werden sollen. Die Abbildung 6.4 soll die Aufgabe der Methode grafisch darstellen.

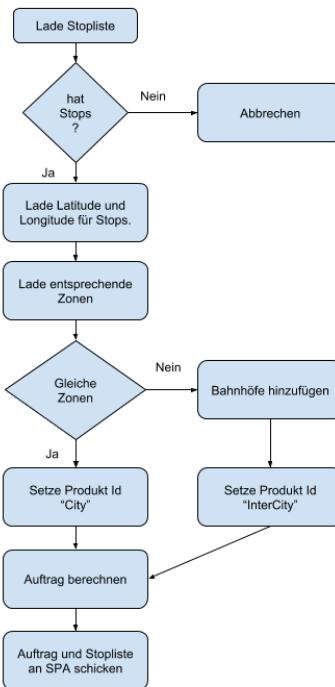


Abbildung 6.4.: Darstellung der *compileTask* Methode

Sollten die Start- und Zieladresse in unterschiedlichen Versorgungsgebieten liegen, werden die verfügbaren Bahnverbindungen angezeigt. Ansonsten werden direkt die Kontaktinformationen für die Start- und Zieladresse gesammelt. Danach wird der Auftrag bestellt und der Benutzer bekommt eine Bestätigung.

## 6.4. Architektur

### 6.4.1. Mini-Backend

Das Backend hat 2 elementare Aufgaben. Das Ausliefern der SPA und das Bereitstellen der Schnittstelle, mit welcher die SPA kommuniziert. Die SPA befindet sich in einem Verzeichnis und wird bei einem Aufruf direkt ausgeliefert. Alle Anfragen der SPA werden an eine bestimmte URL gesendet, die mit der entsprechenden Antwort oder einem Fehler antwortet. Das Mini-Backend soll aber nicht nur die Anfragen

von der SPA an LoBo weiterleiten sondern auch die Komplexität gewisser Prozessabläufe vereinfachen. Obwohl LoBo viele Schnittstellenendpunkte zur Verfügung stellt, werden für gewisse Funktionen mehrere Anfragen benötigt. Für die Abfrage ob eine bestimmte Straßenadresse in einem Gebiet liegt, welches von Imagine Cargo versorgt wird, werden 3 Anfragen an LoBo benötigt. Zuerst muss in LoBo ein neuer Auftrag erstellt werden, mit dem dadurch gewonnenen Auftragstoken muss eine Adresse hinzugefügt werden. Zuletzt muss eine Anfrage gestellt werden, die überprüft ob der Auftrag konform ist. Die Fehlermeldung dieser Anfrage bestimmt, ob die Adresse in einem Versorgungsgebiet von ImagineCargo liegt. Der gesamte Ablauf ist in Abbildung 6.5 grafisch dargestellt. Darin ist zu sehen, wie viel Arbeit vom Mini-Backend gehandhabt wird.

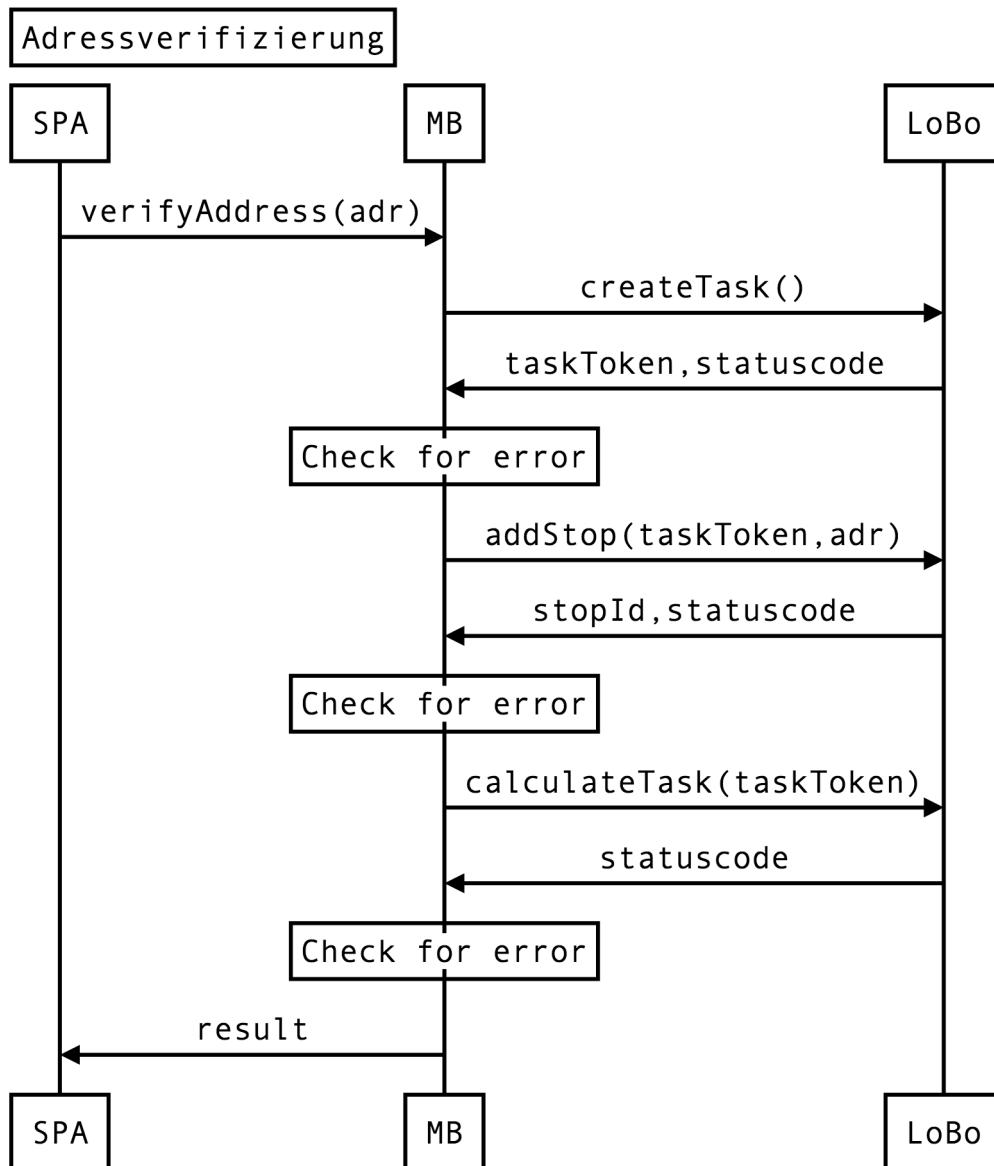


Abbildung 6.5.: Darstellung des Prozesses zur Verifizierung einer Adresse

Somit braucht das Mini-Backend nur die Schnittstellenendpunkte zur Verfügung zu stellen, welche auch zwingend von der SPA gebraucht werden. Diese Schnittstellenendpunkte sind in der folgende Liste aufgeführt und beschrieben.

**createtask** erstellt in Lobo einen neuen Auftrag und übergibt den *tasktoken* an die SPA für alle weiteren anfragen.

**verifyaddress** überprüft in einem temporären Auftrag ob die Adresse im Versorgungsgebiet liegt

**addstop** fügt dem Auftrag eine Adresse hinzu.

**completetask** überprüft ob dem Auftrag noch weitere Stops (z.B. Bahnhöfe) hinzugefügt werden müssen-

**updatestopinfo** fügt Informationen (z.B. Kontaktpersonen) den einzelnen Adressen hinzu.

**updatestoptime** fügt die Abfahrts- und Ankunftszeit den Stops hinzu.

**updatereftime** erneuert die Uhrzeit wann der Auftrag gestartet wird.

**connections** lädt alle Verbindungen zwischen 2 Städten um eine gewisse Uhrzeit.

**ordertask** finalisiert den Auftrag.

Zusätzlich soll das Mini-Backend soviel Fehler handhaben wie möglich. Die SPA soll Fehler nur bekommen wenn ein Dienst nicht verfügbar ist oder ein schwerwiegender Fehler passiert ist.

#### 6.4.2. SPA

Die primäre Aufgabe der Single-page Applikation ist das Aufnehmen der richtigen Benutzerinformationen und Wiedergeben der daraus berechneten Daten. Die SPA wird in Komponenten aufgeteilt, die einzelne Arbeitsschritte ausführen. Diese Komponenten werden in sogenannten *Step*-Komponenten zusammengefasst. In der Abbildung 6.6 sind die Komponenten und ihre Verwendung grafisch dargestellt. In den folgenden Kapiteln werden die Funktionen der Komponenten beschrieben.

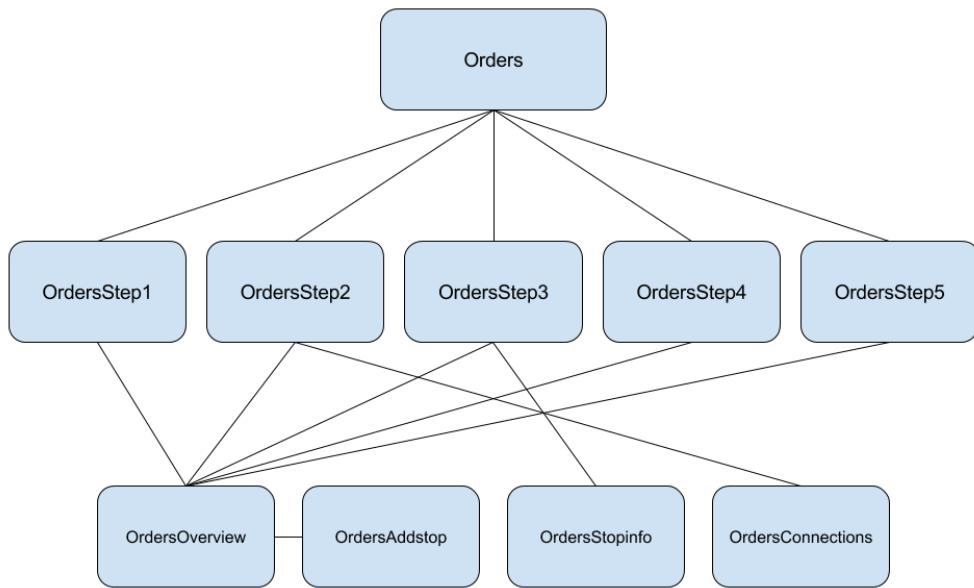


Abbildung 6.6.: Darstellung der Komponenten und ihrer Verwendung

## Orders

Die Orders Komponente hat keine visuelle Repräsentation. Je nach URL-Pfad wird eine der vier OrdersStep Komponente dargestellt. Die Orders Komponente hat die Aufgabe einen Auftrag zu erstellen und den *tasktoken* zu speichern. Beim Neuladen der Webseite, ist die Orders Komponente dafür verantwortlich, den letzten Auftrag wieder Aufzunehmen und den entsprechenden Schritt zu laden.

### OrdersOverview

Die OrdersOverview Komponente wird in jedem Schritt verwendet und zeigt alle relevanten Informationen wie Startzeit, Start- und Zieladresse sowie den Preis des Auftrags. Zusätzlich verwendet die Komponente eine weitere Komponente, welche zur Eingabe der Adressen eingesetzt wird.

### OrdersAddstop

Die OrdersAddstop Komponente versucht die vom Benutzer eingegebene Adresse zu vervollständigen. Dies geschieht mit der Programmierschnittstelle von Google Maps<sup>16</sup>, welche während der Eingabe des Benutzer versucht, die fertige Adresse anzubieten. Die Komponente verifiziert die Adresse ebenfalls mit LoBo.

### OrdersStopinfo

Die OrdersStopinfo Komponente bietet dem Benutzer Eingabefelder an, die zur Eingabe für Informationen wie Kontaktperson, Telefonnummer und sonstige Hinweise verwendet werden können.

### OrdersConnections

Die OrdersConnections Komponente lädt die verfügbaren Bahnverbindungen zwischen zwei Standorten. Die Bahnverbindungen werden mithilfe der Programmierschnittstelle von Opendata.ch<sup>17</sup> geladen. Nach erfolgreicher Auswahl des Benutzers werden die Zeitinformationen (Abfahrts- und Ankunftszeit sowie die jeweiligen Bahnsteige) in den jeweiligen Stops in LoBo gespeichert.

### OrdersStep1

Die OrdersStep1 Komponente ist verantwortlich nach Eingabe der Start und Zieladresse zu übermitteln und den Auftrag zu vervollständigen. Nach erfolgreicher Ausführung wird entweder zur OrdersStep2 oder OrdersStep3 Komponente gewechselt.

### OrdersStep2

Die OrdersStep2 Komponente ist verantwortlich, die OrdersConnections Komponente darzustellen und nach erfolgreicher Bahnverbindungsauswahl zur OrderStep3 Komponente zu wechseln.

### OrdersStep3

Die OrdersStep3 Komponente ist verantwortlich, für die jeweiligen Stops (Start und Ziel) die OrdersStopinfo Komponenten anzuzeigen und eine klickbare Aktion, welche den Auftrag vollständig in LoBo einträgt und nach erfolgreicher Durchführung zu der OrdersStep4 Komponente wechselt.

<sup>16</sup>Goole Maps Programmierschnittstelle <https://developers.google.com/maps/web>

<sup>17</sup>transport.opendata.ch Programmierschnittstelle <https://transport.opendata.ch/>

**OrdersStep4**

Die OrdersStep4 Komponente ist verantwortlich, Eingabefelder für Informationen über das Paket anzuzeigen. Zusätzlich wird eine klickbare Aktion angezeigt, mit welcher zum nächsten Schritt gewechselt werden kann.

**OrdersStep5**

Die OrdersStep5 Komponente ist verantwortlich, Links zu dem erstellten Bestätigungs PDF sowie zur Auftragsstatus Seite bereit zu stellen.

# 7. Prototyp

## 7.1. Tools

### 7.1.1. Entwicklungsumgebung

Für die Versionsverwaltung wird Github<sup>1</sup> verwendet. Als Entwicklungsumgebung wurde auf WebStorm<sup>2</sup> von JetBrians<sup>3</sup> zurückgegriffen. WebStorm bietet viele Tools für effizientes Programmieren in JavaScript. WebStorm hilft die Programmierstandards einzuhalten und vervollständigt viele Eingaben automatisch. Eine zusätzliche grosse Hilfe ist die Erkennung von *NPM Tasks*, welche dann als eine ausführbare Konfiguration abgespeichert werden können. Dadurch lässt sich die gesamte Entwicklungsumgebung mit wenigen Handgriffen aus einem Programm starten.

### 7.1.2. Browser

Für das Ausführen der Single-Page Applikaiton wurde die neuste Version des Google Chrome Canary<sup>4</sup> Browsers verwendet. Zusätzlich wurden zwei Browser Erweiterungen installiert, mit welchen die Entwicklung und das Testen verbessert wurde.

#### React Erweiterung

Die React Browser Erweiterung erlaubt es den Zustand einer React Komponente zur Laufzeit zu inspirieren. Die übergebenen Parameter sowie die verfügbaren Eigenschaften werden angezeigt und können auch angepasst werden. Der Inhalt des *Stores* sowie der des Komponenten internen Status sind verfügbar. In Abbildung 7.1 ist die Ansicht der React Browser Erweiterung mit der OrdersStep2 Komponente zu sehen.

#### Redux Erweiterung

Die Redux Browser Erweiterung zeichnet alle ausgeführten Aktionen auf und listet diese auf einer Zeitachse auf. Die Aktionen können inspiziert und wieder rückgängig gemacht werden. Es wird aufgezeigt, wie der Zustand des *Stores* vor und nach der Aktion aussieht. In Abbildung 7.2 ist die Ansicht der Redux Browser Erweiterung zu sehen.

---

<sup>1</sup>[https://github.com/roosnic1/zhaw\\_bachelor](https://github.com/roosnic1/zhaw_bachelor)

<sup>2</sup>WebStorm Version: 2016.1.3

<sup>3</sup><https://www.jetbrains.com/webstorm/>

<sup>4</sup>Version: 54.0.2793.0 canary (64-bit)

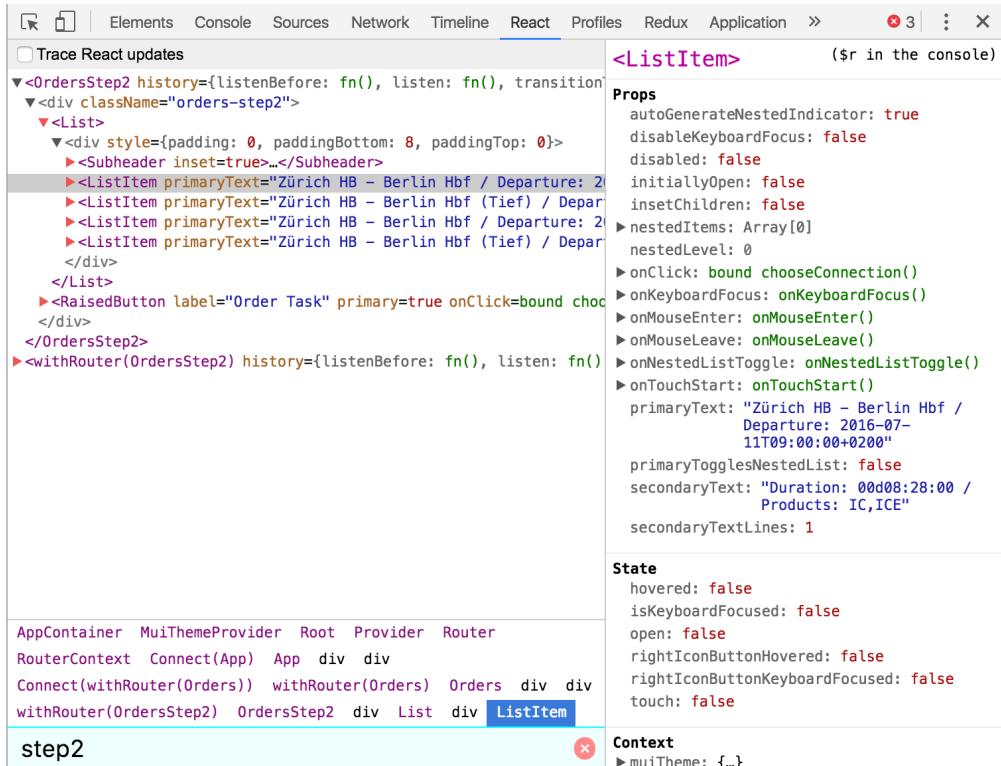


Abbildung 7.1.: React Browser Erweiterung

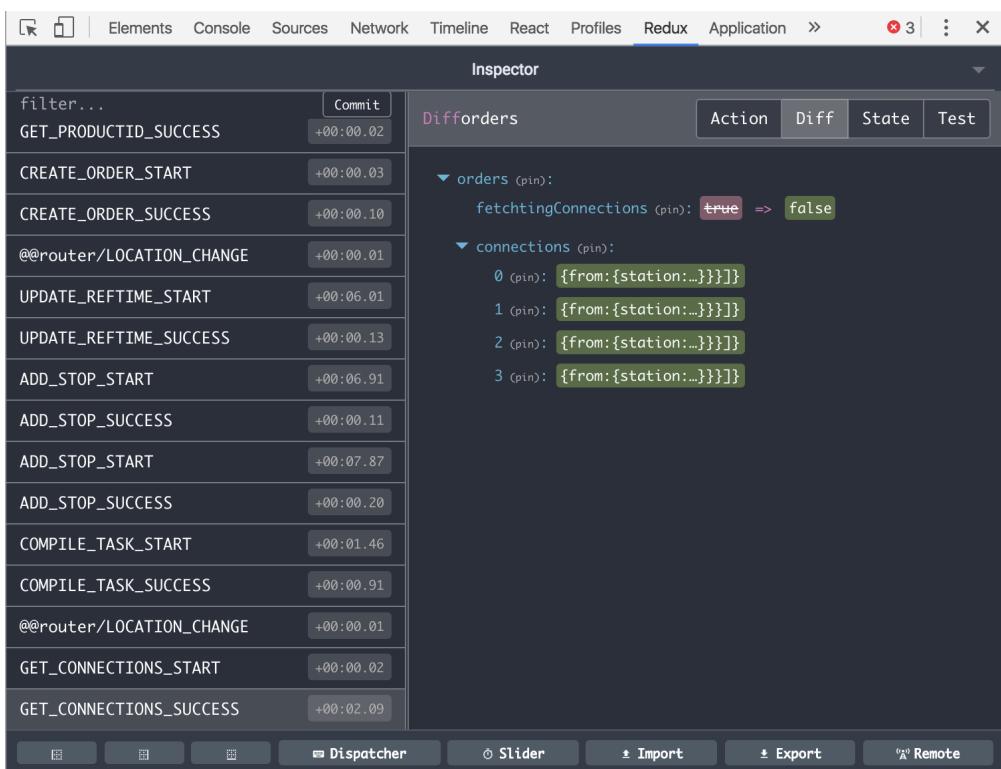


Abbildung 7.2.: Redux Browser Erweiterung

### 7.1.3. Test Server

Als Test Server wurde ein virtueller Ubuntu 16.04 Server verwendet. Als Webserver wurde nginx<sup>5</sup> verwendet. Der Webserver funktioniert als Proxy für den lokal laufende Node.js Webserver. Für Node.js wurde pm2<sup>6</sup> verwendet. Mit pm2 werden Node.js Prozesse überwacht, deren Standard Ausgabe abgefangen und gespeichert und im Falle eines Absturzes neu gestartet wird.

## 7.2. Codebasis

Für die Entwicklung wurde auf einer *ToDo* Single-Page Applikation aufgebaut. Viele Software Frameworks oder Software Paradigmen werden in einem *ToDo* Projekt vorgestellt und lassen sich als Start für eigene Projekte verwenden. Im Rahmen dieser Bachelorarbeit wurde auf das *ToDo-React-Redux* Projekt von Richard Park auf Github<sup>7</sup> zurückgegriffen. Das Projekt implementiert eine einfach *ToDo* Applikation mit React und Redux. Die Daten werden mit Firebase<sup>8</sup> synchronisiert und benötigen deshalb eine asynchrone Handhabung. Dies ist hilfreich, weil die Kommunikation mit dem Mini-Backend auch asynchron stattfindet. Für das Mini-Backend ist bereits eine einfache express.js Applikationen vorhanden. Zusätzlich verwendet das Projekt Werkzeuge wie Webpack, Babel, React-Redux und ESLint, welche im Folgenden genauer beschrieben werden.

### 7.2.1. Webpack

Webpack<sup>9</sup> ist ein modernes Werkzeug, welches hilft eine grosse Codebasis in statische Dateien zu transformieren. Diese statischen Daten können dann zur Verwendung in einer produktiven Umgebung verwendet werden. Die neuste Version von JavaScript erlaubt es, Inhalte zu exportieren beziehungsweise zu importieren. Dadurch kann die Applikation in viele verschiedene Teile aufgeteilt werden, die am Anfang ihres Codeteils auf diejenigen Codeteile verweisen, die zusätzlich benötigt werden. Diese Abhängigkeiten werden von Webpack für einen erfolgreichen produktiven Einsatz gelöst. Zusätzlich ermöglicht Webpack mit sehr wenigen Zeilen einen lokalen Webserver zu starten. Im Folgenden ist die Konfiguration für den verwendeten lokalen Webserver.

```

1  config.devServer = {
2    contentBase: './src',
3    historyApiFallback: true,
4    host: HOST,
5    hot: true,
6    port: PORT,
7    publicPath: config.output.publicPath,
8    stats: {
9      cached: true,
10     cachedAssets: true,
11     chunks: true,
12     chunkModules: false,
13     colors: true,
14     hash: false,
15     reasons: true,
16     timings: true,
17     version: false
18   },
19   proxy: {
20     '/api/v1/*': {
21       target: 'http://localhost:3001',
22       secure: false
23     }
24   }

```

<sup>5</sup><https://nginx.org>

<sup>6</sup><https://github.com/Unitech/pm2>

<sup>7</sup><https://github.com/r-park/todo-react-redux>

<sup>8</sup>Firebase ist ein *Backend as a Service* Anbieter. <https://firebase.google.com/>

<sup>9</sup><https://github.com/webpack/webpack>

25 };

Listing 7.1: Lokaler Webserver

Dieser lokale Webserver ist nur für die SPA zuständig und reagiert nur auf deren Dateiänderungen während der Entwicklung. Für das Mini-Backend wird ein eigener lokaler Webserver verwendet, welcher nur auf Dateiänderungen im Mini-Backend reagiert. In einer produktiven Umgebung werden SPA sowie Mini-Backend vom gleichen Webserver bedient, was zum Problem führt, dass während der Entwicklung die Anfragen an das Mini-Backend umgeleitet werden müssen. In Webpack ist dies mit dem Parameter Proxy möglich, welcher alle Anfragen auf eine gewisse URL umleitet.

## 7.2.2. Babel

Babel<sup>10</sup> ist ein JavaScript transpiler, der es erlaubt, die neueste Version von JavaScript für alle Browser ausführbar zu machen. Obwohl die neueste Version von JavaScript bereits im Juni 2015 veröffentlicht wurde, sind noch nicht alle Browserhersteller in der Lage all diese Erneuerungen in ihren Browser auszuführen. Babel transpiliert diesen neuen Code in bekannten JavaScript ES5 Code, der selbst von älteren Browsern ausgeführt werden kann. Babel wird mit Webpack verwendet und hat den grossen Vorteil, dass die Codebasis auf viele neue und bessere Programmierungseigenschaften zugreifen kann.

## 7.2.3. React-Redux

React-Redux<sup>11</sup> ist eine Erweiterung für React und Redux, welche die beiden Frameworks miteinander verbindet. Redux sieht vor dass es nur einen *Store* gibt, der alle Daten für die gesamte Applikation beinhaltet. Dieser Store wird auf der obersten Ebene von React definiert und einer Provider Komponente übergeben.

```

1  export default function Root({history, onEnter, store}) {
2    return (
3      <Provider store={store}>
4        <Router history={history}>
5          <Route component={App} onEnter={onEnter} path="/" >
6            <IndexRoute component={Home} />
7            <Route component={Orders} path={ORDERS_PATH}>
8              <IndexRoute component={OrdersStep1} />
9              <Route component={OrdersStep2} path={ORDERS_PATH + '/step2'} />
10             <Route component={OrdersStep3} path={ORDERS_PATH + '/step3'} />
11             <Route component={OrdersStep4} path={ORDERS_PATH + '/step4'} />
12           </Route>
13           <Route component={SignIn} path={SIGN_IN_PATH} />
14           <Route component={Tasks} path={TASKS_PATH} />
15         </Route>
16       </Router>
17     </Provider>
18   );
19 }
```

Listing 7.2: Root Komponente

Damit dieser *Store* nicht durch alle Subkomponenten durchgereicht werden muss, wird mit react-redux das Werkzeug *connect* zur Verfügung gestellt. Damit ist es möglich, bei der Definition einer Komponente zu definieren, welcher Teil des *Stores* in der Komponente bereit gestellt werden soll. Im Folgenden Codebeispiel ist zu sehen, wie ein Teil des *Stores* der Orders Komponente mit *connect* zur Verfügung gestellt wird.

<sup>10</sup><https://babeljs.io/>

<sup>11</sup><https://github.com/reactjs/react-redux>

```

1  export default connect(state => {
2    return {
3      orders: state.orders
4    };
5  }, Object.assign({}, ordersActions))(Orders);

```

Listing 7.3: connect im Einsatz

## 7.2.4. ESLint

ESLint<sup>12</sup> ist ein JavaScript linter, der auf Plugins aufbaut. Im Gegensatz zu anderen Codelintern ist ESLint in der Lage, JSX Code zu interpretieren und eignet sich deshalb für React Softwareprojekte. Ein Linter trägt stark zur Qualität und Verwaltbarkeit von Softwareprojekten bei.

# 7.3. Entwicklungsprozess

## 7.3.1. Mini-Backend

Das express.js wurde um eine isolierte Schnittstelle erweitert, die unter der URL (/api/v1/\*) die definierten Methoden des Mini-Backends zur Verfügung stellt. Methoden, die nur Daten zurückliefern sind mit HTTP GET aufzurufen, alle anderen Methoden müssen mit HTTP POST aufgerufen werden. Jede Schnittstellen-Methode, die eine Anfrage an LoBo sendet, benötigt eine gewisse Bearbeitung, welche im Folgenden beschrieben ist.

### LoBo Anfragen

Für eine erfolgreiche Anfrage an LoBo müssen gewisse Eigenschaften erfüllt sein. Die Parameter müssen als Strings gesendet werden. Zusätzlich zur gewünschten Anfrage und den benötigten Parametern muss das gewünschte Antwortformat mitgeschickt werden. Die Parameter müssen in einem URL-Encoded Form gesendet werden. Im Header der Anfrage werden die Client Id und den Hash der zu sendenden Parameter erwartet. Für all diese Anforderungen wurde eine Methode geschrieben, welche diese Aufgaben übernahm.

```

1  function createLoboRequest(action, params = {}) {
2
3    // Ensure that all parameters are strings
4    for (let key in params) {
5      if ({}.hasOwnProperty.call(params, key)) {
6        params[key] = String(params[key]);
7      }
8    }
9
10   // Combine Parameters with defaults
11   const postArray = Object.assign({
12     action: action,
13     clientIp: '127.0.0.1',
14     responseFormat: 'json'
15   }, params);
16
17   // Create Hash
18   const shaObj = new JSSHA('SHA-256', 'TEXT');
19   shaObj.setHMACKey(CONFIG.LOBO_API_PRIVATE_KEY, 'TEXT');
20   shaObj.update(JSON.stringify(postArray));
21
22   return {
23     url: CONFIG.LOBO_API_URL,
24     method: 'POST',

```

<sup>12</sup><http://eslint.org/>

```

25     headers: {
26       'X-Client-Id': CONFIG.LOBO_API_CLIENT_ID,
27       'X-Public-Hash': shaObj.getHMAC('HEX')
28     },
29     form: postArray
30   };
31 }

```

Listing 7.4: Methode für LoBo Anfragen

### Request Promises

Gewisse Methoden wie die in Kapitel 6.4 beschriebene *verifyaddress*, benötigen viele Anfragen an LoBo und andere Schnittstellen. Für die Handhabung dieser Abfolge von verschiedenen Anfragen wurde ein Node.js Modul<sup>13</sup> verwendet, welches erlaubt Anfragen als *Promise* auszuführen. JavaScript *Promise* sind Funktionen, die positiv oder negativ aufgelöst werden können. Zusätzlich sind sie in der Lage in Serie zu funktionieren. Dies erlaubt, dass auf eine asynchrone Antwort gewartet und dann erst die nächste Aktion ausgeführt wird. Gleichzeitig bleibt der Code sehr übersichtlich und ein Fehler kann jederzeit abgefangen werden.

### Polygone

Die Polygone werden benötigt um herauszufinden, ob und welche Bahnhöfe hinzugefügt werden sollen. Mit LoBo lässt sich zwar eine Adresse verifizieren, aber es gibt keine Möglichkeit herauszufinden, ob zwei Adressen im selben Versorgungsgebiet liegen. Obwohl LoBo in der Lage wäre, die vorhandenen definierten Versorgungsgebiete auszugeben, hatte die Version von LoBo, welche im Rahmen dieser Bachelorarbeit verwendet wurde, diese Funktion deaktiviert. Für den Prototypen wurde für die zwei Städte Zürich und Berlin eigene Polygone erstellt. Mit der Web Applikation iso4app<sup>14</sup> können Erreichbarkeitskarten erstellt werden. Mit einer Zeitangabe und einem Startort wird ein Polygon erstellt, welches für die Punkte die Längen und Breitengrade abspeichert. Für Zürich und Berlin wurden die Polygone mit einem Zeitradius von 60 Minuten erstellt. Die einzugebenden Adressen wurden mit Hilfe von Google Maps automatisch vervollständigt. Dies hatte den Vorteil, dass der Längen- und Breitengrad der Adresse ebenfalls bekannt war. Im Mini-Backend wurden zwei Node.js Module<sup>15</sup> hinzugefügt, welche halfen herauszufinden, ob sich ein Punkt in einem Polygon befand oder nicht.

## 7.3.2. SPA

### Aktionen

Aktionen, welche in Redux definiert sind, werden synchron ausgeführt. Um eine asynchrone Anfrage mit Redux an das Mini-Backend auszuführen, müssen mehrere Aktionen definiert werden. Bevor die Anfrage ausgeführt wird, wird dem *Dispatcher* eine START Aktion übergeben. Sobald die Antwort vom Mini-Backend zurückkommt, wird je nach Status der Antwort die SUCCCES Aktion für eine erfolgreiche Anfrage oder die ERROR Aktion für eine fehlgeschlagene dem *Dispatcher* übergeben. Im Reducer wird beim Empfang der START Aktion ein Flag gesetzt, welches aussagt, dass eine asynchrone Aktion ausgeführt wird. Beim Empfang der erfolgreichen beziehungsweise fehlgeschlagenen Aktion wird dieses Flag wieder zurück gesetzt. Im Folgenden ist dieses Redux Verhalten an der Aktion, welche die verfügbaren Bahnverbindungen lädt, dargestellt.

<sup>13</sup>NPM Module request-promise <https://github.com/request/request-promise>

<sup>14</sup><http://iso4app.net/>

<sup>15</sup>Polygon.js <https://github.com/tmpvar/polygon.js>, Vec2.js <https://github.com/tmpvar/vec2.js>

```

1  export function getConnections(from,to,date,pickup) {
2    return dispatch => {
3      dispatch({type: GET_CONNECTIONS_START, payload: null});
4      const connections = {
5        'method': 'POST',
6        'headers': {'Content-Type': 'application/json'},
7        'body': JSON.stringify({from, to, date, pickup})
8      };
9      return fetch('/api/v1/connections',connections)
10     .then(data => data.json())
11     .then(json => {
12       dispatch({
13         type: GET_CONNECTIONS_SUCCESS,
14         payload: json
15       });
16     })
17     .catch(error => {
18       dispatch({
19         type: GET_CONNECTIONS_ERROR,
20         payload: error
21       });
22     });
23   }
24 }

```

Listing 7.5: Aktion getConnections

Die Methode *getConnections* wird mit den Argumenten from, to, date und pickup aus einer Komponente aufgerufen. Zuerst wird der Aktionstyp *GET\_CONNECTIONS\_START* auf Zeile drei an den *Dispatcher* übergeben. Auf Zeile vier bis acht wird die HTTP POST Anfrage zusammen gestellt. Auf Zeile neun wird mit *fetch* die Anfrage ausgeführt und sobald eine Antwort zurückkommt auf Zeile zehn von JSON in ein JavaScript Objekt umgewandelt. Dieses wird mit dem Aktionstyp *GET\_CONNECTIONS\_SUCCESS* auf Zeile 12 bis 15 dem *Dispatcher* übergeben. Sollte ein Fehler zurückkommen, passiert das gleiche mit dem Aktionstyp *GET\_CONNECTIONS\_ERROR* auf Zeile 18 - 21.

```

1  case GET_CONNECTIONS_START:
2    return Object.assign({}, state, {fetchingConnections: true});
3  case GET_CONNECTIONS_SUCCESS:
4    return Object.assign({}, state, {
5      fetchingConnections: false,
6      connections: action.payload
7    });
8  case GET_CONNECTIONS_ERROR:
9    return Object.assign({}, state, {
10      fetchingConnections: false,
11      connections: []
12    });

```

Listing 7.6: Reducer getConnections

Sobald der *Reducer* den Aktionstyp *GET\_CONNECTIONS\_START* erkennt, wird das Attribut *fetchingConnection* auf Wahr geschaltet. Wenn eine positive Antwort zurückkommt, wird der Aktionstyp *GET\_CONNECTIONS\_SUCCESS* empfangen und die Bahnverbindungen im *Store* unter *connections* abgespeichert. Sollte eine negative Antwort zurückkommen wird *connections* mit einem leeren Array überschrieben.

## Komponenten

Komponenten bestehen aus einer JavaScript Klasse, welche von den React Komponenten erben. Die einzige Methode, welche zwingend ist, ist die *render* Methode. Die *render* Methode wird beim Laden der Single-Page Applikation ausgeführt und das zurückgegebene JSX im Browser dargestellt.

```

1  renderConnections() {
2    const { orders } = this.props;
3    if(orders.connections.length > 0) {

```

```

4     return (
5       <List>
6         <Subheader inset={true}>Connections</Subheader>
7         {orders.connections.map(connection => {
8           return (
9             <ListItem
10               primaryText={connection.from.station.name + ' - ' + connection.to.
11                 station.name + ' / Departure: ' + connection.from.departure}
12                 secondaryText='Duration: ' + connection.duration + ' / Products: ' +
13                 connection.products.join()}
14                 onClick={this.chooseConnection.bind(this, connection)}
15               />
16             );
17           })}
18         </List>
19       )
20     }
21   }
22   render() {
23     return (
24       <div className="orders-step2">
25         {this.renderConnections()}
26         <RaisedButton label="Order Task" primary={true} onClick={this.
27           chooseConnection.bind(this)} />
28       </div>
29     );
30   }

```

Listing 7.7: Render Methode der OrderStep2 Komponente

In der `render` Methode können andere Methoden aufgerufen werden, welche wiederum JSX zurück geben. In der `renderConnections` Methode wird über ein Array aus dem *Store* iteriert, welches für jede Bahnverbindung ein *ListItem* mit den entsprechenden Eigenschaften zurück gibt.

## 8. Fazit

Trotz der klaren Strukturen, die dieser Bachelorarbeit zu Grunde lagen, gab es einige Hürden zu nehmen respektive zu umgehen. Die Wahl dieser Verbesserung der Benutzererfahrung mit Hilfe des User Centered Design Prozesse zu bewältigen war richtig. Die Ausführung jedoch benötigte bedeutend mehr Zeit als ursprünglich angenommen und hat ohne die Kompromisse gewaltig mehr Ressourcen gebraucht. Die korrekte Überprüfung der erstellten Prozesse, der Wireframes und des Prototypen durch Testbenutzern, die keine Verbindung zum Projekt oder der Auftraggeberfirma haben, wäre sehr wichtig gewesen. Dank der iterativen Natur des User Centerd Design Prozesses ist dieses Versäumnis jedoch in der nächsten Iteration ohne Probleme nachzuholen. Im allgemeinen lässt dieses iterative Vorgehen mehr Fehler zu, was jedoch dazu führen kann, mehr auszuprobieren und zu wagen. Das Buch von Kim Goodwin „Designing for the Digital Age“, welches stark für die Durchführung des User Centerd Designs zu Rate gezogen wurde, verweist bei der Suche nach Testbenutzern auf professionelle Firmen, die solche Personen stellen. Die Annahme, in der freien Wirtschaft Menschen zu finden, die helfen ein Produkt, das sie selber brauchen zu verbessern, wurde klar widerlegt. Die Wahl der Technologien führte trotz ihrer Neuheit zu keinen erheblichen Problemen. In gewissen Situationen war es schwierig Informationen oder Hilfe für kleinere Probleme beziehungsweise architektonische Fragen zu bekommen. Aber davon abgesehen waren sie die richtige Wahl, um diesen Prototypen damit zu entwickeln. ImagineCargo besitzt nun eine Grundlage, um eine Web Applikation zu entwickeln, mit welcher ein Grossteil der Benutzer zum ersten Mal mit dem Service in Berührung kommen.

## **9. Verzeichnisse**

# Literaturverzeichnis

- [1] Dean Jackson Anne van Kesteren. The xmlhttprequest object. <https://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/>, 2006.
- [2] H. Baum. *Produktivitäts- und Wachstumseffekte der Kurier-, Express- und Paketdienste für die arbeitsteilige Wirtschaft: Studie für den Bundesverband Internationaler Express- und Kurierdienste e.V.* BIEK, Bundesverband Internat. Express- und Kurierdienste, 2004. URL <https://books.google.ch/books?id=haQ5MgAACAAJ>.
- [3] Die Bundesversammlung der Schweizerischen Eidgenossenschaft. Postgesetz (pg) sr 783.0. <https://www.admin.ch/opc/de/classified-compilation/20070597/201210010000/783.0.pdf>, Dezember 2010.
- [4] Jonathan Glock. Node.js framework comparison: Express vs. koa vs. hapi. <https://www.airpair.com/node.js/posts/nodejs-framework-comparison-express-koa-hapi>.
- [5] K. Goodwin and A. Cooper. *Designing for the Digital Age: How to Create Human-Centered Products and Services*. Wiley, 2011. ISBN 9781118079881.
- [6] William Hudson. User stories don't help users: Introducing persona stories. <http://interactions.acm.org/archive/view/november-december-2013/user-stories-dont-help-users-Introducing-persona-stories>, Dezember 2013.
- [7] Madleen Mencler. Entwicklung von speditionen und kep-dienstleistern in Österreich. 2006.
- [8] N/A. Edifact. [http://www.thopas.com/sources/Diverse\\_pdfs/EDIFACT.pdf](http://www.thopas.com/sources/Diverse_pdfs/EDIFACT.pdf).
- [9] Eidgenössische Postkommission PostCom. Postcom jahresbericht 2015. <https://www.admin.ch/opc/de/classified-compilation/20070597/201210010000/783.0.pdf>, 2015.

# Abbildungsverzeichnis

2.1. Mitglieder von KEP&Mail . . . . .	7
2.2. Umsatzanteil der Nationalen Pakete . . . . .	8
2.3. Europäischer Kurier-, Express- und Paketmarkt im Jahre 2011 . . . . .	8
2.4. Einstiegsmaske von Lobo . . . . .	11
2.5. Neuen Auftrag erstellen in Lobo . . . . .	12
2.6. Zeitmodell in Lobo . . . . .	12
2.7. Zeitmodell in Lobo . . . . .	13
3.1. Business Model Canvas von Imagine Cargo . . . . .	14
3.2. Aktueller Prozess für die Erstellung eines Auftrages . . . . .	15
3.3. Auftragsmaske von Deliveo . . . . .	16
3.4. Auftragsmaske für das Erstellen eines neuen Auftrages . . . . .	17
3.5. Auftragsmaske für das Erstellen eines neuen Auftrages . . . . .	18
4.1. Prozessablauf von User Centered Design . . . . .	19
6.1. Visualisierung einer Anfrage in einer konventionellen Webseite . . . . .	41
6.2. Visualisierung einer Anfrage in einer Single-page Applikation . . . . .	42
6.3. Darstellung des Programmierparadigma Flux . . . . .	44
6.4. Darstellung der compileTask Methode . . . . .	45
6.5. Darstellung des Prozesses zur Verifizierung einer Adresse . . . . .	46
6.6. Darstellung der Komponenten und ihrer Verwendung . . . . .	47
7.1. React Browser Erweiterung . . . . .	51
7.2. Redux Browser Erweiterung . . . . .	51
A.1. Wireframe Schritt 1 . . . . .	XI
A.2. Wireframe Schritt 2 . . . . .	XII
A.3. Wireframe Schritt 3 . . . . .	XIII
A.4. Wireframe Schritt 4 . . . . .	XIV
A.5. Wireframe Schritt 5 . . . . .	XV

# Tabellenverzeichnis

5.1. Persona Steckbrief für Mara Hürlimann . . . . .	24
5.2. Charaktermerkmal für Persona Mara Hürlimann . . . . .	24
5.3. Persona Steckbrief für Peter Elsener . . . . .	25
5.4. Charaktermerkmal für Persona Peter Elsener . . . . .	25
5.5. Persona Steckbrief für Laura Energie . . . . .	26
5.6. Charaktermerkmal für Persona Laura Energie . . . . .	26
5.25. Priorisierung der Persona Stories nach Kano . . . . .	36
A.1. Antwort Möglichkeiten beim Kano Modell . . . . .	III

# Listings

7.1. Lokaler Webserver . . . . .	52
7.2. Root Komponente . . . . .	53
7.3. connect im Einsatz . . . . .	54
7.4. Methode für LoBo Anfragen . . . . .	54
7.5. Aktion getConnections . . . . .	56
7.6. Reducer getConnections . . . . .	56
7.7. Render Methode der OrderStep2 Komponente . . . . .	56

# A. Anhang

## A.1. LoBo API

Im folgenden sind die möglichen Schnittstellen Endpunkte von LoBo beschrieben. Die Angaben stammen direkt aus der Beschreibung der Schnittstelle des Entwicklers und werden der Vollständigkeit halber aufgeführt.

**getGrantedActions** returns an array of actions covered by your license key

**getProductList** Lists all publicly available products

**getPaymentList** Lists all available payment methods

**getPriceScale** Get details of the price scale for the given product and list all graduations.

**createTask** Creates an internal (yet hidden) task. The returned tasktoken is a reference to access and change this task, eg. to add stops, set a new reference time, change the productid, ...

**setProductId** Change the product of the given task. Next, you might want to call action='calculateTask' to query the updated cost and transit times.

**setPaymentId** Change the payment for a given task (typically called just before action='orderTask')

**setCustomerNumber** Change the customer of the given task. Please consider that the resulting costs will change if the newly assigned customer is associated with another discount system. Next, you might want to call action='calculateTask' to query the resulting cost.

**setRefTime** Change the reference time of the given task. Next, you might want to call action='calculateTask' to query the updated transit times and/or if the the product is still available at the new time.

**addStop** Verifies an address and, on success (statuscode > 0): adds the address as new stop to the given task

**verifyAddress** Verifies an address. Returned placeid can be used

**addStopByPlaceId** Add a place as stop based on the search with action='verifyAddress' or action='autoCompletePlaceAndStreet'

**addStopByCustomerNumber** Add a customer as stop

**deleteStop** Deletes the given stop from the given task

**setStopSequence** Set a new sequence of the stops.

**getStopList** Returns the current stop list including address information

**calculateTask** Calculates the cost and transit time parameters.

**optimizeTask** Makes a copy of the existing task and minimizes the cost by reordering the stops (traveling sales man problem).

**setTaskInfo** Set additional plain text infos for given task. To reset a field, send empty value. All optional params (notepublic, noteinhouse, noteprivate, contactperson) that are not defined in a request remain unchanged.

**setStopInfo** Set additional plain text infos for given stop. To reset a field, send empty value. All optional params (notepublic, noteinhouse, noteprivate, contactperson, placename) that are not defined in a request remain unchanged.

**setStopTime** Set a fixed time for the given stop. The fixed time can be set either relative to the task's reference time (same date), or on a different date, if appointeddate parameter is given. Optionally you can set the a timecondition 'at', 'as from', or 'by' the fixed time. To reset the fixed time of a stop, simply set the appointedtime parameter to 0.

**orderTask** Place the order in the LoBo system. After this call no more changes can be made to task (via the API).

**autoCompleteStreet** Autocomplete street name based on a combination of a right wildcard and similarity search. If a street spans across an area of different zip codes and/or city names it will be found for every unique combination.

**getCustomer** Get list of customers (matching filter criteria).

**createCustomer** Create new customer.

**getZoneCompilation** List zone details of a given (zone based) product. Zones can be either be based on geopolygons or on postal codes.

**getStatistics** Get usage statistics.

## A.2. Kano Modell

Das Kano Modell welches nach seinem Erfinder Noriaki Kano, Professor an der Universität in Tokio, benannt ist, bestimmt die Notwendigkeit von Kundenwünschen und Erwartungen (Zitieren). Beim Kano Modell wird zwischen fünf Qualitätsebenen unterschieden.

**Basis-Merkmale** sind für den Benutzer so selbstverständlich, dass ihm/ihr ihre Notwendigkeit erst beim nicht vorhanden sein auffällt.

**Leistungs-Merkmal** werden vom Benutzer bewusst wahr genommen und sorgen für Zufriedenheit oder beseitigen Unzufriedenheit.

**Begeisterungs-Merkmale** überraschen den Benutzer und bringen ihm mehr Nutzen und Funktionalität.

**Unerhebliche Merkmale** sind dem Benutzer im Falle des Vorhandensein sowie auch Fehlens egal.

**Rückweisungs-Merkmale** machen den Benutzer unzufrieden, beim nicht vorhanden sein jedoch nicht zufrieden.

Diese Qualitätsebenen geben die Priorität bei der Entwicklung der Kundenwünsche vor. Basis-Merkmale haben die Priorität Hoch, Leistungs-Merkmale haben die Priorität Mittel und Begeisterungs-Merkmale bekommen die Priorität Niedrig. Die restlichen 2 Merkmale können bei der Priorisierung vernachlässigt werden da Sie zum einen nicht implementiert werden sollten und zum andern nur bei fehlen der anderen 3 Merkmale relevant werden.

Der Benutzer antwortet bezüglich eines Produktwunsches bzw. einer Anforderung auf eine positiv formuliert und eine negativ formulierte Frage:

- Funktional: Was würden Sie sagen, wenn die Applikation ..... macht.
- Dysfunktional: Was würden Sie sagen, wenn die Applikation ..... NICHT macht.

Dabei stehen ihm folgende Antworten zur Auswahl:

- Das würde mich sehr freuen
- Das setzte ich voraus
- Das ist mir egal
- Das nehme ich gerade noch hin
- Das würde mich sehr stören

Aus der Kombinationen der Antwort auf die Positive und der Antwort auf die Negative Frage, kann der Wunsch bzw. die Anforderung in eine der oben genannten 5 Qualitätsebenen eingeteilt werden. Es bestehen folgende Möglichkeiten:

Funktionale Antwort	Dysfunktionale Antwort	Merkmal	
Das setze ich voraus	+	Das würde mich stören	= Basis-Merkmal
Das würde mich sehr freuen	+	Das würde mich sehr stören	= Leistungs-Merkmal
Das würde mich sehr freuen	+	Das ist mir egal	= Begeisterungs-Merkmal
Das ist mir egal	+	Das ist mir egal	= Unerhebliches Merkmal
Das würde mich sehr stören	+	Das setze ich voraus	= Rückweisungs-Merkmal

Tabelle A.1.: Antwort Möglichkeiten beim Kano Modell

Antworten welche in der Tabelle A.2 nicht aufgelistet sind, sind unlogisch und werden für die Bewertung ignoriert.

## A.3. Interview Fragen

### A.3.1. Stakeholder Interview Fragen

#### Begriff Definitionen

**Service/Product** damit ist die zu entwickelnde Webapplikation gemeint

**Business** damit ist das Geschäft von Imagine Cargo gemeint

**Projekt** damit ist die Entwicklung des Prototypen gemeint

#### Generelle Fragen

- Was sollte der Service ihrer Meinung nach sein? (Können Sie den kompletten Service in eigenen Worten beschreiben?)
- Für wen soll der Service sein (Beispiele)?
  - Für andere Kuriere?
  - Für Endkunden? (In was für Branchen sind diese Kunden)
  - B2B/B2C?
  - Leads (potenzielle Kunden)?
- Wann soll die erste Version des Services verfügbar sein?
- Was muss die erste Version beinhalten? (MVP)
- Welche Sorgen bereitet ihnen der Service?
- Was ist das Schlimmste was passieren kann?
- Was soll der Service für das Business erreichen?
  - Gewinn
  - Einsparungen
  - Marke und Position im Markt beeinflussen
- Wie definieren Sie persönlich Erfolg für diesen Service?
- Wo sehen Sie sich im Ablauf dieses Projektes?
- Wie sieht der Service 2 bzw. 5 Jahre nach einer erfolgreichen Implementierung aus?

#### Spezifische Fragen

- Soll der Service dem Kunden Informationen anzeigen oder auch eine zu kaufende Dienstleistung anbieten?
- Welche Arbeitsschritte soll der Service ablösen?
- Welche Informationen soll der Kunden nach einem Einkauf erhalten? (Bestätigung, Tracking, Status)
- Welche Informationen wünschen Sie vom Kunden?
- Welche Informationen benötigen Sie minimal um erfolgreich einen Auftrag durchzuführen?

### Marketing/Sales Fragen

- Wer sind ihre Kunden und Benutzer heute und wie soll sich das in den nächsten 5 Jahren ändern?
- Wie passt der Service in die gesamte Servicestrategie?
- Welches sind die grössten Konkurrenten und was für Sorgen bereiten die ihnen?
- rscheidet sich das Produkt von der Konkurrenz?
- Welche 3/4 Qualitäten sollen Menschen mit dem Service und der Firma verbinden?
- Wieso benutzten Kunden diesen Service und nicht den eines Konkurrenten?
- Über was beschweren sich Kunden bzw. was wird am meisten verlangt? Und Wieso?

### Engineering Fragen

- Welche technologischen Entscheidungen wurden bereits getroffen? Wieso?
- Gibt es ein Diagramm welches das bereits existierende System beschreibt?
- Soll der Service nur eine Oberfläche für das Backend werden oder sollen auch andere Funktionen damit erledigt werden?

### Experten Fragen

- Was sind die typischen Demographien und Fähigkeiten der Benutzer und wie fest unterscheiden sich diese?
- Welche Unterschiede in den Benutzer Rollen und Aufgaben können erwartet werden?
- Welche Arbeitsabläufe und Praktiken können vor und nach der Benutzung erwartet werden?

## A.3.2. Stakeholder Interview Zusammenfassung

### Interview mit Nick Blake

**Nicolas:** Was soll der Service deiner Meinung nach sein? Bzw. Kannst du den kompletten Service in deinen eigenen Worten beschreiben?

**Blake:** 1. Der Service soll es den Kunden einfacher machen mit uns Business zu machen. 2. Der Service soll uns Arbeit abnehmen dieses Business für den Kunden zu erledigen. Das sind die Basics

**N:** Für wen soll der Service sein (Beispiele)? Für andere Kuriere, Für Endkunden (welche? Branchen?), B2C oder B2B?

**B:** Ich sehe primär 2 Arten von Kunden. Erfahrene und Unerfahrene. Erfahrene Kunden wissen viel über die Transportindustrie und halten nicht sonderlich viel vom Look & Feel eines Tools. Sie sind mit einer Green Screen Applikation zufrieden. Der unerfahrene Kunden braucht bedeutend mehr Hilfe. Sie müssen entweder mit einer Schritt-für-Schritt Anleitung durch den Prozess geleitet werden oder wir verstecken gewisse Optionen/Schritte welche für sie nicht zwingend notwendig sind. Der Service ist primär für die unerfahrene Kunden gedacht. Die erfahrene Kunden haben kein Problem mit LoBo direkt zu arbeiten. Die unerfahrene Kunden aus dem B2B sowie auch dem B2C Bereich arbeiten in einer Domäne welche mit dem Transportbusiness reichlich wenig zu tun hat aber deren Dienstleistungen benötigen. Sie können aus vielen verschiedenen Domänen kommen. Aus der Kreativebranche (Photografen, Werbeagenturen). Zusätzlich gibt es noch verschiedene Gruppen von Kunden. Solche die sich über Nachhaltigkeit Gedanken machen und solche die einfach nur etwas verschicken wollen.

**N:** Die Kunden die du jetzt erwähnt hast sind aber alle welche aus dem B2C Bereich. Gibt es auch welche die ein bedeutend grösseres Volumen regelmässig verschicken wollen?

**B:** Ich gebe dir Beispiele: Ein grünes Magazin aus Bayern. Welche einmal im Monat ihr Magazin verschickt. Ein grosser Retailer welche eine Nachhaltige Option beim verschicken anbieten will. Ein Betthersteller aus Berlin welcher seine Herstellungskette vervollständigen will und den Transport des fertigen Bettes durch Cargobikes ersetzen will.

**N:** Wie siehst du diese Kunden mit dem Service interagieren?

**B:** Bei den Transportfirmen sicher. Die wollen nicht einmal mit LoBo arbeiten. Die benötigen eine API. Die anderen B2B Kunden haben nicht ein so grosses Volumen und die können den Service benutztten. Für die Magazinverlag kann es die Möglichkeit geben dass Sie in ihrem Verkaufsprozess auf eine API von IC zugreift und dies so automatisiert wird.

**N:** Wann soll die erste Version des Services verfügbar sein?

**B:** As soon as possible.

**N:** Was muss die erste Version beinhalten (MVP)?

**B:** Momentan könnte es einfach eine Erweiterung des bereits existierenden wufoo Forms sein. Das nächste sollte ein Shopping Katalog sein welcher anhand der Angaben (Postcode, City) anzeigt was verfügbar ist. Ein weiter Schritt sollte das automatische Abfüllen in LoBo sein. Aber am schluss sollte der Service das wufoo Form ersetzen.

**N:** Die verbindung mit Lobo ist für dich also kein MVP Kriterium?

**B:** Korrekt das kann erst später kommen. Was wir momentan wollen ist ein Dialog mit den Kunden herstellen. Wenn der gesamte Prozess automatisiert ist, verlieren wir den Kontakt zum Kunden und damit auch ein wichtiges Marketingtool.

**N:** Verstehe Ich das richtig. Während des gesamten Bestellung/Einkaufprozesses soll es für beide Parteien (Kunde/IC) die Möglichkeit geben das Telefon in die Hand zu nehmen und den Prozess so weiter zu führen?

**B:** Genau. Nur so finden wir heraus was die Kunden wollen.

**N:** Welche Sorgen bereitet dir der Service?

**B:** Dass wir die Kontrolle über das zu verarbeitende Volumen verlieren. Sobald der Service online ist verlieren wird das Onboarding von Kunden. Wir müssen also sicher sein dass wir das Volumen verarbeiten können.

**N:** Was ist das schlimmste was passieren kann?

**B:** Dass wir nicht das Tun was wir sagen.

**N:** Was soll der Service für das Business erreichen? (Gewinn, Einsparungen, Marke und Position im Markt beeinflussen)

**B:** Er soll uns Helfen mehr Aufträge an Land zu ziehen. Wenn er seine Arbeit sehr gut macht, haben wir einen super Dienstleistung welche wir teurer verkaufen können. Er soll auch einen grossen Teil der Arbeit automatisieren.

**N:** Wie definierst du persönlich Erfolg für diesen Service?

**B:** Wenn unser Service eine maximal möglichen Wert hat und er besser ist als der der Konkurrenz.

**N:** Wo siehst du dich im Ablauf dieses Projektes?

**B:** Du solltest dich gut mit David absprechen weil er immer besser versteht wie es mit LoBo aussieht.

**N:** Wie sieht der Service in 2 bzw. 5 Jahren nach einer erfolgreichen Implementierung aus?

**B:** Der Service soll eine grösser Produktpalette anbieten. Der Kunde soll das Gefühl haben, dass alle Angebot von IC von einer Plattform aus erreichbar sind. Ein Ort für alle Fragen.

**N:** Wer sind eure Kunden und Benutzer und wie soll sich das in den nächsten 5 Jahren ändern?

**B:** Der Kunde soll uns sagen wie er sich verändern will/soll. Ich glaube die Basics ändern sich nicht und das Transportbusiness wird sich nicht all zu fest verändern. Im E-Commerce Bereich gibt es Anzeichen einer Veränderung. Der Kunde soll die Möglichkeit bekommen wer das Versenden übernehmen soll. Da sehe ich eine grosse Möglichkeit. Auch interessant ist der Fokus von Transportfirmen so genau zu sein wann Sie ankommen. Aber der Kunde immer weniger Interesse daran hat dies so genau zu wissen. Wenn ich mein Toiletttenpapier bestelle, dann muss ich nicht wissen ob das Morgen oder Übermorgen ankommt.

**N:** Wie passt der Service in die gesamte Servicestrategie?

**B:** Es ist ein elementares Tool.

**N:** Wer sind die grössten Konkurrenten und was für Sorgen breiten die?

**B:** Es gibt 2 Gruppen. Es gibt welche die dasselbe tun wie wir. In der Schweiz Swissconnects. In Deutschland Time:Matters. Dann gibt es solche die bedeutend grösser sind und mit Marketingmassnahmen den Kunden das Gefühl geben, sie machen dasselbe wie wir. Die Transportunternehmen sind teilweise sehr alt und schon lange am Markt. Da herrscht teilweise ein eisiger Wind. Wenn eine neue kleine Firma in den Markt kommt und was neues und besseres anbietet sind die ersten Reaktionen Defensive und man wird hart angegriffen.

**N:** Wie unterscheidet sich das Produkt von der Konkurrenz?

**B:** Vor meiner eigene Recherche hätte ich gesagt es sei der Soziale und Umweltansatz den wir verfolgen. Aber jetzt kommt der Aspekt der Transparenz eine bedeutend grössere Rolle. Zu berechnen wie viel CO2 bei einem Transport verbraucht wird ist immer noch ein interessantes Tool, aber es ist bestimmt nicht mehr unser USP.

**N:** Welche 3/4 Qualitäten sollen Menschen mit dem Service und der Firma verbinden?

**B:** Easy to do Business with, Liable, Personal, Sustainable.

**N:** Wieso benutzten Kunden diesen Service und nicht den eines Konkurrenten?

**B:** Weil der Kunden zufrieden ist mit dem was wir tun.

**N:** Welche technologischen Entscheidungen wurden bereits getroffen? Wieso?

**B:** Wir benötigen Cloud Bases Solutions weil wir keinen fixen Standort haben für irgendwelche Infrastruktur. Wir müssen uns vor Augen halten, dass die Boten eine alles selber flicken können Mentalität haben. Und dementsprechend skeptisch gegenüber Softwarelösungen sind. LoBo ist für uns momentan das beste was es zur Zeit gibt. Aber wenn etwas besseres aufkommt müssen wir dies natürlich analysieren.

**N:** Was sind die typischen Demographien und Fähigkeiten der Benutzer und wie fest unterscheiden sich diese?

**B:** Sehr grosse Gruppe. Von Leuten mit Ahnung und solche ohne. Es gibt eine Generation Unterschied.

### Interview mit David Emmerth

**Nicolas:** Was soll der Service deiner Meinung nach sein? Bzw. Kannst du den kompletten Service in deinen eigenen Worten beschreiben?

**David:** Ein möglichst einfaches und simples Tool welches dem Kunden erlaubt einen Lieferauftrag für ein Paket mit gewissen Dimensionen (Grösse und Gewicht) von A nach B zu erstellen. Dabei soll er sehen welche Routen zu welchen Zeit existieren. Der Kunde sieht die verschiedenen Produkte welche IC anbietet. Bei Paketen welche über eine Landesgrenze verschickt wird, muss der Kunde über die zusätzlichen Schritte informiert werden. Die Bezahlung der Dienstleistungen soll auch über den Service abgewickelt werden.

**N:** Für wen soll der Service sein (Beispiele)? Für andere Kuriere, Für Endkunden (welche? Branchen?), B2C oder B2B?

**D:** Primär B2B aber der Service ist offen für alle Kunden. Der Unterschied wird sein zwischen ad-hoc Kunden und Kunden welche regulär Aufträge habe. Für letztere kann man auch andere Prozesse anbieten. z.B. Daueraufträge. Service ist in erster Linien für Kunden welche spontan einen Auftrag haben.

**N:** Aus welchen Branchen kommen diese spontanen Kunden?

**D:** Medizinaltechnik, Kreativindustrie (Druck sachen), Öffentliche Verwaltungen, Rechtsdokumente (Notariat Anwaltsbüro) sind die Primäre Zielgruppen. Dies können wiederkehrende Kunden sein aber ihre Aufträge legen keine Regelmässigkeit an den Tag.

**N:** Wann soll die erste Version des Services verfügbar sein?

**D:** So schnell wie möglich.

**N:** Hinsichtlich der restlichen Prozesse von IC?

**D:** Es ist nicht das absolut dringlichste. POC können wir auch mit dem bestehenden Tool/Prozess machen. Wir brauchen in erstrt Linie Geld um dann die Frontend Geschichten wie auch andere Anschaffungen zu tätigen z.B. LoBo weiter zu entwickeln. Es ist eine Eigenentwicklung welche den Investoren mehr vertrauen gibt. Aber um zu beweisen dass eine Nachfrage für die Dienstleistung besteht, ist es nicht essentiell.

**N:** Was muss die erste Version beinhalten (MVP)?

**D:** Absenderadresse, Lieferadresse, Paketbeschreibung eingeben

**N:** Ist es Intergriert mit Lobo oder nur ein Programm welches ein Dokument ausspuckt. Besteht der Mehrwert in der Integration mit LoBo?

**D:** LoBo braucht eine Absenderadresse, Lieferadresse. Die Packetbeschreibung wird für Time:Matters benötigt. Für die Velokuriere brauchen wir noch Beschreibungen wo das Paket abgeholt bzw. geliefert werden soll (1. Stock bei Herr Müller). Dies soll automatisch einen LoBo Auftrag erzeugen. Weil zur Zeit ist dies ein manueller Arbeitsschrit (Infos aus Mail in LoBo einfügen). Die Bezahlung mit Kreditkarte gehört auch dazu. Aus Kundensicht soll der Einkauf abgeschlossen sein. Aus IC sicht sollen alle Relevanten Informationen in LoBo sein. Lobo braucht für eine volle Automatisierung noch einige Features. Aufträge aufteilen und an Velokuriere in Absende und Zielstadt aufteilen. Diese momentanen Workarounds sollen nicht durch den Service gelöst werden. Aber der Service kann diese Workarounds auch implementieren bis LoBo die Features anbietet. LoBo funktioniert in Stops und wenn der Service in der Lage ist anstatt nur Start und Ziel als Stops einzutragen sondern auch die Bahnhöfe in der Start und Zielstadt einzutragen, besteht schon ein Mehrwert.

**N:** Welche Sorgen bereitet dir der Service?

**D:** Dass das UX nicht gut genug ist. Zu kompliziert oder Mühsam ist. Kunden sollen immer noch in der Lage sein das Telefon in die Hand zu nehmen und das ganze so zu erledigen.

**N:** Was ist das schlimmste was passieren kann?

**D:** Der Kunde gibt was ein und wir (IC) bekommen das nicht mit und nichts passiert. Was auch schlimm ist wenn der Kunde nicht erfährt wieso eine Dienstleistung zu einem gewissen Zeitpunkt oder an einem gewissen Ort nicht verfügbar ist.

**N:** Was soll der Service für das Business erreichen?

**D:** Er soll das Benutzererlebnis der Kunden verbessern. Besonders für Kunden welche nicht über eine längere Salesbeziehung aufgebaut werden. Sobald etwas nach einem fertigen technischen Produkt aussieht, haben die Investoren mehr Freude.

**N:** Wie definierst du persönlich Erfolg für diesen Service?

**D:** Einfach sauber und ästhetisch sinnvoll und integriert mit LoBo. Kunden können auch Velokurier sein. Diese Kunden haben ein anderes Anforderungsprofil und sollten diese über den Service (anstatt über LoBo Login) arbeiten, müssen die Prozesse angepasst werden. Wenn ein Kurier aus Frankfurt sich anmeldet ist der Bahnhof in Frankfurt bereits als Zieladresse eingetragen. Velokurier sollen die Möglichkeit haben selber zu entscheiden wann Sie ein Paket auf den Zug bringen. Wenn es ihr eigener Kunde ist können sie das selber entscheiden. IC braucht minimum 20 Minuten um ein Ticket bei Time Matters zu bestellen. Aber wenn Sie der Meinung sind sie schaffen ein Strecke in einer bestimmten Zeit dürfen sie bestimmen welcher Zug. Time Matters hat keine API. Aber Francesco hat ein Tool entwickelt welches mit einer sehr hohen Genauigkeit anzeigt welche Züge von einem bestimmten Bahnhof fahren und Pakete mitnehmen. Diese Tool wird höchstwahrscheinlich auch für den Service benötigt weil dieser dem Kunde mitteilen muss ab wann eine Lieferung nicht mehr möglich ist.

**N:** Wo siehst du dich im Ablauf dieses Projektes?

**D:** Soviel wie es mich braucht. So wenig wie möglich. Ich brauche dich nicht zu kontrollieren. Ich glaube das ganze steht und fällt mit dem Verständnis von LoBo. LoBo soll aber auch hinterfragt werden

**N:** Wie sieht der Service in 2 bzw. 5 Jahren nach einer erfolgreichen Implementierung aus?

**D:** IC will eine grössere Produktpalette anbieten aber die ist nicht Primär für Endkunden gedacht. In dieser Hinsicht wird sich der Service nicht gross verändern. Wahrscheinlicher ist eine grössere Verbreitung der Dienstleistung z. B. Apps für Handy anbieten. White Label Lösungen sind auch sehr interessant.

**N:** Wer sind eure Kunden und Benutzer und wie soll sich das in den nächsten 5 Jahren ändern?

**D:** Grösser Kunden (Key Accounts) welche mehr Volumen versenden und Rabatte bekommen. Durch die Erweiterung der Produktpalette werden neue Benutzer/Kunden gewonnen aber diese werden wahrscheinlich nicht primär über den Service arbeiten.

**N:** Wie passt der Service in die gesamte Servicestrategie?

**D:** Same Day Deliveries (Der Service bietet dies in erster Linie an) wird nie das primär Zugpferd sein von IC. Aber damit kann eine gute Benutzerbasis geschaffen werden.

**N:** Wer sind die grössten Konkurrenten und was für Sorgen breiten die?

**D:** Bei den Same Day gibt es sehr viele kleinere Konkurrenten. Die grösseren machen fast keine Same Day.

**N:** Wieso benutzten Kunden diesen Service und nicht den eines Konkurrenten?

**D:** Kein Kunde googelt nachhaltiger express Kurier. Aber es gibt viele Kunden die was verschicken müssen und dabei einen transparenten Prozess schätzen.

**N:** Welche technologischen Entscheidungen wurden bereits getroffen? Wieso?

**D:** Lobo ist eine Software über die wir vieles schlechtes sagen aber LoBo kann auch sehr vieles gut. LoBo wurde für Velokuriere entwickelt in einer Stadt und wir missbrauchen es jetzt für unser Business. Jürgen ist nicht immer gleicher Meinung bezüglich Weiterentwicklung. Es gibt noch Bamboo. Aber Lobo Instanzen sollen miteinander reden können.

**N:** Soll der Service dem Kunden Informationen anzeigen oder auch eine zu kaufende Dienstleistung anbieten?

**D:** In 90% der Fälle soll der Kunde nicht per Mail oder Telefon mit uns kommunizieren um seine Dienstleistung zu kaufen. Aber es muss die Möglichkeit geben uns (IC) per Telefon zu erreichen. Tracking soll auch dem Kunde angeboten werden.

**N:** Welche Informationen soll der Kunden nach einem Einkauf erhalten? (Bestätigung, Tracking, Status)

**D:** Lobo kann noch nicht sehr viel. Aber Abgeholt, Abgeliefert und Abgeschlossen wird von Lobo zurück gegeben. Kurier kann seine erledigten Tasks in LoBo eintragen. Möglicherweise kann dies über die API auch ausgelesen werden. Time Matter hat keine API und sie geben auch keine Garantie dass ein Paket auf einem gewissen Zug landet.

**N:** Welche Informationen wünschen Sie vom Kunden?

**D:** Zollinformationen, Comercial Invoice und Angaben für Versendungen.

## A.4. Wireframes

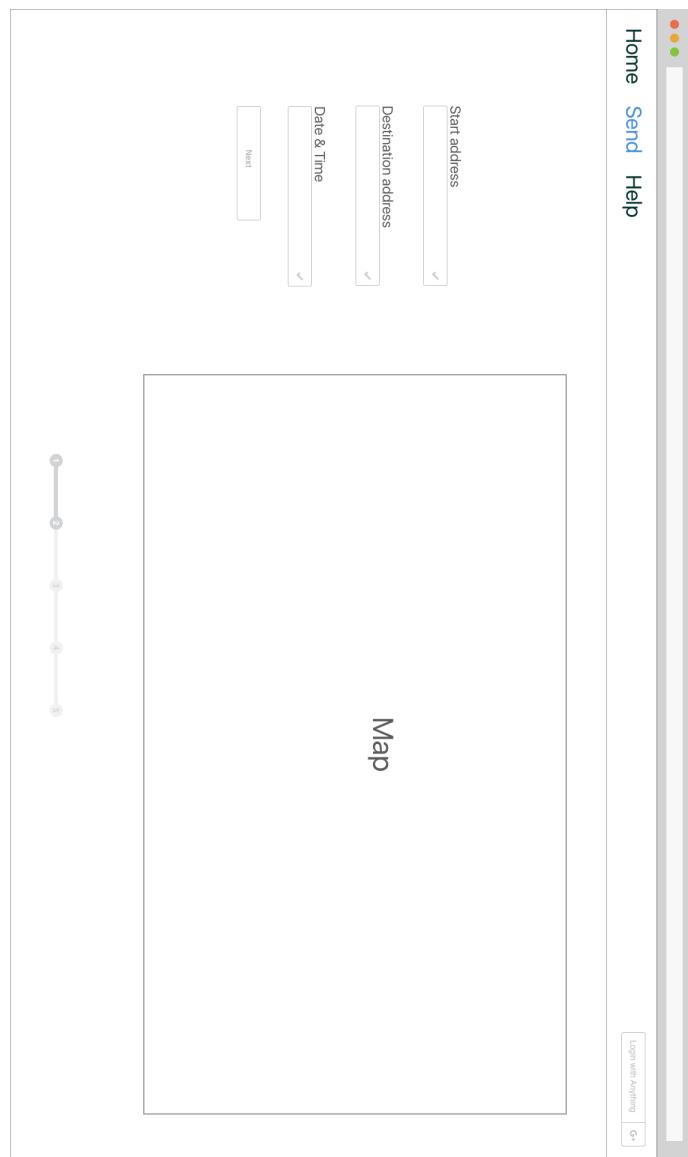


Abbildung A.1.: Wireframe Schritt 1

Start address, pickup time

End address, delivery time

Pickup Name

Pickup Number

Packet Dimensions

Delivery Name

Delivery Number

Delivery Email

Pickup Comment

Delivery Comment

Next

Home **Send** Help

Login with Amazon

Abbildung A.2.: Wireframe Schritt 2

The wireframe illustrates a user interface for a delivery tracking application. At the top, a navigation bar features a logo with three colored dots (red, orange, and green) and includes links for Home, Send, and Help. Below the navigation bar is a large input field for entering a tracking number. To the right of this field is a vertical sidebar containing a 'Login with Amazon' button and a 'Go!' button. The main content area is divided into two sections: 'Start address, pickup time' on the left and 'End address, delivery time' on the right. Between these sections are input fields for Content, Value, and Amount, each with a small downward arrow icon. At the bottom of the main content area is a 'Description' input field. On the far left, a vertical sidebar displays a series of five circular icons connected by a line, with the top icon being red and the others being light blue. A 'Next' button is positioned to the right of this sidebar. The footer of the page contains the text 'Browser 1920 x 1080'.

Abbildung A.3.: Wireframe Schritt 3

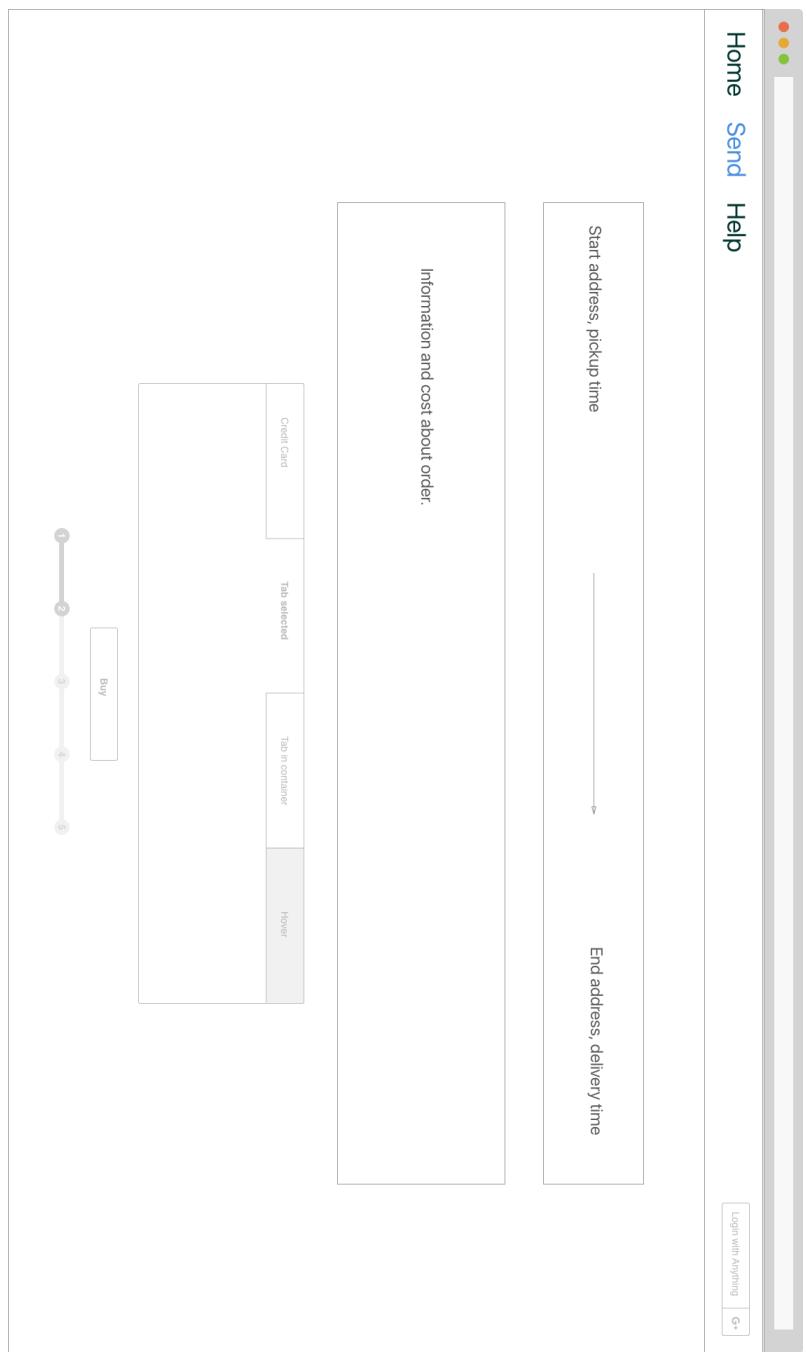


Abbildung A.4.: Wireframe Schritt 4

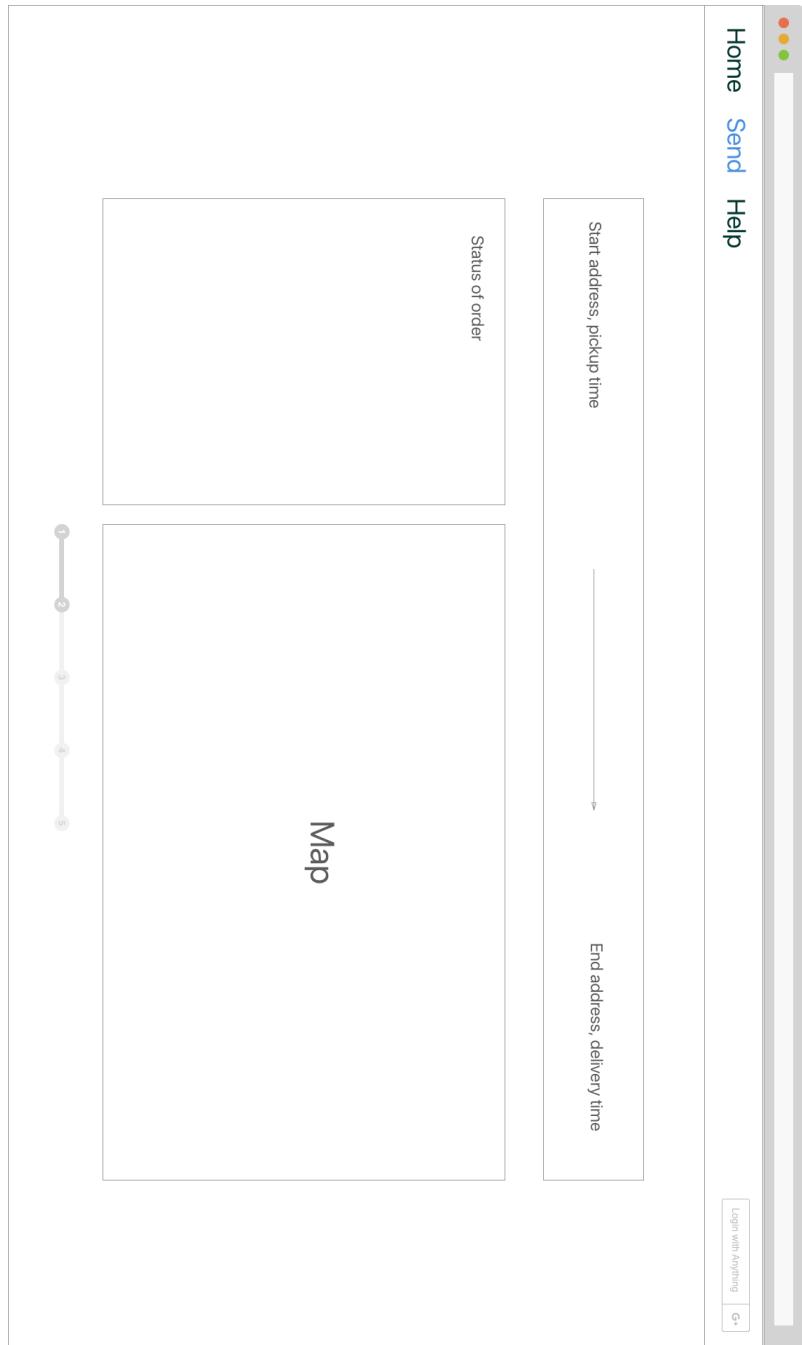


Abbildung A.5.: Wireframe Schritt 5