

HOCHSCHULE LUZERN

BACHELORARBEIT

JTI Pickup Station

Oliver Werlen

Betreut durch Rene Meier
xz

20. Mai 2021

Page

Bachelorarbeit an der Hochschule Luzern – Informatik

Titel: JTI Pickup Station

Studentin/Student: Oliver Werlen

Studentin/Student:

Studiengang: BSc Informatik

Jahr: 2021

Betreuungsperson:

Expertin/Experte:

Auftraggeberin/Auftraggeber:

Codierung / Klassifizierung der Arbeit:

- A: Einsicht (Normalfall)
- B: Rücksprache (Dauer: Jahr / Jahre)
- C: Sperre (Dauer: Jahr / Jahre)

Eidesstattliche Erklärung Ich erkläre hiermit, dass ich/wir die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt haben, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben haben, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht haben, das Vertraulichkeitsinteresse des Auftraggebers wahren und die Urheberrechtsbestimmungen der Fachhochschule Zentralschweiz (siehe Merkblatt «Studentische Arbeiten» auf MyCampus) respektieren werden.

Ort / Datum, Unterschrift _____

Ort / Datum, Unterschrift _____

Abgabe der Arbeit auf der Portfolio Datenbank:

Bestätigungsvisum Studentin/Student

Ich bestätige, dass ich die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank abgelegt habe. Die Verantwortlichkeit sowie die Berechtigungen habe ich abgegeben, so dass ich keine Änderungen mehr vornehmen kann oder weitere Dateien hochladen kann.

Ort / Datum, Unterschrift _____

Ort / Datum, Unterschrift _____

Verdankung

xxx

**Ausschliesslich bei Abgabe in gedruckter Form:
Eingangsvisum durch das Sekretariat auszufüllen**

Rotkreuz, den _____ Visum: _____

Hinweis: Die Bachelorarbeit wurde von keinem Dozierenden nachbearbeitet. Veröffentlichungen (auch auszugsweise) sind ohne das Einverständnis der Studiengangleitung der Hochschule Luzern – Informatik nicht erlaubt.

Copyright © 2019 Hochschule Luzern – Informatik

Alle Rechte vorbehalten. Kein Teil dieser Arbeit darf ohne die schriftliche Genehmigung der Studiengangleitung der Hochschule Luzern – Informatik in irgendeiner Form reproduziert oder in eine von Maschinen verwendete Sprache übertragen werden.

Zusammenfassung

Inhaltsverzeichnis

1 Problem und Vision	5
1.1 Problem	5
1.1.1 Versandkosten/Mindestbestellwert	5
1.1.2 Dauer bis Ware bei Endkonsumenten	5
1.1.3 Angebot nur in begrenztem Zeitraum möglich	5
1.2 Vision	5
2 Stand der Technik	6
2.1 MyPost 24	6
2.1.1 Allgemein	6
2.2 avec	7
2.2.1 avec now	7
2.2.2 avec Box	7
2.2.3 Ablauf Tabakkauf	8
2.3 Starbucks Progressive Web App (PWA)	9
2.3.1 Standortsuche	9
2.3.2 Fazit	10
2.4 Fazit	10
3 Ideen und Konzepte	11
3.1 Systemarchitektur	11
3.2 Frameworks	11
3.2.1 Spring Boot	11
3.2.2 Angular	12
3.3 Weitere Technologien	12
3.3.1 Docker	12
3.3.2 MariaDB	12
3.3.3 Hibernate ORM	12
3.3.4 GitLab	12
3.3.5 Bezahlungsdienst	12
3.3.6 Alterverifikation	13
3.3.7 Karte	13
3.4 Konzepte	13
3.5 Data Transfer Object	13
3.5.1 Hypertext as the engine of application state (HATEOAS)	13
3.5.2 Spring Doc	14
4 Methoden	16
4.1 SoDa	16
4.2 CI/CD	16
4.2.1 build	16
4.2.2 package	17
4.2.3 deploy	17
4.2.4 Sequenzdiagramm	17
4.3 Testing	17
5 Realisierung	18
5.1 Initialisierungsphase	18
5.1.1 Kick-Off Meeting	18
5.1.2 Erstellen des Projektmanagementplans	18
5.1.3 Problem und Vision	18
5.1.4 Requirement Engineering	18
5.1.5 Stand der Technik	18

5.1.6	Meilenstein Abschluss Initialisierungsphase	18
5.2	Konzeptionsphase	19
5.2.1	Sprint 1	19
5.2.2	Sprintreview Sprint 1	20
5.2.3	Sprint 2	21
5.2.4	Sprintreview Sprint 2	25
5.2.5	Sprint 3	25
5.2.6	Sprint 4	28
5.2.7	Sprint 5	30
5.2.8	Sprintreview Sprint 5	34
5.2.9	Sprint 6	34
5.2.10	Meilenstein Abschluss Bestellprozess und Zwischenpräsentation	42
5.3	Realisierungsphase	43
5.3.1	Sprint 7	43
5.3.2	Auswahl der PickUp Station	45
5.3.3	Sprintreview Sprint 7	45
5.3.4	Sprint 8	45
5.3.5	Sprintreview Sprint 8	50
5.3.6	Sprint 9	50
6	Evaluation und Validation	56
7	Ausblick	57
8	Verzeichnisse	58
A	Testprotokolle	66
A.1	Testprotokolle Bestellung	66
B	Projektmanagementplan	75
B.1	Projektorganisation	75
B.1.1	Organisationsplan, Rollen, Zuständigkeiten	75
B.1.2	Projektstrukturplan	76
B.2	Projektführung	77
B.2.1	Rahmenplan	77
B.2.2	Meilensteine	78
B.2.3	Risikomanagement	79
B.2.4	Definition of done	81
B.3	Projektunterstützung	82
B.3.1	Tools für Entwicklung, Test und Abnahme	82
B.3.2	Konfigurationsmanagement	82
B.4	Teststrategie und Drehbuch	83
B.4.1	Teststrategie	83
B.4.2	Testdrehbuch	83
B.5	Bemerkungen	83
C	System-Spezifikation	84
C.1	Systemübersicht	84
C.1.1	Systemarchitektur	84
C.1.2	Kontextdiagramm	85
C.2	Architektur und Designentscheide	86
C.2.1	Modelle und Sichten	86
C.2.2	Daten (Mengengerüst und Strukturen)	86
C.2.3	Entwurfsentscheide	86
C.3	Schnittstellen	89

C.3.1	Externe Schnittstellen	89
C.3.2	Wichtige interne Schnittstellen	89
C.3.3	Benutzerschnittstellen	89
C.4	Environment-Anforderungen	89
C.4.1	Hardware	89
C.4.2	Software	89
D	Software Requirements Specification	90
D.1	Zweck	90
D.1.1	Zielgruppe	90
D.1.2	Produktumfang	90
D.1.3	Definitionen	90
D.1.4	Systemübersicht	90
D.1.5	Abhängigkeiten	90
D.2	Spezifische Anforderungen	91
D.2.1	Funktionale Anforderungen	91
D.2.2	Nicht funktionale Anforderungen	92
D.3	Änderungshistorie	93
D.4	Bemerkungen	93
D.5	Unterschriften	93
E	Sitzungsprotokolle	94
E.1	23.02.2021	94
E.1.1	Ordnungsaufruf	94
E.1.2	Teilnehmer	94
E.1.3	Genehmigung des Protokolls	94
E.1.4	Ankündigungen	94
E.1.5	besprochene Punkte	94
E.1.6	Tagesordnung der nächsten Sitzung	95
E.1.7	Unterschriften	95
E.2	04.03.2021	96
E.2.1	Ordnungsaufruf	96
E.2.2	Teilnehmer	96
E.2.3	Genehmigung des Protokolls	96
E.2.4	Ankündigungen	96
E.2.5	besprochene Punkte	96
E.2.6	Tagesordnung der nächsten Sitzung	97
E.2.7	Unterschriften	97
E.3	11.03.2021	97
E.3.1	Ordnungsaufruf	97
E.3.2	Teilnehmer	97
E.3.3	Genehmigung des Protokolls	97
E.3.4	Ankündigungen	97
E.3.5	besprochene Punkte	98
E.3.6	Tagesordnung der nächsten Sitzung	98
E.3.7	Unterschriften	98
E.4	18.03.2021	98
E.4.1	Ordnungsaufruf	98
E.4.2	Teilnehmer	98
E.4.3	Genehmigung des Protokolls	98
E.4.4	Ankündigungen	98
E.4.5	besprochene Punkte	99
E.4.6	Tagesordnung der nächsten Sitzung	99
E.4.7	Unterschriften	99
E.5	25.03.2021	99

E.5.1	Ordnungsaufruf	99
E.5.2	Teilnehmer	99
E.5.3	Genehmigung des Protokolls	99
E.5.4	Ankündigungen	100
E.5.5	besprochene Punkte	100
E.5.6	Tagesordnung der nächsten Sitzung	100
E.5.7	Unterschriften	100
E.6	01.04.2021	100
E.6.1	Ordnungsaufruf	100
E.6.2	Teilnehmer	101
E.6.3	Genehmigung des Protokolls	101
E.6.4	Ankündigungen	101
E.6.5	besprochene Punkte	101
E.6.6	Tagesordnung der nächsten Sitzung	101
E.6.7	Unterschriften	101
E.7	15.04.2021	101
E.7.1	Ordnungsaufruf	101
E.7.2	Teilnehmer	102
E.7.3	Genehmigung des Protokolls	102
E.7.4	Ankündigungen	102
E.7.5	besprochene Punkte	102
E.7.6	Tagesordnung der nächsten Sitzung	102
E.7.7	Unterschriften	102
E.8	15.04.2021	102
E.8.1	Ordnungsaufruf	102
E.8.2	Teilnehmer	103
E.8.3	Genehmigung des Protokolls	103
E.8.4	Ankündigungen	103
E.8.5	besprochene Punkte	103
E.8.6	Tagesordnung der nächsten Sitzung	103
E.8.7	Unterschriften	103
E.9	12.05.2021	103
E.9.1	Ordnungsaufruf	103
E.9.2	Teilnehmer	104
E.9.3	Genehmigung des Protokolls	104
E.9.4	Ankündigungen	104
E.9.5	besprochene Punkte	104
E.9.6	Tagesordnung der nächsten Sitzung	104
E.9.7	Unterschriften	104
F	Rahmenpläne	105
G	Originale Aufgabenstellung	106
H	Zwischenpräsentation	106

1 Problem und Vision

1.1 Problem

Der Onlinekauf ist beim Zigarettenkauf ein sehr selten genutzter Absatzweg. Dabei sind vor allem die nachfolgenden Punkte verantwortlich für die seltene Nutzung dieses Angebots.

- Versandkosten/Mindestbestellwert
- Dauer bis Ware beim Endkonsumenten
- Angebot nur in begrenztem Zeitraum möglich

1.1.1 Versandkosten/Mindestbestellwert

Bei diversen Onlineshops kommen bei zu geringer Bestellmenge erhebliche Versandkosten hinzu. So kostet der Versand per Paket in der Regel 9 Franken. Bei Kioskolino ist der Versand ab einem Bestellwert von 139.- Fr. Portofrei. [Genf, o.D.] Bei Coop ist die Liefergebühr sogar noch höher. Sie beträgt 17.90 Fr. Die Versandkosten nehmen mit zunehmendem Bestellwert ab. Ab 500.- Fr. ist der Versand kostenlos. [Coop, 2020]

1.1.2 Dauer bis Ware bei Endkonsumenten

Bei der Bestellung bei Kioskolino ist die Ware innerhalb von 1-3 Werktagen beim Konsumenten. [Genf, o.D.] Für die meisten Kunden dauert dies zu lange. Coop verspricht die Lieferung am selben Tag. Dazu können bei der Bestellung verschiedene Zeifenster ausgewählt werden, die Verfügbarkeit ist dabei von der Region abhängig. Die Ware muss aber frühzeitig bestellt werden, um die Lieferung am gleichen Tag garantieren zu können. Zudem bietet Coop auch die Möglichkeit, die Produkte direkt in der Filiale abzuholen. [Coop, 2020]

Das Problem ist aber mit beiden Anbieter identisch. Es muss die Ware sehr früh bestellt werden. Zudem dauert die Lieferung immer zwischen 4 Stunden bis zu 3 Tagen. Eine Lieferung am Sonntag ist dabei nicht möglich, Samstags wird nur am Nachmittag geliefert.

1.1.3 Angebot nur in begrenztem Zeitraum möglich

Die bestellte Ware wird nur zu bestimmten Zeiten ausgeliefert. So ist eine Lieferung an Sonn- und Feiertagen nicht möglich.

1.2 Vision

Durch die JTI Pick-Up Station ist es dem Kunden möglich, seine Ware bequem im Onlineshop zu bestellen und anschliessend direkt und ohne Wartezeit an der gewünschten Pick-Up Station abzuholen.

Die Artikel werden durch den Kunden an der gewählten Pick-Up Station bereitgestellt. Durch das Vorzeigen der Bestellbestätigung durch den Kunden wird der Artikel freigegeben und steht zur Abholung bereit.

Ein Mindestbestellwert muss nicht erreicht werden. Zudem werden keine zusätzlichen Gebühren verlangt.

Die Applikation soll dabei durch eine einfache und intuitive Bedienung eine optimale Benutzerexperience bieten. Die Bestellung soll schnell und einfach ablaufen. Die Abholung soll anschliessend in kurzer Zeit abgewickelt werden. Durch die Umsetzung als PWA ist die Applikation auch ohne aktive Internetverbindung nutzbar.

Durch das Verwenden eines bereits etablierten Altersverifikationsanbieters können zudem die rechtlichen Bedingungen erfüllt werden. Der Bezahlvorgang wird ebenfalls durch einen etablierten Anbieter durchgeführt. Dies garantiert eine sichere und zuverlässige Bezahlabwicklung.

2 Stand der Technik

2.1 MyPost 24

2.1.1 Allgemein

Mit der Pick Post und My Post 24 können Briefe und Pakete an die genannte Pick-Up Station gesendet werden. Es besteht auch das Angebot, Pakete von einer Pick-Up Station zu versenden. Die Auswahl einer Pick-Up Station geschieht dabei mit der Angabe der entsprechenden Station. Der Dienst lässt sich somit in jedem Onlineshop nutzen.

Die Abholung der Artikel muss innerhalb von 10 Tagen geschehen. Sobald die Artikel zur Abholung bereit sind, erhält der Kunde wahlweise eine Bestätigungs-mail oder ein SMS. Mit dem darin enthaltenen Abholcode lässt sich die Ware abholen. [Post-CH-AG, 2015]

Im Grossraum Luzern befinden sich acht My Post 24-Abholstellen. [**myPost24Stations**] Das Design der Abholstelle sieht dabei konventionellen Briefkästen der Post sehr ähnlich.



Abbildung 1: My Post 24-Abholstelle,
Quelle: AG, 2021

Die Lösung der Post behebt dabei aber nicht die in Kapitel 1 beschriebenen Probleme. Das Angebot richtet sich hauptsächlich an Personen, welche bei der Lieferung der Post nicht zuhause sind. Durch das Angebot kann so ein Abholen an der Poststelle vermieden werden. Die Lieferzeit sowie die Lieferkosten bleiben dabei aber vorhanden. Der Bestell- und Bezahlvorgang wird beim jeweiligen Onlineshop durchgeführt.

2.2 avec

2.2.1 avec now

"— Dein Online Lieferservice von avec — Mit avec now haben wir rund um das beliebte Angebot unseres Convenience-Formats avec einen Online-Store lanciert. Zur Auswahl steht ein breites Convenience-Sortiment. Die bestellten Waren werden direkt in unseren Stores zusammengestellt und so schnell wie möglich ausgeliefert." [Valora, 2021e]

Auf der Website von avec now wird dabei eine Lieferzeit von 60 Minuten angepriesen. Der Mindestbestellwert beträgt dabei 20.- Fr. [Valora, 2021a] Das Angebot gilt dabei für einen grossen Teil des Sortiments, darunter auch Tabakwaren. Das Angebot befindet sich dabei noch in der Pilotphase und wird nur im Raum Zürich angeboten. [Valora, 2021f]

2.2.2 avec Box

Der Anbieter geht bei diesem Angebot einen neuen Weg. Das gesamte Einkaufen wird mittels App durchgeführt. Mittels dieser können Produkte dem Warenkorb hinzugefügt und so bezogen werden. In einer ersten Phase sind zu Stosszeiten Mitarbeiter präsent, um die Kunden beim Einkauf zu unterstützen.

Um Tabakprodukte zu beziehen, befindet sich in der App eine integrierte Altersverifikation. Die Tabakwaren werden dabei im Store via Touchscreen ausgewählt. Durch die App wird eine Altersverifikation durchgeführt. [Valora, 2021b]

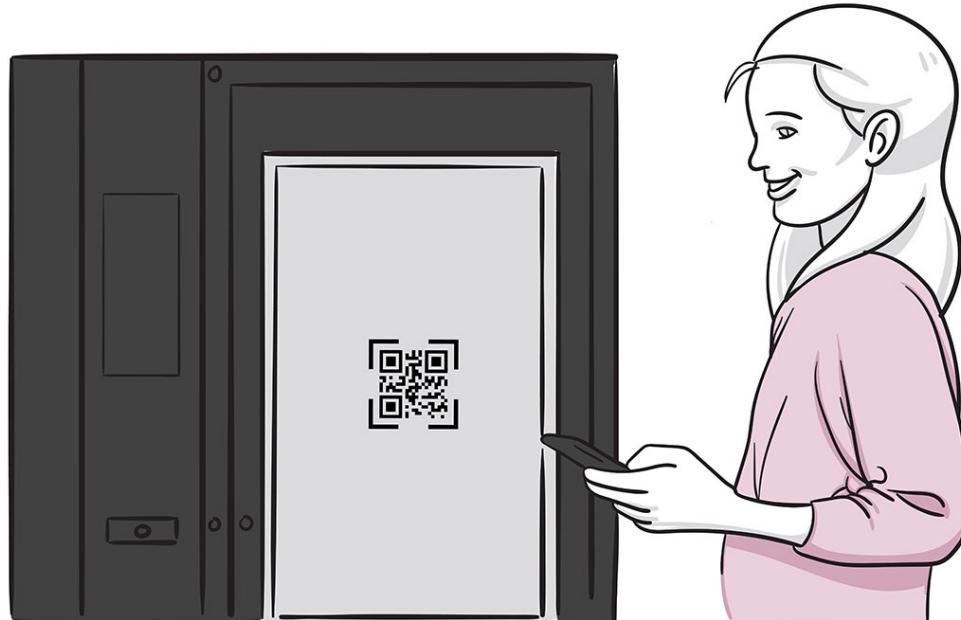


Abbildung 2: Tabakkauf avec box,
Quelle: Valora, 2021b

Nach momentanem Stand steht die avec box am Campus der ETH Zürich. Die Box ist dabei von Montag-Sonntag jeweils von 6:00 - 22:00 in Betrieb. Es ist ein Rollout in weitere Regionen der Schweiz vorgesehen. [Valora, 2021d]

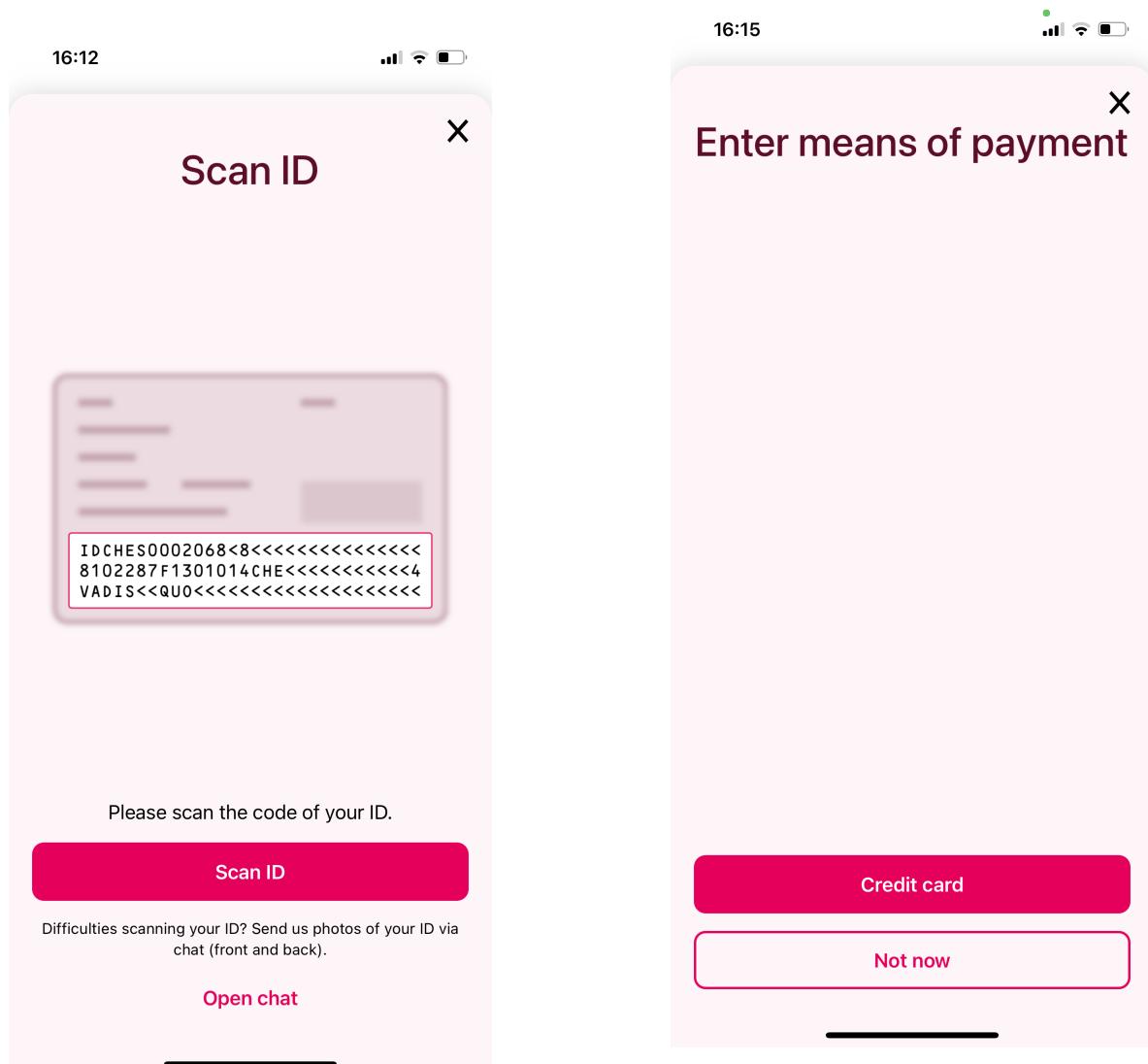
Das Angebot von avec löst einige der genannten Probleme. So kann die Ware direkt bezogen werden.

Es fallen keine Versandkosten an. Zur Zeit ist die avec box nur an einem Standort verfügbar, was die Verfügbarkeit erheblich einschränkt. Zudem ist sie nur zwischen 6:00-22:00 im Betrieb.

2.2.3 Ablauf Tabakkauf

Die avec box ist sehr ähnlich zur JTI Pick-Up Station. Aus diesem Grund wird der Registrierungsvorgang nachfolgend genauer betrachtet.

Registrierung Das Anlegen eines avec-Kontos verläuft dabei analog zur Erstellung von anderen Accounts. Mittels Handynummer und Passwort wird der Account angelegt. Zur Verifikation wird ein Bestätigungscode an die Nummer gesendet. Dieser muss eingegeben werden. Anschliessend wird die Alterverifikation durchgeführt. Hierzu muss die Identitätskarte mit der Kamera eingelesen werden. In einem nächsten Schritt kann die Kreditkarte hinterlegt werden. Der Bezahldienst wird dabei von Datatrans bereitgestellt. Anschliessend ist die Registrierung abgeschlossen und der Einkauf in der avec box könnte beginnen.



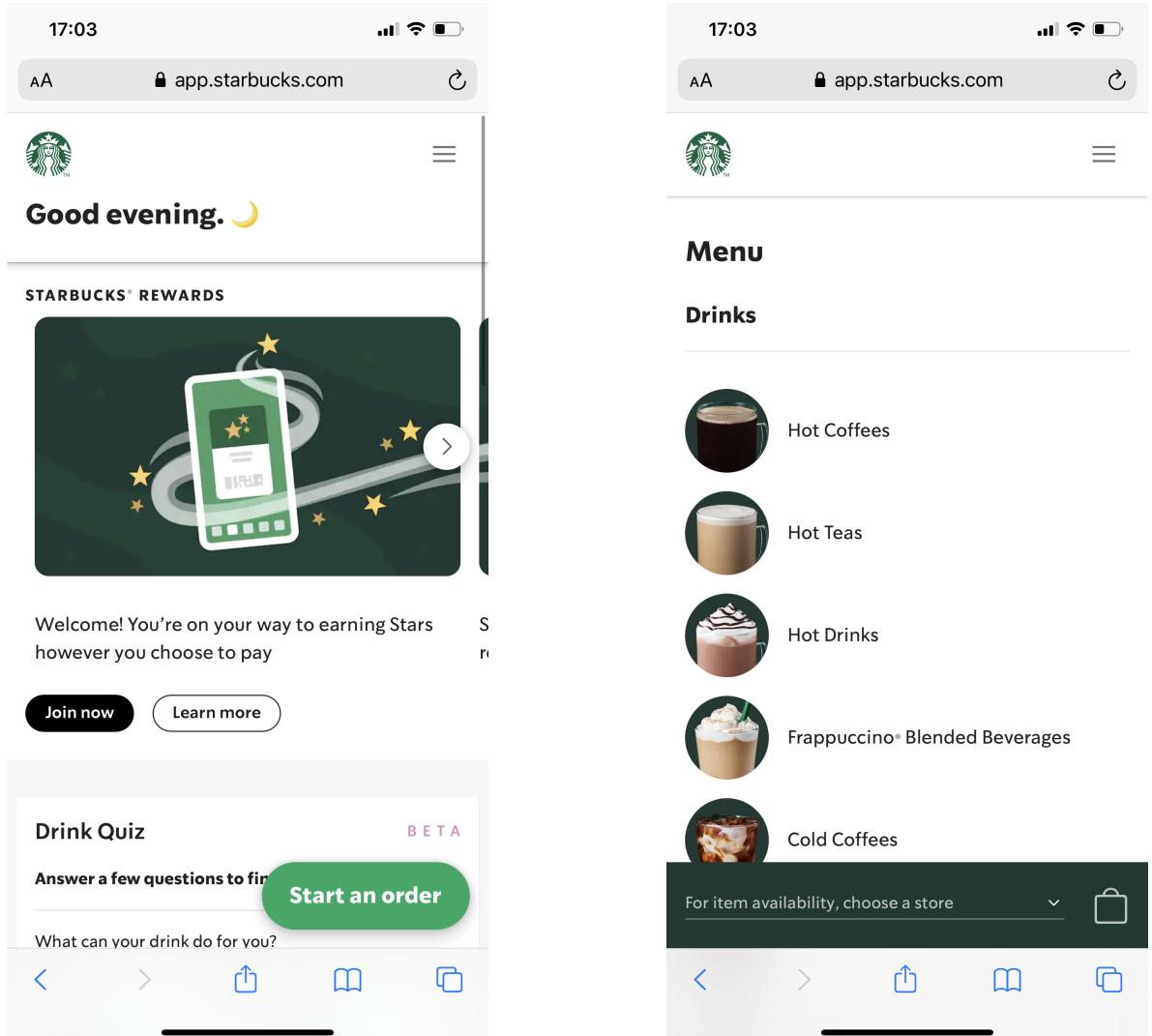
(a) Einlesen der Identitätskarte,
Quelle: Valora, 2021c

(b) Einlesen der Kreditkarte in der avec App,
Quelle: Valora, 2021c

Leider wird in der Applikation keine Auskunft über den Anbieter der Alterverifikation gegeben. Das Vorgehen ist dabei sehr intuitiv und schnell.

2.3 Starbucks PWA

In diversen Quellen wird die PWA von Starbucks immer als eine der besten PWA's genannt. [] Die Applikation ermöglicht es dem Nutzer, die angebotenen Produkte zu bestellen und diese anschließend im Store abzuholen. Der Anwendungszweck ist somit ähnlich zur JTI-Pick-Up Station. Sie ist dabei sehr nahe an einer nativen App, wodurch dem Benutzer die Bedienung sehr leicht fällt. Die Applikation reagiert sehr schnell, es sind keine Ladezeiten zu bemerken. Zudem ist die PWA auch offline nutzbar. Hierbei kommt es zwar zu Einschränkungen in der Nutzung, jedoch lässt sie sich weiterhin bedienen.



2.3.1 Standortsuche

Die Applikation bietet auch ein Feature, um die nächstgelegene Starbucksfiliale anzuzeigen. Hierbei wird auf die aktuelle Position des Nutzers zugegriffen.

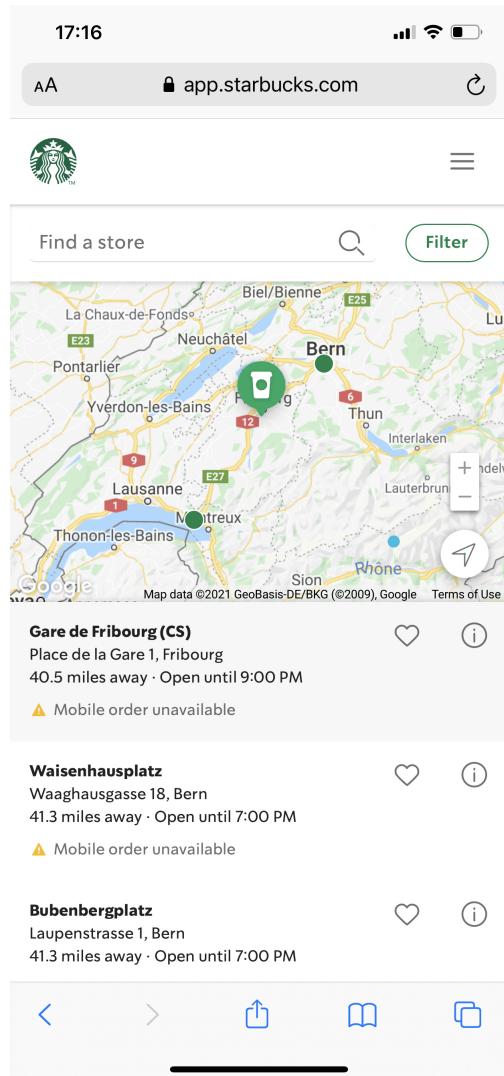


Abbildung 5: Standort in der StarbucksPWA,
Quelle: Starbucks, 2021c

2.3.2 Fazit

Die Applikation von Starbucks zeigt auf, was eine PWA leisten kann. Sie dient als Vorbild für die Applikation der JTI-Pick-Up Station.

2.4 Fazit

Es existieren diverse Produkte, welche einen ähnlichen Ansatz verfolgen wie dieses Projekt. Besonders hervorzuheben ist dabei die avec box. 2.2.2 Der Betreiber verfolgt hier einen ähnlichen Ansatz. Der Kaufvorgang bei Tabakwaren unterscheidet sich dabei kaum von dem in diesem Projekt umzusetzenden. Durch die Analyse des dort verwendeten Vorgehens konnte ein guter Überblick gewonnen werden. Zudem konnte auch gesehen werden, wie die Integration der Altersverifikation umgesetzt wurde. Dieses Wissen ist für die spätere, eigene Umsetzung sehr wichtig.

Die Applikation von Starbucks liefert einen sehr guten Überblick über die Möglichkeiten von PWA's. Besonders Designtechnisch ist diese Anwendung sehr wertvoll.

Die anderen analysierten Angebote liefern keinen Mehrwert für das Projekt, da sie die gestellte Problematik nur bedingt oder gar nicht lösen.

3 Ideen und Konzepte

3.1 Systemarchitektur

Als Systemarchitektur stand die Erweiterbarkeit im Vordergrund. Allerdings sollte die Applikation durch die verwendete Architektur nicht unmöglich komplex werden. Aus diesem Grund wurde bewusst gegen eine Microservicearchitektur entschieden. Die Umsetzung der Applikation mit Microservices würde zwar zu einer besseren Verteilbarkeit und Skalierung führen, der Aufwand der Umsetzung würde jedoch erheblich steigen.

Aus diesem Grund wurde auf eine Schichtenarchitektur gesetzt. Die klassische logische drei Schichten Architektur wurde dabei noch weiter verfeinert. Final wurde eine 6-Schichten-Architektur entworfen. Die Architektur wird zusätzlich in 3 physische Tier aufgeteilt.

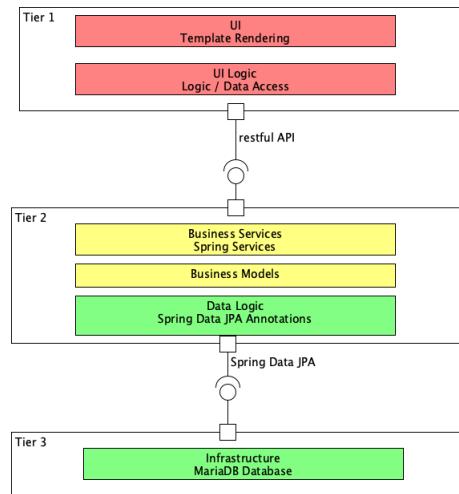


Abbildung 6: Architektur,
Quelle: Autor

Die Architektur ermöglicht dabei eine sehr gute Erweiterbarkeit. Zudem kann die Software verteilt werden und unabhängig voneinander Skaliert werden. Durch die Aufteilung in 6 Tier wird zudem vermieden, dass die einzelnen Tiers zu breit werden. Durch das Verwenden von Frameworks kann zudem die Komplexität gering gehalten werden. Die Schnittstellen innerhalb der einzelnen Layer sind klar vorgegeben. Durch die Kommunikation via REST, bzw. die Spring Data JPA sind die Komponenten untereinander austauschbar. Die Kopplung ist sehr gering.

3.2 Frameworks

3.2.1 Spring Boot

Spring Boot baut auf dem Spring Framework auf. Dieses bietet sich an, um performante Enterprise-Applikationen mit Java zu erstellen. Spring Boot kann dabei sehr einfach erweitert werden, sodass Spring Security, Spring MVC oder Spring Data eingesetzt werden kann.

Spring Boot kommt dabei mit einem Tomcat Server. Dies verringert den Konfigurationsaufwand. Es ist kein Einpacken in ein .war nötig und kein zusätzliches Deployen. Zudem wird das Debugging erheblich erleichtert.

Es übernimmt dabei einen grossen Teil der Beans-Konfiguration von Spring [Waldmann, 2020].

Spring Boot bietet dabei den grossen Vorteil, dass bereits Erfahrung in der Anwendung vorhanden ist. Es ist somit keine Einarbeitungszeit nötig, gängige Fehler können vermieden werden. Aus diesem Grund fiel die Entscheidung gegen Frameworks wie NodeJS.

3.2.2 Angular

Angular ist ein Application Design Framework, um effiziente Single Page Applikationen zu erstellen. Es basiert auf TypeScript [Böhm, 2017]. Zudem bietet Angular die Möglichkeit, als PWA genutzt zu werden [Google, 2021b]. Angular wurde dabei von Google entwickelt und bietet mit dem Material UI bereits viele Elemente, welche von nativen Android-Apps bekannt sind [Google, 2021a].

Wie auch bei Spring ist auch bei Angular bereits Projekterfahrung vorhanden. Ein Einarbeiten ist nicht mehr nötig, die gängigsten Funktionen sind bereits bekannt.

In vorhergehenden Projekten wurde das Frontend mit unterschiedlichen Technologien umgesetzt. Einerseits kam hier Plain JavaScript zum Einsatz, andererseits wurden auch Technologien wie React genutzt. Allerdings überzeugte Angular von all diesen am Meisten. Besonders durch die Verwendung von TypeScript fiel die Wahl auf dieses Framework.

3.3 Weitere Technologien

3.3.1 Docker

Docker ist eine Containertechnologie. Sie erlaubt die Erstellung und den Betrieb von Linux-Containern. Die Container sind dabei sehr leichtgewichtig und modular.

Die einzelnen Teile der Applikation werden in verschiedenen Containern betrieben [RedHat, 2021]. Die Anwendung und Funktion von Docker wird in diversen Modulen an der Hochschule Luzern gelehrt. In vorhergegangenen Projekten wurde Docker bereits im selben Kontext wie hier genutzt.

3.3.2 MariaDB

MariaDB ist ein OpenSource Datenbankmanagementsystem. Es ist durch die Abspaltung von MySQL entstanden. Es handelt sich dabei um ein Relationale Datenbanksystem. Aus Lizenzgründen wird in diesem Projekt MariaDB und nicht MySQL genutzt. Alternativ wäre auch ein Einsatz von PostgreSQL möglich. Der Unterschied zwischen PostgreSQL und MariaDB ist dabei nur marginal. Bei dieser Applikation wurde lediglich aus Erfahrung auf MariaDB gesetzt [DB-Engines, 2021]. Durch die Verwendung von Spring Data wäre es möglich, die Datenbank im Verlauf des Projektes zu wechseln.

3.3.3 Hibernate ORM

Hibernate ORM ist ein Object Relational Mapper. Er ermöglicht dem Entwickler, einfacher mit der Datenpersistierung umzugehen. Hibernate ist zudem ein Teil der Java Persistence API (JPA) und der Spring Data JPA. Ein wichtiger Punkt ist zudem die Performance, welche durch den Einsatz des Lazy Loading Patterns erreicht wird. Die Initialisierung des Objektes wird dabei solange als Möglich hinausgezögert. [Hat, 2021]

3.3.4 GitLab

Zur Versionsverwaltung kam GitLab zum Einsatz. Es wird dabei von der Hochschule Luzern zur Verfügung gestellt. Zudem sind zur Integration in die DevOps Umgebung bereits GitLab Runner vorhanden, um Docker Images zu Builden.

3.3.5 Bezahlungsdienst

Als Bezahlungsdienst wurde Saferpay von Six Payment Services genutzt. Dabei wurde mit der Testversion gearbeitet, diese unterscheidet sich nicht von der Produktiven. Ein Wechsel wäre zudem innert kurzer Zeit durchführbar.

3.3.6 Alterverifikation

Bei der Altersüberprüfung wurde auf Jumio gesetzt. Das Produkt wurde vom Auftraggeber vorgegeben und eine Lizenz bereitgestellt.

3.3.7 Karte

Für die Implementierung der Kartefunktionalität wurde Leaflet genutzt. Die Kartendaten stammen von OpenStreetMap. Zudem wurde Geoapify als MapsAPI eingesetzt.

3.4 Konzepte

3.5 Data Transfer Object

Es handelt sich hier um ein Enterprise Application Architecture Pattern von Martin Fowler. Genauer handelt es sich um ein Distribution Pattern.

Durch den Einsatz dieses Patterns können mehr Daten mit einem Aufruf übertragen werden. Ein weiterer Vorteil ist die klare Trennung zwischen zu serialisierendem Objekt und dem Domain Model. In der Regel wird ein Assembler genutzt, um das DTO auf das Domain Model zu mappen. [Fowler, 2002]

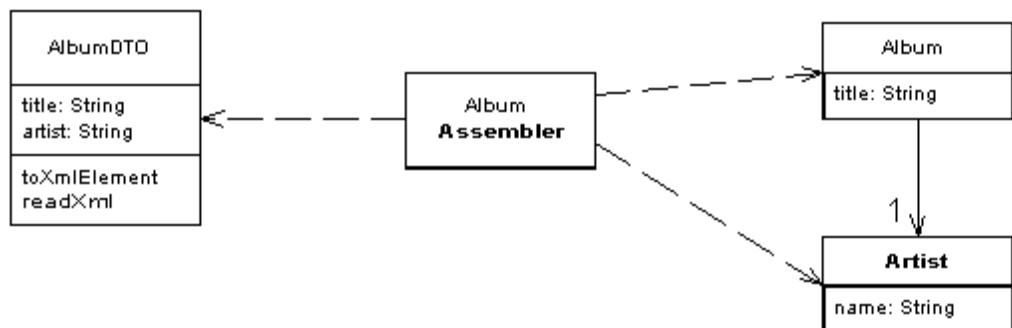


Abbildung 7: DTO Klassendiagramm von Martin Fowler,
Quelle: Fowler, 2002

3.5.1 HATEOAS

In diesem Projekt stand der Einsatz des DTO-Pattern jedoch im Konflikt mit der REST-Abstufung von Leonard Richardson und dem von ihm entworfenen Richardson Maturity Model. Gemäss diesem sollen Daten von einem anderen Domain Model als Hyperlink zurückgegeben werden, um das höchste Level zu erreichen. Somit liefert jeder Aufruf nur das angeforderte Modell zurück. Das Data Transfer Object Pattern dient somit nur zur Abgrenzung zwischen dem zu sendenden Objekt und dem Domain Model.

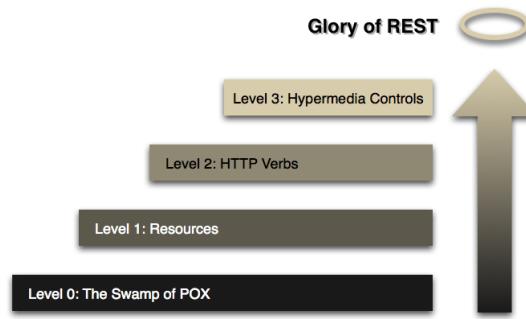


Abbildung 8: Richardson Maturity Model,
Quelle: Fowler, 2010

Dabei wurde auf REST-Level 3 hingearbeitet und entsprechend mit HATEOAS gearbeitet. Dadurch kann das Backend weiter vom Frontend getrennt werden, da bei einer Anpassung der URL auf dem Server keinen Einfluss auf die Funktionalität des Clients hat. Zudem kann die Verständlichkeit der API verbessert und ein Erweitern vereinfacht werden. [Fowler, 2010]

3.5.2 Spring Doc

Die Spring Doc ermöglicht eine sehr einfache und schnelle Dokumentation von REST-API basierend auf der OpenAPI Spezifikation. Um die Dokumentation anzuzeigen, wird das Open Source Tool Swagger UI genutzt. Hiermit lässt sich eine dynamische API-Dokumentation als HTML-Page erstellen.

The screenshot shows a detailed view of an API endpoint from an OpenAPI definition. At the top, it says "OpenAPI definition v0 OAS3". Below that, there's a "Servers" section with a dropdown set to "http://localhost:8080 - Generated server url". The main area is titled "product-controller". It shows a "GET /api/v1/products/{id}" operation. Under "Parameters", there is one parameter named "id" which is required and has type "integer(\$int32)". There is also a "Try it out" button. The "Responses" section shows a single response for code 200, labeled "OK". The "Media type" dropdown is set to "application/json", with a note that it "Controls Accept header". Below this are "Example Value" and "Schema" buttons. The schema is defined as "ProductDTO" with fields: product_id (integer(\$int32)), name (string), description (string), price (number(\$double)), image (string), and links (array). A note at the bottom right of the responses section says "No links".

Abbildung 9: Ausschnitt aus der Swagger Dokumentation,
Quelle: Autoren

4 Methoden

4.1 SoDa

Das Projektmanagement wird mit dem hybriden Projektvorgehen SoDa der Hochschule Luzern durchgeführt. SoDa wurde dabei bereits schon in vorhergehenden Softwareentwicklungsmodulen eingesetzt und hat sich hier bewährt. Die Userstories werden dabei auch im GitLab erfasst.

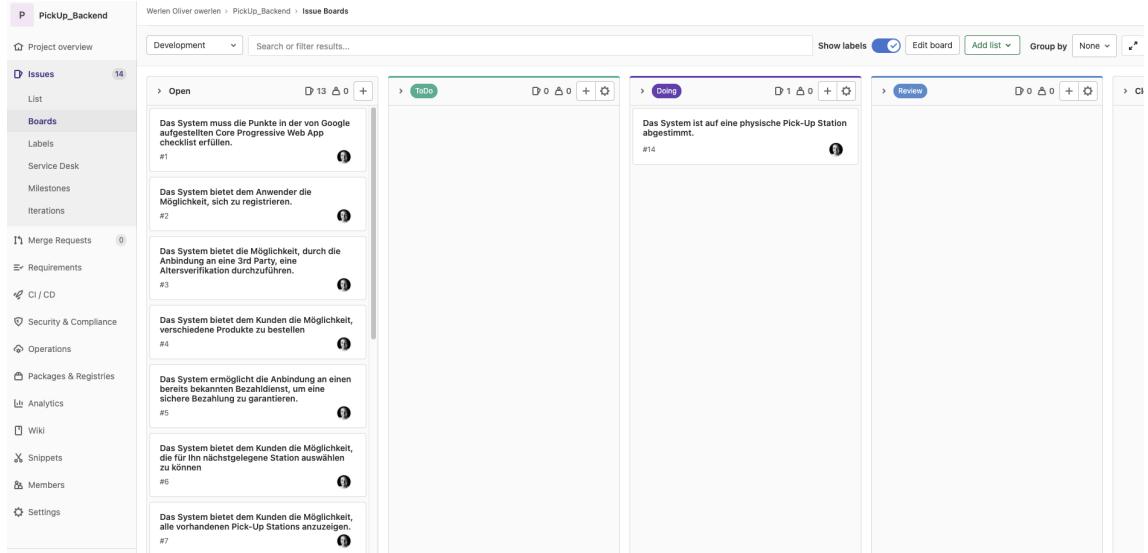


Abbildung 10: GitLab Board,
Quelle: Autor

Für die Sprint Planung werden einzelne User Stories verschoben. Beim Sprint Review werden die Stories in "reviewing" mit den definierten Akzeptanzkriterien mit den Resultaten verglichen. Basierend darauf wird entschieden, ob an der User Story noch weiter gearbeitet werden, das heisst zurück zu "doing" oder die Story in "done" verschoben werden kann. Bei Beginn des nächsten Sprints wird das Vorgehen wiederholt.

Die einzelnen Items sind dabei priorisiert. Elemente mit einer hohen Einstufung werden dabei bei der Bearbeitung vorgezogen.

4.2 CI/CD

Die gesamte Continous Integration and Continous Deployment (CI/CD) Pipeline wurde für dieses Projekt neu erstellt. Dabei besteht der Prozess aus drei Stages:

1. build
2. package
3. deploy

4.2.1 build

Abhängig vom Projekt wird das Projekt gebuilded. Bei der Java Applikation handelt es sich hier um ein maven-package, beim Angular Frontend um ein ng build -prod. Die resultierenden Artifakte werden für zwei Stunden im GitLab gespeichert.

4.2.2 package

In der Package Stage wird das Docker Image aus den vorherigen Artifakten erstellt. Dabei wird das Dockerfile des jeweiligen Produkts genutzt. Der Build wird von einem Shared Runner durchgeführt. Hier werden 4 vom Enterpriselab zur Verfügung gestellt. Anschliessend wird dieses in die Container Registry des GitLab Projekts gepushed.

4.2.3 deploy

Für das Deployment wurde auf den virtuellen Maschinen ein GitLab Runner installiert. Durch den deployment-Tag wird dieser adressiert. Auf diese Maschine wird ein Login ausgeführt und anschliessend das aktuellste Image heruntergeladen. Es wird der bestehende Container gelöscht und aus dem neuen Image ein neuer Container gestartet.

4.2.4 Sequenzdiagramm

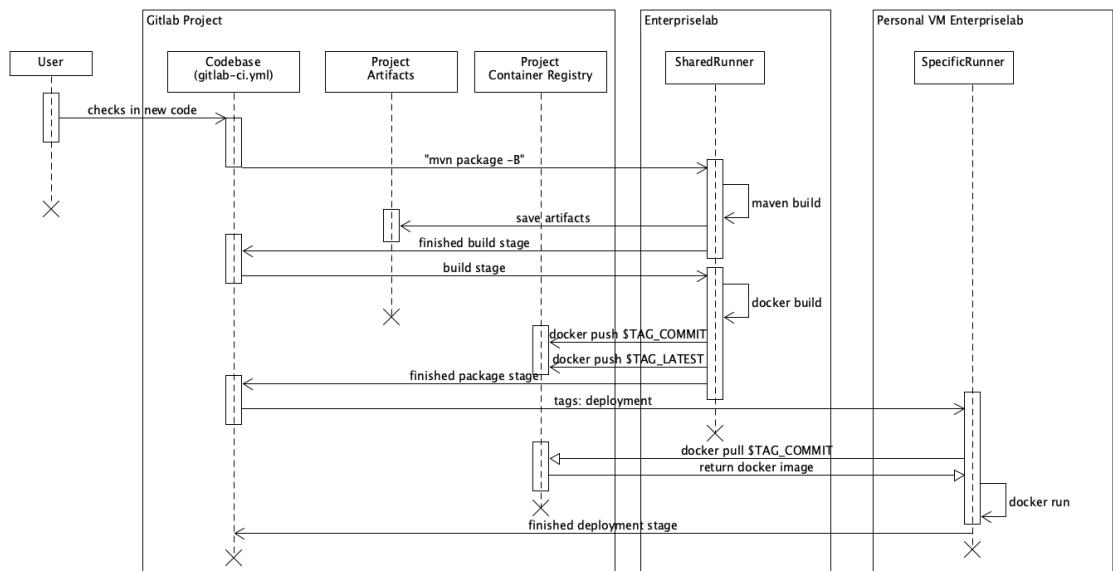


Abbildung 11: Ablauf der CI/CD Pipeline,
Quelle: Autor

4.3 Testing

Das Projektziel ist ein Prototyp. Aus diesem Grund wurde der Fokus auf die Funktion gesetzt. Daher wurden die Unit- und Integrationstests nur konzeptuell umgesetzt. Bei Bedarf können diese erweitert werden.

5 Realisierung

5.1 Initialisierungsphase

5.1.1 Kick-Off Meeting

Das Kick-Off Meeting fand am 23.02.2021 als Zoom Meeting statt. Das Sitzungsprotokoll dazu ist im Kapitel E.1 zu finden.

Es waren bei diesem Meeting alle Projektbeteiligten anwesend. In erster Linie wurde von Herr Meier das genaue Vorgehen bei der Bachelorarbeit vorgestellt. Anschliessend stellte der Auftraggeber das Projekt genauer vor und zeigte dabei seine Erwartungen auf. Die Aufgabenstellung wurde finalisiert und von allen Projektbeteiligten akzeptiert. Zudem schlug der Betreuer vor, im zwei bis drei Wochenrhythmus ein Meeting abzuhalten. Im Anschluss wurde ein Termin für das erste Meeting vereinbart.

5.1.2 Erstellen des Projektmanagementplans

Gemäss Software Development Agile (SoDa) wurde in einem ersten Projektschritt der Projektmanagementplan Berstellt. In diesem wurde auch gleich der Rahmenplan erarbeitet. In diesem sind unter anderem die Meilensteine des Projekts dargestellt. Der Projektstrukturplan erleichterte dabei das Finden. Sie wurden in einem nächsten Schritt genauer spezifiziert und die Deliverables für ein erfolgreiches Erreichen des Meilensteins definiert. Durch den Projektstrukturplan konnten die einzelnen Teilbereiche des Projekts aufgelistet werden. In der folgenden Risikoanalyse wurden die Risiken und entsprechenden Gegenmassnahmen erarbeitet.

Als letztes wurde die Projektunterstützung genauer spezifiziert. Dabei wurden die zu verwendenden Tools sowie die Elemente der Konfigurationseinheit festgelegt.

Als letzter Teil des Projektmanagements wurde die Teststrategie und die Testdrehbücher formuliert. Die Testdrehbücher werden während des Projekts vorlaufend formuliert.

5.1.3 Problem und Vision

Das Kapitel 1 wurde zum Beginn der Initialisierungsphase bearbeitet. Die Hauptprobleme konnten dabei sehr schnell gefunden werden. Die Vision des Projektes konnte mithilfe der Aussagen des Auftraggebers im Kick-Off Meeting E.1 sehr gut beschrieben werden.

5.1.4 Requirement Engineering

Beim Requirements Engineering diente die IEEE Spezifikation 29148-2018 als Grundlage [Doran, 2018]. Um die Requirements zu finden, wurde in einem ersten Schritt eine Analyse der Aufgabenstellung durchgeführt. Ergänzt wurden diese durch Befragungen des Auftraggebers. Durch die so erlangten Informationen konnten die Anforderungen formuliert werden. In einem letzten Schritt wurden diese mit dem Auftraggeber besprochen. Hierbei wurden noch einige Anpassungen gemacht.

5.1.5 Stand der Technik

In diesem Kapitel wurde eine Analyse der bestehenden, vergleichbaren Lösungen durchgeführt. Dabei wurde vor allem das Angebot der Post und von Valflora genauer betrachtet. Die bekannte PWA von Starbucks zeigte dabei sehr gut deren Möglichkeiten auf.

5.1.6 Meilenstein Abschluss Initialisierungsphase

Meilensteinbericht

Termin Meilenstein 2 Der Meilenstein 2 ist am 07.03.2021 abgeschlossen und somit pünktlich fertiggestellt worden.

Beschreibung Meilenstein 2 Die Beschreibung des Meilensteins ist im Abschnitt B.2.2 ersichtlich.

Meilensteinziele/Vorgaben Das übergeordnete Ziel dieses Meilensteins ist die Fertigstellung der Initialisierungsphase. Hierzu war die Auslieferung der nachfolgenden Artefakte notwendig:

- Projektmanagementplan
- Systemspezifikation
- Anforderungsliste

Zusätzlich wurden bereits die Kapitel 1 und 2 fertiggestellt.

Meilensteinzielerreichung Es konnten alle geforderten Artefakte geliefert werden. Die Artefakte wurden bereits mit der Betreuungsperson im Meeting E.3 besprochen und konnten abgenommen werden. Der Meilenstein wurde erfolgreich erreicht.

Fazit Es wurden alle Artefakte erarbeitet. Der Meilenstein wurde somit erreicht und es kann weiter nach Plan gearbeitet werden.

5.2 Konzeptionsphase

5.2.1 Sprint 1

User Story	Number
Das System ist auf eine physische Pick-Up Station abgestimmt.	F.2

Tabelle 1: Userstories Sprint 1,
Quelle: Autor

Spezifikation der Schnittstelle zur Abholung Um eine Abholung der Produkte zu bekommen, braucht es eine Kommunikation zwischen der PWA und der Pick-Up Station. Um diese Schnittstelle genauer spezifizieren zu können, war es in einem ersten Schritt nötig, eine entsprechende Übertragungstechnologie festzulegen. Bei der Auswahl war dabei die Kompatibilität mit verschiedenen Geräten und Browsern ausschlaggebend.

NFC Als eine erste Idee wurde Near Field Communication (NFC) analysiert. NFC eignet sich dabei ideal für die Übertragung von geringen Datenmenge. Dabei funktioniert es bis zu einer Distanz von 10cm. Die Umsetzung wäre dabei sehr einfach mittels einem Raspberry Pi umsetzbar. Einfache NFC-Leser gibt es dabei schon für wenig Geld zu kaufen. Dabei könnte die gesamte Übertragung von der Informatik übernommen werden, eine weitere Schnittstelle zwischen Elektrotechnik und Informatik könnte dabei vermieden werden.

Die Technologie ist dabei sehr robust. Ein Überkleben oder Zerkratzen des Lesers hat keinen Einfluss auf die Funktionalität [congstar, 2021]. Der Abholvorgang wäre sehr einfach und schnell, da nur ein kurzer Kontakt mit dem Smartphone bereits ausreicht.

Kompatibilität Apple schränkt den Zugriff von Webpages auf Standort- und Hardwaredienste unter iOS-Geräten sehr stark ein. Offiziell begründet wurde dies durch die Einschränkung von footprinting und des damit verbundenen Nutzertrackings. Für dieses Projekt ausschlaggebend ist dabei vor allem das Entfernen von NFC- und Bluetooth-Support. Somit ist es auch PWA's nicht mehr möglich, unter iOS auf die genannten Funktionen zuzugreifen [Apple, 2021].

Die Web-NFC-API wird zusätzlich bislang nur von Google Chrome unterstützt. Dies würde die Nutzbarkeit der Applikation sehr stark einschränken [Mozilla, 2021b]. Aus diesem Grund eignet sich NFC nicht für die Verwendung in diesem Projekt.

QR-Code Die Restriktionen von Apple schränken die geeigneten Technologien sehr stark ein. Es bleibt nur noch der Zugriff auf die Kamera, um mit der Pick-Up Station zu kommunizieren. Ursprünglich war dies vom Auftraggeber nicht gewünscht. Das Verwenden einer anderen Technologie würde aber viel zu viele Einschränkungen führen, sodass die PWA mit vielen Geräten unbrauchbar wäre.

Aus diesem Grund musste ein Konzept entwickelt werden, um die Warenausgabe mittels QR-Code auszulösen. Dabei sind zwei Ansätze möglich:

- Fixer QR-Code auf Pick-Up Station, wird von PWA eingelesen.
- Variabler QR-Code in PWA, wird von Pick-Up Station eingelesen.

Aus hardwaretechnischer Sicht ist die Umsetzung der ersten Lösung bedeutend einfacher Umzusetzen, bietet aber auch Fehlerpotential. Ein Überkleben des QR-Codes auf der Station würde die gesamte Station unbrauchbar machen. Eine Abholung der Bestellungen wäre nicht mehr möglich. Die zweite Variante ist hardwaretechnisch anspruchsvoller. Auf der Pick-Up Station muss ein optischer Leser verbaut werden. Mit diesem kann der QR-Code aus der PWA eingelesen und die Bestellung ausgegeben werden.



Abbildung 12: Beispielanwendung QR-Code auf Gerät,
Quelle: tagmotion, 2018

Eine Grafik zum zweiten Lösungsansatz ist dabei im Kapitel 2.2.2 zu finden. Die genaue Wahl wird im Meeting mit dem Auftraggeber besprochen und abgesegnet.

5.2.2 Sprintreview Sprint 1

Im Sprint 1 wurde eine genauere Analyse der Schnittstelle zur Produktabholung durchgeführt. Dabei müssen im Meeting von dieser Woche die gefundenen Lösungen mit dem Auftraggeber besprochen werden. Die Userstory kann erst zu Beginn des nächsten Sprints abgeschlossen werden.

Meilensteinbericht

Termin Meilenstein 3 Der Meilenstein 3 ist am 18.03.2021 abgeschlossen und somit mit drei Tagen Verspätung fertiggestellt worden.

Beschreibung Meilenstein 2 Die Beschreibung des Meilensteins ist im Abschnitt B.2.2 ersichtlich.

Meilensteinziele/Vorgaben Das übergeordnete Ziel dieses Meilensteins ist die Fertigstellung der Initialisierungsphase. Hierzu war die Auslieferung der nachfolgenden Artefakte notwendig:

- Projektmanagementplan
- Systemspezifikation
- Anforderungsliste

Zusätzlich wurden bereits die Kapitel 1 und 2 fertiggestellt.

Meilensteinzielerreichung Es konnten alle geforderten Artefakte geliefert werden. Die Artefakte wurden bereits mit der Betreuungsperson im Meeting E.3 besprochen und konnten abgenommen werden. Der Meilenstein wurde erfolgreich erreicht.

Fazit Es wurden alle Artefakte erarbeitet. Der Meilenstein wurde somit erreicht und es kann weiter nach Plan gearbeitet werden.

5.2.3 Sprint 2

User Story	Number
Das System ist auf eine physische Pick-Up Station abgestimmt.	F.2
Das System bietet dem Kunden die Möglichkeit, verschiedene Produkte zu bestellen.	F.6
Das System muss über eine CI/CD-Pipeline verfügen.	L.6
Das System muss via HTTPS kommunizieren.	L.3

Tabelle 2: Userstories Sprint 2,
Quelle: Autor

CI/CD Pipeline Nach Absprache mit dem Auftraggeber im Meeting wurde entschieden, dass der Prototyp im EnterpriseLab laufen soll. Daraufhin wurde eine Maschine beantragt. Es handelt sich hierbei um ein Ubuntu 16.07 LTS. Die Applikationen sollen dabei als Docker Container deployed werden.

Zuerst wurde geplant, die Pipeline wie im offiziellen Tutorial des Enterpriselabs zu erstellen. Auf Anfrage wurde jedoch ein anderes Vorgehen empfohlen. Nachfolgend wird die Antwort zitiert.

"Wenn es dein Ziel ist eine Spring Boot Applikation zu bilden und dann auf der VM zu deployen dann würde den Container auf den Shared Runner unserer GitLab Instanz bilden lassen und in die Container Registry deines Projekts pushen. Für die Deploy Stage der CI/CD Pipeline kannst du deine VM als privaten GitLab Runner registrieren und so ohne SSH login den Container von der Registry pullen und laufen lassen. Die SSL Termination mit Lets Encrypt würde ich mit einem separaten nginx Container lösen der reverse proxy spielt. Dieser kann dann einfach laufen und muss für Änderungen an der Spring Boot Applikation auch nie modifiziert werden.

Der Vorteil im Vergleich zur Docker Übung ist, dass hier alle Hosts von der GitLab CI/CD Pipeline kontrolliert werden. In der Übung ist der docker-cloud-exercise Host abgekapselt und

pullt mit Watchtower einfach blind das neuste Image von einer Registry. Dieser Aufbau macht IMO mehr Sinn wenn man einfach Container Images von dritten konsumiert, aber ist weniger elegant wenn man selbst Kontrolle über die Source und CI/CD Pipelines hat." [von Uslar, 2021]

In diesem Auszug aus der Email vom Enterpriselabmitarbeiter Cyril von Uslar sind sehr viele Informationen enthalten. Es war ein mehrmaliges Durchlesen nötig, um sich darunter etwas vorstellen zu können. Anschliessend wurde die Pipelineerstellung in die wesentlichen Punkte aufgeteilt um umgesetzt.

- Builden auf dem Shared Runner
- Pushen in die Container Registry des Projekts
- VM als privaten Runner registrieren und deployen
- nginx server als reverse Proxy für SSL Termination

Builden auf dem Shared Runner In diesem Projekt wird die Pipeline für zwei Projekte aufgesetzt. In einem ersten Schritt wurde dies nur für die Spring Applikation durchgeführt. Das Vorgehen unterscheidet sich dabei nur im Buildprozess. Um die Spring Applikation zu Builden, wurde das Dockerfile identisch zum Spring Boot Docker-Tutorial aufgebaut [VMWare, 2021]

```
FROM maven:3.6.3-jdk-11-slim
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

Entgegen dem Tutorial, in welchem noch die Java Version 11 genutzt wird, kommt hier direkt Java 16 zum Einsatz.

Anschliessend wurde das .gitlab-ci.yml File erstellt. Es wurde hier eine Anleitung aus dem Gitlab Blog genutzt. Diese ist jedoch nicht mehr ganz aktuell. Das Deployment wird hier zudem auf ein Kubernetes Cluster durchgeführt. Es musste daher individuell angepasst werden. Es konnte somit nur der Build-Teil übernommen werden. [Lenzo, 2016]

Gitlab stellt zum Builden von Docker-Images bereits mehrere Shared-Runner zur Verfügung.

Shared runners

These runners are shared across this GitLab instance.

The same shared runner executes code from multiple projects, unless you configure autoscaling with [MaxBuilds](#) set to 1 (which it is on GitLab.com).

[Disable shared runners](#) for this project

Available shared runners: 5

● e1f64da7

gitlabrunner-02

#37

[docker](#) [runner02](#) [test-runner](#) [ubuntu 18.04](#)

● piYF7yKo

gitlabrunner-06

#56

[docker](#) [runner06](#) [ubuntu 18.04](#)

● 8bU3YzuD

gitlabrunner-01

#62

[BigBuild](#) [docker](#) [runner01](#) [ubuntu 18.04](#)

● cf1ce3b4

gitlabrunner-03

#49

[docker](#) [runner03](#) [ubuntu 18.04](#)

● ydSz8fXG

gitlabrunner-05

#55

[docker](#) [runner05](#) [ubuntu 18.04](#)

Abbildung 13: Verfügbare Shared-Runner von GitLab,
Quelle: Autoren

Der jeweilige Runner wird dabei mittels eines "fair usage algorithms" zugewiesen. Dabei ist die Anzahl der momentan auszuführenden Jobs pro Projekt entscheidend für die Wahl. Das Projekt mit den am wenigsten laufenden Jobs kommt zuerst. [GitLab, 2021] Zudemachtet der runner auf die verwendeten Tags. In diesem expliziten Beispiel sind mehrere Runner mit dem Docker-Tag versehen. Daher kann nicht sicher gesagt werden, welcher Runner den Docker Build durchführt.

Builden in die Registry des Projekt Um das erstellte Image in die Registry des Projekts zu pushen, werden von GitLab bereits die einzelnen Befehle vorgegeben. Diese können in die package-Stage integriert werden.

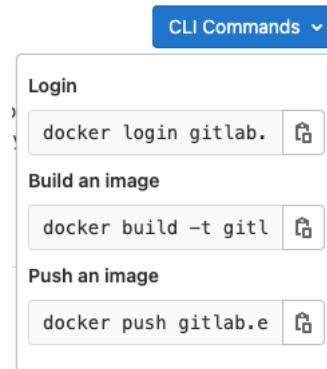


Abbildung 14: CLI Commands GitLab Container Registry,
Quelle: Autoren

Das entsprechende Image wird so in die interne Container Registry des GitLab Projekts gepushed. Dadurch kann auf das Verwenden eines Drittddienstes wie DockerHub verzichtet werden. Beim Build werden immer zwei Images in die Registry geschrieben. Einerseits eines mit dem Tag "latest" und eines mit dem Commit-Hash als Tag. Beim Deployment wird dann immer das Image mit dem aktuellen Commit-Hash verwendet. Dieser Mechanismus dient dazu, dass alle Versionen immer vorhanden sind. "Latest" weist dabei immer auf die aktuelle Version.

VM als privaten Runner registrieren und deployen Um das Deployment ohne Watchtower durchführen zu können, wurde auf die Virtuelle Maschine als Docker-Runner hinzugefügt. Es handelt sich somit dabei um einen specific Runner. GitLab Runner kann dabei einfach mittels Paketmanager auf Ubuntu installiert werden. Anschliessend musste dieser noch konfiguriert werden. Dazu musste der Token sowie die URL von GitLab dem Runner hinzugefügt. Der Runner ist anschliessend im Runner Bereich des Projekts zu finden. Zudem wurde anschliessend die Option "Lock to current project" entfernt, sodass dieser Runner auch direkt für das Angular Projekt genutzt werden kann. Um ein SSH-Login zu vermeiden, wurde der Executor dabei als Shell-Executor definiert. Bei einem SSH-Executor wäre vorgängig die Verbindung via SSH nötig gewesen. Anschliessend können die entsprechenden Befehle für das Deployen in das gitlab-ci.yml integriert werden. Dabei war es zunächst ein Problem, dass die Berechtigung fehlte. Erst nach dem Login mittels GitLab CI-Token konnte der Container erfolgreich gepulled und anschliessend gestartet werden.

Build und Deployment des Frontends Um mit dem nächsten Schritt weiterzufahren, musste zuerst das Deployment des Angular Projekts konfiguriert werden. Hierbei liegt der Hauptunterschied in dem Build-Prozess. Dabei ist es wichtig, die beim Build entstehenden Artifakte in GitLab zu speichern. Nur so kann auf ein erneutes Builden beim Erstellen des Docker Containers verzichtet werden. Zudem werden die Node Module in den Cache gespeichert. Die nachfolgenden Befehle sind identisch zum Vorgehen bei Spring.

nginx als Reverse Proxy Um den nginx-Server gegen Zugriffe von Aussen zu schützen, wurde auf einen Reverse Proxy gesetzt. Der Reverse Proxy stellt dabei die einzige Verbindung ins öffentliche Netz dar. Zudem übernimmt er die Zertifikatsverwaltung des Frontends. Alternativ wäre auch ein Caching möglich, auf die Umsetzung wurde für diesen Anwendungsfall jedoch verzichtet. [KnowHow, 2020]

Die Umsetzung wurde dabei analog zum Tutorial von Alexander Bohndorf durchgeführt. [Bohndorf, o.D.] Dabei wurde der nginx-proxy von jwilder sowie die damit kompatible letsencrypt companion verwendet. Die Umsetzung wurde dabei mittels docker-compose Files durchgeführt.

Abschliessende Bemerkungen Das Erstellen der CI/CD Pipelines des Projekts verlief grösstenteils problemlos. Durch die Grundlage des Enterpriselabs und der sehr guten Dokumentation von GitLab konnte dies sehr schnell umgesetzt werden. Durch das Hinzufügen des Reverse Proxies konnte die Sicherheit des Systems massiv erhöht werden. Zusätzlich wurde so auch gleich die Auslieferung via HTTPS hinzugefügt sowie für ein immer gültiges Zertifikat gesorgt. Beim Backend wurde auf eine zusätzliche Test-Stage verzichtet. Die gesamten Tests werden bereits im Maven-Build ausgeführt. Beim Frontendprojekt wurde dazu eine eigene Stage für e2e Tests festgelegt. Die fertigen Konfigurationen sind im GitLab Projekt enthalten und können dort eingesehen werden.

Bestellen von Produkten Um eine Bestellung durchführen zu können, wurde in einem ersten Schritt das Article Object definiert. Basierend auf diesem wurde das passende Data Transfer Object (DTO) definiert. Um das Mapping zwischen DTOs und Objekt zu vereinfachen, wurde der Object Mapper modelmapper genutzt. Zudem wurde direkt mit HATEOAS gearbeitet. Der Controller stellt dabei die gewohnten CRUD-Operationen zur Verfügung.

Abstimmung auf physische Pick Up Station Im Meeting von dieser Woche wurden dem Auftraggeber die beiden in Kapitel 5.2.1 erarbeiteten Lösungen vorgestellt. Der Entscheid fiel dabei zugunsten der ersten Alternative. Dabei befindet sich der QR-Code fix auf der Station. Diese ist deutlich robuster und einfacher umzusetzen.

Die Schnittstellen zwischen Elektrotechnik und Informatik sind essentiell für die Funktionalität des Endprodukts. Aus diesem Grund wurde in diesem Sprint ein Meeting zwischen den Projektbeteiligten einberufen. Besonders das Senden der Ausgabeanforderung führte dabei zu Problemen. Hierbei soll auf einen Busy-Waiting Ansatz verzichtet werden. Jedoch soll die Lösung auch sehr energiesparend und effizient umsetzbar sein. Da beide Beteiligten keine Erfahrung im Internet of Things (IoT)-Umfeld besitzen, wurde hier der Betreuer Michael Handschuh um Hilfe gebeten.

5.2.4 Sprintreview Sprint 2

In Sprint 2 konnten nicht alle User Stories vollständig erfüllt werden. Es werden daher zwei von drei User Stories im nächsten Sprint weiter bearbeitet. Die Umsetzung der CI/CD Pipeline ist hingegen abgeschlossen. Dies war auch die Hauptarbeit in diesem Sprint. Für die Spezifizierung der Schnittstelle wird beim Meeting mit dem Betreuer eine geeignete Lösung erarbeitet. Bei dem Bestellprozess ist bereits die Abfrage von Produkten an der API möglich. In einem nächsten Schritt wird das passende Frontend entworfen und umgesetzt.

5.2.5 Sprint 3

User Stories In diesem Sprint wurden keine neuen User Stories zum Sprint Backlog hinzugefügt. Es wird weiterhin an den beiden verbleibenden User Stories gearbeitet.

Abstimmung auf eine physische Pick Up Station Bei der Abstimmung auf die physische Pick Up Station musste das geplante Vorgehen mit dem Projektbetreuer besprochen werden. Im Meeting E.5 wurde eine geeignete Lösung gesucht. Dabei standen zwei Möglichkeiten zur Auswahl.

WebHooks WebHooks werden dabei zur Kommunikation zwischen zwei Diensten genutzt. Das Vorgehen ist dabei vergleichbar mit dem aus der Programmierung bekannten Observer Pattern. Dabei wird das Push-Verhalten genutzt. Es sind keine Requests nötig, um allfällige neue Daten zu bemerken. Die API pusht die Daten zum WebHooks-Endpoint. Der Event wird dabei mit POST-Request ausgelöst. Dabei wird Webhooks auch von diversen Zahlungsanbietern wie PayPal genutzt. Es handelt sich hierbei um eine asynchrone Kommunikation. [Laboissone, 2018] WebHooks erfüllt dabei alle Anforderungen, welche zur Kommunikation zwischen API und PickUp Station

benötigt werden.

Allerdings ist noch keine Erfahrung mit WebHooks vorhanden. Jedoch kann zur Erstellung auch Node.js mit Express verwendet werden. Dieses ist bekannt und wurde auch schon in anderen Projekten genutzt.

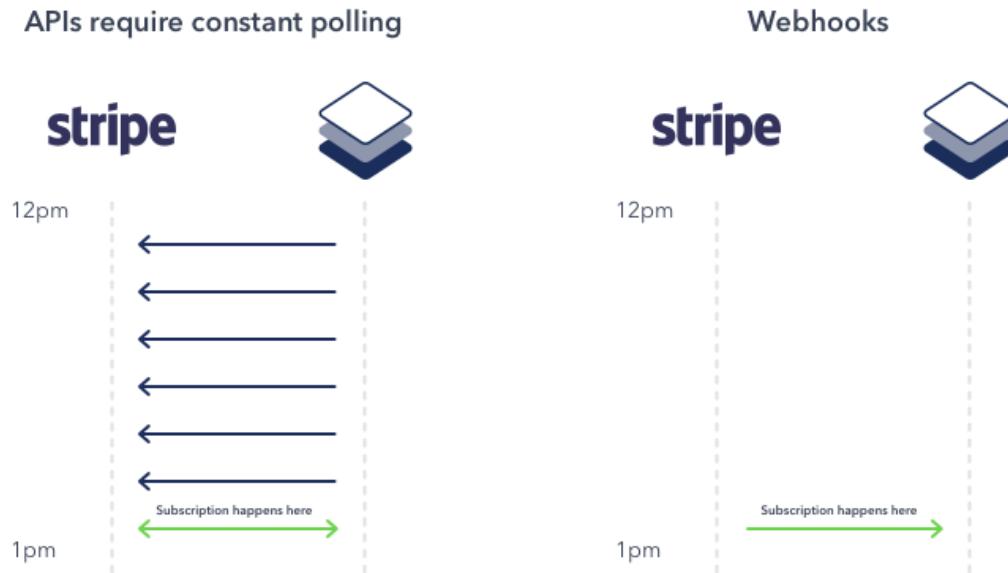


Abbildung 15: Webhooks Funktionsweise,
Quelle: Johnson, 2020

Sobald die entsprechende User Story bearbeitet wird, wird der entsprechende WebHooks-Endpoint erstellt.

Produktbestellung In diesem Sprint wurde zudem an der Produktbestellung gearbeitet. Als Erstes wurde das Darstellen von Produkten ins Auge gefasst. Zudem wurde die Grundstruktur der Angular Applikation erstellt. Dabei wurde strikt nach dem Mobile First Ansatz gearbeitet. Durch Schematic konnte das responsive Menu sowie die Anzeige für die Produkte sehr schnell erstellt werden. Im Anschluss wurde das Menü mit den gewünschten Inhalten gefüllt.

Bei den Produkten wurde das Card Element von Angular Material gesetzt. Dieses wurde in ein Grid Layout verpackt. Damit die Seite auf allen Geräten optimal aussieht, wird das Layout basierend auf der Displaygrösse angepasst. Um dies mit TypeScript umzusetzen, wurde ein Breakpoint Observer eingesetzt. Mithilfe von diesem wird die Anzahl Spalten und die Spaltenhöhe abhängig von der Geräteauflösung festgelegt. Sie wird dabei beim Laden der Seite festgelegt. Es ist hier kein Listener Mechanismus nötig, da sich die Auflösung nicht während der Laufzeit ändert.

Um die Anzahl Spalten bei einem Wechsel in den Landscape Modus anzupassen, wurde ein neuer Observer eingesetzt. Dieser legt die entsprechende Anzahl fest.

```

this.layoutChangesTabletLandscape.subscribe(result => {
  if(result.matches){
    this.cols = 4
    this.rowHeight = "400px"
  }
});
this.layoutChangesTabletPortrait.subscribe(result => {
  if(result.matches){
    this.cols = 3
    this.rowHeight = "350px"
  }
});

```

- (a) Observer, Wechsel Landscape/Portrait
Tablet,
Quelle: Autor

```

if(this.matchedSmartphone){
  this.cols = 2
  this.rowHeight= "300px"
}
if(this.matchedDesktop){
  this.cols = 4
  this.rowHeight = "450px"
}

```

- (b) Spaltenanzahl- und Höhe abhängig von
Auflösung,
Quelle: Autor

Anzeige der Produkte Die Abfrage der Produkte war bereits in einem vorderen Sprint 5.2.3 im Backend erstellt. Ergänzend dazu wurde ein Image Controller definiert. Dieser ermöglicht es, via dem Bildnamen das passende Bild zu erhalten. Bei der Abfrage eines Produkts wird dieser dabei als Link mitgesendet. Die Produkte wurden analog zur Java Klasse als TypeScript Klasse definiert. Die Links via HATEAOOS wurden dabei als eigenes Property links[] definiert.

Anschliessend wurde mittels Angular-http Client ein Get-Request an die entsprechende URL gesendet. Das resultierende Observable wurde abonniert und anschliessend mittels Angular Direktive als Card Element dargestellt.

(a) Produktanzeige auf Pixel Phone,
Quelle: Autor

(b) Produktanzeige auf iPad,
Quelle: Autor

Anschliessend wurde mittels CSS das Design wie gewünscht angepasst.

Sprintreview Sprint 3 Die Userstory von dieser Woche konnten erneut nicht abgeschlossen werden. Aus diesem Grund wurden die Requirements aufgeteilt. Die neuen Requirements sind im Kapitel D zu finden.

5.2.6 Sprint 4

User Story	Number
Das System bietet die Möglichkeit, verschiedene Produkte anzuzeigen	F.5
Das System bietet die Möglichkeit, verschiedene Produkte dem Warenkorb hinzuzufügen	F.6

Tabelle 3: Userstories Sprint 4,
Quelle: Autor

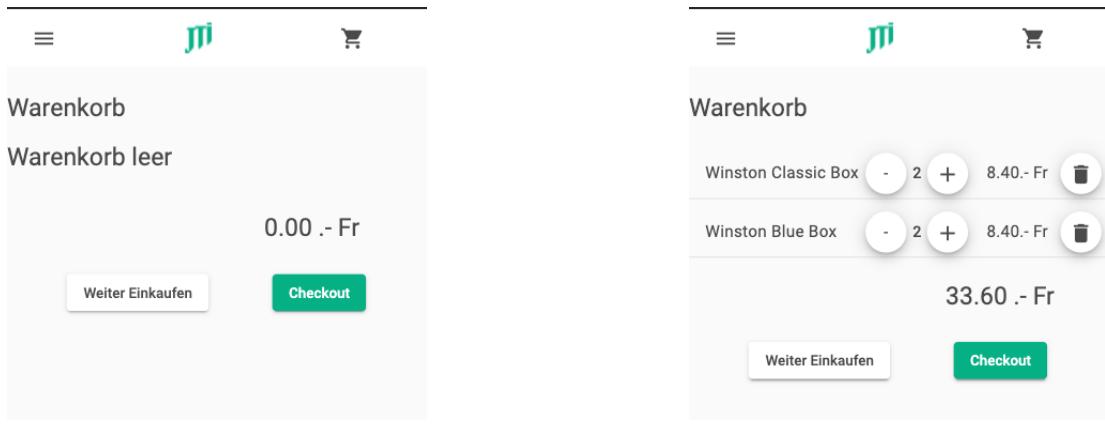
Produkte anzuzeigen Vom Auftraggeber wurden die Produkte in einer Excel Liste abgegeben. Hierbei war auch ein Packshot vorhanden. Es wurde relativ schnell klar, dass dieser nur für die mobile Ansicht geeignet war, da die Auflösung sehr gering war. Auf Nachfrage wurden anschliessend hochauflösende Bilder zur Verfügung gestellt. Allerdings war hier die Qualität zu gut. Das Laden der Seite verzögerte sich merklich. Aus diesem Grund mussten die Bilder komprimiert und in der Auflösung angepasst werden. Zur Komprimierung wurde das Online Tool von <https://compresspng.com/> genutzt. Die Auflösung wurde mit dem Preview Programm von MacOS angepasst. Dadurch konnte die Grösse eines Bildes von ca. 2MB auf 100KB heruntergebrochen werden. Die Bilder sehen dabei für den Betrachter immer noch scharf aus. Zudem konnten die Bilder so alle auf das selbe Format gebracht werden.

Um die Produkte hinzuzufügen, wurde Postman genutzt. Die einzelnen Produkte-JSONs wurden dabei einmal erstellt und könnten bequem via Runner hinzugefügt werden. Zusätzlich wurde eine Sortier-Funktion implementiert. Mittels dieser kann die Darstellungsreihenfolge angepasst werden. Es wurde hier die ngx Order-Pipe genutzt.

Bei einem Klick auf ein Produkt soll zudem die Detail View von diesem angezeigt werden. Dies war mittels Router Link sehr leicht umzusetzen. Mittels dem ProductService wird das ausgewählte Produkt gesetzt und auf der Detail View angezeigt. Das Design dieser Komponente wird auf später verschoben, da die Warenkorbfunktionalität wichtiger ist.

Produkte dem Warenkorb hinzufügen Für die Realisierung des Warenkorbes gestaltete sich als simpel. Angular liefert hierzu ein hauseigenes Tutorial. Dieses wurde erweiter, sodass CartItems gespeichert werden. Zusätzlich werden diese im Local Storage des Browser gespeichert, sodass der Warenkorb auch nach einem Reload noch vorhanden ist [[cartAngular](#)].

Der Warenkorb wird als Angular Material List dargestellt. Die Artikelanzahl kann mittels Buttons angepasst werden. Ein Löschen wird durch einen zusätzlichen Button ermöglicht. Zusätzlich wird das Gesamttotal der im Warenkorb befindlichen Artikel bei jeder Änderung neu berechnet.



Https bei Backend Moderne Browser erlauben das Laden von sogenanntem Mixed-Content per Default nicht mehr. Mixed Content steht dabei für das Laden von Daten via http von einer https Seite. Um dieses Problem zu beheben, muss das Spring Boot Backend auch via https zugreifbar sein. [Mozilla, 2021a]. Mit dem Tool Certbot kann ein gültiges Zertifikat erstellt werden. Dieses muss nun im application.properties des Spring Projekts angegeben werden. Das Tutorial verwendete dabei eine deprecated Version von Certbot. Dank einem Hinweis von Ubuntu wurde auf die neue Version gewechselt [Beni, 2021].

```
certbot certonly --webroot -d bdaf21-owerlen.enterpriselab.ch
--staple-ocsp -m oliverwerlen@bluewin.ch --agree-tos
```

Es wird empfohlen, in Spring mit PKCS12 Files zu arbeiten. Daher wurde das erstellte PEM File zu einem solchen konvertiert.

Certbot führt die Zertifikaterneuerung von selbst aus. Das jetzt verwendete Zertifikat ist 90 Tage gültig, es sollte daher bis zum Projektende keine Probleme bereiten.

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out
keystore.p12 -name jtipickupbackend -CAfile chain.pem -caname root
```

Sprintreview Sprint 4 Im Sprint 4 konnte die Warenkorbfunktionalität abgeschlossen werden. Zudem läuft die Applikation nun auch im produktiven Umfeld. Um die Bestellung abschliessen zu können, muss im nächsten Schritt die Authentifizierung implementiert werden.

5.2.7 Sprint 5

User Story	Number
Das System bietet dem Anwender die Möglichkeit, sich zu registrieren und anschliessend einzuloggen.	F.3
Das System muss durch einen modernen und sicheren Authentifizierungsmechanismus geschützt sein.	L. 4

Tabelle 4: Userstories Sprint 5,
Quelle: Autor

In vorherigen Sprints konnten die User Stories selten Fertiggestellt werden. Aus diesem Grund wurde hier nur eine kleine Story hinzugenommen.

Authentifizierung

Backend In einem vorherigen Projekt wurde bereits eine Authentifizierung in Spring Boot umgesetzt. Das dort verwendete Tutorial funktionierte dabei tadellos. Aus diesem Grund wurde entschieden, bei diesem Projekt identisch vorzugehen. Nachfolgendes Sequenzdiagramm zeigt dabei auf, wie die einzelnen Teile miteinander kommunizieren. Dabei wurde auf eine Token Based Authentifizierung realisiert. Umgesetzt wurde sie mittels Spring Security. Der Token muss dabei bei jedem Request mitgesendet werden. [bezkoder, 2021]

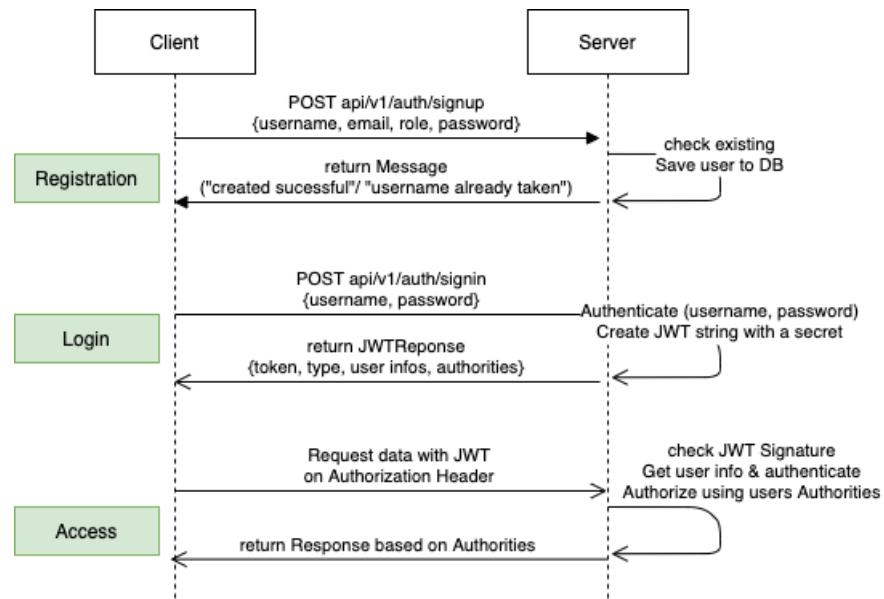


Abbildung 19: Authentication im Backend,
Quelle: Autor

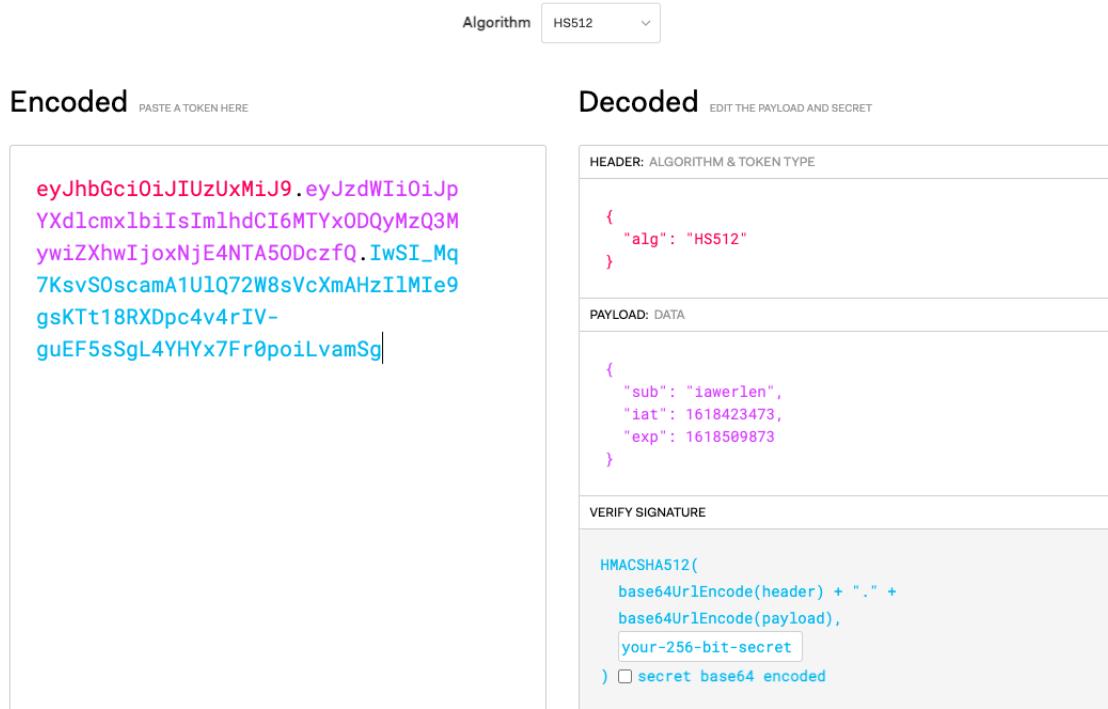


Abbildung 20: Token Auflösung mit jwt.io,
Quelle: Autor

Es wurden zudem Rollen definiert, um die API feingranular sichern zu können. Die Rollen lauten dabei:

- User
- Admin
- Maintenance

Ein entsprechendes Rollenkonzept wird bei Bedarf erstellt und der Dokumentation angefügt.

Frontend Um einen entsprechenden Nutzer zu erstellen bzw. die Eingabe von Logininformationen zu ermöglichen, mussten die entsprechenden Formulare erstellt werden. Dies wurde dabei mittels Reactive Forms umgesetzt. Durch die damit verbundenen Validators wird Clientseitig auf die Komplexität des Passworts, das Übereinstimmen der beiden sowie die Korrektheit der Email Adresse getestet.

The image contains two side-by-side screenshots of a web application interface. Both screenshots feature a header with a menu icon, a logo, and a search bar.

(a) Login Formular, Quelle: Autor

This screenshot shows the login page. It has fields for "Username oder Email-Adresse" (containing "iawerlen") and "Password" (containing "*****"). Below these fields are links for "Passwort vergessen?" and "Anmelden".

(b) Registrierungsformular, Quelle: Autor

This screenshot shows the registration page. It has fields for "Vorname", "Nachname", "Email-Adresse", "Benutzername", "Passwort" (with a note: "Wähle ein Passwort mit mind. 8 Zeichen, Grossbuchstaben und Sonderzeichen"), "Passwort bestätigen", "PLZ", "Ort", "Adresse", and "Land". There is also a checkbox for "Ich habe schon ein Profil" and a "Registrieren" button.

Die Email-Adresse sowie der Nutzernname wird Serverseitig auf die Einzigartigkeit überprüft.

Wie oben erwähnt, muss bei jedem Request der passende Header mitgesendet werden. Um dies ohne grossen Aufwand durchführen zu können, wird analog zum Backend ein passendes Tutorial zum Frontend bereitgestellt. Angular bietet dabei die Möglichkeit, mit Auth-Interceptors zu arbeiten. Dieser fügt jedem API Call den entsprechenden Token im Header an. Der Interceptor dient als Proxy. [bezkoder, 2020]

```

const TOKEN_HEADER_KEY = 'Authorization';           // for Spring Boot back-end

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private token: TokenStorageService) { }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    let authReq = req;
    const token = this.token.getToken();
    if (token != null) {
      authReq = req.clone({ headers: req.headers.set(TOKEN_HEADER_KEY, 'Bearer ' + token) });
    }
    return next.handle(authReq);
  }
}

export const authInterceptorProviders = [
  { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true }
];

```

Abbildung 22: Authentication Interceptor,
Quelle: Autor

Beim erfolgreichen Anmelden wird der Token im Session Storage gespeichert. Zudem wird der Nutzernamen mitgespeichert.

Application		Filter			
Key	Value				
username	iawerlen				
auth-token	eyJhbGciOiJIUzUxM...				
Storage					
Local Storage					
http://localhost:4200					
Session Storage					
http://localhost:4200					
IndexedDB					
Web SQL					
Cookies					

Abbildung 23: Token im Session Storage,
Quelle: Autor

Mittels Angular Direktiven wird das Menu für eingeloggte User angepasst. Es wird der Logout Button eingeblendet. Bei einem Klick auf diesen wird der Session Storage gelöscht.

Automatischer Logout Um die Sicherheit zu erhöhen, wird der Nutzer nach 30 Minuten Inaktivität automatisch ausgeloggt. Für die Umsetzung kam ein das angular-user-idle package zum Einsatz.

```

7      auth.service.ts:29
8      auth.service.ts:29
9      auth.service.ts:29
10     auth.service.ts:29
11     auth.service.ts:29
12     auth.service.ts:29
13     auth.service.ts:29
14     auth.service.ts:29
15     auth.service.ts:29
16     auth.service.ts:29
17     auth.service.ts:29
18     auth.service.ts:29
19     auth.service.ts:29
20     auth.service.ts:29
Time is up!
20     auth.service.ts:32
auth.service.ts:29

```

Abbildung 24: Ablauf von Timer und anschliessender Logout,
Quelle: Autor

Zur besseren Illustrierung wurden die Werte für das Erstellen des Bildes angepasst. Es wurden die Default-Werte beibehalten. Der Timer startet, wenn der Nutzer 10 Minuten inaktiv ist. Anschliessend läuft er 5 Minuten, ehe der Nutzer ausgeloggt wird. [rednez, 2021]

5.2.8 Sprintreview Sprint 5

Im Sprint 5 konnte die geplante User Story erfolgreich abgeschlossen werden.

5.2.9 Sprint 6

User Story	Number
Das System ermöglicht die Anbindung an einen bereits bekannten Bezahlungsdienst, um eine sichere Bezahlung zu garantieren.	F.7

Tabelle 5: Userstories Sprint 6,
Quelle: Autor

Bezahlungsanbieter In der Sitzung E.2 wurde vom Auftraggeber bekannt gegeben, dass sie bei bereits bestehenden Lösungen den Bezahlvorgang mit dem Anbieter Six Payment Services durchführen.

Einführung Six Payment bietet einen Integration Guide an, um die Integration in den eigenen Online Store zu erleichtert. [saferpay, 2021a] Dabei wird eine JSON API zur Verfügung gestellt, um die Bezahlung durchzuführen. Die Implementierung wurde im Sprint 8 5.3.4 angepasst.

Anforderungen Das Vorgehen wurde identisch zum empfohlenen Vorgehen auf der Developer Seite durchgeführt. Dabei wurde zur Consultation die API-Dokumentation durchgelesen. Nachfolgend werden die wichtigsten Requirements aufgeführt.

- JSON API Basic Authentication
- TLS
- mindestens ein aktiver Terminal
- Terminal Nummer und Customer Nummer
- gültiges acceptance agreement für Kreditkarten

[saferpay, 2021b]

Data Security Um falsche Anwendungen und missbrauch von Kreditkarten zu verhindern, wurde von den Kreditkartenorganisationen das Sicherheitsprogramm Payment Card Industry Data Security Standard (PCI DSS) ins Leben gerufen.

Dieser Standard kann dabei erfüllt werden, wenn die Bezahlung auf dem Saferpay Formular durchgeführt wird. Dieses Formular wird direkt beim Anbieter gehostet. Es werden keine Bezahlungsdaten auf dem eigenen Web Server verarbeitet, gesendet oder gespeichert. Somit ist es auch weniger Aufwendig, die PCI DSS merchant certification zu erhalten. [saferpay, 2021b]

Certification Levels Die PCI DSS Zertifikation ist in verschiedene compliance Level aufgeteilt. Die beiden Wichtigsten sind dabei Self Assigned Questionary (SAQ)-A und SAQ-A EP. Nachfolgend wird nur SAQ-A näher anschaut.

Bei SAQ-A wird die Verantwortung vollständig an den Bezahlanbieter abgegeben. Saferpay stellt dabei die Möglichkeit zur Verfügung, die erforderlichen Bedingungen zu erfüllen. Die Bedingungen lauten dabei:

- Nutzen von eigenem HTML Formular Verboten
- Jedes Feld auf der Payment Page muss von einem PCI Zertifizierten Anbieter gehostet werden
- Änderungen an der Bezahlseite via CSS oder JavaScript sind Verboten

Testsystem vs. Live System Saferpay stellt zum Testing einen Testaccount zur Verfügung. Die wesentlichen Unterschiede sind dabei:

- Testsystem und Livesystem sind nicht miteinander verbunden
- Beim Testsystem werden richtige Kreditkarten nicht akzeptiert
- Kein Geld wird beim Testsystem transferiert
- Systemverhalten bleibt identisch
- URLs sehr ähnlich

Auf Livesystem wechseln Um von einem Testsystem auf ein Livesystem zu wechseln, sind einige kleine Änderungen vorzunehmen.

- Authorization Token anpassen
- Anpassen von IDs
- Anpassen der Request URL

Allgemeine Informationen In diesem Projekt wird nur mit dem Testsystem gearbeitet. Ein Wechsel auf ein Livesystem ist nicht geplant.

Integration in Web Applikation In einem ersten Schritt wurde ein Testsystem beim Anbieter angefragt. Um mit der JSON API kommunizieren zu können, sind im Header des Requests einige Parameter mitgegeben werden. Der korrekte Header sieht folgendermassen aus:

```
"Content-Type": "application/json",
"Authorization": "Basic QVBJXzI1Nzc1M183NTU3MTMyMjpKc29uQXBpUHdkMV9Sa1VEQzJzaw==",
```

Der Token wird dabei im Testsystem generiert.

The screenshot shows the 'JSON API basic authentication' section of the back office. On the left, there's a sidebar with options like 'Processing', 'Notifications', 'Payment Means / Terminals', 'Transaction Points Summary', 'JSON API basic authentication' (which is selected), 'JSON API client certificate', 'Saferpay Fields Access Tokens', 'Payment Page configuration', and 'IP permissions'. The main area shows a table of logins:

<input type="checkbox"/> Creation Date	Username	Description
<input type="checkbox"/> 15.04.2021 17:25	API_257753_75571322	Default Test
<input type="checkbox"/> 15.04.2021 16:47	API_257753_99893264	
<input type="checkbox"/> 12.04.2021 11:24	API_257753_11845760	Default Test

At the bottom of the page, there are links for 'Terms of use' and 'Contact', and a note: 'Saferpay is a registered trademark of the SIX Group. Copyright © 2021 SIX Payment Services. All rights reserved.'

Abbildung 25: Erstellen des Basic Authentication Token im Back Office,
Quelle: Autor

Payment Initialization Page An die API muss der entsprechende Request gesendet werden, um den Bezahlvorgang zu starten. Dazu muss der passende Request an die folgende URL gesendet werden:

`https://test.saferpay.com/api/Payment/v1/PaymentPage/Initialize`

In der API-Dokumentation ist dabei ein Demo Request zu finden. Die Parameter lassen sich aus dem Backend herauslesen. Der Body der Anfrage ist in 38b zu finden. Dabei wurde bewusst auf das Erstellen eines entsprechenden TypeScript Interfaces verzichtet. Dies musste jedoch bei der Antwort vorgenommen werden. Es wurde dazu ein Interface passend zur Antwort erstellt. Anschliessend wird die Antwort zum Typen umgewandelt.

Die Anfrage an den Server geschieht dabei asynchron. Somit ist nicht voraussehbar, wann der Request ausgeführt wird, bzw. wann die Antwort gesetzt wird. Jedoch muss in diesem Anwendungsfall auf die URL im Body weitergeleitet werden. Um auf die Antwort zu warten, wurde das selbe Vorgehen wie in 34 verwendet. Wie bereits beschrieben, wird der User auf die Zahlungsseite von Saferpay weitergeleitet. Nach dem Abschluss der Bezahlung wird entweder auf die Success-URL oder die "Fail-URL" von obigem Request umgeleitet.

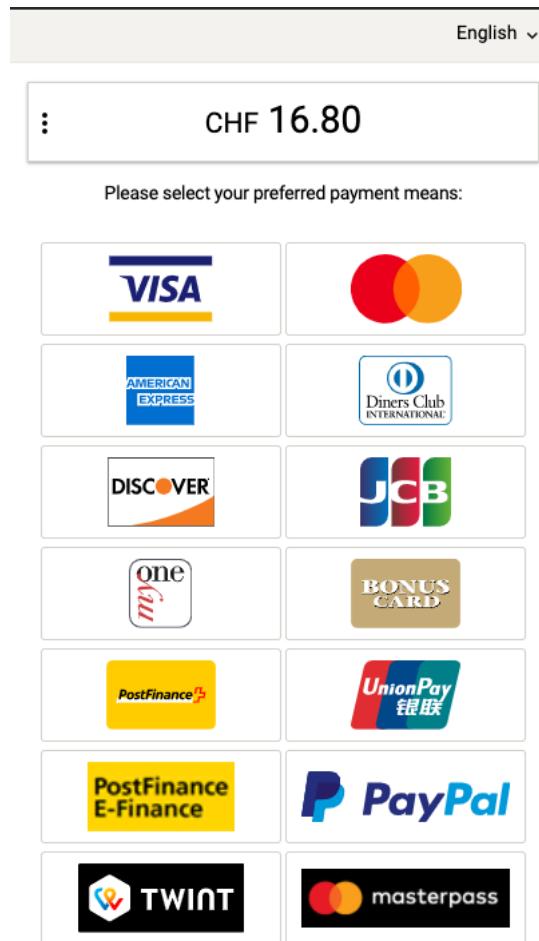


Abbildung 26: Saferpay Bezahlseite,
Quelle: Autor

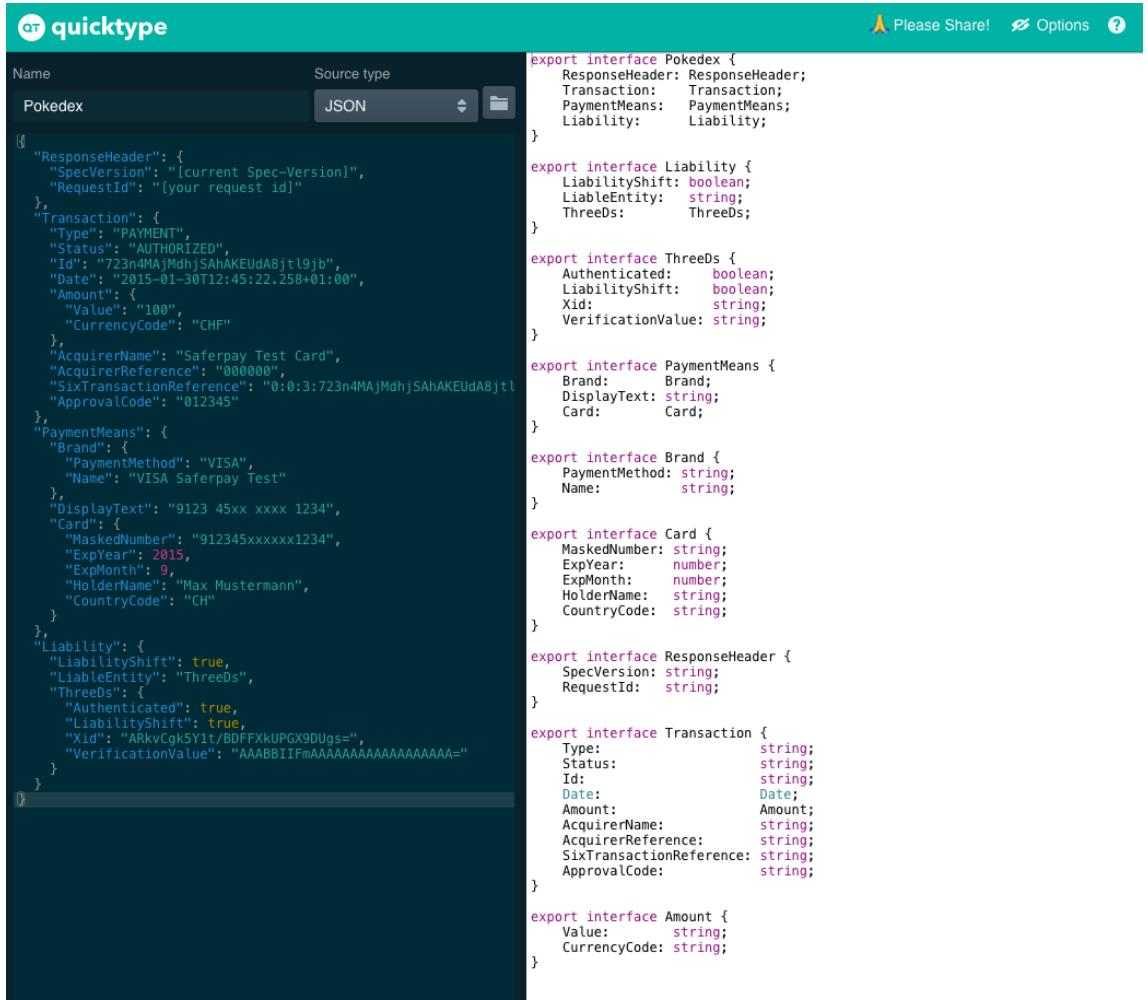
Payment Assert Das Vorgehen zur Bezahlverifikation ist dabei identisch zur Zahlungslösung.

<https://test.saferpay.com/api/Payment/v1/PaymentPage/Assert>

Um die Bezahlung identifizieren zu können, muss der Token aus vorheriger Antwort mitgegeben werden. Dieser wird im Session Storage gespeichert. Der genaue Request sieht folgendermassen aus:

```
let request = {
  "RequestHeader": {
    "SpecVersion": 1.21,
    "CustomerId": this.customerId,
    "RequestId": this.requestId,
    "RetryIndicator": 0
  },
  "Token": this.getToken()
}
```

Um das Interface für die Antwort zu definieren, wurde quicktype genutzt. Damit konnte aus dem Demorequest in der Dokumentation das passende Interface generiert werden.



```

export interface Pokedex {
  ResponseHeader: ResponseHeader;
  Transaction: Transaction;
  PaymentMeans: PaymentMeans;
  Liability: Liability;
}

export interface Liability {
  LiabilityShift: boolean;
  LiableEntity: string;
  ThreeDs: ThreeDs;
}

export interface ThreeDs {
  Authenticated: boolean;
  LiabilityShift: boolean;
  Xid: string;
  VerificationValue: string;
}

export interface PaymentMeans {
  Brand: Brand;
  DisplayText: string;
  Card: Card;
}

export interface Brand {
  PaymentMethod: string;
  Name: string;
}

export interface Card {
  MaskedNumber: string;
  ExpYear: number;
  ExpMonth: number;
  HolderName: string;
  CountryCode: string;
}

export interface ResponseHeader {
  SpecVersion: string;
  RequestId: string;
}

export interface Transaction {
  Type: string;
  Status: string;
  Id: string;
  Date: Date;
  Amount: Amount;
  AcquirerName: string;
  AcquirerReference: string;
  SixTransactionReference: string;
  ApprovalCode: string;
}

export interface Amount {
  Value: string;
  CurrencyCode: string;
}

```

Abbildung 27: PagePaymentAssertResponse,
Quelle: Autor

Es wird von diesem Request das Status Property gecheckt. Bei AUTHORIZED oder CAPTURED wird die Bezahlung als Erfolgreich markiert. Zur Identifikation wird die Request-Id genutzt. Durch diese kann die Bezahlung im Backend eindeutig identifiziert werden.

The screenshot shows the SIX Payment Services Backoffice interface. At the top, there is a navigation bar with links for 'Batch Processing', 'Transactions' (which is highlighted in red), 'Risk & Fraud', 'Secure Card Data', 'Secure PayGate', 'Settings', and 'Online Support'. To the right of the navigation bar are buttons for 'Test Environment', user information ('Oliver Werlen (e257753002)'), and 'Logout'. Below the navigation bar is a red header bar with a dropdown menu for 'English'.

The main content area is titled 'Journal' and shows a list of transactions from the last 30 days. The table has columns for 'Authorization Date', 'State', 'Amount', 'Reference number', 'Payment means', 'Payment means details', 'Terminal', and 'Account'. The transactions listed are:

Authorization Date	State	Amount	Reference number	Payment means	Payment means details	Terminal	Account
19.04.2021 16:51	Authorization (Debit)	CHF 16.80	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797
15.04.2021 13:09	Authorization (Debit)	CHF 50.40	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797
14.04.2021 19:27	Authorization (Debit)	CHF 33.60	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797
14.04.2021 19:12	Authorization (Debit)	CHF 33.60	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797
14.04.2021 19:04	Authorization (Debit)	CHF 33.60	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797
14.04.2021 19:02	Authorization (Debit)	CHF 33.60	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797
14.04.2021 18:59	Authorization (Debit)	CHF 33.60	1	MasterCard	xxxx xxxx xxxx 0006 (US)	<input checked="" type="checkbox"/>	✓ 17731797

Below the table, it says 'Show 7 of 7 transactions'. There are also filter options: 'Transaction pending', 'Partial capture', 'Transaction completed', 'Transaction discarded', and '3-D Secure with liability shift'.

At the bottom of the page, there are links for 'Terms of use' and 'Contact'. A note states: 'Saferpay is a registered trademark of the SIX Group. Copyright © 2021 SIX Payment Services. All rights reserved.'

Abbildung 28: Bezahlhistorie Backoffice,
Quelle: Autor

Der Bezahlprozess ist somit abgeschlossen.

Ablauf Der Bezahlvorgang wird nachfolgend als Sequenzdiagramm dargestellt. Aus Einfachheitsgründen wurde auf die Darstellung des Proxies verzichtet.

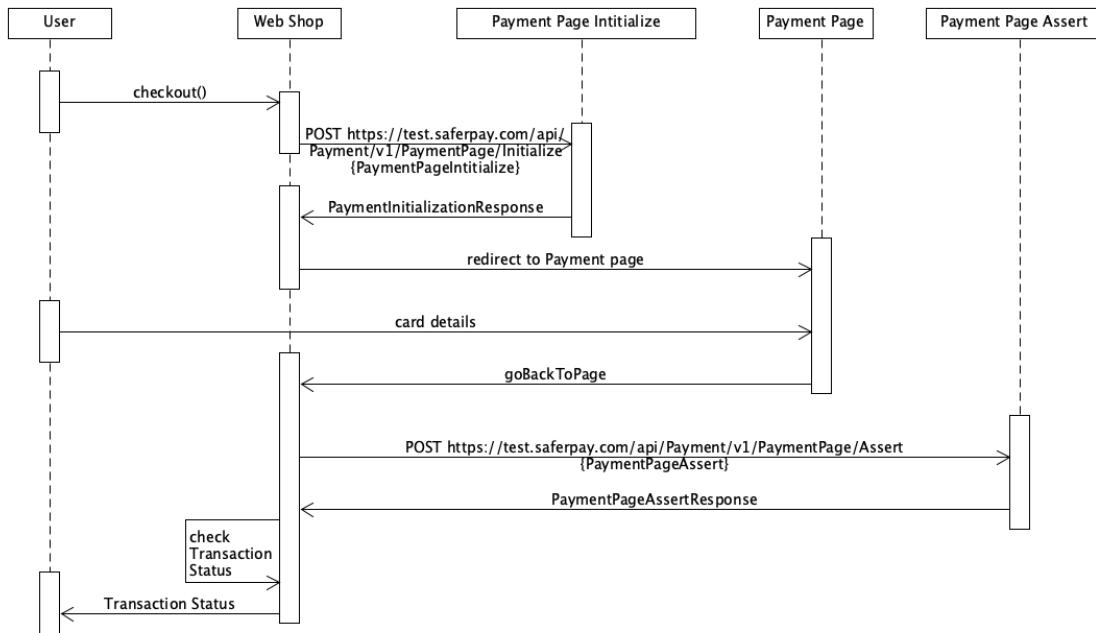


Abbildung 29: Sequenzdiagramm vom Bezahlvorgang,
Quelle: Autor

CORS "Cross-Origin Resource Sharing (CORS) ist ein Mechanismus, der zusätzliche HTTP Header verwendet, um einem Browser mitzuteilen, dass er einer Webanwendung, die auf einer anderen Domain (Origin) läuft, die Berechtigung erteilt, auf ausgewählte Ressourcen von einem Server eines anderen Ursprungs (Origin) zuzugreifen. Eine Webanwendung stellt eine cross-origin HTTP-Anfrage, wenn sie eine Ressource anfordert, die einen anderen Ursprung (Domain, Protokoll und Port) hat, als ihren eigenen." [mozilla, 2021]

Dies wurde auch bei diesem Projekt bemerkt. Die Requests an die API von Six wurden immer von Cross-Origin Resource Sharing (CORS) geblockt.

Setzen von Access Header Wenn man Zugriff auf den betroffenen Server hat, so kann hier mittels entsprechenden CORS Headern gearbeitet werden. Dies wurde beispielsweise bei dem eigenen Backend umgesetzt. Da es sich um eine externe API handelt und dementsprechend kein Zugriff auf die Maschine besteht, ist dies keine mögliche Lösung.

Angular Proxy Angular bietet einen eigenen, integrierten Proxy um CORS zu umgehen. Dabei werden Requests mit einer bestimmten Adresse abgefangen und an den passenden Server umgeleitet.

In diesem Anwendungsfall funktionierte der Angular Proxy nicht wunschgemäß. Zudem eignet sich dieser nur im Entwicklungseinsatz.

Zusätzlicher Proxy auf localhost Es wurde daher ein zusätzlicher Proxy umgesetzt. Dabei wurde ein NodeJS-Server mit express genutzt. Dieser besitzt keine Logik. Er nimmt lediglich Requests an und leitet diese an die Adresse im Header weiter. [Patel, 2020] Die Requests werden nun via Proxy an die API gesendet. Zudem wird nun auch mit der eigenen API über diesen Proxy kommuniziert. Dies löst einerseits die CORS-Problematik mit externen APIs und macht den Einsatz von HTTPS im Backend überflüssig. Die Transport Layer Security (TLS) wurde hier nur aufgrund der Browser Policies gebraucht. Der Proxy ist nun via localhost erreichbar. Hierbei ist die "Mixed Content" Policy nicht aktiv. Bei der Entwicklung bleibt das Backend direkt auch erreichbar. Im produktiven Betrieb würden nur noch Anfragen vom Proxy akzeptiert.

Zudem ist die Verschlüsselung der Kommunikation zwischen Backend und Frontend überflüssig, da sie nur auf der eigenen Maschine abläuft. Es werden keine Daten über eine unsichere Leitung gesendet.

Die Verbindung zur externen API läuft dabei via HTTPS. Dies wird von Six umgesetzt.

Um eine konstante CI/CD zu ermöglichen, wurde dies für den Proxy umgesetzt. Angedacht war dabei, dass der Proxy auch im produktiven Umfeld via loopback-Adresse erreichbar ist. Dies bietet den Vorteil, dass die Browser diesem trauen und nicht via HTTPS zugreifbar gemacht werden muss. Der Traffic bleibt nur lokal auf der Maschine, daher bestehen keine Sicherheitsrisiken. Auf Google Chrome sowie auch Firefox funktionierte dies auch. Auf Safari ist die Loopback Adresse nicht vertrauenswürdig, der Proxy müsste via HTTPS erreichbar sein. Das Problem ist, dass Letsencrypt keine Zertifikate für localhost ausstellt [Letsencrypt, 2017].

Es besteht keine Möglichkeit, dieses Problem unter Safari zu beheben. Daher musste dieser Lösungsansatz verworfen werden. Das Problem wird auf den nächsten Sprint verschoben.

Backend als Proxy Dieses Vorgehen wird in Sprint 8 5.3.4 genauer beschrieben.

Problem mit Enterpriselab Host Am Mittwoch, 14.04. am Nachmittag war der GitLab Host nicht mehr erreichbar. Die Nachfrage beim Enterpriselab Team ergab, dass die Maschine ihre IP-Adresse verloren hatte. Ein Neustart brachte auch keinen Erfolg, da verschiedene wichtige Dienste nicht mehr starteten. Es wurde eine neue Maschine bereitgestellt. Der Grund konnte nicht herausgefunden werden.

Die Umgebung konnte sehr schnell wieder hergestellt werden. Es musste nur Docker und GitLab-Runner installiert und registriert werden. Zudem mussten die Daten neu migriert werden. Der Umfang der gesamten Neuumsetzung lag bei ca. 3h.

Backend Models Die Bestellungen sollen im Backend persistiert werden. Bei der Erstellung der Order-Entität fiel dabei auf, dass diese diverse Beziehungen zu anderen Klassen hat. Dadurch wurde entschieden, dass diese auch implementiert werden.

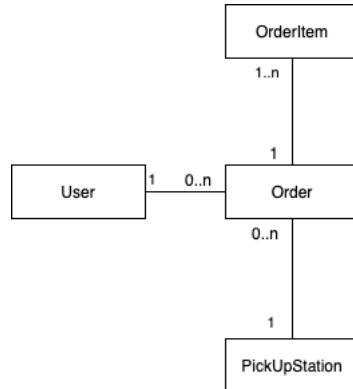


Abbildung 30: Entity Relationship Diagram von Order,
Quelle: Autor

Bei der Implementierung wurde strikt darauf geachtet, dass die gesamte API dem REST-Level 3 nach Richardson Maturity entspricht. Entscheidend dafür ist der Einsatz von **HATEAOS!** (**HATEAOS!**).

```

1
2   {
3     "pickUpStation_id": 1026,
4     "name": "Home station Oliver",
5     "latitude": "46.3983041",
6     "longitude": "7.7512328",
7     "links": [
8       {
9         "rel": "inventories",
10        "href": "http://localhost:8080/api/v1/inventories/pickUpStation/1026"
11      }
12    ],
13

```

Abbildung 31: Response bei Abfrage einer PickUp-Station,
Quelle: Autor

Durch den Einsatz von Model Mapper wurde das Mapping von DTO und Datenbankobjekt vereinfacht 3.5.

5.2.10 Meilenstein Abschluss Bestellprozess und Zwischenpräsentation

Nachfolgend werden die Meilensteinberichte zu den Meilensteinen 3 und 4 zusammen aufgeführt.

Meilensteinbericht

Termin Meilenstein 3 Der Meilenstein 3 ist am 19.04.2021 abgeschlossen und somit pünktlich fertiggestellt worden. Der Meilenstein 4 ist am 21.04.2021 abgeschlossen und somit pünktlich fertiggestellt worden.

Beschreibung Meilenstein 3 Die Beschreibung des Meilensteins ist im Abschnitt B.2.2 ersichtlich.

Meilensteinziele/Vorgaben Das übergeordnete Ziel dieses Meilensteins ist die der Abschluss des Bestellprozesses.

- Testprotokolle
- Demo
- Release
- Zwischenpräsentation

Die Zwischenpräsentation ist im Anhang H zu finden. Die Testprotokolle sind im Anhang A.1 zu finden.

Meilensteinzielerreichung Die Meilensteine konnten grösstenteils erfüllt werden. Einzig die Altersverifikation mittels Jumio wurde nach hinten verschoben. Die hing jedoch mit Abhängigkeiten zum Auftraggeber zusammen. Jedoch ist das Backend bereits sehr weit fortgeschritten. Die Zwischenpräsentation wurde erfolgreich durchgeführt.

Fazit Es konnten viele der geplanten Artefakte abgeliefert werden. Die fehlende Integration von Jumio wurde durch den Fortschritt im Backend kompensiert. Zudem funktioniert die Bezahlung aufgrund von CORS-Problemen nicht auf allen modernen Browern. Hier muss in den kommenden Sprints nachgebessert werden.

5.3 Realisierungsphase

5.3.1 Sprint 7

User Story	Number
Das System bietet dem Kunden die Möglichkeit, alle vorhandenen Pick-Up Stations anzuzeigen.	F.8
Das System bietet dem Kunden die Möglichkeit, die für ihn nächstgelegene Station auswählen zu können.	F.7

Tabelle 6: Userstories Sprint 7,

Quelle: Autor

Anzeigen von Pick-Up Stations

Google Maps Google Maps gilt als der Standard bei digitalen Karten. Seit einigen Jahren verfolgt Google jedoch ein sehr undurchsichtiges Bezahlmodell. Dem Kunden werden 200 Dollar Kredit pro Monat kostenlos zur Verfügung gestellt. Eine weitere Nutzung würde in diesem Anwendungsfall mit 2 Dollar je 1000 Anfragen belastet.

Für den Entscheid gegen Google Maps ausschlaggebend war der Zwang, eine Kreditkarte zu hinterlegen.

Geoapify Als Alternative zu Google Maps wird Geoapify angeboten. Es bietet sehr ähnliche Funktionen, jedoch ohne Zwang, eine Kreditkarte zu hinterlegen. Bei Geoapify sind 3000 Requests pro Tag kostenlos.

Leaflet Leaflet ist eine Open Source native Java Script Library für benutzerfreundliche, interaktive mobile Maps. [Agafonkin, 2020]

OpenLayers Open Layers dient als Alternative zu Leaflet. Dabei ist OpenLayers für komplexere Applikationen ausgelegt. [Spepanov, 2020] Für dieses Projekt wird jedoch nur ein begrenzter Umfang gebraucht, daher wird mit Leaflet gearbeitet.

Open Street Map Als Karte wird dabei Open Street Map genutzt. Open Street Map untersteht dabei einer freien Lizenz. Die Daten werden dabei hauptsächlich von der Community gepflegt. Die Nutzung bleibt dabei kostenlos. Die Open Street Map Contributors müssen auf der Applikation ersichtlich sein. [OpenStreetMap, 2021]

Anzeigen einer Map mit Leaflet Geoapify stellt hierzu einige Tutorials zur Verfügung, um den Einstieg zu vereinfachen. Dabei wird eine einfache Karte erstellt. [Geoapify, 2020]

Marker und Popup Funktionalität Die einzelnen Marker werden aus den Daten vom Backend erstellt. Diese werden via Request vom Backend geladen und angezeigt. Zudem werden die entsprechenden Inventories zu jeder Station geladen. Da hier mit HATEOAS gearbeitet wurde, mussten die Produkte separat geladen werden.

Die Produktverfügbarkeit an einer Station sollte dabei als Popup dargestellt werden. Um dies für den Nutzer übersichtlich zu gestalten, wird die Verfügbarkeit gefärbte Kreise dargestellt. Beim Anzeigen der Popups ergab sich jedoch die Problematik, dass dies auf Geräten aus dem Hause Apple nicht funktionierte. Dabei wurde der Event von Leaflet nicht korrekt erkannt. Nach einigen Recherchen konnte keine Lösung gefunden werden. Ähnliche Fälle sind dabei nicht bekannt.

In der Dokumentation von Leaflet ist jedoch eine Alternative zu den Popups zu finden. Diese Tooltips werden dabei zum Anzeigen von kleinen Texten auf der Karte genutzt. Für die geplante Funktion sind diese somit geeignet.

```
private addMarker(pickUpStation: PickUpStation){
    var marker = L.marker([parseFloat(pickUpStation.latitude), parseFloat(pickUpStation.longitude)], {
        icon: this.markerIcon
    }).bindTooltip(this.inventoryToString(pickUpStation.inventory!)).addTo(this.map!)
    marker.on('click', this.setCurrentItem, [])
}
```

Abbildung 32: Erstellung von Marker und Tooltip,
Quelle: Autor

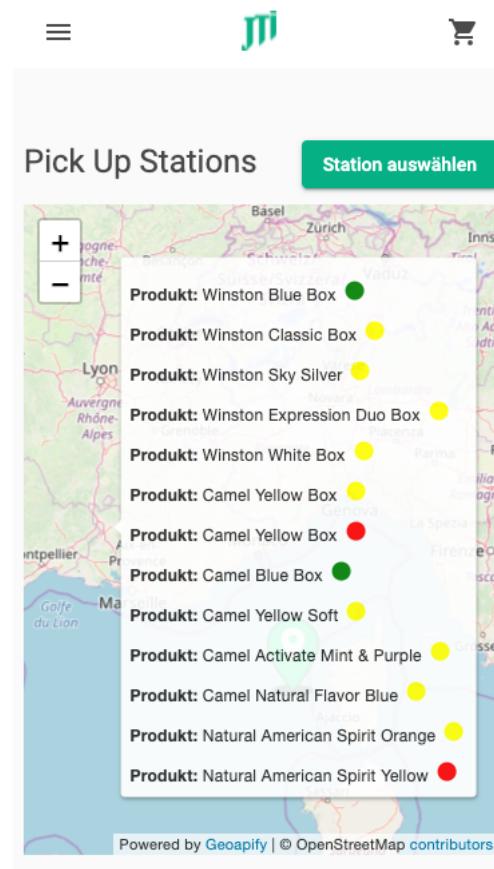


Abbildung 33: Darstellung von Marker und Tooltip,
Quelle: Autor

Ein Problem war dabei, dass die Get-Requests von RxJS asynchron ausgeführt werden. Dies bietet einerseits den Vorteil, dass die Applikation während dem Laden von Daten nutzbar bleibt, andererseits ist in diesem Fall essentiell, dass einige Prozesse aufeinander warten. Das Vorgehen wird dabei nachfolgend kurz chronologisch dargestellt:

1. Laden von aktueller Location des Benutzers
2. Erstellen der Map inkl. Kontrollelemente
3. Laden der PickUp-Stations vom Server
4. Laden des Inventars vom Server pro PickUp
5. Laden von einzelnen Produkten von Inventar

6. Hinzufügen von Marker auf die Map

Wie hier zu sehen, sind die Daten von einander Abhängig. So kann das Inventar nicht geladen werden, ohne dass die PickUp Station vorhanden ist und so weiter. Es muss immer auf das Resultat gewartet werden. Um genau dies zu ermöglichen, triggered RxJS den State des Observables. Dabei gibt es drei wesentliche Zustände: next, completed und error.



Abbildung 34: Observable State,
Quelle: Autor

Next liefert dabei einen Wert, Error einen JavaScript Error oder eine Exception und completed nichts zurück. Error wird bei den hier genutzten Abfragen immer direkt mittels pipe abgefangen. Mit diesem Vorgehen kann auf das vorhergehende Resultat gewartet werden. Dies wird sehr viel genutzt.

5.3.2 Auswahl der PickUp Station

In Abbildung 34 ist zu sehen, dass beim Öffnen des Tooltips ein click-Event abgefangen wird. Dabei wird ein Leaflet Event mitgegeben, welcher unter anderem auch die Koordinaten der geklickten Station enthält. Basierend auf diesen Daten wird die ausgewählte PickUp Station gesetzt. Diese wird im Local Storage des Browsers gespeichert.

```
select-pickup.component.ts:123
{
  originalEvent: MouseEvent, containerPoint: Point, layerPoint: Point, latlng: LatLng, type: "click" 
  > containerPoint: Point {x: 188.01622464364036, y: 429.059036}
  > latlng: LatLng {lat: 42.29698, lng: 7.885466}
  > layerPoint: Point {x: 195, LatLng}
  > originalEvent: MouseEvent {isTrusted: false, _simulated: true}
  > sourceTarget: NewClass {options: {}, _latlng: LatLng, _initHooks: {}, _target: NewClass {options: {}, _latlng: LatLng, _initHooks: {}}, type: "click"}
  > __proto__: Object
}
```

Abbildung 35: Mouse Event,
Quelle: Autor

5.3.3 Sprintreview Sprint 7

Die User Story konnte erfolgreich durchgeführt werden.

5.3.4 Sprint 8

User Story	Number
Das System ermöglicht die Anbindung an einen bereits bekannten Bezahlidienst, um eine sichere Bezahlung zu garantieren.	F.9
Das System bietet die Möglichkeit, eine Bestellung dauerhaft zu speichern	F.7
Das System bietet dem Kunden die Möglichkeit, eine Bestellung durch das Einlesen eines Codes an der Pick-Up Station abzuholen.	F.16

Tabelle 7: Userstories Sprint 8,
Quelle: Autor

In diesem Sprint wird das CORS-Problem von Kapitel 5.2.9 neu angegangen.

Persistierung der Bestellung In einem vorhergehenden Sprint wurden die entsprechenden Methoden und Entitäten für die Erstellung von Orders bereits erstellt. Das Entity Relationship Diagramm 30 liefert dabei einen Überblick.

Die Order hat dabei Beziehungen zu den folgenden Entities:

- User
- PickUpStation
- OrderItem

Dabei wird auch hier mit DTOs gearbeitet. Das Vorgehen ist in 3.5 beschrieben.

Die Order wird Erstellt, sobald der Benutzer auf den Checkout-Button im Warenkorb klickt. Es wird dabei überprüft, ob eine PickUp-Station ausgewählt wurde und der Warenkorb nicht leer ist. Nach dem erfolgreichen Hinzufügen der Order 36 werden die Order-Items erstellt. Hierzu wird pro Item im Warenkorb ein POST-Request durchgeführt 37.

```
this.orderService.createOrder(this.currentPickUp!.pickUpStation_id).
  subscribe(order =>
    {this.order = order, this.addOrderItems()})
```

Abbildung 36: Erstellen der Order,
Quelle: Autor

```
addOrderItems(){
  this.cartItems.forEach(cartItem =>
    this.orderService.createOrderItems(cartItem.quantity, cartItem.product?.product_id, this.order!.order_id)
      .subscribe(orderItem =>
        {this.orderItems.push(orderItem), this.addOrderInitialization()})
  )
}
```

Abbildung 37: Erstellen der OrderItems,
Quelle: Autor

Der Benutzer wird dabei aus dem Authentifizierungstoken im Backend ausgelesen. Nach dem diese Abfragen completed sind, wird der Bezahlvorgang ausgelöst.

Behebung des CORS-Problems Um CORS zu umgehen, wurde ein eigener NodeJS-Proxy implementiert und auf die virtuelle Maschine im GitLab deployed. Wie bereits in 5.2.9 beschrieben, funktioniert dies unter Safari nicht. Der Ansatz mit einem eigenen Proxy wurde verworfen. Dies hing damit zusammen, dass der Reverse-Proxy 5.2.3 die Anfragen an den Proxy nicht korrekt weiterleitete. Das Problem konnte nicht identifiziert werden.

Es wurde nach einem neuen Ansatz gesucht, wobei die Wahl auf die Nutzung des eigenen Backends als Proxy fiel. Dabei übernimmt das Backend noch weitere Aufgaben bei der Bezahlung. So wird der finale Request erst im Backend erstellt. Vom Frontend kommt ein leeres Payment Initialize Objekt. In diesem sind nur die Order-Id sowie die redirect-URLs gesetzt. Es wird im Backend aus dem gesendeten leeren JSON ein entsprechendes Objekt erstellt und die fehlenden Daten gesetzt. Das Erstellen der Klasse wurde dabei analog zu 27 mittels quicktype durchgeführt. Allerdings sind die einzelnen JSON-Properties bei der Saferpay API grossgeschrieben, was nicht mit der Java-Namenskonvention übereinstimmt. Es handelt sich hierbei jedoch nur um DTOs, daher wurde dies hier vernachlässigt.

```

let paymentInitialize1 = {
  "RequestHeader": {
    "SpecVersion": 1.21,
    "CustomerId": this.customerId,
    "RequestId": this.requestId,
    "RetryIndicator": 0
  },
  "TerminalId": this.terminalId,
  "Payment": [
    "Amount": {
      "Value": amount*100,
      "CurrencyCode": "CHF"
    },
    "OrderId": this.getOrderId(),
    "Description": "Description of payment"
  ],
  "ReturnUrls": {
    "Success": environment.paymentSucessfullRedirect,
    "Fail": environment.paymentNotSucessfullRedirect
  }
}

let paymentInitialize = {
  "RequestHeader": {
    "SpecVersion": "",
    "CustomerId": "",
    "RequestId": this.getOrderId(),
    "RetryIndicator": 0
  },
  "TerminalId": "",
  "Payment": {
    "Amount": {
      "Value": "",
      "CurrencyCode": "CHF"
    },
    "OrderId": this.getOrderId(),
    "Description": "Description of payment"
  },
  "ReturnUrls": {
    "Success": environment.paymentSucessfullRedirect,
    "Fail": environment.paymentNotSucessfullRedirect
  }
}

```

(a) Payment Initialization alt,
Quelle: Autor

(b) Payment Initialization neu,
Quelle: Autor

Ausschlaggebend bei diesem Request ist die Order-Id. Diese wird bei der Persistierung der Order gespeichert und anschliessend im Frontend im Local Storage gespeichert. Sie wird auch als Request-Id genutzt. Diese ist im gesamten System einzigartig. Die Bezahlung kann im Backoffice über diese Id eindeutig identifiziert werden.

Setzen der fehlenden Properties Wie in der Abbildung 38b zu sehen ist, beinhaltet der Request nur die Order-Id. Die restlichen Daten werden im Backend gesetzt. Dabei werden die Werte aus dem Properties-File gelesen und zur Laufzeit den Variablen zugewiesen.

```

@Value("${pickupbackend.app.saferpay}")
private String BASIC_AUTH_TOKEN_SAVERPAY;

@Value("${pickupbackend.app.saferpay.customerid}")
private String CUSTOMER_ID_SAVERPAY;

@Value("${pickupbackend.app.saferpay.terminalid}")
private String TERMINAL_ID_SAVERPAY;
@Value("${pickupbackend.app.saferpay.specVersion}")
private String SPEC_VERSION;

```

Abbildung 39: Auslesen der Werte aus .properties-File,
Quelle: Autor

Der Bestellwert wird ebenfalls im Backend gesetzt. Dies macht es unmöglich, dem Bestellwert im Frontend anzupassen und so eine Vergünstigung zu erhalten. Um den Wert zu bestimmen, werden alle OrderItems die zur Order gehören in der Datenbank gesucht und der Produktpreis mit der Anzahl Produkte summiert. Dank streams in Java kann dies sehr elegant durchgeführt werden. Java empfiehlt den Gebrauch von Atomic Datentypen, wenn diese mit Streams bearbeitet werden.

```

public String calculateOrderTotal(Integer order_id){
  AtomicReference<Double> total = new AtomicReference<>( initialValue: 0.0);
  this.orderItemService.getOrderItemByOrder(this.getOrder(order_id)).forEach(orderItem -> total.updateAndGet(v -> v + (orderItem.getProduct().getPrice()*100) * orderItem.getQuantity()));
  return new DecimalFormat( pattern: "#.####").format(total.get());
}

```

Abbildung 40: Berechnung des Order-Totals,
Quelle: Autor

Zudem schreibt die API von Sixpayment vor, dass der Wert ohne Kommastellen angegeben wird.

Dies wird hier durch die Multiplikation mit dem Faktor 100 erreicht. Durch das DecimalFormat werden die Nachkommastellen entfernt.

Senden eines POST-Requests von Spring Um einen POST-Request von Spring an eine andere API zu senden, wurde RestTemplate genutzt. [baeldung, 2021] Dazu wurde eine HttpEntity mit dem gewünschten Body und dem passenden Header erstellt. Das Erstellen dieser wurde in einen Service 42 ausgelagert.

```
public @ResponseBody PaymentInitializationResponse initializePayment(@RequestBody PaymentPageInitialize paymentPageInitialize) {
    final String uri = "https://test.saferpay.com/api/Payment/v1/PaymentPage/Initialize";
    RestTemplate restTemplate = new RestTemplate();
    PaymentInitializationResponse response = restTemplate.postForObject(uri, this.paymentService.setPaymentInitialization(paymentPageInitialize), PaymentInitializationResponse.class);
    return response;
}
```

Abbildung 41: Senden eines Request mit Rest Template,
Quelle: Autor

```
public HttpEntity<PaymentPageInitialize> setPaymentInitialization(PaymentPageInitialize paymentPageInitialize){
    paymentPageInitialize.getRequestHeader().setSpecVersion(SPEC_VERSION);
    paymentPageInitialize.getRequestHeader().setCustomerId(CUSTOMER_ID_SAFERPAY);
    paymentPageInitialize.setTerminalId(TERMINAL_ID_SAFERPAY);
    paymentPageInitialize.getPayment().getAmount().setValue(this.orderService.calculateOrderTotal(Integer.parseInt(paymentPageInitialize.getPayment().getOrderId())));
    return new HttpEntity<>(paymentPageInitialize, this.getHeader());
}
```

Abbildung 42: Erstellen der HttpEntity,
Quelle: Autor

Die Antwort von diesem Request wird vom Backend direkt weitergeleitet. Im Frontend wird der Token im Local Storage gespeichert. Dieser wird beim Payment Assert gebraucht.

Im nächsten Schritt wird das PaymentAssert durchgeführt. Wie auch bereits in obigen Beispiel gesehen, werden auch hier die meisten Daten im Backend gesetzt. Hier wird der Token sowie die Request-Id aus dem Local Storage mitgegeben.

Als Antwort auf diesen Request wird der Statuscode zurückgegeben. Zudem wird die Order in der Datenbank als Bezahlt markiert.

```
if(response.getTransaction().getStatus().equals("AUTHORIZED") || response.getTransaction().getStatus().equals("CAPTURED")){
    this.orderService.markAsPaid(Integer.parseInt(paymentAssertRequest.getRequestHeader().getRequestId()));
    return new ResponseEntity<>(
        HttpStatus.OK
    );
} else {
    return new ResponseEntity<>(
        HttpStatus.BAD_REQUEST
    );
}
```

Abbildung 43: Überprüfung des Zahlungsstatus,
Quelle: Autor

Dieser wird im Frontend überprüft, eine entsprechende Meldung ausgegeben und ein Redirect auf das Nutzerprofil durchgeführt. Der Bezahlvorgang ist abgeschlossen.

Abholung einer Bestellung

Erstellen eines QR-Codes Bereits in Sprint 1 5.2.1 wurde spezifiziert, dass zur Abholung ein QR-Code eingelesen werden muss. Um dies umzusetzen, wurde in einem ersten Schritt ein PickUp-Identifier der PickUp-Station-Entity hinzugefügt. Hierbei handelt es sich um einen 40-Stelligen String, welcher bei der Erstellung generiert wird.

Dieser String wird mit der Java Library ZXing in einen QR-Code verpackt. Bei ZXing handelt es

sich um die Hauptlibrary, welche bei QR-Codes in Java zum Einsatz kommt. Anschliessend wird der Code als Buffered Image zurückgegeben. [baeldung, 2020]

```
public BufferedImage generateQRCodeImage(Integer pickUpStation_id) throws Exception {
    QRCodeWriter qrCodeWriter = new QRCodeWriter();
    BitMatrix bitMatrix = qrCodeWriter.encode(this.getOne(pickUpStation_id).getPickUpStationIdentifier(),
        BarcodeFormat.QR_CODE, width: 300, height: 150);

    return MatrixToImageWriter.toBufferedImage(bitMatrix);
}
```

Abbildung 44: Erstellen eines QR-Code mit ZXing,
Quelle: Autor



Abbildung 45: QR-Code Beispiel,
Quelle: Autor

Dieser Code wird beim finalen Produkt auf der Station positioniert, sodass er gut einlesbar ist.

Lesen des QR-Codes mit Frontend Für das Lesen des QR-Codes wurde die reine JavaScript Library jsQR eingesetzt. [cozmo, 2021] Dabei wurde auf eine Implementierung auf Stackblitz zurückgegriffen und diese entsprechend angepasst. [Yamaguchi, 2020]

Abholmeldung an Backend Wenn ein QR-Code gelesen wird, sendet rxjs einen Call mit dem PickUp Identifier und dem Order-PickUp Token an die API.

```
let pickUp = {
  "pickUpToken": JSON.parse(localStorage.getItem('order') || '{}').pickUpToken,
  "pickUpStationIdentifier": stationToken
}
```

Abbildung 46: PickUp Station Identifier und PickUp Token,
Quelle: Autor

Im Backend kann mittels diesen beiden Tokens einerseits die PickUp Station und auch die Bestellung eindeutig identifiziert werden. Es wird nun analog zu 5.3.4 ein Request an die PickUp Station gesendet. Von dieser ist die entsprechende URL in der Datenbank vorhanden. Gesendet werden dabei die einzelnen Order Items.

Node Server auf Station Auf der PickUp Station läuft ein Node Server. Dieser hat dabei zwei Funktionen:

1. Senden von initialer Nachricht an Backend, wenn Station gestartet wurde
2. Empfangen von Bestellausgabeanforderungen vom Backend

Um dies umzusetzen und den Aufwand im Rahmen zu halten, wurde ein Node Server aufgesetzt. Die Kommunikation mit der Elektrotechnik wurde dabei noch nicht umgesetzt.

Bei jedem Neustart der Station wird dabei ein POST-Request durchgeführt und das momentane Inventar gesendet. Dabei handelt es sich um Updates, sofern die Daten bereits vorhanden waren oder sie werden erstellt. Dies kommt besonders beim Inventar sehr oft vor.

Um die Ausgabeanforderungen empfangen zu können, wurden zudem ein POST-Endpoint erstellt.

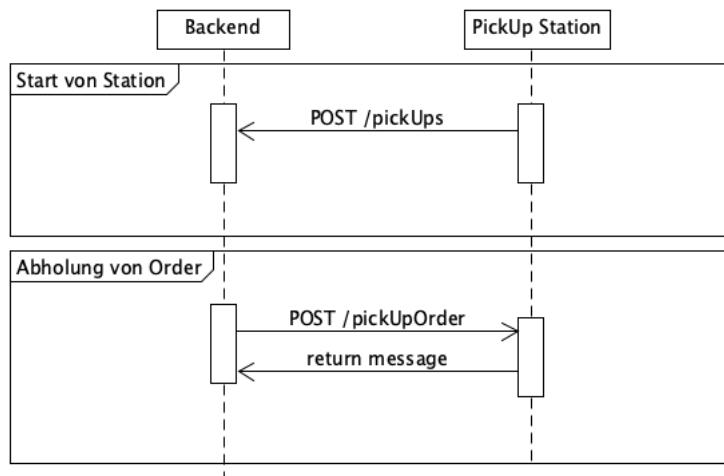


Abbildung 47: Kommunikation zwischen Station und Backend,
Quelle: Autor

Abschliessende Bemerkungen Es besteht keine durchgängige Verbindung zwischen Station und Backend. Die Node-Applikation hört dabei nur auf den entsprechenden Port. Als Unterschied zu WebHooks wird auf die Rückgabe der Meldung gewartet. Gemäss Aussage von der Elektrotechnikseite hält sich diese Wartezeit mit <15 Sekunden sehr stark in Grenzen. Da die Applikation die Request asynchron ausführt, wird die Userexperience nicht beeinflusst.

Zudem sind nur sehr wenige Informationen zur Implementation eines eigenen Webhook-Endpoints zu finden. Beinahe alle Dokumentation befassen sich mit der Nutzung von bereits bestehenden Endpunkten.

5.3.5 Sprintreview Sprint 8

In diesem Sprint konnten die geplanten Userstories umgesetzt werden.

5.3.6 Sprint 9

User Story	Number
Das System bietet die Möglichkeit, durch die Anbindung an eine 3rd Party, eine Altersverifikation durchzuführen.	F.4
Das System muss die Punkte in der von Google aufgestellten Core Progressive Web App checklist erfüllen	F.1

Tabelle 8: Userstories Sprint 9,
Quelle: Autor

Altersverifikation Jumio bietet keine Testversion an. Um die Integration in die Applikation durchführen zu können, wurden vom Auftraggeber der API-Key zur Verfügung gestellt. Zudem

wäre die Möglichkeit bestanden, Initiale Konfigurationen im Backoffice durchführen zu können. Darauf wurde verzichtet, da die gewünschten Einstellungen direkt mit dem API-Call definiert werden können.

Jumio bietet eine GitHub Integration Guide an. In diesem wird sehr gut erklärt, wie der initiale Request definiert werden muss. Um einem erneuten CORS-Problem vorzusorgen, wurde direkt das Vorgehen von 5.3.4 übernommen.

```
let ageVerification = {
  "customerInternalReference" : "",
  "userReference" : "",
  "successUrl" : "https://bdaf21-owerlen.enterpriselab.ch/ageVerificationSucessfull",
  "errorUrl" : "https://bdaf21-owerlen.enterpriselab.ch/error",
  "callbackUrl" : "https://bdaf21-owerlen-02.enterpriselab.ch/api/v1",
  "reportingCriteria" : "",
  "workflowId" : 100,
  "presets" :
  [
    [
      {
        "index" : 1,
        "country" : "CHE",
        "type" : "ID_CARD"
      }
    ],
    "locale" : "ch-CH"
}
```

Abbildung 48: Initialer Request an Jumio API,
Quelle: Autor

Dabei wird hier nur die customerInternalReference und userReference im Backend gesetzt. CustomerInternalReference entspricht dabei dem HashCode vom User, als userReference dient die userId. Zudem ist es vorgegeben, dass die URLs via https ausgeliefert werden sowie keine Sonderzeichen enthalten. Die Callback URL wird nicht aktiv genutzt.

Als Antwort auf den Request werden die transactionReference, der Timestamp und die redirectUrl zurückgegeben. Wie auch bei der Zahlung wird auf die RedirectUrl weitergeleitet. Anschließend beginnt der Prozess bei Jumio.

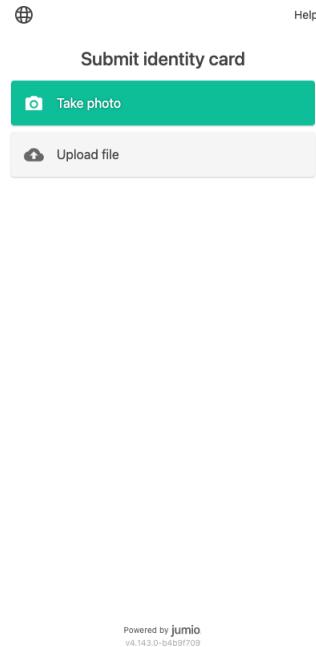


Abbildung 49: Start der Verifikation mit Jumio,
Quelle: Autor

Nach dem Abschluss der Altersverifikation wird je nach Status auf die successUrl oder auf die errorUrl weitergeleitet. Im Success-Fall hängt Jumio der Url noch drei weitere Parameter an. Dabei handelt es sich um den Status, die customerInternalReference und die transactionReference.

Profil sperren Aus rechtlicher Sicht ist es essentiell, dass nur Personen, die nachweislich älter als 18 Jahre sind, den Dienst nutzen können. Um dies zu gewährleisten, dass dies erfüllt ist, wurde auf eine Eigenschaft von Spring Security zurückgegriffen. Bei der Nutzererstellung wird ein Objekt vom Typ User Details Implementation erstellt. Dieses Implementiert das Interface User Details, in welchem unter anderem Methoden zum Sperren und Aktivieren von Profilen definiert sind. Bei der bisherigen Anwendung wurden diese Methoden überschrieben, ohne explizite Werte zu berücksichtigen. Allerdings bietet sich dieser Anwendungsfall an, um eine eigene Implementation der Methoden durchzuführen.

Es wurde die isEnabled() Methode ausgewählt, da diese dem gewünschten Effekt entspricht. Anstatt das hier nur true zurückgegeben wird, greift die Methode auf ein Attribut der Klasse UserDetail- sImpl zu.

```

@Override
public boolean isEnabled() {
    return this.ageVerified;
}

```

Abbildung 50: Überschreiben der isEnabled() Methode,
Quelle: Autor

Profil entsperren Per Default ist der Account gesperrt. Erst bei erfolgreicher Altersverifi- kation wird der Account entsperrt und erlaubt somit ein Einloggen und nutzen der Applikation. Um den Account zu entsperren, wird die TransactionReference im Backend gespeichert. Da diese

bei einer erfolgreichen Verifikation in der Url mitgegeben wird, kann über diese das Property auf dem User-Objekt gesetzt und der Account verfügbar gemacht werden.

Integration von PWA Funktionalität

Offlinefähigkeit Um die Applikation auch ohne aktive Internetverbindung nutzbar zu machen, wurden Service Worker genutzt. Das Hinzufügen zum Projekt übernahm die Angular CLI. wurde ein neuer Command im package.json hinzugefügt.

```
ng add @angular/pwa --project jtiPickUp
```

Service Worker funktionieren mit dem klassischen ng-serve nicht. Um das Projekt trotzdem lokal testen zu können, gibt es die Möglichkeit, einen lokalen Http-Server laufen zu lassen. Um diesen Prozess zu vereinfachen, wurde ein neuer Command im package.json hinzugefügt.

```
"start-pwa": "ng build --prod && http-server -p 8090 -c-1 dist/jtiPickUpStation"
```

Die Service Worker sind nun aktiv. Dies ist auch in der Developer Konsole von Google Chrome zu sehen.

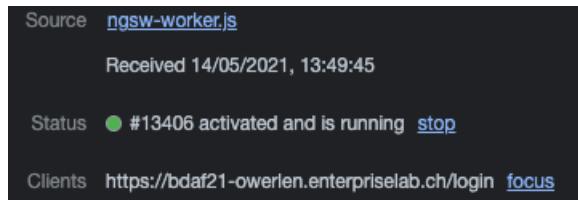


Abbildung 51: Aktiver Service Worker in Google Chrome,
Quelle: Autor

Per Default cached der Service Worker folgende Dateien:

- index.html
- favicon.ico
- build artifacts
- assets
- images und fonts

Google, 2021c Der Service Worker speichert die Version im Hintergrund. Aus performance Gründen wird immer die gespeicherte Version ausgeliefert, auch wenn eine neuere Vorhanden ist.

Caching von Request Für die Applikation reichte das Speichern der obigen Dateien nur bedingt aus. Damit die App auch Offline benutzbar bleibt, war es nötig, die Request zu cachen. Angular bietet zwei Arten von Caching an: performance und freshness. Bei Performance werden die Daten immer aus dem cache geladen, sofern sie das maxAge noch nicht erreicht haben. Dies bringt, wie es der Name bereits sagt, einen enormen Performancegewinn. Bei Freshness werden die Daten immer von der API geladen, sofern das Timeout nicht erreicht wurde. Dies garantiert, dass die Daten aktuell sind.

In der folgenden Tabelle wird aufgezeigt, welche Request von welcher Strategie gebrauch machen.

Performance	Freshness
Laden von Produkten	Laden von Inventar
	Orders by User
	PickUpStations

Tabelle 9: Übersicht über die Verwendung von Freshness und Performance,
Quelle: Autor

Mit der Performance-Strategie werden ausschliesslich Daten geladen, welche sich nur sehr selten ändern. Die Service Worker haben hauptsächlich zum Zweck, die Applikation Offlinefähig zu machen. Der Performancegewinn ist sekundär.

Nachfolgend werden noch die Features aufgeführt, welche im Offline-Modus nicht verfügbar sind:

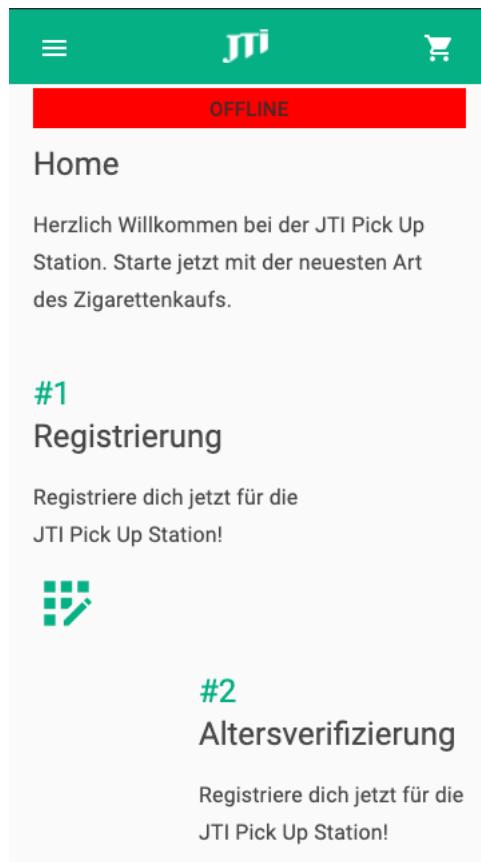
- Login und Registrierung
- Platzieren einer Bestellung
- Abholen einer Bestellung

Anzeigen von Online und Offlinestatus Der User sollte auch informiert werden, wenn die Internetverbindung unterbrochen ist. Um dies umzusetzen, wird ein ng-connection-service bereitgestellt. Dieser kann überwacht werden und somit eine Netzwerkveränderung bemerkt werden.

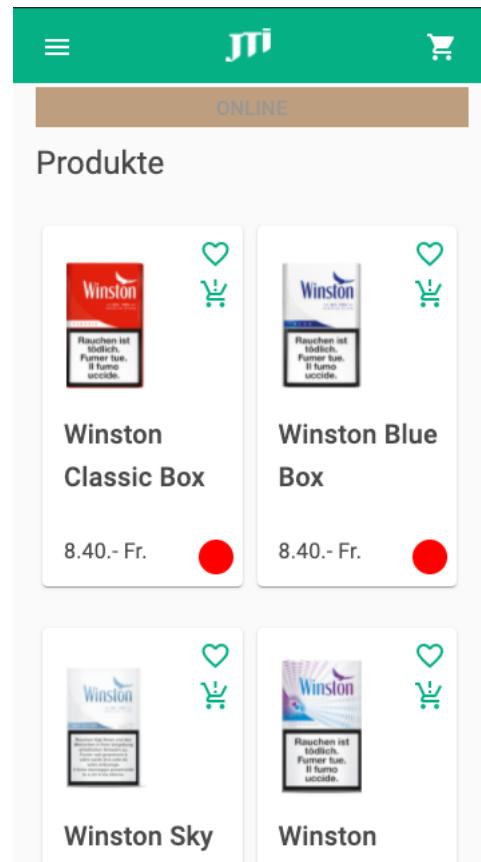
```
    this.connectionService.monitor().subscribe(isConnected => {
      this.isConnected = isConnected;
      if (this.isConnected) {
        this.status = "ONLINE";
      }
      else {
        this.status = "OFFLINE";
      }
    })
```

Abbildung 52: Montoring der Connection-Status,
Quelle: Autor

Entsprechend des Status wird eine Meldung in der Applikation angezeigt. Die Meldung in 53b verschwindet dabei nach zwei Sekunden.



(a) Offline-Meldung,
Quelle: Autor



(b) Übergang Offline zu Online,
Quelle: Autor

6 Evaluation und Validation

7 Ausblick

8 Verzeichnisse

Abbildungsverzeichnis

1	My Post 24-Abholstelle	6
2	Tabakkauf avec box	7
5	Standort in der Starbucks PWA	10
6	Architektur	11
7	DTO Klassendiagramm von Martin Fowler	13
8	Richardson Maturity Model	14
9	Ausschnitt aus der Swagger Dokumentation	15
10	GitLab Board	16
11	Ablauf der CI/CD Pipeline	17
12	Beispielanwendung QR-Code auf Gerät	20
13	Verfügbare Shared-Runner von GitLab	23
14	CLI Commands GitLab Container Registry	24
15	Webhooks Funktionsweise	26
19	Authentication im Backend	31
20	Token Auflösung mit jwt.io	31
22	Authentication Interceptor	33
23	Token im Session Storage	33
24	Ablauf von Timer und anschliessender Logout	34
25	Erstellen des Basic Authentication Token im Back Office	36
26	Saferpay Bezahlseite	37
27	PagePaymentAssertResponse	38
28	Bezahlhistorie Backoffice	39
29	Sequenzdiagramm vom Bezahlvorgang	40
30	Entity Relationship Diagram von Order	41
31	Response bei Abfrage einer PickUp-Station	42
32	Erstellung von Marker und Tooltip	44
33	Darstellung von Marker und Tooltip	44
34	Observable States	45
35	Mouse Event	45
36	Erstellen der Order	46
37	Erstellen der OrderItems	46
39	Auslesen der Werte aus .properties-File	47
40	Berechnung des Order-Totals	47
41	Senden eines Request mit Rest Template	48
42	Erstellen der HttpEntity	48
43	Überprüfung des Zahlungsstatus	48
44	Erstellen eines QR-Code mit ZXing	49
45	QR-Code Beispiel	49
46	PickUp Station Identifier und PickUp Token	49
47	Kommunikation zwischen Station und Backend	50
48	Initialer Request an Jumio API	51
49	Start der Verifikation mit Jumio	52
50	Überschreiben der isEnabled() Methode	52
51	Aktiver Service Worker in Google Chrome	53
52	Monitoring der Connection-Status	54
54	Organigramm	75
55	Projektstrukturplan	76
56	SoDa Rahmenplan	77
57	Risikomatrix	79
58	RisikomatrixNach	81

59	Systemarchitektur	84
60	Kontextdiagramm	85
61	Datenbankschema	86
62	Aufbau Produktives System	87
63	SoDa Rahmenplan Version 1	105

Glossar

Github grösste Versionskontrollplattform [[Github](#)]. 58

GitLab Versionskontrollsysteum aufbauend auf git, Alternative zu Github. 16

React Java Script Library zum Erstellen von User Interfaces [Facebook, 2021]. 12

SoDa Hybrides Projektmanagementvorgehen der Hochschule Luzern. 1, 16

User Story Element in SoDa, werden aus Epics im Format Als Rolle möchte ich Ziel/Wunsch, um Nutzen erstellt [[userStory](#)]. 16

Abkürzungsverzeichnis

SoDa Software Development Agile.....	18
JPA Java Persistence API.....	12
CI/CD Continous Integration and Continous Deployment.....	16
CORS Cross Origin Ressource Sharing.....	40
HATEOAS Hypermedia as the Engine of Application State	1
HATEOAS Hypertext as the engine of application state.....	1
DTO Date Transfer Object.....	25
JKI Japan Tobacco International.....	90
PWA Progressive Web App	1
NFC Near Field Communication	19
DTO Data Transfer Object.....	25
HATEOAS Hypertext as the engine of application state	1
IoT Internet of Things.....	25
PCI DSS Payment Card Industry Data Security Standard.....	35
SAQ Self Assigned Questionary	35
CORS Cross-Origin Ressource Sharing.....	40
TLS Transport Layer Security	40
JPA Java Persistence API	12

Tabellenverzeichnis

1	Userstories Sprint 1	19
2	Userstories Sprint 2	21
3	Userstories Sprint 4	28
4	Userstories Sprint 5	30
5	Userstories Sprint 6	34
6	Userstories Sprint 7	43
7	Userstories Sprint 8	45
8	Userstories Sprint 9	50
9	Übersicht über die Verwendung von Freshness und Performance	54
10	Testprotokoll Test 1, Quelle: Autoren	66
11	Testprotokoll Test 2, Quelle: Autoren	67
12	Testprotokoll Test 3, Quelle: Autoren	68
13	Testprotokoll Test 4, Quelle: Autoren	68
14	Testprotokoll Test 5, Quelle: Autoren	69
15	Testprotokoll Test 6, Quelle: Autoren	70
16	Testprotokoll Test 7, Quelle: Autoren	71
17	Testprotokoll Test 8, Quelle: Autoren	72
18	Testprotokoll Test 9, Quelle: Autoren	73
19	Testprotokoll Test 10, Quelle: Autoren	74
20	Meilensteine, Quelle: Autoren	78
21	Risikoanalyse, Quelle: Autoren	79
22	Risikoanalyse nach Massnahmen, Quelle: Autoren	80
23	Entwicklungstools, Quelle: Autoren	82
24	Testtools, Quelle: Autoren	82
25	Konfigurationseinheit Release 1, Quelle: Autoren	82
26	Funktionale Anforderungen, Quelle: Autoren	91
27	Nicht Funktionale Anforderungen, Quelle: Autoren	92
28	Änderungshistorie der Anforderungen, Quelle: Autoren	93
29	Sitzungsprotokoll, Quelle: Autoren	94
30	Sitzungsprotokoll, Quelle: Autoren	96
31	Sitzungsprotokoll, Quelle: Autoren	97
32	Sitzungsprotokoll, Quelle: Autoren	98
33	Sitzungsprotokoll, Quelle: Autoren	99
34	Sitzungsprotokoll, Quelle: Autoren	101
35	Sitzungsprotokoll, Quelle: Autoren	102
36	Sitzungsprotokoll, Quelle: Autoren	103
37	Sitzungsprotokoll, Quelle: Autoren	104

Literatur

- AG, D. S. P. (2021). Versenden und empfangen rund um die Uhr. Zugriff unter <https://www.post.ch/de/empfangen/empfangsorte/pickpost-my-post-24/my-post-24>. (03.03.2021)
- Agafonkin, V. (2020). an open-source JavaScript library for mobile-friendly interactive maps. Zugriff unter <https://leafletjs.com/index.html>. (23.04.2021)
- Apple. (2021). Tracking Prevention in WebKit. Zugriff unter <https://webkit.org/tracking-prevention/>. (12.03.2021)
- baeldung. (2020). Generating Barcodes and QR Codes in Java. Zugriff unter <https://www.baeldung.com/java-generating-barcodes-qr-codes>. (02.05.2021)
- baeldung. (2021). The Guide to RestTemplate. Zugriff unter <https://www.baeldung.com/rest-template>. (27.04.2021)
- Beni, E. H. (2021). Spring Boot Secured By Let's Encrypt. Zugriff unter <https://dzone.com/articles/spring-boot-secured-by-lets-encrypt>. (06.04.2021)
- bezkoder. (2020). Angular 8 JWT Auth – Token based Authentication with Web Api example. Zugriff unter <https://bezkoder.com/spring-boot-jwt-authentication/>. (06.04.2021)
- bezkoder. (2021). Spring Boot Token based Authentication with Spring Security JWT. Zugriff unter <https://bezkoder.com/spring-boot-jwt-authentication/>. (06.04.2021)
- Böhm, R. (2017). Was sind Angular und Angular JS. Zugriff unter <https://angular.de/artikel/was-ist-angular/>. (14.03.2021)
- Bohndorf, A. (o.D.). DOCKER-COMPOSE SETUP MIT NGINX REVERSE PROXY. Zugriff unter <https://sitegeist.de/blog/typo3-blog/docker-compose-setup-mit-nginx-reverse-proxy.html>. (22.03.2021)
- congstar. (2021). NFC-Near Field Communication. Zugriff unter <https://www.congstar.de/handys/technik-news-trends/nfc/>. (12.03.2021)
- Coop. (2020). Lieferbedingungen. Zugriff unter <https://www.coop.ch/de/wie-wir-liefern.html>. (03.03.2021)
- cozmo. (2021). jsQR. Zugriff unter <https://github.com/cozmo/jsQR>. (04.05.2021)
- Doran, T. (2018). IEEE/ISO/IEC 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. Zugriff unter <https://standards.ieee.org/standard/29148-2018.html>. (27.02.2021)
- DB-Engines. (2021). Vergleich der Systemeigenschaften MariaDB vs. PostgreSQL vs. MySQL. Zugriff unter <https://db-engines.com/de/system/MariaDB%3BPostgreSQL>. (14.03.2021)
- Facebook. (2021). React. Zugriff unter <https://reactjs.org/>. (14.03.2021)
- Fowler, M. (2002). Data Transfer Object. Zugriff unter <https://martinfowler.com/eaaCatalog/dataTransferObject.html>. (7.05.2021)
- Fowler, M. (2010). Richardson Maturity Model. Zugriff unter <https://martinfowler.com/articles/richardsonMaturityModel.html>. (06.05.2021)
- Genf, K. (o.D.). Versand und Zahlungsbedingungen. Zugriff unter <https://www.kiosklino.ch/de/versand-und-zahlungsbedingungen>. (03.03.2021)
- Geoapify. (2020). Angular + Leaflet: step by step tutorial to add a map. Zugriff unter <https://www.geoapify.com/angular-leaflet-step-by-step-tutorial-to-add-a-map>. (23.04.2021)
- GitLab. (2021). Configuring runners in GitLab. Zugriff unter <https://docs.gitlab.com/ee/ci/runners/>. (18.03.2021)
- Google. (2021a). Angular Material. Zugriff unter <https://material.angular.io/>. (14.03.2021)
- Google. (2021b). Angular Service Worker Introduction. Zugriff unter <https://angular.io/guide/service-worker-intro>. (14.03.2021)
- Google. (2021c). Getting started with service workers. Zugriff unter <https://angular.io/guide/service-worker-getting-started>. (08.05.2021)
- Hat, R. (2021). Hibernate ORM. Zugriff unter <https://hibernate.org/orm/>. (06.05.2021)
- HSLU. (o.D. a). Artefakte und Downloads Planungs- und Entwurfsdokumente. Zugriff unter <https://www.hslu.ch/de-ch/informatik/studium/soda/artefakte-und-downloads/>. (02.03.2021)
- HSLU. (o.D. b). Planung und Vorgehen «Lieber ungefähr richtig, als genau falsch». Zugriff unter <https://www.hslu.ch/de-ch/informatik/studium/soda/planung/>. (02.03.2021)

- Johnson, L. (2020). What is a webhook: How they work and how to set them up. Zugriff unter <https://www.getvero.com/resources/webhooks/>. (31.03.2021)
- KnowHow. (2020). Reverse-Proxy-Server-Kernkomponente in Sicherheitsarchitekturen. Zugriff unter <https://www.ionos.de/digitalguide/server/knowhow/was-ist-ein-reverse-proxy/>. (22.03.2021)
- Krüger, N. (2018). How to Write a Software Requirements Specification (SRS Document). Zugriff unter <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>. (27.02.2021)
- Laboissonniere, M. (2018). What are Webhooks? Easy Explanation Tutorial. Zugriff unter <https://snipcart.com/blog/what-are-webhooks-explained-example>. (31.03.2021)
- Lenzo, M. (2016). Continuous delivery of a Spring Boot application with GitLab CI and Kubernetes. Zugriff unter <https://about.gitlab.com/blog/2016/12/14/continuous-delivery-of-a-spring-boot-application-with-gitlab-ci-and-kubernetes/>. (18.03.2021)
- Let's Encrypt. (2017). Zertifikate für localhost. Zugriff unter <https://letsencrypt.org/de/docs/certificates-for-localhost/>. (19.03.2021)
- Mozilla. (2021a). How to fix a website with blocked mixed content. Zugriff unter https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content/How_to_fix_website_with_mixed_content. (06.04.2021)
- Mozilla. (2021b). Web NFC API. Zugriff unter https://developer.mozilla.org/en-US/docs/Web/API/Web_NFC_API#browser_compatibility. (12.03.2021)
- mozilla. (2021). Cross-Origin Resource Sharing (CORS). Zugriff unter <https://developer.mozilla.org/de/docs/Web/HTTP/CORS>. (19.03.2021)
- OpenStreetMap. (2021). OpenStreetMap provides map data for thousands of web sites, mobile apps, and hardware devices. Zugriff unter <https://www.openstreetmap.org/about>. (26.04.2021)
- Patel, S. (2020). A Simple Cors Proxy for Javascript Browser applications. Zugriff unter <https://medium.com/nodejsmadeeasy/a-simple-cors-proxy-for-javascript-applications-9b36a8d39c51>. (19.03.2021)
- Post-CH-AG. (2015). *Allgemeine Geschäftsbedingungen PickPost und My Post 24*. Zugriff unter <https://www.post.ch/de/empfangen/empfangsorte/pickpost-my-post-24/my-post-24/weitere-informationen>
- RedHat. (2021). Docker - Funktionsweise, Vorteile, Einschränkungen. Zugriff unter <https://www.redhat.com/de/topics/containers/what-is-docker>. (14.03.2021)
- rednez. (2021). angular-user-idle. Zugriff unter <https://www.npmjs.com/package/angular-user-idle>. (08.05.2021)
- saferpay. (2021a). All the relevant information for developers. Zugriff unter <https://www.six-payment-services.com/en/site/e-commerce-developer/integration.html>. (12.03.2021)
- saferpay. (2021b). Saferpay Integration Guide. Zugriff unter <https://saferpay.github.io/snadbx/index.html>. (12.03.2021)
- Sam Richard, P. L. (2020). What makes a good Progressive Web App? Zugriff unter <https://web.dev/pwa-checklist/>. (24.02.2021)
- Spepanov, R. (2020). Openlayers vs Leaflet: What Makes One Better Than the Other. Zugriff unter <https://mapsvg.com/blog/openlayers-vs-leaflet>. (23.04.2021)
- Starbucks. (2021a). app.starbucks. Zugriff unter <https://app.starbucks.com>. (09.03.2021)
- Starbucks. (2021b). app.starbucks. Zugriff unter <https://app.starbucks.com/menu>. (09.03.2021)
- Starbucks. (2021c). app.starbucks. Zugriff unter <https://app.starbucks.com/store-locator?map=46.670396,7.455289,10z>. (09.03.2021)
- tagmotion. (2018). NFC vs. QR-Code. Zugriff unter <https://www.tagmotion.de/nfc-vs-qr-code/>. (12.03.2021)
- Valora. (2021a). Avec Now. Zugriff unter <https://www.avecnow.ch>. (09.03.2021)
- Valora. (2021b). Bewährtes und Neues nur für dich. Zugriff unter <https://avec.ch/de/avecbox>. (09.03.2021)
- Valora. (2021c). Scan ID. Zugriff unter Kontoerstellung. (09.03.2021)
- Valora. (2021d). Standorte und Öffnungszeiten. Zugriff unter <https://avec.ch/de/avecbox>. (09.03.2021)

- Valora. (2021e). Über avec now. Zugriff unter <https://www.avecnow.ch/pages/uber-avec-now>. (09.03.2021)
- Valora. (2021f). Versand. Zugriff unter <https://www.avecnow.ch/policies/shipping-policy>. (09.03.2021)
- VMWare. (2021). Spring Boot with Docker. Zugriff unter <https://spring.io/guides/gs/spring-boot-docker/>. (18.03.2021)
- von Uslar, C. (2021). private email communication.
- Waldmann, S. (2020). Spring oder Spring Boot, das ist hier die Frage. Zugriff unter <https://blog.doubleslash.de/spring-vs-spring-boot/>. (14.03.2021)
- Yamaguchi, N. (2020). Angular jsqr. Zugriff unter <https://stackblitz.com/edit/angular-jsqr?file=package.json>. (04.05.2021)

A Testprotokolle

A.1 Testprotokolle Bestellung

Test Nr.	1
Beschreibung	Durch diesen Test wird die Registrierung eines neuen Kunden getestet.
Randbedingungen	<ul style="list-style-type: none"> • Die Testperson hat sich mit ihrer Email-Adresse und Benutzernamen noch nie Registriert. • Die Testperson nutzt den Nutzernamen "test", die Email-Adresse "test@gmail.com" und das Passwort "ABC*123456"
erwartete Resultate	<ul style="list-style-type: none"> • Das Konto der Testperson wird korrekt angelegt. • Ein Popup zeigt dem Nutzer den Status an
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson gibt die angegebenen Daten ein. Die restlichen Daten werden zufällig gewählt 2. Die Testperson klickt auf den Button "Registrieren" 3. Es wird in einem Dialog die Erstellung des Nutzers dargestellt. 4. Es wird automatisch zur Login Page gewechselt
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Das Erstellen des Nutzers wurde vom System via Popup bestätigt.
Test bestanden	Ja

Tabelle 10: Testprotokoll Test 1, Quelle: Autoren

Test Nr.	2
Beschreibung	Durch diesen Test wird die Registrierung eines neuen Kunden getestet, wobei die Email Adresse und der Nutzernname bereits genutzt werden.
Randbedingungen	<ul style="list-style-type: none"> • Der Test 10 ist erfolgreich durchgeführt worden. • Die Testperson nutzt den Nutzernamen test , die Email-Adresse test@gmail.com und das Passwort ABC*1234 ein
erwartete Resultate	<ul style="list-style-type: none"> • Das System gibt dem Nutzer die Antwort, dass ein Benutzer mit dieser Email oder Benutzernamen bereits existiert.
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson gibt die angegebenen Daten ein. Die restlichen Daten werden zufällig gewählt 2. Die Testperson klickt auf den Button "Registrieren" 3. Es wird in einem Dialog mit der Meldung "Benutzer mit dieser Email oder Benutzernamen existiert bereits"
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Das Popup wird wie geplant angezeigt.
Test bestanden	Ja

Tabelle 11: Testprotokoll Test 2, Quelle: Autoren

Test Nr.	3
Beschreibung	Die diesen Test wird die Login-Funktion getestet.
Randbedingungen	<ul style="list-style-type: none"> Der Test 10 ist erfolgreich durchgeführt worden. Die Testperson nutzt den Benutzernamen test und das Passwort ABC*1234
erwartete Resultate	<ul style="list-style-type: none"> Eine Meldung wird angezeigt, mit welcher das erfolgreiche Erstellen des Nutzers bestätigt wird Im Sidenav wird der Logout Button angezeigt.
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> Die Testperson gibt die angegebenen Daten ein. Der Benutzer klickt auf den Login Button
erhaltenes Resultat	<ol style="list-style-type: none"> Das Popup wird wie geplant angezeigt. Das Sidenav wird entsprechend angepasst.
Test bestanden	Ja

Tabelle 12: Testprotokoll Test 3, Quelle: Autoren

Test Nr.	4
Beschreibung	Die diesen Test wird die Login-Funktion getestet. Es wird eine falsche Kombination eingetragen.
Randbedingungen	<ul style="list-style-type: none"> Die Testperson nutzt den Benutzernamen testNoAccess und das Passwort ABC*1234*noAccess
erwartete Resultate	<ul style="list-style-type: none"> Eine Meldung Falscher Benutzername oder Passwort wird angezeigt
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> Die Testperson gibt die angegebenen Daten ein. Der Benutzer klickt auf den Login Button
erhaltenes Resultat	<ol style="list-style-type: none"> Das Popup wird wie geplant angezeigt.
Test bestanden	Ja

Tabelle 13: Testprotokoll Test 4, Quelle: Autoren

Test Nr.	5
Beschreibung	Bei diesem Test wird die Logout Funktion getestet.
Randbedingungen	<ul style="list-style-type: none">• Die Testperson ist erfolgreich eingeloggt.
erwartete Resultate	<ul style="list-style-type: none">• Popup mit der Meldung erfolgreich ausgeloggt
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	1. Die Testperson klickt auf den Logout Button.
erhaltenes Resultat	1. Das Popup wird wie geplant angezeigt.
Test bestanden	Ja

Tabelle 14: Testprotokoll Test 5, Quelle: Autoren

Test Nr.	6
Beschreibung	Bei diesem Test wird das hinzufügen von Produkten in den Warenkorb getestet.
Randbedingungen	<ul style="list-style-type: none"> • Es sind Artikel im Shop vorhanden
erwartete Resultate	<ul style="list-style-type: none"> • Der Artikel ist im Warenkorb vorhanden
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson befindet sich auf der Produktübersicht 2. Die Testperson klickt bei einem zufälligen Produkt auf das Warenkorbsymbol unten rechts 3. Die Person klickt ein anderes Produkt an 4. Die Testperson klickt auf den Button "in den Warenkorb" 5. Die Testperson wechselt zum Warenkorb (Item oben rechts)
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Die beiden Produkte sind im Warenkorb zu finden 2. Die beiden Produkte befinden sich in einfacher Ausführung im Warenkorb 3. Das Total ist korrekt summiert worden.
Test bestanden	Ja

Tabelle 15: Testprotokoll Test 6, Quelle: Autoren

Test Nr.	7
Beschreibung	Bei diesem Test wird die Persistierung des Warenkorbs getestet.
Randbedingungen	<ul style="list-style-type: none"> • Der Test 15 ist erfolgreich abgeschlossen worden.
erwartete Resultate	<ul style="list-style-type: none"> • Die Artikel sind auch nach dem Verlassen der Websi-te und einem erneuten Aufruf immer noch vorhanden.
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson befindet sich im Warenkorb 2. Die Testperson schliesst das Browserfenster 3. Die Testperson öffnet das Browserfenster wieder und navigiert zum Warenkorb 4. Der Warenkorbinhalt bleibt bestehen
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Die Produkte sind immer noch im Warenkorb
Test bestanden	Ja

Tabelle 16: Testprotokoll Test 7, Quelle: Autoren

Test Nr.	8
Beschreibung	Bei diesem Test werden die Funktionen des Warenkorbs getestet.
Randbedingungen	<ul style="list-style-type: none"> • Der Test 15 ist erfolgreich abgeschlossen worden.
erwartete Resultate	<ul style="list-style-type: none"> • Die Artikelanzahl wird erhöht. • Das Gesamtotal wird erhöht. • Die Artikelanzahl wird reduziert. • Beim Erreichen von 0 wird der Artikel entfernt.
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson befindet sich im Warenkorb 2. Die Testperson erhöht die Artikelanzahl auf 4 3. Die Testperson reduziert die Artikelanzahl auf 0
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Es befinden sich keine Artikel mehr im Warenkorb 2. Es wurde ein Popup mit einer entsprechenden Meldung angezeigt.
Test bestanden	Ja

Tabelle 17: Testprotokoll Test 8, Quelle: Autoren

Test Nr.	9
Beschreibung	Bei diesem Test wird die Bezahlfunktion getestet.
Randbedingungen	<ul style="list-style-type: none"> • Der Benutzer ist erfolgreich eingeloggt. • Der Benutzer hat Produkte im Warenkorb.
erwartete Resultate	<ul style="list-style-type: none"> • Der Benutzer wird auf die Bezahlseite weitergeleitet. • Das Gesamttotal wird bei der Bezahlung korrekt ausgegeben. • Der Benutzer wird nach dem Bezahlabschluss auf eine entsprechende Seite weitergeleitet.
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson befindet sich im Warenkorb 2. Die Testperson klickt auf checkout 3. Die Testperson wird auf die Bezahlseite umgeleitet. 4. Die Testperson klickt durch den Bezahlprozess.
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Der Testperson wird eine Meldung zur Bezahlbestätigung ausgegeben.
Test bestanden	Ja

Tabelle 18: Testprotokoll Test 9, Quelle: Autoren

Test Nr.	10
Beschreibung	Bei diesem Test wird die Bezahlfunktion bei einem Abbruch getestet.
Randbedingungen	<ul style="list-style-type: none"> • Der Benutzer ist erfolgreich eingeloggt. • Der Benutzer hat Produkte im Warenkorb.
erwartete Resultate	<ul style="list-style-type: none"> • Der Benutzer wird auf die Bezahlseite weitergeleitet. • Das Gesamttotal wird bei der Bezahlung korrekt ausgegeben. • Der Benutzer bricht den Bezahlvorgang mittels Cancel Button ab. • Der Benutzer wird auf eine entsprechende Seite weitergeleitet.
Testperson	Oliver Werlen
Datum	18.04.2021
Durchführung	<ol style="list-style-type: none"> 1. Die Testperson befindet sich im Warenkorb 2. Die Testperson klickt auf checkout 3. Die Testperson wird auf die Bezahlseite umgeleitet. 4. Die Testperson klickt beim Bezahlvorgang auf den cancel Button.
erhaltenes Resultat	<ol style="list-style-type: none"> 1. Der Testperson wird auf eine Page not found Seite umgeleitet.
Test bestanden	Ja

Tabelle 19: Testprotokoll Test 10, Quelle: Autoren

B Projektmanagementplan

B.1 Projektorganisation

B.1.1 Organisationsplan, Rollen, Zuständigkeiten

In nachfolgendem Diagramm sind alle Projektbeteiligten aufgeführt. Die Projektmitglieder von der Hochschule Luzern Technik und Architektur unterstehen dabei in diesem Projekt keiner hier genannten Person. Sie haben aus Ihrem Projekt entsprechend eigene Projektorganisationen.

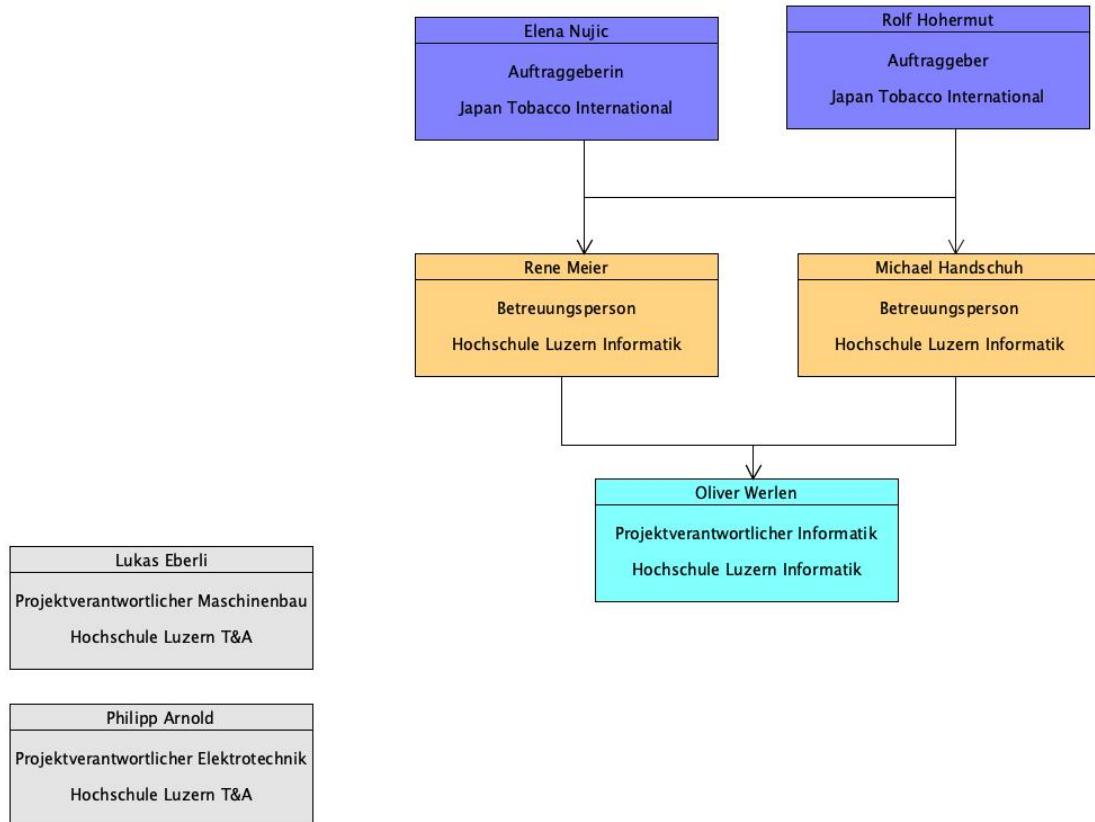


Abbildung 54: Organigramm, Quelle: Autoren

Rollen Im Projekt wird nach dem hybriden Projektmanagementvorgehen SoDa gearbeitet. Es werden die hier genutzten Rollen beibehalten.

- Projektleiter/in
- Product Owner
- Scrum Master
- Scrum Team

[HSLU, o.D. b] Da es jedoch in diesem Projekt nur einen aktiven Projektmitarbeiter gibt, werden alle Rollen von Oliver Werlen übernommen.

B.1.2 Projektstrukturplan

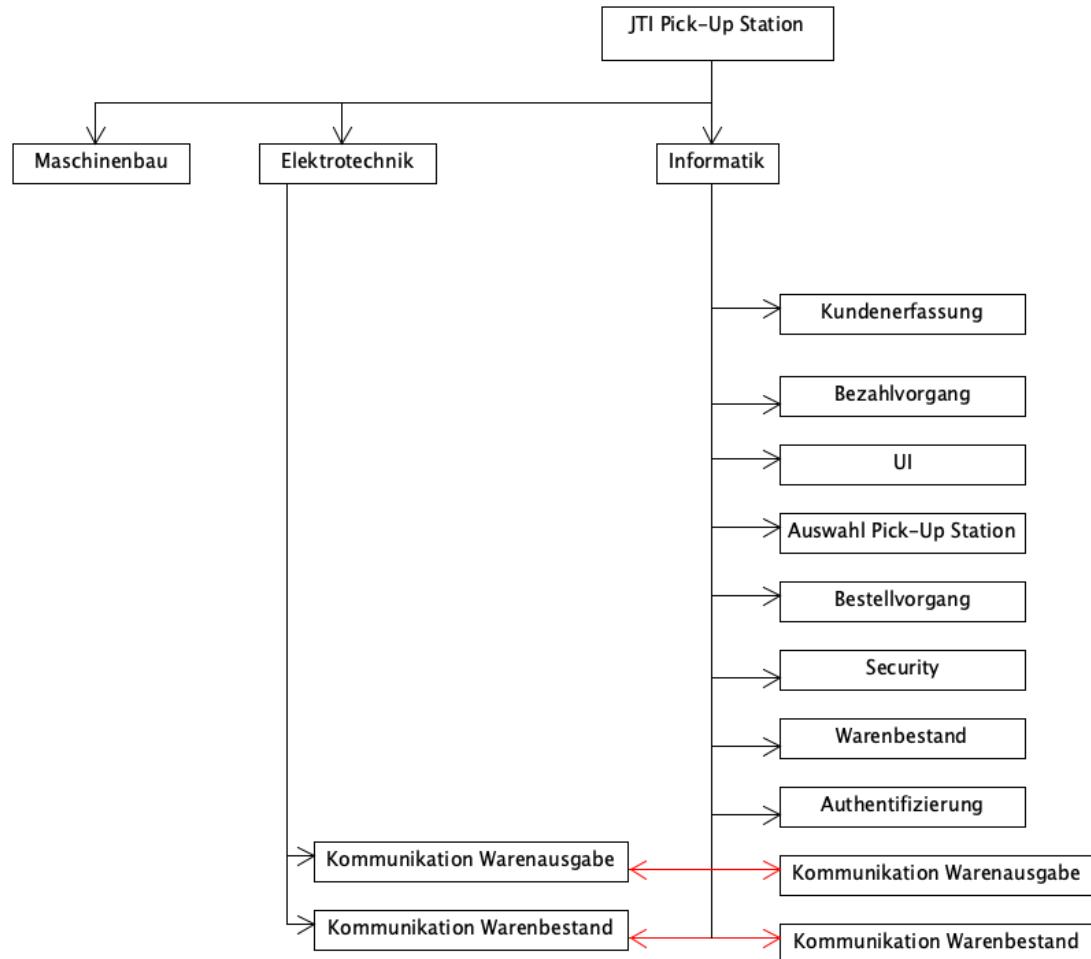


Abbildung 55: Projektstrukturplan,
Quelle: Autoren

Beschreibung Im obigen Projektstrukturplan in Abbildung 55 werden die wichtigsten Teilbereiche der Applikation aufgelistet. Dabei wird der Fokus auf den Informatikteil gelegt. Es werden einzig die Schnittstellen zur Elektrotechnik berücksichtigt. Diese wurden rot eingezzeichnet. Die Teilbereiche beziehen sich dabei hauptsächlich auf die in D erarbeiteten Anforderungen.

B.2 Projektführung

B.2.1 Rahmenplan

Im untenstehenden Rahmenplan wird mittels Zeitstrahl eine Grobplanung dargestellt.

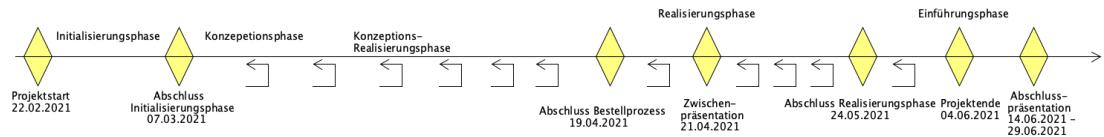


Abbildung 56: Rahmenplan,
Quelle: Autoren

Der Rahmenplan wurde zu Beginn des Projekts grob dargestellt. Im Verlauf des Projekts kann dieser bei Bedarf angepasst werden. Die einzelnen Versionen des Rahmenplans sind im Anhang F zu finden.

B.2.2 Meilensteine

Wie in Abbildung 56 zu sehen gibt es insgesamt sieben Meilensteine. Diese werden in folgender Tabelle beschrieben sowie die nötigen Deliverables aufgezeigt.

Meilenstein	Beschreibung	Deliverables
Projektstart	Das Kick-Off Meeting mit allen Projektteilnehmern wurde durchgeführt und das Projekt freigegeben.	finale Aufgabenstellung
Abschluss Initialisierungsphase	In der Initialisierungsphase wurden alle zum erfolgreichen Start benötigten Unterlagen erstellt. Die Anforderungen wurden von allen Projektmitgliedern akzeptiert.	Projektmanagementplan, Systemspezifikation, Anforderungsliste
Abschluss Bestellprozess	Der Bestellprozess Artikelauswahl, Artikel in Warenkorb, Artikel Bezahlen sowie die Kundenregistrierung sind umgesetzt und getestet.	Testprotokolle zu Abschluss Bestellprozess, Demo Bestellprozess, Release Bestellprozess Release 1
Zwischenpräsentation	Die Zwischenpräsentation ist durchgeführt worden.	Zwischenpräsentation im Anhang, Sitzungsprotokoll
Abschluss Realisierungsphase	Die noch fehlenden Anforderungen aus dem vorherigen Meilenstein sind hier abzuliefern. Es handelt sich dabei um die Auswahl sowie die Abholung an einer Pick-Up Station. Zudem ist die Abfrage des Warenbestandes Teil dieses Meilensteins.	Testprotokolle zu Abholung, Testprotokolle Auswahl, Integration alte Daten, Demo verschiedene Features Release 3
Start Einführung	Der Auftraggeber erhält eine Einführung in die Software	Sitzungsprotokoll zum Ende der Einführungsphase
Projektende	Der Auftraggeber erhält eine Einführung in die Software	Fertige Projektdokumentation, Abgeschlossene Testprotokolle Release 4
Abschlusspräsentation	Die Abschlusspräsentation ist durchgeführt worden.	-

Tabelle 20: Meilensteine, Quelle: Autoren

B.2.3 Risikomanagement

Beim Risikomanagement werden die wichtigsten Risiken für das Projekt ermittelt und passende Gegenmassnahmen ausgearbeitet.

Risiko	Eintrittswahrsch.	Schaden
Falsche Zeiteinschätzung	70	80
Requirements nehmen zu / Requirements ändern sich	60	60
Entwicklerausfall	40	70
Unklare Schnittstellenspez.	40	70
Vernachlässigung Designprozess	20	60
Fehlende technische Kompetenz	20	90
Veränderung im Projektteam	10	70

Tabelle 21: Risikoanalyse, Quelle: Autoren

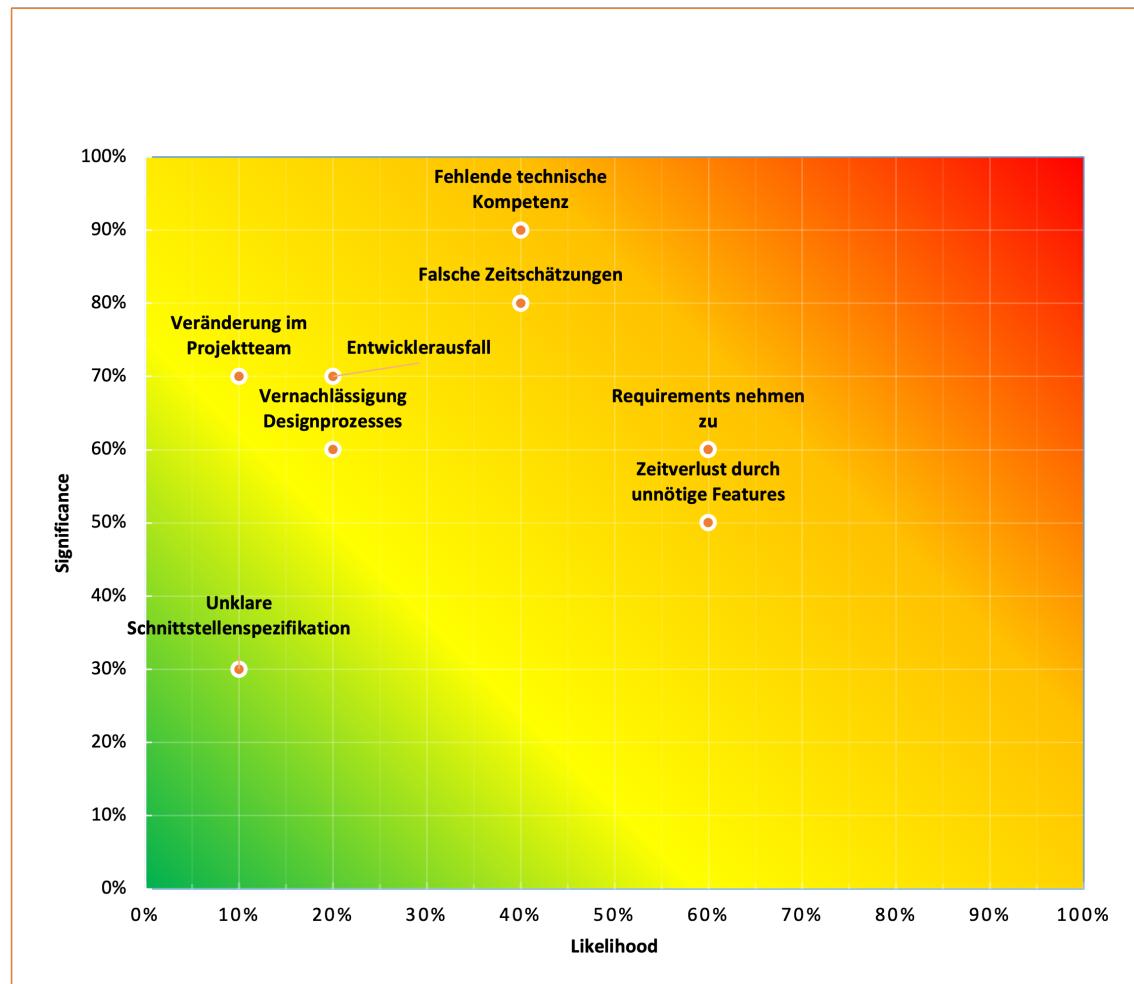


Abbildung 57: Risikomatrix,
Quelle: Autoren

Beschreibung Basierend auf der Risikomatrix in Abbildung 57 müssen für die Risiken im rechten oberen Viertel Gegenmassnahmen erarbeitet werden. In diesem Viertel liegt allerdings nur das Risiko "Requirements nehmen zu". Aus diesem Grund werden hier noch weitere Risiken bearbeitet.

- Requirements nehmen zu
- Zeitverlust durch unnötige Features
- Falsche Zeiteinschätzung
- Fehlende technische Kompetenz

Gegenmassnahmen

Requirements nehmen zu Um eine Veränderung der Requirements während des Projekt zu vermeiden, werden die Requirements in ständigem Kontakt mit den Auftraggebern erarbeitet und von diesen abgenommen.

Zeitverlust durch unnötige Features Um dies zu verhindern werden die entsprechenden User Stories definiert. Es werden dabei nur die Requirements berücksichtigt, welche beim Requirements Engineering erarbeitet und vom Auftraggeber abgenommen wurden.

Falsche Zeiteinschätzung Um eine bessere Zeiteinschätzung zu erlangen, wird auf das Wissen aus vorherigen Projekten zurückgegriffen. Basierend darauf kann die Planung genauer durchgeführt werden.

Fehlende technische Kompetenz Es werden Technologien verwendet, welche bereits bekannt sind. Zudem finden diese in vielen Projekten Anwendung, sodass auf das Wissen von erfahrenen Entwicklern zurückgegriffen werden kann.

Risiko	Eintrittswahrsch.	Schaden
Requirements nehmen zu / Requirements ändern sich	20	10
Zeitverlust unnötige Features	30	40
Falsche Zeiteinschätzung	30	80
Fehlende technische Kompetenz	20	20

Tabelle 22: Risikoanalyse nach Massnahmen, Quelle: Autoren

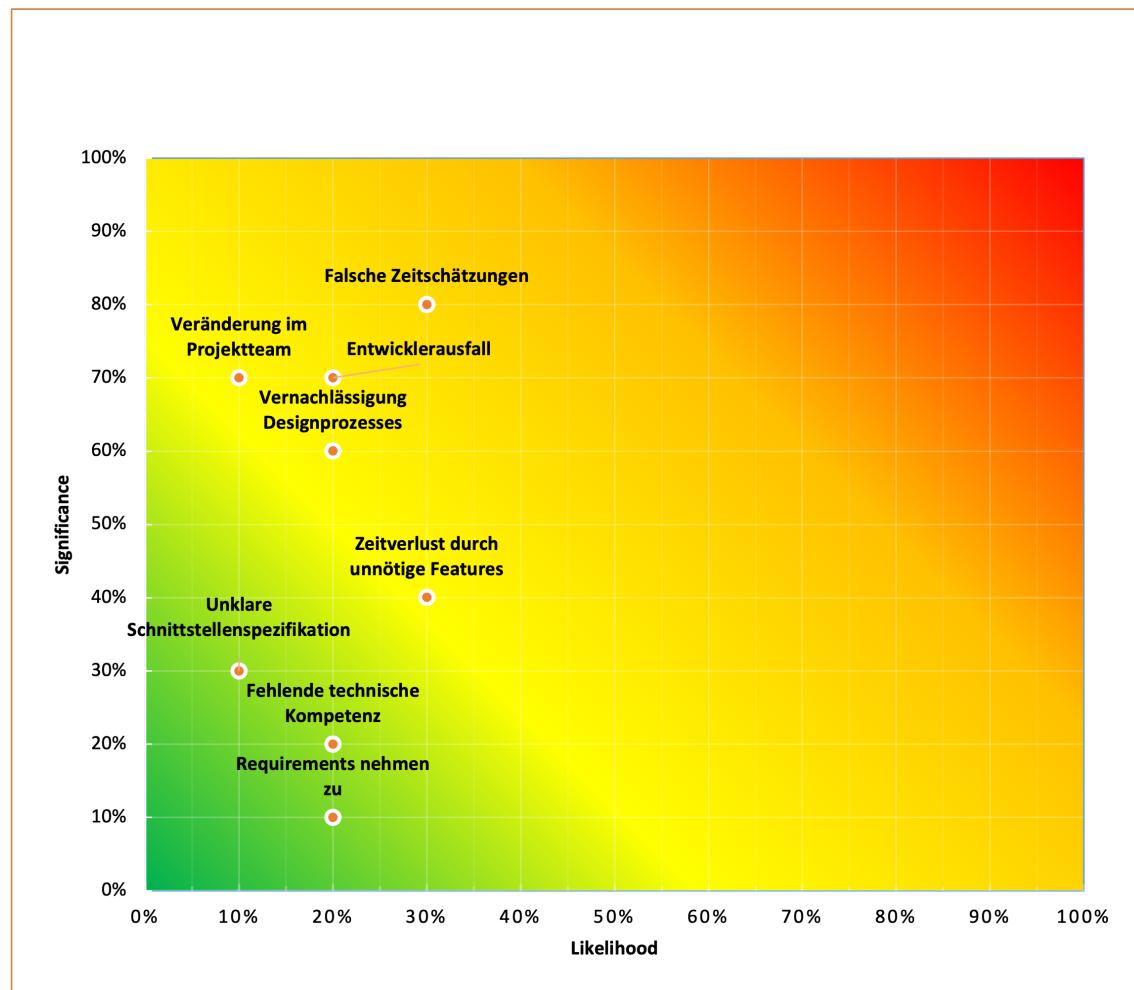


Abbildung 58: Risikomatrix nach Massnahmen,
Quelle: Autoren

B.2.4 Definition of done

In jedem Sprint müssen die nachfolgenden Punkte zwingend erreicht werden, um ein potenziell auslieferbares Produkt zu erhalten:

- Review durchgeführt
- Akzeptanzkriterien erfüllt
- Unit Tests Grün
- CI/CD ohne Fehler
- keine kritischen Bugs
- Clean Code Guidelines eingehalten
- Dokumentation aktuell

B.3 Projektunterstützung

B.3.1 Tools für Entwicklung, Test und Abnahme

Entwicklungstools Bei der Entwicklung des Projekts kommen folgende Programme zum Einsatz:

Typ	Tool	Version
IDE	InteliJ Ultimate	2020.1
IDE	Visual Studio Code	1.53.2
Versionsverwaltung	Git	2.27.0

Tabelle 23: Entwicklungstools, Quelle: Autoren

Testtools Beim Testing kommen folgende Tools zum Einsatz

Typ	Tool	Version
Unit Testing	JUnit	5.6.2
API Testing	Postman	7.36.0

Tabelle 24: Testtools, Quelle: Autoren

B.3.2 Konfigurationsmanagement

Konfigurationseinheit Bei diesem Projekt besteht eine Konfigurationseinheit aus mehreren Teilen. Dabei werden diese bei jedem Release aufgeführt. Zusätzlich dazu kommen noch die Reports der Automatisierten Tests, falls vorhanden auch der Systemtests.

- API
- Datenbank
- Webapplikation
- Dokument

Typ	Version
API	1.0.0
Datenbank	x
Webapplikation	x
Dokumentation	x

Tabelle 25: Konfigurationseinheit Release 1, Quelle: Autoren

Release 1

Testprotokolle Die gesamten Testprotokolle sind im Anhang A zu finden.

B.4 Teststrategie und Drehbuch

B.4.1 Teststrategie

Es handelt sich hier um die Entwicklung eines Prototypen. Die Priorität wird entsprechend gesetzt.

- Funktionalität vor Design
- Sicherheit vor Tempo
- Funktionalität vor Testing

Es wurde sich dabei bewusst auf diese Priorisierungen beschränkt. Aus diesem Grund wurden die Unit- und Integrationstests nur konzeptuell Umgesetzt. Bei Bedarf können diese erweitert werden.

Automated Testing der REST-Schnittstelle Zum Testen der REST-Schnittstelle wird dabei in erster Linie Postman genutzt.

B.4.2 Testdrehbuch

Wie oben genannt wird auf manuelles Testing gesetzt. Die Tests gehen mit den gleichnamigen Meilensteinen einher. Nachfolgend werden diese inklusive den erhaltenen Resultate beschrieben.

Die Testdrehbücher sind dabei direkt in die Testprotokolle A integriert. An dieser Stelle wird auf ein erneutes Beschreiben verzichtet.

B.5 Bemerkungen

Zur Erstellung des Projektmanagementplans wurde die Vorlage der Hochschule Luzern verwendet.
[HSLU, o.D. a]

C System-Spezifikation

C.1 Systemübersicht

C.1.1 Systemarchitektur

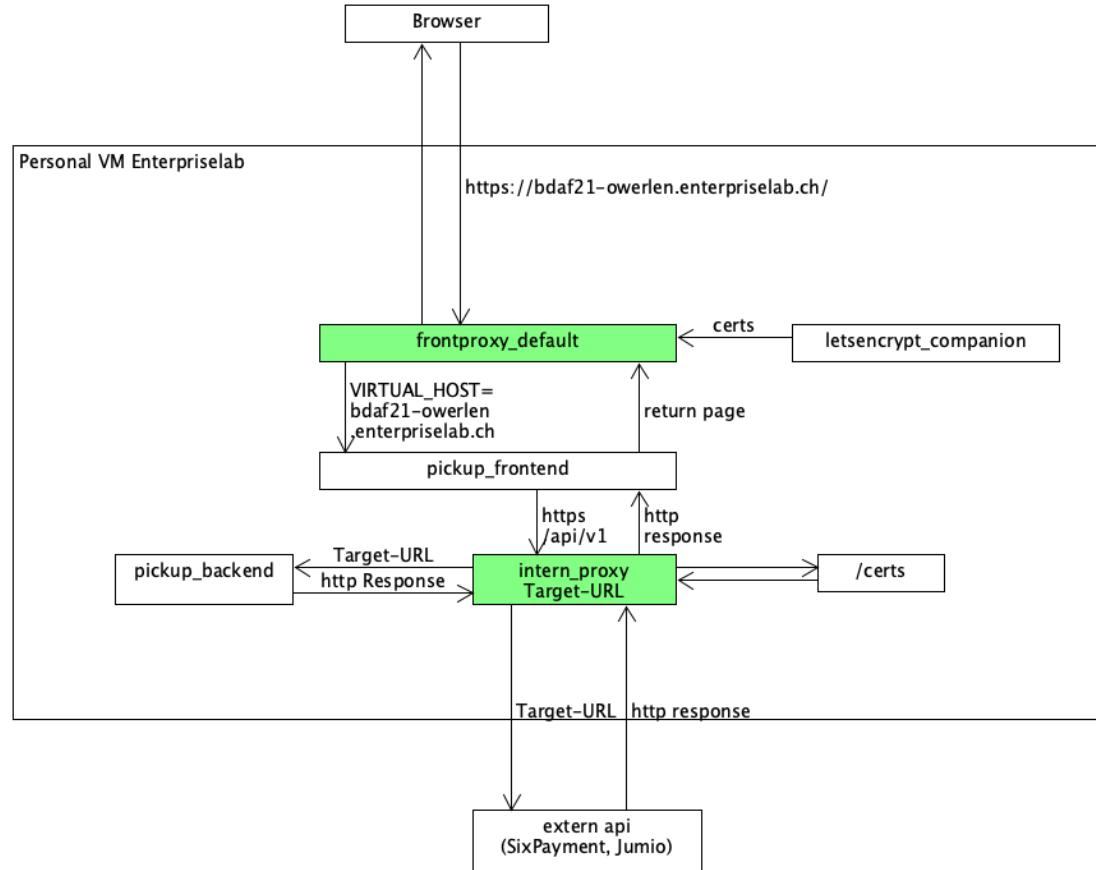


Abbildung 59: Systemarchitektur,
Quelle: Autoren

C.1.2 Kontextdiagramm

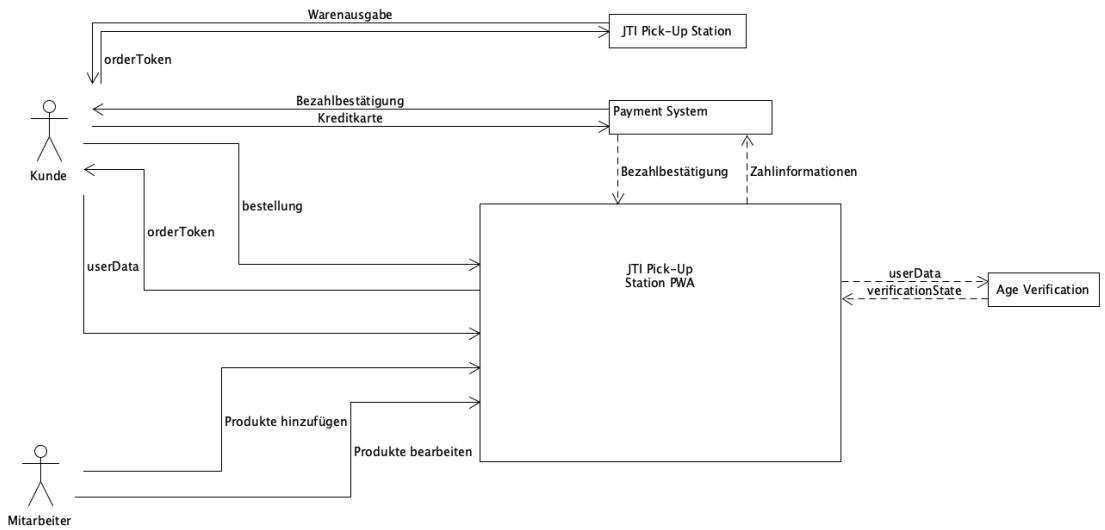


Abbildung 60: Kontextdiagramm,
Quelle: Autoren

C.2 Architektur und Designentscheide

Es wurde bei diesem Projekt auf eine REST-Architektur gesetzt. Die Web-Applikation wird als PWA umgesetzt.

C.2.1 Modelle und Sichten

In diesem Projekt wird zwischen zwei verschiedenen Sichten unterschieden:

- **Kunde** Es handelt sich dabei um die Person, welche in der PWA Produkte bestellt und diese anschliessend abholt.
- **Administrator** Dem Administrator ist es möglich, Produkt hinzuzufügen, zu verändern oder auch zu löschen.
- **Programmierer:** Dieser konzipiert und realisiert die Applikation gemäss den Anforderungen des Auftraggebers.

C.2.2 Daten (Mengengerüst und Strukturen)

Datenbankschema Das Datenbankschema wurde mittels Reverse Engineering mit MySQL Workbench erstellt und ist in der Abbildung 61 ersichtlich.

Abbildung 61: Datenbankschema, Quelle: Autoren

C.2.3 Entwurfsentscheide

Frontend

Technologien

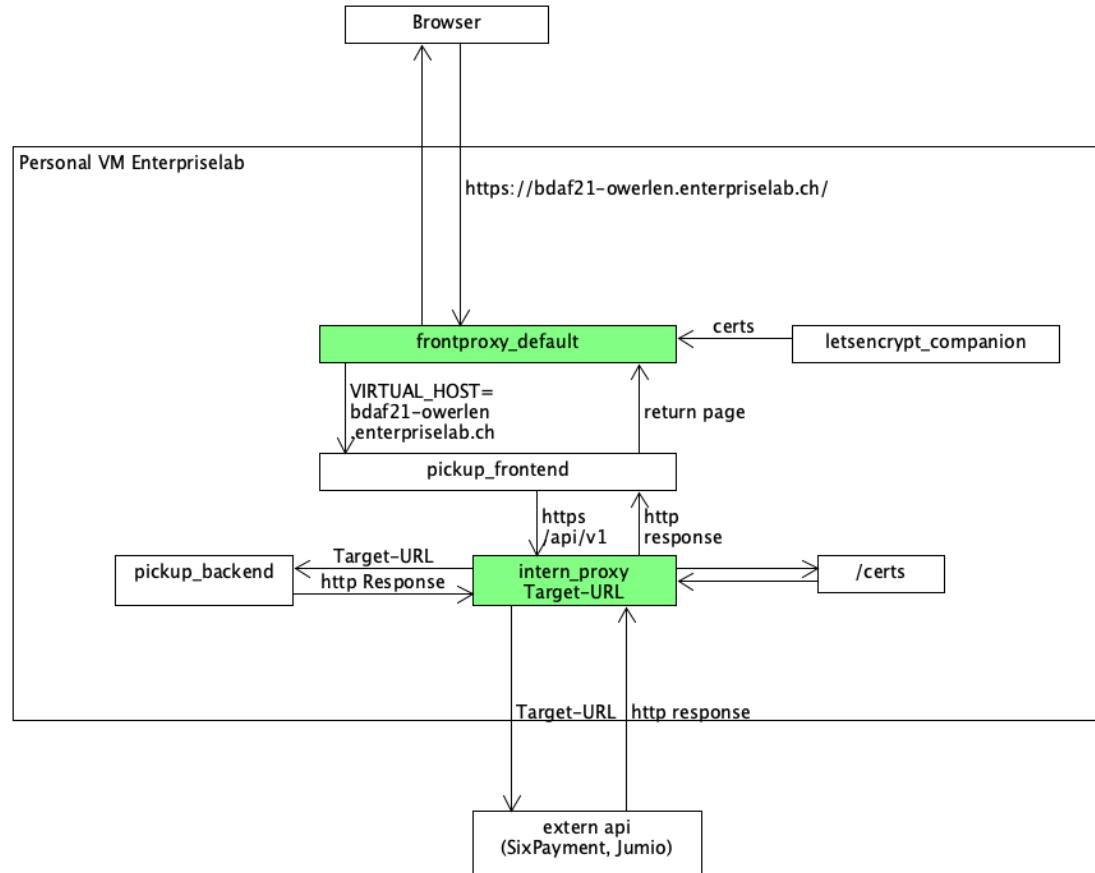


Abbildung 62: Aufbau Produktives System, Quelle: Autoren

Projektstruktur

Backend

Spring Boot Für die Backendentwicklung wurde Spring Boot in der Version 2.3.4 genutzt.

Datenbank

Konfigurationen

Frontend

Backend

C.3 Schnittstellen

C.3.1 Externe Schnittstellen

REST API

C.3.2 Wichtige interne Schnittstellen

Schnittstelle

Steckbrief

Einsatz, Abläufe, Voraussetzungen und Zusicherung

Aufbau und Konfiguration

Fehlerbehandlung

Qualitätsmerkmale

Entwurfsentscheidungen

Beispielverwendung

C.3.3 Benutzerschnittstellen

C.4 Environment-Anforderungen

C.4.1 Hardware

Folgende Hardware wurde für diese Applikation verwendet und kann als ausreichend betrachtet werden:

C.4.2 Software

- Peter

D Software Requirements Specification

D.1 Zweck

Der Auftraggeber will durch die JTI Pickup Station einen neuen Absatzkanal zum Vertrieb seiner Produkte an Endkunden erstellen. Durch das Erstellen einer Softwarelösung soll es möglich sein, Kunden inklusive einer Altersverifikation zu erfassen. Auch ist die Umsetzung einer Kaufabwicklung sowie die Auswahl einer Pick-Up Station Teil dieses Projekts. Die Applikation wird dabei als PWA umgesetzt. Die Software soll dabei mit der physischen Pick-Up Station kompatibel sein. Die Umsetzung von diesen ist Teil von zwei weiteren Bachelorarbeiten an der Hochschule Luzern. "Verweis auf Aufgabenstellung"

D.1.1 Zielgruppe

Zur Zielgruppe dieser Software gehören Kunden und Kundinnen von Japan Tobacco International (JTI) auf der ganzen Welt. Die Software in Kombination mit den physischen Pick-Up Stations soll international eingesetzt werden.

D.1.2 Produktumfang

Der Umfang der Software beginnt bei der Registrierung der Nutzer. Hierbei wird eine bereits vorhandene Alterverifikation eingesetzt, um dies gesetzeskonform Umsetzen zu können. Im Onlineshop werden die verfügbaren Produkte von JTI gelistet. Der Nutzer kann diese Auswählen, anschliessend werden ihm alle Pick-Up Stations, in denen das Produkt verfügbar ist, angezeigt. Der Käufer kann die von ihm gewünschte Station auswählen. Anschliessend wird die Bezahlung per Kreditkarte durchgeführt. In anderen Projekten wurde dabei von JTI bereits ein bekannter Anbieter genutzt. In dieser Software wird darauf zurückgegriffen. Nach erfolgreicher Bezahlung wird ein Code auf dem Gerät des Nutzers gespeichert. Mit diesem kann an der gewünschten Pick-Up Station das bestellte Produkt abgeholt werden.

D.1.3 Definitionen

Risiken Das Risikomanagement wird im Projektmanagementplan in Kapitel B.2.3 detailliert aufgeführt.

D.1.4 Systemübersicht

Die Systemübersicht ist in der Systemspezifikation im Kapitel C zu finden.

D.1.5 Abhängigkeiten

Die Erfüllung der Requirements hängt von diversen Faktoren ab. Wesentlich dabei ist die Abhängigkeit von den Bachelorarbeiten der Studierenden an der Hochschule Luzern Technik und Architektur. Die hier vorhandenen Abhängigkeiten werden während der Realisierung möglichst minimiert.

D.2 Spezifische Anforderungen

D.2.1 Funktionale Anforderungen

F.1	Das System muss die Punkte in der von Google aufgestellten Core Progressive Web App checklist erfüllen. Sam Richard, 2020	Muss
F.2	Das System ist auf eine physische Pick-Up Station abgestimmt.	Muss
F.3	Das System bietet dem Anwender die Möglichkeit, sich zu registrieren und sich anschliessend einzuloggen.	Muss
F.4	Das System bietet die Möglichkeit, durch die Anbindung an eine 3rd Party, eine Altersverifikation durchzuführen.	Muss
F.5	Das System bietet die Möglichkeit, verschiedene Produkte anzuzeigen.	Muss
F.6	Das System bietet die Möglichkeit, verschiedene Produkte dem Warenkorb hinzuzufügen.	Muss
F.7	Das System bietet die Möglichkeit, eine Bestellung dauerhaft zu speichern.	Muss
F.8	Das System bietet dem Kunden die Möglichkeit, verschiedene Produkte zu bestellen.	Kann
F.9	Das System ermöglicht die Anbindung an einen bereits bekannten Bezahlungsdienst, um eine sichere Bezahlung zu garantieren.	Muss
F.10	Das System bietet dem Kunden die Möglichkeit, die für Ihn nächstgelegene Station auswählen zu können.	Muss
F.11	Das System bietet dem Kunden die Möglichkeit, alle vorhandenen Pick-Up Stations anzuzeigen.	Muss
F.12	Das System bietet dem Kunden die Möglichkeit, seine beliebtesten Produkte direkt zu bestellen.	Kann
F.13	Das System bietet dem Dienstleister die Möglichkeit, einen aktuellen Warenbestand zu erhalten.	Kann
F.14	Das System bietet dem Dienstleister die Möglichkeit, bei zu geringem Warenbestand eine Benachrichtigung zu senden.	Muss
F.15	Das System bietet dem Betreiber die Möglichkeit, Artikel hinzuzufügen und Artikel zu bearbeiten.	Kann
F.16	Das System bietet dem Kunden die Möglichkeit, eine Bestellung durch das Einlesen eines Codes an der Pick-Up Station abzuholen.	Muss

Tabelle 26: Funktionale Anforderungen, Quelle: Autoren

D.2.2 Nicht funktionale Anforderungen

ID	Anforderung	Muss/Kann
L.1	Das System soll dem Kunden die Möglichkeit bieten, eine Bestellung mit 5 Klicks zu platzieren	Kann
L.2	Das System bietet die Möglichkeit, International eingesetzt zu werden.	Kann
L.3	Das System muss via HTTPS kommunizieren.	Muss
L.4	Das System muss durch einen modernen und sicheren Authentifizierungsmechanismus geschützt sein.	Muss
L.5	Das System bietet die Möglichkeit, durch die Verwendung von bewährten Programmervorgehen von einem externen Fachmann verstanden zu werden	Muss
L.6	Das System muss über eine CI/CD-Pipeline verfügen	Muss

Tabelle 27: Nicht Funktionale Anforderungen, Quelle: Autoren

D.3 Änderungshistorie

Datum	Änderung	Geändert von	Alte Version
21.03.2021	Aufteilung von Requirements F.6 in F.5, F.6, F.7	Oliver Werlen	Anhang ??
11.05.2021	Hinzufügen des Requirements F.16	Oliver Werlen	

Tabelle 28: Änderungshistorie der Anforderungen, Quelle: Autoren

D.4 Bemerkungen

Als Grundstruktur für die SRS wurde eine Vorlage von Perforce genutzt. [Krüger, 2018] Als Basis dazu diente die IEEE Spezifikation 29148-2018. [Doran, 2018]

D.5 Unterschriften

Mit der Unterschrift gilt die Software Requirement Specification als bestätigt.

Ort, Datum: _____

Visum: _____

E Sitzungsprotokolle

Auf den nachfolgenden Seiten sind alle Protokolle von den durchgeführten Sitzungen ersichtlich.

E.1 23.02.2021

Kick Off Meeting

E.1.1 Ordnungsaufruf

Eine Besprechung aller Projektbeteiligten fand online als Zoom-Meeting am 23.02.2021 um 15:00 Uhr statt.

E.1.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber	-

Tabelle 29: Sitzungsprotokoll, Quelle: Autoren

E.1.3 Genehmigung des Protokolls

Es handelt sich hierbei um die erste Sitzung in diesem Projekt. Es ist noch kein Protokoll vorhanden.

E.1.4 Ankündigungen

Es handelt sich hierbei um das Kickoff Meeting. Als Ziel wird die Finalisierung der Aufgabenstellung genannt.

E.1.5 besprochene Punkte

Aufgabenstellung

- Register, Altersüberprüfung -> Bereits vorhanden, Shop (Brands, Produkt), Warenkorb, Bezahlung mit Kreditkarte, Beim Kauf definieren, wo abgeholt werden soll, reserviert in PickUp Station, An PickUp-> Mittels QR Code, Bestellung ausgeben
- Zahlung mit Twint, Kreditkarte, keine Nachnahme -> data trans, payrex
- Lösungen bereits in Onlineshop
- Postautomat als Beispiel
- Automat muss wissen, welche Artikel er noch hat
- Preis muss Variabel sein, Gratis Paket
- Website als Informationsquelle

Sitzungen

- Abhängig von Projektphase, alle 2-3 Wochen
- Zwischenpräsentation von 20 Min, Resultate vorstellen, Fragen
- Schlusspräsentation 20-25 Minuten

Fragen zur Dokumentation

- Benutzen von vorhandenen Texten aus WiPro?
- Auftrag Start WiPro auch in BAA?

E.1.6 Tagesordnung der nächsten Sitzung

- Besprechung Anforderungen
- Rahmenplan, erste Doku Teile

E.1.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____

Visum: _____

E.2 04.03.2021

E.2.1 Ordnungsaufruf

Eine Besprechung mit den Auftraggebern fand online als Microsoft-Teams-Meeting am 04.03.2021 um 15:00 Uhr statt.

E.2.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber	Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson

Tabelle 30: Sitzungsprotokoll, Quelle: Autoren

E.2.3 Genehmigung des Protokolls

Es handelt sich hierbei um die erste offiziell zu protokollierende Sitzung in diesem Projekt. Auf eine Genehmigung des Protokolls zum Kick-Off Meeting E.1 wird daher verzichtet.

E.2.4 Ankündigungen

In diesem Meeting werden offene Punkte besprochen, welche in der Initialisierungsphase aufgetaucht sind.

E.2.5 besprochene Punkte

Deployment

- Wo soll die Applikation laufen? Enterpriselab für Entwicklung ausreichend, evt. Deployment auf Umgebung von JTI
- Vorgabe vom Auftraggeber?

vorhandene Anbieter

- Anbieter für Altersverifikation? Winston-Camel Registrieren -> Swisscom, Sunrise
- Neue Lösung mittels -> Jumio
- Anbieter für Bezahlvorgang? Datatrans, Payment von Six Payment
- Kontakt zu Experten von JTI

Requirements

- Durchgehen, finalisieren, erweitern
- F.13 Im Kiosk im Aussenbereich -> Nachfüllanfrage an Kioskbetreiber -> Muss Features
- Ändern und hinzufügen von Produkten

Probleme

- Probleme bei aktuellem Absatzkanal -> Wichtigsten Punkte gefunden, AVEC mit Ihrem Produkt reinnehmen.

Kommunikation der Pick-Up Station

- Internet und Strom vorhanden
- Einlesen des Abholcodes mit RFID

E.2.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung wird in zwei Wochen angesetzt.

E.2.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____ Visum: _____

E.3 11.03.2021

E.3.1 Ordnungsaufruf

Eine Besprechung mit der Betreuungsperson fand online als Zoom-Meeting am 04.03.2021 um 13:00 Uhr statt.

E.3.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Handschuh Michael, Betreuungsperson	Meier Rene, Betreuungsperson Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber

Tabelle 31: Sitzungsprotokoll, Quelle: Autoren

E.3.3 Genehmigung des Protokolls

Es handelt sich hierbei um die erste offiziell zu protokollierende Sitzung in diesem Projekt. Auf eine Genehmigung des Protokolls zum Kick-Off Meeting E.1 wird daher verzichtet.

E.3.4 Ankündigungen

In diesem Meeting werden offene Punkte besprochen, welche in der Initialisierungsphase aufgetaucht sind.

E.3.5 besprochene Punkte

Allgemeine Punkte

- Schnittstelle spezifizieren mit ET und Maschinen evt. als Meilenstein
- Meilenstein Zwischenpräsentation
- Meilenstein Abschlusspräsentation
- Wieso NFC und nicht QR-Code
- Datenschutz bei jumio ->
- Schnittstelle 3rd Party Systeme in Risikoanalyse
- Datenschutz bei 3rd Party
- Doku im github Michael Handschuh freigeben

E.3.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit dem Betreuer wird in 2 Wochen stattfinden.

E.3.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____ Visum: _____

E.4 18.03.2021

E.4.1 Ordnungsaufruf

Eine Besprechung mit den Auftraggebern fand online als Microsoft-Teams-Meeting am 18.03.2021 um 13:00 Uhr statt.

E.4.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber	Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson

Tabelle 32: Sitzungsprotokoll, Quelle: Autoren

E.4.3 Genehmigung des Protokolls

Das Protokoll der letzten Sitzung wurde von allen Anwesenden bestätigt

E.4.4 Ankündigungen

In diesem Meeting wird der Abschluss der Initialisierungsphase besprochen. Zusätzlich werden die offenen Punkte zu 5.2.1 besprochen und finalisiert.

E.4.5 besprochene Punkte

Spezifikation Schnittstelle Abholung

- Problem mit NFC/Bluetooth
- Vorstellen der Alternativen
- Besprechung Alternativen -> Alternative 1 wird bevorzugt, QR-Code auf Maschine
- Wenn keine Bestellung, dann auf Webpage
- Produkte, Liste von Produkten wird gesendet, jtiproducts.ch sind alle Produkte vorhanden
- Keine Policy von JTI vorhanden, jtiproducts als Referenz
- Statusmeldungen an Nachfüller

Schnittstelle Produktbestand

- Finalisierung mit Philipp Arnold individuell
- Übertragung mittels JSON Format

E.4.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit den Auftraggebern findet in zwei Wochen statt.

E.4.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____ Visum: _____

E.5 25.03.2021

E.5.1 Ordnungsaufruf

Eine Besprechung mit der Betreuungsperson fand online als Zoom-Meeting am 25.03.2021 um 13:00 Uhr statt.

E.5.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Handschuh Michael, Betreuungsperson	Meier Rene, Betreuungsperson Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber

Tabelle 33: Sitzungsprotokoll, Quelle: Autoren

E.5.3 Genehmigung des Protokolls

Es handelt sich hierbei um die erste offiziell zu protokollierende Sitzung in diesem Projekt. Auf eine Genehmigung des Protokolls zum Kick-Off Meeting E.1 wird daher verzichtet.

E.5.4 Ankündigungen

In diesem Meeting werden offene Punkte besprochen, welche in der Initialisierungsphase aufgetaucht sind.

E.5.5 besprochene Punkte

CI/CD

- Vorzeigen der Pipeline

Besprochene Punkte mit Auftraggebern

- QR-Code anstatt NFC
- Liste mit allen verfügbaren Produkten bereitgestellt.

IoT Allgemein

- Webserver auf Pick-Up Station, um Busy Waiting zu vermeiden
- Pick Up meldet auf Webserver -> No-IP
- Genug Power, um sichere Kommunikation zu gewährleistet
- Web-Hock Prinzip
- State machine um Szenarien durchzugehen, Produkt nicht Abholung bereit
- Experte Stefan Bernet
- sbernet@icloud.com

E.5.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit dem Betreuer wird in 2 Wochen stattfinden.

E.5.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____

Visum: _____

E.6 01.04.2021

E.6.1 Ordnungsaufruf

Eine Besprechung mit den Auftraggebern fand online als Microsoft-Teams-Meeting am 01.01.2021 um 13:30 Uhr statt.

E.6.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber	Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson

Tabelle 34: Sitzungsprotokoll, Quelle: Autoren

E.6.3 Genehmigung des Protokolls

Das Protokoll der letzten Sitzung wurde von allen Anwesenden bestätigt

E.6.4 Ankündigungen

Es wurden keine Ankündigungen geplant

E.6.5 besprochene Punkte

Content der Page

- Qualität der Bilder für Desktop kritisch
- Beschreibung der Produkte
- AGBs
- Content Allgemein, Vorgegeben vom Auftraggeber

E.6.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit den Auftraggebern findet in zwei Wochen statt.

E.6.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____

Visum: _____

E.7 15.04.2021

E.7.1 Ordnungsaufruf

Eine Besprechung mit dem Projektbetreuer fand online als Zoom-Meeting am 15.01.2021 um 13:00 Uhr statt.

E.7.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson Oliver Werlen, Projektleiter	Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber

Tabelle 35: Sitzungsprotokoll, Quelle: Autoren

E.7.3 Genehmigung des Protokolls

Das Protokoll der letzten Sitzung wurde von allen Anwesenden bestätigt

E.7.4 Ankündigungen

Es wurden keine Ankündigungen geplant

E.7.5 besprochene Punkte

Zwischenpräsentation

- Content -> Allgemein Zusammen
- Vorgehen

Fokus auf Funktionalität

- Tests
- Dokumentation

E.7.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit den Auftraggebern findet in zwei Wochen statt.

E.7.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____

Visum: _____

E.8 15.04.2021

E.8.1 Ordnungsaufruf

Eine Besprechung mit dem Projektbetreuer fand online als Zoom-Meeting am 15.01.2021 um 13:00 Uhr statt.

E.8.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Oliver Werlen, Projektleiter Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber	Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson

Tabelle 36: Sitzungsprotokoll, Quelle: Autoren

E.8.3 Genehmigung des Protokolls

Das Protokoll der letzten Sitzung wurde von allen Anwesenden bestätigt

E.8.4 Ankündigungen

Es wurden keine Ankündigungen geplant

E.8.5 besprochene Punkte

Allgemein

- Zuerst Auswahl von Pick Up Station, danach die Produkte
- Bestellung wiederholen
- ProduktNr. auf Liste vorhanden

E.8.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit den Auftraggebern findet in zwei Wochen statt.

E.8.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____

Visum: _____

E.9 12.05.2021

E.9.1 Ordnungsaufruf

Eine Besprechung mit dem Projektbetreuer fand online als Zoom-Meeting am 12.05.2021 um 13:00 Uhr statt.

E.9.2 Teilnehmer

Anwesende Mitglieder	Nicht anwesende Mitglieder
Meier Rene, Betreuungsperson Handschuh Michael, Betreuungsperson Oliver Werlen, Projektleiter	Philipp Arnold, Elektrotechnik Elena Nujic, Auftraggeber Rolf Hohermut, Auftraggeber

Tabelle 37: Sitzungsprotokoll, Quelle: Autoren

E.9.3 Genehmigung des Protokolls

Das Protokoll der letzten Sitzung wurde von allen Anwesenden bestätigt

E.9.4 Ankündigungen

Es wurden keine Ankündigungen geplant

E.9.5 besprochene Punkte

Planung Abschlusspräsentation

- Fixierung auf Woche von 14. Juni

Projektstand

- Keine Probleme

Lösung des Kommunikationsproblems

- VPN
- Dokumentation

E.9.6 Tagesordnung der nächsten Sitzung

Die nächste Sitzung mit der Betreuungsperson findet in zwei Wochen statt.

E.9.7 Unterschriften

Mit der Unterschrift gilt das Sitzungsprotokoll als bestätigt.

Ort, Datum: _____

Visum: _____

F Rahmenpläne

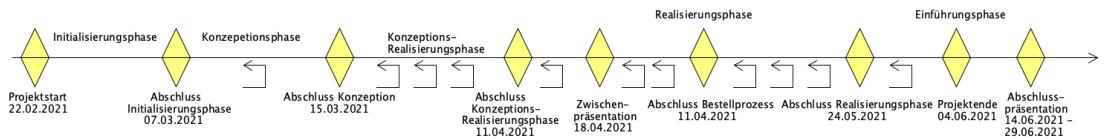


Abbildung 63: Rahmenplan Version 1,

Quelle: Autoren

G Originale Aufgabenstellung

H Zwischenpräsentation

Zwischenpräsentation JTI PickUp Station

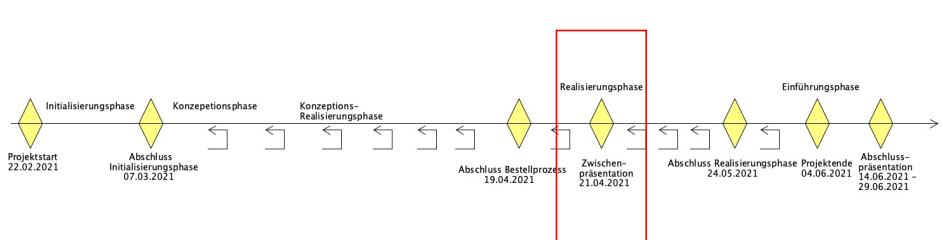
Oliver Werlen

21. April 2021

Inhalt

- Vorgehen
 - Projektstand
 - Technologien
 - Probleme
- Demo
- Technische Details
 - Infrastruktur
 - Transport Layer Security
 - Autorisierung
- Nächste Schritte
 - Kommunikation mit PickUp Stations
 - Progressive Web App
 - Div. Wunsch-Features vom Auftraggeber

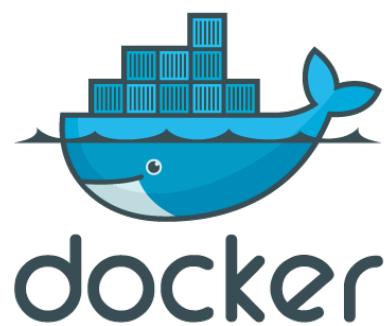
Projektstand



- Software Development agile (SoDa) Zeitstrahl
- Montag: Meilenstein 3 erreicht
 - Grösstenteils erfüllt
 - Anbindung an Jumio zur Alterverifikation
 - Dafür aber bereits Pick Up Funktionalität in Backend

Technologien

- Spring Boot
- Angular
- Docker
- MariaDB
- LaTEX
- Material UI



Probleme

Infrastruktur

- Letzte Woche, EnterpriseLab Maschine kaputt gegangen
- Nicht wirklich nachvollziehbar
- Deployment musste neu aufgebaut werden

CORS

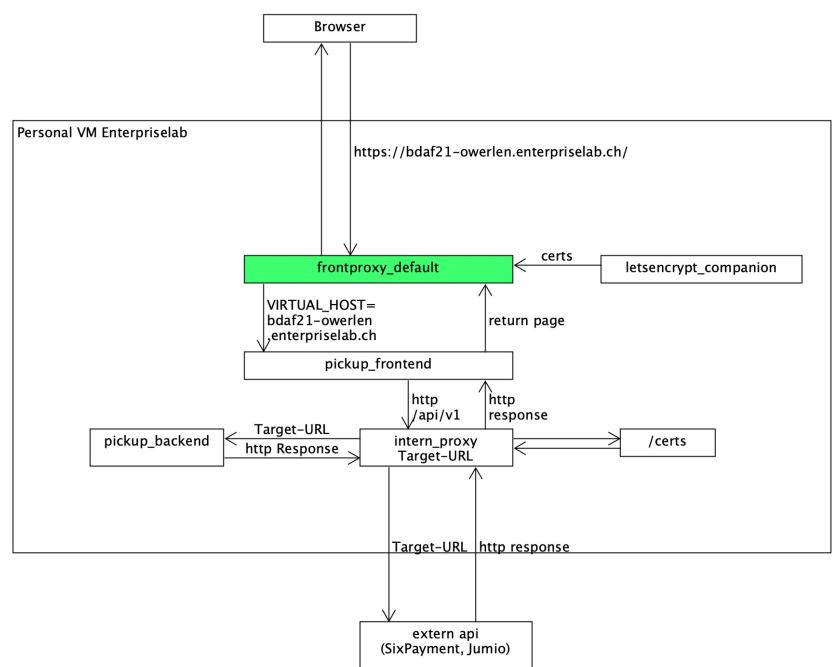
- Proxy benötig, um Anbindung an 3rd Party API zu ermöglichen
 - Six Payment
 - Jumio
- Via Node-Proxy gelöst
- 127.0.0.1 bei Safari nicht Vertrauenswürdig



Demo

Technische Details

- Reverse Proxy für TLS
- Interner Proxy zur Kommunikation mit API's
- Alles lokal, daher keine zusätzliche TLS nötig



Autorisierung

- JSON Web Token
- Rollenbasiert
 - Admin
 - User
 - Maintenance
- Spring Security

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJp  
YXd1cmxlbiIsImlhdCI6MTYxODQyMzQ3M  
ywiZXhwIjoxNjE4NTA50DczfQ.IwSI_Mq  
7KsvS0scamA1UlQ72W8sVcXmAHzIlMie9  
gsKTt18RXDpc4v4rIV-  
guEF5sSgL4YHYx7Fr0poiLvamSg|
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS512"  
}
```

PAYOUT: DATA

```
{  
  "sub": "iawerlen",  
  "iat": 1618423473,  
  "exp": 1618509873  
}
```

VERIFY SIGNATURE

```
HMACSHA512(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
) □ secret base64 encoded
```

Nächste Schritte

- 
- Kommunikation mit PickUp Station
 - Progressive Web App mit Angular
 - Div. Kleinere Wunsch-Features des Auftraggebers nach Zeit
 - Merkliste
 - Schnelle Bestellung

Quellen

- <https://bgasparotto.com/change-spring-boot-server-port>
- <https://www.trendreport.de/docker-im-hinblick-auf-devops/>
- <https://icon-icons.com/icon/angular-logo/169598>
- <https://www.informatik-aktuell.de/betrieb/datenbanken/mariadb-interview-mit-ceo-patrik-sallner.html>

D Software Requirements Specification

D.1 Zweck

Der Auftraggeber will durch die JTI Pickup Station einen neuen Absatzkanal zum Vertrieb seiner Produkte an Endkunden erstellen. Durch das Erstellen einer Softwarelösung soll es möglich sein, Kunden inklusive einer Altersverifikation zu erfassen. Auch ist die Umsetzung einer Kaufabwicklung sowie die Auswahl einer Pick-Up Station Teil dieses Projekts. Die Applikation wird dabei als PWA umgesetzt. Die Software soll dabei mit der physischen Pick-Up Station kompatibel sein. Die Umsetzung von diesen ist Teil von zwei weiteren Bachelorarbeiten an der Hochschule Luzern. "Verweis auf Aufgabenstellung"

D.1.1 Zielgruppe

Zur Zielgruppe dieser Software gehören Kunden und Kundinnen von Japan Tobacco International (JTI) auf der ganzen Welt. Die Software in Kombination mit den physischen Pick-Up Stations soll international eingesetzt werden.

D.1.2 Produktumfang

Der Umfang der Software beginnt bei der Registrierung der Nutzer. Hierbei wird eine bereits vorhandene Alterverifikation eingesetzt, um dies gesetzeskonform Umsetzen zu können. Im Onlineshop werden die verfügbaren Produkte von JTI gelistet. Der Nutzer kann diese Auswählen, anschliessend werden ihm alle Pick-Up Stations, in denen das Produkt verfügbar ist, angezeigt. Der Käufer kann die von ihm gewünschte Station auswählen. Anschliessend wird die Bezahlung per Kreditkarte durchgeführt. In anderen Projekten wurde dabei von JTI bereits ein bekannter Anbieter genutzt. In dieser Software wird darauf zurückgegriffen. Nach erfolgreicher Bezahlung wird ein Code auf dem Gerät des Nutzers gespeichert. Mit diesem kann an der gewünschten Pick-Up Station das bestellte Produkt abgeholt werden.

D.1.3 Definitionen

Risiken Das Risikomanagement wird im Projektmanagementplan in Kapitel B.2.3 detailliert aufgeführt.

D.1.4 Systemübersicht

Die Systemübersicht ist in der Systemspezifikation im Kapitel C zu finden.

D.1.5 Abhängigkeiten

Die Erfüllung der Requirements hängt von diversen Faktoren ab. Wesentlich dabei ist die Abhängigkeit von den Bachelorarbeiten der Studierenden an der Hochschule Luzern Technik und Architektur. Die hier vorhandenen Abhängigkeiten werden während der Realisierung möglichst minimiert.

D.2 Spezifische Anforderungen

D.2.1 Funktionale Anforderungen

F.1	Das System muss die Punkte in der von Google aufgestellten Core Progressive Web App checklist erfüllen. Sam Richard, 2020	Muss
F.2	Das System ist auf eine physische Pick-Up Station abgestimmt.	Muss
F.4	Das System bietet dem Anwender die Möglichkeit, sich zu registrieren.	Muss
F.5	Das System bietet die Möglichkeit, durch die Anbindung an eine 3rd Party, eine Altersverifikation durchzuführen.	Muss
F.6	Das System bietet dem Kunden die Möglichkeit, verschiedene Produkte zu bestellen.	Kann
F.7	Das System ermöglicht die Anbindung an einen bereits bekannten Bezahlungsdienst, um eine sichere Bezahlung zu garantieren.	Muss
F.8	Das System bietet dem Kunden die Möglichkeit, verschiedene Produkte zu bestellen.	Kann
F.9	Das System bietet dem Kunden die Möglichkeit, die für Ihn nächstgelegene Station auswählen zu können.	Muss
F.10	Das System bietet dem Kunden die Möglichkeit, alle vorhandenen Pick-Up Stations anzuzeigen.	Muss
F.11	Das System bietet dem Kunden die Möglichkeit, seine beliebtesten Produkte direkt zu bestellen.	Kann
F.12	Das System bietet dem Dienstleister die Möglichkeit, einen aktuellen Warenbestand zu erhalten.	Kann
F.13	Das System bietet dem Dienstleister die Möglichkeit, bei zu geringem Warenbestand eine Benachrichtigung zu senden	Muss
F.14	Das System bietet dem Betreiber die Möglichkeit, Artikel hinzuzufügen und Artikel zu bearbeiten	Kann

Tabelle 9: Funktionale Anforderungen, Quelle: Autoren

D.2.2 Nicht funktionale Anforderungen

ID	Anforderung	Muss/Kann
L.1	Das System soll dem Kunden die Möglichkeit bieten, eine Bestellung mit 5 Klicks zu platzieren	Kann
L.2	Das System bietet die Möglichkeit, International eingesetzt zu werden.	Kann
L.3	Das System muss via HTTPS kommunizieren	Muss
L.4	Das System muss durch einen modernen und sicheren Authentifizierungsmechanismus geschützt sein	Muss
L.5	Das System bietet die Möglichkeit, durch die Verwendung von bewährten Programmervorgehen von einem externen Fachmann verstanden zu werden	Muss

Tabelle 10: Nicht funktionale Anforderungen, Quelle: Autoren

D.3 Bemerkungen

Als Grundstruktur für die SRS wurde eine Vorlage von Perforce genutzt. [Krüger, 2018] Als Basis dazu diente die IEEE Spezifikation 29148-2018. [Doran, 2018]

D.4 Unterschriften

Mit der Unterschrift gilt die Software Requirement Specification als bestätigt.

Ort, Datum: _____

Visum: _____