

**Projektarbeit**

# **Interner Onlineshop**



**Dozent:** Bruno Hammer  
**Klasse:** L-TIN-18-Fr-a  
**Datum:** 18. September 2019

**Studenten:** - Sven Wetter, [sven.wetter@edu.teko.ch](mailto:sven.wetter@edu.teko.ch)  
- Boban Petrovic, [boban.petrovic@edu.teko.ch](mailto:boban.petrovic@edu.teko.ch)



In Zusammenarbeit mit **swisscom**

## Inhaltsverzeichnis

1. Vorwort.....	4
2. Richtlinien.....	5
3. Aufgabenstellung .....	5
4. Projektorganisation .....	6
5. Management Summary .....	6
6. Analyseteil.....	7
6.1. Grundlage.....	7
6.2. Vorstudie .....	7
6.3. Konzept .....	8
6.4. Lastenheft.....	10
6.4.1. Funktionale Anforderung .....	10
6.4.1.1. Weboberfläche.....	10
6.4.1.2. Sicherheit.....	10
6.4.1.3. Sprache.....	10
6.4.2. Nicht-funktionale Anforderungen.....	10
6.4.2.1. Bedienbarkeit.....	10
6.4.2.2. Browser.....	10
6.5. Pflichtenheft.....	10
6.6. Meilensteine .....	11
6.7. Zeitplan.....	11
6.7.1. Umsetzung.....	11
6.7.2. Ergebnis.....	11
7. Designteil .....	12
7.1. Lösungsweg und Komponenten .....	12
7.2. Implementierung.....	12
7.2.1. Backend.....	12
7.2.1.1. Spring-Configuration.....	12
7.2.1.2. Rest-Services .....	13
7.2.1.3. Datenmodell.....	13
7.2.1.4. Datenbankzugriffe .....	13
7.2.1.5. Services .....	14
7.2.2. Frontend.....	14
7.3. Datenbank .....	17
7.4. Testen.....	18

7.4.1.	Testfälle und Testprotokoll .....	18
7.4.2.	Testbericht .....	18
8.	Schlussteil .....	19
8.1.	Schlussfolgerungen .....	19
8.2.	Verbesserungsvorschläge .....	19
8.3.	Schwierigkeiten .....	19
8.3.1.	Backend .....	19
8.3.2.	Frontend .....	19
8.4.	Ausblick .....	20
8.5.	Schlusswort .....	20
9.	Anhang .....	21
9.1.	Git Repository .....	21
9.2.	Projektantrag .....	21
9.3.	Besprechungseinladung 1 .....	22
9.4.	Protokoll der Besprechung 1 .....	22
9.5.	Besprechungseinladung 2 .....	22
9.6.	Protokoll der Besprechung 2 .....	23
9.7.	Installationsanleitung .....	23
9.8.	Abbildungsverzeichnis .....	23
9.9.	Quellenverzeichnis .....	23

## 1. Vorwort

Das Studium verlangt regelmäßige Projektarbeiten der Studenten, damit die Fähigkeiten validiert, gefestigt und verbessert werden können. Beim ersten Projekt sollen zweier Teams gebildet werden, sofern möglich. Damit alle von dem Projekt gleichermassen profitieren können macht es Sinn, wenn sich ein erfahrener Student mit einem weniger erfahrenen Studenten zusammenschliesst. Den Studenten steht diese Möglichkeit zwar offen, jedoch können natürlich auch zwei unerfahrene Studenten zusammenarbeiten.

Für dieses Projekt haben sich, Boban Petrovic, Software Engineer bei Credit Suisse und Sven Wetter, DevOps Engineer bei Swisscom, zusammengeschlossen.

Da die meisten der Studenten für die erste Projektarbeit ein Spiel alà TicTacToe, 4-Gewinnt oder ähnliches ausgewählt haben, haben wir uns für einen simplen internen Mitarbeitershop entschieden. Wir wollten für uns ein Thema auswählen, das der Realität entspricht. Das Projekt ist daher sogar ein wenig wirtschaftlich orientiert und macht uns mit diversen Tools & Frameworks vertraut, welche auch in der täglichen Arbeitswelt oft genutzt werden.

## 2. Richtlinien

Die TEKO Schweizerische Fachschule hat uns folgende Richtlinien auferlegt:

Umfang:

- Der Umfang der Projektarbeit soll rund 40 Stunden pro Person betragen.

Rahmenbedingung:

- Die Arbeit erfolgt grundsätzlich in 2er Team. Ausnahmen benötigen die Zustimmung der Projektbetreuung.

Abgabe Termin:

- Ein Exemplar muss bis spätestens am 20.09.2019 17:00 beim Sekretariat der TEKO abgegeben werden.

Projektsitzungen:

- 07.06.2019
- 23.08.2019

Präsentation:

Für die Präsentation stehen 2 verschiedene Daten zur Auswahl:

- 27.09.2019 16:15
- 04.10.2019 16:15

## 3. Aufgabenstellung

Die TEKO verlangt, dass wir ein Thema aus einem Ausbildungsgebiet einreichen. Die Themen müssen aus mindestens einem der folgenden Bereiche stammen:

- Software Engineering
- Datenbank
- Systemtechnik
- Netzwerktechnik

## 4. Projektorganisation

In diesem Projekt sind drei Personen involviert. Für die Entwicklung und Instandhaltung des Projektes sind Sven Wetter und Petrovic Boban zuständig.



Abbildung 1: Projektorganisation

## 5. Management Summary

Der Online-Mitarbeitershop ist ein simpler Webshop auf Basis des Spring Frameworks. Trotz der Einfachheit und der Tatsache, dass der Shop in der Realwirtschaft keine große Zukunft hätte, war es uns ein Anliegen, die wichtigen Aspekte wie Sicherheit oder Stabilität nicht außer Acht zu lassen.

Der Webshop ist für die Mitarbeiter nur auf dem lokalen Netzwerk erreichbar, setzt aber trotzdem eine erfolgreiche Authentifizierung mittels Benutzer und Password voraus. Nach erfolgreichem Login kann der Mitarbeiter diverse Produkte bestellen. Nach erfolgter Bestellung wird die Bestellung in der Datenbank abgelegt und eine Bestätigungs-SMS wird versendet. Die Bestellung muss anschliessend von der internen Einkaufsabteilung bestätigt werden. Auf Features wie "Benutzerkonto verwalten", "Bestellung verfolgen" usw. haben wir aufgrund der hohen Komplexität & des grossen Zeitaufwands verzichtet.

Für die Infrastruktur und somit für den Betrieb wird ein Raspberry Pi 3b <sup>1</sup> verwendet. Für den Versand von SMS haben wir von Svens Arbeitgeber, der Swisscom (Schweiz) AG, einen Benutzer erhalten. Dieser ist bis am 30.09.2019 gültig und aktiv.

## 6. Analyseteil

### 6.1. Grundlage

Die Grundlage des Projekts bildete die Idee, einen Webshop für Mitarbeiter zu entwickeln. Aufgrund Bobans jahrelanger Erfahrung als Software Entwickler konnten wir noch vor dem Antrag abschätzen, ob eine Umsetzung überhaupt möglich ist. Bei der Abschätzung hat Boban bewusst Svens Fähigkeiten berücksichtigt, da Sven bisher noch keine nennenswerten Erfahrungen in der Entwicklung verfügt. Ganz nach dem Motto "Ein Team ist nur so stark wie das schwächste Glied". Wir haben uns daher auch entschieden, dass Sven hauptsächlich das Backend entwickeln wird und Boban das Frontend.

Als nächsten Schritt wollten wir in einer Vorstudie, respektive einem "Proof of Concept" herausfinden, "was" und "wie einfach" etwas umzusetzen ist. Aber auch, ob das schwächste Glied sich gut einarbeiten kann und grundlegendes Verständnis mitbringt.

### 6.2. Vorstudie

Bevor wir jedoch mit dem Proof of Concept<sup>2</sup> angefangen haben, haben wir zuerst ein Brainstorming gemacht. Dort haben wir Ideen, Wünsche und Möglichkeiten festgehalten.

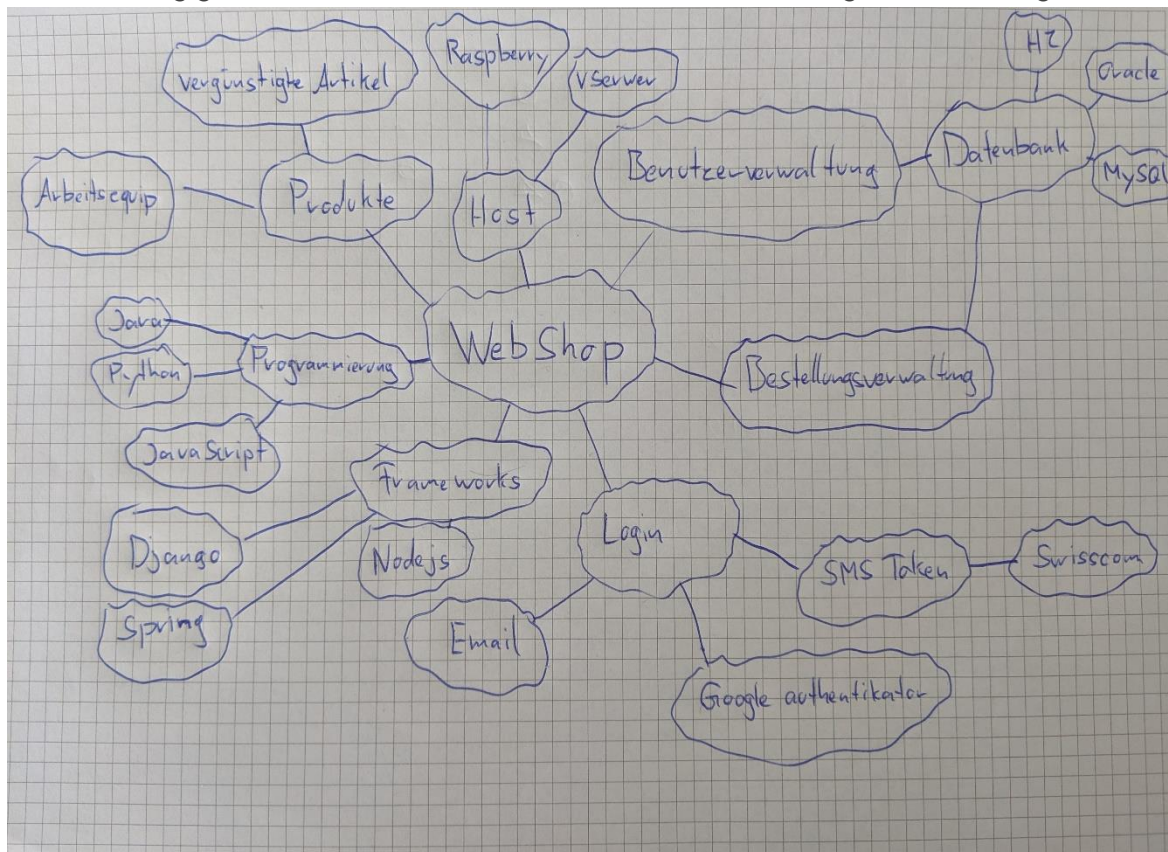


Abbildung 2: Mindmap

Im nächsten Schritt haben wir mit dem eigentlichen PoC begonnen und versucht, die Funktionen und Features als Prototypen umzusetzen. Dadurch konnten wir einen guten Einblick über die Komplexität und den Umfang des Projekts gewinnen. Danach waren wir in der Lage abzuleiten, welche Funktionen & Features schlussendlich im Projekt umgesetzt werden können.

### 6.3. Konzept

Der Mitarbeiter-Onlineshop soll nur über ein internes Netz erreichbar sein. Zudem soll die Webpage alle gängigen Browser unterstützen. Dadurch, dass die Seite nur lokal erreichbar ist, müssen wir nicht mit einem hohen Verbindungsvolumen rechnen und es reicht eine simple Infrastruktur. Die Daten des Shops sollen in einer einfachen Datenbank abgebildet, respektive gespeichert werden.

Aufgrund des einfachen Konzepts sowie der vorhergegangenen Vorstudie haben wir uns für folgendes entschieden:

- Die Webseite und folglich das Backend wird mit dem Spring-Framework <sup>3</sup>umgesetzt.
- Die Umsetzung des Frontend erfolgt mit Angular 5.
- Als Programmiersprache verwenden wir Java 8.
- Das Session Handling wird von Spring im Backend erfolgen.
- Die Authentifizierung soll im 2-Faktor-Auth Verfahren mittels SMS Token erfolgen.
- Der Mitarbeiter soll die Bestellungen einsehen und verwalten, respektive stornieren können. Die Bestellung wird danach an den Einkauf weitergeleitet, welcher die Bestellung bestätigt und auslöst.
- Für die Infrastruktur verwenden wir ein Raspberry Pi 3b.
- Die Datenbank soll auf dem gleichen Host laufen wie der Webshop
- Für die Datenbank nutzen wir H2 (Hibernate) In-Memory.

Für das Frontendkonzept haben wir 2 Mockups erstellt, ein für die Login-Seite und das andere für die eigentliche Shop-Seite.

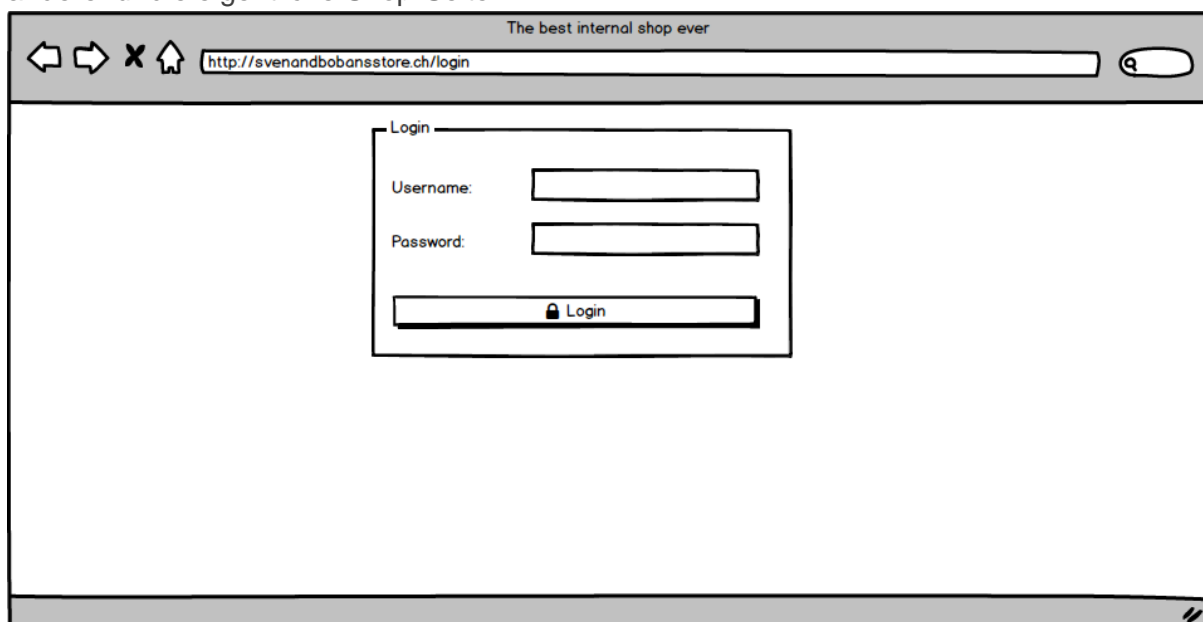
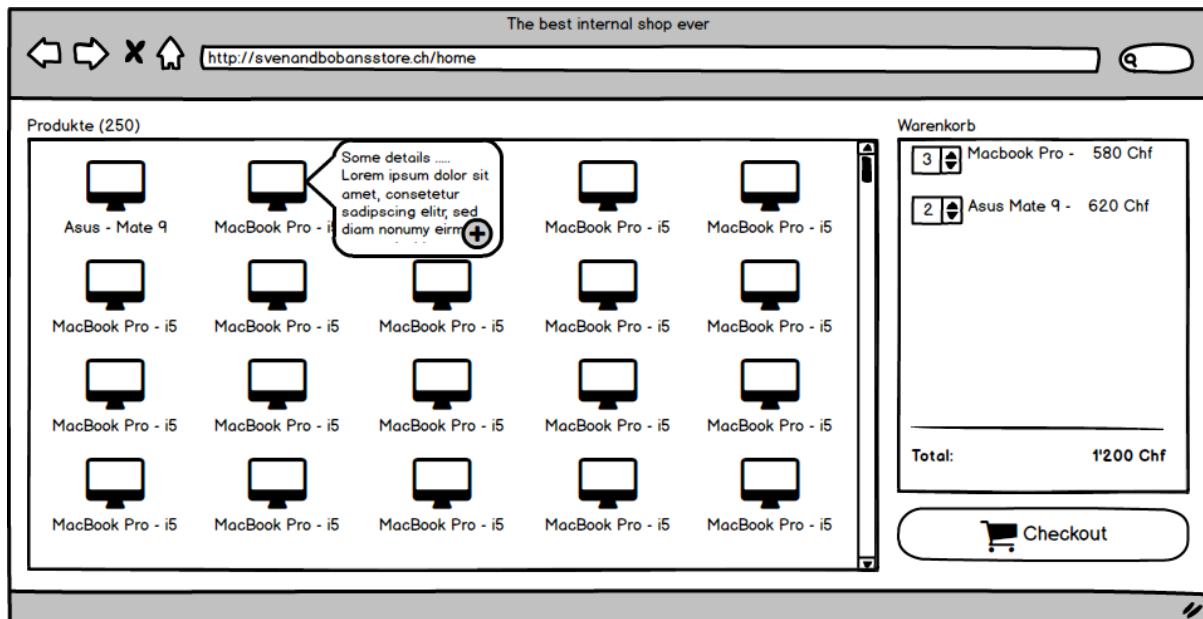


Abbildung 3: Mockup Loginseite





#### Abbildung 4: Mockup Webshop

Da wir während der Entwicklung des Projekts zur Erkenntnis gekommen sind, dass gewisse Funktionen so nicht sinnvoll sind oder zu viel Zeit benötigen, haben wir im Nachhinein entsprechend einige Anpassungen im Konzept vorgenommen. Folgende Punkte wurden abgeändert:

Änderung	Begründung
Die Authentifizierung erfolgt lediglich über Benutzername und Passwort.	Die Umsetzung des 2-Faktor-Auth ist zu zeitintensiv und nicht unbedingt nötig. Der Mitarbeitershop soll ohnehin nur im lokalen Netz erreichbar sein.
Es wird keine Bestellungsverwaltung implementiert.	Nach der Bestellung wird ein Eintrag in der Datenbank erstellt. Der Einkauf soll nach Anbindung des Shops direkten Zugang zu den Bestellungen erhalten und die eigentliche Bestellung auslösen.
Die Datenbank wird als File und nicht als In-Memory umgesetzt.	Die Daten sind nur im Arbeitsspeicher vorhanden. Die Daten müssten zusätzlich als SQL-Statement Daten gesichert werden, da sie sonst nach einem Neustart nicht mehr vorhanden wären. Der Performance-Gewinn durch die In-Memory Funktion ist für dieses Projekt nicht von Bedeutung.

## 6.4. Lastenheft

### 6.4.1. Funktionale Anforderung

#### 6.4.1.1. Weboberfläche

Ein Webshop setzt eine Benutzeroberfläche, respektive Weboberfläche voraus. Diese soll in einem schlichten einfachen Design erfolgen.

#### 6.4.1.2. Sicherheit

Der Webserver soll im Lokalen Netz erreichbar sein. Die Nutzung des Mitarbeitershops setzt aber eine erfolgreiche Authentifizierung voraus.

#### 6.4.1.3. Sprache

Auf die Mehrsprachigkeit wird verzichtet. Stattdessen wird der Shop in Englisch umgesetzt.

### 6.4.2. Nicht-funktionale Anforderungen

#### 6.4.2.1. Bedienbarkeit

Die Bedienung des Webshops soll so einfach wie möglich sein. Es soll vor allem darauf geachtet werden, nicht zu viele Funktionen bereitzustellen (Komplexität).

#### 6.4.2.2. Browser

Der Webshop soll von den gängigsten Browsern genutzt werden können. Dazu gehören unter anderem Chrome, Safari, Mozilla und Microsoft Edge.

## 6.5. Pflichtenheft

- Der Webshop muss im Internet erreichbar sein.
- Nach Abschluss der Bestellung soll eine SMS Notifikation ausgelöst werden
- Die Authentifizierung der Benutzer erfolgt über die registrierte E-Mail-Adresse und über ein Passwort
- Das Session-Handling soll im Frontend und nicht im Backend erfolgen. Die Bestellung wird folglich per REST-Call an das Backend übermittelt.
- Um Kosten zu sparen soll ein Raspberry Pi als Webserver zum Einsatz kommen.
- Die Ladezeit der Website soll nicht länger als 4 Sekunden dauern. Die Seitenwechsel bzw. der Bestellprozess und der damit verbundene Backendvorgang soll ebenfalls nicht länger als 4 Sekunden pro Prozess dauern.
- Das Backend soll mit dem Framework "Spring" umgesetzt werden das Frontend mit Angular 6.

## 6.6. Meilensteine

Die Meilensteine repräsentieren den aktuellen Fortschritt des Projekts. Die Meilensteine sollten Grundsätzlich erreicht werden., können aber in Ausnahmefällen auch vorher abgeschlossen sein. Für unser Projekt haben wir folgende sechs Meilensteine definiert:

Id	Meilenstein	Plantermin	Ist-Termin	Status
M1	Projektstart	27.05.2019	27.05.2019	Erledigt
M2	Proof of Concept	14.06.2019	14.06.2019	Erledigt
M3	Backend fertig	31.08.2019	06.09.2019	Mit Verzögerung erledigt
M4	Frontend fertig	31.08.2019	07.09.2019	Mit Verzögerung erledigt
M5	Dokumentation	18.09.2019	19.09.2019	Mit leichter Verzögerung erledigt
M6	Projektabschluss	20.09.2019	20.09.2019	Erledigt

## 6.7. Zeitplan

### 6.7.1. Umsetzung

Der Zeitplan ist ein ausgezeichnetes Instrument, um den Erfolg des Projekts abzuschätzen. Anfänglich war unser Zeitplan zu detailliert, daher haben wir diesen verworfen und einen neuen erstellt. Im ersten Zeitplan hatten wir beispielsweise die Rubrik Backend in verschiedene Unterpunkte aufgeteilt und jeweils detaillierte Angaben dazu gemacht. Dies wurde im neuen Zeitplan jetzt einfach halber unter "Backend" zusammengefasst.

### 6.7.2. Ergebnis

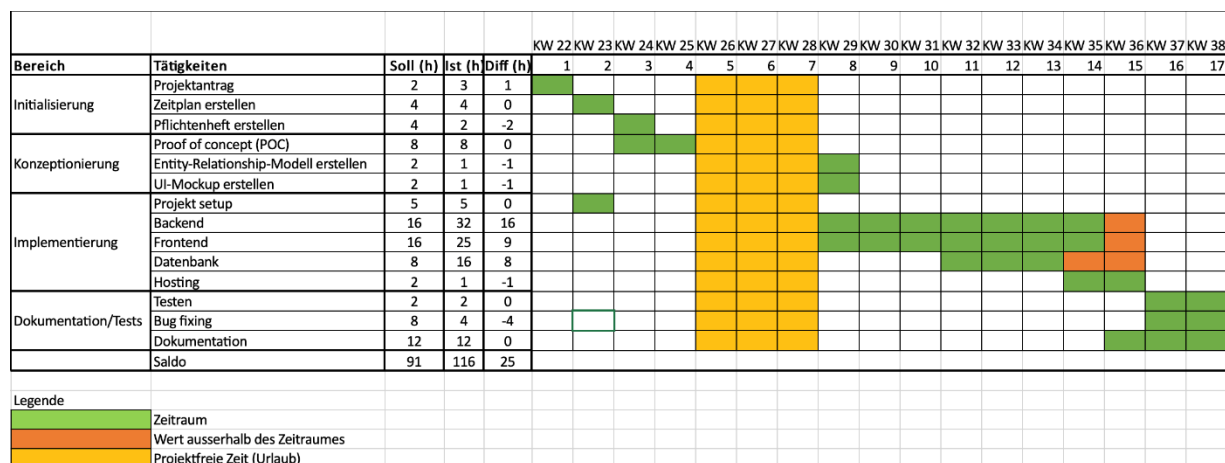


Abbildung 5: Zeitplan

Alles in allem konnten wir den Zeitplan bis auf die vom Projekt vorhergesehenen Hauptpunkte Backend/Frontend & Datenbank einhalten. Geplant war ein Aufwand von 91 Stunden, verteilt auf beide Personen. Der Effektive Aufwand betrug 116 Stunden und dauerte somit 25 Stunden länger als geplant.

Gewisse Punkte wie das Backend, Frontend und Datenbank dauerten erheblich länger, während andere Punkte wie Bug fixing, Hosting etc weniger lang dauerten.

Sven benötigte aufgrund der mangelnden Erfahrung mehr Zeit und war daher zum Teil auf Bobans Hilfe angewiesen. Dies wirkte sich somit auch gleich auf die Hauptarbeit von Boban am Frontend aus. Die Planung des Backend wirkte sich somit auch direkt auf das Frontend aus. Nichtsdestotrotz konnte das Projekt fristgerecht fertiggestellt und abgegeben werden.

## 7. Designteil

### 7.1. Lösungsweg und Komponenten

Nach der Analyse und der Konzepterstellung mussten wir uns einig werden, welche Frameworks, Tools und Programmiersprachen verwendet werden sollen. Da wir für die Programmierung die Sprache Java in der Version 8 ausgewählt haben, entschieden wir uns für das Backend das Framework Spring-Boot zu nehmen. Im Gegensatz zu JSF (Java Server Faces) oder Grails nimmt uns Spring eine Menge Arbeit ab.

Für den Frontend Teil haben wir das Framework Ionic genutzt, welches hauptsächlich auf Angular basiert und dem heutigen Standard entspricht.

Für die Speicherung der Daten wurde die H2 Datenbank verwendet. Dadurch, dass H2 unkompliziert in unsere Entwicklungsumgebung IntelliJ IDE<sup>4</sup> eingebunden werden konnte, ermöglichte es uns eine einfache Erstellung der Datenbank.

### 7.2. Implementierung

#### 7.2.1. Backend



Abbildung 6: Spring-Boot

Die ganze Backend-Implementierung basiert auf dem Spring-Boot Framework. Das Framework hilft uns dabei, viele Punkte zu vereinfachen und möglichst schnell eine Applikation zu entwickeln, die auch produktiv einsetzbar ist.

##### 7.2.1.1. Spring-Configuration

###### MvcConfig

In der MvcConfig Klasse wird unsere Login-Seite registriert. Da wir das Session Handling von Spring machen lassen, haben wir uns entschieden, direkt auch die Default-Login-Page von Spring zu nutzen.

###### WebConfiguration

In der WebConfiguration Klasse haben wir die Einstellungen für die Datenbank Konsole vorgenommen. Diese erlaubt uns, über den Browser auf die Webkonsole zuzugreifen.

## WebSecurity

Das Spring Framework bringt von Haus aus eine Security <sup>5</sup>Implementation mit. Dies ermöglicht es uns, die Sicherheitseinstellungen ohne großen Aufwand zu verändern, respektive unseren Bedürfnissen anzupassen. Unsere Konfiguration ist so eingestellt, dass die Benutzer beim Start der Applikation direkt in einen Zwischenspeicher geladen werden. Zudem ist es für nicht Authentifizierte Benutzer nur möglich die /login Seite aufzurufen, andere Aufrufe werden automatisch auf die Login-Seite umgeleitet.

### 7.2.1.2. Rest-Services

#### CartRestController

Alle GET und POST Methoden, die etwas mit dem Warenkorb zu tun haben, sind mit diesem Controller gemappet. So konnte die Logik für das Hinzufügen und Entfernen von Produkten und das zwischenspeichern des Warenkorbs umgesetzt werden. Auch der Bestellabschluss und das damit verbundene versenden einer SMS wird in diesem Controller abgehandelt.

#### LoginController

Grundsätzlich hätten wir den LoginController auch gleich im MvcConfig umsetzen können. Wir wollten aber aus Designgründen eine eigene Controller Klasse dafür. Im LoginController mappen wir die GET-Aufrufe auf die im MvcConfig registrierte Login Default Webpage.

#### ProductRestController

Dieser RestController enthält die implementierten GET-Methoden, um die Produkte sowie deren Details wie Name, Beschreibung und Preis von der Datenbank herunterzuladen. Beim Aufruf des Webshops macht das Frontend ein GET-Aufruf und holt alle Produkte von der Datenbank.

### 7.2.1.3. Datenmodell

Bei diesem Package macht es keinen Sinn, wenn wir alle Klasse auflisten würden. Wir haben für jede Datenbanktabelle eine Model-Klasse (Cart, Order, Product, User) erstellt. Diese werden von den verschiedenen JPA Repositories benötigt, die den Zugriff auf die Datenbank ermöglichen. Damit das Repository weiss, wie die Tabellen und Spalten benannt sind, werden diese in den Model-Klassen deklariert.

### 7.2.1.4. Datenbankzugriffe

In diesem Repository haben wir für jede Tabelle ein Repository eingerichtet. Da die Repository wie bei den Model-Klassen ziemlich gleich aufgebaut sind, erübrigen sich die einzelnen Erklärungen. Grundsätzlich ermöglicht uns JPA <sup>6</sup>den einfachen Zugriff auf die Datenbank. Es ist sogar in der Lage, mit Schlüsselwörtern eine Methode zu Implementieren. So kann zum Bsp. eine Interface Methode `removeByUserId()` erstellt werden und JPA macht aus den Schlüsselwörtern im Methodennamen "REMOVE" "BY" "USERID" direkt eine Implementation respektive ein passendes SQL Statement dazu.

Dies funktioniert jedoch nicht in allen Fällen, so haben wir bei jeder Methode, die nicht automatisch von JPA implementiert werden konnte, einen SQL Query mitgegeben.

### 7.2.1.5. Services

#### SmsSenderImpl

Für die 2 Faktor Authentifizierung haben wir eine Interface Implementation für das Versenden von SMS über Swisscom erstellt. Die Idee der 2-Faktor Authentifizierung haben wir jedoch wieder verworfen und uns entschieden, den Service als Bestellungsbestätigung zu benutzen.

Die Implementation führt eine POST Methode via REST auf die Swisscom SMS Server aus, um eine SMS zu versenden. Die ganzen Inhalte wie Absender, Empfänger und Text werden von uns im JSON-Format übergeben.

#### UserServiceImpl

Damit die benötigten Benutzerinformationen abgefragt werden können, haben wir in dieser Klasse 2 Methoden implementiert. Zum einen haben wir die Methode `getCurrentUserId()`, die es uns ermöglicht, die Benutzer ID eines Webbesuchers zu ermitteln. Dies brauchen wir, da das Frontend keinerlei Daten über den Benutzer gespeichert hat. Nur so ist es uns möglich, den Warenkorb auch einem Benutzer zu zuordnen.

Des Weiteren haben wir noch die Methode `getMobileFromCurrentUser()`. Dadurch erhalten wir die Mobilnummer eines Benutzers und können somit beim Kaufabschluss das SMS auslösen.

### 7.2.2. Frontend

Für die Realisierung der Benutzeroberfläche wurde das Webframework Ionic<sup>7</sup> gewählt. Das Framework basiert auf Angular und Apache Cordova. Der grösste Vorteil des Frameworks ist es, dass die plattformübergreifende Entwicklung mobiler Anwendungen sichergestellt wird. So sieht die Anwendung auf jedem Gerät anders aus.



Abbildung 7: Hybrider Look

Die Applikation ist in folgende Bereiche aufgeteilt:

- Login
- Produktübersicht
- Produktdetail
- Warenkorb

#### Login

Die Startseite der Applikation ist die Loginseite. Bei einem fehlerhaften Login wird dem User klar gemacht, dass seine Logindaten nicht stimmen. Nach dem erfolgreichen Login wird eine eigene Session aufgebaut und man wird automatisch zur Produktübersicht weitergeleitet.

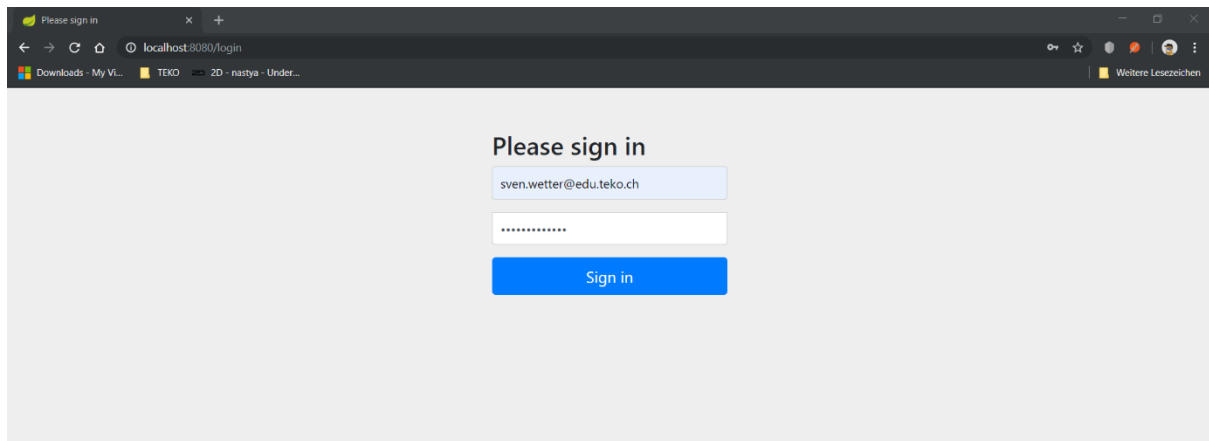


Abbildung 8: Login

## Produktübersicht

Auf der Produktübersicht-Seite werden alle verfügbaren Produkte angezeigt. Mit Hilfe der Suchleiste kann der Benutzer nach bestimmten Produkten suchen. Die Suchkriterien sind entweder der Produktname oder irgendein Wort aus der Produktbeschreibung.

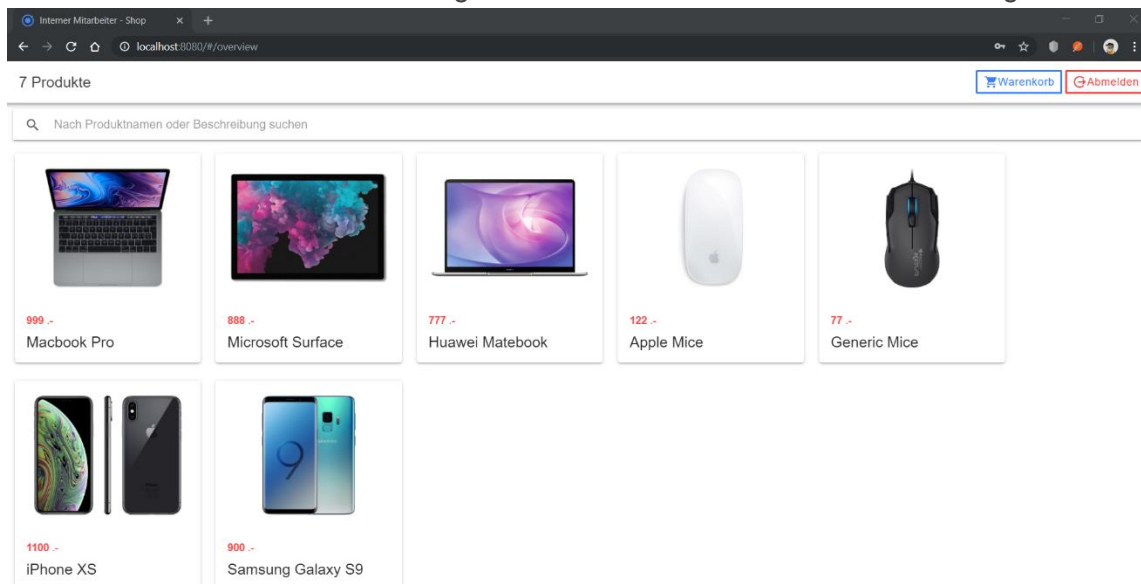


Abbildung 9: Produktübersicht

Die Produktdetails-Seite zeigt folgende Eigenschaften vom Produkt an:

- Anzahl Bewertungen
- Kurzbeschreibung
- Detailbeschreibung
- Preis
- Drei Produktfotos

Ausserdem kann auf dieser Seite das ausgewählte Produkt dem Warenkorb hinzugefügt werden.

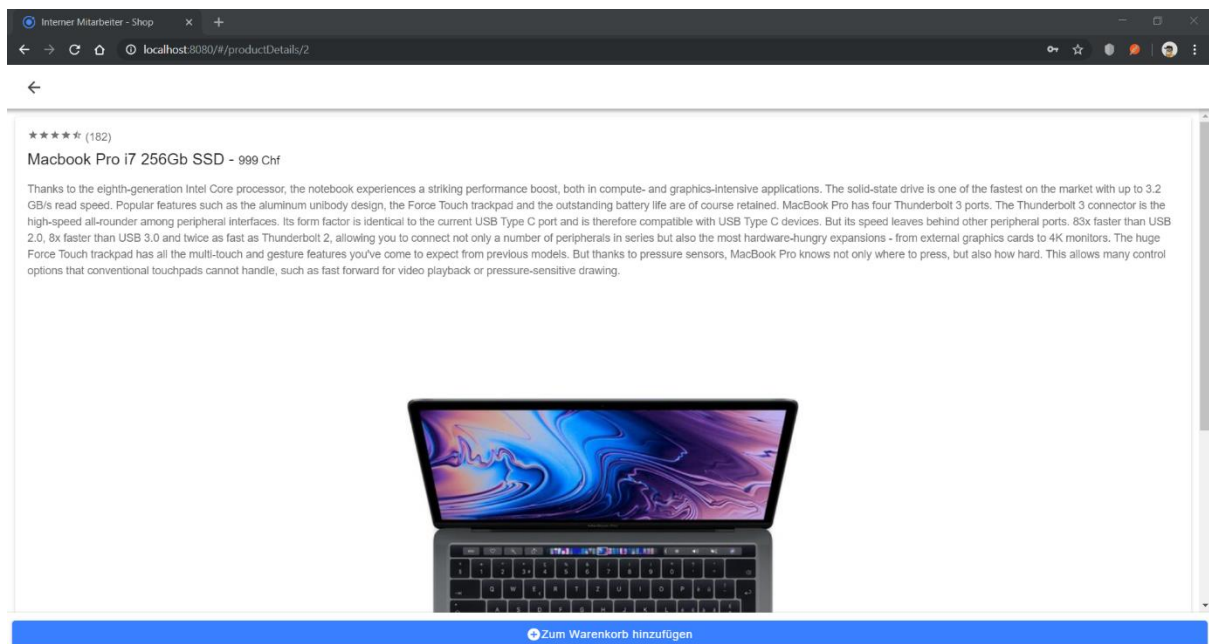



Abbildung 10: Produktsicht

## Warenkorb

Der Warenkorb listet alle Produkte inklusive Anzahl auf, die zuvor vom User dem Warenkorb hinzugefügt wurden. Mit einem Wischzug nach Links () kann man die Einträge aus dem Warenkorb entfernen. Durch Klicken des Buttons „Kaufen“ wird der Bestellvorgang ausgelöst und der Benutzer erhält ein Bestätigungs-SMS auf seinem Handy.

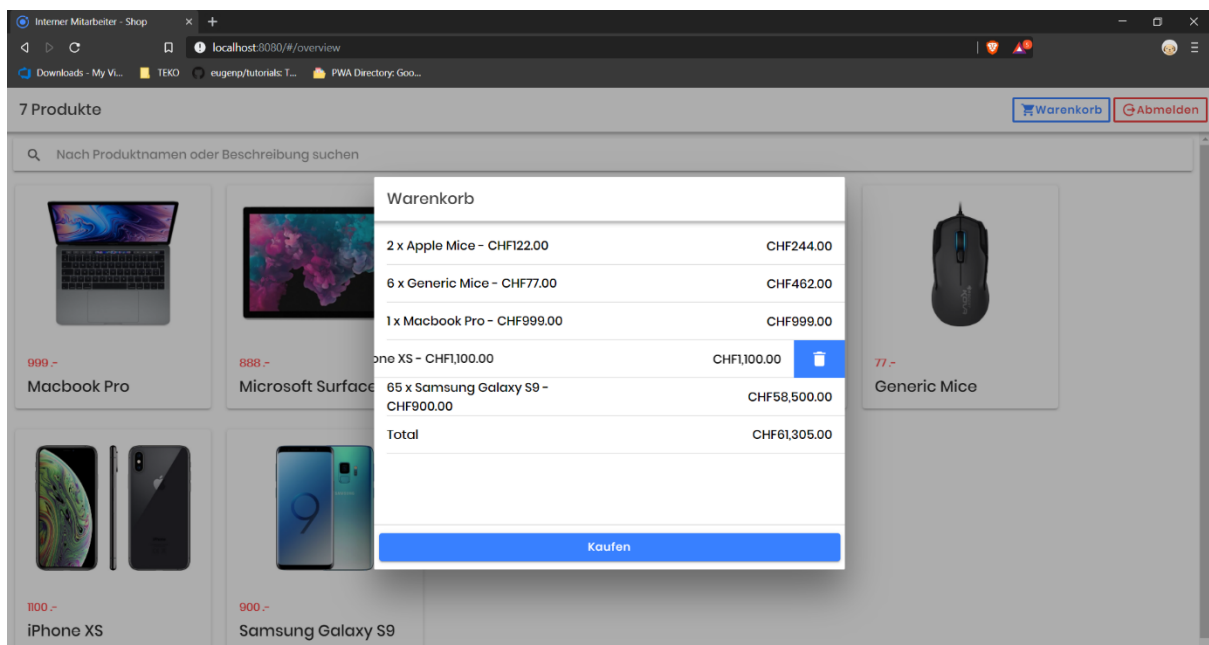


Abbildung 11: Warenkorb



### 7.3. Datenbank

Das System der Datenbank ist H2<sup>8</sup>, welche in der Praxis oftmals als eine In-Memory-Datenbank gebraucht wird. In unserem Fall haben wir eine File-Datenbank gemacht, damit wir die Daten nach dem Server Stopp nicht verlieren.

Für den gesamten Onlineshop sind nur vier Tabellen nötig (Cart, User, Product und Orders).

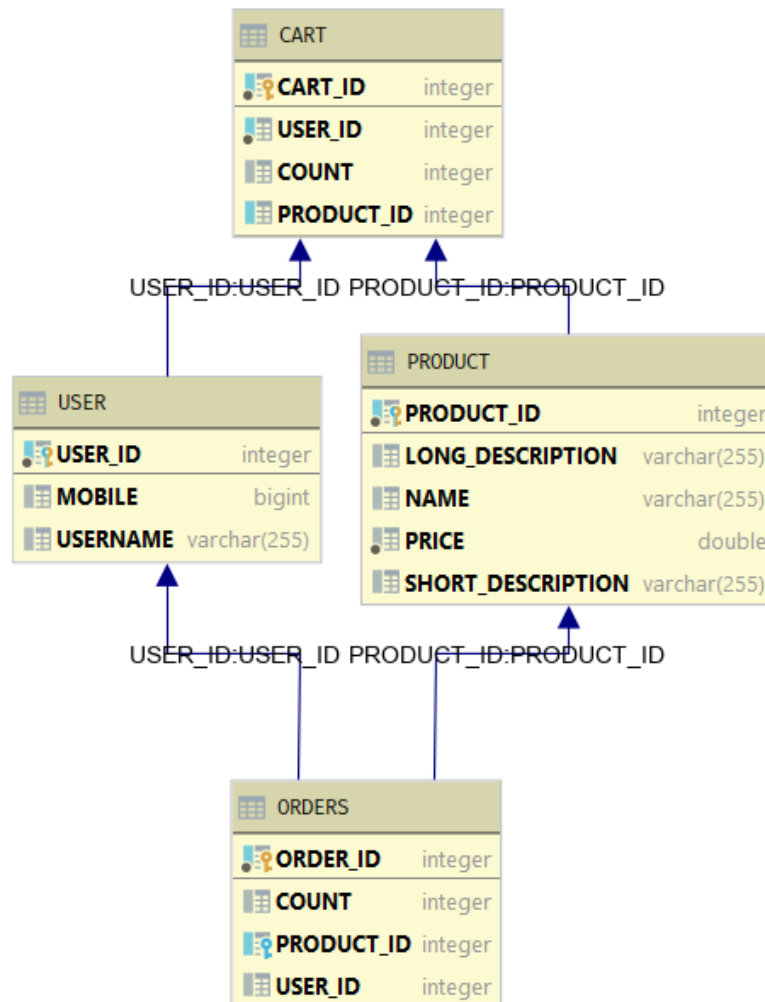


Abbildung 12: Datenbankstruktur

## 7.4. Testen

### 7.4.1. Testfälle und Testprotokoll

**Tester:** Petrovic Boban

**Datum:** 17.09.2019, 19:30 Uhr

ID	Beschreibung	Erwartetes Resultat	Vorbedingungen	Resultat
1	User loggt sich in die Applikation ein	Der User wird auf die Produktübersicht weitergeleitet	User muss in der Datenbank existieren	✓
2	User loggt sich in die Applikation ein mit falschen Daten	Eine Fehlermeldung wird angezeigt	User muss in der Datenbank existieren und das Passwort muss falsch sein	✓
3	User klickt auf ein Produkt	Der User kommt auf die Detailseite des angeklickten Produktes	User muss angemeldet sein	✓
4	User fügt ein Produkt zum Warenkorb hinzu	Eine Bestätigungsmeldung wird angezeigt	User muss angemeldet sein	✓
5	User gibt im Suchfeld auf der Produktübersichtsseite „microsoft“ ein	Es wird nur ein Produkt angezeigt	User muss angemeldet sein	✓
6	User streicht (swiped) im Warenkorb nach links über ein Produkt	Das Produkt wird aus dem Warenkorb gelöscht	User muss angemeldet sein und das Produkt muss im Warenkorb sein	✓
7	User klickt auf „kaufen“ im Warenkorb	Der Kaufprozess wird ausgelöst und eine SMS wird an den User gesendet	User muss angemeldet sein und das Produkt muss im Warenkorb sein	✓
8	User klickt auf den Knopf „Abmelden“	Die Session wird zerstört und der User wird auf die Login Seite weitergeleitet	User muss angemeldet sein	✓

### 7.4.2. Testbericht

Alle acht Testfälle sind erfolgreich durchgeführt worden und es gab keine Probleme während der Testausführung. Die Applikation kann nun ohne Bedenken dem Kunden übergeben werden.

## 8. Schlussteil

### 8.1. Schlussfolgerungen

Das Ziel, einen funktionierenden Webshop für Mitarbeiter inklusive SMS Notifikation und Authentifizierung via Username und Passwort zu erstellen, haben wir erreicht. Der Shop wurde in einem schlichten aber modernem Design umgesetzt. Wir haben erreicht, dass innerhalb kurzer Zeit ein Webshop zur Verfügung steht, ohne auf Aspekte wie Sicherheit oder Stabilität zu verzichten. Trotzdem haben wir den Aufwand unterschätzt und sind im Nachhinein froh, dass wir nicht noch mehr optionale Features in unsere Anforderungen aufgenommen haben.

### 8.2. Verbesserungsvorschläge

Das Design wurde schlicht aber modern umgesetzt. Wir hätten aber als zusätzliches Feature noch das Firmenlogo platzieren können, auch wenn es nur ein intern zugänglicher Shop ist.

Ein weiterer Verbesserungspunkt wäre, dass der angemeldete Benutzer im Header mit seinem Profilbild angezeigt wird.

### 8.3. Schwierigkeiten

#### 8.3.1. Backend

Das Spring Framework bringt von Haus aus sehr viele Implementierte Features mit, dies hat uns eine Menge Arbeit erspart. So mussten wir uns nicht um das Session Handling oder um die Zugriffe der Datenbankverbindungen kümmern. Auch auf REST Basis mussten wir nur noch die Logik, nicht aber die Steuerung selbst implementieren. Trotzdem ist so ein Webshop für Mitarbeiter ein nicht zu unterschätzender Aufwand.

Mangels Erfahrung hat Sven viele Methoden entwickelt, die am Schluss wieder gestrichen wurden, da diese bereits von Spring implementiert waren. So hat er anfänglich die Logik der Datenbankzugriffe selbst implementiert. Das zwar einerseits sehr lehrreich für Sven, jedoch wurde so sehr viel Zeit in unnötige Arbeit gesteckt. Das hätte verhindert werden können, wenn sich Sven mehr mit der Dokumentation des Frameworks beschäftigt hätte.

Weitere Schwierigkeiten gab es bei der Login-Implementation. Anfänglich konnten wir uns nicht authentifizieren, obwohl aus unserer Sicht alles korrekt umgesetzt wurde. Nach langer Recherche wurden wir dann auf einen Bug aufmerksam, der verhindert, dass Passwörter richtig entschlüsselt werden, wenn sie mit \$2b\$ oder \$2y\$ beginnen anstatt mit \$2a\$.

#### 8.3.2. Frontend

Da die Applikation mit Spring-Security abgesichert wurde, war es für die Entwicklung des UI's sehr aufwendig, die Änderungen im „hot-reload“ Modus zu testen. Das Problem tritt aufgrund von CORS (Cross-Origin Resource Sharing) auf, da das UI auf einem eigenen Server gelaufen ist und das Backend wiederum auf einem anderen. Man musste immer die ganze Applikation mit Maven erstellt und anschließend testen, dies dauerte meistens bis zu zwei Minuten und war daher sehr zeitintensiv.

#### 8.4. Ausblick

Der Mitarbeitershop ist nun fertig, die Infrastruktur der Firma fehlt jedoch noch. Man müsste den Shop so umbauen, dass er sich die Benutzer vom Active Directory holen könnte. Die Einkaufsabteilung oder der Vorgesetzte würde ein Interface benötigen, um die Bestellungen zu bestätigen oder abzulehnen. Zudem wäre eine Bestellhistorie oder Bestellverwaltung für die Mitarbeiter sicherlich von Vorteil. So könnte der Mitarbeiter beispielsweise eine Bestellung stornieren, wenn er etwas falsch bestellt hat.

#### 8.5. Schlusswort

Die Umsetzung der Arbeit war für uns beide sehr interessant. Vor allem Sven konnte durch die Arbeit seine Fähigkeiten in der Entwicklung mit Java deutlich verbessern. Doch auch Boban, der bereits sehr viel Erfahrung mitbringt, hat von der Arbeit profitiert und viele neue wertvolle Erfahrungen gesammelt. Zudem ist es ein tolles Gefühl, wenn man nach harter Arbeit das Resultat sieht.

Was wir beide für nächste Projekte bestimmt mitnehmen werden ist, dass die Dokumentation nicht zu unterschätzen ist und sehr viel Zeit in Anspruch nimmt. Damit sollte man auf jeden Fall frühzeitig beginnen.

Trotzdem haben wir es geschafft, ein komplexes Projekt innerhalb der angegebenen Frist umzusetzen, Darauf sind wir sehr stolz.

## 9. Anhang

### 9.1. Git Repository

Das ganze Projekt findet man unter folgender URL

<https://elad.ch/gitblit/summary/~petrovib!svenboban!projektarbeit.git>

### 9.2. Projektantrag

# PROJEKTANTRAG

Mitarbeitershop

Erstellungsdatum: 07.06.2019  
 Empfänger: Bruno Hammer

**Studenteninformationen / Projektmitglieder**

Name, Vorname	Wetter, Sven	Petrovic, Boban
Klasse	L-TIN-18-Fr-a	L-TIN-18-Fr-a
Rolle	Stv. Projektleiter und Entwickler	Projektleiter und Entwickler
Adresse	Martinshöhe 1a 6204 Sempach	Weiherrmatte 12 6102 Malters
Telefon	079 846 4541	076 383 0439
Email	<a href="mailto:sven.wetter@swisscom.com">sven.wetter@swisscom.com</a>	<a href="mailto:boban_95@hotmail.de">boban_95@hotmail.de</a>

**Ausgangslage / Problemstellung**

Die Firma Nixcom (Schweiz) AG bietet zurzeit für die Mitarbeiter keine großen Benefits. Um diese Lücke zu schließen, soll ein Mitarbeitershop entwickelt werden in dem die Mitarbeitenden diverse IT Artikel vergünstigt erwerben können. Die Auswahl der Produkte wird vom Produktmanagement-Team getroffen und laufend aktualisiert. Da wir keine Kooperation mit anderen Onlineshops eingehen müssen, können wir die Artikel noch günstiger anbieten. Da lediglich ausschließlich unsere Mitarbeiter Kunden sein können, entfällt der Registrierungsprozess da alle Mitarbeiter vorgängig in der Datenbank erfasst sind. Damit der Shop benutzt werden kann, muss man sich mittels der Personalnummer und einem One-Time-Password (SMS) authentifizieren.

**Ziel der Arbeit / Persönliche Ziele**

\* Responsiver Mitarbeitershop mit diversen verfügbaren Produkten

**Meilensteine**

1. Onlineshop in schlichtem Design inkl Produktlisting  
 2. Datenbank mit Mitarbeiter Informationen  
 3. Authentifizierung mittels SMS Token  
 4. Fertigstellung

**Projekttermine**

24. Mai 2019, 18:00 Uhr	Deadline Projektantrag
27. Mai 2019	Start der Projektarbeit
7. Juni 2019, 16:30 Uhr	Besprechungstermin 1
23. August 2019	Besprechungstermin 2
20. September 2019, 17:00 Uhr	Projektabgabe

**Aufwand- / Kostenplanung**

Um dieses Vorhaben umzusetzen benötigen wir zwei interne Informatiker mit Datenbank und Programmierkenntnissen. Für die SMS Authentifizierung wird zusätzlich ein „Large Account“ von der Firma Swisscom benötigt. Der WebServer wird auf unserer eigenen Infrastruktur betrieben (Raspberry Pi).

Budgetaufteilung

Interne Personalkosten	12'000 CHF (80 Std)
Interne Infrastrukturkosten:	1'500 CHF
Externe Kosten:	500 CHF

**Auftraggeber**

Auftraggeber ist die Nixcom (Schweiz) AG, Kontaktperson: Jonas Müller, Teamleiter IT-Abteilung ([jonas.mueller@nixcom.ch](mailto:jonas.mueller@nixcom.ch))

Datum, Ort: \_\_\_\_\_

Unterschrift und bewilligt: \_\_\_\_\_

### 9.3. Besprechungseinladung 1

**Teilnehmer:** Hammer Bruno, Wetter Sven, Petrovic Boban

**Datum:** 14.06.2019, 16:30 Uhr

**Sitzungsort:** TEKO Luzern, Zimmer 5.3

**Traktandenliste:**

- Wo hin kommt der Zeitplan im Dokument?
- Kommt jedes Detail in den Zeitplan? z.B. Besprechungseinladung erstellen...
- Wie soll der Aufbau des Berichts sein?
- Wie sollen wir am besten Vorgehen?
- Im welchem Teil kommen die Arbeitsjournale hin? (Analyse, Design oder Schlussteil)
- Wie soll der Anhang aussehen?
- Muss der Anhang in der Projektarbeit vorhanden sein, oder genügt auch ein Verweis auf **GIT** o.ä?
- Muss der Projektantrag ebenfalls in den Anhang?

### 9.4. Protokoll der Besprechung 1

**Teilnehmer:** Hammer Bruno, Wetter Sven, Petrovic Boban

**Datum:** 14.06.2019, 16:30 Uhr

**Sitzungsort:** TEKO Luzern, Zimmer 5.3

**Vorsitz:** Bruno Gfeller

**Entschuldigte Personen:** keine

**Protokollführer:** Boban Petrovic

**Offene Punkte:** keine

Das Meeting hat gezeigt, dass die Schüler nicht genau wussten, wo die Dokumente für das Projekt abzulegen sind. Es sollte alles ins ELAD hochgeladen werden, damit der Dozent das auch sieht. Zudem wurde besprochen, was genau in das Lasten/Pflichtenheft hineinkommt. Ausserdem war unklar, wie der gesamte Bericht aufgebaut werden soll.

### 9.5. Besprechungseinladung 2

**Teilnehmer:** Hammer Bruno, Wetter Sven, Petrovic Boban

**Datum:** 23.08.2019, 16:30 Uhr

**Sitzungsort:** TEKO Luzern, Zimmer 5.3

**Traktandenliste:**

- Braucht man jeweils ein Lastenheft und ein Pflichtenheft?
- Wann ist der Abgabetermin?
- Gibt es eine gemeinsame Note?
- Muss man konsequent und oft den Code commiten?

## 9.6. Protokoll der Besprechung 2

**Anwesende Personen:** Bruno Wetter, Sven Wetter, Boban Petrovic  
**Datum:** 23.08.2019, 16:30 Uhr  
**Vorsitz:** Samuel Nagbe  
**Entschuldigte Personen:** Philipp  
**Protokollführer:** Boban Petrovic

Es wurde vom Dozenten mitgeteilt, dass wir entweder ein Pflichtenheft oder ein Lastenheft brauchen. Der Code muss nicht ständig committed werden, wichtig ist aber, dass am Schluss alles im GitBlit ist. Der Dozent bewertet alles, was auf GitBlit zu finden ist. Es gibt eine gemeinsame Note für die beiden Studenten.

## 9.7. Installationsanleitung

Die gesamte Installationsanleitung ist im GitBlit hinterlegt und auffindbar unter <https://elad.ch/gitblit/blob/~petrovib!svenboban!projektarbeit.git/master/README.md>

## 9.8. Abbildungsverzeichnis

Abbildung 1: Projektorganisation .....	6
Abbildung 2: Mindmap .....	7
Abbildung 3: Mockup Loginseite.....	8
Abbildung 4: Mockup Webshop.....	9
Abbildung 5: Zeitplan .....	11
Abbildung 6: Spring-Boot.....	12
Abbildung 7: Hybrider Look .....	14
Abbildung 8: Login .....	15
Abbildung 9: Produktübersicht.....	15
Abbildung 10: Produktansicht.....	16
Abbildung 11: Warenkorb .....	16
Abbildung 12: Datenbankstruktur .....	17

## 9.9. Quellenverzeichnis

- 
- <sup>1</sup> [www.raspberrypi.org](http://www.raspberrypi.org)
  - <sup>2</sup> [https://de.wikipedia.org/wiki/Proof\\_of\\_Concept](https://de.wikipedia.org/wiki/Proof_of_Concept)
  - <sup>3</sup> <https://spring.io/>
  - <sup>4</sup> [www.jetbrains.com](http://www.jetbrains.com)
  - <sup>5</sup> <https://www.baeldung.com/security-spring>
  - <sup>6</sup> <https://docs.spring.io/spring-data/jpa/docs/current/reference/html>
  - <sup>7</sup> <https://ionicframework.com>
  - <sup>8</sup> <https://www.h2database.com/>