HTBLVA, Wien V, Spengergasse

Höhere Lehranstalt für Textilindustrie und Datenverarbeitung

Ausbildungsschwerpunkt Höhere Informatik

VORSTUDIEN

Gesamtprojekt

Erstellung eines Prototypen für Selfcheckout per App

Untersuchung und Vergleich von Frontend Frameworks auf deren Eignung für mobile Apps
Alexander Wiener 5BHIF Betreuer: Klaus Unger

Ein Vergleich von JVM Sprachen im Umgang mit modernen Programmierschnittstellen
Florian Freimüller 5BHIF Betreuer: Klaus Unger

Vergleich des Java EE Standard-Frameworks im Vergleich zu Spring zur Entwicklung von Business Applications
Paul Maurovich 5BHIF Betreuer: Klaus Unger

Recherche von best practices von UX Design Guidelines für mobile Apps
Dennis Sima 5BHIF Betreuer: Klaus Unger

Untersuchung und die Gegenüberstellung von Payment Providern und deren APIs
Lucas Walter 5BHIF Betreuer: Klaus Unger

Schuljahr 2020/21

Abgabevermerk:

Datum: übernommen von:

Inhaltsverzeichnis

Management Summary

Das Management Summary fasst die wesentlichen Faktoren eines oder mehrerer umfassender Dokumente sehr komprimiert zusammen.
Dabei werden alle entscheidungsrelevanten Dinge beachtet und mit aufgenommen.

Ausgangslage

Im Zuge der Verlagerung des Verkaufsprozesses an den Endkunden, setzen Einzelhändler auf Selfcheckout Prozesse, die zumeist in die Kundenbindungs-Apps der einzelnen Marken integriert sind oder auf Kiosk- und Automatensystemen durchgeführt werden. Will der Kunde über diesen Vertriebskanal einkaufen, muss er die App des Einzelhändlers installieren und sich bei einem Kundenbindungsprogramm anmelden.

Zielsetzung

Ziel ist es, einen Prototypen einer App für iOS und Android zu kreieren, mit welchem Kunden bei jedem am Projekt teilnehmenden Shop einkaufen können. Ein Kunde kann einen Shop auswählen und Produkte mittels Scan oder manueller Eingabe dem Warenkorb hinzufügen. Die steuer- und fiskalrechtlichen Aspekte Österreichs müssen bei der Belegerstellung berücksichtigt werden.

Nutzenziele

Das Projekt „NoQuePOS“ wird infolge der Projektabnahme kommerziell genutzt.
Ziele dieser Unternehmung sind vielfältig.
Diese bestehen aus:

Gewinn-/ Kostenorientierung

Hauptziel der Umsetzung des Projektes ist die Gewinnoptimierung des operativen Geschäftsfeldes. Diese soll durch die Erbringung eines zusätzlichen Produktes möglichst mehr Gewinn in das Unternehmen einbringen.

Prestige

Durch gezieltes Projektmarketing soll unteranderem das Differenzierungspotenzial am Markt erhöht werden. Es soll durch einen neuen Prototypen die Bekanntheit des Unternehmen gesteigert werden, das wiederum erhöhten Umsatz mit sich bringt, wenn neue Kundenstämme gewonnen werden.

Entdeckung

Zusätzliches Hauptziel ist es eine vollkommende neue Erfindung/ Innovation am Markt zu bringen und diesen gegeben falls revolutionieren. Dadurch werden weitere Prozesse, besonders im Einzelhandel, digitalisiert und bedürfen keinerlei menschliche Einflüsse mehr.

Projektziele

Das Projekt „NoQuePOS“ richtet sich bei den Projektzielen an das magische Projektdreieck- Im Konkreten wird die Erreichung eines angestrebten Zustands (siehe Tabelle unten) angestrebt, sowie zu einem bestimmten End-Termin (31.03.2021) mit einem bestimmten Ressourcen-Einsatz von 150-170h pro Projektteammitglied.

Zielart	Ziele: Basisplan	Ziele: adaptiert per 12.10.2020
Hauptziele	• Login & Registrieren • Shop Assignment • Warenhorb editieren • Rechnungs-Historie • Startpage • Einstellungen • Mitgliedskarten • Werbung	• Login & Registrieren • Shop Assignment • Warenhorb editieren • Startpage • Einstellungen
Zusatzziele	• Dashboard	• Rechungs-Historie
Nicht-Ziele	• Projektmarketing	• Projektmarketing • Dashboard • Mitgliedskarten • Werbung

Tabelle 1: Projektziele

Ist-Analyse

Die Ist-Analyse des Projektes „NoQuePOS“ ist eine Untersuchung, bei der die jeweiligen untersuchte Objekt in ihre Bestandteile zerlegt werden. Diese Elemente werden dabei auf der Grundlage von Kriterien erfasst und anschließend geordnet, untersucht und ausgewertet. Dadurch werden Beziehungen und Wirkungen zwischen den Elementen betrachtet.

Technologieanalyse

Da das Projekt ist sehr umfangreich gestaltet und geplant ist, wird es in Frontend- und Backendentwicklung geteilt, zu effizienteren Umsetzung.

Hierfür wurde getrennt die Technologien analysiert:

Frontend Analyse

Grundvoraussetzungen für alle Technologien:

- Existieren schon lange und werden voraussichtlich noch lange existieren
- Cross-Platform und unterstützen iOS und Android
- Werden in der Praxis verwendet
- Unterstützung in uns bekannten IDEs (Visual Studio, Visual Studio Code, IntelliJ)

Wichtige Features:

- Übersetzung – Wie leicht kann die App übersetzt und lokalisiert werden, sodass der Benutzer in der App eine Sprache auswählen kann?
- Anpassbares Design während der Runtime – Dass beim Betreten einer Filiale das gesamte Layout der App verändert werden kann
- Hardwarezugriff (Kamera, Lampe, biometrische Daten, Standortdienste) – Für Funktionen besonders im Shop Assignment wichtig
- Lizenz, die kommerzielle Verwendung erlaubt – Dies ist eine kommerzielle Verwendung, deshalb sollten wir eine Lizenz dafür haben

Nice to have:

- Vorkenntnisse – Kennen sich Teammitglieder bereits mit dem Framework oder ähnlich aufgebauten Frameworks aus?
- Plattformspezifisches Design – Passen sich Elemente wie Buttons an das Betriebssystem an, sodass sich der Benutzer gut zurechtfinden kann?

Dart/Flutter

Flutter kann Apps erstellen, die sehr gut aussehen und sehr starke Performancevorteile bieten. Im Gegensatz zu React Native, Ionic und anderen Webtechnologien ist es näher am Betriebssystem, was auch den Zugriff auf Gerätefunktionen wie die Kamera erleichtert. Allerdings haben wir keine Erfahrung damit, weder mit dem Framework noch der Programmiersprache Dart.

Kriterien:

+ Übersetzung: Ja, sogar schön
+ Anpassbares Design während der Runtime (Für Filialdesign): Ja
+ Hardwarezugriff (Kamera, Lampe, biometrische Daten): Ja
+ Lizenz, die kommerzielle Verwendung erlaubt: BSD-3

Vue Native

Vue bezeichnet sich als einfach zu erlernendes Framework, das man sich strukturieren kann, wie man will. Wir müssten uns deshalb sehr genau ausmachen, wie das Projekt strukturiert ist, da man sonst eventuell leicht den Überblick verlieren könnte. Auch bietet Vue Native kaum dokumentierten Support für Gerätefunktionen außer der Kamera, dem Gyrosensor und Push-Notifications.

Kriterien:

+ Übersetzung: Einfach zu übersetzen mit Plugin vue-i18n
? Anpassbares Design während der Runtime (Für Filialdesign)?
- Hardwarezugriff (Kamera, Lampe, biometrische Daten): Biometrische Authentifizierung nicht vorhanden, Zugriff auf Device APIs ist nicht ausgereift
+ Lizenz, die kommerzielle Verwendung erlaubt (MIT)

Ionic/React

Als einer unserer beiden Favoriten bietet diese Technologie die Vorteile der stark ausgereiften Ionic-APIs und die Vorteile von React mit übersichtlichen Komponenten und Databinding. Der Kamerazugriff erfolgt über MediaDevices.getUserMedia(), was leicht unhandlich ist aber funktionieren kann.

Kriterien:

+ Übersetzung: Einfach zu übersetzen mit i18next
+ Anpassbares Design während der Runtime (Für Filialdesign): Mit CSS oder Theme Switcher Package
+ Hardwarezugriff (Kamera, Lampe, biometrische Daten): Ja, Kamera etwas schwer
+ Lizenz, die kommerzielle Verwendung erlaubt: MIT

Ionic/Angular

Ionic mit Angular ist die besser unterstützte Option, Ionic zu verwenden, wobei der Unterschied nicht mehr allzu groß ist wie früher. Im Gegensatz zu React ist das meist verwendete Übersetzungspackage weit jünger und scheint weniger ausgereift. Designanpassungen könnten auch schwerer werden als mit React.

Kriterien:

+ Übersetzung: Ja, mit Package rxweb
~ Anpassbares Design während der Runtime (Für Filialdesign): Ja aber unhandlich
+ Hardwarezugriff (Kamera, Lampe, biometrische Daten): Ja
+ Lizenz, die kommerzielle Verwendung erlaubt: MIT

React Native

Die Rendering Engine von React Native ist schneller als die von Ionic mit React und React Native existiert länger und kommt von den Entwicklern von React (Facebook).

Kriterien:

+ Übersetzung: Ja
+ Anpassbares Design während der Runtime (Für Filialdesign): Mit CSS oder Theme Switcher Package
+ Hardwarezugriff (Kamera, Lampe, biometrische Daten): Ja
+ Lizenz, die kommerzielle Verwendung erlaubt: MIT

Xamarin

Wir haben alle Erfahrung damit aus dem Unterricht, Xamarin tendiert nur dazu, schnell separate Codebases für Android, Windows und iOS zu benötigen, was zu mehr Fehlern führen kann. Auch wollen wir eher neue Technologien verwenden, die wir davor noch nicht so gut kannten.

Kriterien:

+ Übersetzung: Ja, mit RESX Files
+ Anpassbares Design während der Runtime (Für Filialdesign): Ja
~ Hardwarezugriff (Kamera, Lampe, biometrische Daten): Ja, mit 2 Codebases
+ Lizenz, die kommerzielle Verwendung erlaubt: MIT

Ionic/Vue

Ionic mit Vue ist derzeit in einer Betaphase, weshalb wir es als nicht stabil genug erachten.

Persönliche Interessen:

Alex:
Erlernen neuer Technologien: Kein Xamarin
Flutter oder Ionic/React
Lucas:

- Neue Technologien: Flutter oder Ionic, Kein Xamarin
- Ionic wenn mit React, weil bequemer und modularer, auch wenn es weniger unterstützt wird als Ionic/Angular

Dennis:

- Xamarin → kein Interesse, da schon geübt und Interesse an neueren Technologien
- Vue Native → keinen Sinn, ohne guten Hardware Zugriff (siehe geringe Dokumentation)
- React Native → bietet bessere Performance als Ionic / hat mich auch vor dem Projekt immer interessiert mal zu machen
- Ionic/React → Nachteil keine Nativen Komponenten verwenden

Kriterien → Framework	Übersetzung	Anpassbares Design	Hardware-zugriff	Lizenz	Persönliche Interessen	Gesamt
Dart/Flutter	+	+	+	+	+	+
Vue Native	+	?	-	+	-	~
Ionic/React	+	+	+	+	+	+
Ionic/Angular	+	~	+	+	-	~
React Native	+	+	+	+	+	+
Xamarin	+	+	~	+	-	~

Tabelle 2: Auswertungstabelle Frontend

Fazit

Nachdem wir zu sieben Technologien Vor- und Nachteile recherchiert haben, haben wir uns mittels der Auswertung für folgende drei Möglichkeiten entschieden:

- Ionic/React
- React Native
- Flutter

Parallel zu unserer Auswertung hat der Projektpartner ebenfalls eine Auswertung gemacht, die Flutter als Favoriten hervorbrachte. Durch gemeinsame Absprache haben wir uns dann im Frontend für Flutter entschieden.

Backend Analyse

Für die Entwicklung der NoQuePOS Applikation wird ein Backend benötigt, mit dem Daten in einer Datenbank modifiziert werden können. Zur Auswahl stehen zwei verschiedene REST-API Frameworks, nämlich “ASP.NET Core” und das “Spring Boot” Framework.

ASP.NET Core

ASP.NET Core ist ein Framework zur Entwicklung von REST-APIs von Microsoft.

Vorteile

+ Modern (Sprachdesign, Implementierung)
+ MVC Pattern (Model, View, Control)
+ Open-Source
+ Ständige Updates/Weiterentwicklung durch Microsoft
+ Visual Studio IDE (sehr fortschrittlich und modern)
+ C# hat eine sehr große Community, bei Problemen findet man sehr oft Lösungen
+ Visual Studio erlaubt einfaches Deployment auf Microsoft Azure
+ Man kann Datenbankschemen leicht aus einer Datenbank “importieren” (db scaffold)
+ Authentifizierung ist einfach (Authorized Requests werden unterstützt)
+ Nuget Packet Manager
+ Eingebaute Dependency Injection

Nachteile

- Nicht so gute Dokumentation
    - Lücken in der Dokumentation
- Es können nur Core Module verwendet werden
- Abfragen können nicht via Annotations generiert werden und müssen per Hand geschrieben werden (mehr Aufwand)
- Repository Interfaces müssen manuell implementiert werden
- Services müssen extra in Konfiguration zur Verfügung gestellt werden

GraphQL

GraphQL ist eine Open-Source-Datenabfrage- und Manipulationssprache und ein Laufzeitsystem zum Beantworten von Abfragen mit vorhandenen Daten.

Vorteile

+ Weniger Endpunkte
+ Nur die benötigten Daten werden vom Server geladen
+ Validation und Type-Checking der Daten sind standardmäßig implementiert

Nachteile

- Alle Abfragen geben einen Success Code zurück (selbst wenn keine Daten gefunden worden sind)
- Vor der Abfrage muss ein Schema definiert werden (zusätzliche Arbeit)
- Parsen der Requests muss über Zusatzbibliotheken erfolgenü
- Eine Abfrage kostet in der Regel mehr Bytes als eine REST-Abfrage
- Server muss viel Verarbeitung durchführen, um die Anfrage zu parsen und Parameter zu überprüfen
- Caching ist weitaus schwieriger zu implementieren als mit REST

Spring Boot

Spring Boot ist ein Framework zur Entwicklung von Web Applikationen. Man kann auch reine REST-Applikationen mit dem Spring Boot Framework bauen.

Vorteile

+ Konfliktfreies Abhängigkeitsmanagement durch Gradle/Maven
+ Sehr gute Dokumentation
+ Sehr viele Tutorials im Internet (z.B. baeldung.com)
+ Frei wählbares Pattern (MVC, etc.)
+ Skalierbar
+ Sehr leicht konfigurierbar
+ Integrationstests sind sehr leicht zu schreiben
+ Custom Queries nur mit Annotation
+ Repository Interfaces werden über den Methodennamen automatisch generiert
+ Eingebaute Dependency Injection
+ Automatische Service Configuration
+ IntelliJ als IDE (fortschrittlich, modern, regelmäßige Updates)
+ Alle Funktionen sind auf allen Plattformen verfügbar
+ Einrichtung geht schnell und einfach (auch online möglich)

Nachteile

- Teils große Deploymentfiles wegen unbenötigten Dependencies
- Neustart der Applikation nach Änderungen notwendig
    - Hot Reloading schwierig bis nicht implementierbar
- Kompliziertes Deployment

Fazit

Warum Spring Boot und nicht ASP.NET Core?

- Spring Boot hat eine weitaus bessere Dependency Injection als ASP.NET Core
- Learning Curve ist nicht so steil wie bei ASP.NET Core
- Spring Boot erlaubt Custom Queries, mit denen komplexere Abfragen an den Datenbankserver ermöglicht werden
- Repositories werden in Spring Boot automatisch implementiert und Methoden können über den Namen definiert werden, in ASP.NET nicht
- IntelliJ IDE ziemlich gut und modern
- Die Dokumentation von Spring Boot ist umfangreicher als von ASP.NET

Warum Kotlin und nicht Java?

- Kotlin hat Properties (schönerer und besser wartbarer Code)
- String concatenation in Kotlin geht via $ Zeichen (erleichtertes Handling, effizienterer Code)
- Nullsafe Types
- Kotlin hat “data” Klassen, die sich für Model Klassen gut eignen
- Höhere Typensicherheit als in Java
- Intelligenteres und einfacheres Casting als in Java
- In Kotlin kann man beim Methodenaufruf Values per Variablenname zuweisen
- Kotlin erlaubt Default-Werte bei Methoden
- Kotlin braucht in der Regel weit weniger Zeilen an Code als Java (simplified)
- Kotlin basiert auf Java, kann alles was Java kann und mehr

Gradle oder Maven?

- Gradle liefert bessere Performance als Maven
- Gradle benötigt kein XML (Konfigurations File)
- Gradle hat eine sehr gute Unterstützung von IntelliJ IDE
- Deployment zu Microsoft Azure mit Maven einfacher als mit Gradle

PostgreSQL oder MySQL?

- PostgreSQL unterstützt die Verwendung von eigens erstellten Datentypen
- Beide Datenbanksysteme sind Open Source
- MySQL unterstützt Stored Procedures, allerdings nur mit der Standard SQL Syntax. In PostgreSQL kann man Stored Procedures mit verschiedenen anderen Sprachen schreiben, zum Beispiel mit Ruby, Perl, Python, TCL, PLSQL, SQL und JavaScript.

Warum kein GraphQL?

- Die Implementierung von GraphQL in Spring Boot ist zwar möglich, allerdings ist dies um einiges komplizierter, umständlicher und architektonisch unsauberer als eine REST-API. GraphQL hat zwar weniger Endpunkte, dafür muss der User aber in der Regel mehr Daten von seinem Handy aus an den Server schicken, um das gleiche Resultat zu erhalten wie bei einer REST-API.
- Das Data Caching ist weitaus komplexer zu realisieren als mit einer REST-API
- Es werden mehr Bibliotheken als bei einer Spring Boot REST-API benötigt

Entscheidungstabellen

	Spring Boot	ASP. NET
Dependency Injection	+	+
Open Source	+	+
Dokumentation	+	+~
Deployment	+~	+
Microservice Support	+	+
Persönliche Präferenz	+	+~

Tabelle 3: Auswertung Spring Boot vs. ASP.NET

	Java	Kotlin
Properties	-	+
Typensicher	+	+
Nullsichere Typen	-	+
String concatination	+~	+
Default Werte für Methodenaufrufe	-	+
Argumentzuweisung bei Methodenaufruf über den Variablennamen	-	+

Tabelle 4: Auswertung: Java vs. Kotlin

	Gradle	Maven
Performance	+	+~
Lesbarkeit des Konfigurationsfiles	+	~
Integration in IntelliJ IDEA	+	+
Deployment	~	+

Tabelle 5: Auswertung Gradle vs. Maven

	PostgreSQL	MySQL
Open Source	+	+
Custom types	+	~
Stored Procedures	+	+~

Tabelle 6: Auswertung PostgreSQL vs. MySQL

Persönliche Präferenzen

- Einiges mehr an Erfahrung in Spring Boot als Asp.NET Core
- Betreuungslehrer kennt sich sehr gut mit Spring Boot aus
- Lernen von neuer Sprache (Kotlin) und Gradle bietet eine angemessene Herausforderung
- IntelliJ gefällt uns besser als Visual Studio
- GraphQL erweist sich als sehr komplex und aufwendig zu erlernen (Learning Curve ziemlich steil)
- Wir haben keine Erfahrung mit GraphQL, weder beim Backend noch beim Frontend

Technologieentscheidung Backend

- Framework: Spring Boot
- Programmiersprache: Kotlin
- Build-Management-Automatisierungs-Tool: Gradle
- Versionsverwaltung: GIT
- Datenbank: PostgreSQL

Stakeholderanalyse

Da das Projekt “NoQuePOS“ ein soziales System ist, müssen auch dessen Beziehungen zu sozialen Umwelten betrachtet werden. Dabei ist zu beachten, welche Umwelten „relevant“ für das Projekt sind und maßgeblich den Projekterfolg beeinflussen.
Hierfür werden relevante Projektumwelten in projektinterne und projektexterne Umwelten unterschieden.
Projektexterne Umwelten sind der Betreuer und der Projektauftraggeber/Kunde.
Das Projektteam und der Projektmanager werden als projektinterne Umwelten betrachtet, da ihre Beziehungen zum Projekt dessen Erfolg zentral beeinflussen.
Zudem werden die Beziehungen zwischen den Umwelten und dem Projekt bewertet und diese Bewertung ist mittels Symbolen (+/-/~, etc.) dargestellt.

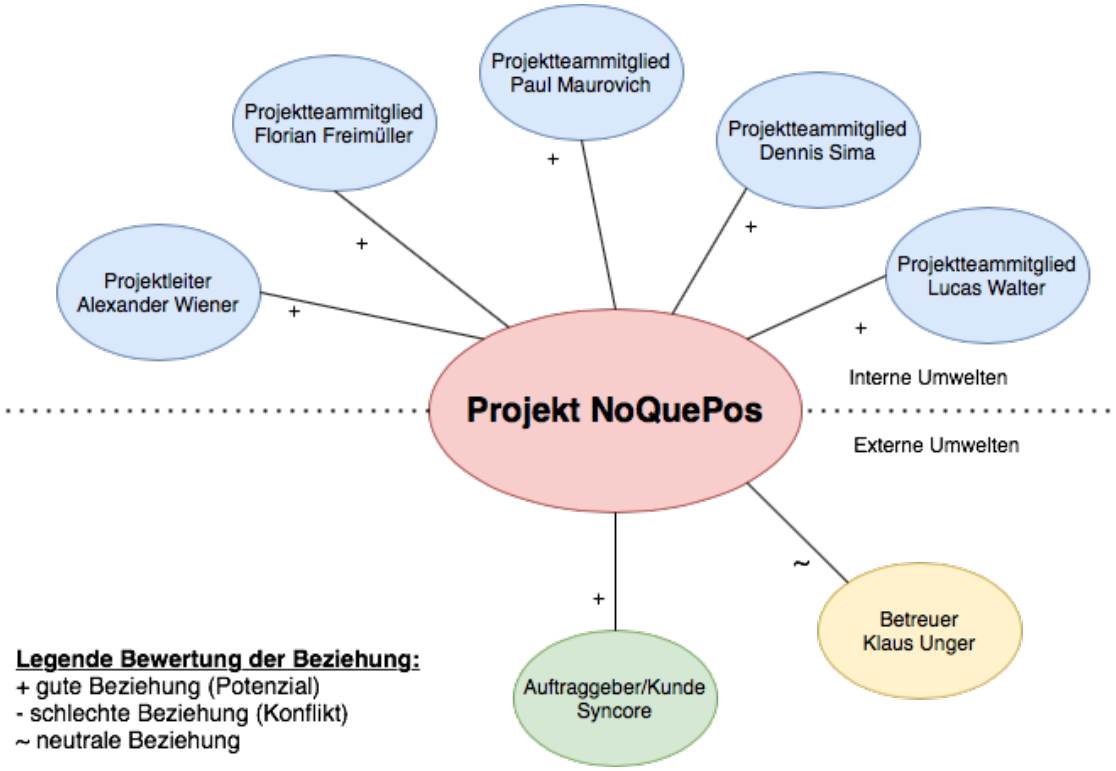


Abbildung 1: Stakeholder

Benutzeranalyse

Das Ziel dieser Benutzeranalyse ist, sich vor Augen führen zu können, welche Endbenutzer die Applikation voraussichtlich verwenden werden. Da die App eine Kassa ersetzen soll, welche jeder verwenden kann, ist es besonders wichtig, dass auch Benutzer mit einbezogen werden, die eventuell nicht so gut mit Mobilgeräten umgehen können. Auch ist das Einkaufsverhalten unterschiedlicher Personen unterschiedlich, so wird mehr oder weniger oder länger oder kürzer eingekauft.

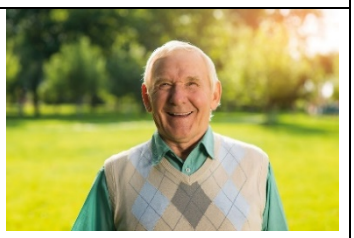
	Die Gelegenheitskäuferin	Der Firmeneinkäufer	Der Wochenendeinkäufer
			
Name	Stefanie Schmidt	Vincent Hawlik	Wilhelm Peter
Alter	18	30	70
Sozialleben	Stefanie ist Schülerin.	Vincent ist Angestellter und hat viel mit Technik zu tun.	Wilhelm ist Pensionist und hat Schwierigkeiten damit, Apps zu bedienen.
Einkaufsverhalten	Stefanie geht in ein Geschäft und nimmt sich vor, eine beliebige „Kleinigkeit“ zu kaufen. Wenn sie einkauft verlässt sie den Laden meist mit wenigen (< 5) Produkten.	Vincent möchte öfters für seine Firma mittlere Mengen an Waren einkaufen und weiß wenn er ins Geschäft geht genau, was er kaufen will.	Wilhelm kauft am Wochenende gerne große Mengen an Produkten ein. Er plant seinen Einkauf davor mit den Angeboten aus Prospekten auf einer Einkaufsliste und arbeitet diese dann ab, sodass er nichts vergisst.

Tabelle 7: Benutzeranalyse

Diese Benutzeranalyse zeigt, dass es wichtig ist:

- Stefanie: Einen schnellen Einkauf bei wenigen Artikeln zu ermöglichen, für den auch eine kurze Pause ausreicht
- Vincent: Es ermöglichen, Einkäufe als Firma zuzulassen, indem der Benutzer eine Adresse auf der Rechnung angeben kann
- Wilhelm: Eine lange Einkaufsliste zu ermöglichen und eine einfache und intuitive Bedienung vorzusehen, sodass auch Pensionisten die App bedienen können.

Kundenanalyse

Der Prototyp bzw. die mobile App wird mehrere Kundenarten und Kundengruppen ansprechen. Wo der weitere Unterschied liegt klärt folgende Analyse:

Business to Business

Zukünftige Kunden und Konsumenten des Produktes werden im sogenannten „B2B-Bereich“ zu finden sein. Da große Handels- und Dienstleistungsunternehmen über unserer App ihre Produkte vertreiben können. Vorwiegend werden das allerdings mittelgroße Unternehmen sein, welche nicht genug geeignetes Humankapital, sowie finanzielles Kapital aufbringen können. Sicherlich werden auch größere Konzerne zu unseren Kunden kommen.

Standortfaktoren des Business-Kunden
Große Ballungszentrum mit viel Internet-Bandbreite und gute Infrastruktur werden gezielt durch Marketing-Kampanien angesprochen. Da vor allem in Städte viele Einkaufstassen, Einkaufszentrum etc. vorzufinden sind.

Business to Customer

Der Endkunde der mobilen App wird mit dieser gegeben falls einkaufen. Dieser muss sowie im B2B-Geschäft durch andere Marketing-Aktionen zum Benutzen der App motiviert werden.

Analyse von B2C:

Art	Beschreibung
Geografische Kriterien	• Länder mit Einkommen von 10$ am Tag pro Person • Einwohner in vorwiegend im gemäßigten Klimazonen • Großstädte und Ballungszentren
Demografische Kriterien	• Alter von 16 - ? • Geschlecht irrelevant • Familienstand irrelevant • Haushaltsgröße irrelevant
Soziografische Kriterien	• Einkommen über 10$ am Tag • Min. Kaufkraft für Existenzminimum • Bildungstand ein Smartphone zu bedienen • Berufstätigkeit irrelevant
Informationsverhalten	• Mundpropaganda • Radio • Fernsehen

	• Social-Media • Werbe-Anzeigen • Werbe-Plakate
Kaufverhalten	• Billige Waren und Güter • Preisbewusstsein • Frequenz der Einkäufe: 2-mal am Tag
Verwendungsverhalten	• Mehrmals täglich → Öffis,… • Unendliches Scrollen durch Werbung

Tabelle 8: Analyse Business 2 Customer

Konkurrenzproduktanalyse

Da die Projektidee nicht komplett neu ist, gibt es bereits einige Organisationen/ Firmen die sich schon mit dem Thema auseinandergesetzt haben.
Diese sind folgend aufgelistet inkl. besonderer Features:

Scandit

- Barcode-Scanning Technologie
- Augmented Reality
- Texterkennung
- ID Scanning
- Unterstützt eine Scan & Go App

Link: https://www.scandit.com/de/ am 15.10.2020

Scan & Go

- White Label App
- Bezahlung per App
- Kundenregistrierung
- JÖ-Card Anbindung
- Google Maps eingebettet
- GPS - & Bluetooth-Freigabe
- Produktempfehlungen
- Digitaler Beleg

Link: https://www.billa.at/sonderkapitel/seiten/billa-apps am 15.10.2020

Snabble

- Verschiedene Shops in einer App
- Bezahlung per App
- Bezahlung an Selfcheckout Kassa
- Kundenkarten Anbindung
- Shopliste + Öffnungszeiten
- GPS-Freigabe
- Altersverfikation mit Personalausweis

Link: https://snabble.io/de am 15.10.2020

MishiPay

- White Label App
- Verschiedene Shops in einer App
- Bezahlung per App
- Warenentsicherung mit der App verknüpft
- Kundenkarten Anbindungen
- Shopliste (Stores in der Nähe)
- Keine Info zu den Stores
- GPS-Freigabe
- Scandit Technologie
- Benutzung auch ohne Account möglich
- Digitaler Beleg
- Web-App

Link: https://mishipay.com/ am 15.10.2020

Penny Scan & Go

- Re-Vision
- QR Code scannen zur Shopidentifikation
- Bezahlung an Self-Checkout Kassa
- Warensicherung durch Stichproben
- Verkauf von Alkohol & Tabak nur mittels Mitarbeiter
- Keine Einstellungen
- Kein digitaler Beleg
- Benutzung ohne Account

Link: https://www.penny.de/penny-go am 15.10.2020

Koala

- Verschiedene Shops in einer App
- Store über Standortliste auswählen
- Bezahlung in der App
- Warensicherung durch Stichproben
- Händische EAN Code Eingabe
- Digitaler Beleg
- Benutzen ohne WLAN & Bluetooth möglich
- Benutzung mit und ohne Account

Link: https://www.koalaapp.de/ am 15.10.2020

Supersmart

- White Label App
- Pro Shop eigene App
- Anbindung zu Validation Station (AI, Gewicht, Bilderkennung)
- AI Recommendations
- Shopidentifikation mit QR Code
- Scandit Technologie
- Kein Account
- Einkaufsverlauf in der App

Link: https://supersmart.me/ am 15.10.2020

Auswertung der Konkurrenzproduktanalyse

Nach dem wir einige Konkurrenzprodukte auf deren Features und Funktionen analysiert und getestet haben, sind wir mithilfe unserer persönlichen Gedanken zu folgendem Entschluss für unseren Prototypen gekommen:

Was wir derzeit nicht machen wollen:

- Augmented Reality einbauen
- Pro Shop eine eigene App
- Warenentsicherung
- Empfehlungen mittels Künstlicher Intelligenz
- Barzahlung an Kassenterminals
- Ausdrucken der Rechnung an Kassenterminals

Was wir machen wollen:

- Nutzung der App mit Account oder mit eingeschränktem Gastmodus möglich
- Eine White Label App die das Design an die Farben des jeweiligen Händlers anpasst
- Shopliste von Shops in der Nähe
- Bezahlung in der App via Stripe-Anbindung
- Digitaler Beleg (Fiskalisierung) als PDF und via E-Mail

Was wir abändern würden:

- Shopidentifikation mittels QR, aber mit der Zusatzmöglichkeit einen 8-stelligen Code einzugeben der ebenfalls den Shop identifiziert
- Einstellungen für die App (App-Präferenzen)
- Eingabe des EANs und Eingabe der Artikelnummer
- Produktempfehlungen anhand der letzten Käufe und des Standortes
- Eine Rechnungshistorie (für alle Nutzer mit einem Account)

SWOT

Mithilfe der SWOT-Analyse haben wir die unter „Internen Faktoren“ unsere Stärken und Schwächen herausgearbeitet und unter „Externe Faktoren“ haben wir Chancen und Risiken abgeschätzt.

Interne Faktoren	Externe Faktoren
Stärken • **Mobile-Crossplatform Erfahrung** • **Microservice Architektur Praxiserfahrung**	Chancen • Folgeaufträge • Beschäftigung für Weiterarbeit am Projekt • Vorzeigeprojekt • Kommerzielle Ziele
Schwächen • **Abstimmung / Einigung aller 5 Personen** • **Zeitmanagement** • **Einarbeitung in neue Technologien nötig** • **KnowHow Lücken**	Risiken • Konkurrenzprodukte/Patente • Ausfall von Projektmitgliedern • Umsetzung während dem Maturajahr

Tabelle 9: SWOT

Lösungsentwurf

Im Rahmen der Zusammenarbeit mit dem Partner wurden nach langem Prozess mehrere Entscheidungen getroffen.

Technologie

Für die Technologien wurden folgende Einschränkungen festgelegt:

- Open Source
- Eine aktuelle Technologie

Daraus hat sich unter Absprache mit dem Partner folgende Lösung ergeben:
Für das Frontend wurde das Projekt, auf das Framework Flutter mit der Programmiersprache Dart, abgestimmt.
Für das Backend wurde das Projekt, auf das Framework Spring Boot mit der Programmiersprache Kotlin und dem Build-Tool Gradle, abgestimmt.

Funktionen

- Nutzung der App mit Account oder mit eingeschränktem Gastmodus möglich
- Eine White Label App, die das Design an die Farben des jeweiligen Händlers anpasst
- Shopliste von Shops in der Nähe
- Digitaler Beleg (Fiskalisierung) als PDF und via E-Mail
- Shopidentifikation mittels QR, aber mit der Zusatzmöglichkeit einen 8-stelligen Code einzugeben der ebenfalls den Shop identifiziert
- Einstellungen für die App (App-Präferenzen)
- Eingabe des EANs und Eingabe der Artikelnummer
- Produktempfehlungen anhand der letzten Käufe und des Standortes
- Eine Rechnungshistorie (für alle Nutzer mit einem Account)

Grobe Umsetzungsplanung des Projekts

Die Planung des Projektes „NoQuePOS“ besteht aus der Organisationsstruktur und den jeweiligen Zeitplan als Basistermine. Dies ist sehr relevant in der Vorprojektphase zu planen um dem Projekt gewisse Rahmenbedingungen zu geben.

Projektorganisation Team

Die Projektorganisation besteht aus dem Kernteam „NoQuePOS“. Dieses ist in Zwei Subteams aufgeteilt, jeweils in Frontend- und Backendentwicklung zur effizienteren Umsetzung des Prototypen. Der Projektleiter Alexander Wiener steht gemeinsam mit dem restlichen Team in ständigen Kontakt zum Projektauftraggeber „SynCore“ mit dem Ansprechpartner Jan Niederreiter. Zudem fungiert Klaus Unger als Diplomprojektbetreuer und als Ansprechperson der Schule „HTL Spengergasse“´, welche das Projekt, nach dem Abschluss bewertet.

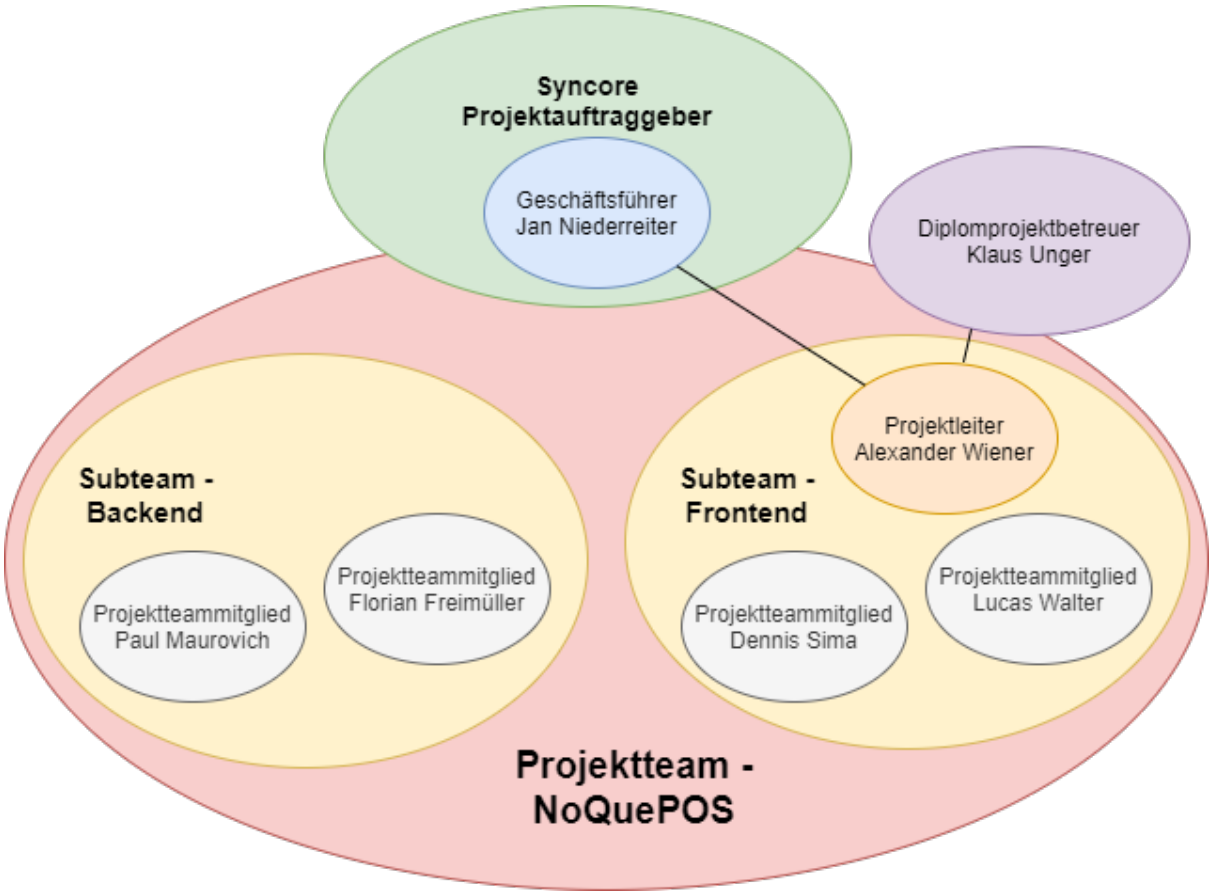


Abbildung 2: Projektteam

Zeitplanung

Das Projekt beginnt mit dem Unterschreiben der Kooperationsvereinbarung. Dadurch wurde der Projektstart eingeleitet. Der Projektpartner/ Auftraggeber stellt bis zum 14.09.2020 die Projektanforderungen zusammen, sodass am 15.9.2020 die Übergabe des Lastenhefts stattfindet. Das Projektteam stellt infolgedessen die Spezifikationen zusammen, die am 23.20.2020 vom Projektpartner abgenommen und genehmigt wird. Danach beginnt das Projektteam mit der ersten Umsetzungsphase, welche am 01.12.2020 einer ersten Analyse und Projektevaluierung dient. Die zweite Phase wird am 04.02.2021 abgeschlossen sein. Danach wird der Prototyp getestet und es fallen eventuelle Nacharbeiten an. Diese werden am 01.03.2021 fertig sein. Die Projektdokumentation wird am 15.03.2021 fällig. Danach fallen die Produktpräsentation und Abnahme an. Projektabnahme ist am 31.03.2021. An diesem Tag endet zudem die Kooperationsvereinbarung.

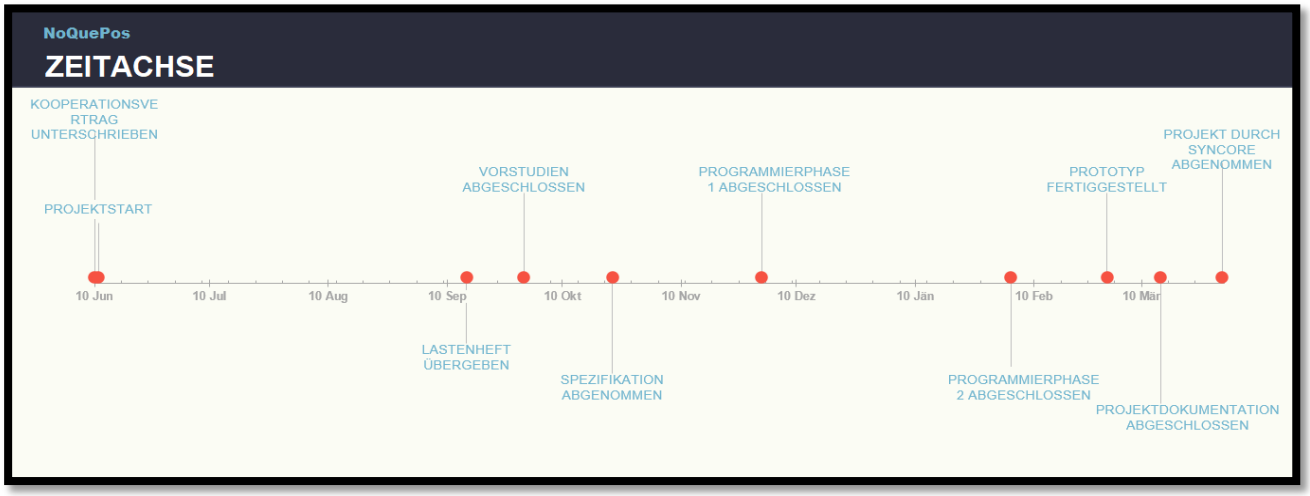


Abbildung 3: Zeitachse des Projekts

Projektrisiko

Eine Hilfestellung zur vollständigen Ermittlung aller Risiken des Projektes ist die Aufgliederung der Risiken in Kategorien.
Diese bestehen aus:

Fachliche Risiken

Das Projektteam befindet sich zum geschriebenen Zeitpunkt noch in ihrer Ausbildung und Abschlussjahr. Daher haben alle Teammitglieder wenig praktische Erfahrung im Umgang mit Software-Projekten. Dies könnte sich zunehmend als großes Risiko entwickeln mit einer erhöhten Eintrittswahrscheinlichkeit.
Risikominimierung wäre die technische Expertise unseres Betreuers, der langjährige Erfahrung mit der technischen Umsetzung etc. mit sich bringt.

Kostenrisiken

Da sich das Projekt weitgehend auf Open Source bewegt, wird nach bestem Willen versucht, sämtliche Kosten zu minimieren. Zudem verlaufen sich planmäßig die Projektkosten in einem niedrigen Ausmaß: Mieten eines Microsoft Azure Servers und Lizenzen für die kommerzielle Nutzung der JetBrains IntelliJ IDEA.
Deshalb spielt der Kostenfaktor in diesem Projekt kein relevantes Risiko.

Terminrisiken

Das Risiko sämtliche Termine der Terminplanung zu überschreiten ist aufgrund mehrerer Ursachen nicht zu unterschätzen. Da das Projekt als End-termin den 31.03.2021 hat, muss dieser unter jeden Umständen erreicht werden. Ursachen hierfür wären nicht zeitgemäßes Erreichen der einzelnen Module/ Arbeitspakete aufgrund Technischen Risiken, Personelle Risiken und Ressourcenrisiken.
Um dieses Risiko zu minimieren wurden bereits einige Gegenmaßnahmen eingeleitet. So wurden am 12.10.2020 die Projektziele neu definiert, um jene Termine nicht zu überschreiten.

Ressourcenrisiken

Das Projekt ist mit enormem Arbeitsaufwand verknüpft, wobei der zielgerichtete Aufwand bei 150-170h pro Mitglied zu rechnen ist. Das Risiko ist sehr hoch, falls man alle Projektziele erreichen möchte, denn diese könnte nicht ausreichend sein.
Um dieses Risiko zu minimieren wurden bereits einige Gegenmaßnahmen eingeleitet. So wurden am 12.10.2020 die Projektziele neu definiert, um diese Ressourcen nicht zu überschreiten.

Qualitätsrisiken

Da es sich im Projektteam um ausschließlich junge und unerfahrene Entwickler handelt, besteht eine erhöhte Gefahr der Unachtsamkeit. Diese könnte z.B. im Projektmanagement, bei der Entwicklung des Prototypen entstehen. Darauffolgend muss man mit einer hohen Eintrittswahrscheinlichkeit rechnen.
Gegenmaßnahmen wären unter anderem „Pair-programming“ bei besonders fehleranfälligen Schnittstellen bei der Umsetzung sowie mehrfaches „Reviewen“ der einzelnen Dokumente beim Projektmanagement.

Personelle Risiken

Aufgrund des großen Personalaufwands von 5 Personen ist das generelle Personelle Risiko eines Ausfalles etc. sehr unwahrscheinlich. Da das Projektteam zudem mäßig groß an Personal aufgestellt ist, sollte dieses Risiko zu vernachlässigen sein. Jedoch muss erwähnt werden, dass dieses Risiko nicht ausgeschlossen ist, besonders bei Krankheiten, wo einzelne Mitglieder für kürzere Zeiten ausfallen könnten.

Risikotabelle

ID	Risikobezeichnung (Ereignis)	Risiko-klasse	Mögliche Risikoursache	EW %	Schadenshöhe (€)	Risiko-wert (€)
R1	Personalausfall ol-	Personal	• Krankenstand • Kündigung	20%	1500€ bei Kündigung 0€ bei Krankenstand	300€
R2	Basistermine nicht erreichen	Termine/ Zeit	• Fehlendes Know-How • Schulstress	10%	6000€	600€
R3	Kostenüberschreitung	Finanzen	• Gebrauch von mehreren Lizenzen	2%	Jeweilige Kosten	Jeweilige Kosten
R4	Nicht die 5. Schulstufe schaffen	Schule	• Viel Schulstress • Burnout • Private Gründe • Ablenkung durch Freizeit	10%	8000€	800€
R5	Hardwareausfall	Technik	• Windows-Viren • Hardware-Defekt	20%	Jeweiliges Gerät/-Komponente ~ 300€	60€
R6	Mängel im Produkt	Mangel	• Fehlende Testung • Unachtsamkeit	35%	15000€	5250€
R7	Uneinigkeiten im Projektteam	Personal	• Diskussionen über Projektentscheidungen	70 %	5000€	3500€

Tabelle 10: Risikotabelle

Risikoinventar (Summe Risikowerte):	10 510€
Projektbudget:	18000€
Risikoindex des Projektes:	58%

Literaturverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

In diesem Abschnitt werden alle Fachbegriffe sowie Abkürzungen, bei denen es zu Unklarheiten kommen könnte, erklärt.

.NET Core	Eine Open Source Plattform für .NET-Apps
AI	Künstliche Intelligenz
Android	Ein mobiles Betriebssystem von Google
Annotation	hier: Metadaten in Quellcode
Device-)APIs	Application Programming Interface, mit dem auf Hardwarefunktionen zugegriffen werden kann.
Augmented Reality	Auf einem Kameravideo werden virtuelle Elemente eingeblendet
Authentifizierung	Feststellung, ob ein Benutzer die Person ist, als die er sich ausgibt. In Applikationen ist dies meist die Anmeldung.
Backend	Die Serverkomponente einer Applikation, welche Daten verarbeitet und speichert.
Caching	Zwischenspeicherung von Daten, sodass diese nicht immer aus der Datenbank kommen müssen und schneller geliefert werden können.
Casting	Die Umwandlung eines Werts von einem Datentypen in den anderen (z.B.: Ein Text wird zu einer Zahl umgewandelt)
Codebase	Der Quellcode einer Applikation
Community	Gruppe von Personen, die sich mit einer Technologie beschäftigt und ggf. auch auf Webseiten helfen kann.
Cross Platform	Eine Applikation, welche auf mehreren Betriebssystemen funktioniert (z.B. Windows, Android, iOS)
CSS	Cascading Style Sheets legen das Aussehen von Websites fest (z.B. Farben, Schriftgröße, etc.)
Custom Queries	Selbstgeschriebene Datenbankabfragen, die granularere Abfragen zulassen als sie mit Datenbankbibliotheken möglich sind
Dashboard	Eine Maske einer Applikation, welche wichtige Daten an einem Ort abbildet und den Benutzer Einstellungen verändern lässt.
Databinding	Eine Technik, mit der Daten „gebunden“ und synchronisiert Werden, um sie anzuzeigen.
Datenbankschema	Eine Darstellung der Objekte in einer Datenbank oder eine Sammlung von Datenbankobjekten
Dependency	Eine Abhängigkeit zwischen Modulen einer Applikation
Dependency Injection	Eine Technologie, welche einem Objekt andere Objekte übergibt, die es braucht.
Deployment	Der Prozess, durch welchen eine Applikation für die Ausführung auf einem System vorbereitet wird.
EAN	europäische Artikelnummerierung, die den Strichcode auf Waren festlegt
Fiskalisierung	Ein Manipulationsschutz für Kassen.

	Alle Transaktionen werden lückenlos erfasst und archiviert.
Framework	Ein Softwarepaket, auf welchem Programme basieren können, das grundlegende Funktionen zur Verfügung stellt
Frontend	Der Teil einer Applikation (hier: die App), den ein Benutzer sieht
Git	Ein Versionsverwaltungssystem
Gyrosensor	Ein Sensor, meist in Smartphones, der Bewegungen des Geräts erkennen kann
Hardware	Ein physisches Gerät
Hot Reload	Eine Technologie, welche während der Programmierung Änderungen direkt in das Programm lädt, ohne dass es neu gestartet werden muss.
HTML	Hyper Text Markup Language, eine Sprache, welche verwendet wird, um das Grundgerüst von Webapplikationen zu bilden.
IDE	Integrated Development Environment, ein Programm, welches Programmierer bei der Softwareentwicklung unterstützt indem es Hilfsmittel wie Debugger anbietet.
IntelliJ IDEA	Ein IDE für Java
Interface	Bietet eine Abstraktionsebene, die die Eingangs- und Ausgangsdaten einer Klasse abbildet
iOS	Ein Mobilbetriebssystem von Apple, welches auf iPhones im Einsatz ist.
Java	Eine objektorientierte Programmiersprache
JavaScript (JS)	Eine Programmiersprache
Kundenbindungsprogramm	Ein Programm, welches Kunden zum Kauf von Produkten anregen will. Ein Beispiel dafür ist jö.
Lokalisierung / Internationalisierung / i18n	Der Prozess, ein Programm für Benutzer mit anderen Sprachen anzupassen, von Übersetzung bis zu Datumsformatierung oder rechtsbündige Textausrichtung
Microservice	Eine Softwarearchitektur, welche eine Applikation auf mehrere „Services“ aufteilt
Mitgliedskarten	Karte, welche Kunden das Gefühl gibt, ein Mitglied des Geschäfts zu sein und damit zur Kundenbindung beiträgt
MVC	Mode View Controller, ein Pattern, welches eine Applikation in Model (Die Datenstruktur), View (Die Anzeigelogik) und Controller (Die Programmlogik) aufteilt.
Null Safety	Einfache Überprüfung, ob eine Variable einen Wert aufweist oder nicht
Open Source	Software mit veröffentlichtem Quellcode, welche meist von freiwilligen Mitwirkenden weiterentwickelt wird.
Package Manager	Ein Programm, welches Softwarebibliotheken verwalten kann
Parser	Programmlogik, die eingehende Daten analysieren, überprüfen und zur Verarbeitung vorbereiten kann
Pattern	Hier: Eine Softwarearchitekturform, welche Lösungen für häufige Probleme bereitstellt und Programme vereinheitlicht.
Performance	Die Effizienz, mit welcher ein Programm ausgeführt werden

	kann
Push-Notifications	Eine Kurznachricht, welche auch während ein Programm nicht ausgeführt wird am Bildschirm des Benutzers aufscheinen kann
Rendering Engine	Komponente eines Programms, die Quellcode wie HTML dem Benutzer anzeigt
Repository	Ein zentraler Ort, an welchem Daten verwaltet und gespeichert werden
Request	Anfrage; Eine Nachricht an einen Server
Response	Antwort; Eine Antwort von einem Server, meist folgend auf eine Request
REST	Representational State Transfer; Eine Form der Softwarearchitektur, welche die Kommunikation zwischen Computersystemen vereinheitlicht.
Runtime	Die Zeit und Prozesse die laufen, während ein Programm ausgeführt wird
Selfcheckout	Technologie, die einem Benutzer erlaubt, seine eigenen Zahlungen abzuwickeln, ohne mit einem traditionellen Kassierer in Kontakt zu treten
Server	Ein Rechner, der anderen Computern Daten zur Verfügung stellt. Es gibt Web Server, Mail Server, File Server u.A.
Service	Programmkomponente, die Aufgaben abarbeite oder auf Requests antwortet.
Shop Assignment	Der Prozess, durch welchen ein Benutzer einem Shop zugewiesen wird, sodass er dann in diesem einkaufen kann
Software	Anweisungen, welche der Computer (die Hardware) ausführt
Standortdienste	Sammlung von APIs, welche Apps den Standort des Geräts zur Verfügung stellt
Startpage	Startseite einer App
Strict Typing	Funktion einer Programmiersprache, dass für jede Variable genau festgelegt werden muss, als welcher Datentyp sie gespeichert ist
String concatenation	Aneinanderknüpfen von Texten
Tutorial	Anleitung
Update	Aktualisierung von Software
Validation	Validierung; die Überprüfung von eingehenden Daten
Versionsverwaltung	Ein System, welches Versionen von Quellcode verwaltet und die Entwicklung erleichtert und nachweislicher macht
Visual Studio	Ein IDE von Microsoft, das großteils für die Entwicklung von C#-Programmen konzipiert ist

Visual Studio Code	Ein IDE von Microsoft, das mittels Plugins flexibel an viele Programmiersprachen angepasst werden kann
Warenentsicherung	Die Entschärfung von Diebstahlsschutzsystemen
White Label App	Eine App, welche mehrere Designs annehmen kann, abhängig davon, für welche Firma sie in Verwendung ist
Windows	Ein Desktop- und Mobilbetriebssystem von Microsoft
XML	Extensible Markup Language; ein Datenformat für die Kommunikation zwischen Computersystemen