SPENGERGASSE

Diploma Thesis

NoQuePOS

Florian Freimüller, Paul Maurovich, Dennis Sima, Lucas Walter, Alexander Wiener

Version 1.0 - 2021-04-07

Inhaltsverzeichnis

Colophon . . . 1
Eidesstattliche Erklärung . . . 2
Kurzfassung . . . 3
Abstract . . . 4
Themenstellung: Ein Vergleich von JVM Sprachen im Umgang mit modernen Programmierschnittstellen . . . 5
Abstract . . . 6
Bewertungskriterien . . . 7
- Lesbarkeit des Codes . . . 7
- Dokumentation . . . 7
- Lines of Codes . . . 7
- Unterstüzte Paradigmen . . . 8
- Beispielstabelle . . . 8

Auswahl der JVM Sprachen . . . 9
- Java . . . 9
- Kotlin . . . 9
- Groovy . . . 10
- Scala . . . 10
- Clojure . . . 10

Auswahl der Schnittstellen . . . 12
- Stripe API [über Bibliotheken und Server-side] . . . 12
- REST APIs [Clients] . . . 13
- Streaming API . . . 13
- E-Mail APIs . . . 13

Stripe API . . . 14
- Java . . . 14
- Kotlin . . . 15
- Groovy . . . 17
- Scala . . . 18
- Clojure . . . 19
- Beurteilungstabelle und Fazit . . . 21

Rest APIs . . . 22

Java . . . 22
Kotlin . . . 24
Groovy . . . 25
Scala . . . 27
Clojure . . . 28
Beurteilungstabelle und Fazit . . . 30
Stream API . . . 31
Java . . . 31
Kotlin . . . 32
Groovy . . . 33
Scala . . . 34
Clojure . . . 35
Beurteilungstabelle und Fazit . . . 35
Java Mail API . . . 37
Java . . . 37
Kotlin . . . 39
Groovy . . . 40
Scala . . . 42
Clojure . . . 43
Beurteilungstabelle und Fazit . . . 44
Fazit . . . 45
References . . . 46
Vergleich des Java EE Standard-Frameworks im Vergleich zu Spring zur Entwicklung von Business Applications . . . 48
Abstract . . . 49
Einleitung . . . 50
Java EE Framework . . . 51
Spring Framework . . . 54
Erste Schritte . . . 57
Java EE Framework . . . 57
Spring Framework . . . 58
Bewertungskriterien . . . 62
Regelmäßige Updates und Wartbarkeit . . . 62
Langfristiger Support . . . 63

Kosten und Programmiereleganz . . . 64
Dokumentation . . . 66
Funktionsumfang . . . 68
Querschnittsfunktion . . . 69
Developer Codebase und Community Support . . . 70
Entscheidungsresümee . . . 76
Verwendung von Spring Boot im Diplomprojekt . . . 78
Glossar . . . 79
Quellen . . . 81
Recherche von best practices von UX Design Guidelines für mobile Apps . . . 85
Abstract . . . 86
Warum ist UI und UX wichtig? . . . 87
Was ist UI & UX? . . . 87
Usability Vs. User Experience . . . 88
UI & UX im Laufe der Geschichte: . . . 89
Wie entsteht User Experience? . . . 91
Den Nutzer verstehen . . . 91
User Research . . . 92
Empathize . . . 92
Create . . . 92
Test . . . 93
Develop . . . 94
UI-Patters für mobile Apps . . . 95
Was ist das? . . . 95
Vorteile von UI-Patterns . . . 96
Beliebteste Patterns in der Smartphone-Welt . . . 97
Dark UI-Design Patterns . . . 98
Wie sollte man UI-Design Patterns anwenden? . . . 100
Beliebteste UI-Patterns Library . . . 100
Weiteres Design . . . 103
Farbwahl . . . 103
Typografie . . . 107
Accessibility und Internationalisierung . . . 110
Warum sollte man auf Accessibility Bezug nehmen? . . . 110

Auf was achten? . . . 110
Internationalisierung . . . 113
Glossar . . . 115
Quellen. . . . 116
Untersuchung und die Gegenüberstellung von Payment Providern und deren APIs. . . . 118
Hintergrund . . . 119
Arten von Zahlungen . . . 120
E-Wallets . . . 121
PCI DSS-Konformität. . . . 122
PSD2 und Strong Customer Authentication . . . 124
Vergleich von Zahlungsanbietern. . . . 127
Vergleichskriterien . . . 127
Anbieter . . . 128
Stripe . . . 130
API: Verfügbarkeit und Funktionen . . . 130
API: Dokumentation . . . 131
Angebotene Zahlungsarten . . . 133
Konformität mit Standards und Zertifizierungen . . . 134
Kosten und Transaktionsgebühren . . . 134
Sonstige Funktionen . . . 134
Fazit . . . 135
Rapyd . . . 137
API: Verfügbarkeit und Funktionen . . . 137
API: Dokumentation . . . 138
Angebotene Zahlungsarten . . . 138
Konformität mit Standards und Zertifizierungen . . . 138
Kosten und Transaktionsgebühren . . . 138
Sonstige Funktionen . . . 139
Fazit . . . 139
Adyen . . . 141
API: Verfügbarkeit und Funktionen . . . 141
API: Dokumentation . . . 141
Angebotene Zahlungsarten . . . 143

Konformität mit Standards und Zertifizierungen 143
Kosten und Transaktionsgebühren 143
Sonstige Funktionen 144
Fazit 144
Braintree 146
API: Verfügbarkeit und Funktionen 146
API: Dokumentation 146
Angebotene Zahlungsarten 147
Konformität mit Standards und Zertifizierungen 147
Kosten und Transaktionsgebühren 147
Sonstige Funktionen 147
Fazit 148
Fazit 149
Glossar 151
Quellen 152
Untersuchung und Vergleich von Frontend Frameworks auf deren Eignung für mobile Apps 158
Bedürfnisse 159
Die Arten der Apps 160
Native Apps 160
Web Apps 160
Cross-Plattform (Hybride) Apps 161
Welche Sprachen verwendet werden 163
Swift 163
Java 163
C++ 163
HTML 5 163
TypeScript 163
Frontend Kompaktlösungen 164
Appcelerator 164
RhoMobile 164
MoSync 165
Sencha Ext JS 165
Frontend Frameworks 167

Weitere Kriterien: 167
Flutter 168
Ionic/Angular. 174
Ionic/React 175
React Native. 177
Xamarin 179
Auswertung / Vergleich 182
Schlussfolgerung der Arbeit 183
Appendix A: Kooperationsvertrag 184
Appendix B: User Stories 187
Appendix C: Technical and organizational measurements (TOMs) 226
Appendix D: Procedural directory 235
Appendix E: Privacy Policy 245
Appendix F: Project Health Monitor. 250
Appendix G: Vorstudien 252
Appendix H: Spezifikationen 286
Appendix I: Statusberichte 382
Appendix J: Change Request. 401

Colophon

Spengergasse Press, Vienna

© 2021 by Florian FREIMÜLLER, Paul MAUROVICH, Dennis SIMA, Lucas WALTER, Alexander WIENER

Schuljahr 2020/21

Datum:	**übernommen von:**

Table 1. Abgabevermerk

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Ort, Datum	**VerfasserInnen**
Wien, am 07.04.2021	Florian FREIMÜLLER
Wien, am 07.04.2021	Paul MAUROVICH
Wien, am 07.04.2021	Dennis SIMA
Wien, am 07.04.2021	Lucas WALTER
Wien, am 07.04.2021	Alexander WIENER

Kurzfassung

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. Dies ist für viele Kunden umständlich, da für die meisten Händler eine eigene App benötigt wird.

Ziel des Projekts NoQuePos ist es, für dieses Problem eine Lösung zu bieten. Es handelt sich um den Prototypen einer App für Android und iOS, der das Einkaufen und mobile Kassieren per App in jedem teilnehmenden Geschäft des Projekts ermöglicht. Wenn ein Kunde in einem Geschäft einkaufen will, scannt er mit der App einen im Geschäft angebrachten QR-Code, um sich anzumelden. Folglich kann er Barcodes von Produkten in diesem Geschäft scannen und seinen Einkauf direkt in der App bezahlen.

Durch NoQuePos werden herkömmliche Kassen abgelöst, die zu Schlangen bei Ausgängen und einem enormen Platzverlust in Geschäften führen. Kunden, die mit NoQuePos einkaufen, können Produkte aus dem Regal nehmen, Barcodes davon scannen und ihren Einkauf in der App bezahlen, ohne Zeit an der Kassa zu verlieren.

Abstract

As part of moving the sales process toward the customer, retail stores use selfcheckout-processes, usually integrated into customer loyalty programs of connected brands or kiosks. If a customer wishes to purchase goods via this method, he needs to install the app of the specific retailer and sign up for a loyalty program. As this requires a separate app for each retailer it adds complexity for the customer.

The goal of NoQuePos is to provide a solution to this problem. NoQuePos is a prototype of a mobile app for Android and iOS enabling the customer to shop at any participating store. If a customer wishes to shop in a store he scans a QR Code displayed in the store in order to sign in. Now the customer is able to scan barcodes of products in this store and pay for them from directly within the app.

NoQuePos aims to replace conventional cash registers which cause lines at exits and result in an enormous loss of available space for merchandise. Customers shopping with NoQuePos can take products from store shelves, scan their barcodes and pay for their products within the app, without spending time waiting at a cash register.

Themenstellung: Ein Vergleich von JVM Sprachen im Umgang mit modernen Programmierschnittstellen

Florian Freimüller <fre18149@spengergasse.at>

Abstract

In diesem Paper wird mithilfe von eigens ausgewählten Bewertungskriterien bewertet, welche JVM Sprache wie gut geeignet ist, um verschiedene Programmierschnittstellen anzusteuern.

Die Erwartungshaltung ist, dass die Ergebnisse in allen Sprachen ziemlich gleich sind, da die meisten JVM-Sprachen in der Auswahl syntaktisch ähnlich aufgebaut sind.

Um die Sprachen gut vergleichen zu können, wird für jede Programmierschnittstelle in jeder berücksichtigten JVM Sprache ein Code Snippet geschrieben. Die Aufgabe, die mit dem Snippet erfüllt werden soll, wird im Vorhinein definiert und dieses Snippet wird dann größtenteils zum Vergleich zwischen den Sprachen herangezogen.

Bewertungskriterien

Um die Schnittstellen so gut wie möglich bewerten zu können, wird eine Beurteilungstabelle erstellt. Pro Kriterium können maximal 5 Punkte und minimal 0 Punkte vergeben werden. Diese Tabelle setzt sich aus folgenden Kriterien zusammen:

Lesbarkeit des Codes

Ein wichtiger Aspekt bei der Beurteilung ist, wie lesbar der Code ist, wenn die Schnittstelle angesteuert wird. Hierbei wird vor ein Augenmerk darauf gelegt, ob der Code durch das Ansprechen der Schnittstelle unlesbar wird oder nicht.

Dokumentation

Bei der Dokumentation wird beurteilt, ob es eine Schnittstellendokumentation für die jeweilige Sprache gibt. Sollte es eine geben, wird bewertet, wie gut und übersichtlich die Dokumentation gestaltet ist.

Lines of Codes

Je weniger Codezeilen benötigt werden, um ein Beispiel in der jeweiligen Sprache zu programmieren desto mehr Punkte werden hier vergeben. Als Grundlage wird der Code von der Sprache genommen, welche am wenigsten Zeilen für das jeweilige Beispiel benötigt.

Eine Zeile ist:

- Eine Annotation
- Ein Statement (ein Statement über mehrere Zeilen = eine Codezeile)
- Eine Deklaration (einer Klasse, eines Interfaces, einer Variable etc.)

Nicht zu Codezeilen zählt folgendes:

- Zeilen, die **nur** eine Klammer schließen/öffnen
- Leere Zeilen

Unterstüzte Paradigmen

Bei dem Kriterium "Unterstützte Paradigmen" wird darauf geachtet, dass benötigte Paradigmen unterstützt werden (zum Beispiel funktionale Programmierung für reactive Programming). Nur wenn das Paradigma nicht unterstützt wird, gibt es einen Punkteabzug.

Beispielstabelle

Sprache	**Java**	**Kotlin**	**Groovy**	**Scala**	**Clojure**
Lesbarkeit	4	5	3	4	5
Dokumentation	5	5	5	4	2
Lines of Code	2	4	4	3	5
Unterstützte Paradigmen	5	5	5	5	5
Ergebnis	16	19	17	16	17

Figure 1. Beispielsbeurteilungstabelle

Auswahl der JVM Sprachen

Um möglichst viele Vergleichswerte zu haben, werden die Schnittstellen in fünf verschiedenen JVM Sprachen verglichen.

Java

Java ist eine objektorientierte Programmiersprache und wurde im Jahr 1995 von James Gosling veröffentlicht und wird bis heute in sehr vielen Bereichen verwendet. Da Java eine general purpose language ist und Java dank der JVM (Java virtual machine) plattformunabhängig ist, kann Java für sehr viele Anwendungsimplementierungen eingesetzt werden, angefangen von simplen Konsolenprogrammen bis hin zu Anwendungen auf Bordcomputern von Automobilen. [1]

Tiobe Index

Laut dem Tiobe Index ist Java die zweitpopulärste Sprache (nach C).[2]

Kotlin

Die Programmiersprache Kotlin wurde von der Firma JetBrains entwickelt und im Jahre 2011 veröffentlicht. Wichtig bei der Erstellung von Kotlin war sowohl, dass Kotlin problemlos mit Java gemeinsam verwendet werden kann als auch, dass der in Kotlin geschrieben Code eleganter und effizienter ist als der equivalente Java Code. Hauptsächlich wird Kotlin für Android Applikationen verwendet, allerdings ist es ebenso möglich, die Sprache für Web-Applikationen oder auch native Applikationen zu verwenden, da Kotlin eine general purpose language ist. Außerdem werden in Kotlin sowohl objektorientierte als auch funktionale Programmierung unterstüzt.[3][4]

Tiobe Index

Kotlin belegt auf dem Tiobe Index den 39ten Platz.

Groovy

Apache Groovy ist eine OpenSource general purpose language, in der man sowohl Skripte schreiben, als auch, wie in Java, objektorientierte Applikationen erstellen kann. Außerdem wird auch funktionale Programmierung in Groovy unterstützt. Groovy kann, wie Java und Kotlin auch, in etlichen Anwendungsfällen verwendet werden und man kann alle Java Bibliotheken auch in Groovy verwenden. [5][6]

Tiobe Index

Groovy ist auf dem 17ten Platz auf dem Tiobe Index und hat damit seit dem letzten Jahr 31 Plätze gut gemacht.

Scala

Scala ist eine general purpose langauge mit voller Java Interoperabilität und wurde von Martin Odersky entwickelt. Die erste Version von Scala wurde am 20.01.2004 veröffentlicht und die Sprache wird immer noch weiterentwickelt. Wie bei den vorherigen Sprachen auch kann Scala für sehr viele Anwendungsfälle verwendet werden. [7] [8]

Tiobe Index

Auf dem Tiobe Index belegt Scala den 26ten Platz.

Clojure

Clojure ist eine funktionale JVM Sprache, die im Jahre 2007 veröffentlicht wurde. Mit Clojure kann man sowohl Anwendungen schreiben als auch Skripte. Da die Sprache, wie alle anderen berücksichtigten JVM Sprachen, eine general purpose language ist, kann man mit Clojure Webapplikationen, mobile Applikationen und

native Applikationen erstellen. [9]

Tiobe Index

Clojure ist die in dieser Auswahl am wenigsten populäre Sprache und belegt nur den 60ten Platz im Tiobe Index.

[1] FreeJavaGuide: History of Java programming language, https://www.freejavaguide.com/history.html abgerufen am 25.03.2021

[2] Toibe: TIOBE Index for April 2021, https://www.tiobe.com/tiobe-index/ abgerufen am 05.04.2021

[3] Deshmane, Rohini: Introduction to Kotlin, https://medium.com/@rohinideshmane.21/introduction-to-kotlin-5f39b31610e0 abgerufen am 25.03.2021

[4] Kotlinlang: Calling Java from Kotlin, https://kotlinlang.org/docs/java-interop.html abgerufen am 25.03.2021

[5] Scand: Groovy vs Java: Detailed Comparison and Tips on the Language Choice, https://scand.com/company/blog/groovy-vs-java/ abgerufen am 05.04.2021

[6] Groovy-lang: A multi-faceted language for the Java platform, https://groovy-lang.org/ abgerufen am 05.04.2021

[7] Hicks, Matt: Scala vs. Java: Why Should I Learn Scala?, https://www.toptal.com/scala/why-should-i-learn-scala abgerufen am 05.04.2021

[8] javaTpoint: History of Scala, https://www.javatpoint.com/history-of-scala abgerufen am 05.04.2021

[9] Hickey, Rich: A history of Clojure, https://download.clojure.org/papers/clojure-hopl-iv-final.pdf abgerufen am 05.04.2021

Auswahl der Schnittstellen

Bei den behandelten Schnittstellen wurde darauf geachtet, dass diese häufig Anwendung finden und es daher auch einen Grund für die Entwickler dieser Schnittstellen gibt, diese Schnittstellen so kompatibel wie möglich zu gestalten.

Stripe API [über Bibliotheken und Server-side]

Stripe ist ein Zahlungsanbieter, der im Diplomprojekt verwendet wird.

Im nachfolgenden Diagramm ist der Marktanteil der größten Zahlungsanbieter zu sehen, in dem Stripe den zweiten Platz belegt:

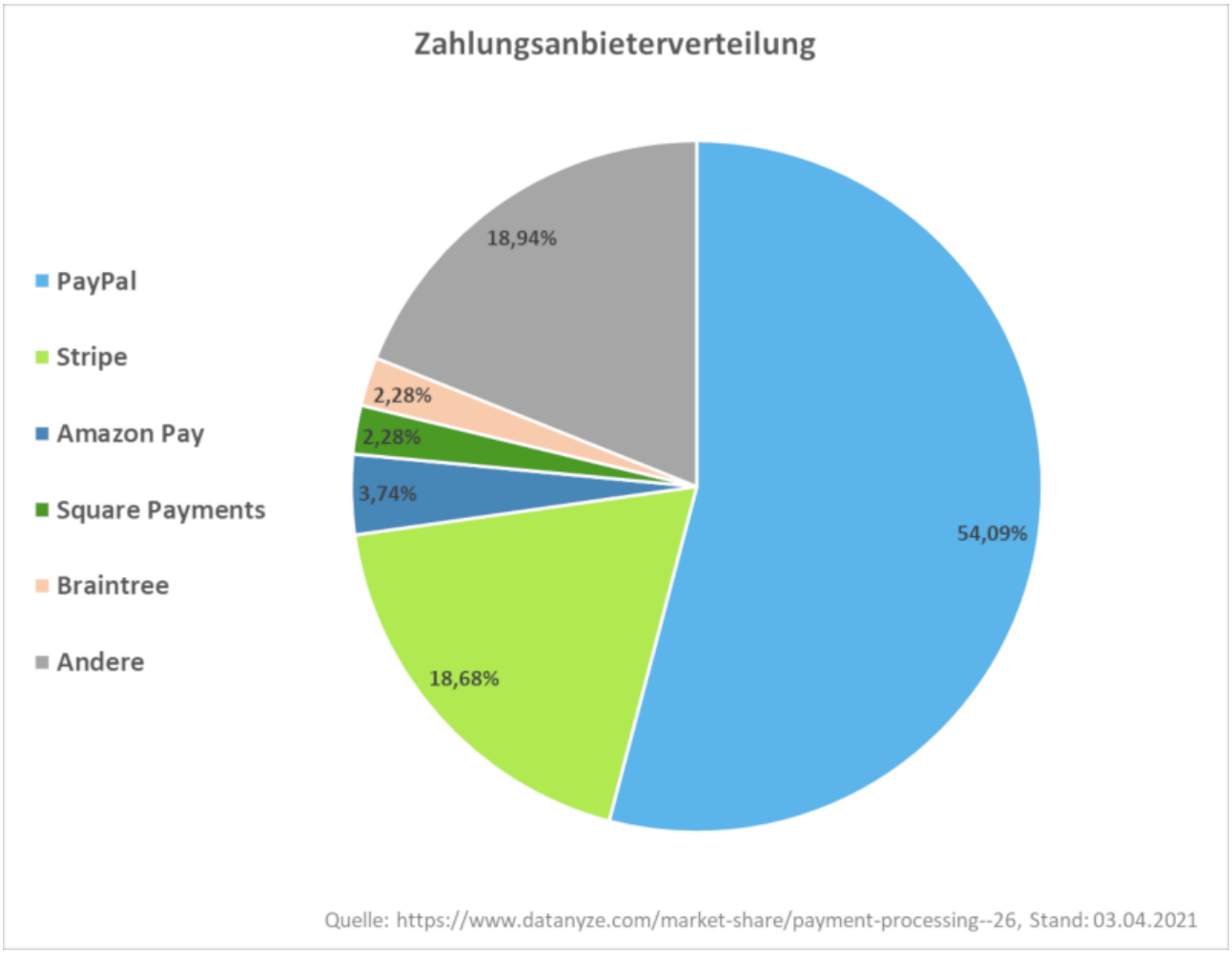


Abbildung 1. Zahlungsanbieterverteilung

REST APIs [Clients]

Da heutzutage sehr viele Services als REST-API zur Verfügung gestellt werden ist es oftmals notwendig, REST-APIs mithilfe von Clients anzusprechen. Dies kann sowohl in Mobilapplikationen als auch in serverseitigen Anwendungen der Fall sein.

Streaming API

In Java gibt es die Streaming-API, in diesem Kapitel wird verglichen, welche Alternativen oder nativen Sprachfeatures es in den anderen Sprachen gibt.

E-Mail APIs

E-Mail APIs werden vor allem in Backend Applikationen benötigt, um Benutzer*innen Informationen per E-Mail zu senden.

Stripe API

In allen Sprachen wird

- eine Zahlung durchgeführt,
- die ID der Zahlung gespeichert und
- der Status der Zahlung mithilfe der ID abgefragt und auf die Konsole ausgegeben.

Die verwendete Bibliothek ist in allen Sprachen "stripe-java".

Java

Code Snippet

Zuerst wird eine Klasse erstellt, mit der eine Zahlung getätigt werden kann und die auch den Status per Methode zurückgibt.

```
/* File: Payment.java */
public class Payment {
    public String makePayment(Long amount, String stripeToken,
RequestOptions options) throws StripeException {
        ChargeCreateParams params = ChargeCreateParams.builder()
                .setAmount(amount)
                .setCurrency("EUR")
                .setDescription("testpayment")
                .setSource(stripeToken)
                .build();

        Charge charge = Charge.create(params, options);
        return charge.getId();
    }

    public String getStatus(String chargeId, RequestOptions options)
throws StripeException {
        return Charge.retrieve(chargeId, options).getStatus();
    }
}
//Lines: 7
```

Anschließend werden die RequestOptions festgelegt und die Funktionen der

Payment Klasse aufgerufen.

```
/* File: Main.java */

    public static void main(String[] args) {
        try {
            RequestOptions options = RequestOptions.builder()
                    .setApiKey(STRIPE_API_KEY)
                    .build();

            Payment payment = new Payment();
            String id = payment.makePayment(1000L, PAYMENT_TOKEN,
options);

            System.out.println(payment.getStatus(id, options));

        } catch(StripeException stripeException) {
            stripeException.printStackTrace();
        }
    }

// Lines: 7
```

Bewertung

- Lines of Code: 14 Zeilen. → 2
- Lesbarkeit: Der Code ist leicht verständlich, wird durch das in Java notwendige Exception-handling allerdings etwas unübersichtlich. → 4/5
- Dokumentation: In der Dokumentation [10] werden alle Endpunkte dokumentiert und es gibt auch Beispiele für verschiedene Sprachen, darunter auch Java. → 5/5
- Unterstützte Paradigmen: Die Bibliothek unterstützt objektorientierte Programmierung, allerdings wird keine funktionale Programmierung berücksichtigt, diese wäre in diesem Fall sinnvoll, da man dadurch zum Beispiel mithilfe eines Observers auf Änderungen des Status achten könnte. → 3/5

Kotlin

Code Snippet

```
/* File: Payment.kt */

class Payment {

    fun makePayment(amount: Long, stripeToken: String, options:
RequestOptions): String {
        val params = ChargeCreateParams.builder()
            .setAmount(amount)
            .setCurrency("EUR")
            .setDescription("testpayment")
            .setSource(stripeToken)
            .build()
        val charge = Charge.create(params, options)
        return charge.id
    }

    fun getStatus(chargeId: String, options: RequestOptions): String {
        return Charge.retrieve(chargeId, options).status
    }
}

//Lines: 7
```

```
/* File: main.kt */

fun main(args: Array<String>) {
    val options = RequestOptions.builder()
        .setApiKey(STRIPE_API_KEY)
        .build()
    val payment = Payment()
    val id = payment.makePayment(1000L, PAYMENT_TOKEN, options)
    println(payment.getStatus(id, options))
}

//Lines: 5
```

Bewertung

- Lines of Code: 12 Zeilen. → 4/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: In der Dokumentation werden alle Endpunkte dokumentiert und es gibt auch Beispiele für verschiedene Sprachen, darunter zwar Java aber leider nicht Kotlin. Da der Code in Kotlin allerdings fast derselbe ist wie der in Java geschriebene Code gibt es hier keinen Punkteabzug.→ 5/5
- Unterstützte Paradigmen: Die Bibliothek unterstützt objektorientierte

Programmierung, allerdings wird keine funktionale Programmierung berücksichtigt, diese wäre aber in diesem Fall sinnvoll, da man dadurch zum Beispiel mithilfe eines Observers auf Änderungen des Status achten könnte. → 3/5

Groovy

Code Snippet

```
/* File: Payment.groovy */

class Payment {
    String makePayment(Long amount, String stripeToken, RequestOptions
options) {
        ChargeCreateParams params = ChargeCreateParams.builder()
                .setAmount(amount)
                .setCurrency("EUR")
                .setDescription("testpayment")
                .setSource(stripeToken)
                .build()
        Charge charge = Charge.create(params, options)
        charge.id
    }

    def getStatus(String chargeId, RequestOptions options) {
        Charge.retrieve(chargeId, options).status
    }
}

//Lines: 7
```

```
/* File: Main.groovy */

static main(args) {
    def options = RequestOptions.builder()
            .setApiKey(STRIPE_API_KEY)
            .build()
    Payment payment = new Payment()
    String id = payment.makePayment(1000L, PAYMENT_TOKEN, options)
    println(payment.getStatus(id, options))
}

//Lines: 5
```

Bewertung

- Lines of Code: 12 Zeilen. → 4/5

- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: In der Dokumentation werden alle Endpunkte dokumentiert und es gibt auch Beispiele für verschiedene Sprachen, darunter zwar Java aber leider nicht Groovy. Da der Code in Groovy allerdings fast derselbe ist wie der in Java geschriebene Code gibt es hier keinen Punkteabzug. → 5/5
- Unterstützte Paradigmen: Die Bibliothek unterstützt objektorientierte Programmierung, allerdings wird keine funktionale Programmierung berücksichtigt, diese wäre aber in diesem Fall sinnvoll, da man dadurch zum Beispiel mithilfe eines Observers auf Änderungen des Status achten könnte. → 3/5

Scala

Code Snippet

```
/* File: Payment.scala */

class Payment {
  def makePayment(amount: Long, stripeToken: String, options:
RequestOptions): String = {
    val params = ChargeCreateParams.builder()
      .setAmount(amount)
      .setCurrency("EUR")
      .setDescription("testpayment")
      .setSource(stripeToken)
      .build()
    val charge = Charge.create(params, options)
    charge.getId
  }1

  def getStatus(id: String, options: RequestOptions): String = {
    Charge.retrieve(id, options).getStatus
  }
}

//Lines: 7
```

```scala
/* File: Main.scala */

  def main(args: Array[String]): Unit = {
    def options = RequestOptions.builder()
      .setApiKey(STRIPE_API_KEY)
      .build()
    val payment = new Payment()
    val id = payment.makePayment(1000L, PAYMENT_TOKEN, options)
    println(payment.getStatus(id, options))
  }

//Lines: 5
```

Bewertung

- Lines of Code: 12 Zeilen. → 4/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: In der Dokumentation werden alle Endpunkte dokumentiert und es gibt auch Beispiele für verschiedene Sprachen, darunter zwar Java aber leider nicht Scala. Da der Code in Scala allerdings fast derselbe ist wie der in Java geschriebene Code gibt es hier keinen Punkteabzug. → 5/5
- Unterstützte Paradigmen: Die Bibliothek unterstützt objektorientierte Programmierung, allerdings wird keine funktionale Programmierung berücksichtigt, diese wäre in diesem Fall sinnvoll, da man dadurch zum Beispiel mithilfe eines Observers auf Änderungen des Status achten könnte. → 3/5

Clojure

Code Snippet

Zuerst werden die Funktionen make-payment und get-status definiert, mit denen die benötigten Funktionen implementiert werden.

```
;;File: payment.clj

(defn make-payment
  [amount stripeToken ^RequestOptions options]
        (let [chargeParams (-> (ChargeCreateParams/builder)
                               (.setSource stripeToken)
                               (.setCurrency "EUR")
                               (.setDescription "testpayment")
                               (.setAmount amount)
                               (.build)
                               )]
        (.getId (Charge/create chargeParams options)))
)

(defn get-status
  [id requestOptions] (.getStatus (Charge/retrieve id requestOptions))
  )

;;Lines: 6
```

Als nächstes werden die RequestOptions definiert und die beiden Methoden werden aufgerufen und der Status der Zahlung wird auf die Konsole ausgegeben.

```
;;File: core.clj

(defn -main
  ([] (let [options (-> (RequestOptions/builder)
                        (.setApiKey STRIPE_API_KEY)
                        (.build))]
        (let [id (payment/make-payment 1000 PAYMENT_TOKEN options)]
          (println (payment/get-status id options)))
        )
   )
  )
;;Lines: 4
```

Bewertung

- Lines of Code: 10 Zeilen. → 5/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: In der Dokumentation werden alle Endpunkte dokumentiert und es gibt auch Beispiele für verschiedene Sprachen, darunter zwar Java aber leider nicht Clojure. Da der Code in Clojure trotz unterschiedlicher Syntax fast derselbe ist wie der in Java geschriebene Code gibt es hier keinen Punkteabzug.

→ 5/5

- Unterstützte Paradigmen: Die Bibliothek unterstützt objektorientierte Programmierung, allerdings wird funktionale Programmierung insofern nicht berücksichtigt dass man zum Beispiel mithilfe eines Observers auf Änderungen des Status achten könnte. Die Implementierung in Clojure (einer funktionalen Sprache) ist jedoch leicht möglich. → 3/5

Beurteilungstabelle und Fazit

Sprache	**Java**	**Kotlin**	**Groovy**	**Scala**	**Clojure**
Lines of Code	3	4	4	4	5
Lesbarkeit	4	5	5	5	5
Dokumentation	5	5	5	5	5
Unterstützte Paradigmen	3	3	3	3	3
Ergebnis	15	17	17	17	18

Figure 2. Beurteilungstabelle Stripe-API

Da Java die meisten Zeilen Code benötigt und durch das notwendige Exceptionhandling auch unlesbarer wird, belegt Java in diesem Fall den letzten Platz.

Dass Clojure auf dem ersten Platz liegt, liegt daran, dass die Schreibweise in Clojure am kompaktesten ist. Allgemein führt der Einsatz der Stripe-API nicht dazu, dass der Code unlesbarer wird, allerdings wäre es besser, wenn man den Status mit einem Observer asynchron überwachen könnte.

[10] Stripe: Stripe API, https://stripe.com/docs/api/ abgerufen am 04.04.2021

Rest APIs

In allen Sprachen wird die Rest-API von https://reqres.in/ verwendet. Als Code sample wird jeweils ein GET-Request und ein POST-Request abgesendet und das Resultat soll als Objekt in einer Variable abgespeichert werden.

In allen Sprachen wird die Feign-Bibliothek verwendet, da diese in allen Sprachen verwendet werden kann.
Die DTO Klassen werden nicht zur Bewertung herangezogen.

Java

Code Snippet

Um die Rest-API aufzurufen, wird ein Client erstellt, der die Funktionen der API deklariert.

```
/* File: UserFeignClient.java */

public interface UserFeignClient {
    @RequestLine("GET /users/{id}")
    GetUser getUser(@Param("id") int id);

    @RequestLine("POST /users")
    @Headers("Content-Type: application/json")
    CreateUser.Response createUser(CreateUser.Request createUser);
}
// Lines: 6
```

Anschließend wird ein Client mithilfe des FeignBuilders erstellt und die Funktionen werden aufgerufen.

```java
/* File: Main.java */

    public static void main( String[] args )
    {

        UserFeignClient client = Feign.builder()
                .client(new OkHttpClient())
                .encoder(new GsonEncoder())
                .decoder(new GsonDecoder())
                .target(UserFeignClient.class,
"https://reqres.in/api");


        GetUser getUserResponse = getUser(client);
        CreateUser.Response createUserResponse =
                createUser(client, new CreateUser.Request("Testuser",
"Programmer")));
    }

    public static GetUser getUser(UserFeignClient client) {
        return client.getUser(2);
    }

    public static CreateUser.Response createUser(UserFeignClient
client, CreateUser.Request request) {
        return client.createUser(request);
    }
// Lines: 8
```

Bewertung

- Lines of Code: 14 Zeilen. → 4/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Die Dokumentation [11] ist sehr umfangreich und bietet auch zahlreiche Beispiele zum Einsatz der Bibliothek, außerdem werden verschiedenste Encoder/Decoder vorgestellt, die von der Bibliothek unterstützt werden. → 5/5
- Unterstützte Paradigmen: Die OpenFeign Bibliothek unterstützt sowohl objektorientierte Programmierung als auch funktionale Programmierung (mit CompletableFuture Objekten). → 5/5

Kotlin

Code Snippet

Zuerst wird ein interface mit den beiden Methoden, die anschließend aufgerufen werden, deklariert.

```
/* File: UserFeignClient.kt */

interface UserFeignClient {
    @RequestLine("GET /users/{id}")
    fun getUser(@Param("id") id: Int): GetUser

    @RequestLine("POST /users")
    @Headers("Content-Type: application/json")
    fun createUser(createUser: CreateUserRequest): CreateUserResponse
}

// Lines: 6
```

Nun wird eine Instanz des UserFeignClients mithilfe des FeignBuilders erstellt.

```
/* File: Main.kt */

fun main() {
    val userFeignClient = Feign.builder()
            .client(OkHttpClient())
            .encoder(GsonEncoder())
            .decoder(GsonDecoder())
            .target(UserFeignClient::class.java,
"https://reqres.in/api")

    val getUserResponse = getUser(userFeignClient)
    val createdUser = createUser(userFeignClient, CreateUserRequest(
        name = "Testuser",
        job = "Programmer"
    ))
}

fun getUser(client: UserFeignClient): GetUser {
    return client.getUser(2)
}

fun createUser(client: UserFeignClient, user: CreateUserRequest):
CreateUserResponse {
    return client.createUser(user)
}

// Lines: 8
```

Bewertung

- Lines of Code: 14 Zeilen. → 4/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Die Dokumentation ist zwar sehr umfangreich und enthält viele Beispiele, allerdings gibt es leider keine Beispiele für den Umgang mit Kotlin. Da jedoch fast kein Unterschied bei der Umsetzung in Kotlin zu der Umsetzung in Java besteht, werden hierfür keine Punkte abgezogen→ 5/5
- Unterstützte Paradigmen: Die OpenFeign Bibliothek unterstützt sowohl objektorientierte Programmierung als auch funktionale Programmierung (mit CompletableFuture Objekten). → 5/5

Groovy

Code Snippet

Um auf die Rest-API zuzugreifen wird ein Interface mit den Methoden, die später aufgerufen werden, deklariert.

```
/* File: UserFeignClient.groovy */

interface UserFeignClient {
    @RequestLine("GET /users/{id}")
    GetUser getUser(@Param("id") int id);

    @RequestLine("POST /users")
    @Headers("Content-Type: application/json")
    CreateUser.Response createUser(CreateUser.Request createUser);
}

// Lines: 6
```

Mit dem FeignBuilder wird der Client instanziert und die Methoden werden aufgerufen.

```
/* File: Main.groovy */

    static main(args) {
        def client = Feign.builder()
                .client(new OkHttpClient())
                .encoder(new GsonEncoder())
                .decoder(new GsonDecoder())
                .target(UserFeignClient.class, "https://reqres.in/api")

        def user = getUser(client)
        def createdUser = client.createUser(new CreateUser.Request
("Testuser", "Programmer"))
    }

    static def getUser(UserFeignClient client) {
        client.getUser(2)
    }

    static def createUser(UserFeignClient client, CreateUser.Request
user) {
        client.createUser(user)
    }


//Lines: 8
```

Bewertung

- Lines of Code: 14 Zeilen. → 4/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Die Dokumentation ist zwar sehr umfangreich und enthält viele Beispiele, allerdings gibt es leider keine Beispiele für den Umgang mit Groovy. Da jedoch fast kein Unterschied bei der Umsetzung in Groovy zu der Umsetzung in Java besteht, werden hierfür keine Punkte abgezogen. → 5/5
- Unterstützte Paradigmen: Die OpenFeign Bibliothek unterstützt sowohl objektorientierte Programmierung als auch funktionale Programmierung (mit CompletableFuture Objekten). → 5/5

Scala

Code Snippet

Zuerst wird ein trait erstellt, in dem die Routen und Parameter definiert werden.

```scala
/* File: UserFeignClient.scala */

trait UserFeignClient {
  @RequestLine("GET /users/{id}")
  def getUser(@Param("id") id: Int): GetUser

  @RequestLine("POST /users")
  @Headers(Array[String]("Content-Type: application/json"))
  def createUser(createUser: CreateUserRequest): CreateUserResponse
}

// Lines: 6
```

Der Client wird mit dem FeignBuilder erstellt und anschließend werden die Methoden des Clients aufgerufen.

```scala
/* File: Main.scala */

  def main(args: Array[String]): Unit = {
    val userFeignClient = Feign.builder()
      .client(new OkHttpClient())
      .encoder(new GsonEncoder())
      .decoder(new GsonDecoder())
      .target(classOf[UserFeignClient], "https://reqres.in/api")

    val getUserResponse = getUser(client = userFeignClient)
    val createUserResponse = createUser(client = userFeignClient,
createUserRequest = CreateUserRequest(
       name = "Testuser",
       job = "Programmer"
    ))
  }

  def getUser(client: UserFeignClient) :GetUser = {
    client.getUser(2)
  }

  def createUser(client: UserFeignClient, createUserRequest:
CreateUserRequest): CreateUserResponse = {
    client.createUser(createUserRequest)
  }

// Lines: 8
```

Bewertung

- Lines of Code: 14 Zeilen → 4/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Die Dokumentation ist zwar sehr umfangreich und enthält viele Beispiele, allerdings gibt es leider keine Beispiele für den Umgang mit Scala. Da jedoch fast kein Unterschied bei der Umsetzung in Scala zu der Umsetzung in Java besteht, werden hierfür keine Punkte abgezogen. → 5/5
- Unterstützte Paradigmen: Die OpenFeign Bibliothek unterstützt sowohl objektorientierte Programmierung als auch funktionale Programmierung (mit CompletableFuture Objekten). → 5/5

Clojure

Code Snippet

Zuerst wird ein Interface definiert, in dem die REST-Methoden definiert werden, die aufgerufen werden sollen.

```
;; File: userFeignClient.clj

(definterface userFeignClient
  (^{RequestLine "GET /users/{id}"}  getUser [^{Param "id"} id])
  (^{RequestLine "POST /users"} ^{Headers ["Content-Type:
application/json"]} createUser [user] )
)

;; Lines: 3
```

Anschließend wird ein Client mit dem FeignBuilder instanziert und verwendet, um die Requests abzusenden.

```
;; File: core.clj

(defn getUser
  [client] (. client getUser 2)
)

(defn createUser
  [client createUserRequest] (. client createUser createUserRequest)
)

(defn -main
  ([] (let [client (-> (Feign/builder)
          (.client (new OkHttpClient))
          (.encoder (new GsonEncoder))
          (.decoder (new GsonDecoder))
          (.target userFeignClient/userFeignClient
"https://reqres.in/api"))]
          (let [getUserResponse (getUser client)])
          (let [createUserResponse (createUser client {:name "Testuser"
:job "Programmer"})])
        )
    )
)

;; Lines: 8
```

Bewertung

- Lines of Code: 11 Zeilen → 5/5
- Lesbarkeit: Der Code ist leicht verständlich, allerdings sorgen die Annotationen

beim Interface dafür, dass der Code etwas unübersichtlich wird. → 4/5

- Dokumentation: Die Dokumentation ist zwar sehr umfangreich und enthält viele Beispiele, allerdings gibt es leider keine Beispiele für den Umgang mit Clojure. Da Clojure sich syntaktisch stärker von Java unterscheidet als die anderen berücksichtigten JVM Sprachen werden hier Punkte abgezogen. → 3/5
- Unterstützte Paradigmen: Die OpenFeign Bibliothek unterstützt sowohl objektorientierte Programmierung als auch funktionale Programmierung (mit CompletableFuture Objekten). → 5/5

Beurteilungstabelle und Fazit

Sprache	**Java**	**Kotlin**	**Groovy**	**Scala**	**Clojure**
Lines of Code	4	4	4	4	5
Lesbarkeit	5	5	5	5	4
Dokumentation	5	5	5	5	3
Unterstützte Paradigmen	5	5	5	5	5
Ergebnis	19	19	19	19	17

Figure 3. Beurteilungstabelle Rest-APIs

Die Ergebnisse sind bei allen Sprachen sehr ähnlich, da in allen Sprachen dieselbe Bibliothek verwendet werden konnte und diese Biblothek auch sehr gut geeignet ist, um Rest-APIs anzusteuern.

Clojure belegt aufgrund der Dokumentation, die nur für Java verfasst wurde, den letzten Platz, außerdem wird der Code durch die Annotationen in Clojure etwas unübersichtlich.

[11] OpenFeign: Feign makes writing Java http clients easier, https://github.com/OpenFeign/feign#readme abgerufen am 03.04.2021

Stream API

In allen Sprachen soll mit der Stream API (oder der Alternative in der jeweiligen Sprache) eine Liste von Lebensmitteln in insgesamt drei Methoden:

- nach dem Namen sortiert werden.
- nach der Kategorie gruppiert werden.
- nach Lebensmitteln durchsucht werden, die weniger als einen Euro kosten.

Die Liste soll unverändert bleiben.

Java

In Java wird die Stream API verwenden.

Code Snippet

```
public static List<Food> sortFoodByName(List<Food> food) {
    return food.stream().sorted(Comparator.comparing(Food::getName
)).collect(Collectors.toList());
}

public static Map<Category, List<Food>> groupFoodByCategory(List<Food>
food) {
    return food.stream().collect(Collectors.groupingBy(Food:
:getCategory));
}

public static List<Food> findFoodWorthLessThanAEuro(List<Food> food) {
    return food.stream().filter(f -> f.getPrice() < 1.0).collect
(Collectors.toList());
}

//Lines: 6
```

Bewertung

- Lines of Code: 6 Zeilen → 5/5
- Lesbarkeit: Der Code ist leicht verständlich, im Vergleich zu den anderen Sprachen fällt jedoch auf, dass die sorted(), groupingBy() und filter() Methoden

nicht direkt auf dem Listenobjekt sind, sondern auf einem Stream Objekt sind. Außerdem muss der Stream in zwei von drei Fällen wieder mit einem eigenen Call zu einer Liste umgewandelt werden. → 3/5

- Dokumentation: Die Dokumentation [12] enthält zahlreiche Beispiele und dokumentiert die komplette Stream API. → 5/5
- Unterstützte Paradigmen: Die Stream API sorgt dafür, dass Collections funktional verarbeitet werden können, die gestreamte Liste wird nicht verändert. → 5/5

Kotlin

In Kotlin wird keine Stream API verwendet, da die Listen in Kotlin bereits die Funktionalitäten der Stream API implementieren.

Code Snippet

```
fun sortFoodByName(food: List<Food>): List<Food> = food.sortedBy { it
.category }

fun groupFoodByCategory(food: List<Food>): Map<Category, List<Food>> =
food.groupBy(Food::category)


fun findFoodWorthLessThanAEuro(food: List<Food>): List<Food> = food
.filter { f->f.price<1.0 }

//Lines: 6
```

Bewertung

- Lines of Code: 6 Zeilen → 5/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Da die Dokumentation von Collections in der Kotlinlang Dokumentation [13] vollständig ist, die Dokumentation eine Suchfunktion hat und bei den jeweiligen Funktionen auch Anwendungsbeispiele vorhanden sind erhält Kotlin alle Punkte für die Dokumentation. → 5/5

- Unterstützte Paradigmen: Die Methoden, die statt der Stream API verwendet werden, sind funktionale Konstrukte und die Listen, die verwendet werden, werden nicht verändert. → 5/5

Groovy

In Groovy wird die Stream API nicht verwendet, da die Listen in Groovy bereits die Funktionen der Stream API implementieren.

Code Snippet

```
static def sortFoodByName(List<Food> food) {
    food.toSorted {f->f.name}
}

static def groupFoodByCategory(List<Food> food) {
    food.groupBy {f->f.category}
}

static def findFoodWorthLessThanAEuro(List<Food> food) {
    food.findAll {f->f.price < 1}
}

//Lines: 6
```

Bewertung

- Lines of Code: 6 Zeilen → 5/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Die Dokumentation von Groovy für den Umgang mit Collections [14] enthält gute Beispiele und Beschreibungen für alle Methoden, die equivalent zu den Funktionen der Stream API in Java sind. → 5/5
- Unterstützte Paradigmen: Die Methoden, die statt der Stream API verwendet werden, sind funktionale Konstrukte und die Listen, die verwendet werden, werden nicht verändert. → 5/5

Scala

In Scala wird die Stream API nicht verwendet, da die Listen in Scala bereits die Funktionen der Stream API implementieren.

Code Snippet

```
def sortFoodByName(food: List[Food]): List[Food] = {
  food.sortBy(f => f.name)
}

  def groupFoodByCategory(food: List[Food]): Map[String, List[Food]] =
{
  food.groupBy(f => f.category)
}

def findFoodWorthLessThanAEuro(food: List[Food]): List[Food] = {
  food.filter(f => f.price<1)
}

//Lines: 6
```

Bewertung

- Lines of Code: 6 Zeilen → 5/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: In der Dokumentation von Scala [15] werden die Funktionen, die statt der Stream API verwendet werden können, dokumentiert und manche (nicht alle, zum Beispiel die "filter" Funktion) haben auch Beispiele dabei. Da nicht überall Beispiele sind gibt es hier einen Punkt Abzug, da alle anderen berücksichtigten JVM Sprachen mehr Beispiele gebracht haben. → 4/5
- Unterstützte Paradigmen: Die Methoden, die statt der Stream API verwendet werden, sind funktionale Konstrukte und die Listen, die verwendet werden, werden nicht verändert. → 5/5

Clojure

In clojure.core gibt es bereits die Funktionen der Stream API, deshalb wird die Stream API nicht verwendet.

Code Snippet

```
(defn sort-food-by-name
  [food] (sort-by :name food))

(defn group-food-by-category
   [food] (group-by :category food))

(defn find-food-worth-less-than-a-euro
  [food] (filter #(< (:price %) 1) food))

;;Lines: 6
```

Bewertung

- Lines of Code: 6 Zeilen → 5/5
- Lesbarkeit: Der Code ist leicht verständlich. → 5/5
- Dokumentation: Die Dokumentation, die in dem Cursive Plugin von dem IntelliJ Marktplatz [16] inkludiert ist verfügt pro Funktion, die statt einer Stream API Funktion verwendet wird, sehr viele Beispiele mit verschiedenen Anwendungsfällen und erklärt auch die Funktion sehr detailliert, aus diesem Grund bekommt Clojure alle Punkte. → 5/5
- Unterstützte Paradigmen: Clojure ist eine funktionale Programmiersprache und die Funktionen, die anstelle der Stream API verwendet werden, sind auch funktional konzipiert. → 5/5

Beurteilungstabelle und Fazit

Sprache	Java	Kotlin	Groovy	Scala	Clojure
Lines of Code	5	5	5	5	5
Lesbarkeit	3	5	5	5	5
Dokumentation	5	5	5	4	5
Unterstützte Paradigmen	5	5	5	5	5
Ergebnis	18	20	20	19	20

Figure 4. Beurteilungstabelle Stream-API und Alternativen

Mit 18 Punkten belegt Java den letzten Platz, das liegt daran, dass der Code in Java durch den Einsatz der Stream-API etwas unleserlicher wird als der Code der anderen Sprachen.

Ansonsten haben alle Sprachen 19-20 Punkte, das liegt daran, dass die Alternativen zur Stream-API in den anderen Sprachen direkt auf den Collections sind (beziehungsweise core Funktionen in Clojure).

[12] Oracle: Stream documentation, https://docs.oracle.com/javase/8/docs/api/?java/util/stream/Stream.html abgerufen am 05.04.2021

[13] KotlinLang: Documentation, https://kotlinlang.org/docs/ abgerufen am 05.04.2021

[14] Groovy-lang: Working with collections, http://docs.groovy-lang.org/next/html/documentation/working-with-collections.html abgerufen am 05.04.2021

[15] Scala-lang: List documentation, https://www.scala-lang.org/api/2.12.5/scala/collection/immutable/List.html abgerufen am 05.04.2021

[16] Ideogram Limited: Cursive (Clojure IntelliJ Plugin), https://plugins.jetbrains.com/plugin/8090-cursive abgerufen am 05.04.2021

Java Mail API

In Java, Kotlin, Groovy und Scala wird die javax.mail Bibliothek verwendet, in Clojure hingegen wird die Bibliothek postal [17] verwendet.

In jeder Sprache soll eine simple E-Mail mit einem Text versendet werden.

Java

Code Snippet

Zuerst werden die Eigenschaften der SMTP-Session festgelegt (host, ssl und authentication). Anschließend wird eine Session instanziert und mithilfe dieser Session wird über die statische Methode des Transport Objekts eine Nachricht versendet.

```java
public static void sendEmail(String body) {
    Properties properties = new Properties();
    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.ssl.enable", true);
    properties.put("mail.smtp.auth", true);

    Session session = Session.getInstance(properties, new
Authenticator() {
                @Override
                protected PasswordAuthentication
getPasswordAuthentication() {
                    return new PasswordAuthentication("5devhif",
MAIL_PASSWORD);
                }
            });

    try {
        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress("5DevHIF"));
        message.setRecipient(Message.RecipientType.TO, new
InternetAddress("fre18149@spengergasse.at"));
        message.setSubject("Message from Java");
        message.setText(body);

        Transport.send(message);
    } catch (AddressException e) {
        e.printStackTrace();
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    sendEmail("This message has been sent via javax.mail");
}

//Lines: 22
```

Bewertung

- Lines of Code: 22 Zeilen → 0/5
- Lesbarkeit: Der Code ist sehr umständlich, vor allem durch den Einsatz des Properties Objektes, da hier die Werte per String zugewiesen werden müssen wodurch die Fehleranfälligkeit steigt. Die Benötigte try/catch Anweisung sorgt auch nicht dafür, dass der Code Lesbarer wird. → 1/5
- Dokumentation: Die Dokumentation der javax.mail API [18] bietet eine Auflistung der Properties, die gesetzt werden können, zeigt allerdings nicht, wie man eine

authentifizierte Session erstellt. → 2/5

- Unterstützte Paradigmen: Die javax.mail API ist für objektorientierte Programmierung gebaut und funktionale oder reaktive Programmierung wird nicht unterstützt. Das wäre eventuell sinnvoll, um Rückmeldungen des Servers besser verarbeiten zu können. → 2/5

Kotlin

Code Snippet

```
fun sendEmail(body: String?) {
    val properties = Properties()
    properties["mail.smtp.host"] = "smtp.gmail.com"
    properties["mail.smtp.ssl.enable"] = true
    properties["mail.smtp.auth"] = true
    val session = Session.getInstance(properties, object :
Authenticator() {
            override fun getPasswordAuthentication():
PasswordAuthentication {
                return PasswordAuthentication("5devhifmailtest",
MAIL_PASSWORD)
            }
        })

    val message: Message = MimeMessage(session).apply {
        setFrom(InternetAddress("5DevHIF"))
        setRecipient(Message.RecipientType.TO, InternetAddress
("fre18149@spengergasse.at"))
        subject = "Message from Kotlin"
        setText(body)
    }
    Transport.send(message)
}

fun main(args: Array<String>) {
    sendEmail("This message has been sent via javax.mail")
}

//Lines: 16
```

Bewertung

- Lines of Code: 16 Zeilen → 2/5
- Lesbarkeit: Der Code ist, wie in Java auch, sehr umständlich, vor allem durch

den Einsatz des Properties Objektes, da hier die Werte per String zugewiesen werden müssen, was zu einer hohen Fehleranfälligkeit führt. Da hier allerdings kein try/catch Konstrukt benötigt wird und der Einsatz von apply beim Erstellen der MimeMessage die Lesbarkeit verbessert, schneidet Kotlin in diesem Fall besser als Java ab. → 3/5

- Dokumentation: Die Dokumentation der javax.mail API bietet eine Auflistung der Properties, die gesetzt werden können, zeigt allerdings nicht, wie man eine authentifizierte Session erstellt. → 2/5
- Unterstützte Paradigmen: Die javax.mail API ist für objektorientierte Programmierung gebaut und funktionale oder reaktive Programmierung wird nicht unterstützt. Das wäre eventuell sinnvoll, um Rückmeldungen des Servers besser verarbeiten zu können. → 2/5

Groovy

Code Snippet

```
static def sendEmail(body) {
    Properties properties = new Properties()
    properties["mail.smtp.host"] = "smtp.gmail.com"
    properties["mail.smtp.ssl.enable"] = true
    properties["mail.smtp.auth"] = true
    Session session = Session.getInstance(properties, new
Authenticator() {
        @Override
        protected PasswordAuthentication getPasswordAuthentication() {
            new PasswordAuthentication("5devhifmailtest", MAIL_PASS)
        }
    })

    Message message = new MimeMessage(session)
    message.setFrom(new InternetAddress("5DevHIF"))
    message.setRecipient(Message.RecipientType.TO, new InternetAddress
("fre18149@spengergasse.at"))
    message.subject = "Message from Groovy"
    message.setText(body)

    Transport.send(message)
}

static main(args) {
    sendEmail("This message has been sent via javax.mail")
}

//Lines: 17
```

Bewertung

- Lines of Code: 17 Zeilen → 1/5
- Lesbarkeit: Der Code ist, wie in Java auch, sehr umständlich, vor allem durch den Einsatz des Properties Objektes, da hier die Werte per String zugewiesen werden müssen, was zu einer hohen Fehleranfälligkeit führt. Da hier allerdings kein try/catch Konstrukt benötigt wird schneidet Groovy in diesem Fall besser als Java ab. → 2/5
- Dokumentation: Die Dokumentation der javax.mail bietet eine Auflistung der Properties, die gesetzt werden können, zeigt allerdings nicht, wie man eine authentifizierte Session erstellt. → 2/5
- Unterstützte Paradigmen: Die javax.mail API ist für objektorientierte Programmierung gebaut und funktionale oder reaktive Programmierung wird

nicht unterstützt. Das wäre eventuell sinnvoll, um Rückmeldungen des Servers besser verarbeiten zu können. → 2/5

Scala

Code Snippet

```
def sendEmail(body: String): Unit = {
  val properties: Properties = new Properties
  properties.put("mail.smtp.host", "smtp.gmail.com")
  properties.put("mail.smtp.ssl.enable", true)
  properties.put("mail.smtp.auth", true)
  val session = Session.getInstance(properties, new Authenticator {
    override def getPasswordAuthentication: PasswordAuthentication = {
      new PasswordAuthentication("5devhif", MAIL_PASS)
    }
  })

  val message: Message = new MimeMessage(session)
  message.setFrom(new InternetAddress("5DevHIF"))
  message.setRecipient(Message.RecipientType.TO, new InternetAddress
("fre18149@spengergasse.at"))
  message.setSubject("Message from Scala")
  message.setText(body)

  Transport.send(message)
}

def main(args: Array[String]): Unit = {
  sendEmail("This message has been sent via javax.mail")
}

//Lines: 16
```

Bewertung

- Lines of Code: 17 Zeilen → 1/5
- Lesbarkeit: Der Code ist, wie in Java auch, sehr umständlich, vor allem durch den Einsatz des Properties Objektes, da hier die Werte per String zugewiesen werden müssen, was zu einer hohen Fehleranfälligkeit führt. Da hier allerdings kein try/catch Konstrukt benötigt wird schneidet Scala in diesem Fall besser als Java ab. → 2/5

- Dokumentation: Die Dokumentation der javax.mail API bietet eine Auflistung der Properties, die gesetzt werden können, zeigt allerdings nicht, wie man eine authentifizierte Session erstellt. → 2/5
- Unterstützte Paradigmen: Die javax.mail API ist für objektorientierte Programmierung gebaut und funktionale oder reaktive Programmierung wird nicht unterstützt. Das wäre eventuell sinnvoll, um Rückmeldungen des Servers besser verarbeiten zu können. → 2/5

Clojure

Code Snippet

```
(defn send-email
  [message] (postal/send-message {:host "smtp.gmail.com"
                                  :user "5devhif"
                                  :pass MAIL_PASSWORD
                                  :ssl true}
                                 {:from "5DevHIF"
                                  :to "fre18149@spengergasse.at"
                                  :subject "Message from Clojure"
                                  :body message} )
)

(defn -main
([] (send-email "This message has been sent via Postal")))

;;Lines: 4
```

Bewertung

- Lines of Code: 4 Zeilen → 5/5
- Lesbarkeit: Der Code ist sehr leicht verständlich und effizient. → 5/5
- Dokumentation: Die Dokumentation der postal Bibliothek [19] ist umfangreich, hat sehr viele Beispiele. → 5/5
- Unterstützte Paradigmen: Die postal Bibliothek wurde für Clojure designed, folglich unterstützt die postal Bibliothek auch funktionale Programmierung. → 5/5

Beurteilungstabelle und Fazit

Sprache	Java	Kotlin	Groovy	Scala	Clojure
Lines of Code	0	2	1	2	5
Lesbarkeit	1	3	2	2	5
Dokumentation	2	2	2	2	5
Unterstützte Paradigmen	2	2	2	2	5
Ergebnis	5	9	7	8	20

Figure 4. Beurteilungstabelle Stream-API und Alternativen

Clojure sticht hier deutlich heraus, da eine andere Bibliothek verwendet wurde, die extra für Clojure designed wurde. Das größte Problem bei der javax.mail API ist, dass Properties verwendet werden, obwohl sich eine eigene Konfigurationsklasse deutlich besser zum Erstellen einer SMTP Session eignen würde, da dadurch Fehler durch hardgecodete Strings vermieden werden könnten.

[17] Drewr: com.draines/postal, https://clojars.org/com.draines/postal abgerufen am 06.04.2021

[18] JavaEE: JavaMail API documentation, https://javaee.github.io/javamail/docs/api/ abgerufen am 06.04.2021

[19] CljDoc: postal documentation, https://cljdoc.org/d/com.draines/postal/2.0.4/doc/readme abgerufen am 06.04.2021

Fazit

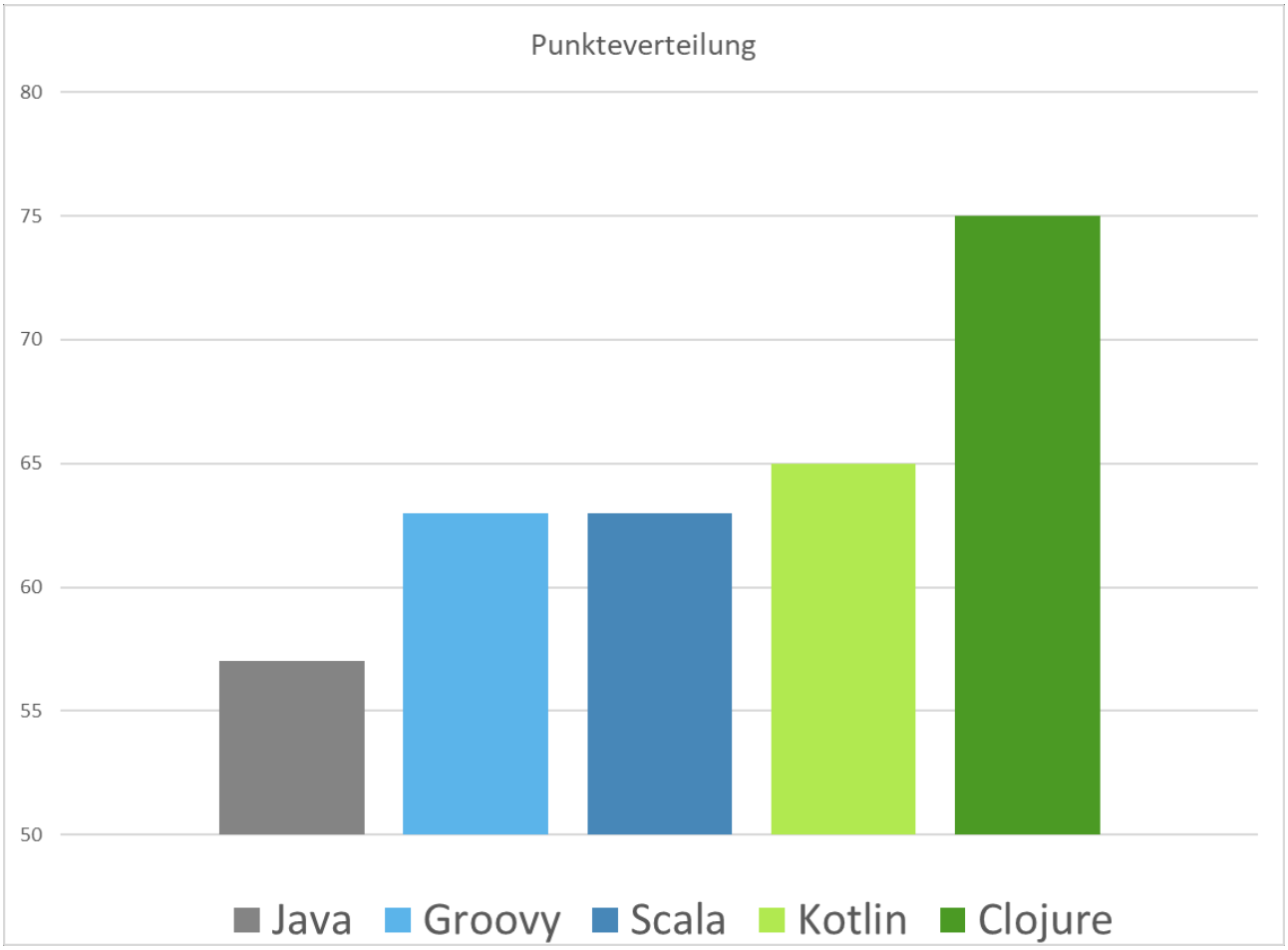


Abbildung 2. Abschlusstabelle

Wie in dem Diagramm zu sehen ist, belegt Clojure mit 75 Punkten den ersten Platz, gefolgt von Kotlin auf dem zweiten Platz mit 65 Punkten. Auf dem dritten Platz sind sowohl Groovy als auch Scala mit jeweils 63 Punkten und auf dem letzten Platz ist Java mit 57 Punkten.

Dass Java den letzten Platz belegt ist insofern nicht überraschend, da die anderen Sprachen alle das Ziel haben, effizienter zu sein. Auch, dass Clojure auf dem ersten Platz ist, ist kein Wunder, da die Syntax von Clojure es einem erlaubt, sehr effizienten Code in nur wenigen Zeilen zu schreiben.
Allerdings ist wichtig zu erwähnen, dass Clojure die längste Einarbeitungszeit benötigt, wenn man schon eine der anderen berücksichtigten JVM Sprachen kann.

References

FreeJavaGuide: History of Java programming language, https://www.freejavaguide.com/history.html abgerufen am 25.03.2021

Toibe: TIOBE Index for April 2021, https://www.tiobe.com/tiobe-index/ abgerufen am 05.04.2021

Deshmane, Rohini: Introduction to Kotlin, https://medium.com/@rohinideshmane.21/introduction-to-kotlin-5f39b31610e0 abgerufen am 25.03.2021

Kotlinlang: Calling Java from Kotlin, https://kotlinlang.org/docs/java-interop.html abgerufen am 25.03.2021

Scand: Groovy vs Java: Detailed Comparison and Tips on the Language Choice, https://scand.com/company/blog/groovy-vs-java/ abgerufen am 05.04.2021

Groovy-lang: A multi-faceted language for the Java platform, https://groovy-lang.org/ abgerufen am 05.04.2021

Hicks, Matt: Scala vs. Java: Why Should I Learn Scala?, https://www.toptal.com/scala/why-should-i-learn-scala abgerufen am 05.04.2021

javaTpoint: History of Scala, https://www.javatpoint.com/history-of-scala abgerufen am 05.04.2021

Hickey, Rich: A history of Clojure, https://download.clojure.org/papers/clojure-hopl-iv-final.pdf abgerufen am 05.04.2021

Stripe: Stripe API, https://stripe.com/docs/api/ abgerufen am 04.04.2021

OpenFeign: Feign makes writing Java http clients easier, https://github.com/OpenFeign/feign#readme abgerufen am 03.04.2021

Oracle: Stream documentation, https://docs.oracle.com/javase/8/docs/api/?java/util/

stream/Stream.html abgerufen am 05.04.2021

KotlinLang: Documentation, https://kotlinlang.org/docs/ abgerufen am 05.04.2021

Groovy-lang: Working with collections, http://docs.groovy-lang.org/next/html/documentation/working-with-collections.html abgerufen am 05.04.2021

Scala-lang: List documentation, https://www.scala-lang.org/api/2.12.5/scala/collection/immutable/List.html abgerufen am 05.04.2021

Ideogram Limited: Cursive (Clojure IntelliJ Plugin), https://plugins.jetbrains.com/plugin/8090-cursive abgerufen am 05.04.2021

Drewr: com.draines/postal, https://clojars.org/com.draines/postal abgerufen am 06.04.2021

JavaEE: JavaMail API documentation, https://javaee.github.io/javamail/docs/api/ abgerufen am 06.04.2021

CljDoc: postal documentation, https://cljdoc.org/d/com.draines/postal/2.0.4/doc/readme abgerufen am 06.04.2021

Vergleich des Java EE Standard-Frameworks im Vergleich zu Spring zur Entwicklung von Business Applications

Paul Maurovich <mau18259@spengergasse.at>

Abstract

In diesem Paper werden die zwei (Web)-Frameworks Java EE und Spring auf unterschiedliche Kriterien untersucht:

- **Regelmäßige Updates und Wartbarkeit**: Wann und wie oft kommen Updates? Ist noch mit Updates zu rechnen? Wie wartbar ist der Code/das Framework?
- **Langfristiger Support**: Unter welcher Leitung gibt es noch Support? Ist mit langfristigem Framework Support zu rechnen?
- **Kosten und Programmiereleganz**: Ist das Framework Open Source, welche Lizenzkosten, welche Programmierparadigmen und welche allgemeinen Prinzipien gibt es?
- **Dokumentation**: Gibt es vorhandene, übersichtliche und inhaltlich aussagekräftige Dokumentationen und Anleitungen? Was beinhaltet die Dokumentation?
- **Funktionsumfang**: Welche Funktionen und welche Unterschiede gibt es zwischen den Frameworks?
- **Querschnittsfunktion**: Welche Metriken, Dependency Injection und welche Querschnittsfunktionen gibt es?
- **Developer Codebase und Community Support**: Wie groß ist die Community, gibt es genügende Hilfestellungen und Informationen im Internet zu dem Framework und wie aktuell wird das jeweilige Framework genutzt?

Die aus den Kriterien resultierenden Vor- und Nachteile werden gegenübergestellt und bewertet.

Um die Arbeit angemessen verstehen zu können, werden fortgeschrittene Fähigkeiten und Erfahrungen in Informatik bzw. in der Programmierung vorausgesetzt. Das Paper dient als Basis einer Entscheidungsgrundlage des zu wählenden Frameworks für die Entwicklung von Business Applikationen.

Einleitung

Java Entwickler und Java Entwicklerinnen Teams stehen, sei es im Projekt für einen Kunden/einer Kundin oder für das eigene Unternehmen, vor der Entscheidung, welches (Web)-Framework sie für die Entwicklung von Business Applikationen im Projekt wählen sollten. Sei es *Hibernate*, *Spring*, *Apache Wicket*, *Grails* oder *Java EE* bzw. *Jakarta EE*, für solch ein Projekt stehen viele Frameworks zur Auswahl. [20] Die Entscheidung fällt somit schwer, welches Framework am besten für das Projekt geeignet ist, weswegen hierfür in diesem Paper auf die zwei Frameworks Spring und Java EE, speziell auf die Entwicklung von Business Applikationen und Web Lösungen (REST), eingegangen wird.

Zu den jeweiligen Kategorien wird eine Bewertung in Form von Punkten vergeben, welchen den positiven Nutzungsgrad widerspiegeln. Hierfür werden Punkte auf einer Skala von minimal 0 Punkten bis maximal 5 Punkten vergeben, wobei 0 Punkte für *"Nicht ausreichend bzw. nicht vorhanden"* und 5 Punkte für *"Sehr gut"* steht. Die Punktezahl wird am Ende jeder Kategorie evaluiert und am Ende des Papers wird ein finales resümierendes Fazit abgegeben.

Wichtig zu beachten ist, dass es kein "perfektes Framework" für die Entwicklung von Business Applikationen gibt, denn es kommt ganz darauf an, was die **spezielle und individuelle** Business Lösung können und welche Rahmenbedingungen, wie Kosten, Umfang und Entwicklungsstandards, es erfüllen soll.

[20] Shankar, Ramya (2021): 10 Best Java Frameworks to Use in 2021, https://hackr.io/blog/java-frameworks abgerufen am 06.04.2021

Java EE Framework

Java Enterprise Edition, kurz Java EE, früher auch J2EE genannt, ist ein Framework, ausgelegt für mittel bis große Projekte, welches es ermöglicht, umfassende und meist komplizierte (Unternehmens)-Anwendungen zu entwickeln. Vor allem für Webanwendungslösungen im Unternehmensbereich, meist in Kombination mit REST-API Schnittstellen und verteiltem Computing, kommt Java EE häufig zum Einsatz. Das Framework baut auf *Java SE* (Java Standard Edition) auf und wird meist mittels einer Monolithen-Architektur umgesetzt. Seit Oracle die Leitung für das Framework abgegeben hat, wurde es auf *Jakarta EE* umbenannt und Fokus auf *Cloud Migration* gelegt. Java EE baut auf offiziell auf der Verwendung von der objektorientierten Programmiersprache Java, inoffiziell könnte es auch mit anderen Programmiersprachen wie Kotlin funktionieren. [21]

Benutzt wird es hauptsächlich zur Entwicklung von browserbasierten Web- und Enterprise (Business)-Applikationen und bietet zahlreiche Spezifikationen wie beispielsweise:

- **Web**: *Web(Servlets)* für die Handhabung von Web Requests und Responses; *WebSockets*; *Java Server Faces*
- **Web Service**: *Java API* für RESTful Webservices, *JSON Processing* und *Binding* sowie *XML Webservices (SOAP)*
- **Enterprise**: Container für *Dependency Injection*; *Enterprise JavaBean APIs*; *Java Persistence- und Transaction API*; Senden und Empfangen von Enterprise Nachrichten durch *Java Message Service*
- **Weitere**: *Bean Validation API*; Lang laufende Hintergrundaufgaben werden durch *Batch Anwendungen* unterstützt; Anbindung von Java-Servern an Enterprise Information Systemen durch die *Java EE Connector Architektur* [22]
- **JPA und JMS**: Datenbank Verbindungen und Datenbank Transaktionen

- **Bean Validation**: Deklariert eindeutige Constraints über eine Methode, ein Feld oder einer Klasse einer *JavaBeans-Komponente*
- **Application Server**: Genaue Konfiguration notwendig, jedoch vertiefende eigene Definierungen möglich [23]

Eine grundlegende Java EE Architektur sieht wie folgt aus: [24]

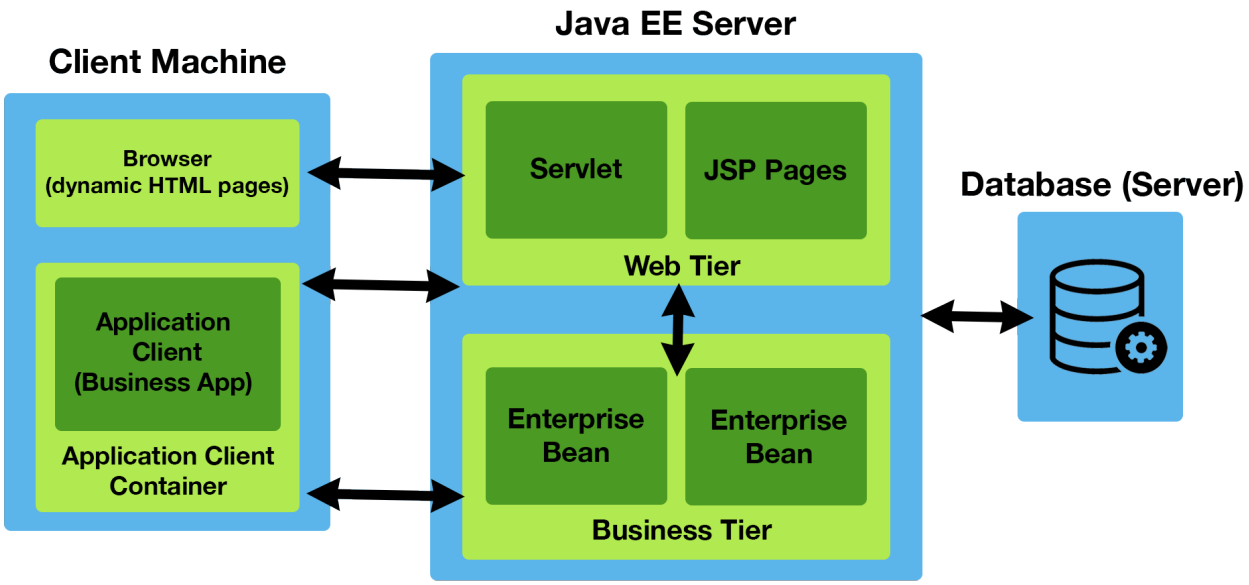


Figure 1. Java EE Architektur Überblick

Die Architektur besteht hauptsächlich aus drei Komponenten. Die *"Client Machine"* ist das Endgerät: *Browser* und/oder die *Business-Applikation*. Diese Ebene hat Verbindungen zum zweiten Teil der Architektur, dem *"Java EE Server"*, welcher den Web Container mit den *Web Servlets* sowie den *JSP Pages* (Jakarta Server Pages) und dem *Business Container*, wo sich die *Enterprise Beans* befinden, beinhaltet. Der Server ist mit einer *Datenbank* bzw. einem *Datenbank-Server* verbunden, worüber Datenbanktransaktionen mittels *"CRUD"-Operationen* durchgeführt werden können. Transaktionen können zum Beispiel mit einer REST-API oder mit einer SOAP-API abgewickelt werden.

[21] Wikipedia: Jakarta EE, https://de.wikipedia.org/wiki/Jakarta_EE abgerufen am 06.04.2021

[22] JavaTPoint: Java EE Specifications, https://www.javatpoint.com/java-ee abgerufen am 06.04.2021

[23] O'Reilly: Java Platform: Enterprise Edition, https://www.oreilly.com/library/view/java-ee-6/9781449338329/ch01.html abgerufen am 06.04.2021

[24] Oracle Docs: Distributed Multitiered Applications, https://docs.oracle.com/javaee/7/tutorial/overview003.htm abgerufen am 06.04.2021

Spring Framework

Das Spring Framework ist eine Erweiterung von der Java EE Plattform ausgehend, hauptsächlich für die Entwicklung von Java Applikationen im Hinblick auf Web Services und REST Schnittstellen. Vor allem die zwei Komponenten von Spring, Spring Boot und Spring MVC, werden dafür häufig in Gebrauch gebracht. Hauptaugenmerk liegt auf der Entwicklung von Business Logik, ohne dass viel Konfiguration von der Umgebung sowie Spezifikationen notwendig ist. [25]

Spring (Boot) kommt mit einem automatisch konfigurierten Anwendungskontext sowie einem Web Server, wie "Tomcat", um den Einstieg zu erleichtern und aufwendiges Aufsetzen des Frameworks zu vermeiden. Wohingegen Java EE offiziell nur mit der Programmiersprache Java läuft, so benötigt das Spring Framework keine spezifische Sprache, es wird jedoch meist in Kombination mit Java, Groovy oder Kotlin verwendet. [26] [27]

Der Schwerpunkt von Spring (Boot) liegt auf der Entwicklung von Microservices. Es ist größtenteils dafür ausgelegt und verfügt, wie auch Java EE, über einige Module. Diese sind zum Beispiel:

- **Datenbank Zugang, Daten Integrität**: JDBC, JMS, ORM, OXM, Transaktionen
- **Web**: Servlet, (Web)-Socket, Portlet
- **Core Container**: Beans, Core, Context und SpEL
- **Testing**: JUnit, Mockito etc.
- **Weitere**: AOP, Aspects, Messaging [28]

In diesem Beispiel wird sich auf die Spring Komponente **Spring Boot** fokussiert. Die Architektur dessen sieht wie folgt aus: [29] [30]

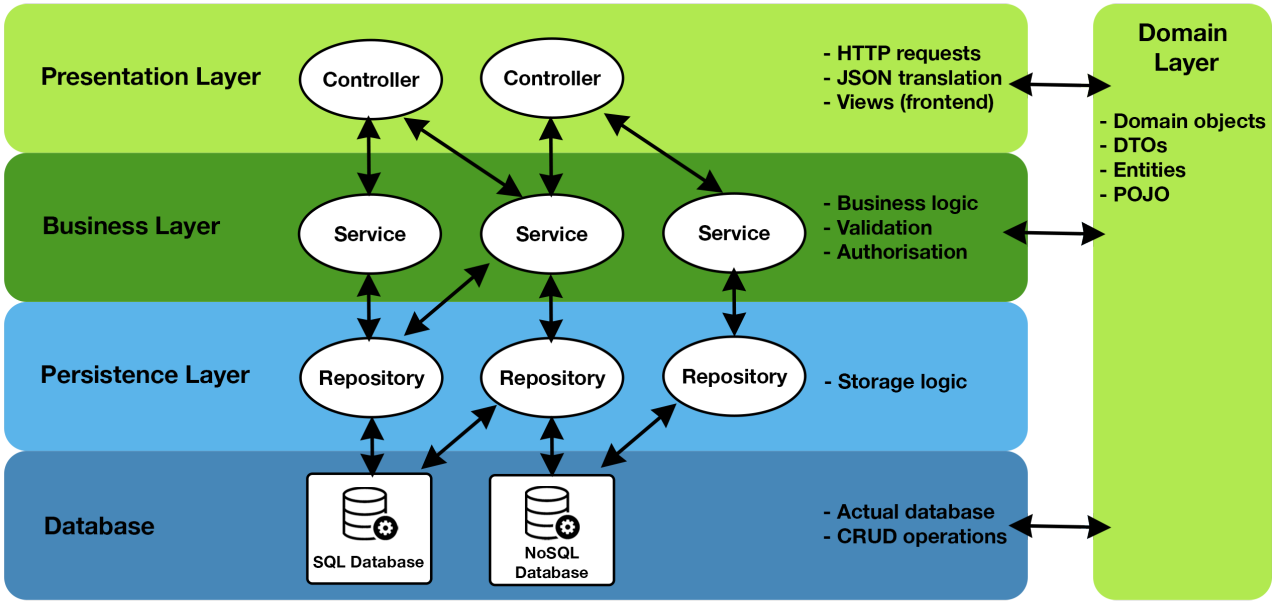


Figure 2. Spring Boot Framework Architektur Überblick

Die Spring Boot Framework Architektur besteht aus fünf Teilen. Die *"Presentation Layer"* Ebene, welche sich zum Beispiel um eingehende und ausgehende HTTP Requests/Responses kümmert. Sie übersetzt die Daten von und zu JSON (Objekten), um den datenmässigen Umgang mit der Applikationen zu gewährleisten. Grundsätzlich ist sie mit einer Frontend HTML oder XML Seite, oder der Business Applikation, verbunden.

Die *"Business Logic"* Ebene beinhaltet die Services, welche mit Business Logik ausgestattet sind. Es ist auch möglich "Fassaden" zwischen den Ebenen zu platzieren, um Datenintegrität und eine bessere Zugriffskontrolle zu gewährleisten. Die eingehenden Daten werden in diese Ebene transferiert und dort validiert bzw. autorisiert.

Um Daten abzugleichen und auf die gewählten Speicher Lösungen zugreifen zu können, wird die *"Persistence Layer"* Ebene benötigt. Sie verfügt über mehrere Repositorien, welche Datenbankzugriffe festlegen bzw. im Grunde eine geeignete Datenbankabfrage im Hintergrund ausführen.

Die *"Domain Layer"* Ebene agiert mit der *"Presentation-, Business- und Database-*

Ebene" und beinhaltet Domänen Objekte, *DTO* Klassen, (Domain)-Entities sowie *POJO*.

Schlussendlich werden die Daten beispielsweise in einer SQL und NoSQL Datenbank gespeichert und dort verfügbar gemacht. Die *"Persistence Layer"* Ebene führt mit der *"Database Layer"* Ebene *"CRUD"-Operationen* (Create, Read, Update und Delete) durch, welche das Hinzufügen, Lesen, Aktualisieren und Löschen von Daten bereitstellen.

[25] Spring: Spring Framework, https://spring.io/projects/spring-framework abgerufen am 06.04.2021

[26] Spring Docs: Language Support, https://docs.spring.io/spring-framework/docs/current/reference/html/languages.html abgerufen am 06.04.2021

[27] Oracle Docs: Java EE language support, https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.htmlL abgerufen am 06.04.2021

[28] Spring Docs: Spring Modules, https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch01s02.html abgerufen am 06.04.2021

[29] JavaTPoint: Spring Boot Architecture, https://www.javatpoint.com/spring-boot-architecture abgerufen am 06.04.2021

[30] Orts, Rogelio (2018): Layered Architecture & Spring Boot, https://medium.com/@RogelioOrts/layered-architecture-spring-boot-af7dc071d2b5 abgerufen am 06.04.2021

Erste Schritte

Die sehr grundlegenden REST-API Realisierungsbeispiele werden mittels der *IntelliJ IDEA* Entwicklungsumgebung umgesetzt, da sich diese sehr gut für die beiden Frameworks eignet.

Java EE Framework

1. Zu Beginn in *IntelliJ IDEA* ein neues Projekt erstellen und als Typ *"Java Enterprise"* auswählen. Danach bei *"Projekt Template"* den Punkt *"Rest service"* wählen und zum Beispiel *"GlassFish 5.0.0"* als *"Application server"* auswählen. Ein Application Server wie "Glassfish" muss jedoch zuvor installiert werden. [31]
2. Anschließend die gewählten Dependencies überprüfen und geeignete Projekt- und Packagenamen festlegen.

Nach der Erstellung des Projektes wurden von *IntelliJ IDEA* Demo Klassen angelegt und die Dependencies werden in der `pom.xml` Datei verwaltet:

```xml
<!-- File: pom.xml -->
<dependencies>
    <dependency>
        <groupId>javax.ws.rs</groupId>
        <artifactId>javax.ws.rs-api</artifactId>
        <version>2.1.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.7.0</version>
        <scope>test</scope>
    </dependency>
    <!-- ... -->
</dependencies>
```

Außerdem wird ein grundlegender API Controller in beispielsweise, je nach gewählten Namen, der Klasse `JavaEETest.java` angelegt: [32]

```
// File: JavaEETest.java
@Path("/hallo-java-ee") // < 1 >
public class JavaEETest {
    @GET // < 2 >
    @Produces("text/plain") // < 3 >
    public String hello() {
        return "Willkommen zu Java EE!"; // < 4 >
    }
}
```

1. Name der Basis Web Route nach der URL
2. Festlegung der Operation (*GET*, *POST*, *PUT*, *DELETE*)
3. Definierung des Rückgabeformats, welches in diesem Fall normaler Text ist
4. Rückgabe eines String, um die Funktion zu testen

Nachdem das Projekt gestartet wurde, ist beispielsweise, je nach URL Definierung, unter `http://localhost:8080/JavaEEDemo-1.0-SNAPSHOT/hello-world` die sehr grundlegende REST-API abrufbar:

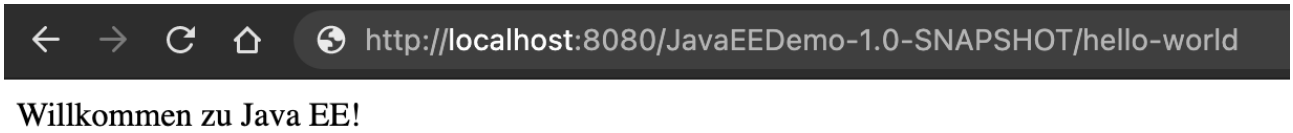


Figure 3. Java EE REST API GET-Response

Spring Framework

In diesem Beispiel wird sich auf eine Komponente von dem Spring Framework konzentriert: Spring Boot, in Kombination mit Java und dem Build Tool *Maven*.

Mithilfe des Spring Initializrs ist es möglich, ein fertiges Spring Boot Projekt Template generieren zu lassen. Dafür werden folgende Schritte benötigt:

1. Erstens die Initialisierungswebseite besuchen und wichtige Informationen wie das Build Tool, die Programmiersprache, die Versionen und geeignete Projekt-

und Packagenamen vergeben.

2. Anschließend mit Klick auf den Knopf *"ADD DEPENDENCIES..."* die gewünschten Abhängigkeiten wie *"Spring Web"*, *"Spring Data JPA"*, *"H2 Database"* und *"Thymeleaf"* mit erneutem Klick hinzufügen.
3. Letztlich auf den Knopf *"GENERATE"* klicken und ein ZIP-Ordner mit dem vorkonfiguriertem Projekt wird automatisch heruntergeladen und ist, sofern es in einer Entwicklungsumgebung wie *IntelliJ IDEA* geladen wird, einsatzbereit. Eine Konfiguration des Application Servers ist bei Spring Boot nicht notwendig. [33] [34]

Die Dependencies werden in der später generierten `pom.xml` Datei, in dem die Dependencies verwaltet werden, wie folgt angezeigt:

```xml
<!-- File: pom.xml -->
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
    </dependency>
    <!-- ... -->
</dependencies>
```

Anschließend das Projekt öffnen und beispielsweise einen Controller `BootTestController.java` anlegen:

```java
//File: BootTestController.java
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("api/v1/test") // < 1 >
public class BootTestController {

    @GetMapping("/hello") // < 2 >
    public String hello() {
        return "Willkommen zu Spring Boot!"; // < 3 >
    }
}
```

1. Name der Basis Web Route nach der URL
2. Name der spezifischen Route, welcher der Basis Route ergänzt wird, der Methode
3. Rückgabe eines einfachen Strings, um die Funktion zu testen

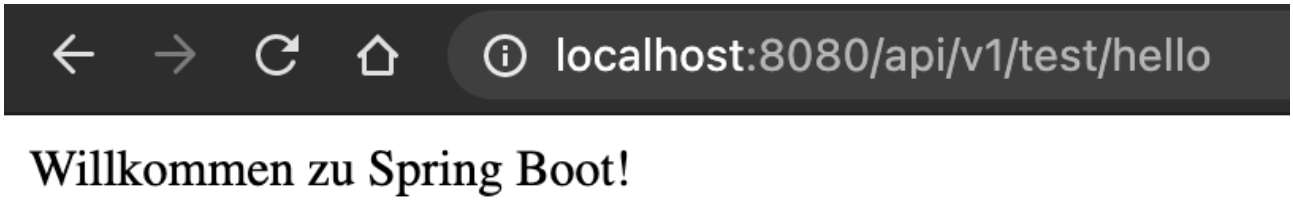
Darüber hinaus hat der *"Spring Initializr"* auch eine Main Klasse erzeugt, welche nach nötigem Gebrauch, mit zum Beispiel *@Bean Annotations* ergänzt werden kann.

```java
//File: DemoApplication.java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

Die Applikation kann nun entweder mit dem spezifischen Startknopf der gewählten Entwicklungsumgebung oder mit dem Befehl, je nach gewähltem Build Tool, `./mvnw spring-boot:run` mit *Maven* gestartet werden.

> Für den Gebrauch von *Maven* Kommandos muss *Maven* zuerst auf der *CLI* installiert werden oder kann in der *IntelliJ IDEA* mit Klick auf den Knopf *"Maven"* in der rechten Leiste benutzt werden.

Nun ist unter `http://localhost:8080/api/v1/test/hello` die sehr grundlegende REST-API abrufbar:



Willkommen zu Spring Boot!

Figure 4. Spring Boot REST API GET-Response

[31] Apache Netbeans: Getting Started with Java EE Applications, https://netbeans.apache.org/kb/docs/javaee/javaee-gettingstarted.html abgerufen am 06.04.2021

[32] Oracle Docs: Java EE Getting Started, https://docs.oracle.com/javaee/7/tutorial/cdi-basic001.htm abgerufen am 06.04.2021

[33] Spring: Building an Application with Spring Boot, https://spring.io/guides/gs/spring-boot/ abgerufen am 06.04.2021

[34] Spring: Spring Initializr, https://start.spring.io/ abgerufen am 06.04.2021

Bewertungskriterien

Regelmäßige Updates und Wartbarkeit

Java EE, seit Java EE 8 umbenannt auf Jakarta EE, erhält regelmäßig neue Versionsupdates. Die Frequentierung der Major Updates ist durchschnittlich alle zwei bis vier Jahre und enthält meist neue Features und Verbesserungen. Das letzte Hauptupdate (Jakarta EE 9) fand im Jahr 2020 statt. [35] Das Spring Framework wird kontinuierlich aktualisiert, dessen letztes großes Update war im Jahr 2017 mit der Version Spring 5.0 und als letztmaliger "Stable Release" gilt die Version *Spring 5.3.4* (Stand: 06.04.2021), welche im Februar 2021 veröffentlicht wurde.[36] Zwar erhalten beide Frameworks regelmäßig Updates, wohingegen das Spring Framework öfters Updates bekommt als Java EE. Dies ist wohl auch der stetig expandierenden Nutzerbasis von Spring und der Leitungsabgabe von Java EE durch Oracle geschuldet. Auch Dependencies werden seitens beider Frameworks im Laufe von Updates mit aktualisiert.

Durch Dependency Injection und Cloud Migration von den zwei Frameworks, bei Spring ist dies beispielsweise *Spring Boot*, ist eine gute Wartbarkeit und Aktualität gegeben. Spring Boot verfügt über *"Plain Old Java Objects (POJO)"*, welche sich durch kleine und "leichtgewichtige Klassen" auszeichnen.[37] Sie ermöglichen eine präzise Wartbarkeit, da jegliche Logik in kleinen Klassen leicht erreichbar und nicht zu umfassend verschachtelt platziert ist. Auch Java EE verfügt über Dependencies, welche einfach aktualisiert werden können. Meist reicht das bloße Ändern der Version und eine Aktualisierung der Abhängigkeiten. Bei der Wartbarkeit beider Frameworks kommt es vor allem darauf an, ob eine *Monolithen-* oder *Microservice Architektur* gewählt wurde. Letzteres bietet einen weit ausgehend mehr wartbaren Code, da der Code je nach Spezifikationen in verschiedene Module aufgesplittet ist. Hunderte Klassen in einem Package, unübersichtliche Klassennamen und hunderte Codezeilen in Klasse sind bei beiden Frameworks

hauptverantwortlich für schlechte Wartbarkeit.

Durch immer neue Updates und deren neuen Funktionen und Verbesserungen wird die Wartbarkeit immer besser, sei es mit der Cloud Migration von Java EE 8 oder Spring Boot, beide wurden auf den heutigen Stand der Technik gehoben und erreichen somit die volle Punktezahl von 5 Punkten.

Fazit:

	Java EE	**Spring**
Punkte	5	5

Table 2. Punkte Resümee "Regelmäßige Updates"

Langfristiger Support

Wie bereits im vorhergehenden Kapitel erwähnt, erhalten beide Frameworks stetig neue Versionsupdates. Jedoch wirkt seit 2017 Oracle, der damalige Leiter von der Java Enterprise Plattform, nicht mehr primär an der Entwicklung von Java EE mit, da sie die Leitung dafür aus mangelnder Interesse einer Weiterentwicklung abgegeben haben.[38] Dies zeigt auf, dass Java EE immer weniger Relevanz in der heutigen Software Gemeinschaft hat und einen langfristigen Support fragwürdig macht. Zwar wird mit einigen Updates in der Zukunft gerechnet, vor allem notwendige Sicherheitsupdates, doch bahnbrechende Weiterentwicklungsupdates werden wohl auf sich warten lassen.

Die Website *"JRebel"* hat beispielsweise einige Entwickler befragt, ob sie von Java EE zu Spring (vice versa) migriert hätten bzw. dies tun möchten. Die Befragung ergab, dass lediglich 14 Prozent von Spring zu Java EE und im Kontrast dazu 36 Prozent von Java EE zu Spring migriert haben bzw. den Wechsel durchführen wollen.[39] Deswegen erhält Java EE resümierendes für diese Kategorie 3 Punkte.

Viele Portale sprechen von dem *"Tod von Java EE"*, nachdem Oracle die Leitung

dafür abgegeben hat und allgemein bessere Frameworks auf den Markt gebracht worden sind.[40] [41] ***"Negotiations Failed: How Oracle killed Java EE"*** [42], so schreibt es beispielsweise der Autor *Markus Krag* in seinem Blog. In dem Bericht geht hervor, dass es einen Markenstreit zwischen Oracle und der Eclipse Foundation, der neuen Leitung von Jakarta EE, gab, welcher in keiner Einigung resultierte und Java EE dadurch einiges an Relevanz und Ansehen in der Software Gemeinschaft kostete.

Seitens des Spring Frameworks, vor allem bei den zwei Komponenten Spring Boot und Spring MVC, gibt es keine Anzeichen eines nahestehenden Endes des Supports. [43] Unter der Leitung der Apache Foundation gewinnt das Framework immer mehr und mehr an Interesse und Nutzung unter der Entwicklergemeinschaft. Nicht nur sind große skalierbare Projekte mit dem Framework möglich, es kommen auch regelmäßig stabile (Major) Versionen auf den Markt. Diesen Fakten geschuldet, erhält das Spring Framework in dieser Kategorie die volle Punktezahl von 5 Punkten und Java EE 3 Punkte.

Fazit:

	Java EE	**Spring**
Punkte	3	5

Table 3. Punkte Resümee "Langfristiger Support"

Kosten und Programmiereleganz

Seit der Übernahme durch die *Eclipse Foundation* ist Jakarta EE Open Source.[44] Oracle verfügt über die Markenrechte von "Java EE", weswegen die neue Leitung es auf "Jakarta EE" mit zusätzlich neuem Packagenamen umbenannt hat.[45] Dadurch ist Jakarta EE größtenteils kostenlos zu nutzen, jedoch gibt es neben den frei zugänglichen Java EE Servern wie "Tomcat" oder "Glassfish", auch kostenpflichtige Server.

Java EE bietet folgende Prinzipien:

- **Cloud und PaaS**: Cloud Migration (Web), durch Java EE 8, und PaaS (Platform as a service)
- **Aspect oriented programming (AOP)**
- **Java Programmierparadigmen**: Die Standardprogrammierparadigmen von Java
- **Design Paradigmen POJO**: Unterstützung von POJO (Plain Old Java Object)
- Java EE unterstützt die **Reactive Programmierung** (Paradigma): Daten werden mit statischen oder dynamischen Datenflüssen verarbeitet [46]

Jedoch verfügt es nicht über so viele Prinzipien wie Spring. Das Spring Framework verfolgt zudem neuartige Paradigmen, wonach Java EE dabei zurückliegt.

Das Spring Framework unterliegt der Apache-Lizenz, welche eine Free-Software-Lizenz ist. Es ist somit unentgeltlich und auch Open-Source.[47] Auch Spring hat sowohl kostenlose als auch kostenpflichtige Module und Server, welche aber grundsätzlich nicht notwendig sind. Das Framework verfügt unter anderem über folgende Prinzipien:

- **Lightweight**: Spring ist einfach aufgebaut und benötigt nicht viel Speicherplatz, beispielsweise ist die Basis Version nur ein Megabyte groß.
- **Inversion of control (IOC)**: Entwickler müssen Komponenten wie Bibliotheken nicht selbst erstellen/anlegen, sondern diese lediglich durch Dependency Injection in einer Konfigurationsdatei, zum Beispiel in der *POM* Datei, bestimmen. *Spring IOC* hat die Aufgabe, alle Dependencies lauffähig zu vereinen.
- **Aspect oriented programming (AOP)**: Spring unterstützt auch die aspektorientierte Programmierung. Wartbarkeit und Modularität wird durch die Trennung von logischen Aspekten und der Business Logik gewährleistet. *AOP* trennt diese zwei Komponenten, was bei der einfachen objektorientierten

Programmierung schwer(er) möglich ist.

- **Container**: Spring unterteilt Code in Container und handhabt Lebenszyklen und Anwendungskonfigurationen.
- Spring unterstützt auch die **Reactive Programmierung** mit der Dependency *"Reactor"*, vor allem in Kombination mit einer Microservice Architektur. [48]

Fazit:

	Java EE	**Spring**
Punkte	3	5

Table 4. Punkte Resümee "Kosten, Eleganz beim Programmieren"

Dokumentation

Java EE bietet eine Dokumentation von Oracle, welche bei Weitem nicht so umfangreich, leicht zugänglich und übersichtlich wie die von dem Spring Framework ist. Die Oracle Dokumentation ist verschachtelt durch einige Links erreichbar und listet alle nennenswerten Packages chronologisch auf, was an die Standard Java Dokumentation erinnert. Die Dokumentation hat sich durch Jakarta jedoch verbessert. [49]

Die Spring Framework Dokumentation bietet zwar auch die einzelnen Package Dokumentationen, aber übersichtlicher und leichter verständlich dargestellt.[50] Außerdem gibt es zahlreiche Guides und Anleitungen, wie eine bestimmte Sache von Spring genau funktioniert, wie zum Beispiel wie man die ersten Schritte mit dem Framework durchführt oder wie eine REST-API mit Spring Boot aufgesetzt wird. Auch allgemein bietet Spring mehr Anleitung als das Java EE Framework. Beide teilen jedoch einige Portale, so wie zum Beispiel die Webseite *"https://www.baeldung.com/[Baeldung]"*, welche sich auf Java und dem Spring Framework spezialisiert hat und hilfreiche Tutorials liefert. [51] [52]

Grundsätzlich bietet die Dokumentation beider Frameworks:

- **Klassen Beschreibungen und deren Nutzung**: Übersicht, Exceptions, Interfaces etc.
- **Genaue Package Beschreibungen**: Übersicht, Spezifikationen, Interfaces etc.
- **Genaue Methoden Dokumentation**: Beispiele, Exceptions, Konstruktoren, Parameter und Datentypen
- **Installations und Get Started Anleitungen**
- **Modulerklärungen**: Wie beispielsweise Anleitung für Integrationen, Web Servlets, Daten Transaktionen/Zugriffe etc.
- Ansicht von **veralteten** "deprecated" **Klassen, Packages und Methoden**
- **Allgemeine Dokumentation der Programmiersprache**: Java EE hat die Java-Dokumentation und Spring die Java-, Kotlin- oder Groovy-Dokumentation

Den Fakten geschuldet, dass Spring eine bessere Dokumentation im Hinblick auf Übersichtlichkeit, Inhalt und Erreichbarkeit liefert, erhält Spring die volle Punktezahl und Java EE, da allgemein wenig(er) zu Java EE bzw. Jakarta EE im Internet zu finden ist, nur 3 Punkte. Auch allgemein die Art der Aufbereitung der Dokumentation ist uneinsichtig bei Java EE, da es seit der Übernahme durch die Eclipse Foundation unterschiedliche Dokumentation gibt. Jedoch bieten beide eine fundierte und vertrauenswürdige Dokumentation und liefern die gewünschten Informationen.

Fazit:

	Java EE	**Spring**
Punkte	3	5

Table 5. Punkte Resümee "Dokumentation"

Funktionsumfang

Das Spring Framework bietet eine breite Palette an Komponenten, wie Spring Boot, Spring MVC, Spring Batch, Spring Data oder Spring Security: [53]

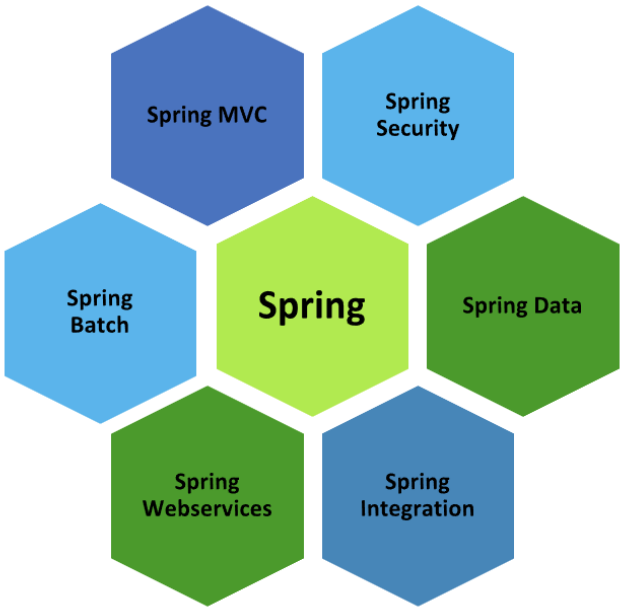


Figure 5. Spring Framework Komponenten

Java EE verfügt hingegen nicht über viele Komponenten, es ist eher einheitlich gestaltet. Allgemein teilen beide Frameworks ähnliche Features:

- **Dependency Injection**
- **Web Anwendungen**
- **Application Server**: Wobei bei Java EE die Konfiguration von diesem notwendig ist und bei Spring (Boot) "out of box" konfiguriert kommt.
- **Datenbanken Verfügbarkeit**: JPA etc. [54]

Nennenswerte Unterschiede: [55]

Faktor	Java EE	Spring (Boot)
Einarbeitung	Aufwendig und Webserver Konfiguration notwendig	Einfach, viele Features kommen "out of box"

Faktor	Java EE	Spring (Boot)
Programmiersprache	Java	Keine spezifische Sprache
UI	JSF2	Spring MVC
Testing	Arquillian (AppServer nötig)	Spring Testing (Mockito, …), JUnit
Transaktionen	JTA	JTA/Spring Data
AOP	Interceptor	Spring AOT
XML-lastig	Wenig	Viel
Geschwindigkeit	Schneller als Spring	Langsamer als Java EE

Table 6. Java EE vs Spring

Beide Frameworks bieten viele Features, Spring hingegen hat einige mehr. Dies liegt auch daran, dass es mehr Frameworks unterstützt und selbst beinhaltet. Java EE ist hingegen, laut Selbsttests, bei Applikationsstartzeiten um rund zehn Prozent schneller als Spring. Im Hinblick auf die Entwicklungen von Business Applikationen reichen die Features beider Framework grundlegend aus, Spring aber erleichtert durch mehr Funktionen und Frameworks die Implementierung der Applikationen, weswegen Spring insgesamt 4 Punkte, durch unter anderem die niedrigere Geschwindigkeit, und Java EE, durch den höheren Konfigurationsaufwand als Spring, 3 Punkte erhält.

	Java EE	Spring
Punkte	3	4

Table 7. Punkte Resümee "Funktionsumfang"

Querschnittsfunktion

Java EE sowie auch Spring unterstützen die Programmiersprache Java sowie Dependency Injection, weswegen sie einige Querschnittsfunktionen teilen:

- **Logging und Tracing**: Mit *log4j*, *Zipkin*, *Sleuth* und dem *ELK-Stack*
- **Caching**: Mechanismus, welcher es erlaubt, oft zu gegriffene Objekte und Informationen, temporär zwischenzuspeichern und so ein erneutes Laden zu unterbinden. Daten werden im Cache der Applikation gespeichert.
- **Security**: Sicherung von Daten und Zugriffskontrolle durch hohe Authentifizierungsstandards. Zudem gibt es *"Spring Security"*, was in einer Art auf beide Frameworks anwendbar ist.

Spring hat automatisierte Sicherheitsfunktionen in die Security Architektur implementiert, Java EE hingegen ist nicht so ausgebaut und hat keine speziellen Funktionen, wie "lightweight" Funktionen, *LDAP* (Lightweight Directory Access Protocol), *Web Form Authentifizierung* sowie *HTTP Authentisierung (Web Requests)*.

- **Health Endpunkt Metriken**: Vor allem in Kombination mit einer Microservice Architektur bieten *"Health Endpoint"* Metriken an, den Gesundheitsstand der Applikation, wie Uptime oder Latenz, zu überprüfen. Das Spring Framework bietet dafür zudem den **Actuator** an. [56]

	Java EE	**Spring**
Punkte	3	5

Table 8. Punkte Resümee "Querschnittsfunktion"

Developer Codebase und Community Support

Seitens beider Frameworks gibt es eine mittel bis große Developer-Base. Java EE bzw. Jakarta EE wird laut der Webseite stackshare.io, welche unterschiedliche Frameworks, Programmiersprachen etc. bewertet und aufzeigt, welche Technologien heutzutage verwendet werden, von **nur** 29 Unternehmen genutzt. Darunter "TripAdvisior", "Biting Bit" und "IWB". Außerdem hat Java EE auch Integrationen in "Eclipse", "NetBeans IDE" sowie "Apache Wicket". [57]

Auch Spring ist auf derselben Webseite vertreten. Angaben zufolge benutzen 501 Unternehmen Spring in ihrem Stack, darunter "Accenture", "Zalando" und auch "deleokorea". [58]

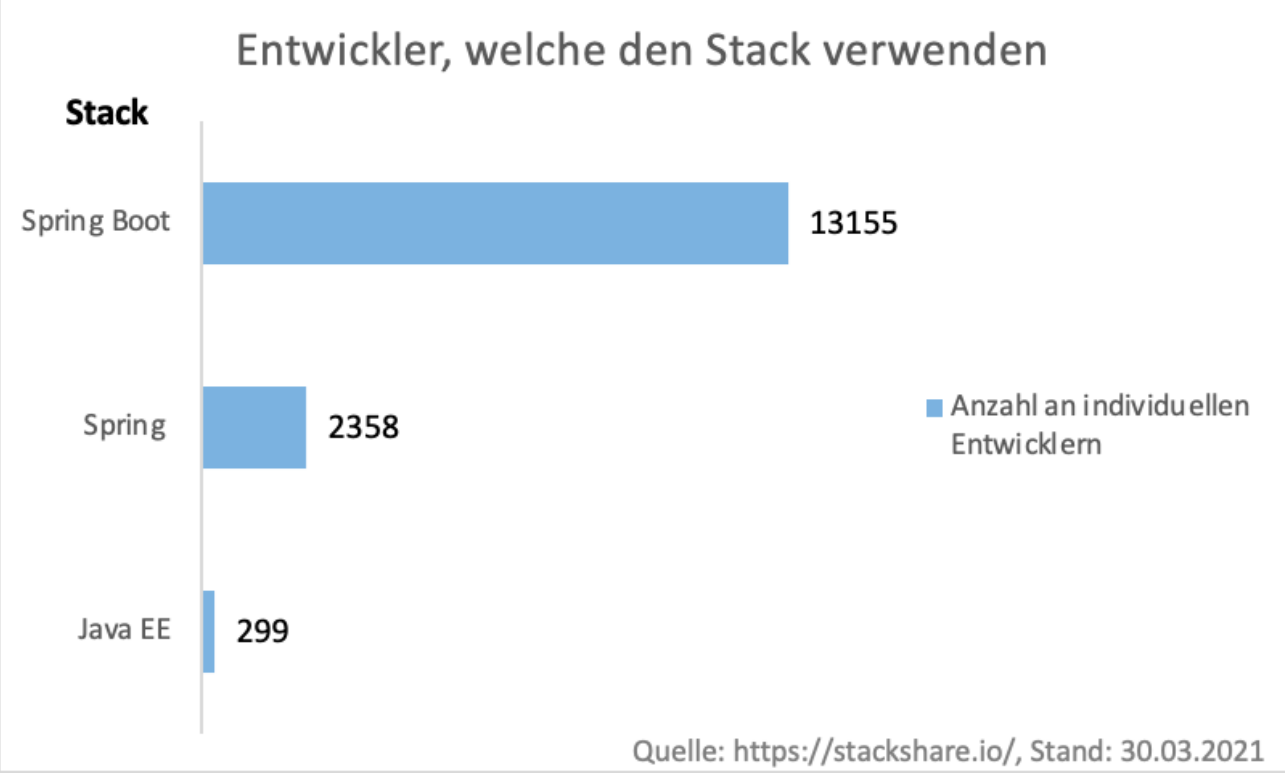


Figure 6. Entwickler, welche den Stack verwenden

Rund 13.155 Entwickler und Entwicklerinnen haben bekannt gegeben, dass sie die Spring Komponente Spring Boot in ihrem Stack benutzten, bei Spring sind es rund 2.358 und Java EE nur 299 Entwickler und Entwicklerinnen. [59]

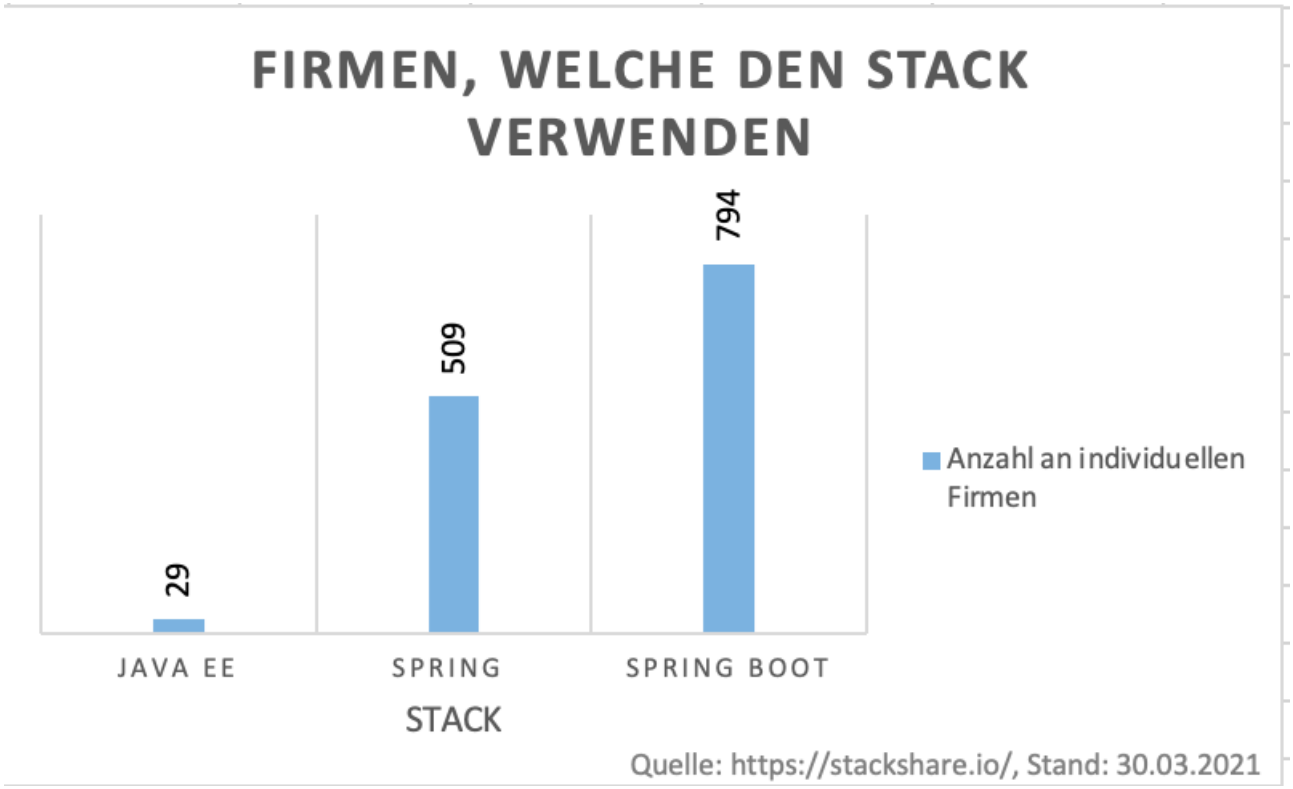


Figure 7. Firmen, welche den Stack verwenden

Auch zeigt der Trend, dass viele Firmen auf neue Stacks wie Spring und folglich Spring Boot setzen und nur mehr wenige Unternehmen Java EE als Stack angeben, welchen sie verwenden. Dies zeigt den heutigen Einsatz der zwei Frameworks ziemlich eindeutig, denn Spring (Boot) hat hierbei klar die Führung.

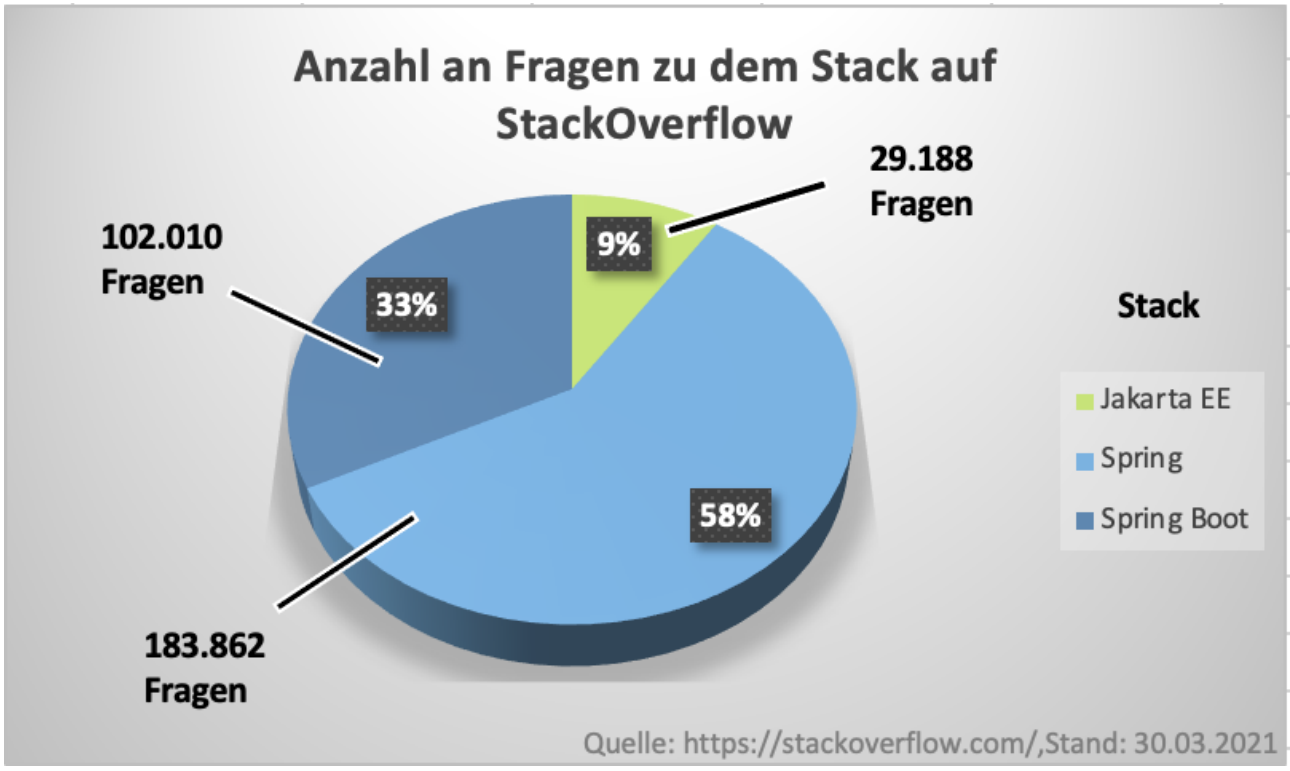


Figure 8. Anzahl an Fragen auf StackOverflow zu dem Stack

Am wohl bekanntesten Coding Portal, "StackOverflow", wo täglich tausende Coding spezifische Fragen gestellt werden, ist verzeichnet, dass es bei Spring insgesamt über 100.000 Fragen gibt, bei Spring Boot sogar mehr als 180.000 Fragen.[60] [61] Java EE bzw. Jakarta EE hat demnach nur mehr als 29.000 Fragen.[62] Dies zeigt, dass eine größere Community hinter Spring (Boot) steht und es diesbezüglich sehr viele Fragen bzw. Informationsquellen gibt, wohingegen Java EE nur etwa ein Drittel der Fragen von Spring Boot hat, somit weniger relevant ist und Entwickler weniger Fragen bzw. hilfreiche Informationen auf "StackOverflow" diesbezüglich zur Verfügung stehen.

Durch die wenige Benutzung (von Firmen) und Information auf "StackOverflow", erhält Java EE eine Punktezahl von 3 Punkten, wohingegen Spring (Boot) mit weitaus gehend mehr Entwickler und Entwicklerinnen, Firmen und Informationen die volle Punktezahl von 5 Punkten erhält.

	Java EE	**Spring**
Punkte	3	5

Table 9. Punkte Resümee "Developer Codebase und Community Größe"

[35] Wikipedia: Jakarta EE Versionshistorie, https://de.wikipedia.org/wiki/Jakarta_EE#cite_note-13 abgerufen am 06.04.2021

[36] Github: Spring Open Source Repository Versions, https://github.com/spring-projects/spring-framework/releases abgerufen am 06.04.2021

[37] Spring: Spring Integration, https://spring.io/projects/spring-integration abgerufen am 06.04.2021

[38] Delabassee, David (2017): Opening Up Java EE - An Update, https://blogs.oracle.com/theaquarium/opening-up-ee-update abgerufen am 06.04.2021

[39] JRebel: Java EE vs. Spring: Which is More Popular?, https://www.jrebel.com/blog/java-ee-vs-spring abgerufen am 06.04.2021

[40] Bien, Adam (2019): Java EE Is Dead - Completely Dead, https://adambien.blog/roller/abien/entry/java_ee_is_dead_completely abgerufen am 06.04.2021

[41] Voß, Tobias (2019): Java EE ist tot - es lebe Spring (Boot)!, https://blog.viadee.de/java-ee-ist-tot-es-lebe-spring abgerufen am 06.04.2021

[42] Krag, Markus (2019): Negotiations Failed: How Oracle killed Java EE, https://headcrashing.wordpress.com/2019/05/03/negotiations-failed-how-oracle-killed-java-ee abgerufen am 06.04.2021

[43] Github: Spring Framework Versions, https://github.com/spring-projects/spring-framework/wiki/Spring-Framework-Versions abgerufen am 07.04.2021

[44] Jakarta: Jakarta EE Open Source, https://jakarta.ee/ abgerufen am 06.04.2021

[45] Eclipse: Eclipse Transformer, https://projects.eclipse.org/proposals/eclipse-transformer abgerufen am 06.04.2021

[46] Stearns, John/Chinnici, Roberto/Sahoo (2006): Java EE Technical Details, https://www.oracle.com/technical-resources/articles/javaee/introduction-javaee5.html abgerufen am 06.04.2021

[47] Oracle Docs: Spring Framework Licence, https://docs.oracle.com/cd/E93130_01/oipa_license_guide/Content/OIPA%20License%20Guide/Third%20Party%20Licenses/Spring%20Framework/Spring%20Framework.htm abgerufen am 06.04.2021

[48] Spring: Spring Framework Features, https://spring.io/why-spring abgerufen am 06.04.2021

[49] Oracle Docs: Java EE Documentation, https://docs.oracle.com/javaee/7/index.html abgerufen am 06.04.2021

[50] Spring Docs: Spring Framework Documentation, https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/ abgerufen am 06.04.2021

[51] Baeldung: Java EE Guides, https://www.baeldung.com/?s=java+ee abgerufen am 06.04.2021

[52] Baeldung: Spring Guides, https://www.baeldung.com/?s=spring abgerufen am 06.04.2021

[53] Spring: Spring Framework Components, https://spring.io/projects/spring-framework abgerufen am 06.04.2021

[54] Oracle: Java EE at a Glance, https://www.oracle.com/java/technologies/java-ee-glance.html abgerufen am 06.04.2021

[55] Pressmar, Daniela (2018): Java EE vs Spring – Gemeinsamkeiten, Unterschiede und Entscheidungskriterien, https://blog.doubleslash.de/jee-vs-spring-gemeinsamkeiten-unterschiede-und-entscheidungskriterien/ abgerufen

am 06.04.2021

[56] Spring Docs: Spring Security Reference, https://docs.spring.io/spring-security/site/docs/4.2.15.RELEASE/reference/htmlsingle/ abgerufen am 06.04.2021

[57] Stackshare: Java EE, https://stackshare.io/java-ee abgerufen am 06.04.2021

[58] Stackshare: Spring, https://stackshare.io/spring abgerufen am 06.04.2021

[59] Stackshare: Spring Boot, https://stackshare.io/spring-boot abgerufen am 06.04.2021

[60] StackOverflow: Questions tagged "spring" , https://stackoverflow.com/questions/tagged/spring abgerufen am 06.04.2021

[61] StackOverflow: Questions tagged "spring-boot" , https://stackoverflow.com/questions/tagged/spring-boot abgerufen am 06.04.2021

[62] StackOverflow: Questions tagged "jakarta-ee" , https://stackoverflow.com/questions/tagged/jakarta-ee abgerufen am 06.04.2021

Entscheidungsresümee

Die einzelnen Bewertungen haben folgendes Ergebnis ergeben:

Kriterium	Java EE Framework	Spring Framework
Regelmäßige Updates	5	5
Langfristiger Support	3	5
Kosten, Eleganz beim Programmieren	3	5
Dokumentation	3	5
Funktionsumfang	3	4
Developer Codebase und Community Größe	3	5
Querschnittsfunktion	3	5
Ergebnis	**23**	**34**

Table 10. Entscheidungsresümee Java EE vs. Spring

Das Spring Framework gewinnt mit deutlichem Abstand die Auswertung, was nicht bedeutet, dass Java EE schlecht(er) ist, denn es kommt immer darauf an, welches Ziel die Applikation haben soll. Beide sind ähnlich aufgebaut, haben Dependency Injection, sind modular aufgebaut, "stable" und für Performance und hohe Verfügbarkeit ausgelegt. Doch folgende Punkte sind zu beachten:

- Java EE eignet sich für leichte, skalierbare und monolithische Anwendungen.
- Spring (Boot) ist für Anwendungen mit *GUI* im Frontend und Microservice Architekturen empfehlenswert.
- Beide Frameworks bieten Enterprise Support.
- Spring hat ein großes Ökosystem, was einen Wechsel von Spring zu anderen Frameworks erschwert.

- Spring hat längere Build/Start Zeiten als Java EE.
- Beide Frameworks sind für kleine aber auch große Projekte als Business Applikation in Unternehmen einsetzbar.
- Beide Frameworks sind im Markt ausreichend etabliert, haben Community Support und sind anerkante nützliche Frameworks.
- Java EE bzw. Jakarta EE ist nicht mehr so modern und innovativ, wie Spring es ist.
- Die Lernkurve ist bei beiden Frameworks mittelmäßig, bei Spring jedoch ein wenig steiler, da es mehr Module und Zusätze gibt.

Die beiden Frameworks werden oftmals als Konkurrenten gesehen, wo sie doch so ähnlich sind, da Spring auf Java EE aufbaut und somit eine Art Erweiterung dessen ist. Resümierend gesehen überwiegt jedoch das Spring Framework mit seinen Funktionen und Querschnittsfunktionen, dem Community Support, der Wartbarkeit, der Update Regelmäßigkeit sowie den Kosten und der Eleganz beim Programmieren. Es eignet sich somit besser als das Java EE Framework für die Entwicklung von Business Applikationen.

Verwendung von Spring Boot im Diplomprojekt

Im Diplomprojekt wurde als Framework auf die Verwendung von Spring, genauer *Spring Boot*, gesetzt.

Grund dafür war, dass bereits viel Erfahrung und praktische Programmierfähigkeiten in der Informatik Ausbildung gesammelt und im Spring Boot Framework erlernt wurden und so eine Programmierung mit dem Framework am leichtesten fiel. In Kombination mit der Programmiersprache *Kotlin* und dem Build Tool *Gradle* wurde eine REST-API Lösung, welche auf einen Server deployed wurde, für das Projekt umgesetzt. Das Hauptaugenmerk lag auf der Verwendung einer Microservice Architektur statt einer Monolithen-Architektur, um einzelne Module unabhängiger und einzeln startfähig zu machen sowie neues Know-How zu erlangen.

Ausschlaggebend war außerdem, die sehr gute Dokumentation des Frameworks, die herausstechenden Funktionen wie ein vorkonfigurierter eingebetteter Application Server, automatisierte Build Abläufe, die zahlreichen Frameworks und produktionsfähige Metriken wie Health Endpoints sowie allgemein die Arbeit, welche Spring dem Backend Team durch vorgefertigte Templates, Projekte und Module abgenommen hat. Spring Boot hat sich als ein sehr gutes Framework für die Entwicklung der REST-APIs herausgestellt und ein Server Deployment ist leicht gefallen.

Glossar

Build Tool

Automatisiert den Prozess der Erstellung ausführbarer Dateien. Die Software wird erstellt und beispielsweise werden nötige Dependencies heruntergeladen und verwaltet.

CLI

Ist das "Command Line Interface", welches es ermöglicht, Kommandos auszuführen.

ELK-Stack

Steht für Elasticsearch, Logstash und Kibana. Es ermöglicht das Tracing, die Verarbeitung und die visuelle Aufbereitung von zum Beispiel Metrik Daten, wie Uptime, einer Applikation.

Framework

Programmiergerüst, bei dem vorgefertigte Rahmen, wie Funktionen und Elemente, bereitgestellt und somit der Einstieg in die jeweilige Technologie erleichtert wird.

GUI

Ist eine grafische Benutzeroberfläche, auf der Daten visuell angezeigt werden.

JSP Pages

"Jakarta Server Pages" sind Seiten gebaut durch *"JHTML"*, welche es erlauben, Java Code in beispielsweise HTML- und XML-Dateien einzubetten.

Microservice Architektur

Anwendungen werden in kleine Module aufgeteilt und diese werden somit besser separat steuerbar und unabhängiger. Zusammen bilden alle Module die Anwendung.

Monolithen Architektur

Alle Software Komponenten befinden sich in einem großen Anwendungssystem, sie sind zentral, einzelne Softwareteile untrennbar und kaum unabhängig steuerbar.

REST-API

Ist eine Programmierschnittstelle, welche über HTTP-Anfragen mittels "CRUD"-Operationen agiert.

SOAP-API

Mit diesem Netzwerkprotokoll können Daten in Form von Envelopes zwischen Systemen ausgetauscht werden.

Quellen

Shankar, Ramya (2021): 10 Best Java Frameworks to Use in 2021, https://hackr.io/blog/java-frameworks abgerufen am 06.04.2021

Wikipedia: Jakarta EE, https://de.wikipedia.org/wiki/Jakarta_EE abgerufen am 06.04.2021

JavaTPoint: Java EE Specifications, https://www.javatpoint.com/java-ee abgerufen am 06.04.2021

O'Reilly: Java Platform: Enterprise Edition, https://www.oreilly.com/library/view/java-ee-6/9781449338329/ch01.html abgerufen am 06.04.2021

Oracle Docs: Distributed Multitiered Applications, https://docs.oracle.com/javaee/7/tutorial/overview003.htm abgerufen am 06.04.2021

Spring: Spring Framework, https://spring.io/projects/spring-framework abgerufen am 06.04.2021

Spring Docs: Language Support, https://docs.spring.io/spring-framework/docs/current/reference/html/languages.html abgerufen am 06.04.2021

Oracle Docs: Java EE language support, https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.htmlL abgerufen am 06.04.2021

Spring Docs: Spring Modules, https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch01s02.html abgerufen am 06.04.2021

JavaTPoint: Spring Boot Architecture, https://www.javatpoint.com/spring-boot-architecture abgerufen am 06.04.2021

Orts, Rogelio (2018): Layered Architecture & Spring Boot, https://medium.com/@RogelioOrts/layered-architecture-spring-boot-af7dc071d2b5 abgerufen am 06.04.2021

Apache Netbeans: Getting Started with Java EE Applications, https://netbeans.apache.org/kb/docs/javaee/javaee-gettingstarted.html abgerufen am 06.04.2021

Oracle Docs: Java EE Getting Started, https://docs.oracle.com/javaee/7/tutorial/cdi-basic001.htm abgerufen am 06.04.2021

Spring: Building an Application with Spring Boot, https://spring.io/guides/gs/spring-boot/ abgerufen am 06.04.2021

Spring: Spring Initializr, https://start.spring.io/ abgerufen am 06.04.2021

Wikipedia: Jakarta EE Versionshistorie, https://de.wikipedia.org/wiki/Jakarta_EE#cite_note-13 abgerufen am 06.04.2021

Github: Spring Open Source Repository Versions, https://github.com/spring-projects/spring-framework/releases abgerufen am 06.04.2021

Spring: Spring Integration, https://spring.io/projects/spring-integration abgerufen am 06.04.2021

Delabassee, David (2017): Opening Up Java EE - An Update, https://blogs.oracle.com/theaquarium/opening-up-ee-update abgerufen am 06.04.2021

JRebel: Java EE vs. Spring: Which is More Popular?, https://www.jrebel.com/blog/java-ee-vs-spring abgerufen am 06.04.2021

Bien, Adam (2019): Java EE Is Dead - Completely Dead, https://adambien.blog/roller/abien/entry/java_ee_is_dead_completely abgerufen am 06.04.2021

Voß, Tobias (2019): Java EE ist tot - es lebe Spring (Boot)!, https://blog.viadee.de/java-ee-ist-tot-es-lebe-spring abgerufen am 06.04.2021

Krag, Markus (2019): Negotiations Failed: How Oracle killed Java EE, https://headcrashing.wordpress.com/2019/05/03/negotiations-failed-how-oracle-

killed-java-ee abgerufen am 06.04.2021

Github: Spring Framework Versions, https://github.com/spring-projects/spring-framework/wiki/Spring-Framework-Versions abgerufen am 07.04.2021

Jakarta: Jakarta EE Open Source, https://jakarta.ee/ abgerufen am 06.04.2021 Eclipse: Eclipse Transformer, https://projects.eclipse.org/proposals/eclipse-transformer abgerufen am 06.04.2021

Stearns, John/Chinnici, Roberto/Sahoo (2006): Java EE Technical Details, https://www.oracle.com/technical-resources/articles/javaee/introduction-javaee5.html abgerufen am 06.04.2021

Oracle Docs: Spring Framework Licence, https://docs.oracle.com/cd/E93130_01/oipa_license_guide/Content/OIPA%20License%20Guide/Third%20Party%20Licenses/Spring%20Framework/Spring%20Framework.htm abgerufen am 06.04.2021

Spring: Spring Framework Features, https://spring.io/why-spring abgerufen am 06.04.2021

Oracle Docs: Java EE Documentation, https://docs.oracle.com/javaee/7/index.html abgerufen am 06.04.2021

Spring Docs: Spring Framework Documentation, https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/ abgerufen am 06.04.2021

Baeldung: Java EE Guides, https://www.baeldung.com/?s=java+ee abgerufen am 06.04.2021

Baeldung: Spring Guides, https://www.baeldung.com/?s=spring abgerufen am 06.04.2021

Spring: Spring Framework Components, https://spring.io/projects/spring-framework abgerufen am 06.04.2021

Oracle: Java EE at a Glance, https://www.oracle.com/java/technologies/java-ee-glance.html abgerufen am 06.04.2021

Pressmar, Daniela (2018): Java EE vs Spring – Gemeinsamkeiten, Unterschiede und Entscheidungskriterien, https://blog.doubleslash.de/jee-vs-spring-gemeinsamkeiten-unterschiede-und-entscheidungskriterien/ abgerufen am 06.04.2021

Spring Docs: Spring Security Reference, https://docs.spring.io/spring-security/site/docs/4.2.15.RELEASE/reference/htmlsingle/ abgerufen am 06.04.2021

Stackshare: Java EE, https://stackshare.io/java-ee abgerufen am 06.04.2021

Stackshare: Spring, https://stackshare.io/spring abgerufen am 06.04.2021

Stackshare: Spring Boot, https://stackshare.io/spring-boot abgerufen am 06.04.2021

StackOverflow: Questions tagged "spring" , https://stackoverflow.com/questions/tagged/spring abgerufen am 06.04.2021

StackOverflow: Questions tagged "spring-boot" , https://stackoverflow.com/questions/tagged/spring-boot abgerufen am 06.04.2021

StackOverflow: Questions tagged "jakarta-ee" , https://stackoverflow.com/questions/tagged/jakarta-ee abgerufen am 06.04.2021

Recherche von best practices von UX Design Guidelines für mobile Apps

Dennis Sima <sim18354@spengergasse.at>

Abstract

Diese Arbeit bietet die Grundlage für User Interface- und User Experience-Design Guidelines für mobile Applikationen. Es werden nicht die Basics des App-Designs erklärt, sondern auch Themen, wie Accessibility, Internationalisierung etc., welche für eine moderne Smartphone-App oftmals entscheident sein können, um erfolgreich zu werden.

Es ist dennoch zu beachten, dass es sich um keine vollständige Richtlinien handelt, da diese Schriftfassung als kompakte Recherche zum Thema UI und UX dienen soll in einem für den Leser einfachen dünnen Format.

Warum ist UI und UX wichtig?

Was ist UI & UX?

Damit der Enduser mit einer Software kommunizieren kann (z.B. Daten bearbeiten, speichern etc.) braucht dieser eine Schnittstelle, um dies einfach zu tätigen. In der modernen Softwareentwicklung ist auch oft die Rede des Frontends, wobei dies als obersten Sammelbegriff für die gesamte User-Applikation verwendet wird. Da der User keine Hintergrundprozesse des Backends im Normalfall mitbekommt, ist das Userinterface, alles, was der User sieht und bedienen darf. Es ist auch beim UI folglich das Screendesign die Rede, da es nur um das Design der Applikation geht.

Um das User Interface von der User Experience zu unterscheiden, kann man logisch differenzieren, dass das UI, jene Komponenten zusammenfasst, was der User sieht und die UX wie der User die Applikation bedient. So gesehen stehen UI und UX komplementär zueinander und sind in der modernen Frontend-Entwicklung nicht mehr wegzudenken.

Darüber hinaus ist das UX-Design so gesehen essenzieller als das Screendesign, da sich im Regelfall aus der User Experience das Design der App ergibt und nicht umgekehrt.

> *"Vergleicht man nur ein Handy aus den 1990er Jahren mit einem Smartphone von heute wird deutlich, wie komplex Produkte zum Teil geworden sind. Genau dann wird die User Experience essenziell wichtig für die Produktentwicklung."*
>
> — Prenger C. (2019 14. August): matter-of-design, https://matter-of-design.com/darum-ist-user-experience-so-wichtig aufgerufen am 06.04.2021

User Interface	User Experience
• Visuelles Aussehen der Applikation • Fokussiert sich auf das Produkt • Elemente die der User sieht und bedienen darf	• Identifizierung der Bedürfnisse eines Benutzers • Kümmert sich um dem Zweck und die Funktionalität • Fokussiert sich auf die Probleme der User

Table 11. Table UI VS. UX

Usability Vs. User Experience

Usability ist vom englischen übersetzt die Benutzerfreundlichkeit. Es beschreibt wie sehr eine App effizient und gebrauchstauglich ist.

> Als Enduser fällt schlechte Usability mehr auf als Gute.

Der Unterschied zwischen UX und Usability ist jedoch, dass bei der Usability nur der Fokus auf die effiziente Nutzergestaltung gelegt wird. Wohingegen User Experience das komplette Nutzererlebnis samt Emotionen, Gefühle und Freude beschreibt.

Besonderes Augenmerk sollte bei Apps beispielsweise auf die Menüleiste für die Navigation gerichtet sein, da dieses eventuell zentrale Element ist. Zudem muss stets eine gute Lesbarkeit der Inhalte gegeben sein. In diesem Fall ist es wichtig, dass nicht nur genügend große Buchstaben verwendet werden, sondern auch, dass die ausgewählte Schriftart unabhängig von der verwendeten Plattform gut lesbar ist. In diesem Fall sind Buchstaben mit einfachem Design von Vorteil.

In Bezug auf die Benutzerfreundlichkeit geht es nicht nur um das optische Design der App. Es geht auch um die Inhalte, zum Beispiel, wenn es vor allem um eine intuitive Navigation geht. Denn neben der Menüplatzierung ist die Beschriftung der Themenfelder äußerst wichtig. Deshalb sollte großen Wert auf geläufige Begriffe

gelegt werden. So können sich User sehr in der App zurechtfinden und die Usability ist gewährleistet.[63]

UI & UX im Laufe der Geschichte:

Die historische Entwicklung von der sogenannten User Interface und der User Experience begann mit der Erfindung des HTML (Hypertext Markup Language), welcher der Informatiker Tim Berner Lee im Jahre 1990 im Laufe seines Projektes im europäischen Kernforschungszentrum CERN betrieb. Damals war es oberstes Ziel möglich einfach Daten und Informationen sowohl möglich effizient und visuell ansprechend über Computer zu versenden.[64] Über HTML konnte man strukturiert Daten und Informationen darstellen.

Hier sieht man ein einfaches HTML-Beispiel, wie früher das UI ausgesehen hat:

```
<!DOCTYPE html>
<html lang="de">
   <head>
      <meta charset="UTF-8">
      <meta content="Das sind alle Informationen über HTML"
name="description">
      <title>HTML lernen: Das Grundgerüst</title>
   </head>
   <body>
      <h1>Hauptüberschrift: <i>Meine erste HTML-Seite</i></h1>
      <p>Dies ist meine <strong>erste</strong> HTML-Seite!</p>
      <h2>Überschrift des zweiten Absatzes</h2>
      <p>Dies ist der zweite Absatz meiner <em>ersten HTML-Seite</em>
.</p>
      <p><b>Anmerkung</b>: Ein typisches Anwendungsbeispiel des &lt;
b&gt;-Tags.</p>
      <ol>
         <li>erster Punkt der geordneten Testliste</li>
         <li>zweiter Punkt der geordneten Testliste</li>
         <li>dritter Punkt der geordneten Testliste</li>
      </ol>
   </body>
</html>
```

Dieses HTMl erzeugt folgendes UI:

Figure 9. HTML-Frontend Beispiel

Seit der Erfindung des HTML wurde der optischen Gestaltung für die Enduser mehr Aufmerksamkeit geschenkt und wurde laufend weiterentwickelt. Sowohl die Ansprüche heutiger Apps und Websites haben sich angepasst, wodurch man heutzutage wesentlich mehr braucht als ein HTML-file um den User zufriedenzustellen mit allen Funktionen.

[63] vgl. https://www.seobility.net/de/wiki/Usability

[64] vgl. https://de.wikipedia.org/wiki/Tim_Berners-Lee

Wie entsteht User Experience?

Für User Experience gibt es keine einheitlichen Vorgaben oder konkrete Prozesse, da diese sich individuell an den spezifischen Endnutzer und der eigentlichen Zielgruppe für das Software-Produkt richtet und jene Kunden unterschiedliche „Needs“ und „Pains“ haben:

Den Nutzer verstehen

Der Enduser steht immer im Mittelpunkt, mit samt seiner Wünsche und Bedürfnisse. Je nach Budget und Zeitdruck kann man dies mittels Umfragen direkt am Markt oder durch Kundenbeobachtungen abschätzen.

Zentrales Element sind hierbei die Personas, welche realitätsgetreu die Kunden abbilden, welche schließlich die Software Applikation nutzen sollen.

Im Zuge des UX-Designprozesses werden ebenso auch erste Wireframes oder Klick-Dummys visualisiert.[65]

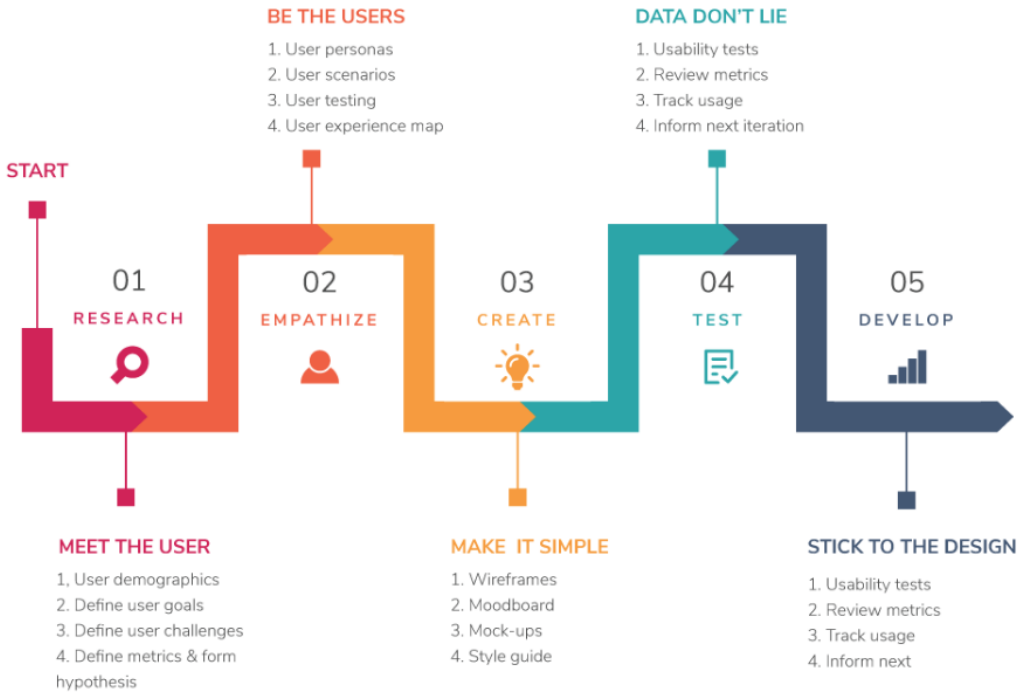


Figure 10. UX-Design Prozess, Quelle: https://medium.com/nyc-design/ux-ui-design-process-for-beginner-753952bb2241

Es gibt in der Theorie sehr viele Arten von gängigen UX-Prozessen, aber grundätzlich meinen alle das Selbe: ***Customer first!***

User Research

Üblicherweise beginnt der UX-Design Prozes mit einer sogenannten User Research. Hier versucht man zu verstehen, was die Kunden aktuell und in Zukunft auch für Features brauchen oder ebenso auch nicht wollen. Diese Recherche kann über mehrere Formen passieren. Zum einen über Marktumfragen oder Kundenbeobachtungen. Sei gesagt, dass Marktumfragen wiederrum sehr kostenintensiv sind, aber meist bessere und genauere Ergebnisse liefern.

Wichtig zu verstehen ist genauso, dass man nicht nur die zukünftigen Kunden analysiert, sondern auch die Projektanforderungen, sowie eine Wettbewerbsanalyse.[66]

Empathize

In dieser Phase muss man sich in den Kunden hineinversetzen. Gängige Methoden wären unteranderem mehrere Personas zu erstellen, welche die Zielgruppe darstellen sollen. Zusätzlich werden diese Personas, abhängig ihres persönlichen Background verschiedene Senarios durchleben, wie, wann, wo, warum, diese Persona mit der App interagiert.:[66]

Create

Mit den bisher gesammelten Daten wird im Anschluss erste Design-Entwürfe erstellt. Diese können in Form von Wireframes, Click-Dummies etc dargestellt werden.[67]

Um erste Entwürfe zu skizzieren, kann dies mit zahlreichen Online-Tools oder auf Papier erstellt werden.

Hier sieht man einen Beispiel-Wireframe der Smartphone-App:

Figure 11. Wireframe-Design von einer App

Test

Beim Testen ist essenziell zu beobachten, wie der Erstentwurf bei den potentiellen User ankommen wird:[66]

- Prototyp soll markttauglich genug sein, um genaues Marktfeedback zu bekommen
- Ziel ist zu Erkennen, ob der Design-Prototyp nach Plan funktioniert
- Feedback kritisch analysieren und einbauen

Develop

In der letzten Phase wird das fertige Produkt/ Prototyp entwickelt, Feedback der Kunden eingearbeitet und grunsätzliche Feinarbeiten an der App erledigt. Danach können eventuelle Beta-Tests angestrebt werden.

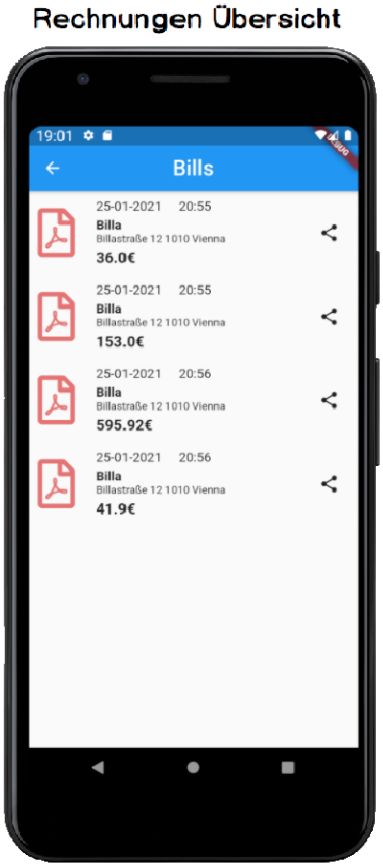


Figure 12. Wireframe-Design zum Prototypen

[65] vgl. https://academy.technikum-wien.at/ratgeber/was-ist-ux-design/

[66] vgl. https://www.uxmatters.com/mt/archives/2020/08/mobile-app-ux-design-process.php

[67] vgl. https://www.mobileappdaily.com/mobile-app-design-guidelines

UI-Patters für mobile Apps

Was ist das?

Besonders wichtig im Allgemeinen ist, nicht nur in der Software Entwicklung, so wenig wie möglich das Rad neu zu erfinden. Es gibt viele bereits fertige Software-Lösungen zum Einbinden etc. Genauso ist es auch beim Screen Design, Zeit und Arbeit zu sparen.

Gewisse UI-Patterns wurden nicht speziell nur für UI-Designer entwickelt und definiert. Sie sollen darüber hinaus auch der Software Architektur und den Programmierern zugutekommen.

Heutzutage sind UI-Patterns vollständig in die Software Entwicklung integriert und nicht mehr wegzudenken. Man findet diese in allen gängigen Apps und sogar Websites. Vor allem auch Endnutzer profitieren von solchen Patterns. Durch den Erkennungseffekt der bedienbaren Elementen wissen Nutzer in wenigen Sekunden wie diese jene Komponenten handhaben.[68]

Wichtig ist bei UI-Patterns, dass nicht alle Lösungsentwürfe für alle App-Lösungen geeignet sind. Diese müssen immer stets an den individuellen Use Case angepasst werden.

> *„Most of us wear t-shirts. My t-shirt and yours may vary in size and fit, but both are recognizable as t-shirts. We can add a nifty little pocket, details on the short sleeves, and print all kinds of stuff on them. However, the structure is pretty much the same regardless of how much we tailor the garment to our liking.“*
>
> — MARIA DE LA RIVA. (2020 12. August): Careerfoundry, https://careerfoundry.com/en/blog/ui-design/user-interface-patterns/#1-what-are-ui-design-patterns aufgerufen am 06.04.2021

Vorteile von UI-Patterns

Nicht zum Verwechseln sind UI-Patterns allerdings mit einem Baukasten, mit welchen man Features zusammenbauen kann und die Arbeit damit getan ist. UI-Patterns sind für UI-Designers wie Baupläne, mit denen man sich bei der Frontendgestaltung richten kann.

Zusätzlich sind UI-Patterns eine gemeinsame Sprache für die Kommunikation für UI-Designer. Dadurch werden vor allem auch Missverständnisse reduziert und sorgen für Kontinuität bei der Zusammenarbeit mit mehreren Designern am selben Projekt.

Bei den Endusern der Apps werden Design-Patterns auch zum Vorteil. Da sich viele Apps von der Gestaltung nicht allzu unterscheiden, haben Nutzer bereits eine gewisse Vorstellung auf welcher App-Page welche Elemente zum Vorschein kommen.

Beispielweise erwarten sich User als Mindestanforderungen von einer Login-Seite einer App Zwei Input-Felder (Für E-Mail und Passwort) und einen „Bestätigen“-Button zu einloggen in die App. Andere Elemente wären unschlüssig und führen zu Verwirrungen der Nutzer.[68]

Hier kann man als exemplarisches Beispiel eine Login-Maske einer Smartphone-App sehen, welche diverse Patterns nachgeht und erfüllt:

(Dadurch weiß der Nutzer ohne Überlegen, was die Intention dieser Page ist)

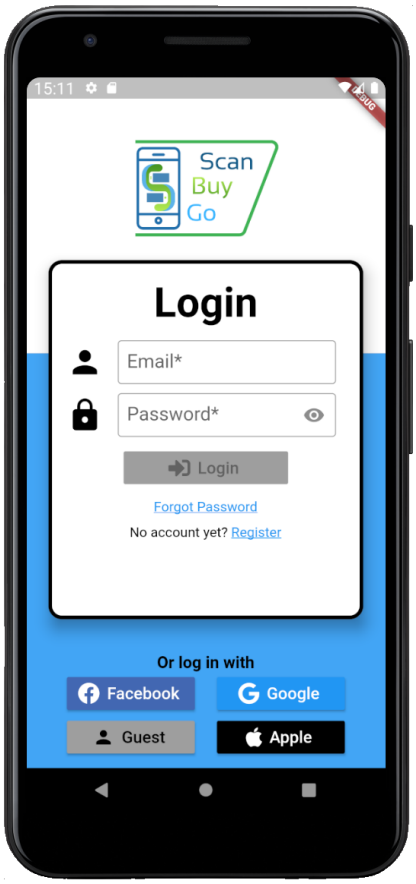


Figure 13. Design Patterns einer Login-Maske von einer App

Beliebteste Patterns in der Smartphone-Welt

Pauschal kann man nie definieren welche Patterns in eine App gehören. Jedoch gibt es aufffallend viele Apps die zumindest folgende aufweisen:[68]

1. **Teilen**: Ermöglicht, meist als Button, dem User seinen Inhalt auf Social-Media-Plattformen zu teilen
2. **Navigation**: Ermöglicht, meist als Pfeil nach links, dem User zur vorigen Page zu gelangen oder zurück zur Startseite
3. **Input/output:** Ermöglicht dem User Information zu schreiben und abschicken

und darauffolgend Feedback zu seiner Aktion zu bekommen

4. **Content Struktur:** In welcher Art werden Elemente dem User präsentiert? Wirkt es für dem User als wäre der Inhalt organisiert, intuitiv und zugänglich?

Dark UI-Design Patterns

Direkt vergleichbar mit den „normalen“ UI-Design Patterns gibt es zusätzlich die sogenannten „dark UI-Design Pattern“. Wie der Name bereits ausdrückt stecken für den Enduser meist nicht ganz moralische Design Entwürfe zur Verfügung. Man versucht den Usern in der App (gibt es in allen Frontend-Software-Lösungen) zu einer speziellen Aktion zu drängen, der dieser eventuell ohne dieses Design-Pattern nicht machen würde. Natürlich versuchen sowohl UX- und UI-Designer geteilt mit der Psychologie der zukünftigen Kunden zu spielen, um diese zu überlisten.

Trotzdem sind Dark Patterns heutzutage in der Frontend-Gestaltung sehr üblich und auf vielen Apps und auch Websites zu finden. Trotzdem sind sie allgemein unbeliebt und gefährden bei Missbrauch die Beziehung eines Unternehmens zu seinen Kunden.

Beliebteste „Dark Patterns“

Auffallend ist, dass Dark Patterns überwiegend von Shopping-Applikationen und bei der Anmeldung der User angewended wird:[69]

Verwirrende Fragen & Sprache:

Hier versucht man den User oftmals durch mehrere Verneinungen zu verwirren und mehrere vermeidlich ähnliche Fragen zu täuschen.

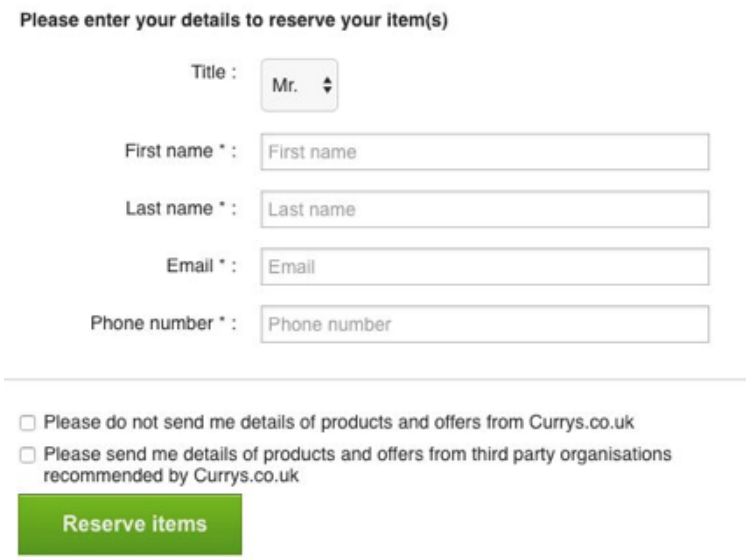


Figure 14. Beispiel: Verwirrende Sprache und Fragen bei der Registrierung, Quelle: https://www.darkpatterns.org/types-of-dark-pattern/trick-questions

Warenkorb-"Schleicherei"

Auf manchen Verkaufs-Apps kann es sein, dass man mit einem automatisch aktivierten Optionsfeld ein zusätzliches Produkt hinzugefügt bekommt. Wie beispielsweise auf godaddy.com

Premium Mitgliedschaft

Das Design und das Nutzererlebnis machen es möglichst einfach etwas zu kaufen, aber im Nachhinein sehr schwer es wieder abzubestellen. Dadurch muss man oft Umwege durch die App nehmen.

Ablenkung

Das Design wird entsprechend gestaltet, dass der User unterbewusst seine Aufmerksamkeit auf etwas lenkt und etwas anderes wichtiges vernachlässigt oder sogar übersieht. Wenn beispielsweise etwas groß funkelt, angezeigt wird, schaut der Mensch automatisch.

Versteckte Kosten

Dem User werden erst kurz vor dem Check-out Prozess alle Kosten angezeigt und nicht vorher.

Schuld einreden

Um den User ein schlechteres Gefühl zu bescheren, wenn dieser zum Beispiel ein Werbeangebot ablehnen will, wird diesen mit gezielter Sprache eingeredet, dass er Schuld hat, dieses Angebot nicht anzunehmen.

Zahlreiche Beispiele sind unter https://confirmshaming.tumblr.com zu finden.

Wie sollte man UI-Design Patterns anwenden?

Zunächst ist wichtig zu verstehen, dass man nicht einfach mit einzelnem UI-Design Patterns eine grafische Oberfläche erstellen kann oder eher sollte. Zuerst muss man diese noch speziell für den eigenen Use Case anpassen.

Beschreibe das Problem, welches aktuell bearbeitet wird. Dann wäre ein Blick in eine UI-Design Pattern Library empfehlenswert, wie anderen Designers ähnliche Lösungen entwickelt haben. Danach stellen sich die Frage, was man daraus lernen kann in Bezug zu dem eigenen Problem und erstellt im Anschluss seine eigene Lösung.[68]

Beliebteste UI-Patterns Library

In dem gesamten World Wide Web gibt sehr viele UI-Design Libraries. Zu Beginn sollte man sich nicht nur auf eine fokussieren und in der weiteren Folge wird man seine Lieblings-Library finden.

Bekanntere UI-Design Libaries sind folgende:

1. https://mobbin.design/patterns
2. https://pttrns.com/
3. https://uigarage.net
4. https://material.io/
5. https://www.mobile-patterns.com/

> Für Cross Plattform App oder Progressive Web-Apps gibt es wenig bis keine Unterschiede zum Design für iOS oder Android. Bei einer Native-App sieht dies ein wenig anders aus. Man sollte hierbei achten, ob das jeweilige Design des Patterns auf dieser App-Plattform passt.

Apple Human Interface Guidelines

Apples eigener App Store legt im Gegensatz zu dem von Mitbewerber Google angebotenen Google Play Store sehr großen Wert auf Standards.

Zusammengefasst, sind diese in gesamt 5 Punkten unterteilt: [70]

1. Safty
2. Performance
3. Business
4. Design
5. Legal

Sollte eine App nicht die Mindeststandards der UI/UX-Guidelines entsprechen wird sie auf dem Weg in den App Store abgelehnt. Deswegen ist es absolut essenziell für jeden UI/UX-Designer sich mit Apples Vorgaben auseinandergesetzt zu haben.

Zu finden sind diese Richtlinien von Apple unter: https://developer.apple.com/design/human-interface-guidelines/

Android Material Design Guidelines

Ebenfalls wie Apple besitzt auch Google für ihre Plattform Empfehlungen und Hilfeleistungen für Entwickler und Designer. Im Gegensatz zu Apple sind aufgrund der optionalen Gegebenheiten von Android besitzen viele Apps im Google Play Store keine bis wenig Standards.

Zu finden sind diese Empfehlungen/Hilfestellungen von Google unter: https://material.io

Material.io enthält sowohl Elemente und Komponente für iOS, Android, Web und dem Frontend-Framework Flutter

[68] vgl. https://careerfoundry.com/en/blog/ui-design/user-interface-patterns/#1-what-are-ui-design-patterns

[69] vgl. https://www.darkpatterns.org/types-of-dark-pattern

[70] vgl. https://developer.apple.com/app-store/review/guidelines/

Weiteres Design

Für eine gelungene App spielen hierbei noch mehrere Faktoren eine Rolle. Es ist vielmehr ein Zusammenspiel von zusätzlich, einerseits der **Farbwahl**, welche man keineswegs von sowohl ihren Nutzen und Effektivität nicht unterschätzen darf und andererseits der **Typografie**.

Farbwahl

Eine grundlegende Entscheidung für jede mobile App wird jene sein eine geeignete Farbe zu suchen. Deswegen ist es von großer Bedeutung eine Farbpalette mit Bedacht auszuwählen. Zusätzlich spielen Farben ebenso auch in der Psychologie des Menschen eine Rolle, mit Emotionen und Gefühlen, welche unterbewusst ausgelöst werden.

Deswegen verhilft die sogenannte Color Theory diesen Prozess der Farbwahl zu beschleunigen. Dieser umfasst zudem das Color Wheel.[71]

Das Color Wheel

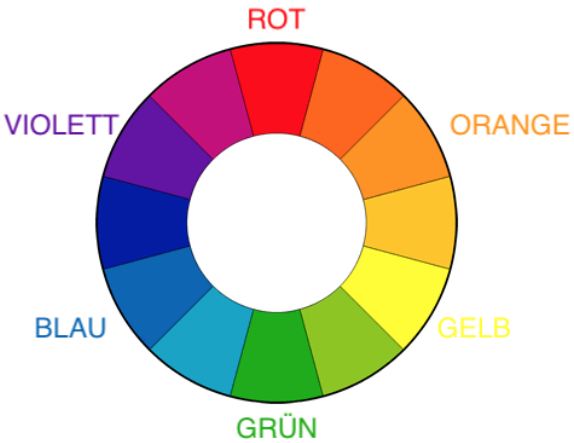


Figure 15. Das Color Wheel, Quelle: https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Primary_colors

Der Aufgabenbereich des Color wheel, auch Farbkreis genannt, besteht darin Beziehungen zwischen Primär-, Sekundär,- und Tertitärfarbe aufzuzeigen. Einen

ersten Entwurf lieferte im Jahre 1666 Sir Isaac Newton.

Dabei umfassen diese Primärfarben Blau, Rot und Gelb. Aus diesen Grundfarben ergeben sich wiederum die Sekundärfarben Grün, Orange und Violett. Die zusätzlichen Tertiärfarben sind im Anschluss grün-gelb, gelb-orange, orange-rot, rot-violett, violett-blau und blau-grün.[71]

Warme und kühle Farben

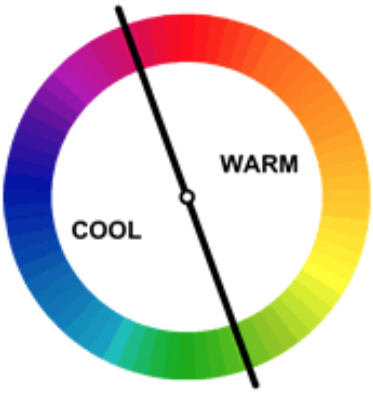


Figure 16. Warme oder kalte Farben im Color Wheel, Quelle: https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Primary_colors

Grundsätzlich kann man Farben in zwei große Überkategorien fassen. Da die Augen warme oder kaltes Licht speziell war nehmen, gilt dies ebenso bei Farben. Es werden beispielsweise die Farben von Rot über Orange bis Gelb als warme Farbe bezeichnet, da der Mensch diese mit warmen Gegebenheiten assoziiert, wie das Feuer (Rot-Orange) und die Sonne (Gelb). Dem Gegenübergestellt sind die kühlenden Farben. Mit Blau wird beispielweise kühles frisches Wasser verbunden. [71]

Farbharmonie

Um passende Farben für das User Interface auszuwählen ist es wichtig eine gewisse Harmonie zwischen den Farben aufzubauen.

Hierfür gibt es mehrere Möglichkeiten Farben zu bestimmen: [71]

Monochromatische Farbvariante

Die monochromatische Farbvariante besteht aus mehreren Farbtönen aus derselben Primärfarbe. Diese entstehen, wenn man diese Farbe heller oder dunkler aussehen lässt.[71]

Komplementäre Farbvariante

Figure 17. Eine komplementäre Farbmischung, Quelle: https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Primary_colors

Komplementärfarben sind in den meisten Fällen schwierig anwendbar, funktionieren dennoch als Farben, damit ein Element auffällig aussieht.[71]

Komplementärfarben sind schlecht für Text geeignet.

Analoge Farbvariante

Figure 18. Eine analoge Farbmischung, Quelle: https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Primary_colors

Da diese Farben sehr nah nebeneinander liegen, fungieren diese einheitlich. Diese Farbkombination lässt sich vor allem in der Natur vorfinden und ist dem menschlichen Auge sehr angenehm. Üblicherweise verwendet man eine dominante

Farbe, eine Zweite zur Unterstützung und zusätzlich eine dritte als Akzentfarbe.[71]

Triadische Farbvariante

Figure 19. Eine triadische Farbmischung, Quelle: https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Primary_colors

Aufgrund dieser Farbkombination wird eine lebendigere Stimmung erzeugt, mit der zusätzlich, im Gegensatz zur analogen Variante, mehr Kontrast zum Vorschein kommt. Üblicherweise wird eine dominantere ausgewählt und die restlichen Farben nur zur Unterstützung verwendet.[71]

Welche Farben für welche App?

Für die Gestaltung des Frontends darf man die Wichtigkeit und Wirkung dieser niemals unterschätzen. Mit einer adäquaten Farbgestaltung des UI kann man dem User positiv emotional und unterbewusst beeinflussen. Beispielsweise strahlt die Farbe Blau Ruhe, Objektivität, Neutralität und Klarheit. Dieses verstärkt das Vertrauen der User und gibt diesen ein sicheres Gefühl als bei anderen Farben.

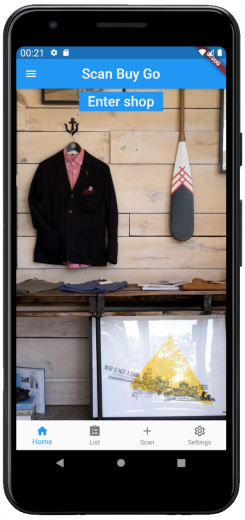


Figure 20. Blau wäre eine Option für eine vertrauenswürdige Payment-App

Corporate Identity

Die sogenannte Corporate Identity ist das Selbstbild jedes Unternehmens. Es umfasst alle Strategien, welche zu einer besseren, sowohl Firmeninternen und -externen, Ausstrahlung verhelfen.[72]

Eine Corporate Identity wird in weitere Untergebiete unterteilt:

- Corporate Design (CD)
- Corporate Behaviour (CB)
- Corporate Culture
- Corporate Communication (CC)
- Corporate Philosophy (CP)

Corporate Design in der App-Entwicklung

Das CD richtet sich spezifisch auf den Wiedererkennungseffekt eines Unternehmens, welches die entsprechenden Firmenwerte mit sich tragen soll.

Grundsätzlich ist davon auszugehen, dass die Farbwahl mit denen des Firmenlogos möglichst übereinstimmen sollte, wenn dieses in der App verwendet werden soll, da dadurch ein größerer Wiedererkennungseffekt von der App ausgestrahlt wird. In Summe umfasst das Corporate Design sämtliche Bereiche um die Farbauswahl, Logodesign und Schriftdesign.

Typografie

In der weiten Welt der App-Entwicklung werden ungefähr 95% des gesamten Contents über Text dargestellt. Das heißt, dass der User egal ob bewusst oder unterbewusst mit dem Text kommunizieren wird. Wenn ein Logo beispielsweise ein einen Schriftzug beherbergt, fällt dieser immer dem menschlichen Auge zuerst auf. Dies unterstreicht die Wichtigkeit von Typografie.[73]

Bei iOS und Android-Apps sind der Default-Schriftzug Roboto beziehungsweise Helvetica Neue.[74]

Lesbarkeit

Obwohl Smartphones mit der Zeit tendenziell größer werden, ist es auf den kleinen Bildschirm nicht immer leicht viel Text unterzubringen. Besonders für Apps die großen Mengen an Text dem User aufbereiten, ist die Lesbarkeit sehr von Bedeutung. Optimal wäre es vor allem auch im mobilen App Bereich lesbare Schrift, welche auch bei direkter Sonneneinstrahlung gut lesbar ist.

Professionalität:

Die Schriftart sollte stets mit dem Image der App und der eigentlichen Zielgruppe übereinstimmen. Wenn beispielsweise eine Banking-App keine seriöse Schrift wählt, verunsichert dies viele Kunden.[73]

Schriftgröße

Die Guidelines von Apple geben Aufschluss darüber, dass bei iOS-Apps die Schriftgröße nur zwischen 11 und 17p betragen darf. Für Android-Apps ist diese Richtlinie ebenfalls legitim. Zu beachten gilt weiter auch, dass die ungefähre Zeichenanzahl pro Linie im Hochformat bei 35bis 50 liegt. Dadurch entsteht die Herausforderung Sätze nicht kompliziert zu halten und dennoch genug Informationen an den User zu bringen.[73]

Kontrast

Ebenso wichtig ist auch der Spielraum zwischen Buchstaben und dem Hintergrund. Den Besten Kontrast liefert dementsprechend schwarz auch weiß. Allerdings soll hier auch Bezug genommen werden zum Logo, Corporate Design etc.[73]

[71] vgl. https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm

[72] vgl. https://www.ionos.at/startupguide/unternehmensfuehrung/corporate-identity/

[73] vgl. https://de.yeeply.com/blog/app-design-die-bedeutung-der-typografie/

[74] vgl. https://mobilbranche.de/2014/08/typografie-beim-app-design

Accessibility und Internationalisierung

Heutzutage bestehen Apps im Gegensatz zu früher, aus mehr als nur funktionalen Features. Mehr Design, mehr Experience, mehr Usability sind die Folge um User zufriedenzustellen und nicht mehr wegzudenken. Jedoch wird sehr schnell die Bedeutung von Accessability vergessen, also eine Hilfestellung für beeinträchtigte Menschen.

> *„Whilst many development teams see accessibility as a 'nice-to-have', it is crucial in today's mobile world."*
>
> — Caspar Geerlings (2019, 21. Oktober): How to create an accessible app (and why you should); Quelle: https://medium.com/oberonamsterdam/how-to-create-an-accessible-app-and-why-you-should-5493f41f8bdb

Warum sollte man auf Accessibility Bezug nehmen?

Ein großes Missverständnis liegt auf Accessibility. Viele UI- und UX-Designer, sowie Entwickler nehmen dies als optionales Feature war. Jedoch wird geschätzt, dass ungefähr 15%, also 1.000.000.000 Menschen davon betroffen sind.[75] Zusätzlich möchte der User eventuell auch die App während dem Autofahren oder beim Joggen verwenden. Das bedeutet, dass es auf jeden Fall sinnvoll und in der heutigen Welt essentiell geworden ist, seine App accessible zu designen und entwickeln.[76]

Auf was achten?

Um für möglichst viele User verfügbar zu sein, sollte man ebenso auf die verschiedenen Aspekte der Accessibility eingehen. Was sind die Bedürfnisse, die daraus resultieren? Was brauchen die User?[76]

Sehschwäche

Da man hauptsächlich mit den visuell mit dem Smartphone kommuniziert, ist dies ein großes Hinderniss für zahlreiche Menschen. Hinzu kommen mehrere Ausprägungen, wie Farbenblindheit und Blindheit über mehrere Stufen bis zum kompletten Sehverlust. Abhilfe schafft hier nicht explitzit auf Farben zu setzten und das UI ebenfalls auch anders zu gestalten, z.B. größeren Zoom der Texte auf 200% erlauben etc. Zusätzlich würde es der Accessibility gut tun einen Screen Reader oder ähnlich einzubinden.[76]

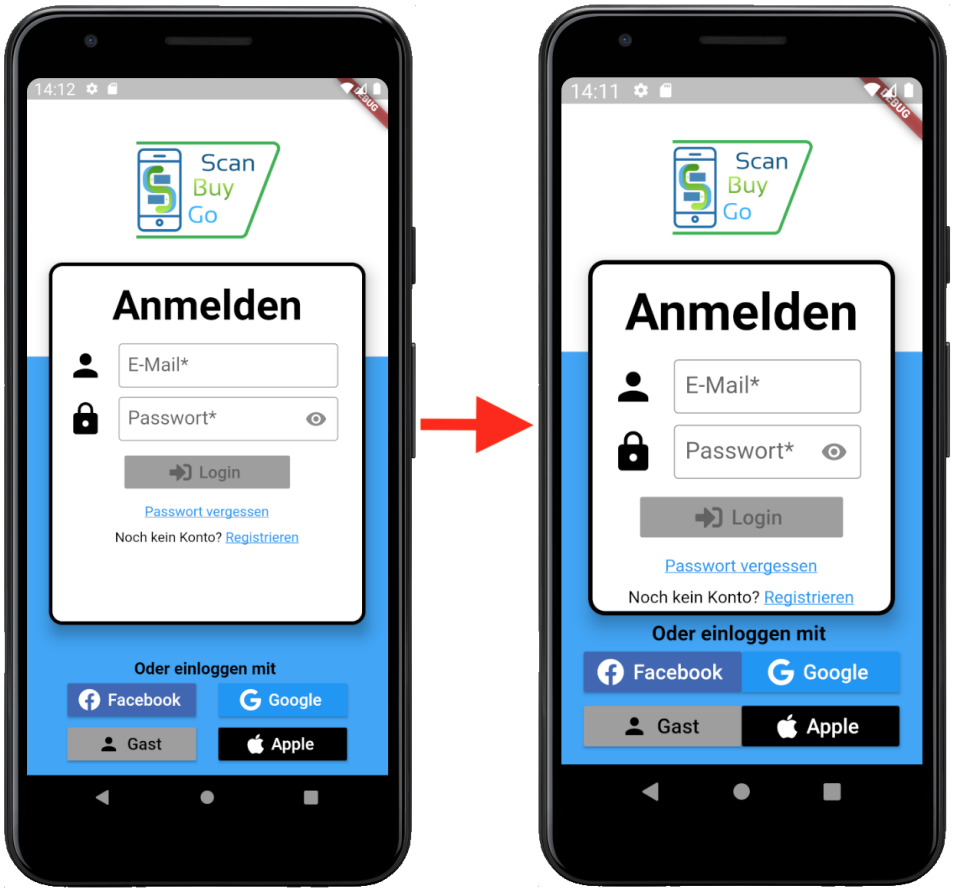


Figure 21. Beispiel für größere Schrift

Hörschwäche

Diese Beeinträchtigung ist normallerweise bei Apps kein allzu großes Problem. Bei Videos oder Sound-files könnten als Option zusätzlich mittels Transkription angezeigt werden.[76]

Lernschwäche

Unter diesem Übergriff werden sämtliche Konzentrations-, Lese- und Schreibschwächen gemeint. Diese kann beispielsweise mit kurzen Sätzen, mehr Icons oder über Spracheingabe statt dem Schreiben in der Smartphone-App optimieren.[76]

Was liefern Apple and Google?

Da sowohl die Betriebssysteme iOS und Android zahlreiche Accessibility-Features in ihr Ecosystem integriert haben, sind diese Funktionen ebenso für Apps verwendbar.

Gängige iOS-Features sind: [77]

- Untertitel und Audiobeschreibungen
- Voice-over
- Übersetzung
- Geführter Zugang
- Anpassung anzeigen

Gängige Android-Features sind: [77]

- Audio und Bildschirmtext
- Anzeigeoptionen
- Interaktionskontrollen
- Screenreader

Internationalisierung

Um global zu denken und eine mobile App erfolgreich über Vertriebskanäle wie gängige App-Stores zu vertreiben ist es äußerst ratsam ebenfalls diese in möglichst viele Sprachen zu übersetzen. In der IT-Welt ist auch häufig von i18n die Rede. Dies hat den folgenden Hintergrund, da genau 18 Buchstaben zwischen dem *"I"* und dem *"n"* von Internationalization sind.

Aus der Praxis empfiehlt sich zumindest eine App, als absolute Mindestanforderung in Englisch bereitzustellen. Bestenfalls sogar in die meistgesprochene Sprache der Welt. Diese sind beispielsweise: Englisch, Deutsch, Französisch, Spanisch, Portuguisisch, Italienisch, Japanisch und Russisch. Sei gesagt, dass dies vorwiegend von westlichen Ländern gesprochen wird, da in anderen bevölkerungsstarken Ländern, wie China und Indien mehrere Sprachen gesprochen werden.

Figure 22. Eine App in mehere Sprachen ist mittlerweile schon Pflicht

Lokalisierung

Lokalisierung auch l10n genannt ist die Anpassung von nicht textlichen Elementen wie Formatierung von Datum, Uhrzeit, Adressen, Icons und Währungen etc. Die bewahrheitet sich in der Entwicklung als sehr mühsam, aber erhöht und verbessert den Komfort und die User Experience der Nutzer.[78]

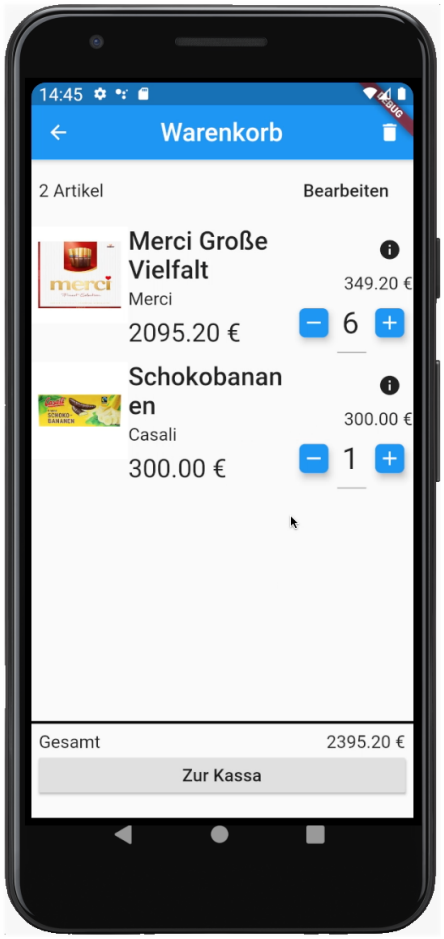


Figure 23. Beim Warenkorb sind besipielsweise das Datum und die Währung wichtig

[75] vgl. https://mobilbranche.de/2014/08/typografie-beim-app-design

[76] vgl. https://medium.com/oberonamsterdam/how-to-create-an-accessible-app-and-why-you-should-5493f41f8bdb

[77] vgl. https://plusqa.com/2020/03/26/the-best-accessibility-apps-in-2020/

[78] vgl. https://www.infopulse.com/de/blog/internationalisierung-der-mobilen-apps-wege-und-methoden-zur-steigerung-der-einnahmen-um-26/

Glossar

HTML

Ist die Grundstruktur, mit der Websites aufgebaut sind und Text dargestellt werden.

Wireframes

Ein visueller Design-Entwurf, wie Elemente in einer App dargestellt werden.

Click-Dummys

Ist eine reine Frontend-Applikation zum Testen, ohne Backend-Zugriff und komplexere Funktionen.

Cross-Plattform Apps

Ein Framework, welches dem Frontend ermöglicht auf iOS und Android lauffähig zu sein.

PWA

Progressive Web Apps sind Websites, welche zusätzlich als mobile Apps optimiert sind.

Quellen

Prenger C. (2019 14. August): matter-of-design, https://matter-of-design.com/darum-ist-user-experience-so-wichtig aufgerufen am 06.04.2021

Seobility: Usability, vgl. https://www.seobility.net/de/wiki/Usability aufgerufen am 07.04.2021

Wikipedia: Tim Berners-Lee, vgl. https://de.wikipedia.org/wiki/Tim_Berners-Lee aufgerufen am 07.04.2021

Technikum-wien: WAS IST UX-DESIGN?, vgl. https://academy.technikum-wien.at/ratgeber/was-ist-ux-design aufgerufen am 07.04.2021

Medium: UX/UI Design Process (Bild), https://medium.com/nyc-design/ux-ui-design-process-for-beginner-753952bb2241 aufgerufen am 07.04.2021

Uxmatters: Mobile App UX Design Process, vgl. https://www.uxmatters.com/mt/archives/2020/08/mobile-app-ux-design-process.php aufgerufen am 07.04.2021

obileappdaily: Mobile App Design Guidelines to Build an App Users Love, vgl. https://www.mobileappdaily.com/mobile-app-design-guidelines aufgerufen am 07.04.2021

Careerfoundry: What Are User Interface (UI) Design Patterns? Everything You Need To Know, vgl. https://careerfoundry.com/en/blog/ui-design/user-interface-patterns/#1-what-are-ui-design-patterns aufgerufen am 07.04.2021

Darkpatterns: TYPES OF DARK PATTERN, vgl. https://www.darkpatterns.org/types-of-dark-pattern aufgerufen am 07.04.2021

Apple Inc.: App Store Review Guidelines, vgl. https://developer.apple.com/app-store/review/guidelines aufgerufen am 07.04.2021

Tigercolor: Basic color schemes - Introduction to Color Theory, vgl.

https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm aufgerufen am 07.04.2021

Tigercolor: Basic color schemes - Introduction to Color Theory (Bilder), https://www.tigercolor.com/color-lab/color-theory/color-theory-intro.htm#Primary_colors aufgerufen am 07.04.2021

Ionos: Corporate Identity, vgl. https://www.ionos.at/startupguide/unternehmensfuehrung/corporate-identity aufgerufen am 07.04.2021

Yeeply: App Design: Die Bedeutung der Typografie, vgl. https://de.yeeply.com/blog/app-design-die-bedeutung-der-typografie aufgerufen am 07.04.2021

Mobilbranche: 3 typografische Fauxpas im App-Design., vgl. https://mobilbranche.de/2014/08/typografie-beim-app-design aufgerufen am 07.04.2021

Medium: How to create an accessible app (and why you should), vgl. https://medium.com/oberonamsterdam/how-to-create-an-accessible-app-and-why-you-should-5493f41f8bdb aufgerufen am 07.04.2021

Plusqa: The Best Accessibility Apps in 2020, vgl. https://plusqa.com/2020/03/26/the-best-accessibility-apps-in-2020 aufgerufen am 07.04.2021

Infopulse: Internationalisierung der mobilen Apps: Wege und Methoden zur Steigerung der Einnahmen um 26%, vgl. https://www.infopulse.com/de/blog/internationalisierung-der-mobilen-apps-wege-und-methoden-zur-steigerung-der-einnahmen-um-26 aufgerufen am 07.04.2021

Untersuchung und die Gegenüberstellung von Payment Providern und deren APIs

Lucas WALTER <wal18388@spengergasse.at>

Hintergrund

Im Rahmen der Entwicklung eines Prototypen einer Selfcheckout-App für den DACH-Raum (Deutschland, Österreich, Schweiz) soll ein Zahlungsanbieter, der aus dem Projekt entstandene Anforderungen erfüllt, ausgewählt werden.

Die wichtigste Besonderheit an diesem Projekt ist, dass Produkte zwar von Kunden vor Ort eingekauft werden sollen, die Zahlung allerdings aus technischer Sicht eine Online-Zahlung darstellt, da der Kunde keine Kassa und kein physisches Kreditkartenterminal besucht und stattdessen seine Zahlungsdaten von seinem Endgerät an einen Zahlungsserver übermittelt.

Arten von Zahlungen

Grundsätzlich lassen sich Zahlungen grob auf Online- und Offlinezahlungen einteilen:

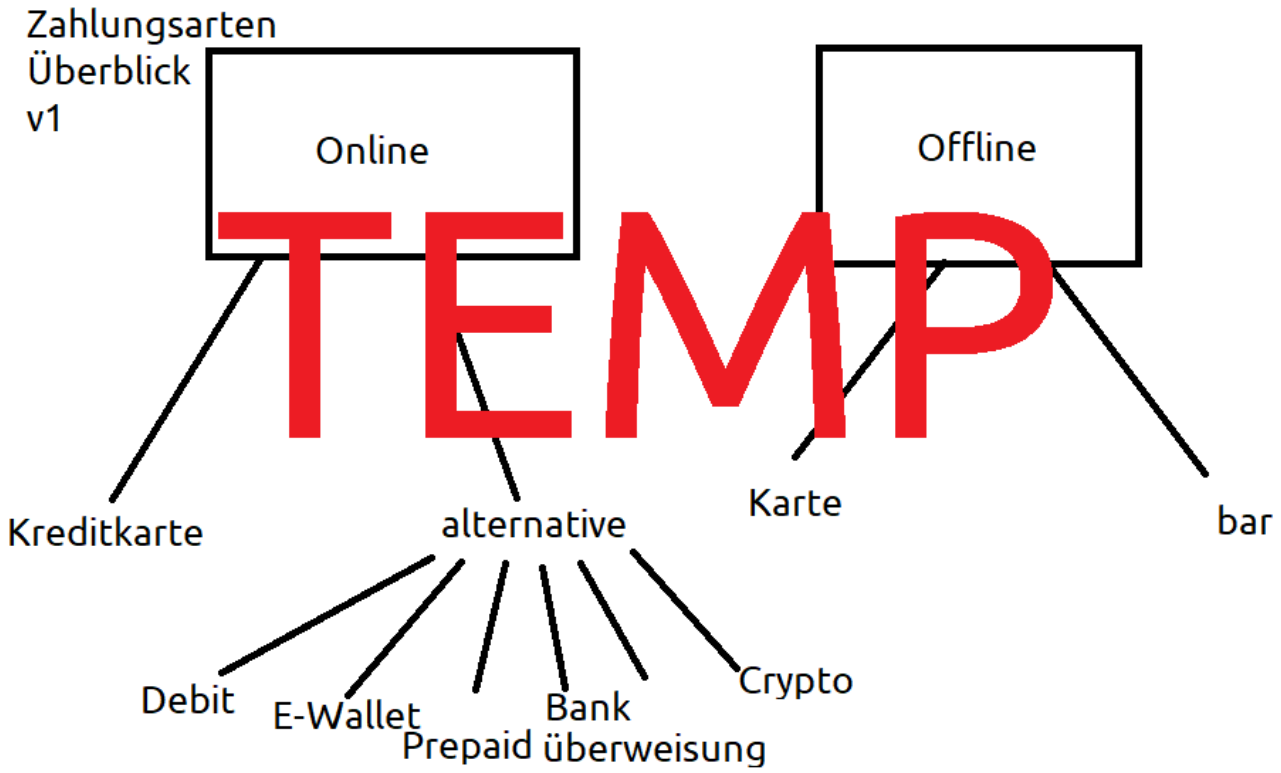


Abbildung: Zahlungsarten

Bei Offlinezahlungen (z.B. Zahlung mit Kreditkarte per NFC oder Magnet- oder Chipleser) ist der Kunde physisch anwesend, bei Offlinezahlungen werden nur Daten der Kreditkarte übergeben.

Diese Arbeit fokussiert sich rein auf Zahlungen im Onlinebereich.

E-Wallets

E-Wallets oder Cyberwallets sind virtuelle "Geldbörsen", die ihren Benutzern erlauben, bei einem Anbieter ein Guthaben aufzuladen. Anbieter, die solche Lösungen anbieten inkludieren im DACH-Raum[79]:

- Google Pay
- Apple Pay
- Amazon Pay
- PayPal
- Neteller
- Skrill
- Bluecode
- paydirekt (Deutschland)

Nicht jeder Anbieter bietet allerdings Onlinezahlungen und Offlinezahlungen an Terminals an.

Ein wichtiger Vorteil solcher Wallets ist die Möglichkeit, Onlinezahlungen schnell und ohne erneutes Ausfüllen von Kreditkarteninformationen abzuwickeln und somit Absprünge während dem Zahlungsprozess zu verringern.

[79] [Infographic] Mobile Payments and Digital Wallets in Europe (2018): Clearhaus, https://www.clearhaus.com/blog/mobile-payments-and-digital-wallets/ abgerufen am 30.03.2021

PCI DSS-Konformität

PCI DSS, auch Payment Card Industry Data Security Standard, ist ein Standard für Informationssicherheit in der digitalen Zahlungsabwicklung[80]. Er wird von dem PCI Security Standards Council ausgegeben, einem Rat, der aus American Express, Discover, JCB, Mastercard und Visa besteht.[81]

Darin wird unter anderem festgelegt, welche Sicherheitsauflagen Zahlungsanbieter und Integrationen zu ihren Systemen zu erfüllen haben.

Jeder Zahlungsanbieter muss abhängig von seinem Transaktionsvolumen und dem Anbieter[82] unterschiedliche Kriterien für eine Zertifizierung erfüllen. Diese Kriterien werden im Falle von Visa in vier Ebenen aufgeteilt [83]:

- Level 1: Mehr als 6 Millionen Transaktionen pro Jahr
- Level 2: 1 bis 6 Millionen Transaktionen pro Jahr
- Level 3: 20.000 bis 1 Million Transaktionen pro Jahr
- Level 4: Weniger als 20.000 Transaktionen pro Jahr

Eine Zertifizierung ist 12 Monate gültig[84].

Da es sich bei dem konkreten Anwendungsfall im Projekt um einen Prototypen handelt, muss noch nicht vollständig auf PCI DSS-Compliance geachtet werden, bei dem Vergleich von Anbietern wird allerdings darauf eingegangen, ob der Anbieter solche Zertifizierungen besitzt.

[80] PCI Security Standards Council, https://www.pcisecuritystandards.org/ abgerufen am 02.04.2021

[81] PCI Security: PCI Security Standards Council, https://www.pcisecuritystandards.org/pci_security/ abgerufen am 02.04.2021

[82] PCI DSS v3.2.1 Quick Reference Guide (2018): PCI Security Standards Council, https://www.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf?agreement=true&time=1617368112327 abgerufen am 02.04.2021.

[83] Baker, Alice (2020): A guide to the 4 PCI DSS compliance levels, IT Governance Blog En, https://www.itgovernance.eu/blog/en/a-guide-to-the-4-pci-dss-compliance-levels abgerufen am 02.04.2021.

[84] Learn More about Service Providers, Visa, https://usa.visa.com/splisting/splistinglearnmore.html#pdvsp abgerufen am 02.04.2021

PSD2 und Strong Customer Authentication

PSD (oder: Payment Services Directive, Zahlungsdiensterichtlinie) ist eine EU-Richtlinie, die Zahlungsdienstleistungen und Zahlungsanbieter im EU-Raum reguliert.

Eine wichtige Änderung von PSD2 ist die Pflicht zu Strong Customer Authentication (kurz SCA) bei Onlinezahlungen. Das bedeutet, dass besonders bei Onlinezahlungen klar sichergestellt werden muss, dass der Karteninhaber die Zahlung tätigt.

Dies muss mit mindestens zwei von drei Faktoren überprüft werden:

- Knowledge (Wissen über etwas, beispielsweise ein Passwort)
- Possession (Besitz von einem Objekt, beispielsweise eine Kreditkarte oder ein Smartphone mit einem Einmalcode)
- Inheritence (Etwas was der Benutzer ist, beispielsweise ein Fingerabdruck oder Gesichtserkennung)

Diese drei Faktoren sind stark vergleichbar zu dem Mehrstufenauthentifizierungsparadigma "Something you know, something you have, something you are", wo dieselben Faktoren für eine sichere Mehrstufenauthentifikation empfohlen werden.

Die Vorgaben von SCA im Vergleich zu einer Kartenzahlung an einem Kassenterminal werden in der folgenden Grafik (in strichlierten Boxen) erörtert:

SCA wird meist von Kreditkartenunternehmen übernommen, Zahlungsanbieter und Integrationen zu ihnen müssen dies aber unterstützen. Hierfür existiert das 3-D Secure Protokoll, das von Kreditkartenunternehmen unter unterschiedlichen Namen implementiert wird. Beispiele für solche Namen sind:

- Visa: Visa Secure[85] (oder Verified by Visa[86])
- Mastercard: SecureCode oder Mastercard® Identity Check™[87]
- Discover ProtectBuy[88]

Für Integrationen mit Zahlungsanbietern ist wichtig, dass der Zahlungsanbieter sowohl als auch die Integration 3-D Secure unterstützen.

Wenn eine Zahlung mit 3D Secure abgewickelt wird kann ein Liability Shift, also eine Übertragung der Verantwortlichkeit vom verkaufenden Händler zum Kreditkartenunternehmen, geschehen. Dadurch ist im Fall eines Konflikts der Kreditkartenausgeber dafür verantwortlich, nachzuweisen, ob der Kreditkartenbesitzer die Zahlung autorisierte und nicht mehr der Händler. [89] [90]

[85] Visa: Visa Secure, https://www.visa.de/bezahlen-mit-visa/genutzte-technologien/visa-secure.html abgerufen am 02.04.2021

[86] card complete: Visa Secure, https://www.cardcomplete.com/fuer-unternehmen/e-m-commerce/3-d-secure/visa-secure/ abgerufen am 02.04.2021

[87] Mastercard: Mastercard® Identity Check™ Vorteile & Anmeldung, https://www.mastercard.at/de-at/privatkunden/services-wissenswertes-innovationen/innovationen/idcheck.html abgerufen am 02.04.2021

[88] Discover Global Network: ProtectBuy, https://www.discoverglobalnetwork.com/en-us/business-resources/

fraud-security/products-tools/protect-buy abgerufen am 02.04.2021

[89] Paysafe Developer: 3D Secure Results and Liability Shift, https://developer.paysafe.com/en/classic-apis/3ds/test-and-go-live/3d-secure-results-and-liability-shift/#Liability abgerufen am 06.04.2021

[90] Stripe Docs: Card authentication and 3D Secure, https://stripe.com/docs/payments/3d-secure abgerufen am 06.04.2021

Vergleich von Zahlungsanbietern

Vergleichskriterien

Bevor ein Vergleich zwischen Anbietern sinnvoll durchführbar ist, ist es wichtig, Kriterien und Gewichtungen aufzustellen, nach welchen jeder Anbieter evaluiert werden kann.

Die Kriterien, welche im Angesicht des gegebenen Projekts gewählt werden, sind:

- API: Verfügbarkeit, Dokumentation, Unterstützung und Support für Entwickler
- angebotene Zahlungsarten
- Konformität mit Standards und Zertifizierungen
- Transaktionsgebühren
- Weitere relevante Funktionen, die eine Integration erleichtern oder einen Vorteil gegenüber anderen Anbietern darstellen

Diese Kriterien werden nach einer Punktevergabe von 0 bis 10 folgendermaßen gewichtet:

Kriterium	**Gewichtung**
API: Verfügbarkeit und angebotene Funktionen	25%
API: Qualität der Dokumentation	25%
angebotene Zahlungsarten	25%
Konformität mit Standards und Zertifizierungen	5%
Kosten und Gebühren	5%
Sonstige Funktionen	15%

Diese Gewichtung bildet die Prioritäten für das gegebene Projekt ab. Folgend sind einige Begründungen sowie Zusatzinformationen zu dieser Gewichtung:

- 50% der Gewichtung liegen auf der API und Funktionen, die der Anbieter durch diese zur Verfügung stellt und wie gut diese dokumentiert sind. Besonders im gegebenen Projektzusammenhang ist dies ein sehr wichtiger Teil, da die Applikation künftig um weitere Funktionen erweitert werden könnte und fehlende Möglichkeiten vonseiten des Zahlungsanbieters ein großes Problem darstellen können. Eine gute Dokumentation ist auch wichtig, da sie eine Integration stark vereinfachen kann.
- 25% der Gewichtung nehmen angebotene Zahlungsanbieter im DACH-Raum ein. Da jeder potentielle Kunde nur wenige Zahlungsoptionen eingerichtet hat, verwendet bzw. verwenden will ist es wichtig, so viele Zahlungsmöglichkeiten wie möglich anzubieten, um die meisten Kunden zu erreichen und einen Abbruch des Einkaufs bei der Zahlung zu verhindern. Weltweit sind 73% von Onlinehändlern der Meinung, dass abgebrochene Zahlungen im Onlinehandel ein Problem sind[91].
- Kosten, Gebühren und Zertifizierungen werden mit nur insgesamt 10% kaum berücksichtigt, da sich diese Faktoren häufig verändern und für den Prototypen und diese Analyse kaum relevant sind.
- Sonstige Funktionen beinhalten schwer quantifizierbare Vorteile oder Nachteile einzelner Anbieter. Sie können beispielsweise eine vorgefertigte, gehostete Zahlungsseite oder eine White-Label Lösung beinhalten.

Anbieter

Verglichen werden die folgenden Zahlungsanbieter:

- Stripe
- Rapyd

- Adyen
- Braintree

Diese Liste zielt darauf ab, unterschiedliche Anbieter für Onlinezahlungen im DACH-Raum abzudecken. Sie ist nicht vollständig.

Anbieter wurden nur in die Liste aufgenommen, wenn sie die folgenden Grundkriterien erfüllten:

- Der Anbieter bietet Zahlungen in Deutschland, Österreich und der Schweiz an
- Der Anbieter unterstützt Onlinezahlungen
- Der Anbieter bietet eine API an, mit welcher individuelle Applikationen außerhalb von vorgefertigten Integrationen (z.B. WooCommerce-Integrationen) Zahlungen abwickeln können
- Der Anbieter ermöglicht für diesen Vergleich Zugriff auf das API-Interface
- Informationen für den Vergleich müssen öffentlich auf der Website oder per Support erhältlich sein

Auch werden direkte Anbindungen zu Banken, Kreditkartenunternehmen oder z.B. Apple Pay oder PayPal nicht verglichen.

[91] How COVID-19 has reshaped the SMB checkout: Online (2020): Paysafe, https://www.paysafe.com/en/blog/how-covid-19-has-reshaped-the-smb-checkout-online/#c182623:%7E:text=73%25%20of%20online%20businesses%20say%20cart%20abandonments%20are%20an%20issue%20for%20them abgerufen am 30.03.2021

Stripe

Das Leitmodell von Stripe ist:

> Our mission is to increase the GDP of the internet
>
> — About Us: Stripe Inc., https://stripe.com/en-at/about abgerufen am 01.04.2021.

Als primär für Entwickler gestaltete [92] Lösung bietet Stripe weltweite Zahlungen mittels REST-API oder Integrationen mit tausenden eCommerce-Systemen und Plattformen [93] an.

API: Verfügbarkeit und Funktionen

Stripe legt großen Wert darauf, dass fast jede Operation über die REST-API abgewickelt werden kann. Außerhalb der Verifizierung als Firma kann von Zahlungen über Kartenausstellung bis hin zu Auszahlungen fast jede Operation automatisiert werden.

Für die API werden Libraries für Ruby, Python, PHP, Java, Node.JS, Go und C#/.NET sowie ein CLI (Command Line Interface) angeboten. Mobile SDKs werden für Android sowie für iOS angeboten[94]. Für Dart existieren mehrere von Privatpersonen erstellte Packages[95] für server- und clientseitige Applikationen.

Rückwärtskompatible Änderungen in der API werden konstant ohne neue API-Versionen veröffentlicht, für nicht rückwärtskompatible Änderungen werden neue Versionen veröffentlicht, die Entwickler aktiv auswählen müssen. Dies geschieht entweder mittels einer manuellen Einstellung im Dashboard oder durch Mitsenden des HTTP-Headers `Stripe-Version`. Neue Versionen werden unregelmäßig in Abständen von ungefähr 1-6 Monaten veröffentlicht, alte Versionen werden weiterhin unterstützt. [96]

Die Stripe-API ist hochverfügbar und meldet eine Betriebszeit von 99.999% in den

letzten 90 Tagen ab 01.04.2021 [97]. Die letzte schwerwiegende Störung (Stand 01.04.2021) war circa 2 Stunden lang und am 10.07.2019. [98] [99]

Da kaum Ausfälle bekannt sind und die API sehr fortgeschritten ist schneidet Stripe bei diesem Kriterium mit **10/10 Punkten** ab.

API: Dokumentation

Stripe bietet[100] einen Quickstart-Guide[101] und eine vollständige API-Referenz für die REST-API an. Weiters existieren detaillierte Referenzen für das Stripe CLI[102] und die SDKs für Android und iOS.

Für jeden Codeblock in der Dokumentation und in Hilfeartikeln werden genaue Instruktionen pro verwendeter Library angegeben, für jede Route der API können Requestcodezeilen in der gewählten Programmiersprache und Responseobjekte eingesehen werden. Diese Codebeispiele beinhalten auch bereits im Testmodus existierende Objekte und API-Keys, sodass sie auch direkt in ein Programm eingefügt werden können.

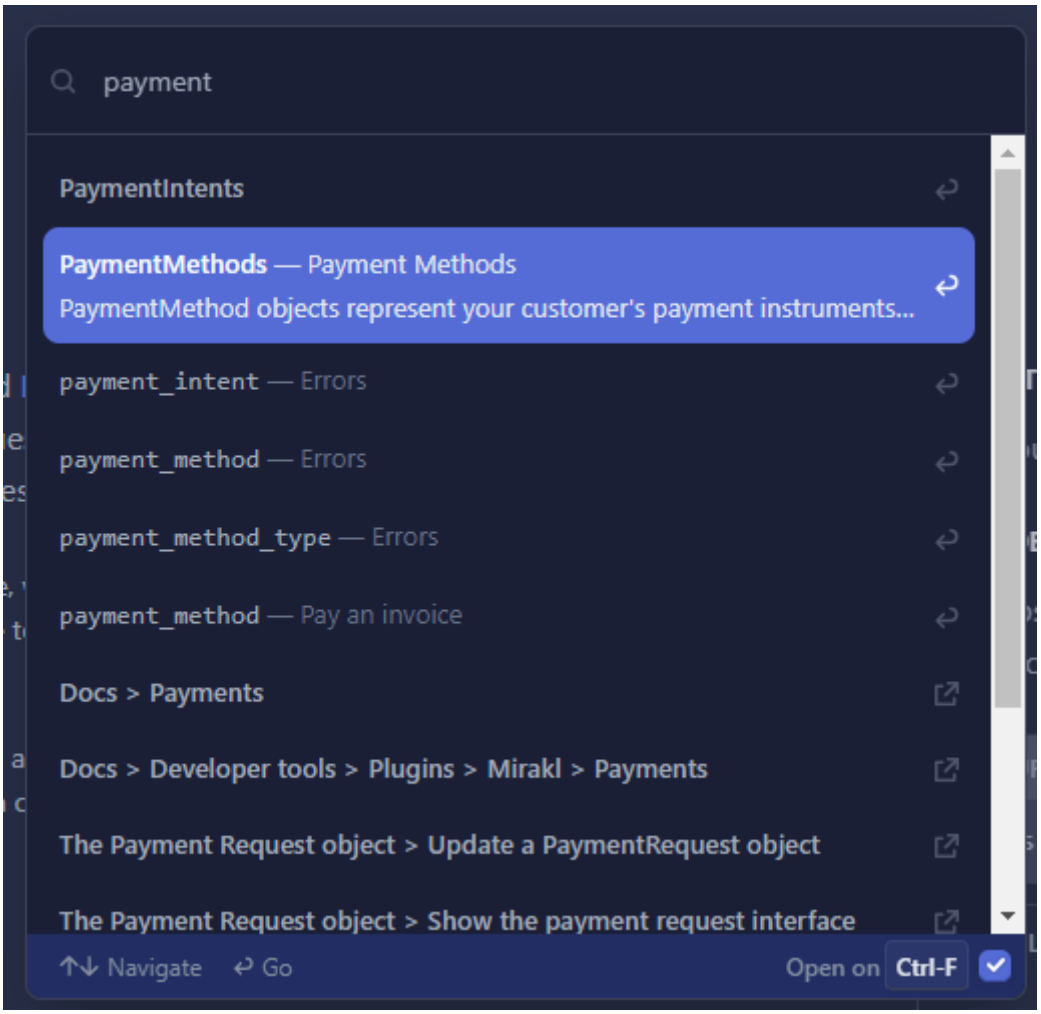
POST /v1/payment_intents Java

```java
Stripe.apiKey = "sk_test_[redacted]";

List<Object> paymentMethodTypes =
  new ArrayList<>();
paymentMethodTypes.add("card");
Map<String, Object> params = new HashMap<>();
params.put("amount", 2000);
params.put("currency", "eur");
params.put(
  "payment_method_types",
  paymentMethodTypes
);

PaymentIntent paymentIntent =
  PaymentIntent.create(params);
```

RESPONSE


```json
{
  "id": "pi_1GgXkKELqGlyTn1tp6l55UyY",
  "object": "payment_intent",
  "amount": 2000,
  "amount_capturable": 0,
  "amount_received": 0,
  "application": null,
  "application_fee_amount": null,
  "canceled_at": null,
  "cancellation_reason": null,
  "capture_method": "automatic",
  "charges": {
    "object": "list",
    "data": [],
    "has_more": false,
    "url": "/v1/charges?payment_intent=pi_1GgXkKELqGlyTn1tp6l55UyY"
  },
```


Pro API-Operation werden Objekte und ihre Attribute gezeigt und genauer erklärt. Mit einer Suchfunktion kann nach Properties, Requests, Objekten und Hilfeartikeln gesucht werden:



Da diese Dokumentation sehr detailliert ist, die gesamte API abdeckt und fertige, auf den Benutzer zugeschnittene, Codebeispiele zur Verfügung stellt, schneidet Stripe bei diesem Kriterium mit **10/10 Punkten** ab.

Angebotene Zahlungsarten

Stripe bietet abhängig von der Region des Händlers unterschiedliche Zahlungsarten an.

Für Händler in Österreich werden Zahlungen mittels Visa, Mastercard, Maestro, American Express, Apple Pay, Google Pay, iDeal, Sofort, SEPA, Klarna, Giropay, Alipay, Bancontact, Cartes Bancaires und WeChat Pay ermöglicht.

Ein Nachteil an Stripe ist allerdings, dass PayPal nicht akzeptiert wird. Dies ist ein starker Nachteil, besonders da PayPal 2019 20,2% des E-Commerce-Umsatzes in

Deutschland ausmachte[103].

Da Stripe sehr viele Zahlungsmethoden inklusive E-Wallets anbietet, PayPal oder Diners Club allerdings nicht akzeptiert schneidet Stripe hier mit **8/10 Punkten** ab.

Konformität mit Standards und Zertifizierungen

Stripe ist zertifiziert als PCI Service Provider Level 1[104] . Da dies die industrieübliche Zertifizierung für Zahlungsanbieter ist, werden hierfür **10/10** Punkte vergeben.

Kosten und Transaktionsgebühren

Transaktionsgebühren von Stripe[105] sind abhängig von der Position des Händlers von Region zu Region unterschiedlich, für diesen Vergleich werden die Preise für österreichische Händler verglichen.

Für innereuropäische Kredit- und Debitkarten werden pro Transaktion 1,4% und € 0,25 verrechnet, für internationale Karten werden 2,9% und € 0,25 verrechnet (2% werden für Währungsumrechnungen verrechnet)

Für lokale Zahlungsmethoden wie SEPA-Überweisungen werden andere Preise [106] angewandt.

Stripe liegt im preislichen Mittelfeld, hat allerdings Ausnahmen bei einigen Zahlungsarten, was die Errechnung von Kosten undurchsichtiger machen kann. Stripe erhält für dieses Kriterium **8/10 Punkte**.

Sonstige Funktionen

- Stripe bietet einen vollständigen Testmodus mit eigenen Daten, Keys sowie Endpoints an.
- Mittels Stripe Checkout und Stripe.js kann schnell eine PCI-konforme, von Stripe

gehostete, Zahlungsseite mit modernem User Interface aufgesetzt werden

- Mit Webhooks können zahlreiche Ereignisse in Echtzeit von Stripe an Anwendungsserver übermittelt werden
- Stripe unterstützt die Übergabe eines Idempotenztokens, um sicherzustellen, dass wiederholte Anfragen nur zu einer Aktion (z.B. Zahlung) führen können
- Sämtliche Objekt-IDs in Stripe sind mit einem Präfix versehen, das den Entwickler schnell erkennen lässt, um welches Objekt es sich handelt. So beginnen beispielsweise IDs von PaymentIntents mit pi_ und IDs von Rechungen (Invoices) mit in_.
- Stripe führt eine Liste von Testkartennummern[107] , welche im Testmodus spezifische Vorgänge wie 3D Secure Verifikationen simulieren.
- Der Support von Stripe kann Anfragen zu Problemen mit APIs schnell und akkurat beantworten.

Fazit

API: Funkt.	API: Dokum.	Zahlungs-arten	Zertif.	Kosten	Sonstige	Gesamt
10/10	10/10	8/10	10/10	8/10	10/10	94%

[92] About Us: Stripe, Inc., https://stripe.com/en-at#:~:text=Designed%20for%20developers abgerufen am 01.04.2021.

[93] Stripe Partner Program: Stripe, Inc., https://stripe.com/en-at/partner-program#who-partners-with-stripe:~:text=Thousands%20of%20companies abgerufen am 01.04.2021.

[94] Documentation: Stripe Docs, https://stripe.com/docs abgerufen am 01.04.2021.

[95] pub.dev: Stripe, https://pub.dev/packages?q=stripe abgerufen am 01.04.2021

[96] API upgrades: Stripe Docs, https://stripe.com/docs/upgrades abgerufen am 01.04.2021.

[97] Stripe System Status: Stripe, Inc., https://status.stripe.com/ abgerufen am 01.04.2021.

[98] Singleton, David (2019): Root cause analysis: significantly elevated error rates on 2019-07-10, Stripe, https://stripe.com/rcas/2019-07-10 abgerufen am 01.04.2021.

[99] Davis, Vincy (2019): Stripe's API suffered two consecutive outages yesterday causing elevated error rates and response times, Packt Hub, https://hub.packtpub.com/stripes-api-suffered-two-consecutive-outages-

yesterday-causing-elevated-error-rates-and-response-times/ abgerufen am 01.04.2021.

[100] Stripe, Inc.: Stripe API Reference, Stripe Docs, https://stripe.com/docs/api abgerufen am 01.04.2021.

[101] Stripe, Inc: Quick Start, https://stripe.com/docs/development/quickstart abgerufen am 01.04.2021

[102] Stripe, Inc.: Stripe CLI Reference, Stripe Docs, https://stripe.com/docs/cli abgerufen am 01.04.2021.

[103] Coelsch, Caroline (2020): Studie: Online-Payment 2020, EHI, https://www.ehi-shop.de/image/data/PDF_Leseproben/Studie_Online-Payment_2020_Leseprobe.pdf abgerufen am 01.04.2021.

[104] Service Providers, Visa, https://www.visa.com/splisting/searchGrsp.do?companyNameCriteria=stripe abgerufen am 01.04.2021

[105] Stripe, Inc.: Pricing & fees, Stripe, https://stripe.com/en-at/pricing abgerufen am 01.04.2021.

[106] Stripe: Local Payment methods, https://stripe.com/en-at/pricing/local-payment-methods abgerufen am 07.04.2021

[107] Stripe Docs: Testing, https://stripe.com/docs/testing abgerufen am 06.04.2021

Rapyd

API: Verfügbarkeit und Funktionen

Rapyd verfügt über eine detaillierte, weitreichende REST-API, welche alle Abläufe rund um Zahlungen automatisieren kann.

Die API ist aufgeteilt in Rapyd Collect, Rapyd Disburse, Rapyd Wallet und Rapyd Issuing [108].

Rapyd bietet iOS- und Android-SDKs an[109] , allerdings keine Libraries für serverseitige Integrationen. Libraries von Privatpersonen sind auch nicht vorhanden [110] [111] .

Besonderheiten an der API sind:

- Die spezifischen Header `access_key, salt, signature, timestamp` werden verwendet [112], die Signatur muss pro Request ein BASE64-verschlüsselter SHA-256 Hash aus HTTP-Methode, URL, Salt, Timestamp, Access Key, Secret Key und Body sein.
- Um eine Zahlung mit gegebenen Karten- und Kundendaten zu erstellen muss in separaten Requests eine Wallet, ein Customer und erst dann ein Payment erstellt werden.

Die API liefert detaillierte Fehlermeldungen mit genauen Erklärungen, welche Objekte fehlerhaft sind und wieso eine Anfrage abgelehnt wurde.

Statusdaten werden seitens Rapyd nicht veröffentlicht.

Rapyd erhält hierfür **7/10 Punkte**, da die API zwar sehr fortgeschritten ist, allerdings keine Uptime ermittelt werden kann, keine Libraries für Server existieren und durch Faktoren wie die Requestsignatur die Komplexität stark erhöht wird.

API: Dokumentation

Rapyd bietet Quick Start Guides, die von der Erstellung eines Testkontos bis zum ersten API-Aufruf anhand von Beispielen erklären, wie Anfragen an die API erstellt werden sollen [113]. Darin wird auch eine Postman-Collection angeboten, die alle Routen und ihre Dokumentation beinhaltet. Wenn in dieser Postman-Collection API-Keys eingegeben werden, können sofort alle Routen aufgerufen werden.

Die API-Referenz erörtert sämtliche Objekte, Aufrufe und Fehlermeldungen und blendet seitlich JSON-Beispiele ein.

Da die Dokumentation sehr detailliert und Anweisungen sehr klar sind erhält Rapyd hier **10/10 Punkte**.

Angebotene Zahlungsarten

Rapyd bietet in Österreich, Deutschland und der Schweiz Kreditkartenzahlungen, in Österreich und Deutschland Banküberweisungen an [114]. E-Wallets fehlen allerdings in Europa vollständig.

Da E-Wallets besonders im gegebenen Projekt bei Mobilzahlungen sehr wichtig sind und Zahlungen per Bankkonto nicht im gesamten DACH-Raum verfügbar sind erhält Rapyd für dieses Kriterium nur **5/10 Punkte**.

Konformität mit Standards und Zertifizierungen

Die Zahlungsinfrastruktur von Rapyd ist kompatibel mit PCI-DSS und PSD2-Richtlinien [115] und erhält somit **10/10 Punkte**.

Kosten und Transaktionsgebühren

Rapyd bietet weltweite Transaktionen für 2,9% + 30 Cent Gebühr pro Transaktion[116], für Zahlungen mit europäischen Karten werden 1,2% + 25 Cent verrechnet[117].

Rapyd ermöglicht vergleichsweise billige Zahlungen mit europäischen Karten und liegt bei internationalen Karten im Mittelfeld. Da der Preis einfach gestaltet ist und keine Ausnahmen gemacht werden erhält Rapyd hier **9/10 Punkte**.

Sonstige Funktionen

- Rapyd bietet einen Sandboxmodus, in welchem API-Anfragen ähnlich zum Produktivsystem getestet werden können
- Mit Webhooks können zahlreiche Ereignisse in Echtzeit von Rapyd an Anwendungsserver übermittelt werden
- Rapyd versieht Objekt-IDs mit Prefixes (cus_… für Customer/Kunden, payment_… für Zahlungen)
- Mittels Rapyd Checkout kann eine PCI/DSS-konforme Lösung schnell in ein existierendes System integriert werden. Rapyd Checkout wird als eigene gehostete Popupseite oder als Toolkit für einzelne Elemente auf einer existierenden Website angeboten[118].

Fazit

API: Funkt.	API: Dokum.	Zahlungs-arten	Zertif.	Kosten	Sonstige	Gesamt
7/10	10/10	5/10	10/10	9/10	10/10	79,5%

[108] Rapyd Docs: Guides & Use Cases, https://docs.rapyd.net/build-with-rapyd/docs abgerufen am 06.04.2021

[109] Rapyd Docs: Introduction to Mobile SDK, https://docs.rapyd.net/works-with/reference/introduction-to-mobile-sdk abgerufen am 06.04.2021

[110] Maven Repository: rapyd, https://mvnrepository.com/search?q=rapyd abgerufen am 02.04.2021.

[111] NuGet: rapyd, https://www.nuget.org/packages?q=rapyd abgerufen am 02.04.2021.

[112] Rapyd Docs: Make your first API call, https://docs.rapyd.net/build-with-rapyd/docs/make-your-first-api-call#step-4-build-auth-headers abgerufen am 06.04.2021

[113] Rapyd Docs: Get Started, https://docs.rapyd.net/build-with-rapyd/docs/getting-started abgerufen am 06.04.2021

[114] Rapyd: Countries and payments, https://www.rapyd.net/network/countries-and-payments/ abgerufen am

02.04.2021

[115] Rapyd: Collect Online, https://www.rapyd.net/platform/collect/online/#single-blocks:~:text=Get%20peace%2Dof%2Dmind%20with%20a%20fully%2Dcertified%20PSD2%20and%20PCI%2DDSS%20Compliant%20solution abgerufen am 02.04.2021

[116] Rapyd: Pricing, https://www.rapyd.net/rapyd-pricing/ abgerufen am 06.04.2021

[117] Rapyd Service Agreement, http://iconslib.rapyd.net/assets/client-portal/Service_Agreement.pdf abgerufen am 06.04.2021

[118] Rapyd Docs: Rapyd Checkout Overview, https://docs.rapyd.net/build-with-rapyd/docs/rapyd-checkout-overview abgerufen am 06.04.2021

Adyen

Adyen bietet Offlinezahlungen an Points of Sale sowie Onlinezahlungen an.

API: Verfügbarkeit und Funktionen

Adyen bietet eine REST-API an, über welche Zahlungen abgewickelt, Abonnements eingerichtet und Daten verändert werden können.

Für Verbindungen zur API werden Libraries für Ruby, Java, PHP, Python, C#, NodeJS und Go angeboten [119].

Adyen veröffentlicht Statusdaten[120], schwerwiegende Ausfälle sind seit Mai 2020 nicht bekannt (Stand: 06.04.2021)[121].

Da die API ausführlich sämtliche Zahlungsfunktionen abdeckt erhält Adyen für dieses Kriterium **10/10 Punkte**.

API: Dokumentation

Adyen dokumentiert die API mit Quick Start Guides[122] und einer API-Referenz (API Explorer)[123] .

Die Quick Start Guides beinhalten an relevanten Stellen Beispielcode für cURL und alle Libraries:

curl Ruby Java PHP Python C# NodeJS Go

```
curl https://checkout-test.adyen.com/v66/paymentMethods \
-H "x-API-key: YOUR_X-API-KEY" \
-H "content-type: application/json" \
-d '{
  "merchantAccount": "YOUR_MERCHANT_ACCOUNT",
  "countryCode": "NL",
  "amount": {
    "currency": "EUR",
    "value": 1000
  },
  "channel": "Web",
  "shopperLocale": "nl-NL"
}'
```

[124]

Der API Explorer bietet JSON-Beispiele für mehrere Business Cases an, die gleich simuliert werden können und erklärt Request- und Responseparameter.

POST /payments v67 latest

https://checkout-test.adyen.com/v67/payments

Sends payment parameters (like amount, country, and currency) together with other required input details collected from the shopper. To know more about required parameters for specific payment methods, refer to our payment method guides. The response depends on the payment flow:

- For a direct flow, the response includes a `pspReference` and a `resultCode` with the payment result, for example **Authorised** or **Refused**.
- For a redirect or additional action, the response contains an `action` object.

Request parameters Expand all parameters

accountInfo

Shopper account information for 3D Secure 2.

For 3D Secure 2 transactions, we recommend that you include this object to increase the chances of achieving a frictionless flow.

additionalData [string] Object

This field contains additional data, which may be required for a particular payment request.

The `additionalData` object consists of entries, each of which includes the key and value.

Select additionalData

amount Required

The amount information for the transaction (in minor units). For BIN or card verification requests, set amount to 0 (zero).

applicationInfo

Make a card payment

- Make a card payment
- Make a card payment (using encrypted card details)
- Make a card payment with 3D Secure redirect authentication
- Make a card payment with 3D Secure 2 native authentication
- Tokenize card details for one-off payments
- Tokenize card details for a subscription
- Make a one-off payment with a token and CVV
- Make a subscription payment with a token
- Make an Apple Pay payment
- Make a giropay payment
- Make a Google Pay payment
- Make an iDEAL payment
- Make a Klarna payment
- Make a Sofort payment
- Split a payment between a sub-merchant and a platform account

```
    "expiryMonth": "03",
    "expiryYear": "2030",
    "holderName": "John Smith",
    "cvc": "737"
  },
  "returnUrl": "https://your-company.com/...",
  "merchantAccount": "YOUR_MERCHANT_ACCOUNT"
}
```

Run

[125]

Adyen erhält für die Dokumentation **10/10 Punkte**.

Angebotene Zahlungsarten

Adyen bietet Zahlungen mit allen globalen Zahlungsarten und E-Wallets an.

Im DACH-Raum werden Zahlungen mittels: Visa, Mastercard, American Express, Sofort, Alipay, Klarna, Apple Pay, UnionPay, Google Pay, Amazon Pay, SEPA (Nur DE, AT), JCB, Diners Club, Discover, PayPal

Diese weite Kombination von Zahlungsmethoden und PayPal erreicht in diesem Kriterium **10/10 Punkte**.

Konformität mit Standards und Zertifizierungen

Adyen ist PCI DSS-zertifizierter Level 1 Service Provider [126]. Somit erhält Adyen hierfür **10/10 Punkte**.

Kosten und Transaktionsgebühren

Im Vergleich zu anderen Zahlungsanbietern verrechnet Adyen pro Zahlungsart unterschiedliche Gebühren [127].

Eine Bearbeitungsgebühr von 0,10€ wird pro Zahlung verrechnet, dazu werden Kosten abhängig von der verwendeten Zahlungsart aufgeschlagen.

Bei Visa, Mastercard, V-Pay und Maestro werden Kosten nach Interchange++ verrechnet. Diese sind im EU-Raum reguliert, weshalb sie immer unter 0,2% für Debitkarten beziehungsweise 0,3% für Kreditkarten liegen [128] [129].

Die Aufstellung der Transaktionsgebühren je Zahlungsart ist wie folgt:

Zahlungsart	Gebühren (exkl. €0,10 Bearbeitungsgebühr)
Amazon Pay	€ 0,80 + Interchange++ je nach verwendeter Karte[130]
Apple Pay, Google Pay	abhängig von verwendeter Karte
Diners Club oder Discover	3,95% oder Interchange++
EPS	1,3% + € 0,20
Maestro, V-pay	Interchange++
Mastercard, Visa	Interchange++
PayPal	abhängig von PayPal, üblicherweise 3,4% + 35 Cent [131]
SEPA	€ 0,25

Die Konditionen von Adyen sind mit 10 Cent Bearbeitungsgebühr und sehr geringem prozentuellem Abschlag im Vergleich sehr gut. Da allerdings für jede Zahlungsart andere Kosten angewandt werden und sich Interchange++ regelmäßig verändert kann es zu preislichen Planungsschwierigkeiten kommen. Adyen erhält trotzdem **9/10 Punkte**.

Sonstige Funktionen

- Adyen bietet vorgefertigte UI-Elemente an, um Zahlungen entgegenzunehmen [132], diese können entweder als einzelne Felder in ein UI integriert oder als gesamte Komponente importiert werden.
- Adyen bietet einen separaten Testmodus an, in dem API-Integrationen getestet werden können. Für die Testumgebung wird ein eigenes Konto angelegt.

Fazit

API: Funkt.	API: Dokum.	Zahlungs-arten	Zertif.	Kosten	Sonstige	Gesamt
10/10	10/10	10/10	10/10	9/10	10/10	99,5%

[119] Adyen Docs: Get Started, https://docs.adyen.com/online-payments/get-started, abgerufen am 06.04.2021

[120] Adyen: Platform Status, https://www.adyen.com/status abgerufen am 06.04.2021

[121] Adyen: Status history, https://www.adyen.com/landing/platform-status/history abgerufen am 06.04.2021

[122] Adyen Docs: Online Payments, https://docs.adyen.com/online-payments abgerufen am 06.04.2021

[123] Adyen: API Explorer, https://docs.adyen.com/api-explorer/#/CheckoutService/v67/overview abgerufen am 06.04.2021

[124] Bild: Adyen Docs: Web Drop-in integration guide, https://docs.adyen.com/online-payments/drop-in-web, abgerufen am 06.04.2021

[125] Bild: Adyen API Explorer, https://docs.adyen.com/api-explorer/#/CheckoutService/v67/post/payments, abgerufen am 06.04.2021

[126] Adyen: Certifications, https://www.adyen.com/platform/certifications abgerufen am 06.04.2021

[127] Adyen: Pricing, https://www.adyen.com/pricing?navItem=europe abgerufen am 06.04.2021

[128] Adyen: EU Interchange fee cap (2015), https://www.adyen.com/blog/all-you-need-to-know-about-the-eu-interchange-cap abgerufen am 06.04.2021

[129] Visa: Domestic Multi-lateral Interchange Fees Austria (2018), https://www.visa.co.uk/dam/VCOM/regional/ve/unitedkingdom/PDF/fees-and-interchange/austria-october-2018.pdf abgerufen am 06.04.2021

[130] Adyen: Amazon Pay, https://www.adyen.com/payment-methods/amazon-pay abgerufen am 06.04.2021

[131] PayPal AT: Händler- und Verkäufergebühren, https://www.paypal.com/at/webapps/mpp/merchant-fees abgerufen am 02.04.2021

[132] Adyen Docs: Pre-built UI options, https://docs.adyen.com/online-payments/prebuilt-ui abgerufen am 06.04.2021

Braintree

Braintree ist ein PayPal angehöriger[133] Zahlungsanbieter.

API: Verfügbarkeit und Funktionen

Braintree bietet zwei Arten von APIs an: REST per SDK und GraphQL.

Client SDKs werden für Android, iOS und JavaScript angeboten, Server SDKs für Java, C#/.NET, Node.js, Python und Ruby. [134] Für Flutter existieren zwei inoffizielle Wrapper Packages zu den Mobile SDKs [135] [136] .

Braintree bietet ein vollständig separates Sandboxsystem für Test-API-Aufrufe.

Als einziger Anbieter auf dieser Liste bietet Braintree eine GraphQL-API neben fortgeschrittenen SDKs für bekannte Programmiersprachen. Deshalb erreicht Braintree hier **10/10 Punkte**.

API: Dokumentation

Braintree bietet zwei Dokumentationsseiten für jeweils die SDKs und die GraphQL-API an.

Die SDK-Dokumentation bietet genaue Anleitungen für die Verwendung der SDKs mit Codesamples für jedes angebotene SDK. Die Referenz des SDK ist im Vergleich zu anderen Anbietern statt auf Routen auf Business Cases (z.B. Transaktion > Verkauf) [137] aufgeteilt und bietet für jeden Fall Codesamples und Methodendokumentationen.

Die GraphQL-Dokumentation besteht ebenfalls aus Anleitungen für spezifische Aufgaben (z.B. Erstellung von Transaktionen) [138] neben der nach Queries und Mutations sortierten API-Referenz [139].

Sie bietet auch einen GraphQL-basierten API-Explorer an, in dem Anfragen per

Browser direkt gegen Sandboxkonten ausprobiert werden können.[140]

Beide API-Zugriffsmethoden sind sehr detailliert dokumentiert, weshalb Braintree **10/10 Punkte** erhält.

Angebotene Zahlungsarten

Braintree bietet Zahlungen mit Visa, Mastercard, Maestro, JCB, Discover, American Express, Diners Club, UnionPay, PayPal, Apple Pay, Google Pay, Samsung Pay, Bancontact, eps, giropay, iDeal und Sofort an. Da dies die meisten Kreditkarten, E-Wallets und Zahlungsmethoden neben einer PayPal-Integration sind erhält Braintree hierfür **10/10 Punkte**.

Konformität mit Standards und Zertifizierungen

Braintree ist zertifizierter PCI DSS Level 1 Anbieter [141] [142] [143] und erhält somit in diesem Kriterium **10/10 Punkte**.

Kosten und Transaktionsgebühren

Braintree verrechnet für Kartenzahlungen oder Zahlungen per Google Pay oder Apple Pay 1,9% + 30 Cent pro Zahlung [144], Zahlungen über PayPal kosten wegen Gebühren von PayPal 3,4% + 35 Cent [145].

Braintree ist generell für Zahlungen mit europäischen Karten teurer als andere Anbieter. Durch ein klares Gebührenmodell für alle Zahlungsarten kann allerdings besser mit Transaktionsgebühren gerechnet und geplant werden. Braintree erhält deshalb 7/**10 Punkte** in diesem Kriterium.

Sonstige Funktionen

- Braintree stellt in seinen Mobile und Web SDKs Drop-In UIs zur Verfügung, mit welchen schnell eine benutzerfreundliche, PCI-konforme, Zahlungsseite in eine

existierende App eingebaut werden kann.

Fazit

API: Funkt.	API: Dokum.	Zahlungs-arten	Zertif.	Kosten	Sonstige	Gesamt
10/10	10/10	10/10	10/10	7/10	10/10	98,5%

[133] Braintree Payments: Braintree Privacy Statement, https://www.braintreepayments.com/at/legal/braintree-privacy-policy abgerufen am 02.04.2021

[134] Braintree Developer Documentation: Overview, https://developers.braintreepayments.com/start/overview#client-and-server-sdks abgerufen am 02.04.2021

[135] pub.dev:flutter_braintree, https://pub.dev/packages/flutter_braintree abgerufen am 02.04.2021

[136] pub.dev:braintree_payment, https://pub.dev/packages/braintree_payment abgerufen am 02.04.2021

[137] Braintree Developer Docs: Transaction: Sale, https://developers.braintreepayments.com/reference/request/transaction/sale/java abgerufen am 02.04.2021

[138] Braintree GraphQL API: Creating Transactions, https://graphql.braintreepayments.com/guides/creating_transactions/ abgerufen am 02.04.2021

[139] Braintree GraphQL API: Reference, https://graphql.braintreepayments.com/reference/#Mutation abgerufen am 02.04.2021

[140] Braintree GraphQL API: API Explorer, https://graphql.braintreepayments.com/explorer/ abgerufen am 02.04.2021

[141] Braintree Payments: Data Security, https://www.braintreepayments.com/at/features/data-security abgerufen am 02.04.2021

[142] Visa: Visa Global Registry of Service Providers, https://www.visa.com/splisting/searchGrsp.do abgerufen am 02.04.2021

[143] Mastercard: Site Data Protection (SDP) Program Registered Service Provider List, https://www.mastercard.us/content/dam/mccom/global/documents/Sitedataprotection/site-data-protection-pci-list.pdf abgerufen am 02.04.2021

[144] Braintree Payments: Pricing, https://www.braintreepayments.com/at/braintree-pricing abgerufen am 02.04.2021

[145] PayPal AT: Händler- und Verkäufergebühren, https://www.paypal.com/at/webapps/mpp/merchant-fees abgerufen am 02.04.2021

Fazit

Die Aufstellung der Punkte jeweiliger Anbieter ist wie folgt (sortiert nach Gesamtwertung, absteigend):

Anbieter	**API: Funkt.**	**API: Dokum.**	**Zahlungs-arten**	**Zertif.**	**Kosten**	**Sonstige**	**Gesamt**
Adyen	10/10	10/10	10/10	10/10	9/10	10/10	**99,5%**
Braintree	10/10	10/10	10/10	10/10	7/10	10/10	**98,5%**
Stripe	10/10	10/10	8/10	10/10	8/10	10/10	**94%**
Rapyd	7/10	10/10	5/10	10/10	9/10	10/10	**79,5%**

Aus diesem Vergleich geht hervor, dass sich Anbieter auf dieser Liste nur in angebotenen Zahlungsarten und Preisen stark unterscheiden. APIs sind zumeist vorhanden, gut unterstützt und dokumentiert, nötige Zertifizierungen sind überall vorhanden und alle Anbieter auf dieser Liste bieten fertige Integrationen und eine gewisse Anzahl an anderen sonstigen Funktionen an.

Adyen und Braintree konnten bei den angebotenen Zahlungsarten durch eine Möglichkeit, PayPal-Zahlungen anzunehmen, gut punkten. Rapyd schnitt bei API-Funktionen aufgrund fehlender serverseitiger SDKs schlechter ab und verlor bei den angebotenen Zahlungsarten Punkte, da E-Wallets im DACH-Raum nicht unterstützt werden und Bankzahlungen nicht überall möglich sind.

Adyen, Braintree und Stripe sind im Projektzusammenhang drei gut wählbare Anbieter. Mit jedem dieser Anbieter kann eine sinnvolle Integration für den Prototypen im Selfcheckout erstellt werden, die Entscheidung fällt im Endeffekt auf Faktoren wie die Wichtigkeit eines schnellen Supportteams, PayPal, persönliche Erfahrungen mit einzelnen Zahlungsanbietern oder Erfahrungen mit deren APIs aus.

Adyen eignet sich am Besten, wenn so viele Zahlungsarten wie möglich billig abgedeckt werden sollen, Braintree bietet fortgeschrittene SDKs und eine einfach verwendbare GraphQL-API und Stripe bietet schnelle Unterstützung bei API-Fragen als auch SDKs für die meisten Plattformen, inklusive open source Libraries für Dart/Flutter.

Glossar

API

Application Programming Interface, eine Schnittstelle, mit welcher Programme und Entwickler mit einem Softwaresystem kommunizieren können.

bzw.

beziehungsweise

NFC

Near Field Communication, eine Technologie, welche Datenaustausch über geringe Entfernungen ermöglicht. Wird häufig für kontaktloses Zahlen bei Kreditkarten verwendet.

SDK

Sammlung von Programmbibliotheken, die es Entwicklern ermöglichen, auf einem System basierende Anwendungen zu erstellen

Quellen

- Stripe, Inc.: Stripe API Reference, Stripe Docs, https://stripe.com/docs/api abgerufen am 01.04.2021.
- Stripe, Inc.: Pricing & fees, Stripe, https://stripe.com/en-at/pricing abgerufen am 01.04.2021.
- [Infographic] Mobile Payments and Digital Wallets in Europe (2018): Clearhaus, https://www.clearhaus.com/blog/mobile-payments-and-digital-wallets/ abgerufen am 30.03.2021
- PCI Security Standards Council, https://www.pcisecuritystandards.org/ abgerufen am 02.04.2021
- PCI Security: PCI Security Standards Council, https://www.pcisecuritystandards.org/pci_security/ abgerufen am 02.04.2021
- PCI DSS v3.2.1 Quick Reference Guide (2018): PCI Security Standards Council, https://www.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf?agreement=true&time=1617368112327 abgerufen am 02.04.2021.
- Baker, Alice (2020): A guide to the 4 PCI DSS compliance levels, IT Governance Blog En, https://www.itgovernance.eu/blog/en/a-guide-to-the-4-pci-dss-compliance-levels abgerufen am 02.04.2021.
- Learn More about Service Providers, Visa, https://usa.visa.com/splisting/splistinglearnmore.html#pdvsp abgerufen am 02.04.2021
- Visa: Visa Secure, https://www.visa.de/bezahlen-mit-visa/genutzte-technologien/visa-secure.html abgerufen am 02.04.2021
- card complete: Visa Secure, https://www.cardcomplete.com/fuer-unternehmen/e-m-commerce/3-d-secure/visa-secure/ abgerufen am 02.04.2021
- Mastercard: Mastercard® Identity Check™ Vorteile & Anmeldung, https://www.mastercard.at/de-at/privatkunden/services-wissenswertes-

innovationen/innovationen/idcheck.html abgerufen am 02.04.2021

- Discover Global Network: ProtectBuy, https://www.discoverglobalnetwork.com/en-us/business-resources/fraud-security/products-tools/protect-buy abgerufen am 02.04.2021
- Paysafe Developer: 3D Secure Results and Liability Shift, https://developer.paysafe.com/en/classic-apis/3ds/test-and-go-live/3d-secure-results-and-liability-shift/#Liability abgerufen am 06.04.2021
- Stripe Docs: Card authentication and 3D Secure, https://stripe.com/docs/payments/3d-secure abgerufen am 06.04.2021
- How COVID-19 has reshaped the SMB checkout: Online (2020): Paysafe, https://www.paysafe.com/en/blog/how-covid-19-has-reshaped-the-smb-checkout-online/#c182623:%7E:text=73%25%20of%20online%20businesses%20say%20cart%20abandonments%20are%20an%20issue%20for%20them abgerufen am 30.03.2021
- About Us: Stripe, Inc., https://stripe.com/en-at#:~:text=Designed%20for%20developers abgerufen am 01.04.2021.
- Stripe Partner Program: Stripe, Inc., https://stripe.com/en-at/partner-program#who-partners-with-stripe:~:text=Thousands%20of%20companies abgerufen am 01.04.2021.
- Documentation: Stripe Docs, https://stripe.com/docs abgerufen am 01.04.2021.
- pub.dev: Stripe, https://pub.dev/packages?q=stripe abgerufen am 01.04.2021
- API upgrades: Stripe Docs, https://stripe.com/docs/upgrades abgerufen am 01.04.2021.
- Stripe System Status: Stripe, Inc., https://status.stripe.com/ abgerufen am 01.04.2021.
- Singleton, David (2019): Root cause analysis: significantly elevated error rates on 2019-07-10, Stripe, https://stripe.com/rcas/2019-07-10 abgerufen am 01.04.2021.

- Davis, Vincy (2019): Stripe's API suffered two consecutive outages yesterday causing elevated error rates and response times, Packt Hub, https://hub.packtpub.com/stripes-api-suffered-two-consecutive-outages-yesterday-causing-elevated-error-rates-and-response-times/ abgerufen am 01.04.2021.
- Coelsch, Caroline (2020): Studie: Online-Payment 2020, EHI, https://www.ehi-shop.de/image/data/PDF_Leseproben/Studie_Online-Payment_2020_Leseprobe.pdf abgerufen am 01.04.2021.
- Service Providers, Visa, https://www.visa.com/splisting/searchGrsp.do?companyNameCriteria=stripe abgerufen am 01.04.2021
- Stripe: Local Payment methods, https://stripe.com/en-at/pricing/local-payment-methods abgerufen am 07.04.2021
- Stripe Docs: Testing, https://stripe.com/docs/testing abgerufen am 06.04.2021
- Rapyd Docs: Guides & Use Cases, https://docs.rapyd.net/build-with-rapyd/docs abgerufen am 06.04.2021
- Rapyd Docs: Introduction to Mobile SDK, https://docs.rapyd.net/works-with/reference/introduction-to-mobile-sdk abgerufen am 06.04.2021
- Maven Repository: rapyd, https://mvnrepository.com/search?q=rapyd abgerufen am 02.04.2021.
- NuGet: rapyd, https://www.nuget.org/packages?q=rapyd abgerufen am 02.04.2021.
- Rapyd Docs: Make your first API call, https://docs.rapyd.net/build-with-rapyd/docs/make-your-first-api-call#step-4-build-auth-headers abgerufen am 06.04.2021
- Rapyd Docs: Get Started, https://docs.rapyd.net/build-with-rapyd/docs/getting-started abgerufen am 06.04.2021
- Rapyd: Countries and payments, https://www.rapyd.net/network/countries-and-payments/ abgerufen am 02.04.2021

- Rapyd: Collect Online, https://www.rapyd.net/platform/collect/online/#single-blocks:~:text=Get%20peace%2Dof%2Dmind%20with%20a%20fully%2Dcertified%20PSD2%20and%20PCI%2DDSS%20Compliant%20solution abgerufen am 02.04.2021
- Rapyd: Pricing, https://www.rapyd.net/rapyd-pricing/ abgerufen am 06.04.2021
- Rapyd Service Agreement, http://iconslib.rapyd.net/assets/client-portal/Service_Agreement.pdf abgerufen am 06.04.2021
- Rapyd Docs: Rapyd Checkout Overview, https://docs.rapyd.net/build-with-rapyd/docs/rapyd-checkout-overview abgerufen am 06.04.2021
- Adyen Docs: Get Started, https://docs.adyen.com/online-payments/get-started, abgerufen am 06.04.2021
- Adyen: Platform Status, https://www.adyen.com/status abgerufen am 06.04.2021
- Adyen: Status history, https://www.adyen.com/landing/platform-status/history abgerufen am 06.04.2021
- Adyen Docs: Online Payments, https://docs.adyen.com/online-payments abgerufen am 06.04.2021
- Adyen: API Explorer, https://docs.adyen.com/api-explorer/#/CheckoutService/v67/overview abgerufen am 06.04.2021
- Bild: Adyen Docs: Web Drop-in integration guide, https://docs.adyen.com/online-payments/drop-in-web, abgerufen am 06.04.2021
- Bild: Adyen API Explorer, https://docs.adyen.com/api-explorer/#/CheckoutService/v67/post/payments, abgerufen am 06.04.2021
- Adyen: Certifications, https://www.adyen.com/platform/certifications abgerufen am 06.04.2021
- Adyen: Pricing, https://www.adyen.com/pricing?navItem=europe abgerufen am 06.04.2021

- Adyen: EU Interchange fee cap (2015), https://www.adyen.com/blog/all-you-need-to-know-about-the-eu-interchange-cap abgerufen am 06.04.2021
- Visa: Domestic Multi-lateral Interchange Fees Austria (2018), https://www.visa.co.uk/dam/VCOM/regional/ve/unitedkingdom/PDF/fees-and-interchange/austria-october-2018.pdf abgerufen am 06.04.2021
- Adyen: Amazon Pay, https://www.adyen.com/payment-methods/amazon-pay abgerufen am 06.04.2021
- PayPal AT: Händler- und Verkäufergebühren, https://www.paypal.com/at/webapps/mpp/merchant-fees abgerufen am 02.04.2021
- Adyen Docs: Pre-built UI options, https://docs.adyen.com/online-payments/prebuilt-ui abgerufen am 06.04.2021
- Braintree Payments: Braintree Privacy Statement, https://www.braintreepayments.com/at/legal/braintree-privacy-policy abgerufen am 02.04.2021
- Braintree Developer Documentation: Overview, https://developers.braintreepayments.com/start/overview#client-and-server-sdks abgerufen am 02.04.2021
- pub.dev:flutter_braintree, https://pub.dev/packages/flutter_braintree abgerufen am 02.04.2021
- pub.dev:braintree_payment, https://pub.dev/packages/braintree_payment abgerufen am 02.04.2021
- Braintree Developer Docs: Transaction: Sale, https://developers.braintreepayments.com/reference/request/transaction/sale/java abgerufen am 02.04.2021
- Braintree GraphQL API: Creating Transactions, https://graphql.braintreepayments.com/guides/creating_transactions/ abgerufen am 02.04.2021

- Braintree GraphQL API: Reference, https://graphql.braintreepayments.com/reference/#Mutation abgerufen am 02.04.2021
- Braintree GraphQL API: API Explorer, https://graphql.braintreepayments.com/explorer/ abgerufen am 02.04.2021
- Braintree Payments: Data Security, https://www.braintreepayments.com/at/features/data-security abgerufen am 02.04.2021
- Visa: Visa Global Registry of Service Providers, https://www.visa.com/splisting/searchGrsp.do abgerufen am 02.04.2021
- Mastercard: Site Data Protection (SDP) Program Registered Service Provider List, https://www.mastercard.us/content/dam/mccom/global/documents/Sitedataprotection/site-data-protection-pci-list.pdf abgerufen am 02.04.2021
- Braintree Payments: Pricing, https://www.braintreepayments.com/at/braintree-pricing abgerufen am 02.04.2021
- PayPal AT: Händler- und Verkäufergebühren, https://www.paypal.com/at/webapps/mpp/merchant-fees abgerufen am 02.04.2021

Untersuchung und Vergleich von Frontend Frameworks auf deren Eignung für mobile Apps

Alexander Wiener <wie17087@spengergasse.at>

Bedürfnisse

In diesem Kapitel wird erarbeitet, dass es wichtig ist seine Anforderungen an die App zu kennen, bevor man sich für eine Sprache und ein Framework entscheidet. Wie in den folgenden Kapiteln nämlich herausgerarbeitet wird hat alles seine Vor- und Nachteile, daher ist es umso wichtiger zu wissen, welche Sachen man benötigt und welche Funktionen nicht gebraucht werden. Vor der Programmierung sollte daher eine Liste erstellt werden, die beinhaltet welche Funktionen/Möglichkeiten die Sprache und das Framework haben sollten. Ebenso ist es wichtig zu wissen, ob man Dinge wie Biometrische Daten (FaceID, FingerPrint, etc...) benötigt. Auch hier gibt es unterschiede beim Programmieren.

Die Arten der Apps

Native Apps

Unter nativen Apps versteht man, Mobile-Anwendungen, die speziell für ein bestimmtes Betriebssystem entwickelt werden Jede Plattform (iOS, Android) hat ihre eigenen Programmiersprachen. Während für den Apple-Stack Swift/Objective-C verwendet wird , so sieht es bei Android mit Java wieder ganz anders aus. Beim Windows Phone wiederum wird vor allem mit C++ und JavaScript gearbeitet. Bei Nativen Apps muss somit für jedes Betriebssystem die App in der jeweiligen Programmiersprache erarbeitet werden. Native Apps zeichnen sich auch dadurch aus, dass diese keine Internetverbindung brauchen, um funktioinsfähig zu sein.

Vorteile: Die Native Programmierung hat den Vorteil, dass diese zu 100% den Funktionalitäten der Geräte angepasst sind, daher können alle Features der Smartphones genutzt werden. Der Zugang zu den Hardware Eigenschaften (Kamera, GPS, etc…) ist bei der Native App ebenfalls sehr einfach.

Nachteile: Eine App pro Betriebssystem, das bedeutet möchte man iOS und Android abdecken, so müssen 2 Apps programmiert werden. Das kostet Zeit bei der Entwicklung, Zeit bei der Wartung und Schlussendlich jede Menge Geld.

Web Apps

Die Art der Web App Entwicklung ist darauf ausgerichtet, die App von jedem Gerät und Browser zugänglich zu machen. Die App wird hier im Gegensatz zur nativen Programmierung wird hier eine App für alle Betriebsysteme entwickelt. Die Web App macht sich HTML und CSS zu nutze und wird in einem Webbrowser durch eine URL ausgeführt. Nach dem Öffnen der App passt sich diese dann an das jeweilige Gerät an. Durch diese Umsetzung ist es ebenfalls nicht erforderlich die App auf dem Smartphone zu installieren. Deshalb sind diese meist auch nicht im App Store

vertreten. Auf Apple wäre dies via Safari oder anderen heruntergeladenen Browsern möglich. Auf Android wäre es zum Beispiel mit Google Chrome verwendbar.

Vorteile: Eine App für alle Geräte, das spart Zeit und Wartungsaufwand, sowie viel gespartes Geld am Ende. Es wird kein App Store für den Betrieb benötigt, man ist also nicht auf die Nutzungsbedingungen der App Stores beschränkt

Nachteile: Der größte Nachteil ist der schlechte bzw. fehlende Hardware Zugriff auf den Geräten. Ein ebenso nicht unwesentliches Kontra, ist dass die App nur mittels Internet funktioniert.

Fazit: Daher sollte vorab bedacht werden, wo und wie die App in der Verwendung sein wird.

Cross-Plattform (Hybride) Apps

Die Hybride Programmierung kombiniert die native und die Web App miteinander. Es wird hauptsächlich mit Web-Sprachen programmiert, dennoch wird der Hardware Zugriff nicht eingeschränkt. Cross-Platform Apps werden via App Store installiert.

Vorteile: Durch die Kombination hat man bei der Hybriden App ein besseres Erlebnis als bei den Web Apps

Nachteile: Die Unterstützung ist nicht zu 100% die Selbe, wie bei nativer Programmierung, da neue Funktionen des öftern etwas länger auf sich warten lassen. Einige Elmente müssen Seperat programmiert werden, da sowohl iOS als auch Android bestimme Vorgaben bei einigen Elementen haben, die mittels Cross-Platform bisher noch nicht gelöst werden konnten. Daher muss an manchen Stellen ebenso wie bei Nativen Apps doppelt programmiert werden. Dennoch hält sich der Aufwand meist in Grenzen.

Fazit: Aus der technischen Sicht, würde derzeit Cross-Platform am meisten Sinn machen, da man die vollen Funktionen der Geräte nutzen kann, ohne dass man eine App für jedes mobile Betriebsystem machen muss. Wenn man finanzielle Aspekte hinzuzieht, würden Web-Apps durchaus auch Sinn machen, da man somit die 30% Abgaben auf In-Game-Köufe nicht bezahlen muss, da die App nicht über den App Store vertrieben werden muss. Allerdings, kann man bei den WebApps oft nicht so gut oder teilweise erst später den vollen Funktionsumfang der Hardware Features der Smartphones nutzen.

Welche Sprachen verwendet werden

Swift

Swift ist eine neuere Alternative zu Objective-C von Apple. Zwar basiert Swift auf Objective-C, dennoch soll Swift um einiges unkomplizierter und komfortabler sein. Diese Programmiersprache ist vor allem für iOS, macOS, watchOS und tvOS Apps zu benutzen, da diese logischerweise darauf ausgelegt ist. Swift ist eine intuitive und gut lesbare Sprache im Bereich der Mobile-Applications

Java

Java ist eine Objektorientiere Programmiersprache. Java ist eine durchdachte und eher leichtgewichtige Sprache, die bewusst auf einige Funktionien, die C++ bietet, verzichtet. Die Syntax von Java ist an C und C++ angelehnt. Durch die Ähnlichkeit zu C und C++ fällt ein Umstieg auf Java leicht.

C++

C++ ist eine von der ISO genormte Erweiterung der Programmiersprache C

HTML 5

TypeScript

Frontend Kompaktlösungen

Appcelerator

Erklärung: Appcelerator ist ein Komplettpaket im Bereich der Mobilen Programmierung, denn es bietet die Möglichkeit eine App mit dem App-Designer zu bauen, sowie man ebenfalls ein Dashboard inkludiert hat, das mit einigen Statistiken zur App glänzen kann. Es wird außerdem die Möglichkeit geboten die App im Cross-Platform stil zu programmieren. Dies wird mithilfe von JavaScript umgesetzt. Für viele ebenso relevant ist der inkludierte API Builder, der sicherlich einiges an Zeit sparen kann. Einstellungen zu Push-Benachrichtigungen sind auch ein angepriesenes Feature.

Preis: Die Studio IDE und der API Builder sind gratis,. Für den App Designer und die API Calls sowie die App Preview muss man 99$ pro Monat bezahlen. Ebenso besteht die möglichkeit noch mehr zu kaufen, dies muss man sich allerdings selbst zusammenbauen und dementsprechend variiert der Preis. Hier besteht zum Beispiel die Möglichkeit noch eine Crash detection und Performance Analysen zu bekommen, sowie auch automatiertes Testen zu benutzen. Als Extra werden noch Cloud Kapazitäten geboten, die mit 15$ / Monat anfangen.

RhoMobile

Erklärung: Rhomobile Suite ist ein Software Stack für App-Entwickler, der unter anderem die Möglichkeit bietet mit Ruby zu programmieren, was den Focus auf die Einfachheit und Produktivität lenkt. Es wir auf Cross-Platform Entwicklung gesetzt und zusätzlich ist es auch möglich HTML/CSS/JS zu verwenden. Programmiert wird mittels RhoStudio Extension in Eclipse. Der Sinn von RhoMobile besteht laut Hersteller darin, dass Firmen sichere, aber dennoch dem Customer-Standart entsprechende Apps progammieren können.

Preis: Das Basis App Framework (Rhodes, RhoStudio, RhoElements) ist gratis. Gegen Bezahlung erhält man besseren Support sowie einige extra Features wie das Lesen von Barcodes oder automatische Daten verschlüsselung. Die Preise sind auf Anfrage.

MoSync

Erklärung: Ist ein gratis open-source Software Development Kit. Mit MoSync greift man ebenfalls auf C++, HTML5 und JavaScript zurück. MoSync ist ebenfalls mittels Eclipse verwendbar. Einer der Vorteile von MoSync ist, dass man sicher und schnell Files in der Cloud mit anderen Usern (sogar Personen die keinen MoSync-Account besitzen) teilen kann. Mittels der Platform ist es möglich, dass man überall und jederzeit daran Arbeiten kann. Ebenso soll die Datensicherung und Wiederherstellung sehr gut funktionieren.

Preis: Open Source SDK. Daher keine Kosten.

Sencha Ext JS

Erklärung: Ist eine Komplettlösung mit App-Baukasten der durch Drag and Drop einiges an Zeit beim Programmieren spart Ebenfalls ist es möglich mit Sencha Test zusätzlich zu Testen, hierbei geht es um Unit und End-To-End Tests. Es besteht die Option Statistiken und Heatmaps zu verwenden um Monitoring und Datenauswertung zu machen.

Preis: Ab 1800€ / Jahr für je einen Entwickler

Fazit: Die meisten oben genannten Lösungen sind kostenpflichtig, dafür bekommt man wirklich etwas geboten, das durchaus sehr viel Zeit und Ressourcen spart. Wenn man eine App schnell auf den Markt bringen will, so sind diese Lösungen sicherlich von Vorteil, da sie Arbeit abnehmen. Ebenso ist vermutlich auf lange Sicht auch eine Kostenreduktion bei den Mitarbeitern ein positiver wirtschaftlicher

Aspekt. Von der technischen Sicht, kriegt man einige Hilfestellungen, die vor allem den Erstellungs Prozess der App verkürzen, aber auch das Überwachen und Testen, sowie einige Analysen anbieten, was für kommerzielle Programmierung sicherlich einen starken Vorteil bringt.

Im Diplomprojekt wurde von so einer Lösung abgesehen, da es für zu teuer gewesen wäre und die Features bis auf die App-Baukasten und das automatisierte Testen, für das Projekt im aktuellen Stadium nicht relevant gewesen wären. Ebenso hätten es vermutlich zu viel Arbeitszeit gespart, da die App zu schnell Fertig geworden wäre.

Frontend Frameworks

Bei der Auswahl bei den Frameworks gibt es entscheidende Kritierien, die natürlich bei jeder App unterschiedlich sind. Daher ist eine allgemeine Aussage schwer zu treffen. Für diese Untersuchung gibt es folgende wichtige Kriterien. Alle Frontend Frameworks müssen schon etwas länger existieren und sollten auch in naher Zukunft nicht ohne Weiterführung und Support auskommen müssen. Aufgrund dieser zwei Punkte ist Ionic mit Vue aus der möglichen Auswahl rausgefallen, da sich dieses derzeit noch in einer Betaphase befindet.

Weitere Kriterien:

Übersetzung

Kann man in dem Framework eine Internationalization umsetzen? Im Jahr 2021 sollten Apps in mehreren Sprachen verfügbar sein. Auch hier wird unterschieden, wie einfach sich eine Internationalisierung umsetzen lässt.

Anpassbares Design (während der Runtime)

Wie leicht ist es Designs umzusetzen und vor allem lässt sich das Design während der Nutzung ändern.

Hardwarezugriff

Viele Apps benötigen Zugriff auf die Kamera, auf Biometrische Sensoren und auch auf andere mögliche Funktionen der Smartphones. Hier wird unterschieden, wie einfach das Framework solche Schnittstellen zulässt.

Support

Es ist wichtig, dass regelmäßige Updates erfolgen, um die App auch zukunftssicher

machen zu können. Regelmäßige Updates sind hierfür wichtig, allerdings ist der Abstand der Updates ebenfalls subjektiv zu werten, da für viele Entwickler zu häufige updates für mehraufwand sorgen können für andere dennoch kein Problem darstellen.

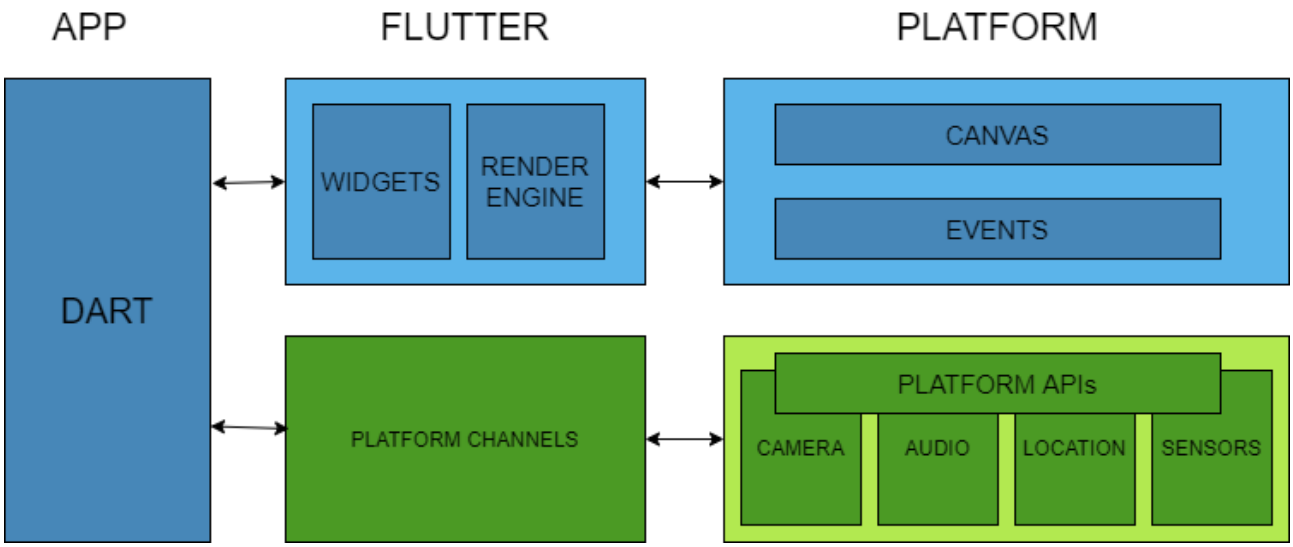
Dokumentation

Gibt es eine gute Dokumentation? Wie ausgereift und verständlich sind die Dokumentationen?

Flutter

Disclaimer: Da im Diplomprojekt mit Flutter gearbeitet wurde, ist in diesem Teil auch erworbenes Wissen eingeflossen, deshalb ist die Erklärung / Analyse genauer und auch teils detailierter.

Flutter ist ein Open-Source UI Entwicklungs-Kit. Die zugrundeliegende Programmiersprache ist Dart. Das Framework wird für die Programmierung von Apps verwendet. Das Framework selbst ist mittels C++ geschrieben worden.



Funktionsweise von Flutter [146]

Übersetzung

Die Übersetzung in Flutter ist relativ einfach vor allem sobald man diese aufgesetzt hat. In der laufenden Entwicklung hat man dann für jede Sprache, die man unterstützen will, ein JSON file in dem man die verschiedenen Elemente dann übersetzt. Im Code selbst, werden dann statt Strings einfach die Feld-Namen verwendet, die als Key für die Übersetzung fungieren.

Anpassbares Design

Flutter ermöglicht es während der Runtime die Designs zu verändern. Hier geht es vor allem um das Ändern der Farben während dem Benutzen der App. Ebenso können natürlich alle Widgets während der Runtime geändert werden, dazu muss man nicht viel machen, da dies mittels Navigator funktioniert.

Hardwarezugriff

Da Flutter sehr eng mit der Hardware kommuniziert, ist der Hardwarezugriff einfach. Für diese Use Cases gibt es bereits fertige Packages, die eingebaut werden können.

Support

Flutter versucht ungefährt jedes Quartal ein stable Update zu releasen. Erst im März 2021 kam Flutter 2.0 auf den Markt. Updates sind einfach mit dem Befehl "flutter upgrade" durchzuführen.

Dokumentation

Obwohl Flutter noch (im Vergleich zu Anderen) relativ "neu" ist, wird es sehr stark von Google unterstützt und es gibt eine durchaus beachtliche Dokumentation. Ebenfalls gibt es viele Kurzvideos zu bestimmten Widgets oder Funktionen, die

einem die Arbeit beim einlesen/einarbeiten erleichtern. Die Flutter Dokumentation ist vor allem sehr organisiert und einfach zu lesen.

Bonus

Für Flutter gibt es unzähloge fertige Packages, die einem das Leben als Entwickler erleichter, da man nicht alles von Grund auf neu machen muss. Für viele Use Cases gibt es bereits fertige Umsetzungen, die in die App eingebaut werden können. Ein Beispiel dafür wären Barcode Scanner. Hierfür ist es lediglich notwendig auf pub.dev danach zu suchen und eine dependency zu setzen. Dies ist, wie im unten stehenden Bild ersichtlich, alles detailiert unter dem Reiter "Installing" nachzulesen. Das verwenden von Packages ist simpel, die einzige Hürde ist es packages zu finden, die auch noch supported werden und laufend auf updates auch reagieren.

Published Jan 5, 2021 • Latest: 1.0.2 / Prerelease: 2.0.0-nullsafety.0

FLUTTER | ANDROID | IOS

302

Readme | Changelog | Example | Installing | Versions | Scores

Use this package as a library

1. Depend on it

Add this to your package's pubspec.yaml file:

```
dependencies:
  flutter_barcode_scanner: ^1.0.2
```

2. Install it

You can install packages from the command line:

with Flutter:

```
$ flutter pub get
```

Alternatively, your editor might support `flutter pub get`. Check the docs for your editor to learn more.

3. Import it

Now in your Dart code, you can use:

```
import 'package:flutter_barcode_scanner/flutter_barcode_scanner.dart';
```

Gut zu wissen

In Flutter dreht sich alles um Widgets. Alles was in der App dann sichtbar ist, ist ein Widget. Ein Widget kann wiederum in ein anderen Widget gepackt werden. Was ist

nun also ein Widget? Es ist die Komponente, die Logik, Interaktion und Darstellung bündelt.

Zusammengefasst

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. Die Community wächst und somit auch der Support unter den Entwicklern.

Code Snippets

Beispiel eines Home-Screens in Flutter

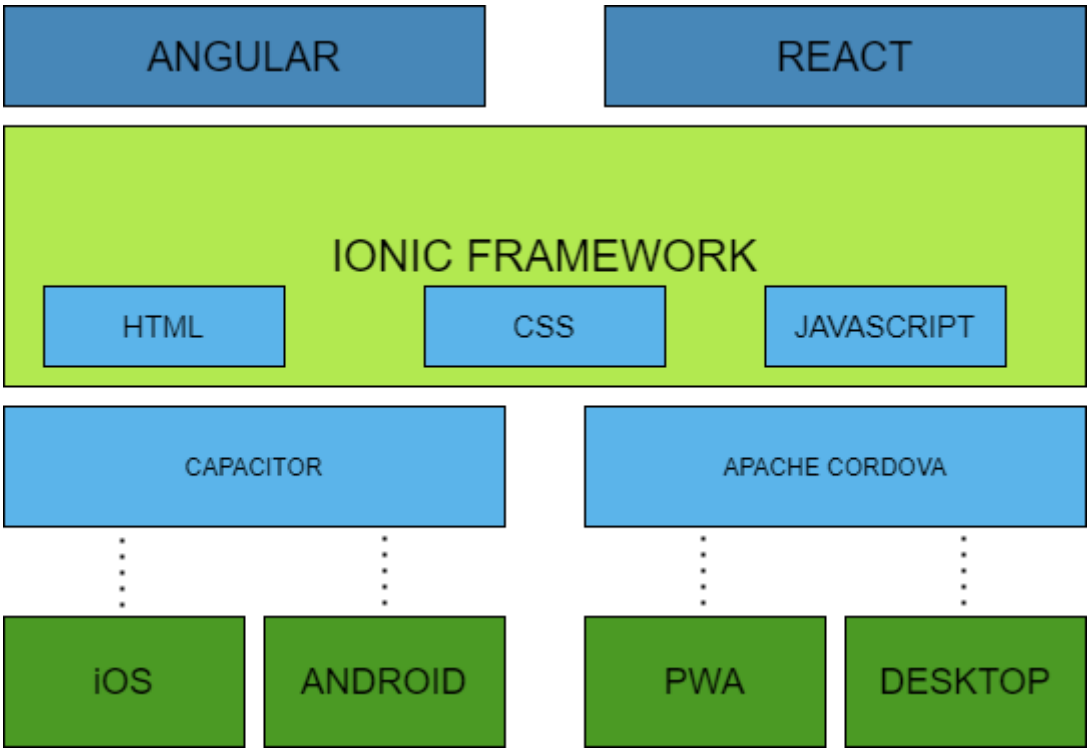
```
class _StartPageState extends State<StartPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('NoQuePOS'),
        centerTitle: true,
        actions: [
          IconButton(
            icon: Icon(
              Icons.shopping_cart,
              color: Colors.white,
            ),
            onPressed: () {
              _navigateToCartPage(context);
            },
          )
        ],
      ),
      bottomNavigationBar: NavigationBar(),
      drawer: DrawerMenu(),
      body: Container(
        decoration: BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/images/start.png'),
            fit: BoxFit.cover,
          ),
        ),
        child: Align(
          alignment: Alignment.topCenter,
          child: RaisedButton(
            onPressed: () {
              _navigateToScanPage(context);
            },
            color: Colors.blue,
            child: Text(
              translate('start_page.enter_button'),
              style: TextStyle(
                fontSize: 20,
                color: Colors.white,
              ),
            ),
          ),
        ),
      ),
    );
  }
```

Beispiel einer Appbar in Flutter

```
Scaffold(
  appBar: AppBar(
    iconTheme: IconThemeData(
      color: Color(0xffFF0000), //OR Colors.red or whatever you want
    ),
    title: Text("Title"),
    backgroundColor: Color(0xffFAFAFA),
  ),
)
```

Ionic/Angular

Ist ein Open-Source Webframework, dass vor allem für Cross-Platform und Progressive Webs Apps geeignet ist. Ionic mit Angular basiert, wie Angular auf TypeScript.



Architektur Ionic [147]

Übersetzung

Die Übersetzung ist mittels rxweb Package möglich, allerdings ein wenig umständlicher in der Handhabung, als andere Frameworks. Dennoch gibt es für die

Internatinalisierung bei Angular eine gute Dokumentation, die eine Step-by-Step Anleitung bereitstellt.

Anpassbares Design

Das Anpassen von Designs während der Runtime ist prinzipiell möglich, aber im Vergleich zu anderen Frameowrks eher unhandlich.

Hardwarezugriff

Der Hardwarezugriff bei Angular ist sehr gut und auch schon ausgereift. Im Ionic Framework gibt es das cordova-plugin-camera Plugin, welches die Schnittstelle zur Kamera bereitstellt.

Support

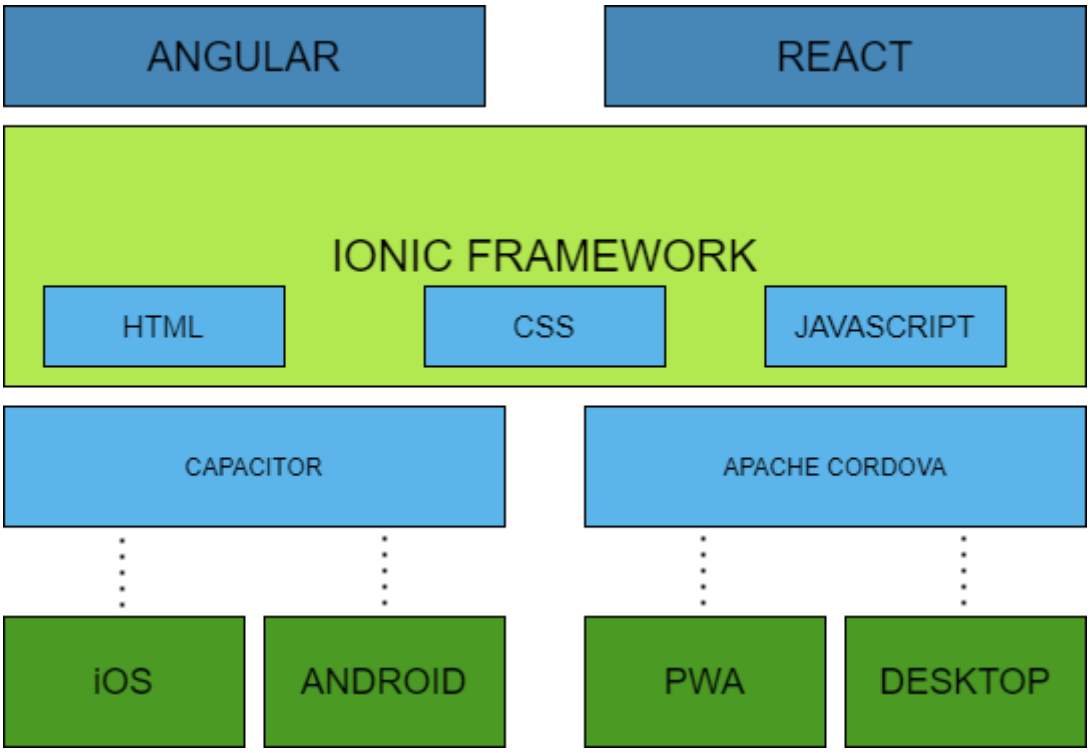
Major Releases werden alle sechs Monate veröffentlicht. Daneben gibt es noch Minor Releases, die sich mit API changes befassen, die keinen großen Eingriff vornehmen. Diese werden ungefähr ein Mal pro Monat released.

Dokumentation

Ionic hat eine übersichtliche und auch weitreichende Dokumentation, die ebenfalle jedes mal nach Major updates auch angepasst wird und somit auch die User Experience weiter verbessert wird.

Ionic/React

Ist ein Open-Source UI und Native API Projekt, dass vor allem für Cross-Platform und Progressive Webs Apps geeignet ist. Ionic mit React basiert, wie React auf JavaScript.



Architektur Ionic [148]

Übersetzung

Die Übersetzung in Ionic/React ist relativ einfach vor allem sobald man diese aufgesetzt hat. In der laufenden Entwicklung hat man dann für jede Sprache, die man unterstützen will, ein JSON file in dem man die verschiedenen Elemente dann übersetzt. Im Code selbst, werden dann statt Strings einfach die Feld-Namen verwendet, die als Key für die Übersetzung fungieren. Als zusatz ist es ebenso möglich, direkt mit dem String zu Arbeiten. Im JSON File wird also kein Key verwendet, sonder direkt der Text und mittels ":" dann die Übersetzung dahinter gemacht. Ermöglicht wird dies durch i18next.

Anpassbares Design

Ionic/React ermöglicht es während der Runtime die Designs zu verändern. Hierfür kann CSS oder ein bereits vorhandenes Theme Switcher Package verwendet werden. Durch die gute Dokumentation, stellt auch das Ändern des Designs während dem Benutzer der App kein Problem dar.

Hardwarezugriff

Im vergleich zu Flutter, fällt hier der Kamera Zugriff etwas schwerer aus, dennoch ist mittels Ionic/React der Hardwarezugriff generell auch relativ einfach möglich.

Support

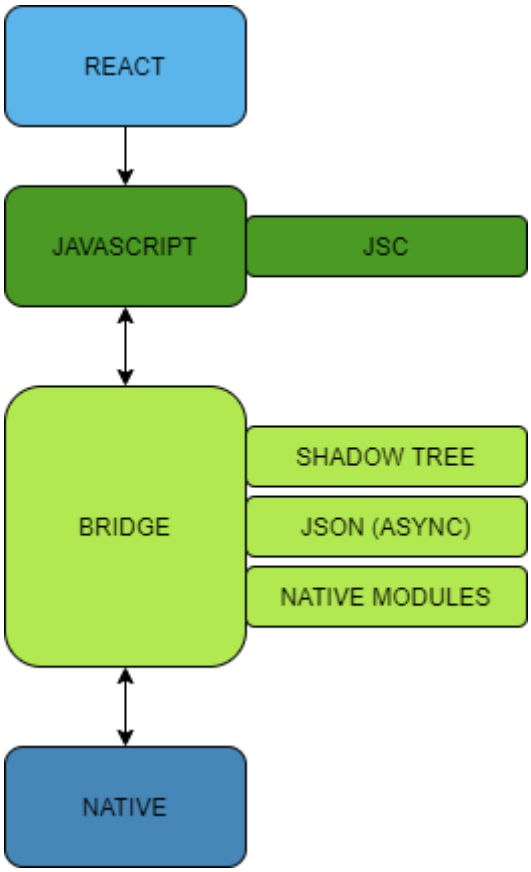
Major Releases werden alle sechs Monate veröffentlicht. Daneben gibt es noch Minor Releases, die sich mit API changes befassen, die keinen großen Eingriff vornehmen. Diese werden ungefähr ein Mal pro Monat released.

Dokumentation

Ionic hat eine übersichtliche und auch weitreichende Dokumentation, die ebenfalle jedes mal nach Major updates auch angepasst wird und somit auch die User Experience weiter verbessert wird.

React Native

React Native ist ein Open-Source-Mobile Application Framework, das speziell für die Entwicklung von Apps für Android, Android TV, iOS, macOS, Web und Windows geeignet ist. Das Framework wurde von Facebook entwickelt.



Architektur React Native [149]

Übersetzung

Die Übersetzung in React Native ist relativ einfach vor allem sobald man diese aufgesetzt hat. In der laufenden Entwicklung hat man dann für jede Sprache, die man unterstützen will, ein JSON file in dem man die verschiedenen Elemente dann übersetzt. Im Code selbst, werden dann statt Strings einfach die Feld-Namen verwendet, die als Key für die Übersetzung fungieren. Als zusatz ist es ebenso möglich, direkt mit dem String zu Arbeiten. Im JSON File wird also kein Key verwendet, sonder direkt der Text und mittels ":" dann die Übersetzung dahinter gemacht. Ermöglicht wird dies durch i18next.

Anpassbares Design

React Native ermöglicht es während der Runtime die Designs zu verändern. Hierfür kann CSS oder ein bereits vorhandenes Theme Switcher Package verwendet

werden. Durch die gute Dokumentation, stellt auch das Ändern des Designs während dem Benutzer der App kein Problem dar.

Hardwarezugriff

Dadurch, dass es Native Framework ist, fällt die Einbindung der Hardware relativ einfach und ist auch leicht umsetzbar.

Support

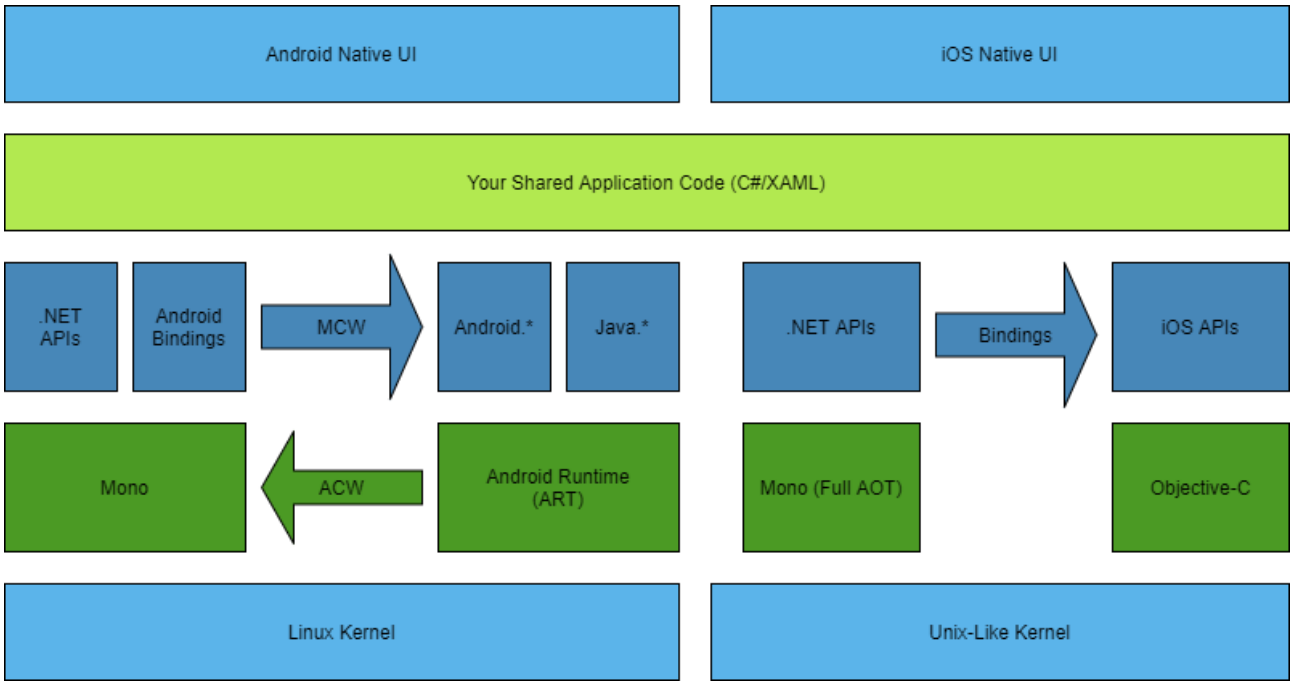
Es werden regelmäßig jedes Monat neue Updates released. Die Updates werden über das GitHub Repository ausgerollt. Bevor das update eingebaut wird, gibt es eine Testphase für 1 Monat, wo reviewt werden kann. Ebenso können sich die Entwickler in der Phase mit den Änderungen vertraut machen.

Dokumentation

Es gibt eine generelle Dokumentation, dennoch ist diese bei weitem nicht so ausgereift, wie bei anderen Frameworks

Xamarin

Xamarin ist eine Open-Source Platform zum Erstellen von leistungsfähigen Anwendungen für den Mobilen Entwicklungsbereich, wie auch für Windows. Xamarin verwendet C# als Programmiersprache und hat Xamarin.Android, Xamarin.iOS und eine Xamarin.Essential Libraries, die für Native, aber auch für Cross-Platform Apps verwendet werden können.



Architektur Xamarin [150] https://docs.microsoft.com/en-us/xamarin/xamarin-forms/release-notes/

Übersetzung

Die Übersetzung in Xamarin erfolgft mittels RESX Files. Im Vergleich zu Flutter und React ist, hier allerdings etwas mehr Aufwand zu betreiben, dennoch gibt es dazu eine detailierte Dokumentation auf docs.microsoft.com. Ebenfalls muss man iOS und Android etwas seperat behandeln, da es nicht einheitlich ist und man somit mehr Aufwand erfordert.

Anpassbares Design

Es ist möglich die Design während der Runtime zu ändern

Hardwarezugriff

Der Hardwarezugriff ist etwas schwieriger, da man für iOS und Android jeweils 2 unterschiedliche Codebases benötigt. Es ist grundseätztlich möglich sowohl auf Apple Geräten, als auch bei Android auf die Hardware zuzugreifen, dennoch ist es mit mehr Aufwand verbunden, als bei anderen Frameworks. Hierfür gibt es

Libraries aus Xamarin.Essentials.

Support

Xamarin hat regelmäßige Updates, die ebenfalls auch immer in der Roadmap angepriesen werden, man kann sich also schon vorab darauf einstellen, was in der Zukunft auf einen zukommt. Ebenfalls steht auch dabei, wie lange diese Version supported wird.

Dokumentation

Xamarin hat eine vollständige Dokumentation samt intuitiver Navigation, die sehr weitreichend ist. Vom Anfänger bis zum Profi ist alles dabei.

Code Snippets:

```
private FahrzeugContext GetContext()
        {
            var db = new FahrzeugContext();
            db.Database.EnsureDeleted();
            db.Database.EnsureCreated();
            db.Import("data.sql");
            return db;
        }
```

===

[146] medium.com:Cross-platform mobile apps development in 2021: Xamarin vs React Native vs Flutter vs Kotlin Multiplatform, https://medium.com/xorum-io/cross-platform-mobile-apps-development-in-2021-xamarin-vs-react-native-vs-flutter-vs-kotlin-ca8ea1f5a3e0 abgerufen am 06.04.2021

[147] ICT-BZ.ch:Ionic Architektur, https://m335.ict-bz.ch/tag-1/ionic-architektur abgerufen am 06.04.2021

[148] ICT-BZ.ch:Ionic Architektur, https://m335.ict-bz.ch/tag-1/ionic-architektur abgerufen am 06.04.2021

[149] formidable.com : The New React Native Architecture Explained: Part Four, https://formidable.com/blog/2019/lean-core-part-4/ abgerufen am 06.04.2021

[150] docs.microsoft.com : What is Xamarin? , https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin#how-xamarin-works abgerufen am 06.04.2021

Auswertung / Vergleich

Für jedes Kriterium, das für die Bewertung der Frameworks heranzgeogen wurde, können maximal 10 Punkte erreicht werden.

	Übersetzung	Anpassbares Design	Hardwarezugriff	Support	Dokumentation
Flutter	8	8	8	8	8
Ionic Angular	9	5	8	8	8
Ionic React	9	8	7	8	8
React Native	9	8	8	8	6
Xamarin	6	7	5	9	9

Table 12. Auswertungs Tabelle

Wie man in der Tabelle oben sehen kann, liegen die Frameworks alle sehr nah bei einander. Durch diese Tabelle wird die Annahme am Beginn der Arbeit nochmals deutlich. Das perfekte Framework ist immer abhängig von den Anforderungen. Allgemein kann man sagen, dass fast alle sehr gut Dokumentiert sind und ebenfalls laufen Updates bekommen. Für das Diplomprojekt wurde Flutter verwendet, da das Team etwas komplett neues Lernen wollte und Flutter auch sehr ansprechend ist.

Schlussfolgerung der Arbeit

Es gibt zahlreiche Möglichkeiten im Frontend Bereich eine App zu entwickeln. Von Kompaktlösungen, die Geld kosten, bis zu Open Source Frameworks ist alles enthalten. Wichtig ist, dass vorab Kritierien festgelegt werden, die das jeweilige Framework erfüllen muss, um die Applikation umzusetzen. "Das Framework" gibt es hierbei nicht, denn jeder hat andere Anforderungen und Angewohnheiten, die eine Auswahl am Ende dann festlegen, denn die Frameworks sind im Großen und Ganzen alle sehr gut. Flutter hebt sich mit seiner besonderen Art dennoch ein wenig hervor, da es mit den Widgets eine doch sehr Bildhafte und einfache Programmierung ist und dadurch auch relativ schnell zu lernen ist. Die Syntax Highlighter vereinfachen die Lesbarkeit und auch die Fehlerbehebung sehr. Ebenfalls scheint die Zukunft von Flutter als relativ sicher.

Appendix A: Kooperationsvertrag

KOOPERATIONSVEREINBARUNG

zwischen
1. SynCore Systems EDV Dienstleistungen GmbH
(Name des Unternehmens)

vertreten durch Jan Niederreiter
(Name des Vertreters)
(in der Folge „die Projektpartnerin)

und
2. Alexander Wiener, Lucas Walter, Paul Maurovich, Dennis Sima, Florian Freimüller
(in der Folge „das Projektteam")

PRÄAMBEL

Das Projektteam und die Projektpartnerin beabsichtigen gemäß der Verordnung über die abschließenden Prüfungen in den berufsbildenden mittleren und höheren Schulen, BGB. II Nr. 70/2000 vom 24.2.2000, die Planung und Durchführung eines Diplomprojekts: *Konzeption und Entwicklung eines Prototypen für die Selfcheckout Kassenlösungen der Zukunft*. Durch die Zusammenarbeit soll insbesondere den Mitgliedern des Projektteams die Möglichkeit eingeräumt werden, im Rahmen ihrer schulischen Ausbildung bei der Erstellung der Diplomarbeit an die Verhältnisse im technischen Berufsleben herangeführt zu werden, um dabei die in der Schule erworbenen theoretischen Kenntnisse und Fähigkeiten in der Praxis anzuwenden bzw. zu erweitern.

§ 1
Gegenstand

Gegenstand ist die Erstellung von Arbeitsergebnissen zum Thema der Diplomarbeit. Dieses Thema ist der Projektbeschreibung und dem Pflichtenheft zu entnehmen, welches vor Projektstart angefertigt wird. Die Projektpartnerin wird jedoch darauf hingewiesen, dass es sich um ein Projekt im Zusammenhang mit der schulischen Ausbildung handelt und daher jede Haftung des Projektteams ausgeschlossen ist.

§ 2
Laufzeit

Die vorliegende Kooperation tritt am 10.06.2020 in Kraft und wird bis zum Ende der Reife- und Diplomprüfung der Höheren Technischen Bildungs- und Versuchsanstalt Spengergasse abgeschlossen.

§ 3
Rechte und Pflichten des Projektteams

Die Mitglieder des Projektteams haben das Recht, die Räumlichkeiten der Projektpartnerin samt Infrastruktur und EDV-Infrastruktur im für die Projektabwicklung erforderlichen Ausmaß während der Geschäftszeiten der SynCore Systems und nach vorheriger Absprache durch die Projektpartnerin mitzubenutzen. Das Projektteam verpflichtet sich, die im Gegenstand genannten Arbeiten sorgfältig und unter möglichster Schonung der Interessen der Projektpartnerin durchzuführen. Das Projektteam unterliegt der Betriebsordnung der Projektpartnerin. Das Projektteam verpflichtet sich zur Geheimhaltung aller ihm zur Kenntnis gelangenden Geschäfts- und Betriebsgeheimnisse.

§ 4
Rechte und Pflichten der Projektpartnerin

Die Projektpartnerin verpflichtet sich, dem Projektteam beratend zur Verfügung zu stehen und alles zu unterlassen, was der Vollendung des Projekts entgegensteht. Die Projektpartnerin wird dem Projektteam, die für das Projekt notwendigen Hilfsmittel zur Verfügung stellen.

Sollte das Projektteam im Rahmen dieser Kooperationsvereinbarung eine Erfindung machen, die nach dem Gebrauchsmustergesetz bzw. dem Patentgesetz (PatG) schützbar ist, gilt diese Erfindung als Diensterfindung im Sinne des PatG und die §§ 6-19 PatG (in der geltenden Fassung) entsprechend. Das Projektteam verpflichtet sich, die Projektpartnerin von einer im Rahmen der Kooperationsvereinbarung gemachten Erfindung unverzüglich in Kenntnis zu setzen. Die Projektpartnerin hat daraufhin das Recht, binnen vier Wochen ab dieser Bekanntgabe zu erklären, dass sie/er das Patentrecht für sich beansprucht. In diesem Fall steht dem Projektteam eine entsprechende Vergütung nach den einschlägigen Bestimmungen des PatG (in der geltenden Fassung) zu.

Sollte das Projektteam im Rahmen dieser Kooperationsvereinbarung ein Werk schaffen, dem Schutz im Sinne des Urheberrechtsgesetzes zukommt, verpflichtet es sich, die Projektpartnerin davon unverzüglich zu informieren. Die Projektpartnerin hat daraufhin die Möglichkeit, binnen vier Wochen ab dieser Bekanntgabe, mit dem Projektteam einen Werknutzungsvertrag abzuschließen.

§ 5
Einsicht und Präsentation

Da die Tätigkeit des Projektteams auch Inhalt bzw. Grundlage der an der Höheren Technischen Bildungs- und Versuchsanstalt Spengergasse zu erstellenden Diplomarbeit ist, berechtigt die Projektpartnerin die zuständigen Organe des Bundes zur Einsicht und Kontrolle, um die in der Verordnung über die abschließenden Prüfungen an den berufsbildenden mittleren und höheren Schulen genannten Aufgaben zu erfüllen.

Das Projektteam ist auch berechtigt, Ergebnisse der Diplomarbeit bei der mündlichen Reife- und Diplomprüfung zu präsentieren. Die zuständigen Organe des Bundes sind ihrerseits wiederum gegenüber jedermann zur Wahrung der Geschäfts- und Betriebsgeheimnisse der Projektpartnerin verpflichtet.

Wien, am 10.6.2020

Wien, am 10.6.2020

SynCore Systems
EDV Dienstleistungen GmbH
Nordbahnstraße 36/1/Top 2.1
1020 Wien
office@syncore.at

Projektpartnerin

Projektteam

Appendix B: User Stories

Scan Buy Go User Stories

Alexander Wiener
Florian Freimüller
Paul Maurovich
Lucas Walter
Dennis Sima

Legende zur Priority

1: Diese User Stories sind systemrelevant. Das bedeutet, diese sind notwendig, damit der Prozess eines Einkaufs durchgeführt werden kann. Inkludiert hierfür sind unter anderem das Registrieren, Login, Beginn des Einkaufes (Checkin in Shop), Produkte, Warenkorb, Bezahlung

2: Diese User Stories sind nicht systemrelevant. Allerdings bieten diese einen Komfort für den Benutzer und vereinfachen und erweitern die Experience

3: Diese User Stories sind nicht systemrelevant und werden nur bei genügend Zeit miteingearbeitet

Legende zu Estimate

Die Schätzungen sind in Storypoints.

Abbildungsverzeichnis

Abbildung 1: Login_Register_Frontend 27
Abbildung 2: Login_Register_Backend 27
Abbildung 3: Password_Reset_Frontend 28
Abbildung 4: Password_Reset_Backend 29
Abbildung 5: Shop_Assignment_Frontend 30
Abbildung 6: Shop_Assignment_Backend 31
Abbildung 7: Receipt_Frontend 32
Abbildung 8: Receipt_Backend_1 33
Abbildung 9: Receipt_Backend_2 34
Abbildung 10: Shopping_Cart_Frontend_1 35
Abbildung 11: Shopping_Cart_Frontend_2 36
Abbildung 12: Shopping_Cart_Backend 37

Scan Buy Go

Title: Login	**Priority:** 1	**Estimate:** 20
Als Kunde kann ich mich nach der einmaligen Registrierung mit meiner E-Mail und meinem Passwort einloggen, sodass ich als eingeloggter User alle Funktionalitäten der App nutzen kann.		
Acceptance Criteria **Gegeben ist** dass der User auf der Login-Seite ist **Wenn** er alle benötigten Inputfelder ausgefüllt hat und den Knopf "Anmelden" drückt **Dann** wird er auf die Start-Seite weitergeleitet		
Prozesse **Frontend:** Abbildung 1: Login_Register_Frontend **Backend**: Abbildung 2: Login_Register_Backend		

Title: Registrierung	**Priority:** 2	**Estimate:** 13
Als Kunde kann ich mich mit meinen persönlichen Daten* registrieren, sodass ich einen Account besitze. * Vorname, Zuname, E-Mail, Geburtsdatum, Straße, Hausnummer, …		
Acceptance Criteria **Gegeben ist,** dass der User auf der Registrierungsseite ist **Wenn** er alle benötigten Inputfelder ausgefüllt hat und den Knopf "Registrieren" drückt **Dann** wird er auf die Emailbestätigungsseite weitergeleitet		
Prozesse: **Frontend:** Abbildung 1: Login_Register_Frontend **Backend**: Abbildung 2: Login_Register_Backend		

Scan Buy Go

Title: Gast-Modus	**Priority:** 3	**Estimate:** 5
Als Kunde kann ich Registrierung und Login überspringen, sodass ich als Gast ohne Account einkaufen kann.		
Acceptance Criteria **Gegeben ist,** dass der User auf der Login-Seite ist **Wenn** er auf den Knopf "Gast" drückt **Dann** wird er auf die Start-Seite weitergeleitet		
Prozesse: **Frontend:** Abbildung 1: Login_Register_Frontend **Backend**: Abbildung 2: Login_Register_Backend		

Title: Passwort vergessen	**Priority:** 2	**Estimate:** 13
Als Kunde kann ich mein Passwort mit meiner E-Mail zurücksetzen, sodass ich ein neues Passwort für meinen bestehenden Account festlegen kann.		
Acceptance Criteria **Gegeben ist,** dass der User seine Email bereits bestätigt hat **Und** auf der Passwort zurücksetzen-Seite ist **Wenn** er sein neues Passwort 2 mal ausgefüllt hat und den Knopf "Passwort zurücksetzen" drückt **Dann** wird er auf die Start-Seite weitergeleitet		
Prozesse: **Frontend:** Abbildung 3: Password_Reset_Frontend **Backend**: Abbildung 4: Password_Reset_Backend		

Scan Buy Go

Title: E-Mail bestätigen	**Priority:** 1	**Estimate:** 5
Als Kunde muss ich** nach der Anmeldung einen Code aus einer gesendeten E-Mail eingeben, sodass ich mit meinem Account einkaufen kann. **Nur, wenn eine Anmeldung mit E-Mail und Passwort vorliegt, externe Anbieter sind ausgenommen. (Bei ihnen wurde die E-Mail bereits bestätigt)		
Acceptance Criteria **Gegeben ist,** dass der User sich bereits mit seinen Daten registriert hat **Und** auf der Emailbestätigungsseite ist **Wenn** er seinen Verifizierungscode aus seinem E-Mail Postfach in die Inputfelder eingibt und "Bestätigen" drückt **Dann** wird er auf die Login-Seite weitergeleitet		
Prozesse: **Frontend:** Abbildung 1: Login_Register_Frontend **Backend**: Abbildung 2: Login_Register_Backend		

Title: QR Code scannen	**Priority:** 1	**Estimate:** 8
Als Kunde kann ich den QR Code mit der Kamera scannen, sodass ich im richtigen Store eingeloggt bin und mit dem Einkauf starten kann.		
Acceptance Criteria **Gegeben ist**, dass der User vor einem Shop steht **Und** er in diesem einkaufen will **Wenn** er den QR Code scannt **Dann** befindet er sich auf der Startseite (Im Shop) und kann mit dem Einkauf beginnen		
Prozesse: **Frontend:** Abbildung 5: Shop_Assignment_Frontend **Backend**: Abbildung 6: Shop_Assignment_Backend		

Scan Buy Go

Title: Shop Listen Auswahl	**Priority:** 3	**Estimate:** 5
Als Kunde kann ich einen Store aus der Liste auswählen, sodass ich in diesem Store eingeloggt bin und mit dem Einkauf starten kann.		
Acceptance Criteria **Gegeben ist**, dass der User vor einem Shop steht **Und** er in diesem Store einkaufen will **Wenn** er über die Listenauswahl den Shop auswählt **Dann** wird er in den Shop eingeloggt und kann mit dem Einkauf beginnen		
Prozesse: **Frontend:** Abbildung 5: Shop_Assignment_Frontend **Backend**: Abbildung 6: Shop_Assignment_Backend		

Title: Store Verlassen Manuell	**Priority:** 1	**Estimate:** 2
Als Kunde kann ich auf den Button „Store verlassen“ klicken, sodass der gesamte Warenkorb gelöscht bin, ich nicht mehr in diesem Store eingeloggt bin und wieder die Möglichkeit habe, mich bei einem anderen Store anmelden zu können.		
Acceptance Criteria **Gegeben ist**, dass der User den aktuellen Shop verlassen will **Wenn** er auf "Shop verlassen" drückt **Dann** wird er aus dem Shop abgemeldet und befindet sich dann wieder auf der Startseite der App		

Scan Buy Go

Title: Neuer Store QR inkl. auto abmelden	**Priority:** 3	**Estimate:** 2
Als Kunde kann ich einen QR Code mittels Kamera scannen, sodass ich nicht mehr im letzten Store eingeloggt bin und im neuen Store einkaufen kann.		
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er einen QR Code eines anderen Shops scannt **Wenn** er den neuen QR Code scannt **Dann** wird er gefragt ob er aus dem aktuellen Shop abgemeldet werden soll und im neuen Shop angemeldet werden soll		
Prozesse: **Frontend:** Abbildung 5: Shop_Assignment_Frontend **Backend**: Abbildung 6: Shop_Assignment_Backend		

Title: Produkt hinzufügen/Scan	**Priority:** 2	**Estimate:** 8
Als Kunde kann ich den QR-Code oder Barcode eines Produktes einscannen, sodass ich das Produkt, bevor es meinem virtuellen Warenkorb hinzugefügt wird, nochmals reviewen kann.		
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er ein Produkt aus diesem Shop in der Hand hält **Wenn** er über den Scanbildschirm den Barcode des Produkts einscannt **Dann** wird dieses Produkt seinem Warenkorb hinzugefügt		
Prozesse: **Frontend:** Abbildung 10: Shopping_Cart_Frontend_1 **Backend**: Abbildung 12: Shopping_Cart_Backend		

Scan Buy Go

Title: Produkt hinzufügen/Liste	**Priority:** 2	**Estimate:** 5
Als Kunde kann ich optional durch eine Liste ein Produkt suchen und hinzufügen, sodass ich das Produkt, bevor es meinem virtuellen Warenkorb hinzugefügt wird, nochmals reviewen kann.		
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er ein Produkt aus diesem Shop in der Hand hält **Wenn** er dieses Produkt in der Liste der Produkte dieses Shops sucht und auswählt **Dann** wird dieses Produkt seinem Warenkorb hinzugefügt		
Prozesse: **Frontend:** Abbildung 10: Shopping_Cart_Frontend_1 **Backend**: Abbildung 12: Shopping_Cart_Backend		

Title: Produkt hinzufügen/Kategorien	**Priority:** 2	**Estimate:** 5
Als Kunde kann ich ein Produkt optional durch sämtliche Kategorien finden und hinzufügen, sodass ich das Produkt, bevor es meinem virtuellen Warenkorb hinzugefügt wird, nochmals reviewen kann.		
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er ein Produkt aus diesem Shop in der Hand hält **Wenn** er das Produkt kaufen will **Dann** kann er es mittels Kategorien suchen und dem Warenkorb hinzufügen		
Prozesse: **Frontend:** Abbildung 10: Shopping_Cart_Frontend_1 **Backend**: Abbildung 12: Shopping_Cart_Backend		

Scan Buy Go

Title: Produkt hinzufügen/Stückzahl	**Priority:** 2	**Estimate:** 1
Als Kunde kann ich, nachdem ich das Produkt gescannt, in einer Liste gefunden habe oder durch die Kategorien entdeckt habe, auswählen, welche Stückzahl ich von diesem Produkt kaufen möchte, sodass ich das Produkt nicht mehrmals suchen brauche und die jeweilige Produktanzahl meinem Warenkorb hinzugefügt wird.		
Acceptance Criteria **Gegeben ist**, dass der User ein Produkt hinzugefügt hat **Wenn** er die Anzahl von diesem Produkt ändert **Dann** wird die Anzahl des Produkts im Warenkorb aktualisiert		
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2 **Backend**: Abbildung 12: Shopping_Cart_Backend		

Title: Warenkorb ansehen	**Priority:** 1	**Estimate:** 8
Als Kunde kann ich während des Einkaufens jederzeit meinen virtuellen Warenkorb ansehen, sodass mir alle bereits gescannten Produkte inkl. Bild und Stückzahl angezeigt werden und was die aktuelle Zwischensumme in Euro beträgt.		
Acceptance Criteria **Gegeben ist**, dass der Benutzer in einem Shop angemeldet ist **Wenn** er auf "Warenkorb" tippt **Dann** wird ihm eine Liste an Produkten ausgegeben		
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2		

Scan Buy Go

Title: Warenkorb/Produkt entfernen	**Priority:** 1	**Estimate:** 2
Als Kunde kann ich während des Einkaufens jederzeit in meinem virtuellen Warenkorb Produkte entfernen, sodass diese nicht mehr im Warenkorb vorhanden sind und ich somit das entfernte Produkt nicht bezahlen muss.		
Acceptance Criteria **Gegeben ist**, dass der Benutzer sich auf der Warenkorbseite befindet **Und** er mindestens ein Produkt im Warenkorb hat **Wenn** er ein Produkt aus der Liste wischt **Dann** wird dieses Produkt aus dem Warenkorb entfernt		
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2		

Title: Warenkorb/Stückzahl erhöhen/verringern	**Priority:** 1	**Estimate:** 2
Als Kunde kann ich während des Einkaufens jederzeit in meinem virtuellen Warenkorb die Stückzahl eines bereits eingescannten Produktes erhöhen oder verringern, sodass ich das Produkt im Nachhinein nicht erneut einscannen muss.		
Acceptance Criteria **Gegeben ist**, dass der Benutzer sich auf der Warenkorbseite befindet **Und** er mindestens ein Produkt im Warenkorb hat **Wenn** er die Anzahl des Produkts verändern will **Dann** wird die Produktanzahl im Warenkorb und die Einkaufssumme verändert		
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2		

Scan Buy Go

Title:	Produktdetails nach Produkt hinzufügen	**Priority:**	2	**Estimate:**	3
Als Kunde kann ich, nachdem ich ein Produkt eingescannt, manuell in der Produktliste oder in den Kategorien gefunden habe und bevor ich es in den Warenkorb hinzufüge, die Produktdetails inkl. Bild ansehen, sodass ich mir vorher noch die Produktdetails durchlesen kann.					
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er ein Produkt hinzugefügt hat **Wenn** er mehr über das Produkt erfahren will **Dann** kann er sich die Produktdetails anzeigen lassen					
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2 **Backend**: Abbildung 12: Shopping_Cart_Backend					

Title:	Warenkorb/Produktdetails	**Priority:**	2	**Estimate:**	8
Als Kunde kann ich während des Einkaufens jederzeit in meinem virtuellen Warenkorb von allen hinzugefügten Produkten die Details inkl. Bild anzeigen lassen, sodass ich auch im Nachhinein Inhaltsstoffe etc. durchlesen kann.					
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er ein Produkt im Warenkorb hat **Wenn** er über die Warenkorbseite das Produkt auswählt **Dann** werden ihm Informationen über das Produkt angezeigt					
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2 **Backend**: Abbildung 12: Shopping_Cart_Backend					

Scan Buy Go

Title: Warenkorb löschen	**Priority:** 2	**Estimate:** 1
Als Kunde kann ich während des Einkaufens jederzeit den gesamten Warenkorb löschen, sodass ich mit dem Einkaufen im selben Geschäft neu beginnen kann.		
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er ein Produkt im Warenkorb hat **Wenn** er den Warenkorb löscht **Dann** wird der Warenkorb geleert		
Prozesse: **Frontend:** Abbildung 11: Shopping_Cart_Frontend_2		

Title: Zahlen	**Priority:** 2	**Estimate:** 20
Als Kunde kann ich während des Einkaufens jederzeit den gesamten Warenkorb zahlen, sodass der weitere „Payment-Prozess“ gestartet wird.		
Acceptance Criteria **Gegeben ist,** dass der User auf der Warenkorb-Seite ist **Und** Produkte dem Warenkorb hinzugefügt wurden **Wenn** er den Knopf "Zahlen" drückt **Dann** wird er auf die Zahlungs-Seite weitergeleitet		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Scan Buy Go

Title: Biometrische Daten	**Priority:** 2	**Estimate:** 5
Als Kunde kann ich mit meinen am Smartphone gespeicherten biometrischen Daten die Zahlung starten, sodass sichergestellt wird, dass der Besitzer des Smartphones einkauft und das Smartphone nicht gestohlen ist.		
Acceptance Criteria **Gegeben ist,** dass der Kunde die Zahlung gestartet hat **Wenn** er seine gerätespezifischen biometrischen Daten erfolgreich bestätigt hat **Dann** wird die Zahlung durchgeführt		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Title: Zahlungsmöglichkeiten auswählen	**Priority:** 2	**Estimate:** 8
Als Kunde kann ich eine der vom Payment-Provider angebotenen Zahlungsmöglichkeiten wählen, sodass ich damit den Einkauf bezahlen kann.		
Acceptance Criteria **Gegeben ist,** dass der Kunde die Zahlung gestartet hat **Wenn** er seine Zahlungsdaten eingegeben hat **Dann** wird die Zahlung durchgeführt		

Scan Buy Go

Title: Zahlungsmöglichkeiten gespeichert	**Priority:** 2	**Estimate:** 4
Als Kunde kann ich meine vom Payment-Provider gespeicherte Zahlungsmöglichkeit wählen, sodass ich diese zur Bezahlung verwenden kann, ohne sie erneut hinzufügen zu müssen.		
Acceptance Criteria **Gegeben ist,** dass der Kunde die Zahlung gestartet hat **Wenn** er seine bereits gespeicherte Zahlungsmöglichkeit auswählt **Dann** wird die Zahlung durchgeführt		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Title: App/Rechnung/PDF + Rechnung erstellen	**Priority:** 2	**Estimate:** 20
Als Kunde kann ich direkt nach erfolgreicher Bezahlung meine Rechnung als PDF in der App ansehen, sodass ich die Möglichkeit besitze, diese gleich zu teilen, zu drucken oder zu versenden.		
Acceptance Criteria **Gegeben ist,** dass der User bereits erfolgreich den Warenkorb gezahlt hat **Wenn** er er seinen Kauf als PDF sieht **Dann** hat er die Möglichkeit seine Rechnung als PDF zu versenden als Link		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Scan Buy Go

Title: E-Mail/Rechnung/PDF	**Priority:** 2	**Estimate:** 3
Als Kunde kann ich nach erfolgreicher Bezahlung meine Rechnung als PDF per E-Mail erhalten, sodass ich diese überall zugänglich in meiner Mailbox habe.		
Acceptance Criteria **Gegeben ist,** dass der User bereits erfolgreich den Warenkorb gezahlt hat **Wenn** in seinem E-Mail Postfach seine zugeschickte Rechnung sieht **Dann** kann er diese als PDF öffnen		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Title: Nach dem Einkauf	**Priority:** 2	**Estimate:** 2
Als Kunde kann ich nach erfolgreicher Bezahlung auswählen, ob ich das aktuelle Geschäft verlassen oder nochmal einkaufen möchte, sodass ich nicht erneut den QR Code des Geschäfts scannen muss, falls ich einen weiteren Einkauf starten möchte.		
Acceptance Criteria **Gegeben ist,** dass der User bereits erfolgreich den Warenkorb gezahlt hat **Wenn** er den Knopf "Shop ausloggen" drückt **Dann** wird er aus dem aktuellen Shop abgemeldet		

Scan Buy Go

Title: Alte Rechnungen ansehen	**Priority:** 2	**Estimate:** 3
Als Kunde kann ich mir alle abgeschlossenen Rechnungen anzeigen lassen, sodass ich meine letzten Einkäufe nachvollziehen kann.		
Acceptance Criteria **Gegeben ist,** dass der User auf der Rechnungen-Seite ist **Wenn** die er die Liste aktualisiert **Dann** sieht er seine letzten Rechnungen		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Title: Alte Rechnungen filtern	**Priority:** 2	**Estimate:** 8
Als Kunde kann ich alle meine vergangenen Rechnungen nach Datum/Uhrzeit, Unternehmen inklusive Filiale filtern und anzeigen lassen, sodass ich nach einzelnen Rechnungen suchen kann.		
Acceptance Criteria **Gegeben ist,** dass der User auf der Rechnungen-Seite ist **Wenn** die er die Liste nach einem Kriterium filtert **Dann** sieht er alle Rechnungen die seiner Suchabfrage entsprechen		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Scan Buy Go

Title: Alte Rechnungen teilen	**Priority:** 2	**Estimate:** 1
Als Kunde kann ich alle meine vergangenen Rechnungen teilen, sodass ich diese per E-Mail, SMS, WhatsApp etc. versenden kann.		
Acceptance Criteria **Gegeben ist,** dass der User auf der Rechnungen-Seite ist **Wenn** die er auf das "Share-Icon" klickt **Dann** hat er die Möglichkeit seine Rechnung als PDF zu versenden als Link		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Title: Einkaufszettel	**Priority:** 2	**Estimate:** 3
Als Kunde kann ich mir einen virtuellen Einkaufzettel zu Nutze machen, sodass ich mir Produkte aufschreiben kann, welche ich in Zukunft kaufen will und nicht vergessen möchte.		
Acceptance Criteria **Gegeben ist**, dass der User in der App angemeldet ist **Und** er einen Einkaufszettel erstellen möchte **Wenn** er in der App auf der Navigationsleiste auf "Liste" klickt **Dann** wird er zur Einkaufszettel-Seite weitergeleitet und kann dann die Liste befüllen		

Scan Buy Go

Title: Passwort ändern	**Priority:** 2	**Estimate:** 2
Als Kunde kann ich mir ein neues Passwort setzen, sodass ich nicht mehr das alte, sondern nur mehr das neue Passwort zur Anmeldung benutzen kann.		
Acceptance Criteria **Gegeben ist**, dass der User sein Passwort ändern möchte **Wenn** er in der App bei den Einstellungen auf "Passwort ändern" klickt **Dann** dann kommt er zu dem Passwort ändern Screen wo er sein altes Passwort und 2 mal das neue Passwort eingeben muss		

Title: Daten ändern	**Priority:** 2	**Estimate:** 3
Als Kunde kann ich meine persönlichen Daten ändern, sodass ich diese bei Stammdatenänderungen entsprechend aktualisieren kann. Anrede Vorname Nachname Geburtsdatum UID		
Acceptance Criteria **Gegeben ist**, dass der User seine Daten ändern möchte **Wenn** er in die Einstellungen geht **Dann** kann er alle seine Daten in den Einstellungen ändern und speichern		

Scan Buy Go

Title: Account löschen	**Priority:** 2	**Estimate:** 2
Als Kunde kann ich meinen Account inklusive aller Daten löschen, sodass ich unwiderruflich keinen Account mehr zum Anmelden habe.		
Acceptance Criteria **Gegeben ist**, dass der User seinen Account löschen möchte **Wenn** er in der App in den Einstellungen auf "Account löschen" klickt **Dann** kann er nach erneuter Bestätigung via E-Mail, seinen Account löschen und wird aus der App ausgeloggt		

Title: Datenauskunft	**Priority:** 2	**Estimate:** 1
Als Kunde kann ich meine gespeicherten Daten anzeigen lassen, sodass ich mir diese durchlesen kann und darüber im Klaren bin.		
Acceptance Criteria **Gegeben ist**, dass der User gerne nach DSGVO Konformität seine verarbeiteten Daten wissen möchte **Wenn** er in den Einstellungen auf "Datenauskunft" klickt **Dann** bekommt er einen vollen Ausschnitt seiner verarbeiteten Daten via E-Mail zugesendet		

Scan Buy Go

Title: Newsletter verwalten	**Priority:** 2	**Estimate:** 1
Als Kunde kann ich meine Newsletter Präferenzen ändern, sodass ich mich von all meinen Newslettern abmelden kann.		
Acceptance Criteria **Gegeben ist**, dass der User seine Newsletter verwalten möchte **Wenn** er in den Einstellungen auf "Newsletter" klickt **Dann** kann er auf der Newsletter-Seite, alle Newsletter verwalten, bei denen er angemeldet ist		

Title: Email/Rechnung	**Priority:** 2	**Estimate:** 1
Als Kunde kann ich einstellen, ob mir immer nach jedem Einkauf die Rechnung per E-Mail als PDF zugesendet wird, sodass ich diese überall zugänglich in meiner Mailbox habe.		
Acceptance Criteria **Gegeben ist**, dass der User in der App angemeldet ist **Wenn** er Rechnungen direkt auf seine Email erhalten möchte **Dann** kann er in den Einstellungen aktivieren, dass er automatisch die Rechnungen an seine aktuelle Email geschickt bekommt		
Prozesse: **Backend**: Abbildung 9: Receipt_Backend_2		

Scan Buy Go

Title: Sprache	**Priority:** 2	**Estimate:** 5
Als Kunde kann ich einstellen, in welcher Sprache* die gesamte App dargestellt wird, sodass es zu keinerlei Verwirrungen aufgrund der Sprache kommen kann. *Deutsch, Englisch, Französisch, Italienisch, Spanisch, Portugiesisch, Chinesisch, Japanisch		
Acceptance Criteria **Gegeben ist**, dass der User in der App angemeldet ist **Und** er in den Einstellungen ist **Wenn** die Sprache auf Deutsch, Englisch oder Französisch geändert wird **Dann** wird die App automatisch in die jeweilige Sprache übersetzt		

Title: FAQ	**Priority:** 2	**Estimate:** 2
Als Kunde kann ich mir im Hamburger Menü jederzeit diverse Fragen rund um die App durchlesen, sodass ich bei etwaigen Problemen nicht sofort den Kundendienst kontaktieren muss.		
Acceptance Criteria **Gegeben ist**, dass der User in der App angemeldet ist **Wenn** er Fragen bezüglich der App hat **Dann** kann er im Hamburger-Menü auf FAQ klicken und wir dann zur FAQ Seite weitergeleitet		

Scan Buy Go

Title: Abmelden	**Priority:** 2	**Estimate:** 2
Als Kunde kann ich mich im Hamburger Menü jederzeit von der App abmelden, sodass ich nicht mehr in meinem Konto angemeldet bin oder das Konto wechseln kann.		
Acceptance Criteria **Gegeben ist**, dass der User in der App angemeldet ist **Und** er sich aus der App abmelden will **Wenn** er im Hamburger-Menü auf "Abmelden" klickt **Dann** wird er abgemeldet und zurück zur Login-Seite geschickt		

Title: Kontakt	**Priority:** 2	**Estimate:** 2
Als Kunde kann ich im Hamburger Menü jederzeit den Kundensupport kontaktieren, sodass ich bei Fragen oder Problemen einen Ansprechpartner habe.		
Acceptance Criteria **Gegeben ist**, dass der User in der App angemeldet ist **Und** er Kontakt aufnehmen möchte **Wenn** er im Hamburger-Menü auf "Kontakt" klickt **Dann** wird er zur Kontakt-Seite weitergeleitet		

Scan Buy Go

Title: Warenkorb bearbeiten	**Priority:** 2	**Estimate:** 3
Als Kunde kann ich im Warenkorb mehrere Produkte gleichzeitig auswählen, sodass ich diese gemeinsam aus dem Warenkorb entfernen kann.		
Acceptance Criteria **Gegeben ist**, dass der User in einem Shop angemeldet ist **Und** er mindestens ein Produkt im Warenkorb hat **Wenn** er mehrere Produkte gleichzeitig entfernen will **Dann** kann er sie auswählen und sie werden entfernt		
Prozesse: **Frontend**: Abbildung 11:		

Title: Email ändern	**Priority:** 2	**Estimate:** 3
Als Kunde kann ich meine Email-Adresse ändern, sodass ich mich mit meiner neuen Email anmelden kann		
Acceptance Criteria **Gegeben ist**, dass der User seine E-Mail Adresse ändern möchte **Wenn** er in den Einstellungen auf "E-Mail ändern" klickt **Dann** muss der User seine alte E-Mail, sein Passwort und die neue E-Mail Adresse eingeben und wird bei Erfolg zur Verifikations-Seite weitergeleitet		

Scan Buy Go

Title: Newsletter anmelden	**Priority:** 2	**Estimate:** 5
Als Kunde kann ich mich für den Newsletter des aktuell eingeloggten Shops anmelden, sodass ich diverse Angebote und Informationen von diesem Store bekomme.		
Acceptance Criteria **Gegeben ist**, dass der User einen Newsletter eines Shops abonnieren möchte **Wenn** er in einem Shop eingeloggt ist **Dann** kann er in den Einstellungen den Newsletter des aktuellen Shops abonnieren		

Scan Buy Go

Nicht funktionale Anforderungen

Anforderung	#1
Kategorie	Leistung
Beschreibung	Das System darf nicht länger als drei Sekunden benötigen, um ein Produkt in der Datenbank zu finden und darzustellen.
Problembeschreibung	Der Gesamteinkauf verzögert sich für den Kunden.

Anforderung	#2
Kategorie	Leistung
Beschreibung	Das System muss in der Prototypphase der Verarbeitung von 100 Anfragen pro Minute standhalten.
Problembeschreibung	Größere Kundenstämme benötigen mehr Backend-Ressourcen.

Anforderung	#3
Kategorie	Qualität
Beschreibung	Die App muss Barrierefreiheitsanforderungen von Benutzern gerecht werden. Das beinhaltet eine Option für größere Schrift und Übersetzung der App in die Sprache des Benutzers.
Problembeschreibung	Jeder Kunde hat spezielle Bedürfnisse.

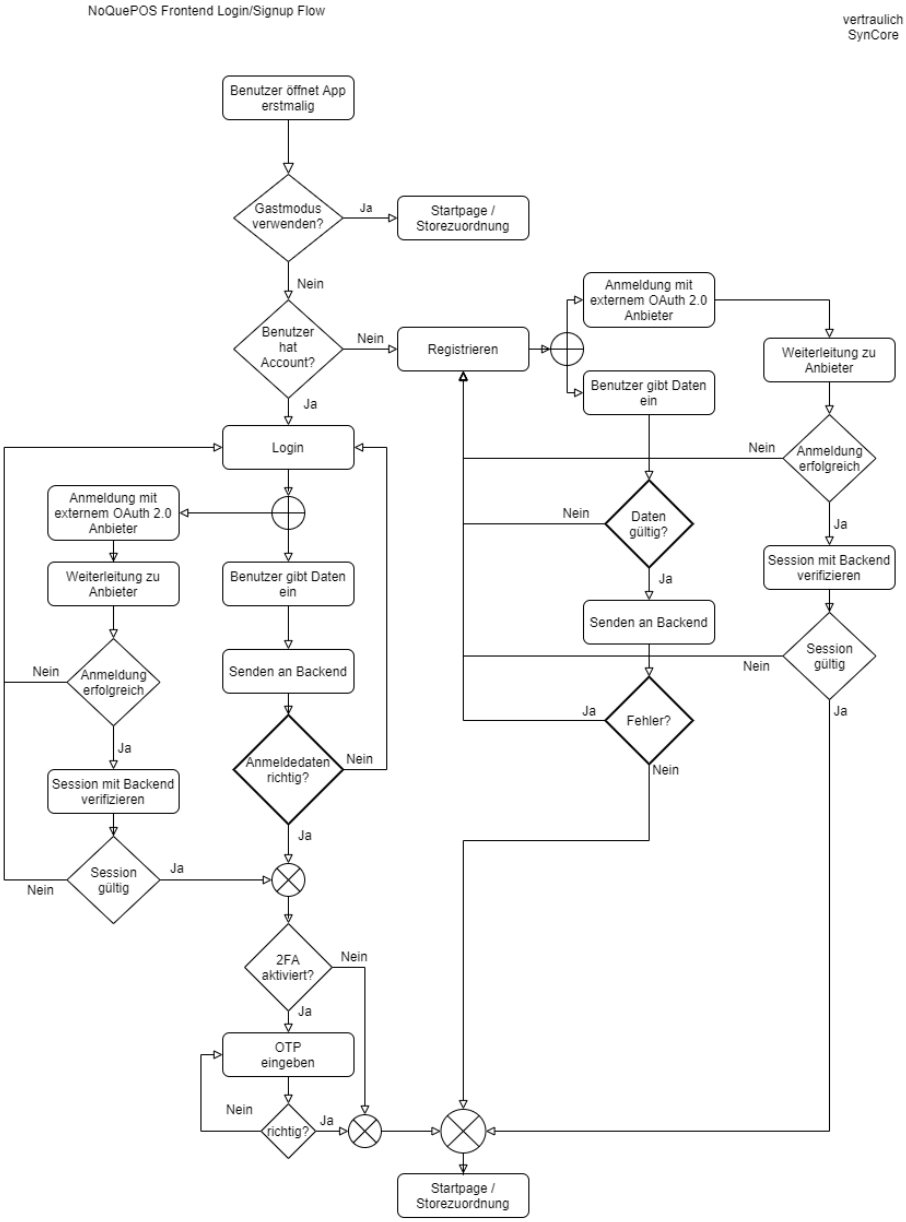
Anforderung	#4
Kategorie	Qualität
Beschreibung	Die App muss auf Android und iOS ausführbar sein
Problembeschreibung	Kunden benutzen sowohl Android und iOS-basierte Smartphones.

Scan Buy Go

Anforderung	#4
Kategorie	IT-Sicherheit
Beschreibung	Die App blockiert nach jeder erneuten Code-Anfrage für die 2-Faktor Authentifizierung eine erneute Anfrage des Codes für 30 Sekunden.
Problembeschreibung	Verhinderung der Überlastung des Backend.

Anforderung	#5
Kategorie	IT-Sicherheit
Beschreibung	Das System muss den unautorisierten Zugriff auf die Kundenstammdaten verhindern.
Problembeschreibung	Aufgrund der Cyber-Security müssen kritische Daten verschlüsselt gespeichert werden.

Anhang: Prozessdarstellungen Frontend und Backend



Datenvalidierung

- Email gültig (.*@.*\..*)
- Passwort erfüllt Richtlinien
  - 8+ Zeichen
  - [A-Z]
  - [0-9]
  - min. 1 Sonderzeichen
- ReCaptcha als Spamschutz?

Externe OAuth Anbieter

Ermöglichen Benutzern, sich mit Diensten wie Google, Facebook, Twitter, Apple anzumelden. Ein genereller Ablauf ist hier dargestellt, der Benutzer wird auf eine externe Seite des Betreibers weitergeleitet und stimmt der Verwendung seiner Daten zu (Zugriffe auf Email und Name sind dadurch möglich, ohne dass der Benutzer diese eingeben muss)

Two-Factor Authentification (2FA)

Benötigt eine weitere Information vom User, bevor er sich anmelden kann oder gravierende Änderungen an seinem Konto vornehmen kann. Diese Information ist ein One Time Password (OTP) oder ein Time based One Time Password, abhängig davon, ob es nur bei Bedarf generiert und z.B. per SMS oder E-Mail versandt wird oder ständig von einer App wie Google Authenticator oder Authy generiert wird.

Scan Buy Go

Abbildung 1: Login_Register_Frontend

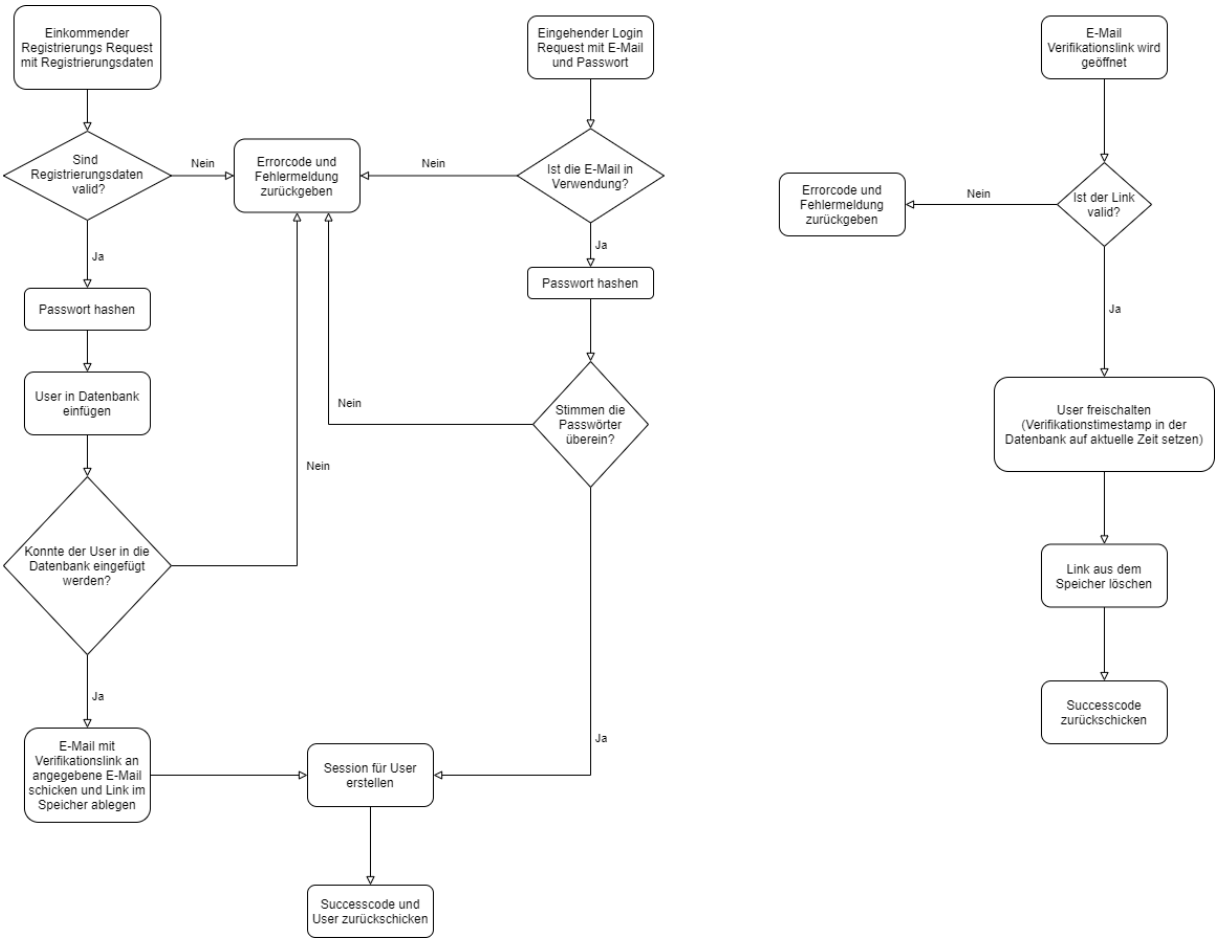


Abbildung 2: Login_Register_Backend

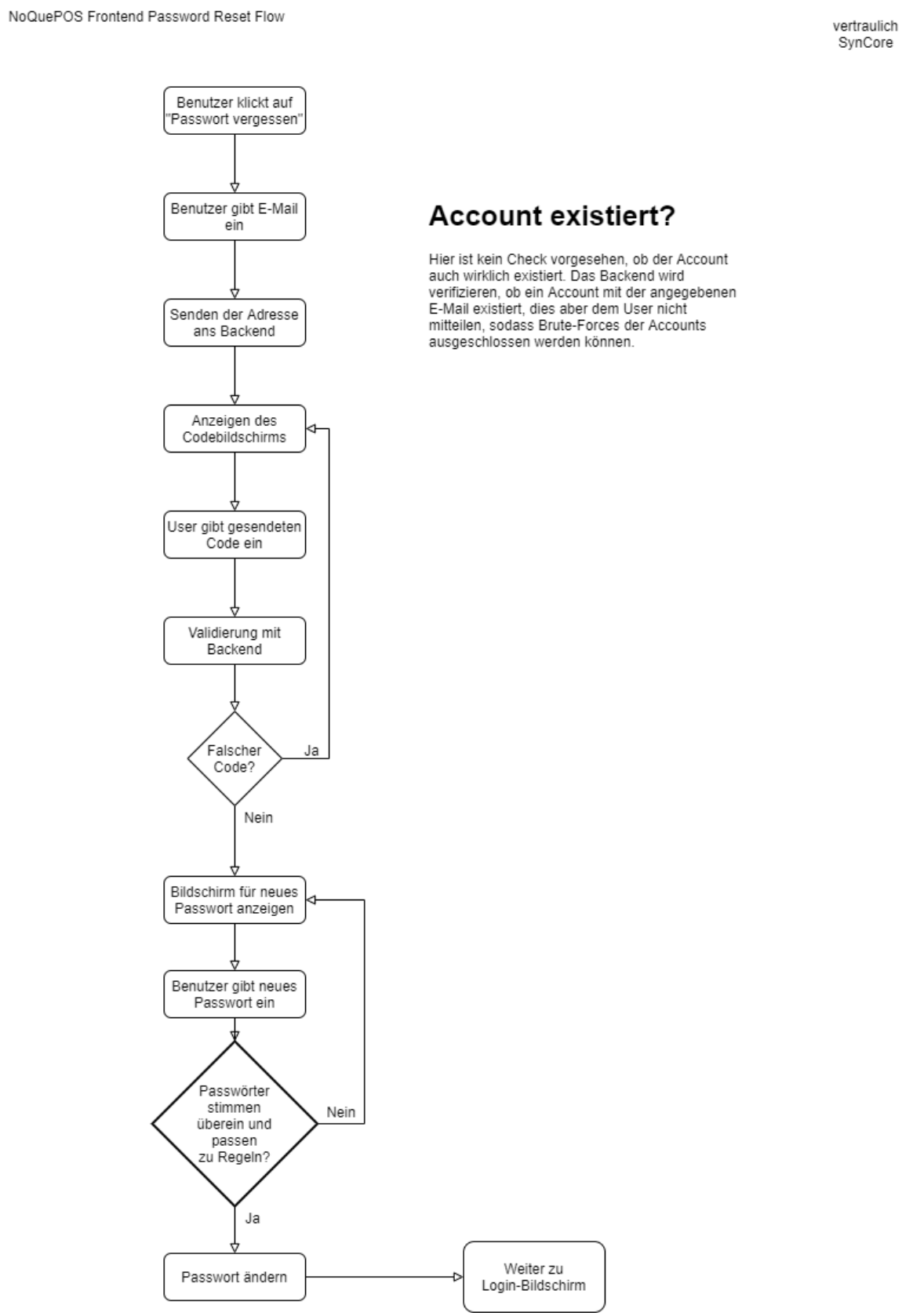
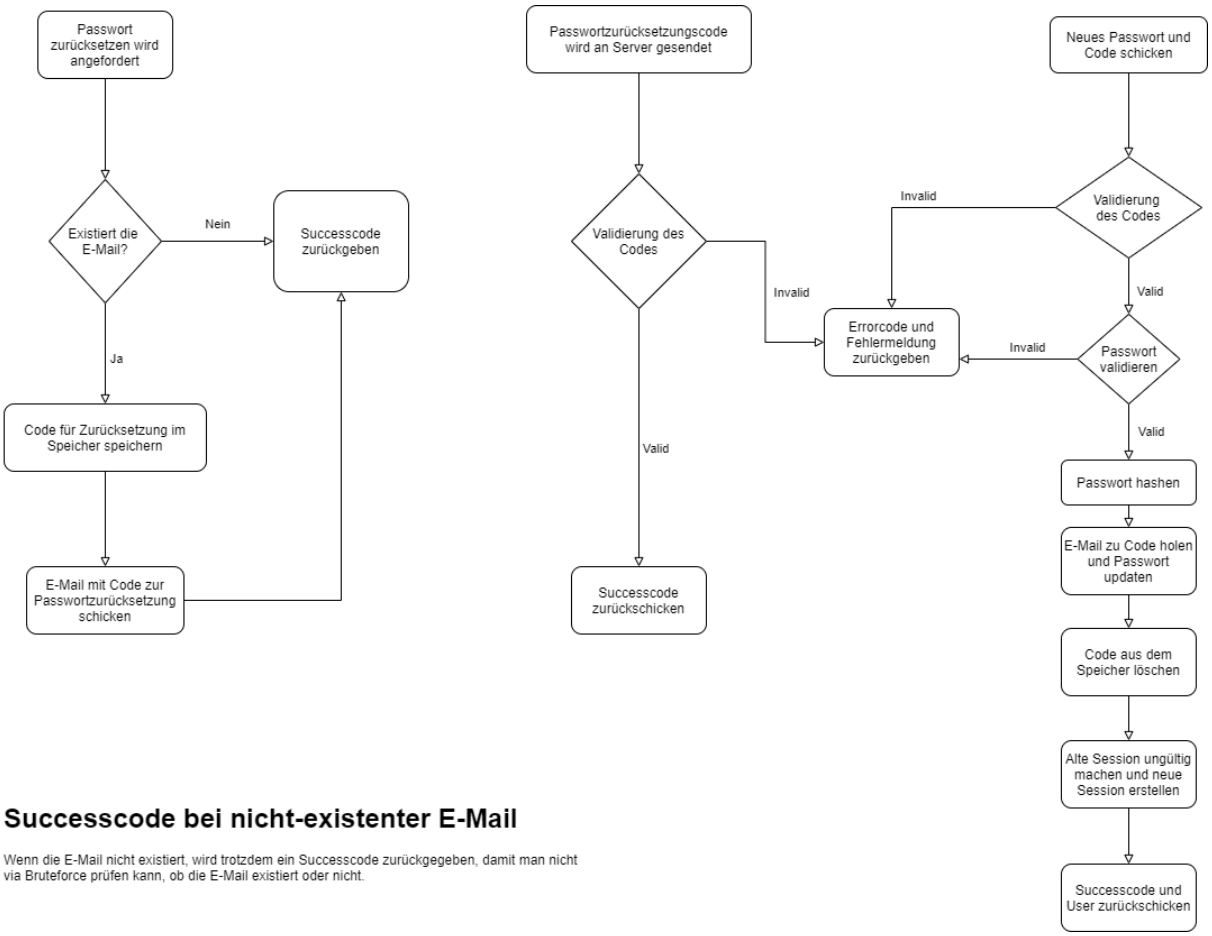


Abbildung 3: Password_Reset_Frontend

Scan Buy Go



Successcode bei nicht-existenter E-Mail

Wenn die E-Mail nicht existiert, wird trotzdem ein Successcode zurückgegeben, damit man nicht via Bruteforce prüfen kann, ob die E-Mail existiert oder nicht.

Abbildung 4: Password_Reset_Backend

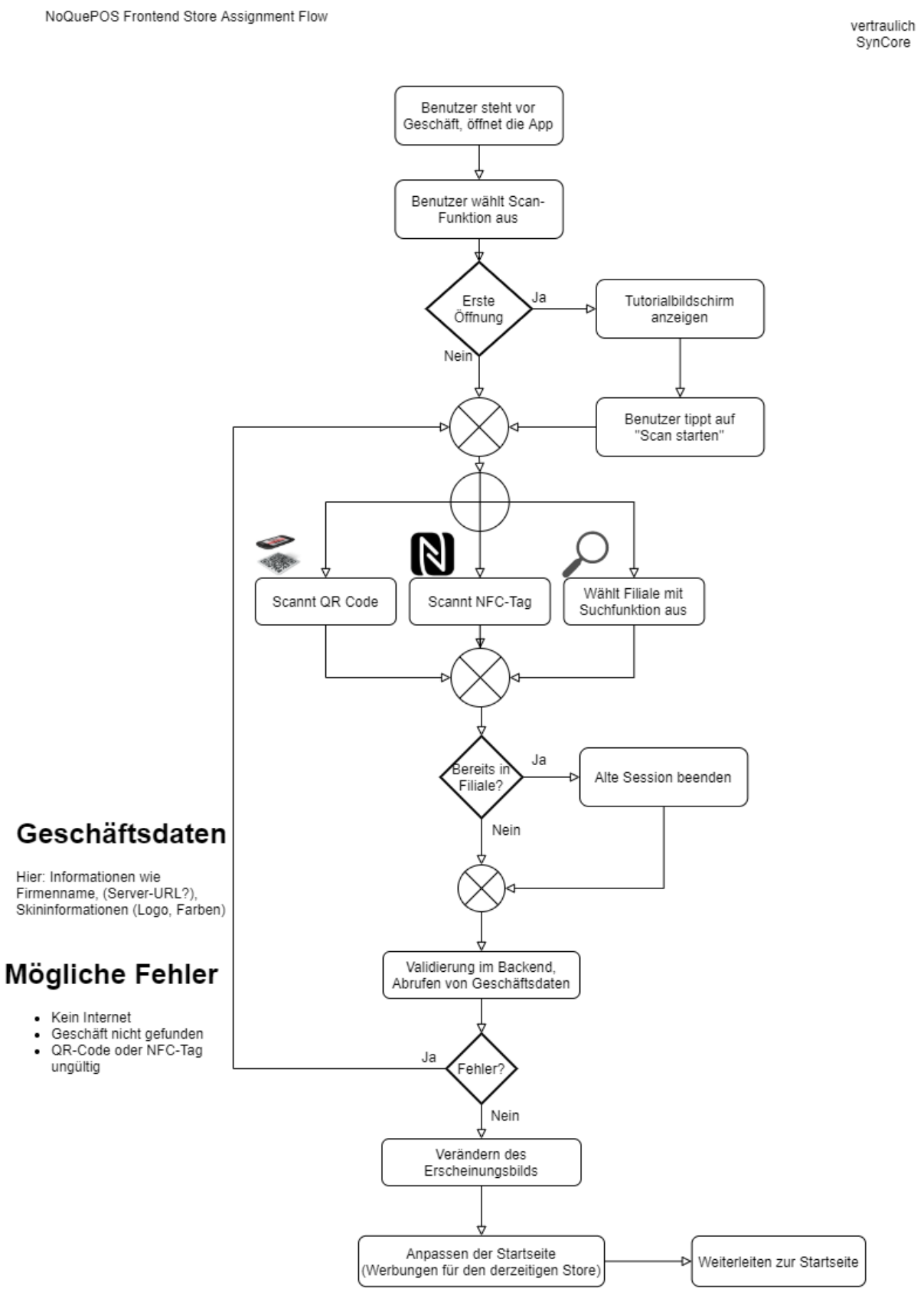


Abbildung 5: Shop_Assignment_Frontend

Scan Buy Go

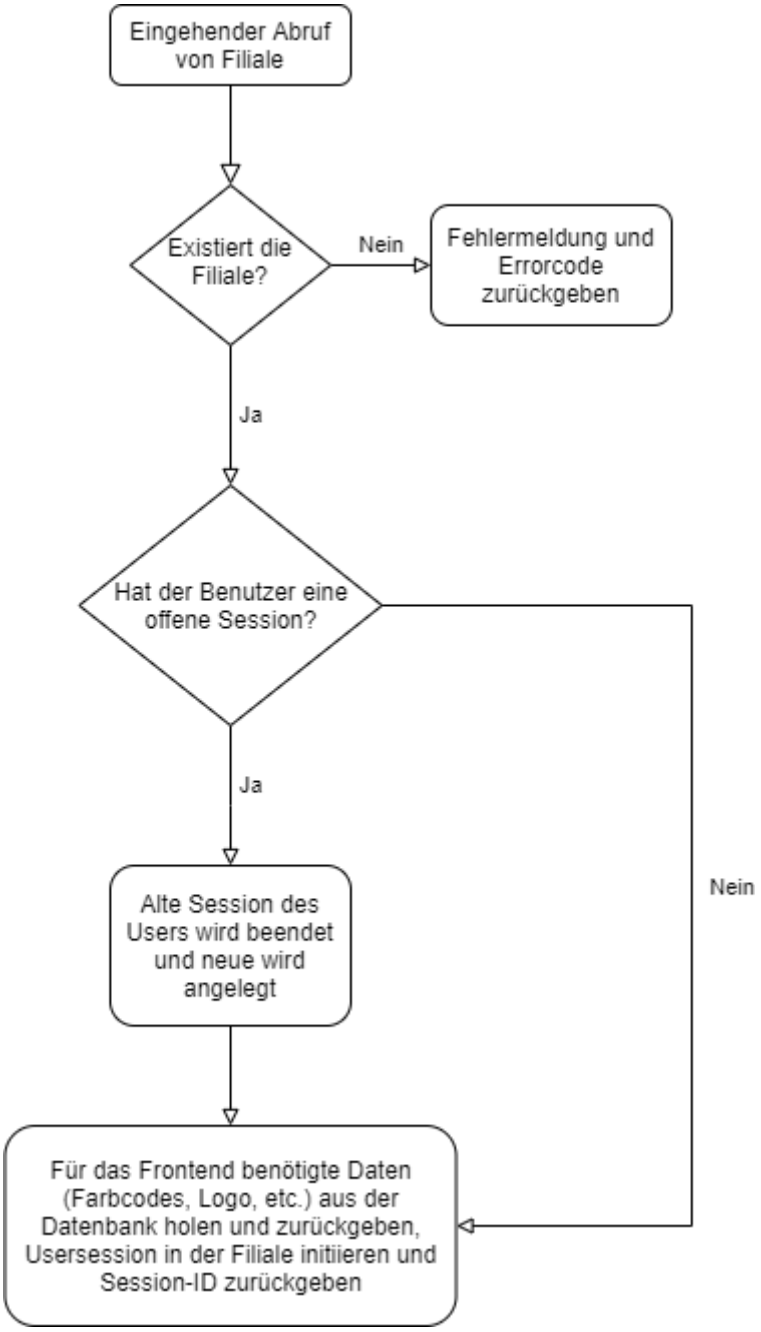


Abbildung 6: Shop_Assignment_Backend

Scan Buy Go

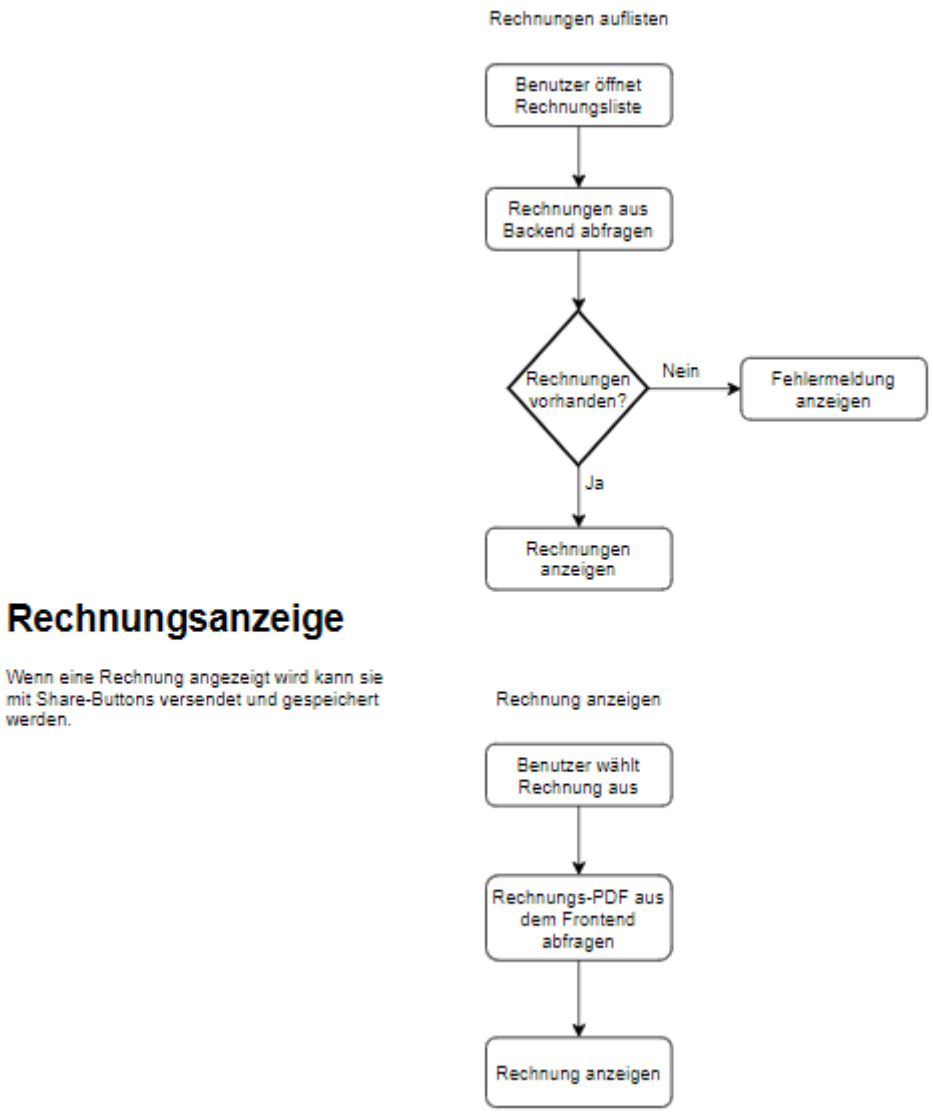


Abbildung 7: Receipt_Frontend

Scan Buy Go

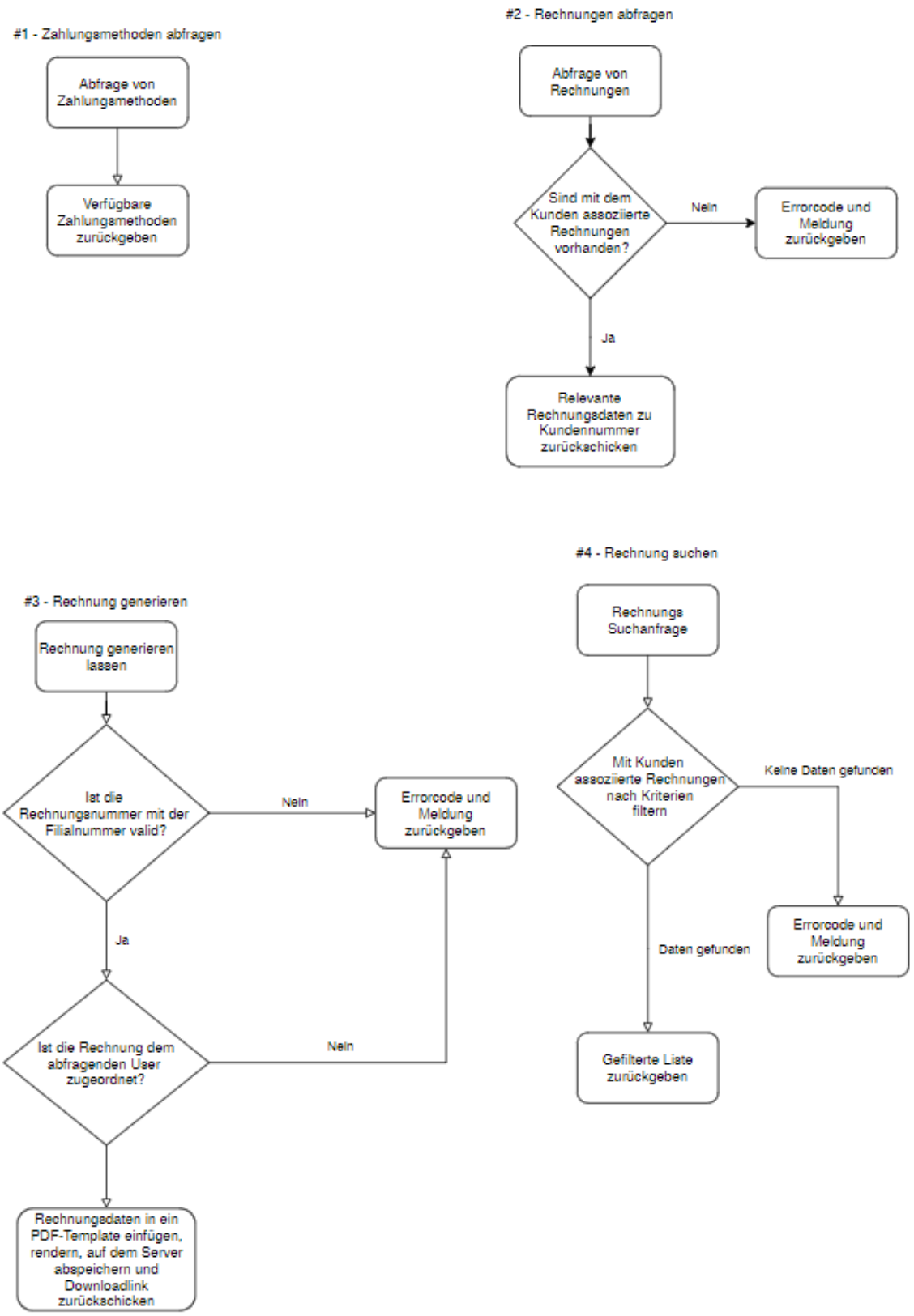


Abbildung 8: Receipt_Backend_1

Scan Buy Go

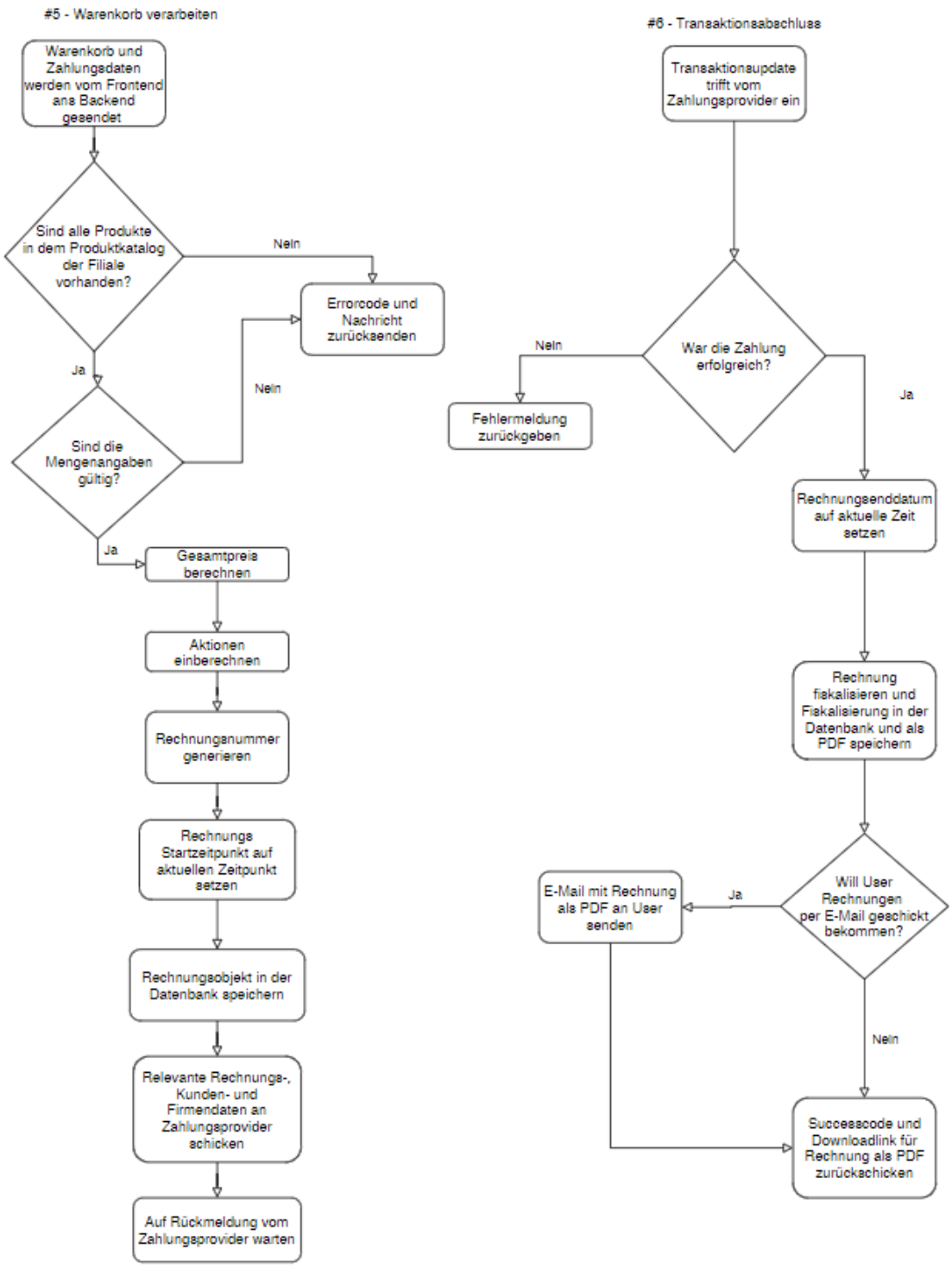


Abbildung 9: Receipt_Backend_2

Scan Buy Go

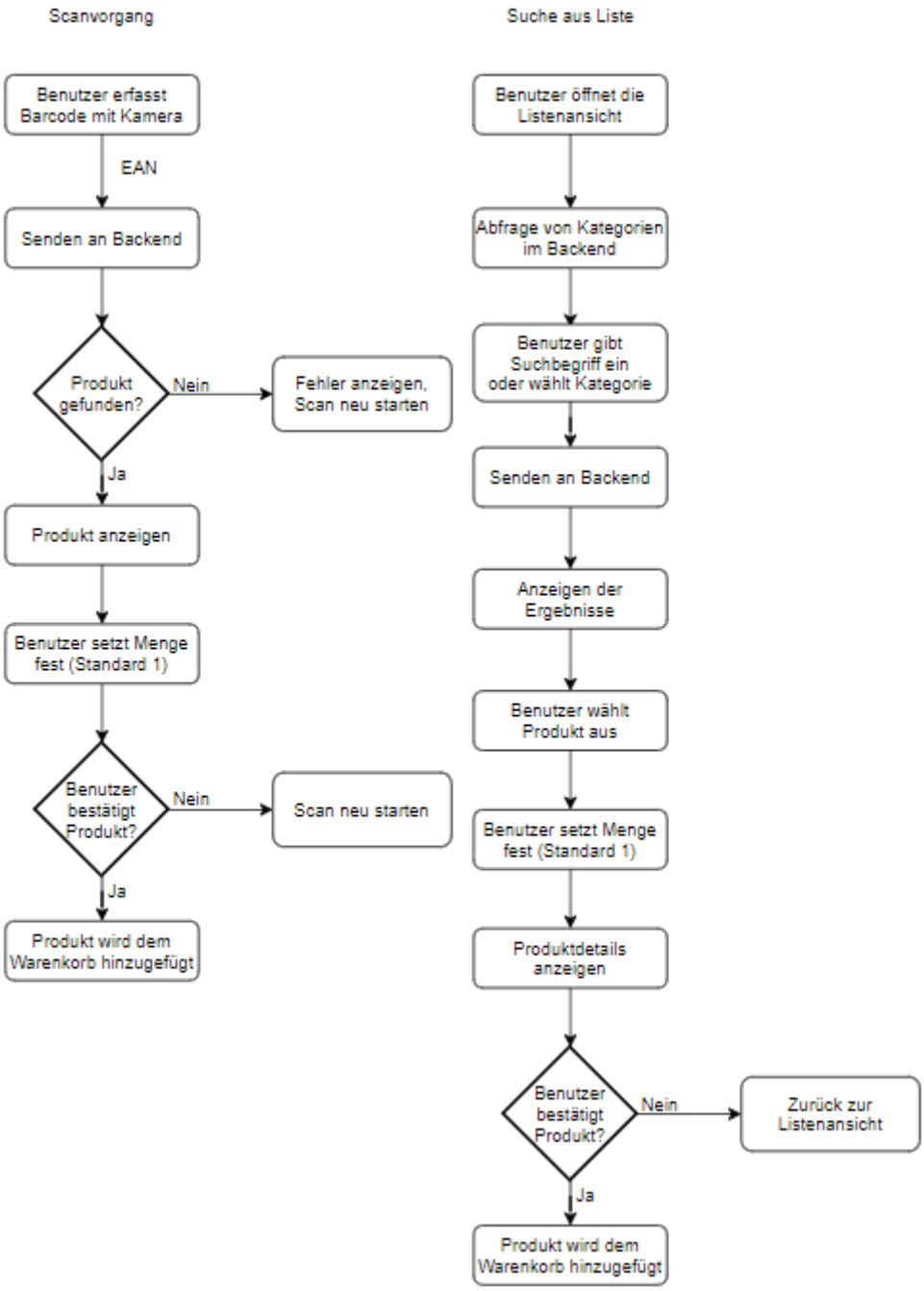


Abbildung 10: Shopping_Cart_Frontend_1

Scan Buy Go

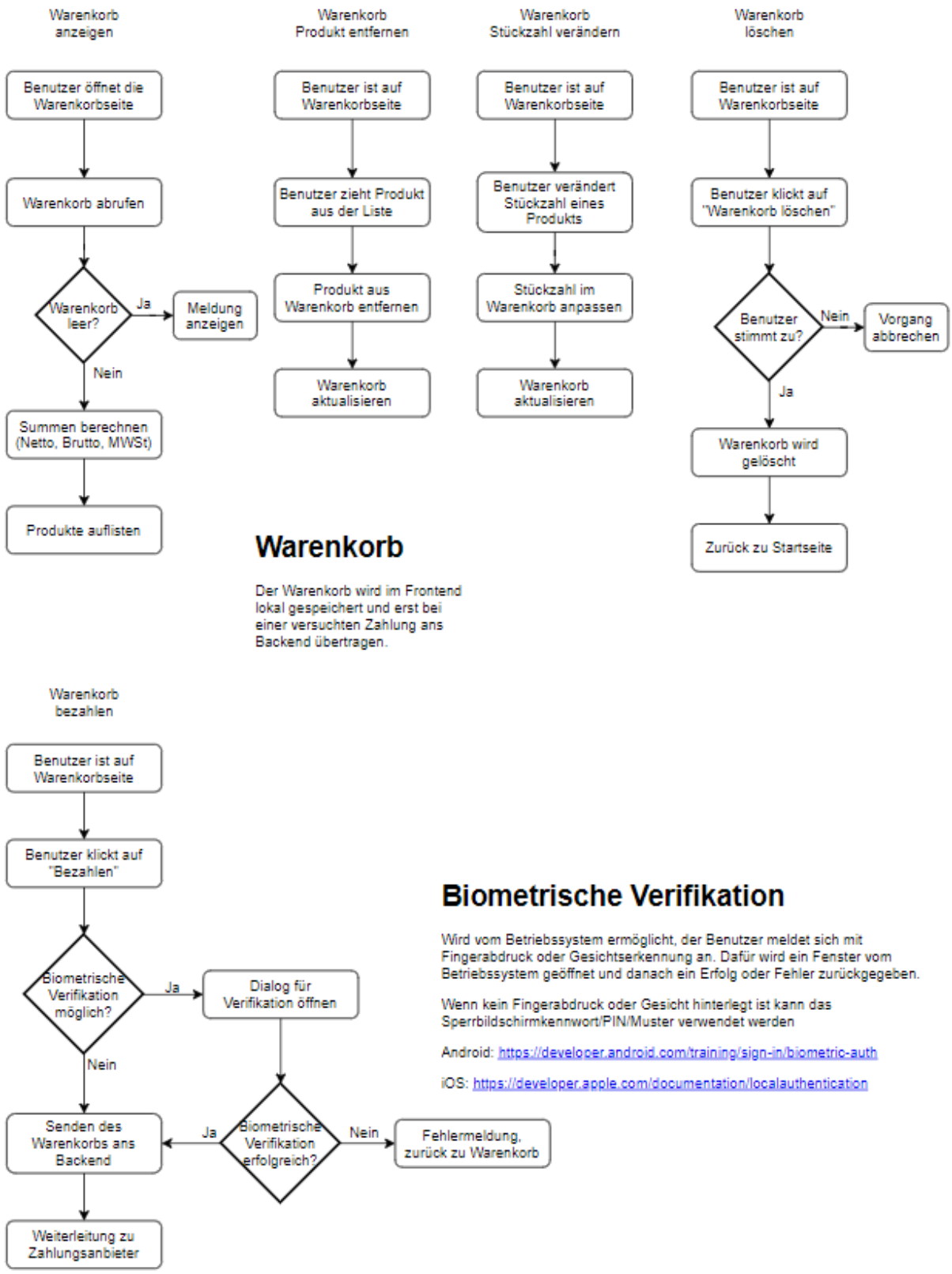
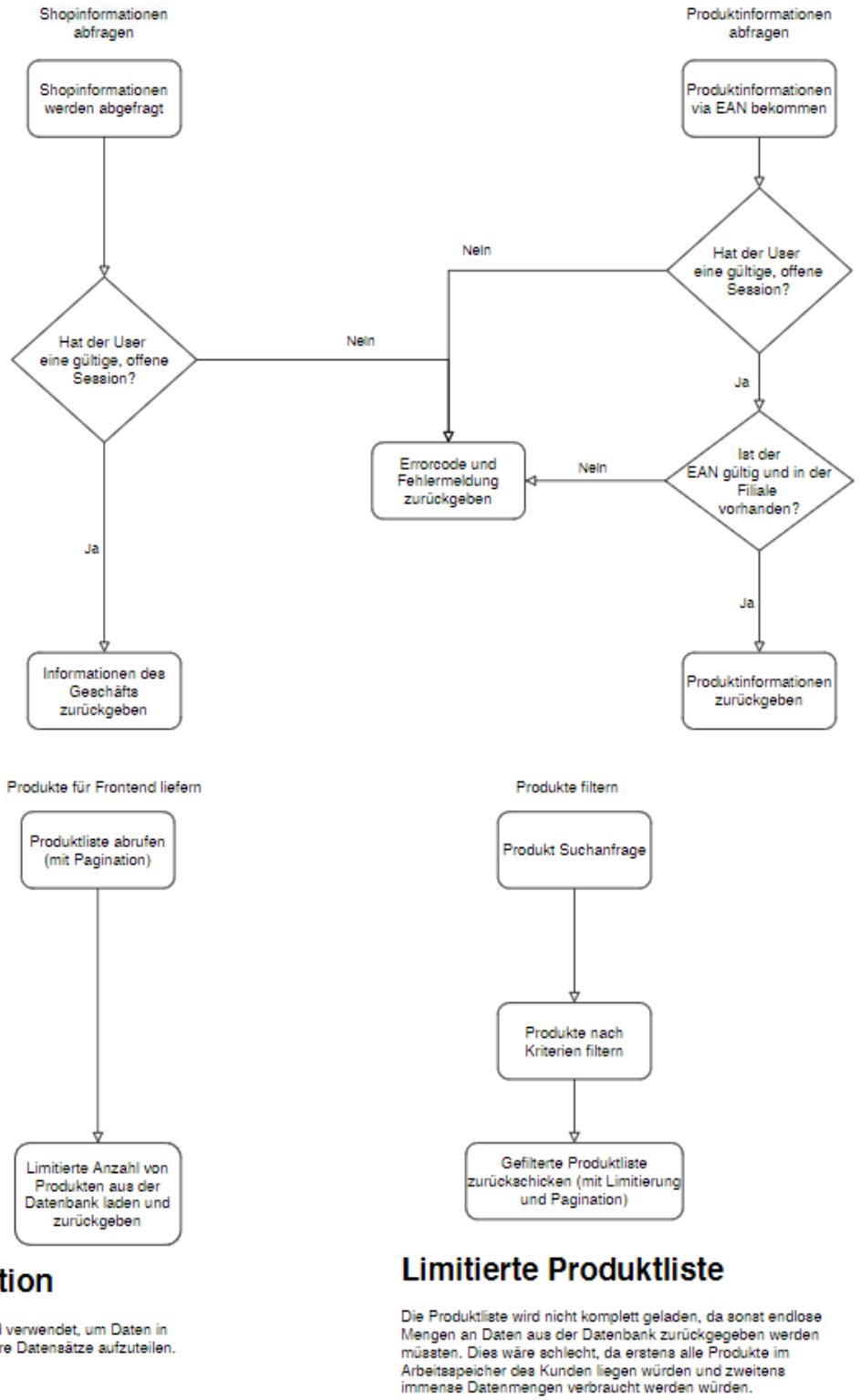


Abbildung 11: Shopping_Cart_Frontend_2



Pagination

Pagination wird verwendet, um Daten in mehrere kleinere Datensätze aufzuteilen.

Limitierte Produktliste

Die Produktliste wird nicht komplett geladen, da sonst endlose Mengen an Daten aus der Datenbank zurückgegeben werden müssten. Dies wäre schlecht, da erstens alle Produkte im Arbeitsspeicher des Kunden liegen würden und zweitens immense Datenmengen verbraucht werden würden.

Abbildung 12: Shopping_Cart_Backend

Appendix C: Technical and organizational measurements (TOMs)

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

ScanBuyGo
Technical and Organisational Measures

GDPR Art. 5

Technical and Organisational Measures

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Table of Contents

Table of Contents **2**

Introduction **3**

Production Measures **4**
- Introduction 4
- Technical Measures 4
  - Encryption in transit 4
  - Encryption at rest 4
  - Data minimization 4
  - Password requirements 4
  - Logging 5
- Organisational Measures 5
  - Principle of least privilege 5
  - Purpose limitation 5
  - Physical device security 5
  - Hardware Tokens and Multi-Factor Authentication 5
  - Customer support 5
  - Working with third parties 6
  - No local copies of data 6

Development Measures **6**
- Introduction 6
- Technical Measures 6
  - Use anonymized or mocked data for testing 6
  - Use secured connections and devices for programming 6
  - Use a separate testing environment 7
  - Code Reviewing 7
- Organisational Measures 7
  - Only deploy to production environments from the master branch 7
  - Code sharing 7
  - Only program in safe areas 7
  - Password requirements for developers 7

Technical and Organisational Measures

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Introduction

This document lays out Measures to be taken by ScanBuyGo's Team and Partners in order to protect sensitive data. It is separated into sections for production and development, explaining further measures to be taken throughout the product's active development and maintenance.

Technical and Organisational Measures

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Production Measures

Introduction

This section aims to lay out how to handle potentially sensitive information in ScanBuyGo's production environment including but not limited to customer data or payment information.

Technical Measures

Encryption in transit

Any and all traffic to and from the ScanBuyGo Application Servers is encrypted using modern cryptography. Examples for modern and safe encryption include TLS 1.2 or greater (commonly referred to as HTTPS) or SHA-256 encryption.

Encryption at rest

Data stored on ScanBuyGo Systems, aside from other access control measures, is stored at rest, meaning no data can be read from a hard drive without authorization. This includes encrypting hard drives, using a TPM (Trusted Platform Module) and its functions wherever possible and encrypting sensitive data at the application layer using a hashing function like SHA-256 or bcrypt.

Data minimization

Only as few data as is required shall be processed or stored. In any case, the data stored or returned shall only be as much as is absolutely required to fulfill a purpose or provide the user a better experience.

Password requirements

Any staff or administrator password in any internal or external systems needs to meet the following requirements:

- More than 8 characters
- No common or easily guessable password (e.g. Password123)
- Contain at least one number
- Contain at least one special character

The best and most secure passwords exceed these requirements. It is encouraged to use a password generator whenever possible.
Passwords must be changed if they might have been compromised or guessed.

Technical and Organisational Measures

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Logging

Wherever possible, any access to ScanBuyGo Applications is logged. These logs are to be reviewed weekly for any anomalies suggesting potential breaches of confidentiality. Log data is only retained for 14 days in accordance with the privacy policy.

Organisational Measures

Principle of least privilege

The principle of least privilege is to be considered when setting or handling permissions. Any user, employee or administrator shall only have access to as little data and as few systems as absolutely necessary. To assure this, permissions need to be audited on a regular basis, removing any unnecessary access.

Purpose limitation

Any data that is stored or processed shall only be used in accordance with and for a purpose outlined in the Procedural Directory and Privacy Policy Documents.

Physical device security

Any work devices used to access customer data need to fulfill security requirements in order to protect stored data on them:

- The device shall be locked whenever not in use.
- The device shall not be used in a public area
- The same device/user/container shall not be used for private or unrelated purposes.
- Whenever possible, the entire or sections of the drive in use shall be encrypted.

Hardware Tokens and Multi-Factor Authentication

Whenever possible, any alternative or addition to passwords should be preferred or used. This may include government-issued certificate signatures, one-time tokens or hardware tokens.

Customer support

Customer support shall be able to assist customers with their inquiries without compromising their security or causing a breach of confidentiality. Therefore, while assisting a customer:

- The customer’s identity needs to be confirmed: This can be done either by requesting entered personal data from them such as a full name, address and birthdate or by confirming their email address. Before it is assured that it is the actual customer no data shall be revealed and no account changes shall be made.

- Passwords shall not be reset by support agents: There is a secure way to request a password reset requiring an email to be sent to the customer via the application. If a user requests to have their other personal data edited or appended this can be done but only after confirming their identity.

Working with third parties

Whenever working with a third party, be it an employee, partner or provider, required contracts including proper confidentiality clauses shall be signed and stored even after the collaboration has ended.

No local copies of data

Whenever handling customer data, it shall be avoided to copy and store it locally. If this is required, the local copies shall be removed after use.

Development Measures

Introduction

As ScanBuyGo is in constant development and will require to be maintained to compete with other products in the market appropriate measures need to be taken in order to protect customer data and the product.

Technical Measures

Use anonymized or mocked data for testing

Do not use data from real customers for testing whenever avoidable.

Use secured connections and devices for programming

Use a separate device, user or container for development that is not used for other purposes. For connecting to development environments and code sharing services use secured connections like HTTPS, SFTP or SSH.

Technical and Organisational Measures

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Use a separate testing environment

Use a dedicated, fully separate environment for testing the application. It shall not have any connection to a database, server, etc. used in production and shall not have actual customer data stored in it.

Code Reviewing

Merges between branches require the entire team (respectively Frontend or Backend Team) to approve a pull request. Furthermore, no comments can be open while merging. This is to ensure the entire team is on the same page and to prevent harmful code from being pushed into production.
It is recommended to adopt pair programming in addition to this to provide and receive code feedback while programming.

Organisational Measures

Only deploy to production environments from the master branch

Any deployment into production needs to come from the master branch. This ensures that if it is a feature it has been reviewed 2-3 times by the entire team and that no unapproved development code can end up in a release build, potentially leaking data.

Code sharing

Code may only be shared between team members in internal communication systems. These systems are:

- Azure Repos
- Microsoft Teams

Only program in safe areas

Shoulder Surfing and other attack vectors shall be avoided by only using work devices and programming only in areas where not many people are around. Details about features and database structures shall not be shared outside the team.

Password requirements for developers

Any staff or administrator password in any internal or external systems needs to meet the following requirements:

- More than 8 characters
- No common or easily guessable password (e.g. Password123)

- Contain at least one number
- Contain at least one special character

The best and most secure passwords exceed these requirements. It is encouraged to use a password generator whenever possible.
Passwords must be changed if they might have been compromised or guessed.

Appendix D: Procedural directory

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

ScanBuyGo Procedural Directory

GDPR Art. 30

Table of Contents

Table of Contents 1

Data processors and recipients 2

Data processing overview 3

Data processed in detail 4

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Data processors and recipients

Link to contract with controller	Name and details of controller	Categories of processing	Technical and organisational security measures
https://stripe.com/at/legal	Stripe, Inc. 510 Townsend Street, San Francisco, CA 94103, USA, https://www.stripe.com	Payment processing	Communications with Stripe containing sensitive payment data are performed between the end device and Stripe directly, encryption of traffic using SSL/TLS
https://azure.microsoft.com/en-us/support/legal/	Microsoft Azure	Server Hosting	Data is not stored directly on Azure Services, it is being passed on to a Heroku database using encrypted data transfer.
N/A	Heroku, https://www.heroku.com/	Data storage	Data is stored in an encrypted database

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Data processing overview

Purposes of processing	Categories of individuals	Categories of personal data
Sign-in, Registration	Customers	Contact Details (Email, Full Name, VAT ID, Address)
		Password
		IP address
		Date of Birth
Shop assignment	Customers	Location (Current Store)
Shopping Cart	Customers	Lifestyle information (Cart contents)
Payment	Customers	Financial details (Payment method)
		Contact details (Full Name, Address)
Invoicing	Customers	Location (Current Store)
		Contact Details (Full Name, Address)
		Financial Details (Payment method)

All data processed by us is only required to provide the service ScanBuyGo to our customers:

- Contact Details are required to identify users for their accounts and to offer adding their details to invoices.
- IP addresses are used to prevent abuse
- The date of birth is used to limit the purchasing of age-gated items like alcohol to minors.
- The current store and contents of the user's cart are necessary to complete a purchase using our app.

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Data processed in detail

Data processed for account operations (Login, registration):

Category of data	Data used	Recipients	Retention schedule	Lawful basis	Legitimate interest	Rights available to individuals	Source of personal data	Consent
Contact Details	Email address	ScanBuyGo	Account lifetime*	Art. 6(1)(a) - Consent	N/A	Access, data portability, rectification, erasure	Entered by Customer during Registration	Given upon registering
Contact Details	Full Name (First name, Last name)	ScanBuyGo	Account lifetime*	Art. 6(1)(a) - Consent	N/A	Access, data portability, rectification, erasure	Entered by Customer during Registration	Given upon registering
Contact Details	VAT ID	ScanBuyGo	Account lifetime*	Art. 6(1)(a) - Consent	N/A	Access, data portability, rectification, erasure	Entered by Customer during Registration	Given upon registering
Contact Details	Address	ScanBuyGo	Account lifetime*	Art. 6(1)(a) - Consent	N/A	Access, data portability,	Entered by Customer during	Given upon registering

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

						rectification, erasure	Registration	
Account details	Password	ScanBuyGo (hashed)	Account lifetime*	Art. 6(1)(a) - Consent	N/A	Access, data portability, rectification, erasure	Entered by Customer during Registration	Given upon registering
Personal Details	Date of Birth	ScanBuyGo	Account lifetime*	Art. 6(1)(a) - Consent	N/A	Access, data portability, rectification, erasure	Entered by Customer during Registration	Given upon registering
Connection Details	IP Address	ScanBuyGo	14 days	Art. 6(1)(f) - Legitimate interest	Abuse prevention, QoS	N/A	Sent by device when contacting the service	N/A

*Account lifetime: Until the customer deletes their account (or the account is removed for other reasons)

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Data processed for shopping without completing a purchase:

Category of data	Data used	Recipients	Retention schedule	Lawful basis	Legitimate interest	Rights available to individuals	Source of personal data	Consent
Location	Unique store identifier the user is at	ScanBuyGo	End of shopping session OR 24 hours	Art. 6(1)(b) - Contract	N/A	Erasure, access	Entered by the user (or scanned from a QR Code) upon entering a shop	N/A
Lifestyle information	Products scanned	ScanBuyGo	End of shopping session OR 24 hours	Art. 6(1)(b) - Contract	N/A	Erasure, access, rectification	Entered by the user (or scanned barcode) upon selecting a product	N/A
Connection Details	IP Address	ScanBuyGo	14 days	Art. 6(1)(f) - Legitimate interest	Abuse prevention, QoS	N/A	Sent by device when contacting the service	N/A

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Data processed for shopping and completing a purchase:

Category of data	Data used	Recipients	Retention schedule	Lawful basis	Legitimate interest or law	Rights available to individuals	Source of personal data	Consent
Location	Unique store identifier the user is at	ScanBuyGo, Shop	7 years	Art. 6(1)(b) - Contract, Art. 6(1)(c) - Legal obligation	Invoice retention as defined by UStG § 11 Abs 2 3	Access	Entered by the user (or scanned from a QR Code) upon entering a shop	N/A
Lifestyle information	Products scanned and purchased	ScanBuyGo, Shop	7 years	Art. 6(1)(b) - Contract, Art. 6(1)(c) - Legal obligation	Invoice retention as defined by UStG § 11 Abs 2 3	Access	Entered by the user (or scanned barcode) upon selecting a product	N/A
Contact Details	Full Name (First name, Last name)	ScanBuyGo, Shop, Stripe**	7 years	Art. 6(1)(b) - Contract, Art. 6(1)(c) - Legal obligation	Invoice retention as defined by UStG § 11 Abs 2 3	Access	Entered by Customer during Registration	Given upon registering
Contact Details	VAT ID	ScanBuyGo, Shop	7 years	Art. 6(1)(b) - Contract,	Invoice retention as	Access	Entered by Customer	Given upon registering

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

				Art. 6(1)(c) - Legal obligation	defined by UStG § 11 Abs 2 3		during Registration	
Contact Details	Address	ScanBuyGo, Shop, Stripe**	7 years	Art. 6(1)(b) - Contract, Art. 6(1)(c) - Legal obligation	Invoice retention as defined by UStG § 11 Abs 2 3	Access	Entered by Customer during Registration	Given upon registering
Financial Details	Payment method	ScanBuyGo, Shop, Stripe**	7 years	Art. 6(1)(b) - Contract, Art. 6(1)(c) - Legal obligation	Invoice retention as defined by UStG § 11 Abs 2 3	Access	Provided by payment provider (Stripe) after completing payment	Given upon registering
Connection Details	IP Address	ScanBuyGo	14 days	Art. 6(1)(f) - Legitimate interest	Abuse prevention, QoS	N/A	Sent by device when contacting the service	N/A
Financial Details***	Credit card details (Number, CVV, Expiry Date)	Stripe	See Stripe Retention Policy *	Art. 6(1)(b) - Contract	N/A	N/A	Entered by Customer during checkout	N/A

* For Data Retention Policies of Stripe, Inc. see https://stripe.com/at/privacy#security-and-retention

** Contact information is passed to Stripe, Inc. due to EU Payment regulations and to lower the risk of fraudulent transactions.

Procedural Directory

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

*** Detailed card information is not stored or handled by ScanBuyGo servers, it is sent to Stripe, Inc. by the frontend application directly

Appendix E: Privacy Policy

Diplomprojekt: ScanBuyGo
Mitglieder: Freimüller, Wiener, Sima, Walter, Maurovich
Klasse: 5BHIF

Datenschutzrichtlinie

Zuletzt aktualisiert am 18.12.2020

1. ÜBER SCANBUYGO

ScanBuyGo ("ScanBuyGo", "wir" oder "uns") ist eine mobile Selfcheckout App, welche es ermöglicht, direkt im Geschäft mit einem Mobilgerät Einkäufe zu tätigen, ohne in der Schlange stehen zu müssen.

ScanBuyGo heißt Sie herzlich willkommen und lädt Sie ein, mit dieser neuartigen App Einkäufe durchzuführen.

2. WELCHE INFORMATIONEN SAMMELN WIR?

Zweck	Verwendete Daten	Speicherdauer
Anmeldung in Ihr Benutzerkonto	E-Mail-Adresse, Passwort (verschlüsselt gespeichert)	Lebensdauer des Kontos
Rechnungslegung, Zahlung	Anrede, Vorname, Nachname, Adresse, UID, gekaufte Produkte, Zahlungsart	7 Jahre
Altersverifikation für den Kauf altersbeschränkter Artikel	Geburtsdatum	Lebensdauer des Kontos
Einkauf	Ausgewählter Store, eingescannte Produkte (Produktname und Anzahl)	Lebensdauer der Shopsession

Bei der Verwendung der App im Gastmodus werden nur eingescannte Produkte und Rechnungsdaten (Zahlungsart, ausgewählter Store, gekaufte Produkte) gespeichert.

Um der Speicherung von Daten minderjähriger vorzubeugen, erlauben wir keine Registrierungen von Benutzern vor dem 16. Lebensjahr. Dies wird mittels der Eingabe des Geburtsdatums während der Registrierung überprüft.

3. WO WERDEN IHRE DATEN VERARBEITET UND GESPEICHERT?

Ihre Daten werden von unserer API (= application programming interface) verarbeitet, diese API wird auf einem europäischen Server von Microsoft Azure gehostet. Nachdem die Daten verarbeitet worden sind, werden Sie in eine Heroku Datenbank, welche sich physisch in Europa befindet, gespeichert.

4. WOFÜR VERWENDEN WIR IHRE DATEN?

Es werden ausschließlich notwendige Daten gespeichert. Die Daten werden genutzt für

- Benutzerkonto (Registrierung),
- Erstellung von Rechnungen und
- Abwicklung von Zahlungen über (Stripe, Inc. 510 Townsend Street, San Francisco, CA 94103, USA, https://www.stripe.com).

Ihre Daten werden nicht an andere Parteien weitergegeben.

5. WIE LANGE SPEICHERN WIR DATEN?

Wir sammeln nur notwendige Daten damit die App funktioniert. Die folgenden Daten werden gespeichert (mit Speicherdauer):

- **Benutzerkonto:** Diese Daten werden so lange gespeichert, bis das Nutzerkonto von dem Inhaber gelöscht wird bzw. eine Löschung von diesem beantragt wird.
- **Zahlungs- und Rechnungsdaten:** Steuerrechtliche Daten über Zahlung und Rechnung müssen wir gesetzlich gemäß § 11 Abs 2 3. Unterabsatz UStG **7 Jahre** speichern. Darüberhinausgehend so lange sie für die Abgabenbehörde in einem anhängigen Verfahren von Bedeutung sind.
- **Sessiondaten:** Betreten Sie einen Shop, wird eine sogenannte Shopsession erstellt. Diese wird direkt nach Verlassen bzw. Abschluss der Zahlung oder bei Inaktivität nach spätestens 24 Stunden gelöscht.

6. WAS GEBEN WIR AN DRITTE WEITER?

Sollte Sie sich dazu entscheiden, den Newsletter eines Stores zu abonnieren, geben wir nur Ihre E-Mail-Adresse an den Storebesitzer weiter, damit der Inhaber Ihnen die Werbung zukommen lassen kann.

Da wir zur Zahlung den Drittanbieter Stripe (Stripe, Inc. 510 Townsend Street, San Francisco, CA 94103, USA, https://www.stripe.com) verwenden, geben wir zum Zeitpunkt der Zahlung die von Ihnen eingegebenen Zahlungsinformationen (Kreditkartennummer, Ablaufdatum, CVV sowie Ihren Namen und Ihre Adresse) an den Zahlungsanbieter weiter, damit die Zahlungsart verifiziert werden kann und auch eine Zahlung an den Merchant erfolgen kann.

Die Einkäufe, die über die App getätigt werden, werden den Storebesitzern weitergeleitet, hierbei wird die erzeugte Rechnung an den Händler weitergeleitet. Persönliche Daten, die über die Rechnung weitergeleitet werden, sind Vorname, Nachname, Anschrift und UID-Nummer.

7. COOKIES

Wir verwenden keine Cookies. Einzig verwenden wir sogenannte HTTP Header (nach IETF RFC 7231 Abschnitt 5), mit welchen wir anhand von zufällig generierten Schlüsseln speichern, ob und mit welchem Konto Sie angemeldet sind und ob und in welchem Geschäft Sie einkaufen. Diese Header werden lediglich von unserer Applikation verwendet und sind für Dritte nicht zugänglich.

8. WELCHE DATEN WERDEN LOKAL GESPEICHERT?

Bei Verwendung der App werden auch Daten auf Ihrem Gerät (Smartphone, Tablet,...) gespeichert, um die Funktionalität der App zu gewährleisten. Dafür speichern wir lokal die folgenden Daten:

- Session Token für die Benutzeranmeldung (Eine zufällig generierte, persönliche Zeichenkette, mit welcher Sie sich in Ihrem Benutzerkonto authentifizieren)
- Session Token für die Storeanmeldung (Eine zufällig generierte Zeichenkette, mit welcher Sie sich in einem Geschäft authentifizieren)
- Inhalt Ihres Warenkorbs (Produktnummer, Produktname, Produktbild, Menge)

Warenkorb- und Storedaten werden gelöscht, sobald sie einen Store verlassen und lokale Benutzerdaten werden gelöscht, sobald Sie sich aus Ihrem Benutzerkonto abmelden.

Falls Sie die Einkaufslistenfunktion verwenden wird Ihre Einkaufsliste ebenfalls lokal gespeichert.

9. DATENSCHUTZMASSNAHMEN

Die ScanBuyGo App verfügt über folgende Datenschutzmaßnahmen:

- **Regelmäßige Überprüfung/Überarbeitung:** Regelmäßig wird überprüft, ob es Datenleaks in der App gibt und falls ja, werden diese behoben. Des Weiteren werden bei neuen Datenschutzverordnungen/Datenschutzempfehlungen neue Änderung im Datenschutzkonzept der App getroffen.
- **App Infrastruktur:** Die App ist so entwickelt, dass jegliche Interaktionen zwischen App und Server verschlüsselt (SSL) sind und somit ein Abgreifen der Daten erschwert wird.
- **2-Faktor-Authentifizierung (2FA):** Die App ermöglicht eine 2FA einzustellen, welche bei jedem erfolgreichen Login Versuch einen Code an Ihr Smartphone sendet und somit sichergestellt wird, dass Sie sich einloggen wollen und niemand anderes.
- **Ändern von Passwort:** Nicht nur wird das Passwort verschlüsselt gespeichert, sondern bei angefragter Änderung von diesem, wird eine temporäre Session erstellt, welche es nur dem Smartphone erlaubt es zu ändern.

Wir können jedoch nicht versichern, dass die App zu 100 % sicher ist, nicht einmal das Internet ist zu 100 % sicher. Wir geben aber unser Bestes, um Ihre Daten zu schützen.

10. IHRE RECHTE IM ZUSAMMENHANG MIT PERSONENBEZOGENEN DATEN

Sie sind unter anderem nach geltendem Recht dazu berechtigt:

- Auskunft über Ihre gespeicherten personenbezogenen Daten zu erhalten
- Ihre gespeicherten personenbezogenen Daten zu aktualisieren
- Löschung oder Einschränkung der Verwendung Ihrer persönlichen Daten zu verlangen
- Datenübertragbarkeit zu verlangen
- Auskunft über Dritte welche Ihre Daten verarbeiten zu erhalten
- Beschwerde bei der Datenschutzbehörde zu erheben

Manche dieser Funktionen können Sie in den Einstellungen der App finden, wir stehen Ihnen dafür aber auch gerne per E-Mail an datenschutz@scanbuygo.at zur Verfügung.

Die Löschung, Einschränkung der Verarbeitung oder Bearbeitung Ihrer persönlichen Daten auf Belegen ist aus steuerrechtlichen Gründen unter Umständen nicht möglich, da wir Belegdaten sieben Jahre aufbewahren müssen und diese nicht verändert oder anonymisiert werden können.

11. PRIVACY POLICY UPDATES

Da sich unsere App weiterentwickelt können auch neue Services und Features dazukommen, aus diesem Grund werden wir, sofern notwendig, ab und an die Privacy Policy überarbeiten. Im Falle eines Updates der Privacy Policy werden Sie beim Start der App dazu aufgefordert, sich diese erneut durchzulesen und zu akzeptieren, sofern Sie mit den Aktualisierungen einverstanden sind.

Sollten sich Teile dieser Datenschutzrichtlinie als ungültig oder rechtswidrig erweisen tritt rein die ungültige Passage, nicht aber die gesamte Richtlinie, außer Kraft.

12. KONTAKT

Sollten Sie Fragen, Anfragen oder Bedenken in Bezug auf Ihre persönlichen Daten haben, kontaktieren Sie bitte unseren Datenschutzbeauftragten via E-Mail:
datenschutz@scanbuygo.at

Appendix F: Project Health Monitor

Project Health Monitor

Alexander Wiener, Dennis Sima, Paul Maurovich, Florian Freimüller, Lucas Walter

Project Health Monitor

by 2021-02-23

ITERATION	SPRINT 1	SPRINT 2	SPRINT 3	SPRINT 4	SPRINT 5	SPRINT 6	SPRINT 7	SPRINT 8	SPRINT 9
SCOPE [4]	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
SCALE [4]	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
SKILL	🟡 [3]	🟡 [3]	🟢	🟢	🟢	🟢	🟢	🟢	🟢
OUTSIDE FACTORS	🟢	🔴 [1]	🔴 [1]	🟡 [1]	🟢	🟡 [2]	🟡 [2]	🟡 [2]	🟢

Outside factors include school work, tests, issues with the partner company, etc.

Explanations:

[1] Due to a dispute with the partner company some pressure was put onto the project team including a raise in administrative work on the project. After the end of the cooperation the project's scope and (slightly) its scale were reduced, removing some features requested by the partner that would not be needed for the project much longer.

[2] In Sprints 6-8 due to more school stress, exams, etc. the decision was made to cut down on project work and instead work on school tasks and our diploma theses. This was possible as the project was ahead of its planned goals.

[3] Both the Frontend and Backend Teams had to learn about the technologies used in the project.

[4] The project's scale and scope were well-fixated before development on the product started (before Sprint 1). While some changes were made to it in Sprints 3 and 4 after parting ways with the partner company the scale was never under- or overwhelming and the scope was always clear during development.

Currently progress on the project is going well, we are meeting our milestones and sprint goals.

Appendix G: Vorstudien

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 4
Ausgangslage 4
Zielsetzung 4
Nutzenziele 4
Gewinn-/ Kostenorientierung 4
Prestige 4
Entdeckung 4
Projektziele 5
Ist-Analyse 6
Technologieanalyse 6
Frontend Analyse 6
Dart/Flutter 7
Vue Native 7
Ionic/React 7
Ionic/Angular 8
React Native 8
Xamarin 8
Ionic/Vue 8
Persönliche Interessen: 9
Fazit 9
Backend Analyse 10
ASP.NET Core 10
GraphQL 11
Spring Boot 11
Fazit 12
Warum Spring Boot und nicht ASP.NET Core? 12
Warum Kotlin und nicht Java? 12
Gradle oder Maven? 12
PostgreSQL oder MySQL? 12
Warum kein GraphQL? 13
Entscheidungstabellen 13
Persönliche Präferenzen 14
Technologieentscheidung Backend 14
Stakeholderanalyse 15

Benutzeranalyse 16
Kundenanalyse 17
Business to Business 17
Business to Customer 17
Konkurrenzproduktanalyse 18
Auswertung der Konkurrenzproduktanalyse (Alex) 21
Was wir derzeit nicht machen wollen: 21
Was wir machen wollen: 21
Was wir abändern würden: 21
SWOT 22
Lösungsentwurf 23
Technologie 23
Funktionen 23
Grobe Umsetzungsplanung des Projekts 24
Projektorganisation Team 24
Zeitplanung 25
Projektrisiko 26
Fachliche Risiken 26
Kostenrisiken 26
Terminrisiken 26
Ressourcenrisiken 26
Qualitätsrisiken 27
Personelle Risiken 27
Risikotabelle 28
Literaturverzeichnis 29
Abbildungsverzeichnis 29
Tabellenverzeichnis 29
Glossar 30

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)

<u>Wichtige Features:</u>

- 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.

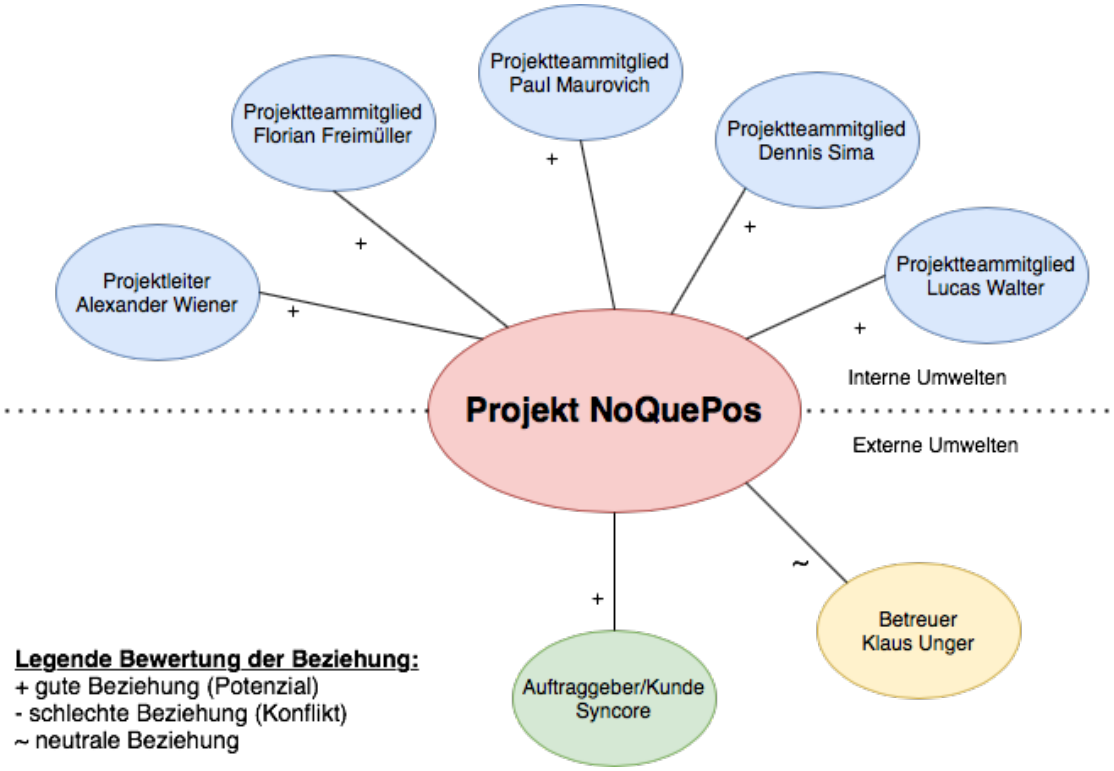


Abbildung 1: Stakeholder

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.

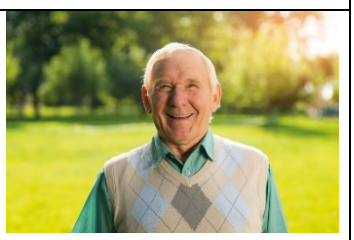
	Die Gelegenheitskäuferin	Der Firmeneinkäufer	Der Wochenendeinkäufer
			
Name	Stefanie Schmidt	Vincent Hawlik	Wilhelm Peter
Alter	18	30	70
Sozialleben	Stefanie ist Schülerin.	Vincent ist Angestellter und hat viel mit Technik zu tun.	Wilhelm ist Pensionist und hat Schwierigkeiten damit, Apps zu bedienen.
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.

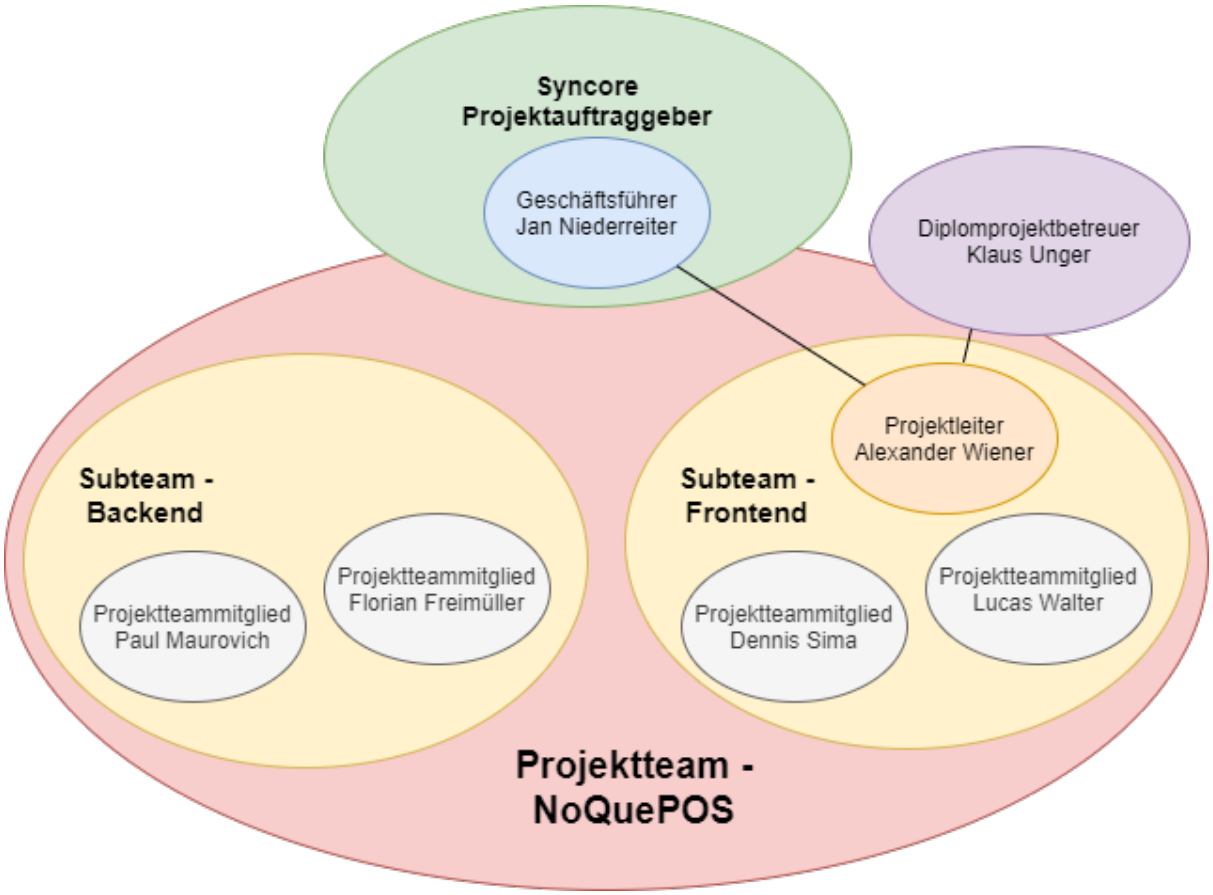


Abbildung 2: Projektteam

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

Abbildung 1: Stakeholder 15
Abbildung 2: Projektteam 24
Abbildung 3: Zeitachse des Projekts 25

Tabellenverzeichnis

Tabelle 1: Projektziele 5
Tabelle 2: Auswertungstabelle Frontend 9
Tabelle 3: Auswertung Spring Boot vs. ASP.NET 13
Tabelle 4: Auswertung: Java vs. Kotlin 13
Tabelle 5: Auswertung Gradle vs. Maven 13
Tabelle 6: Auswertung PostgreSQL vs. MySQL 13
Tabelle 7: Benutzeranalyse 16
Tabelle 8: Analyse Business 2 Customer 18
Tabelle 9: SWOT 22
Tabelle 10: Risikotabelle 28

Glossar

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

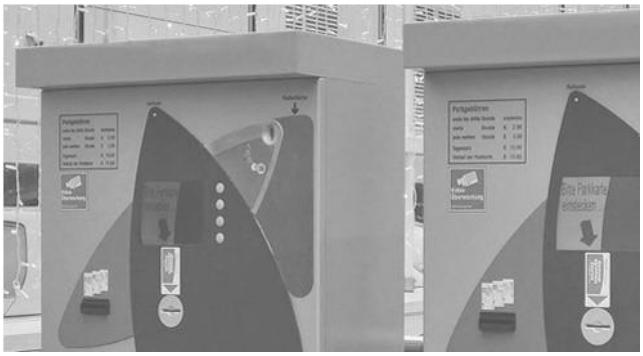
.NET Core	Eine Open Source Plattform für .NET-Apps
AI	Künstliche Intelligenz
Android	Ein mobiles Betriebssystem von Google
Annotation	hier: Metadaten in Quellcode
Device-)APIs	Application Programming Interface, mit dem auf Hardwarefunktionen zugegriffen werden kann.
Augmented Reality	Auf einem Kameravideo 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

Appendix H: Spezifikationen



ScanBuyGo: Spezifikation

Firma	[Firma] Spengergasse 20 1050 Wien
Thema	ScanBuyGo: Vorgabe Spezifikation
Datum	05.09.2020
Erstellt durch	Alexander Wiener wie17087@spengergasse.at

1 Änderungen 11
2 Begriffe und Abkürzungen 11
3 Über dieses Dokument 11
4 Vorlage und Formatierung 11
5 Spezifikationsaufbau Front End 11
6 Epic "Login" 11
6.1 Userstory "Anmeldung" 11
6.1.1 Beschreibung 11
6.1.2 Screendesign 11
6.1.3 Funktionen der Masken 12
6.1.4 Grafische Prozessdarstellung 12
6.1.5 Schnittstellen 14
6.1.6 Lokale Funktionen 14
6.1.7 Akzeptanzkriterien 14
6.1.8 Abgrenzung 14
6.1.9 Voraussetzungen 14
6.2 Userstory "Anmeldung mit externem Anbieter" 14
6.2.1 Beschreibung 14
6.2.2 Screendesign 14
6.2.3 Funktionen der Masken 15
6.2.4 Grafische Prozessdarstellung 15
6.2.5 Schnittstellen 16
6.2.6 Lokale Funktionen 16
6.2.7 Akzeptanzkriterien 16
6.2.8 Abgrenzung 16
7 Epic "Registrierung" 17
7.1 Userstory "Registrierung" 17
7.1.1 Beschreibung 17
7.1.2 Screendesign 17
7.1.3 Funktionen der Masken 18
7.1.4 Grafische Prozessdarstellung 19
7.1.5 Schnittstellen 20
7.1.6 Akzeptanzkriterien 20
7.1.7 Abgrenzung 20
7.2 Userstory "Registrierung mit externem Anbieter" 20
7.2.1 Beschreibung 20
7.2.2 Screendesign 20
7.2.3 Funktionen der Masken 21
7.2.4 Grafische Prozessdarstellung 22
7.2.5 Schnittstellen 24
7.2.6 Akzeptanzkriterien 24
7.2.7 Abgrenzung 24
7.3 Userstory "E-Mail bestätigen" 24
7.3.1 Beschreibung 24
7.3.2 Screendesign 24

7.3.3 Funktionen der Masken 25
7.3.4 Grafische Prozessdarstellung 25
7.3.5 Schnittstellen 26
7.3.6 Akzeptanzkriterien 27
7.3.7 Abgrenzung 27
7.3.8 Voraussetzungen 27
7.4 Userstory "Gast-Modus" 27
7.4.1 Beschreibung 27
7.4.2 Screendesign 27
7.4.3 Funktionen der Masken 27
7.4.4 Grafische Prozessdarstellung 28
7.4.5 Schnittstellen 28
7.4.6 Lokale Funktionen 28
7.4.7 Akzeptanzkriterien 28
7.4.8 Abgrenzung 28
7.5 Userstory "Passwort vergessen" 28
7.5.1 Beschreibung 28
7.5.2 Screendesign 28
7.5.3 Funktionen der Masken 31
7.5.4 Grafische Prozessdarstellung 31
7.5.5 Schnittstellen 32
7.5.6 Akzeptanzkriterien 33
7.5.7 Abgrenzung 33
7.5.8 Voraussetzungen 34
8 Epic „Shop Assignment“ 34
8.1 Userstory "QR-Code scannen" 34
8.1.1 Beschreibung 34
8.1.2 Screendesign 34
8.1.3 Funktionen der Masken 34
8.1.4 Grafische Prozessdarstellung 34
8.1.5 Schnittstellen 35
8.1.6 Lokale Funktionen 36
8.1.7 Akzeptanzkriterien 36
8.1.8 Abgrenzung 36
8.1.9 Voraussetzungen 36
8.2 Userstory "Shops in meiner Nähe" 36
8.2.1 Beschreibung 36
8.2.2 Screendesign ..... **Fehler! Textmarke nicht definiert.**
8.2.3 Funktionen der Masken 36
8.2.4 Schnittstellen 37
8.2.5 Lokale Funktionen 38
8.2.6 Akzeptanzkriterien 38
8.2.7 Abgrenzung 38
8.2.8 Voraussetzungen 38
8.3 Userstory "Store verlassen manuell" 38
8.3.1 Beschreibung 38
8.3.2 Screendesign 38
8.3.3 Funktionen der Masken 38
8.3.4 Grafische Prozessdarstellung 38

8.3.5 Schnittstellen 39
8.3.6 Lokale Funktionen 40
8.3.7 Akzeptanzkriterien 40
8.3.8 Abgrenzung 40
8.3.9 Voraussetzungen 40
8.4 Userstory "Automatische Abmeldung vom alten Store bei erneutem Scan" 40
8.4.1 Beschreibung 40
8.4.2 Funktionen der Masken 40
8.4.3 Grafische Prozessdarstellung 40
8.4.4 Schnittstellen 41
8.4.5 Lokale Funktionen 41
8.4.6 Akzeptanzkriterien 41
8.4.7 Abgrenzung 41
8.4.8 Voraussetzungen 41
9 Epic „Shopping Cart“ 42
9.1 Userstory "Produkt per Scan hinzufügen" 42
9.1.1 Beschreibung 42
9.1.2 Screendesign 42
9.1.3 Funktionen der Masken 42
9.1.4 Grafische Prozessdarstellung 42
9.1.5 Schnittstellen 44
9.1.6 Akzeptanzkriterien 44
9.1.7 Abgrenzung 44
9.1.8 Voraussetzungen 44
9.2 Userstory "Produkt per Kategorie hinzufügen" 44
9.2.1 Beschreibung 44
9.2.2 Screendesign 44
9.2.3 Funktionen der Masken 45
9.2.4 Grafische Prozessdarstellung 45
9.2.5 Schnittstellen 45
9.2.6 Akzeptanzkriterien 46
9.2.7 Abgrenzung 46
9.2.8 Voraussetzungen 46
9.3 Userstory "Produkt per Liste hinzufügen" 46
9.3.1 Beschreibung 46
9.3.2 Screendesign 46
9.3.3 Funktionen der Masken 46
9.3.4 Grafische Prozessdarstellung 47
9.3.5 Schnittstellen 47
9.3.6 Akzeptanzkriterien 48
9.3.7 Abgrenzung 48
9.3.8 Voraussetzungen 48
9.4 Userstory "Produkt hinzufügen/Stückzahl" 48
9.4.1 Beschreibung 48
9.4.2 Screendesign 48
9.4.3 Funktionen der Masken 49
9.4.4 Lokale Funktionen 49
9.4.5 Akzeptanzkriterien 49
9.4.6 Abgrenzung 49

9.4.7 Voraussetzungen 49
9.5 Userstory "Produktdetails nach Scan" 49
9.5.1 Beschreibung 49
9.5.2 Screendesign 50
9.5.3 Funktionen der Masken 50
9.5.4 Grafische Prozessdarstellung 50
9.5.5 Lokale Funktionen 50
9.5.6 Akzeptanzkriterien 50
9.5.7 Abgrenzung 50
9.5.8 Voraussetzungen 51
9.6 Userstory "Warenkorb ansehen" 51
9.6.1 Beschreibung 51
9.6.2 Screendesign 51
9.6.3 Funktionen der Masken 51
9.6.4 Grafische Prozessdarstellung 52
9.6.5 Lokale Funktionen 52
9.6.6 Akzeptanzkriterien 52
9.6.7 Abgrenzung 52
9.6.8 Voraussetzungen 52
9.7 Userstory "Produkt im Warenkorb entfernen" 52
9.7.1 Beschreibung 52
9.7.2 Screendesign 52
9.7.3 Funktionen der Masken 52
9.7.4 Lokale Funktionen 52
9.7.5 Akzeptanzkriterien 52
9.7.6 Abgrenzung 52
9.7.7 Voraussetzungen 53
9.8 Userstory "Warenkorb Produktstückzahl erhöhen/verringern" 53
9.8.1 Beschreibung 53
9.8.2 Screendesign 53
9.8.3 Funktionen der Masken 53
▪ Lokale Funktionen 53
9.8.4 Akzeptanzkriterien 53
9.8.5 Abgrenzung 53
9.8.6 Voraussetzungen 53
9.9 Userstory "Produktdetails (Warenkorb)" 53
9.9.1 Beschreibung 53
9.9.2 Screendesign .... **Fehler! Textmarke nicht definiert.**
9.9.3 Funktionen der Masken 53
9.9.4 Lokale Funktionen 54
9.9.5 Akzeptanzkriterien 54
9.9.6 Abgrenzung 54
9.9.7 Voraussetzungen 54
9.10 Userstory "Warenkorb löschen" 54
9.10.1 Beschreibung 54
9.10.2 Screendesign 54
9.10.3 Funktionen der Masken 54
9.10.4 Lokale Funktionen 54
9.10.5 Akzeptanzkriterien 54

9.10.6 Abgrenzung 54
9.10.7 Voraussetzungen 55
9.11 Userstory "Warenkorb bearbeiten" 55
9.11.1 Beschreibung 55
9.11.2 Screendesign 55
9.11.3 Funktionen der Masken 55
9.11.4 Lokale Funktionen 56
9.11.5 Akzeptanzkriterien 56
9.11.6 Abgrenzung 56
9.11.7 Voraussetzungen 56
9.12 Userstory "Biometrische Daten" 56
9.12.1 Beschreibung 56
9.12.2 Grafische Prozessdarstellung 56
9.12.3 Lokale Funktionen 56
9.12.4 Akzeptanzkriterien 56
9.12.5 Abgrenzung 56
10 Epic „Start Page“ 56
10.1 Userstory "Home" 56
10.1.1 Beschreibung 56
10.1.2 Screendesign 56
10.1.3 Funktionen der Masken 56
10.1.4 Grafische Prozessdarstellung 57
10.1.5 Schnittstellen 57
10.1.6 Lokale Funktionen 57
10.1.7 Akzeptanzkriterien 57
10.1.8 Abgrenzung 57
10.1.9 Voraussetzungen 57
11 Epic „Zahlung“ 57
11.1 Userstory "Zur Kasse" 57
11.1.1 Beschreibung 57
11.1.2 Screendesign 57
11.1.3 Funktionen der Masken 57
11.1.4 Grafische Prozessdarstellung 57
11.1.5 Schnittstellen 58
11.1.6 Lokale Funktionen 58
11.1.7 Akzeptanzkriterien 58
11.1.8 Abgrenzung 58
11.1.9 Voraussetzungen 58
11.2 Userstory "Bezahlen" 59
11.2.1 Beschreibung 59
11.2.2 Screendesign 59
11.2.3 Funktionen der Masken 59
11.2.4 Grafische Prozessdarstellung 59
11.2.5 Schnittstellen 59
11.2.6 Lokale Funktionen 60
Es wird geprüft im vorab, ob die eingegebenen Daten in Textfelder der definierten Richtlinien entsprechen. 60
Beim Aktivieren der Slide Checkbox, werden die eingegebenen Kartendaten lokal auf dem Gerät gespeichert. 60
Eine Bestätigung von Zahlung wird dem Benutzer angezeigt, erst wenn das Response von VerifyPayment positiv ist („success“: true) 60

11.2.7 Akzeptanzkriterien 60
Die Zahlung bei Stripe erfolgreich ist. 60
Die Zahlungsbestätigung ist auf FE ersichtlich 60
11.2.8 Abgrenzung 60
Anlegen eines Handlerkonto bei Stripe 60
11.2.9 Voraussetzungen 60
Der Handler muss ein gültiges Konto bei Stripe angelegt haben. 60
12 Spezifikationsaufbau für Back End 60
13 Epic Registrierung 61
13.1 Userstory Registrierung 61
13.1.1 Beschreibung 61
13.1.2 Schnittstellen 61
13.1.3 Datenbankstrukturen DB 61
13.1.4 Lokale Funktionen 62
13.1.5 Akzeptanzkriterien 62
13.1.6 Abgrenzung zu anderen Prozessen 62
13.1.7 Voraussetzungen 62
13.1.8 Grafische Prozessdarstellung 62
13.2 Userstory "Registrierung mit externem Anbieter" 63
13.2.1 Beschreibung 63
13.2.2 Schnittstellen 63
13.2.3 Lokale Funktionen 63
13.2.4 Akzeptanzkriterien 64
13.2.5 Abgrenzung zu anderen Prozessen 64
13.2.6 Grafische Prozessdarstellung 64
13.3 Userstory "E-Mail bestätigen" 64
13.3.1 Beschreibung 64
13.3.2 Schnittstellen 64
13.3.3 Datenbankstrukturen DB 64
13.3.4 Lokale Funktionen 64
13.3.5 Akzeptanzkriterien 65
13.3.6 Abgrenzung zu anderen Prozessen 65
13.3.7 Voraussetzungen 65
13.3.8 Grafische Prozessdarstellung 65
14 Epic “Login” 67
14.1 Userstory "Login" 67
14.1.1 Beschreibung 67
14.1.2 Schnittstellen 67
14.1.3 Datenbankstrukturen DB 68
14.1.4 Lokale Funktionen 68
14.1.5 Akzeptanzkriterien 68
14.1.6 Abgrenzung zu anderen Prozessen 68
14.1.7 Voraussetzungen 68
14.1.8 Grafische Prozessdarstellung 68
14.2 Userstory „Login mit externem Anbieter“ 69
14.2.1 Beschreibung 69
14.2.2 Schnittstellen 69
14.2.3 Lokale Funktionen 69
14.2.4 Akzeptanzkriterien 70

14.2.5 Abgrenzung zu anderen Prozessen 70
14.2.6 Voraussetzungen 70
14.2.7 Grafische Prozessdarstellung 70
14.3 Userstory "Gastmodus" 70
14.3.1 Beschreibung 70
14.3.2 Schnittstellen 70
14.3.3 Datenbankstrukturen DB 71
14.3.4 Lokale Funktionen 71
14.3.5 Akzeptanzkriterien 71
14.3.6 Abgrenzung zu anderen Prozessen 71
14.3.7 Voraussetzungen 71
14.3.8 Grafische Prozessdarstellung 71
14.4 Userstory "Passwort vergessen" 71
14.4.1 Beschreibung 71
14.4.2 Schnittstellen 71
14.4.3 Datenbankstrukturen DB 72
14.4.4 Lokale Funktionen 72
14.4.5 Akzeptanzkriterien 72
14.4.6 Abgrenzung zu anderen Prozessen 73
14.4.7 Voraussetzungen 73
14.4.8 Grafische Prozessdarstellung 73
15 Epic „Shop Assignment“ 74
15.1 Userstory "QR-Code scannen" 74
15.1.1 Beschreibung 74
15.1.2 Schnittstellen 74
15.1.3 Datenbankstrukturen DB 75
15.1.4 Lokale Funktionen 77
15.1.5 Akzeptanzkriterien 77
15.1.6 Abgrenzung zu anderen Prozessen 77
15.1.7 Voraussetzungen 77
15.1.8 Grafische Prozessdarstellung 77
15.2 Userstory "Shops in meiner Nähe" 77
15.2.1 Beschreibung 77
15.2.2 Schnittstellen 78
15.2.3 Lokale Funktionen 78
15.2.4 Akzeptanzkriterien 78
15.2.5 Abgrenzung zu anderen Prozessen 78
15.2.6 Voraussetzungen 78
15.2.7 Grafische Prozessdarstellung 78
15.3 Userstory "Shop manuell verlassen" 79
15.3.1 Beschreibung 79
15.3.2 Schnittstellen 79
15.3.3 Datenbankstrukturen DB 79
15.3.4 Lokale Funktionen 79
15.3.5 Akzeptanzkriterien 79
15.3.6 Abgrenzung zu anderen Prozessen 79
15.3.7 Voraussetzungen 80
15.3.8 Grafische Prozessdarstellung 80
16 Epic „Shopping Cart“ 80

16.1 Userstory "Produkt per Scan hinzufügen" 80
16.1.1 Beschreibung 80
16.1.2 Schnittstellen 80
16.1.3 Datenbankstrukturen DB 80
16.1.4 Lokale Funktionen 81
16.1.5 Akzeptanzkriterien 81
16.1.6 Abgrenzung zu anderen Prozessen 81
16.1.7 Voraussetzungen 81
16.1.8 Grafische Prozessdarstellung 81
16.2 Userstory "Produkt per Kategorie hinzufügen" 82
16.2.1 Beschreibung 82
16.2.2 Schnittstellen 82
16.2.3 Datenbankstrukturen DB 83
16.2.4 Lokale Funktionen 83
16.2.5 Akzeptanzkriterien 83
16.2.6 Abgrenzung zu anderen Prozessen 83
16.2.7 Voraussetzungen 83
16.2.8 Grafische Prozessdarstellung 83
16.3 Userstory "Produkt per Liste hinzufügen" 83
16.3.1 Beschreibung 83
16.3.2 Schnittstellen 83
16.3.3 Datenbankstrukturen DB 84
16.3.4 Lokale Funktionen 84
16.3.5 Akzeptanzkriterien 84
16.3.6 Abgrenzung zu anderen Prozessen 84
16.3.7 Voraussetzungen 84
16.3.8 Grafische Prozessdarstellung 84
16.4 Userstory „Produkt hinzufügen/Stückanzahl“ 84
16.5 Userstory „Produktdetails nach Scan“ 84
16.6 Userstory „Warenkorb ansehen“ 84
16.7 Userstory „Produkt im Warenkorb ansehen“ 84
16.8 Userstory „Produkt im Warenkorb entfernen“ 85
16.9 Userstory „Warenkorb Produktstückzahl erhöhen/verringern“ 85
16.10 Userstory „Produktdetails (Warenkorb) 85
16.11 Userstory „Warenkorb löschen“ 85
16.12 Userstory „Warenkorb bearbeiten“ 85
16.13 Userstory „Biometrische Daten“ 85
17 Epic „Start Page“ 85
17.1 Userstory „Home“ 85
18 Epic „Zahlung“ 85
18.1 Userstory "Zur Kasse" 85
18.1.1 Beschreibung 85
18.1.2 Datenbankstrukturen DB 85
18.1.3 Lokale Funktionen 86
18.1.4 Externe Schnittstellen 86
18.1.5 Akzeptanzkriterien 86
18.1.6 Abgrenzung zu anderen Prozessen 86
18.1.7 Voraussetzungen 86
18.1.8 Grafische Prozessdarstellung 86

18.2 Userstory "Bezahlen" 86
18.2.1 Beschreibung 86
18.2.2 Datenbankstrukturen DB 87
18.2.3 Lokale Funktionen 87
18.2.4 Externe Schnittstellen 87
18.2.5 Akzeptanzkriterien 87
18.2.6 Abgrenzung zu anderen Prozessen 87
18.2.7 Voraussetzungen 87
18.2.8 Grafische Prozessdarstellung 87

1 Änderungen

Version	Wann	Wer	Änderung
V01	05.09.2020	JN	Initial Release

2 Begriffe und Abkürzungen

Begriff	Erklärung
Epic	Anforderung an einen Prozess, in unserer Definition kann ein Epic mehrere User Stories enthalten
User Story	Verbale, teilweise auch umgangssprachliche, nicht unbedingt technische Anforderungsbeschreibung eines Teilprozesses, der in sich abgeschlossen ist – siehe https://de.wikipedia.org/wiki/User_Story

3 Über dieses Dokument

In diesem Dokument werden die Vorgaben für die Erstellung der ScanBuyGo Spezifikation (Phase 1/Prototyp) beschrieben.

4 Vorlage und Formatierung

5 Spezifikationsaufbau Front End

Der Aufbau der Struktur für die Spezifikation der Front End Prozesse soll wie folgt aufgebaut werden:

- Epic 1
    - Userstory 1
        - Inhaltliche Beschreibung
        - Screendesign
        - Funktionsbeschreibung auf Basis des Screendesigns
        - Grafische Prozessdarstellung
        - Schnittstellen
            - Back End WS Funktion A
            - ....
            - Back ENd WS Funktion Z
        - Lokale Funktionen
        - Akzeptanzkriterien
        - Abgrenzung zu anderen Prozessen
        - Voraussetzungen
    - Userstory 2
        - …
    - Userstory N
- Epic 2
- …
- Epic N

Beispiel und Erklärung der einzelnen Struktur-Ebenen (Login)

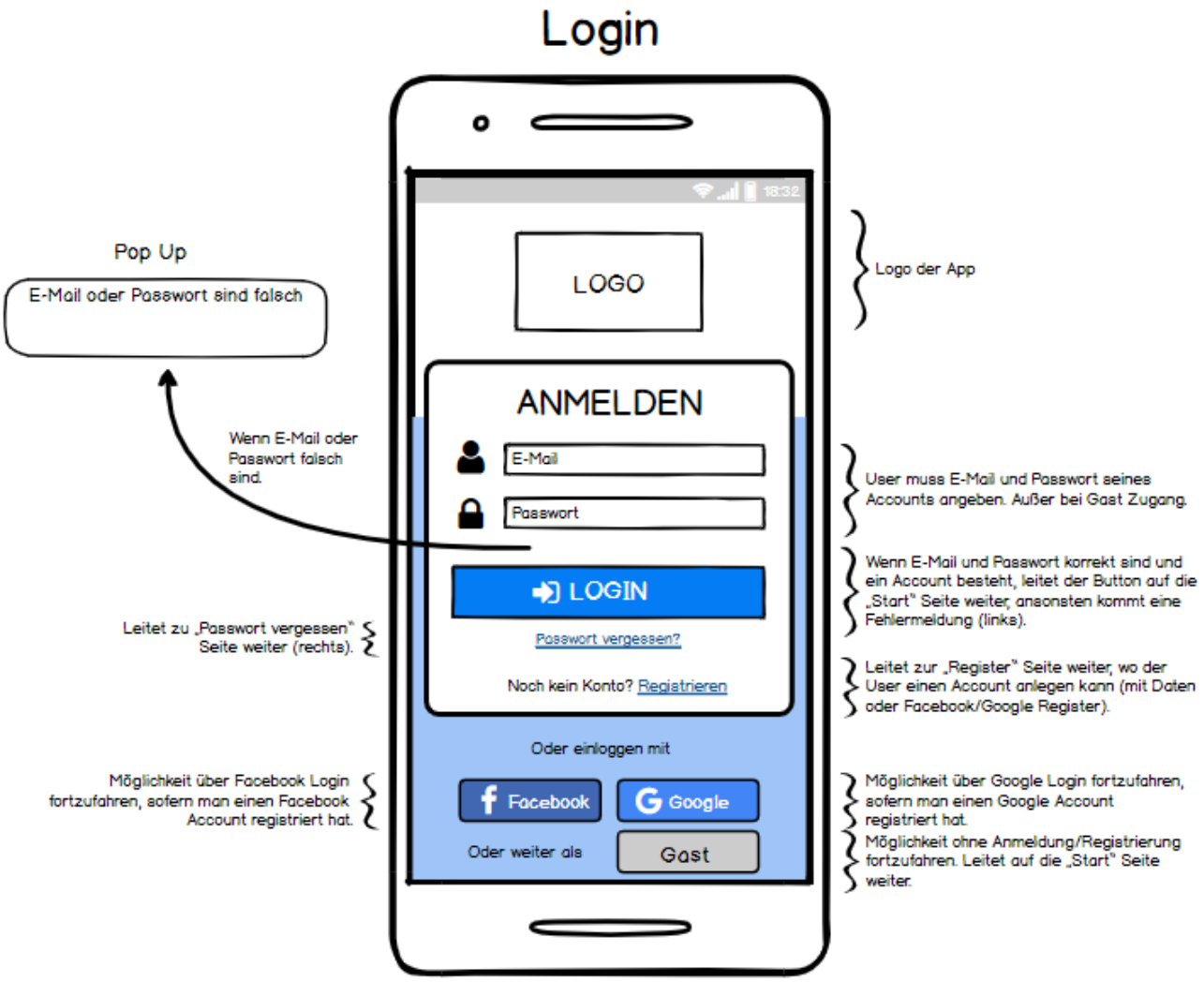
6 Epic "Login"

6.1 Userstory "Anmeldung"

6.1.1 Beschreibung

Als Kunde kann ich mich nach der einmaligen Registrierung mit meiner E-Mail und meinem Passwort einloggen, sodass ich als eingeloggter User alle Funktionalitäten der App nutzen kann.

6.1.2 Screendesign



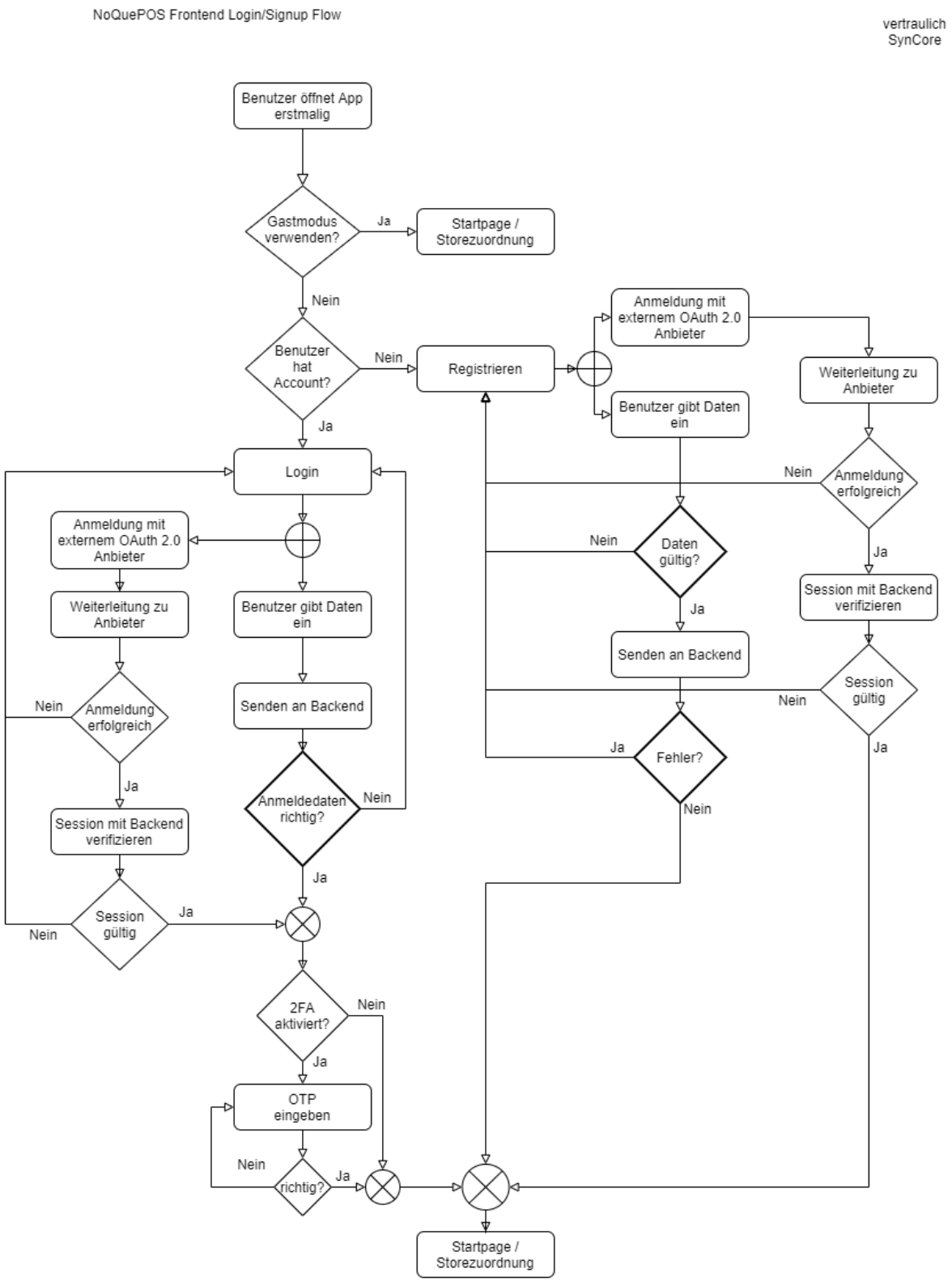
6.1.3 Funktionen der Masken

Tabellarische Darstellung: Button oder Feld mit der entsprechenden Funktionsbeschreibung auf Niveau der User Story!

Zum Beispiel:

Funktion	Beschreibung
Button „Login“	Schickt die vom Anwender eingegeben Credentials an die Funktion „Login“ vom NQP Webservice.
Textbox „E-Mail“	Es wird überprüft, ob eine gültige E-Mail eingegeben wurde (xxx@yyy.zzz)
Textbox „Passwort“	Eingegebener Text wird versteckt
Button „Passwort vergessen“	Leitet den Benutzer zur „Passwort vergessen“ Seite weiter

6.1.4 Grafische Prozessdarstellung



Datenvalidierung

- Email gültig (.*@.*\..*)
- Passwort erfüllt Richtlinien
  - 8+ Zeichen
  - [A-Z]
  - [0-9]
  - min. 1 Sonderzeichen
- ReCaptcha als Spamschutz?

Externe OAuth Anbieter

Ermöglichen Benutzern, sich mit Diensten wie Google, Facebook, Twitter, Apple anzumelden. Ein genereller Ablauf ist hier dargestellt, der Benutzer wird auf eine externe Seite des Betreibers weitergeleitet und stimmt der Verwendung seiner Daten zu (Zugriffe auf Email und Name sind dadurch möglich, ohne dass der Benutzer diese eingeben muss)

Two-Factor Authentification (2FA)

Benötigt eine weitere Information vom User, bevor er sich anmelden kann oder gravierende Änderungen an seinem Konto vornehmen kann. Diese Information ist ein One Time Password (OTP) oder ein Time based One Time Password, abhängig davon, ob es nur bei Bedarf generiert und z.B. per SMS oder E-Mail versandt wird oder ständig von einer App wie Google Authenticator oder Authy generiert wird.

6.1.5 Schnittstellen

6.1.5.1 "Login" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "password": "P@ssw0rd"
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "sessionId": ""
}
```

6.1.6 Lokale Funktionen

Die sessionId wird lokal gespeichert und bei späteren Requests im Header übergeben.

6.1.7 Akzeptanzkriterien

- Ein existierender Benutzer kann sich anmelden.
- Die Session ID wird lokal gespeichert

6.1.8 Abgrenzung

Registierung, Passwort vergessen, Startpage, Login mit externen Anbietern

6.1.9 Voraussetzungen

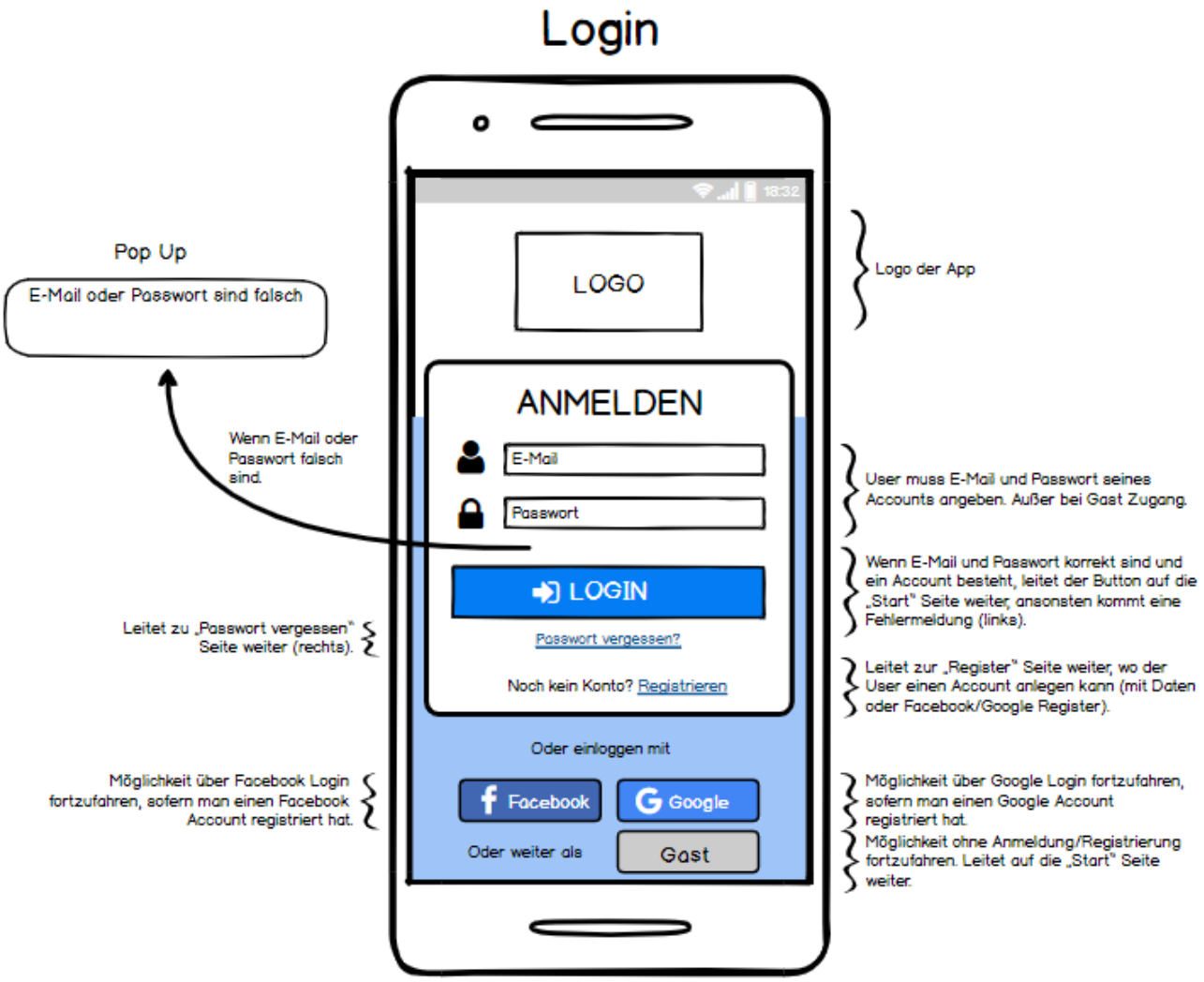
Registrierung

6.2 Userstory "Anmeldung mit externem Anbieter"

6.2.1 Beschreibung

Als Kunde kann ich mich mithilfe der Google oder Facebook Anmelde-Funktion einloggen, sodass ich als eingeloggter User alle Funktionalitäten der App nutzen kann.

6.2.2 Screendesign



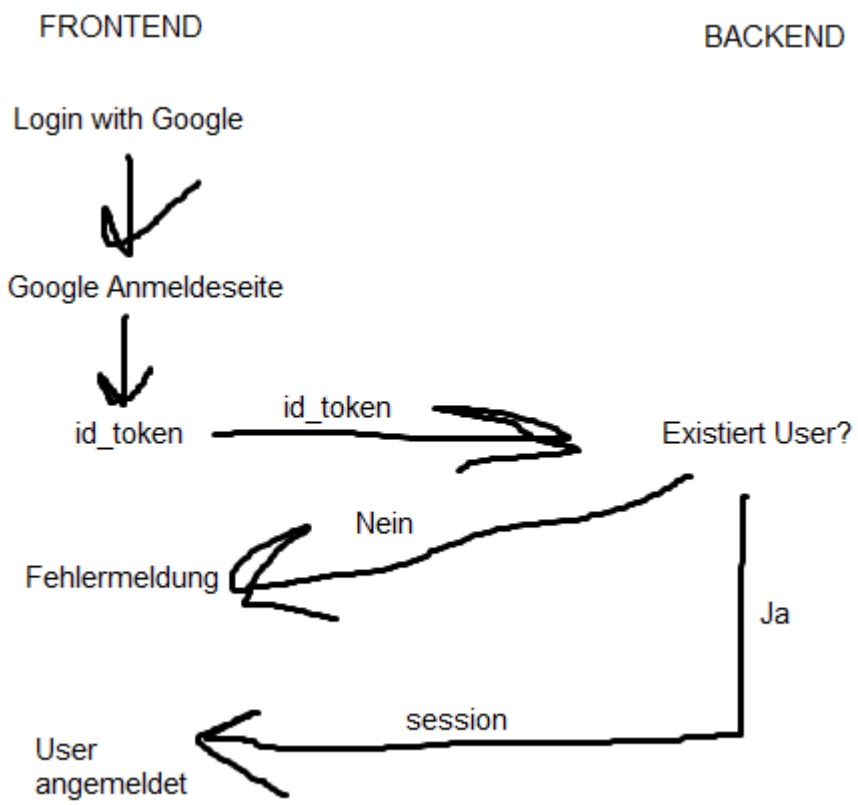
6.2.3 Funktionen der Masken

Tabellarische Darstellung: Button oder Feld mit der entsprechenden Funktionsbeschreibung auf Niveau der User Story!

Zum Beispiel:

Funktion	Beschreibung
Button „Facebook“	Schickt die vom Anwender eingegeben Credentials an die Funktion „Login“ vom NQP Webservice.
Button „Google“	Es wird überprüft, ob eine gültige E-Mail eingegeben wurde (xxx@yyy.zzz)
Button „Apple“	Eingegebener Text wird versteckt

6.2.4 Grafische Prozessdarstellung



6.2.5 Schnittstellen

6.2.5.1 "LoginUserExternal" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "provider": "google", // google, facebook, apple,...
    "token": "HG3kJGL3KJGH3kg234" // Google id_token
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "sessionId": ""
}
```

6.2.6 Lokale Funktionen

Die sessionId wird lokal gespeichert und bei späteren Requests im Header übergeben.

6.2.7 Akzeptanzkriterien

- Ein existierender Benutzer kann sich anmelden.
- Die Session ID wird lokal gespeichert

6.2.8 Abgrenzung

Registierung, Passwort vergessen, Startpage, Login

7 Epic "Registrierung"

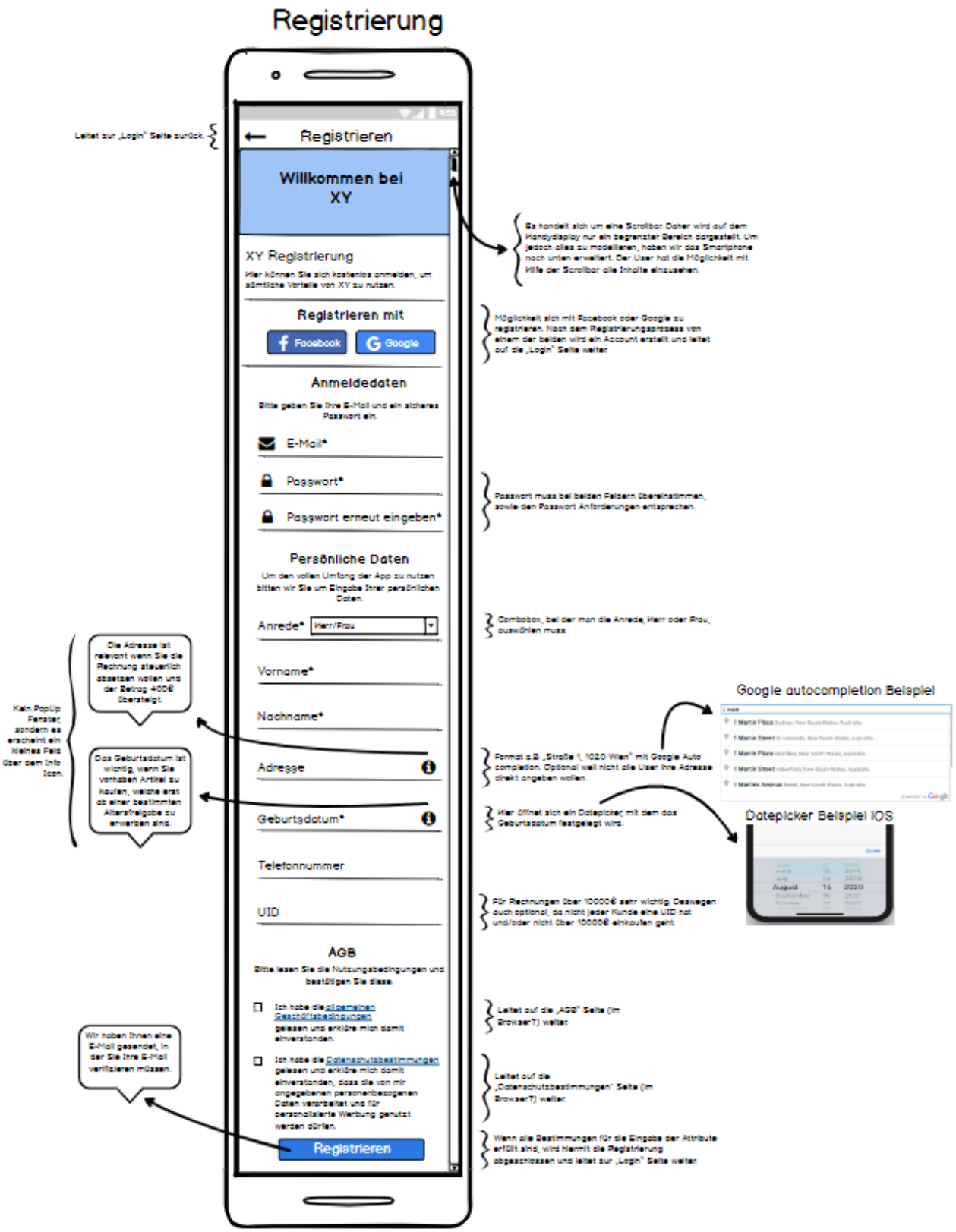
7.1 Userstory "Registrierung"

7.1.1 Beschreibung

" Als Kunde kann ich mich mit meinen persönlichen Daten* registrieren, sodass ich einen Account besitze.

* Vorname, Zuname, E-Mail, Geburtsdatum, Straße, Hausnummer, …"

7.1.2 Screendesign



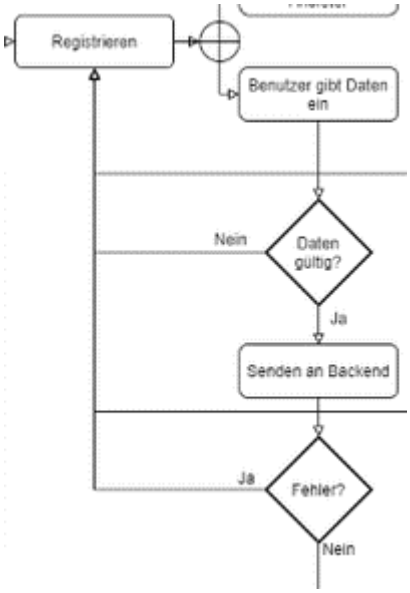
7.1.3 Funktionen der Masken

Gültige Felder werden mit einem Häkchen versehen

Funktion	Beschreibung
Textbox „E-Mail“	Es wird überprüft, ob eine gültige E-Mail eingegeben wurde (xxx@yyy.zzz)

Textbox „Passwort“	Eingegebener Text wird versteckt, mindestens 8 Stellen, mind. ein Großbuchstabe, mind. eine Zahl
Textbox „Passwort wiederholen“	Inhalt stimmt mit der Textbox „Passwort“ überein
Dropdown „Anrede“	eines von „Herr“, „Frau“ oder „Andere“ ausgewählt
Textbox „Vorname“, Textbox „Nachname“	Nicht leer, nicht länger als 200 Zeichen
Textbox „Adresse“	Wird mit der Google Maps Places API validiert, eine Schaltfläche bietet Informationen
Datepicker „Geburtsdatum“	Muss angegeben werden, muss 14 Jahre zurückliegen, eine Schaltfläche bietet Informationen
Zahleneingabefeld „Telefonnummer“	Nur Zahlen, maximale Länge 50 Zeichen
Textbox „UID“	maximale Länge 20 Zeichen
Checkbox „AGB“	muss aktiviert sein, Link „allgemeine Geschäftsbedingungen“ führt zu einer Website
Checkbox „Datenschutz“	muss aktiviert sein, Link „Datenschutzbestimmungen“ führt zu einer Website
Button „Registrieren“	Deaktiviert während Felder unvollständig oder falsch sind, bei Klick wird im Backend die Funktion RegisterUser aufgerufen und leitet nach erfolgreicher Anmeldung zur „E-Mail verifizieren“ Seite weiter

7.1.4 Grafische Prozessdarstellung



7.1.5 Schnittstellen

7.1.5.1 "RegisterUser" Funktion im Webservice des Back End

POST Request

```
{
    "requestDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "password": "P@ssw0rd",
    "gender": "Herr",
    "firstname": "Freddi",
    "lastname": "Müller",
    "address": "Straße 1, 1020 Wien",
    "birthdate": "2020-09-07",
    "phone": "068120160041",
    "vatid": "ATU 57615634",
    "terms": true,
    "privacy": true
}
```

Response:

```
{
    "success": true,
    "responseDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "countdownTime": 30
}
```

7.1.6 Akzeptanzkriterien

- Wenn sich der Kunde erfolgreich registrieren kann
- Wenn das Benutzerprofil in der Datenbank erstellt ist

7.1.7 Abgrenzung

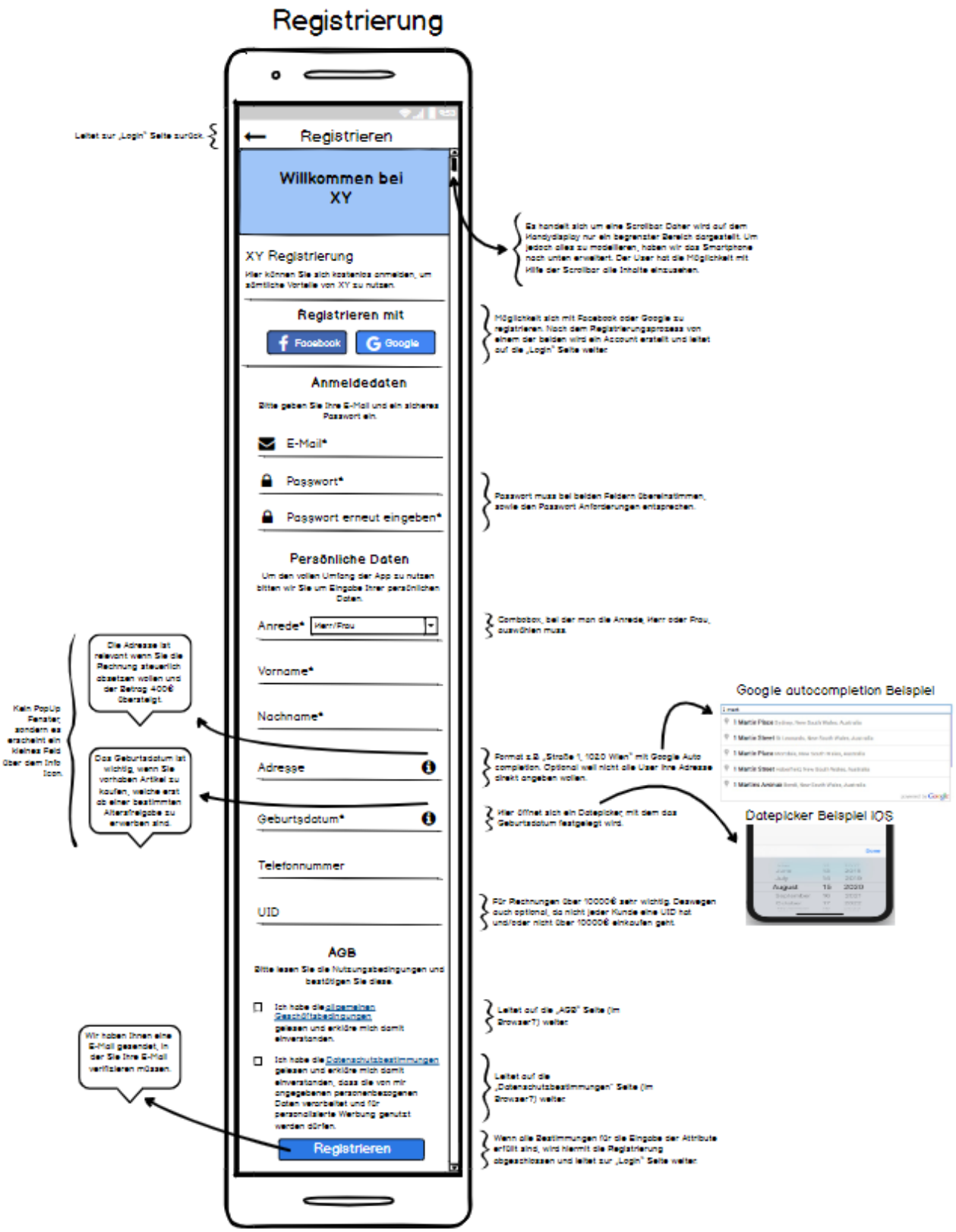
- E-Mail bestätigen
- Anmeldung
- Registrierung mit externem Anbieter

7.2 Userstory "Registrierung mit externem Anbieter"

7.2.1 Beschreibung

„Als Kunde kann ich mich mit meinen bereits bestehenden Facebook oder Google Account Daten registrieren, sodass ein neuer Account mit meinen externen Facebook oder Google Daten erstellt wird.“

7.2.2 Screendesign



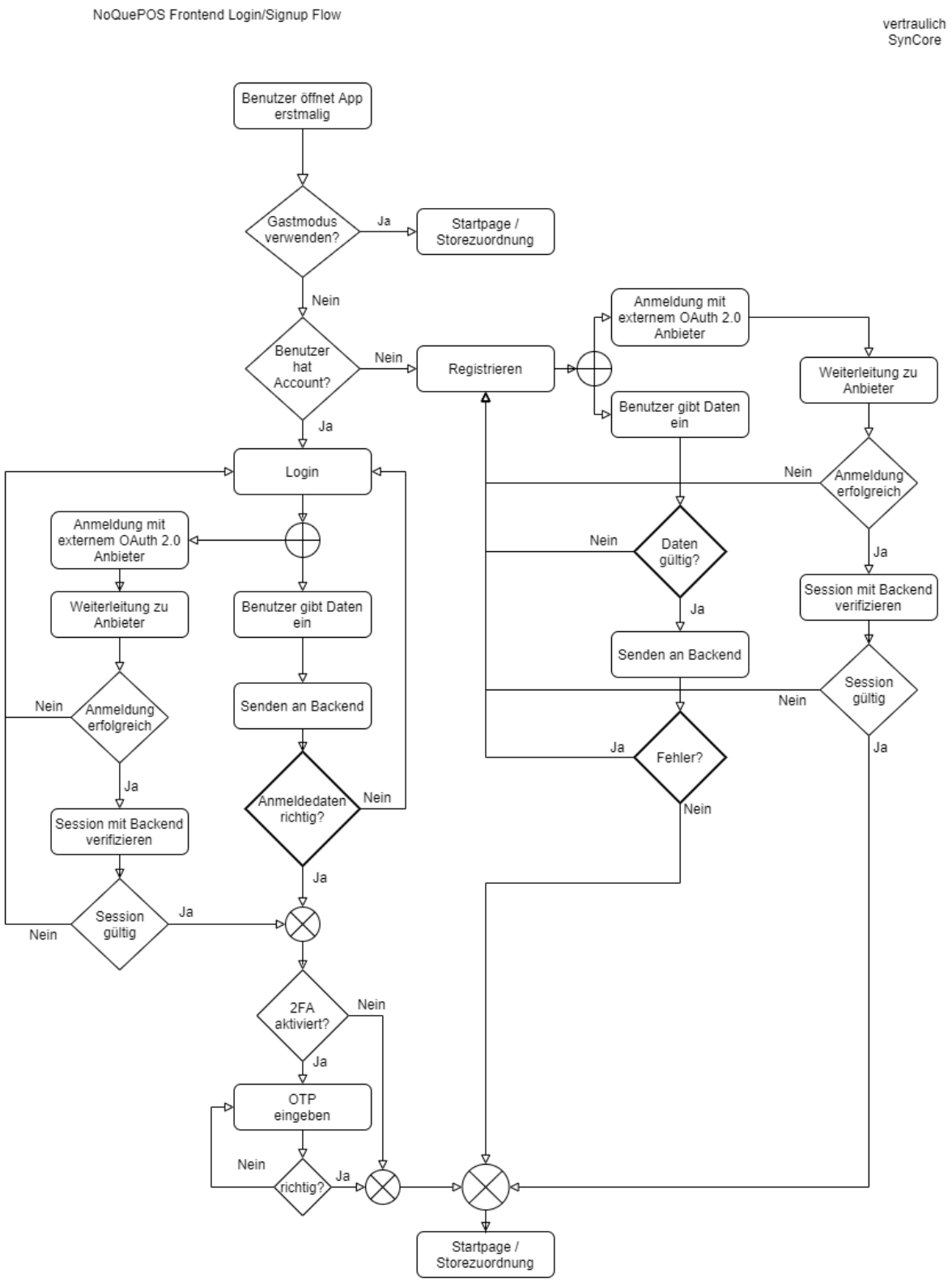
7.2.3 Funktionen der Masken

Vorhandene Felder sind mit Daten des Providers befüllt und können bearbeitet werden.

Funktion	Beschreibung
Textbox „E-Mail“	Ausgegraut, nur zur Anzeige

Dropdown „Anrede“	eines von „Herr“, „Frau“ oder „Andere“ ausgewählt
Textbox „Vorname“, Textbox „Nachname“	Nicht leer, nicht länger als 200 Zeichen
Textbox „Adresse“	Wird mit der Google Maps Places API validiert, eine Schaltfläche bietet Informationen
Datepicker „Geburtsdatum“	Muss angegeben werden, muss 14 Jahre zurückliegen, eine Schaltfläche bietet Informationen
Zahleneingabefeld „Telefonnummer“	Nur Zahlen, maximale Länge 50 Zeichen
Textbox „UID“	maximale Länge 20 Zeichen
Checkbox „AGB“	muss aktiviert sein, Link „allgemeine Geschäftsbedingungen“ führt zu einer Website
Checkbox „Datenschutz“	muss aktiviert sein, Link „Datenschutzbestimmungen“ führt zu einer Website
Button „Registrieren“	Deaktiviert während Felder unvollständig oder falsch sind, bei Klick wird im Backend die Funktion RegisterUserExternal aufgerufen

7.2.4 Grafische Prozessdarstellung



Datenvalidierung

- Email gültig (.*@.*\..*)
- Passwort erfüllt Richtlinien
  - 8+ Zeichen
  - [A-Z]
  - [0-9]
  - min. 1 Sonderzeichen
- ReCaptcha als Spamschutz?

Externe OAuth Anbieter

Ermöglichen Benutzern, sich mit Diensten wie Google, Facebook, Twitter, Apple anzumelden. Ein genereller Ablauf ist hier dargestellt, der Benutzer wird auf eine externe Seite des Betreibers weitergeleitet und stimmt der Verwendung seiner Daten zu (Zugriffe auf Email und Name sind dadurch möglich, ohne dass der Benutzer diese eingeben muss)

Two-Factor Authentification (2FA)

Benötigt eine weitere Information vom User, bevor er sich anmelden kann oder gravierende Änderungen an seinem Konto vornehmen kann. Diese Information ist ein One Time Password (OTP) oder ein Time based One Time Password, abhängig davon, ob es nur bei Bedarf generiert und z.B. per SMS oder E-Mail versandt wird oder ständig von einer App wie Google Authenticator oder Authy generiert wird.

7.2.5 Schnittstellen

7.2.5.1 "RegisterUserExternal" Funktion im Webservice des Back End

Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "password": "P@ssw0rd",
    "gender": "Herr",
    "firstname": "Freddi",
    "lastname": "Müller",
    "address": "Straße 1, 1020 Wien",
    "birthdate": "2020-09-07",
    "phone": "068120160041",
    "vatid": "ATU 57615634",
    "terms": true,
    "privacy": true
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "email": "email@example.com"
}
```

7.2.6 Akzeptanzkriterien

- Wenn sich der Kunde erfolgreich registrieren kann
- Wenn das Benutzerprofil in der Datenbank erstellt ist

7.2.7 Abgrenzung

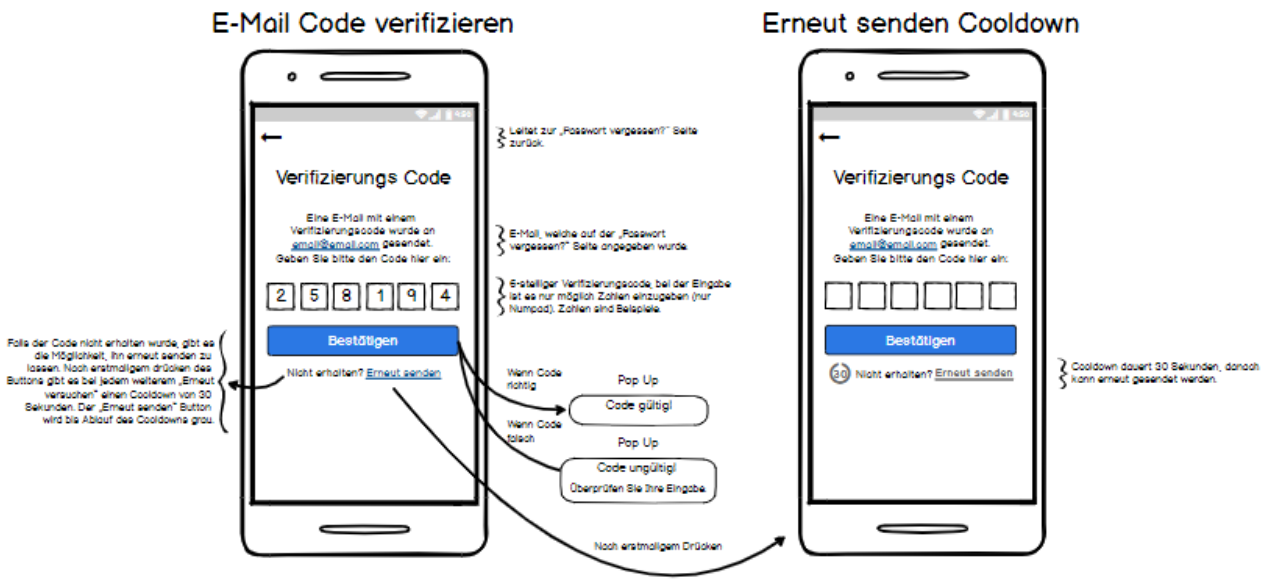
- E-Mail bestätigen
- Anmeldung
- Registrierung mit externem Anbieter

7.3 Userstory "E-Mail bestätigen"

7.3.1 Beschreibung

" Als Kunde muss ich nach der Anmeldung einen Code aus einer gesendeten E-Mail eingeben, sodass ich mit meinem Account einkaufen kann."

7.3.2 Screendesign



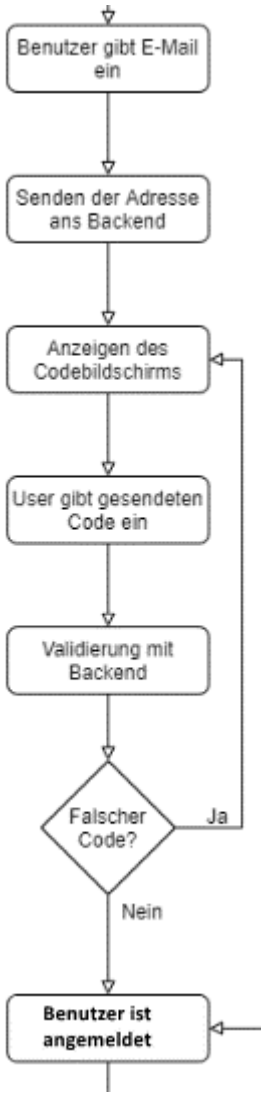
7.3.3 Funktionen der Masken

Tabellarische Darstellung: Button oder Feld mit der entsprechenden Funktionsbeschreibung auf Niveau der User Story!

Zum Beispiel:

Funktion	Beschreibung
Button „Bestätigen“	Schickt den vom Anwender eingegeben Code an die Funktion „VerifyEmail“ vom NQP Webservice.
Button „Erneut Senden“	Schickt eine Anfrage an die Funktion „ResendEmail“ vom NQP Webservice und setzt den Cooldown zurück. Der Button ist wenn der Cooldown aktiv ist deaktiviert.
Cooldown	Zählt nach jeder versendeten E-Mail 30 Sekunden herunter, bis eine neue E-Mail angefordert werden kann.
Zahleneingabefeld „Verifikationscode“	Lässt den Benutzer sechs Ziffern eingeben
Button „Zurück“	Bringt den Benutzer zurück zur Registrierungsseite

7.3.4 Grafische Prozessdarstellung



7.3.5 Schnittstellen

7.3.5.1 "VerifyEmail" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "code": 422739,
    "email": "email@example.com"
}
```

Response:

```
{
    "success": true,
    "responseDate": "2020-09-07T13:10:59"
}
```

7.3.5.2 "ResendEmail" Funktion im Webservice des Back Ends

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "email": "email@example.com"
```

```
}
```

Response:

```
{
    "success": true,
    "responseDate": "2020-09-07T13:10:59"
}
```

7.3.6 Akzeptanzkriterien

- In der Datenbank wird festgehalten, dass die E-Mail Adresse verifiziert wurde
- E-Mails mit Codes werden versandt
- Benutzer ist danach angemeldet

7.3.7 Abgrenzung

Registierung, Passwort vergessen

7.3.8 Voraussetzungen

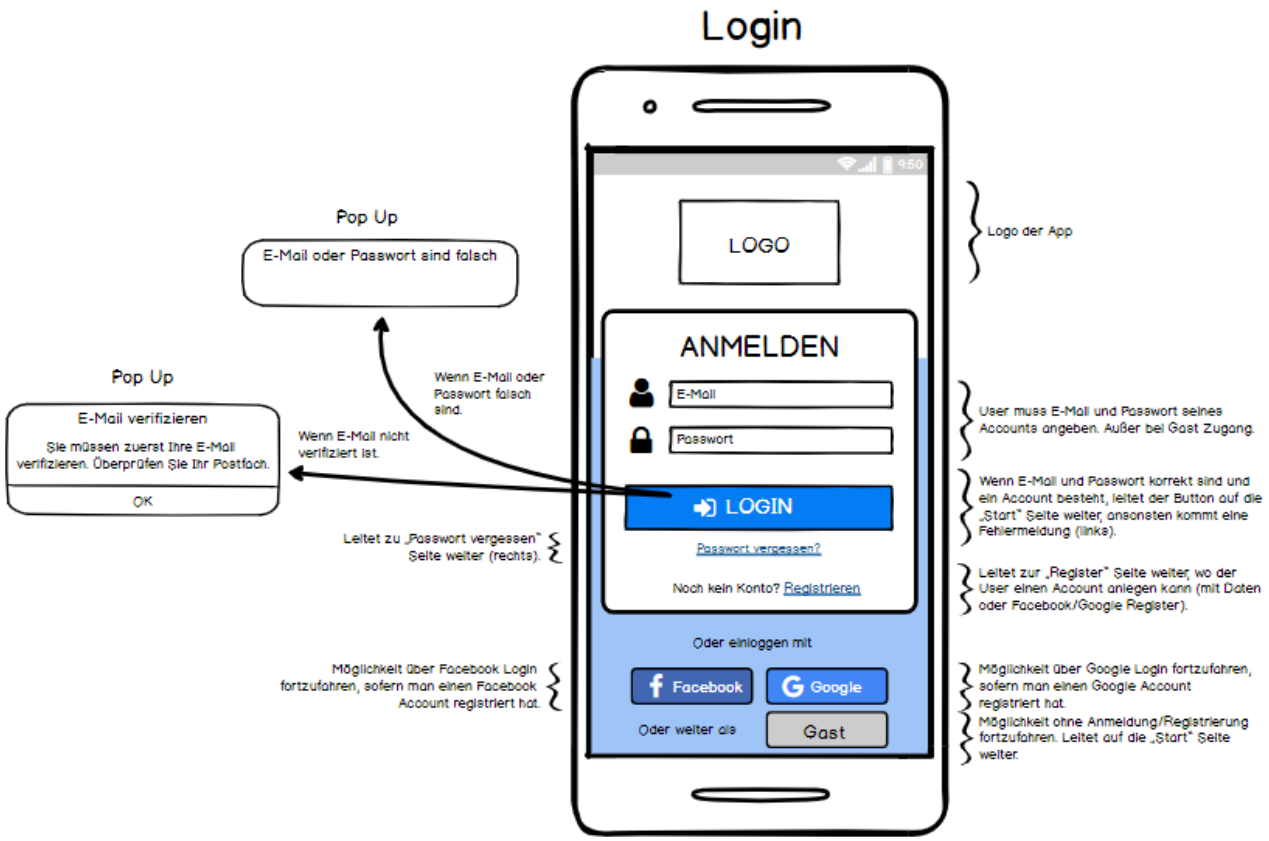
- Login
- Registrierung

7.4 Userstory "Gast-Modus"

7.4.1 Beschreibung

" Als Kunde kann ich Registrierung und Login überspringen, sodass ich als Gast ohne Account einkaufen kann."

7.4.2 Screendesign



7.4.3 Funktionen der Masken

Funktion	Beschreibung
Button „Gast“	Erstellt im Backend einen temporären User (RegisterTemporaryUser), leitet zur Start Page weiter

7.4.4 Grafische Prozessdarstellung

7.4.5 Schnittstellen

7.4.5.1 "RegisterTemporaryUser" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "sessionId": ""
}
```

7.4.6 Lokale Funktionen

Der Session Token und dass der Benutzer ein Gast ist wird lokal gespeichert und bei folgenden Requests im Header übergeben

7.4.7 Akzeptanzkriterien

- Ein Session Token wird generiert und gespeichert und kann danach validiert werden
- Ein temporärer Benutzer wird erstellt
- Temporärer Benutzer wird gelöscht

7.4.8 Abgrenzung

Registierung, Passwort vergessen, Login mit externem Anbieter, Login

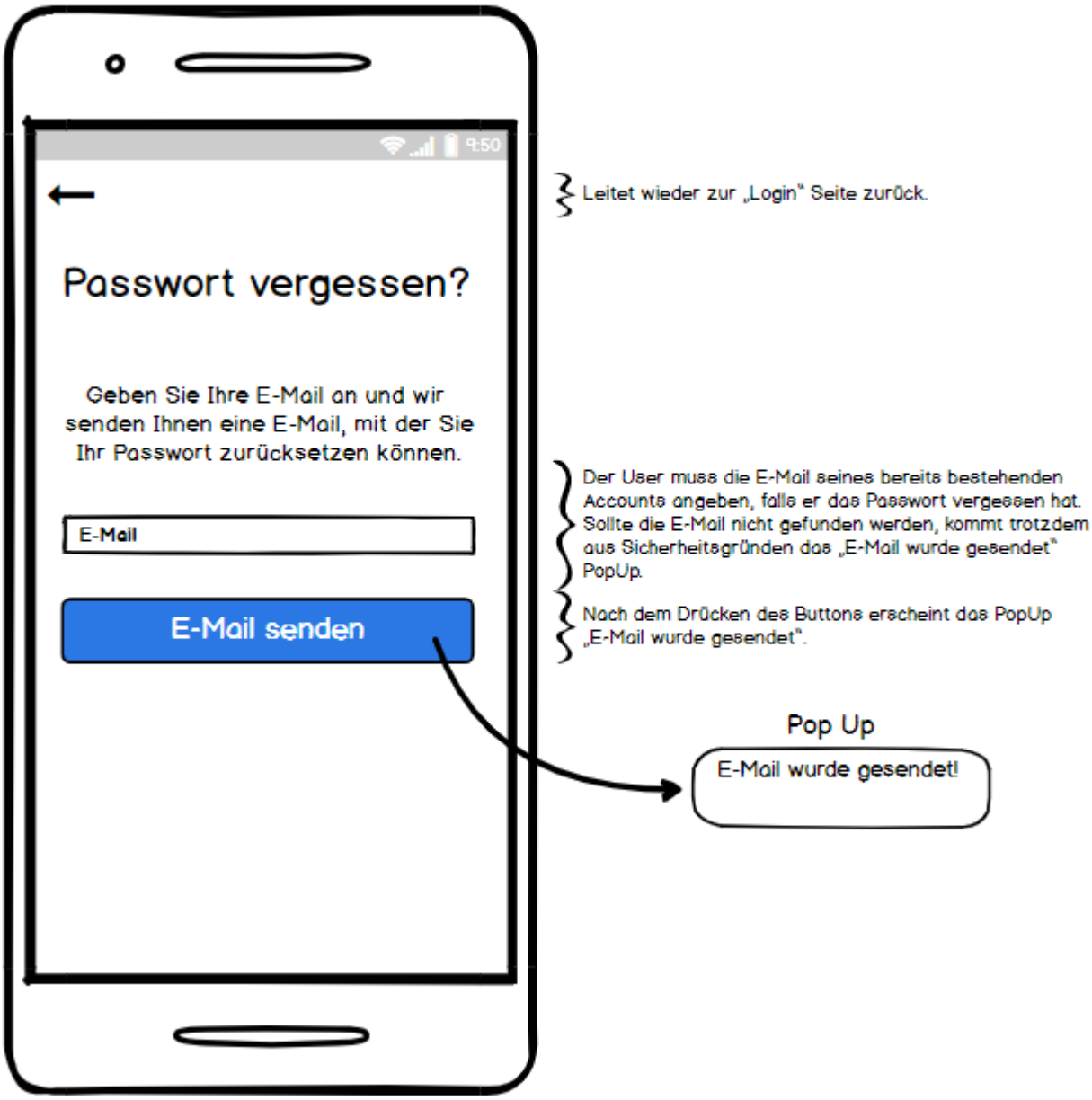
7.5 Userstory "Passwort vergessen"

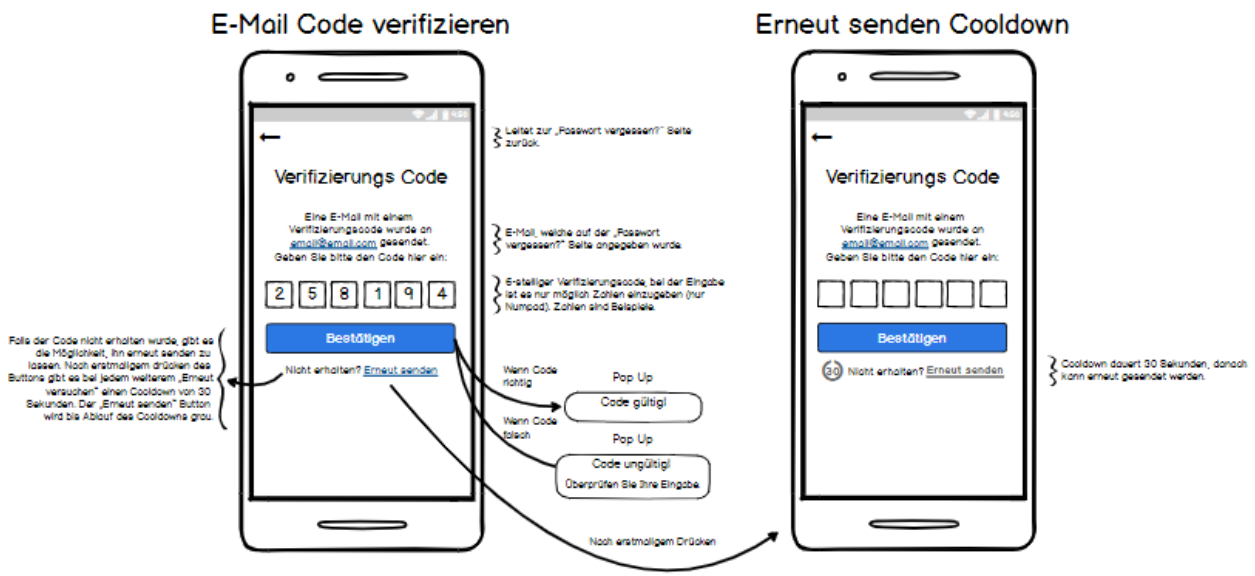
7.5.1 Beschreibung

" Als Kunde kann ich mein Passwort mit meiner E-Mail zurücksetzen, sodass ich ein neues Passwort für meinen bestehenden Account festlegen kann."

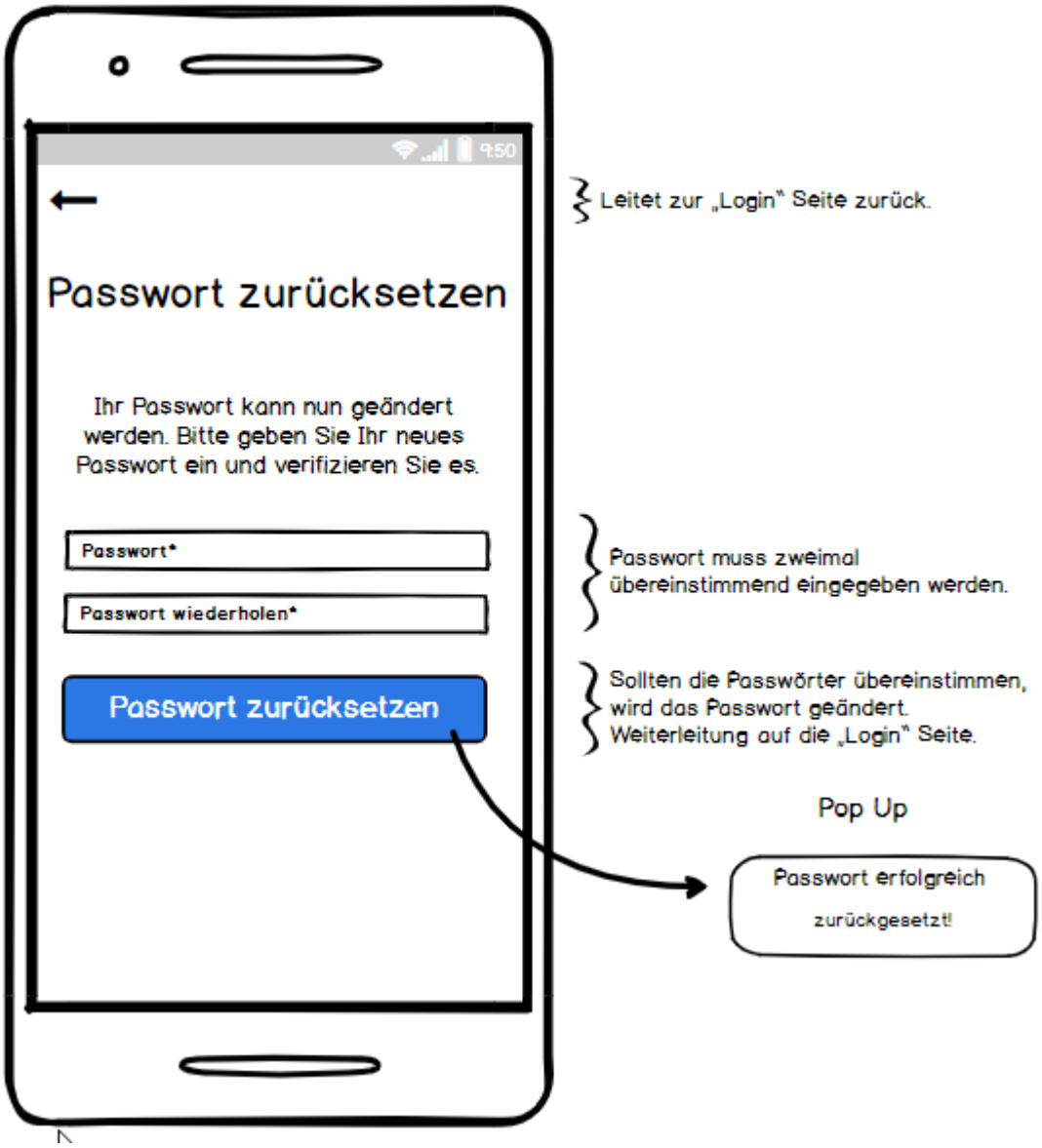
7.5.2 Screendesign

Passwort vergessen





Passwort zurücksetzen/ändern



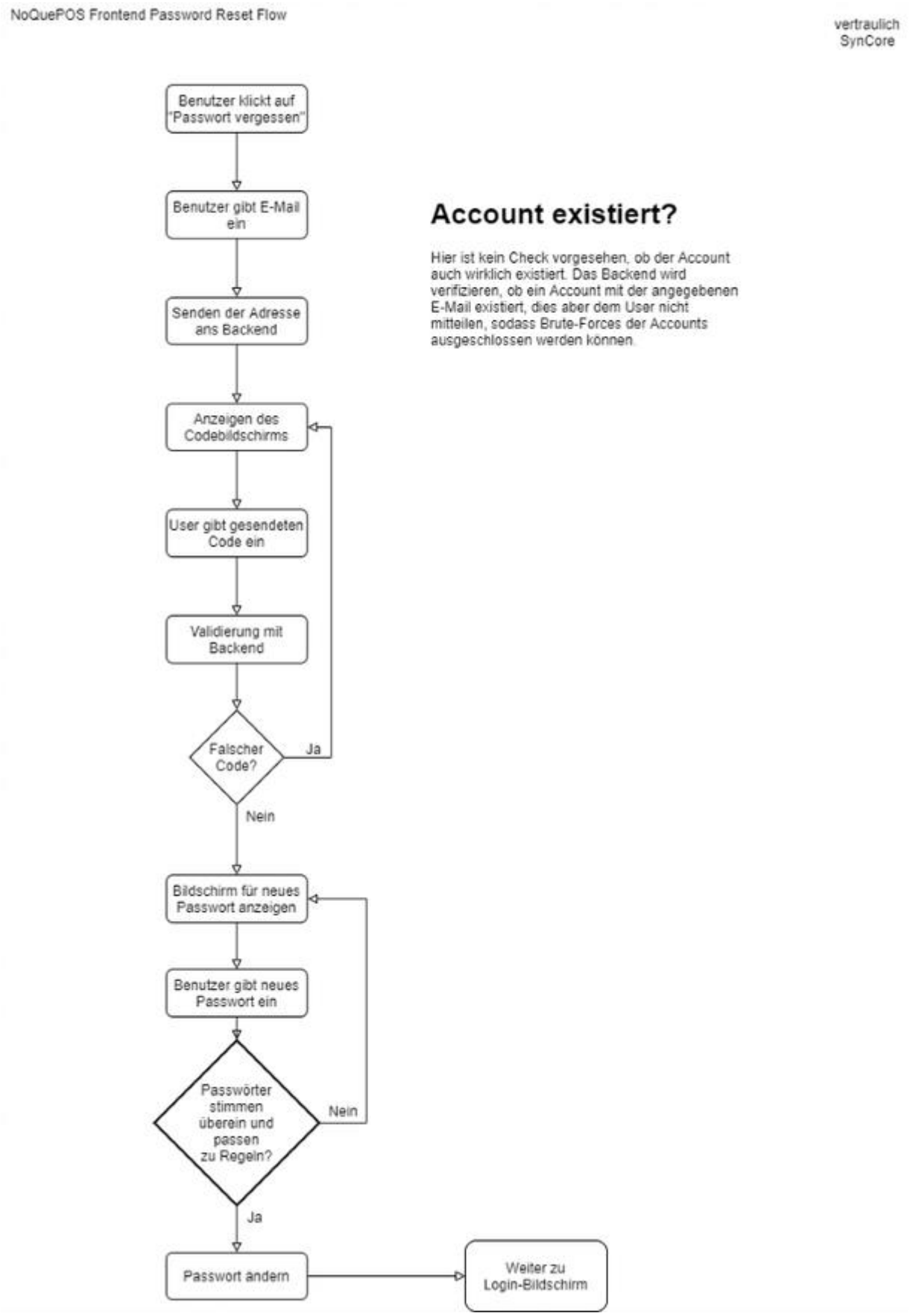
7.5.3 Funktionen der Masken

Tabellarische Darstellung: Button oder Feld mit der entsprechenden Funktionsbeschreibung auf Niveau der User Story!

Zum Beispiel:

Funktion	**Beschreibung**
Eingabefeld „E-Mail“	Lässt den Benutzer eine E-Mail eingeben
Button „E-Mail senden“	Schickt die vom Anwender eingegebe E-Mail-Adresse an die Funktion „ForgotPasswordEmail“ vom NQP Webservice und leitet den Benutzer zum Codebildschirm weiter.
Button „Bestätigen“	Schickt den vom Anwender eingegeben Code an die Funktion „VerifyEmail“ vom NQP Webservice.
Button „Erneut Senden“	Schickt eine Anfrage an die Funktion „ResendEmail“ vom NQP Webservice und setzt den Cooldown zurück. Der Button ist wenn der Cooldown aktiv ist deaktiviert.
Cooldown	Zählt nach jeder versendeten E-Mail 30 Sekunden herunter, bis eine neue E-Mail angefordert werden kann.
Zahleneingabefeld „Verifikationscode“	Lässt den Benutzer sechs Ziffern eingeben
Button „Zurück“ Vergessen	Bringt den Benutzer zurück zur Loginseite
Button „Zurück“ Zurücksetzen	Bringt den Benutzer zurück zur Seite „Passwort vergessen“
Eingabefeld „Passwort“	Eingegebener Text wird versteckt, mindestens 8 Stellen, mind. ein Großbuchstabe, mind. eine Zahl
Eingabefeld „Passwort wiederholen“	Inhalt stimmt mit der Textbox „Passwort“ überein
Button „Passwort zurücksetzen“	Schickt den vom Anwender eingegeben Code an die Funktion „ResetPassword“ vom NQP Webservice und leitet den Benutzer zur Anmeldeseite weiter.

7.5.4 Grafische Prozessdarstellung



7.5.5 Schnittstellen

7.5.5.1 " ForgotPasswordEmail" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "countdownTime": 30
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "success": true,
    "session": ""
}
```

7.5.5.2 " ValidatePasswordVerificationCode" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "code": 125521,
    "email": "email@example.com",
    "session": ""
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "email": "email@example.com",
    "valid": true
}
```

7.5.5.3 " ResetPassword" Funktion im Webservice des Back End

POST Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "session": "",
    "email": "email@example.com",
    "password": "P@ssw0rd",
    "code": 125521
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true
}
```

7.5.6 Akzeptanzkriterien

- Der Benutzer kann sich mit dem neuen Passwort anmelden
- Der Benutzer wird auf den Login-Screen weitergeleitet

7.5.7 Abgrenzung

Registierung, Login

7.5.8 Voraussetzungen

- Registrierung, Benutzerprofil
- E-Mail bestätigen (Codemaske)
- Login

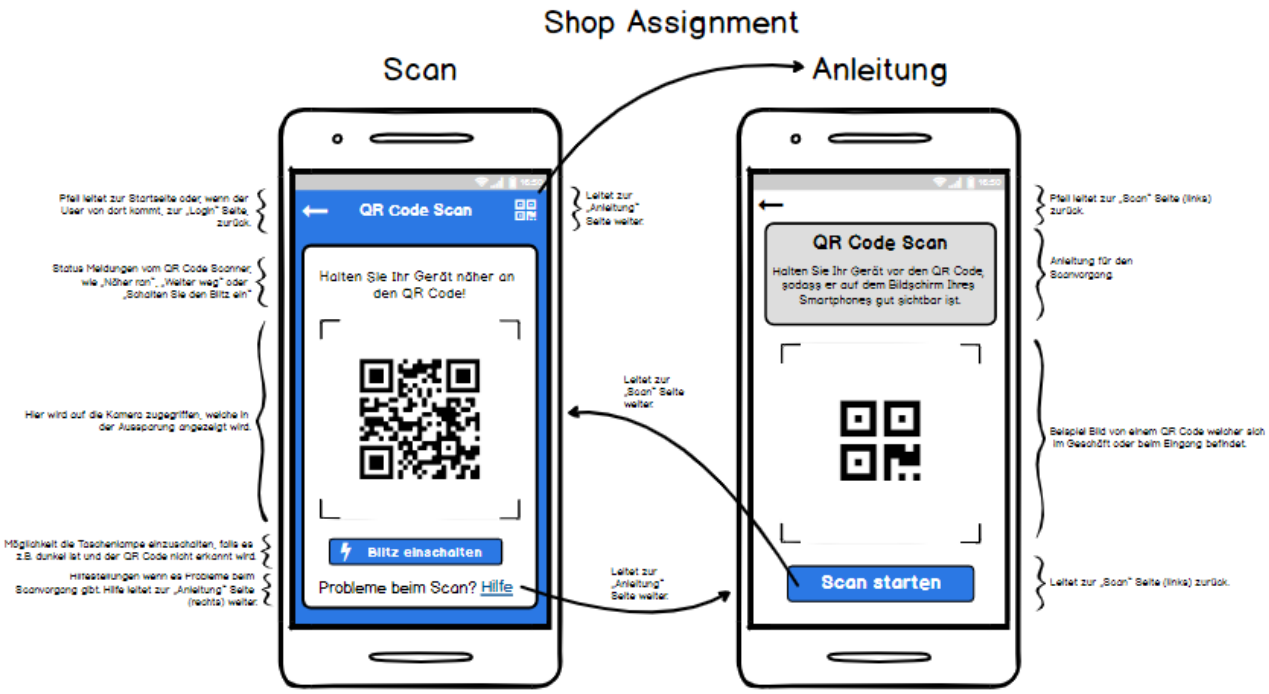
8 Epic „Shop Assignment“

8.1 Userstory "QR-Code scannen"

8.1.1 Beschreibung

" Als Kunde kann ich den QR Code mit der Kamera scannen, sodass ich im richtigen Store eingeloggt bin und mit dem Einkauf starten kann."

8.1.2 Screendesign



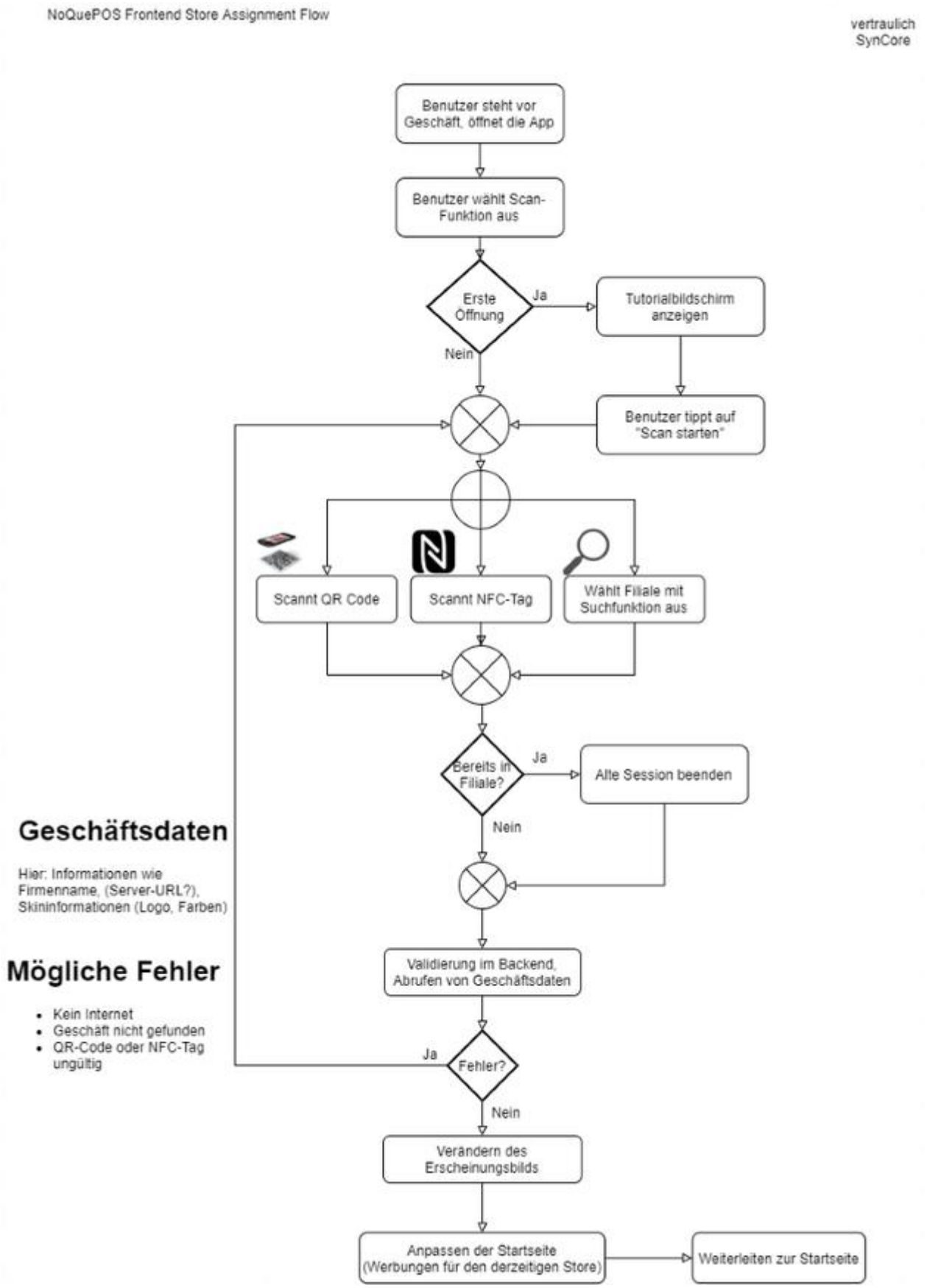
8.1.3 Funktionen der Masken

Tabellarische Darstellung: Button oder Feld mit der entsprechenden Funktionsbeschreibung auf Niveau der User Story!

Zum Beispiel:

Funktion	Beschreibung
Button „Zurück“	Leitet den Benutzer zur Startpage zurück
Button „Hilfe“	Leitet den Benutzer zur Anleitungsseite weiter
Button „Blitz einschalten“	Aktiviert die Taschenlampe des Geräts (falls möglich/vorhanden)
Button „Scan starten“	Leitet den Benutzer zur Scanseite zurück
Kamera	Erfasst QR-Codes und schickt eine Anfrage an die Funktion „AssignShop“ im Backend

8.1.4 Grafische Prozessdarstellung



8.1.5 Schnittstellen

8.1.5.1 "AssignShop" Funktion im Webservice des Back End

Request:
Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

Body:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "shopId": "4555-4482" // Merchant ID - POS ID
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "pos": {
        "id": 2345,
        "name": "Merkur",
        "address": "Simmeringer Hauptstraße 434/2 1030 Wien",
        "primaryColor": "32a852",
        "secondaryColor": "32a852",
        "primaryTextColor": "ffffff",
        "secondaryTextColor": "000000",
        "logo": "https://example.com/img.png",
    },
    "shoppingSessionToken": ""
}
```

8.1.6 Lokale Funktionen

Der shoppingSessionToken wird gespeichert und bei darauffolgenden Requests im Store beigefügt, die Shopadresse und der Shopname werden angezeigt und das Design der App wird angepasst

8.1.7 Akzeptanzkriterien

- Der Benutzer hat eine gültige Shopping Session – d.h. ist angemeldet
- Das Design der App ändert sich
- Marktdaten sind geladen

8.1.8 Abgrenzung

Werbung, Produktlisten, Shop verlassen

8.1.9 Voraussetzungen

- Login möglich
- Shop vorhanden

8.2 Userstory "Shops in meiner Nähe"

8.2.1 Beschreibung

"Als Kunde kann ich einen Store aus der Liste auswählen, sodass ich in diesem Store eingeloggt bin und mit dem Einkauf starten kann."

8.2.2 Funktionen der Masken

Funktion	Beschreibung
Button „Zurück“	Leitet den Benutzer zur Startpage zurück

Button „Hilfe“	Leitet den Benutzer zur Anleitungsseite weiter
Button „Blitz einschalten“	Aktiviert die Taschenlampe des Geräts (falls möglich/vorhanden)
Button „Scan starten“	Leitet den Benutzer zur Scanseite zurück
Kamera	Erfasst QR-Codes und schickt eine Anfrage an die Funktion „AssignShop“ im Backend

8.2.3 Schnittstellen

8.2.3.1 "ListCloseShops" Funktion im Webservice des Back End

Request:
Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

Body:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "latitude": 48.224184,
    "longitude": 16.389374,
    "start": 0, // Index des ersten Items
    "count": 20
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "shops": [
        {
            "id": "4555-4482", // ShopId
            "distance": 100, // Meter
            "image": "https://scanybugo.example.com/images/merkur.jpg",
            "name": "Merkur",
            "address": "Simmeringer Hauptstraße 434/2 1030 Wien",
            "openFrom": 700,
            "openTo": 1800,
        }
    ],
    "shopsCount": 100 // gesamte Anzahl an Shops
}
```

8.2.3.2 "AssignShop" Funktion im Webservice des Back End

Request:
Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

Body:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "shopId": "4555-4482" // Merchant ID - POS ID
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "pos": {
        "id": 4482,
        "name": "Merkur",
        "address": "Simmeringer Hauptstraße 434/2 1030 Wien",
        "primaryColor": "32a852",
        "secondaryColor": "32a852",
        "primaryTextColor": "ffffff",
        "secondaryTextColor": "000000",
    },
    "shoppingSessionToken": "33fdd1b6-b496-4b33-9f7d-df96679d32fe"
}
```


8.2.4 Lokale Funktionen

- Beim Öffnen der Liste wird die aktuelle Position des Benutzers ermittelt und ans Backend übermittelt.
- Es wird ausgerechnet, wie lange ein Shop noch geöffnet hat
- Die Distanz zum Shop wird in ein lesbares Format konvertiert (x Meter, Kilometer)

8.2.5 Akzeptanzkriterien

- Darstellung einer Liste, sortiert nach Entfernung

8.2.6 Abgrenzung

QR-Code scannen, Shopliste/Suche

8.2.7 Voraussetzungen

- Login möglich
- Shop vorhanden
- AssignShop existiert

8.3 Userstory "Store verlassen manuell"

8.3.1 Beschreibung

Als Kunde kann ich auf den Button „Store verlassen“ klicken, sodass ich nicht mehr in diesem Store eingeloggt bin und wieder die Möglichkeit habe, mich bei einem anderen Store anmelden zu können.

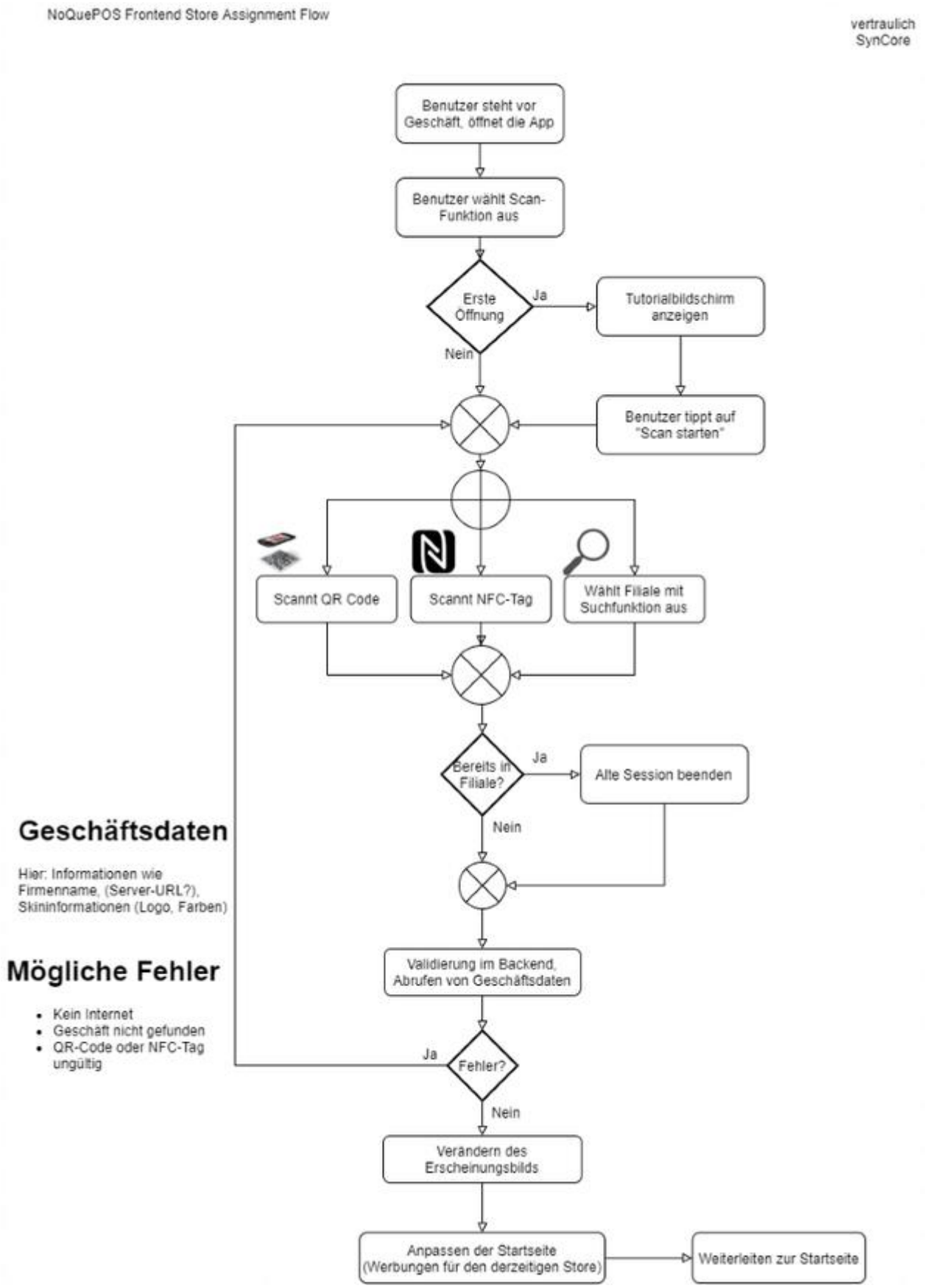
Als Kunde kann ich während des Einkaufens jederzeit den Einkauf abbrechen, sodass der gesamte Warenkorb gelöscht wird und ich aus dem Shop ausgeloggt werde.

8.3.2 Screendesign

8.3.3 Funktionen der Masken

Funktion	Beschreibung
Button „Shop verlassen“	Öffnet ein Pop-Up, in dem der Benutzer das Verlassen bestätigen soll, schickt eine Anfrage an die Funktion „LeaveShop“ im Backend

8.3.4 Grafische Prozessdarstellung



8.3.5 Schnittstellen

8.3.5.1 "LeaveShop" Funktion im Webservice des Back End

POST Request:

Header:

SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShoppingSessionToken: 33fdd1b6-b496-4b33-9f7d-df96679d32fe

Body:

```
{
    "requestDate": "2020-09-07T13:10:59"
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true
}
```

8.3.6 Lokale Funktionen

- Warenkorb wird gelöscht
- Design wird zurückgesetzt
- ShoppingSessionToken löschen

8.3.7 Akzeptanzkriterien

- Man wird aus dem Store ausgeloggt
- Das Standarddesign wird angewandt
- Der Warenkorb wird gelöscht

8.3.8 Abgrenzung

Shop betreten, Abmeldung (des Benutzers)

8.3.9 Voraussetzungen

- Login möglich
- Shop Assignment möglich
- Navigation vorhanden

8.4 Userstory "Automatische Abmeldung vom alten Store bei erneutem Scan"

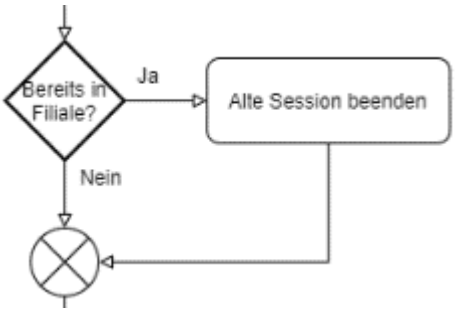
8.4.1 Beschreibung

" Als Kunde kann ich einen QR Code mittels Kamera scannen, sodass ich nicht mehr im letzten Store eingeloggt bin und im neuen Store einkaufen kann."

8.4.2 Funktionen der Masken

Funktion	Beschreibung
Button „Zurück“	Leitet den Benutzer zur Startpage zurück
Button „Hilfe“	Leitet den Benutzer zur Anleitungsseite weiter
Button „Blitz einschalten“	Aktiviert die Taschenlampe des Geräts (falls möglich/vorhanden)
Button „Scan starten“	Leitet den Benutzer zur Scanseite zurück
Kamera	Erfasst QR-Codes und schickt eine Anfrage an die Funktion „AssignShop“ im Backend

8.4.3 Grafische Prozessdarstellung



8.4.4 Schnittstellen

8.4.4.1 "ListCloseShops" Funktion im Webservice des Back End

Request:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "latitude": 48.224184,
    "longitude": 16.389374,
    "start": 0, // Index des ersten Items
    "count": 20
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "shops": [
        {
            "id": "4555-4482", // ShopId
            "distance": 100, // Meter
            "image": "https://example.com/images/merkur.jpg",
            "name": "Merkur",
            "address": "Simmeringer Hauptstraße 434/2 1030 Wien",
            "openFrom": 700,
            "openTo": 1800,
        }
    ],
    "shopsCount": 100 // gesamte Anzahl an Shops
}
```

8.4.5 Lokale Funktionen

- Beim Öffnen der Liste wird die aktuelle Position des Benutzers ermittelt und ans Backend übermittelt.
- Es wird ausgerechnet, wie lange ein Shop noch geöffnet hat
- Die Distanz zum Shop wird in ein lesbares Format konvertiert (x Meter, Kilometer)

8.4.6 Akzeptanzkriterien

- Darstellung einer Liste, sortiert nach Entfernung

8.4.7 Abgrenzung

Shop betreten, QR-Code scannen, Shopliste/Suche

8.4.8 Voraussetzungen

- Login möglich
- Shop vorhanden

- AssignShop existiert

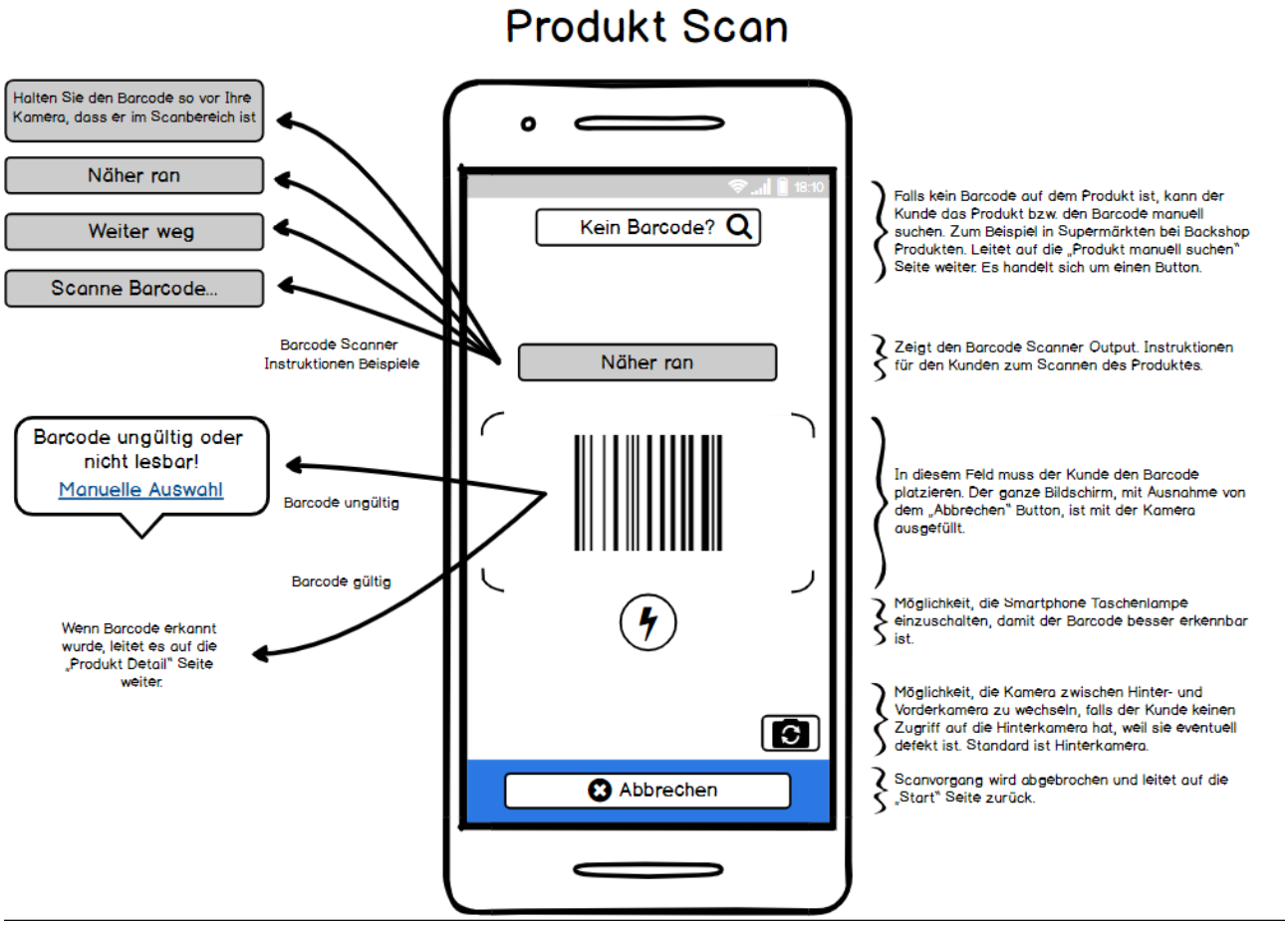
9 Epic „Shopping Cart“

9.1 Userstory "Produkt per Scan hinzufügen"

9.1.1 Beschreibung

"Als Kunde kann ich den QR-Code oder Barcode eines Produktes einscannen, sodass ich das Produkt, bevor es meinem virtuellen Warenkorb hinzugefügt wird, nochmals reviewen kann."

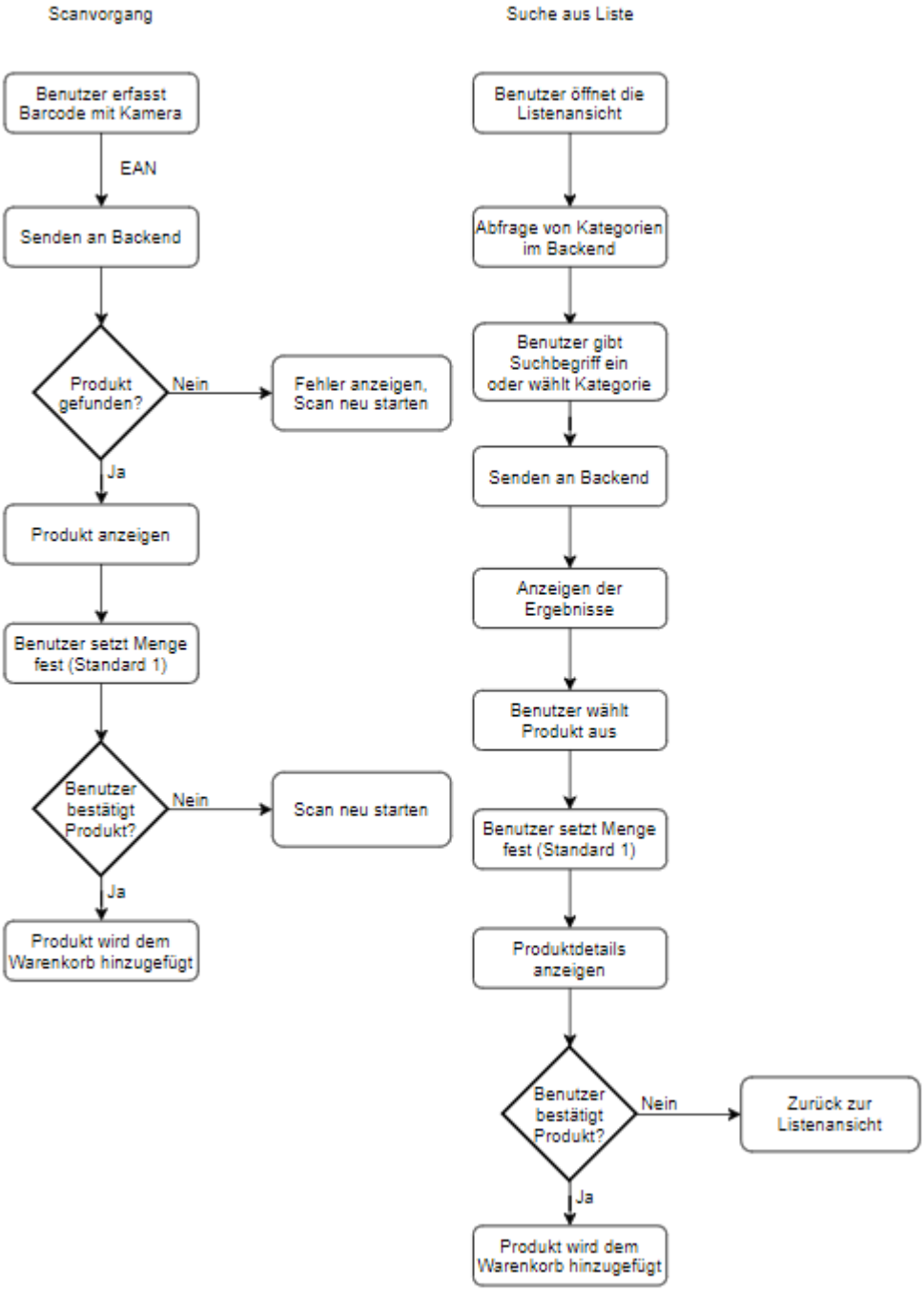
9.1.2 Screendesign



9.1.3 Funktionen der Masken

Funktion	Beschreibung
Button „Kein Barcode“	Leitet den Benutzer zur Produktliste/-suche weiter
Scanbereich	Zeigt die Kamera an
Button „Blitz einschalten“	Aktiviert die Taschenlampe des Geräts (falls möglich/vorhanden)
Button „Abbrechen“	Leitet den Benutzer zurück zur Startpage

9.1.4 Grafische Prozessdarstellung


Scanvorgang
Benutzer erfasst Barcode mit Kamera
EAN
Senden an Backend
Produkt gefunden?
Nein
Fehler anzeigen, Scan neu starten
Ja
Produkt anzeigen
Benutzer setzt Menge fest (Standard 1)
Benutzer bestätigt Produkt?
Nein
Scan neu starten
Ja
Produkt wird dem Warenkorb hinzugefügt
Suche aus Liste
Benutzer öffnet die Listenansicht
Abfrage von Kategorien im Backend
Benutzer gibt Suchbegriff ein oder wählt Kategorie
Senden an Backend
Anzeigen der Ergebnisse
Benutzer wählt Produkt aus
Benutzer setzt Menge fest (Standard 1)
Produktdetails anzeigen
Benutzer bestätigt Produkt?
Nein
Zurück zur Listenansicht
Ja
Produkt wird dem Warenkorb hinzugefügt

9.1.5 Schnittstellen

9.1.5.1 "GetProductDetails" Funktion im Webservice des Back End

Request:
Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShoppingSessionToken: 33fdd1b6-b496-4b33-9f7d-df96679d32fe

Body:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "articleCode": "57392048220" // EAN oder Artikelnummer
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "product": {
        "articleCode": "57392048220",
        "image": "https://example.com/images/cola.jpg",
        "name": "Coca Cola",
        "brand": "The Coca Cola Company",
        "info": "0,5 Liter Flasche",
        "price": 300,
        "quantityEditable": true,
    }
}
```

9.1.6 Akzeptanzkriterien

- Der Detailbildschirm (falls vorhanden) wird mit Produktdaten angezeigt, falls das Produkt vorhanden ist
- Fehlermeldung, wenn kein Produkt gefunden wurde

9.1.7 Abgrenzung

Zum Warenkorb hinzufügen

9.1.8 Voraussetzungen

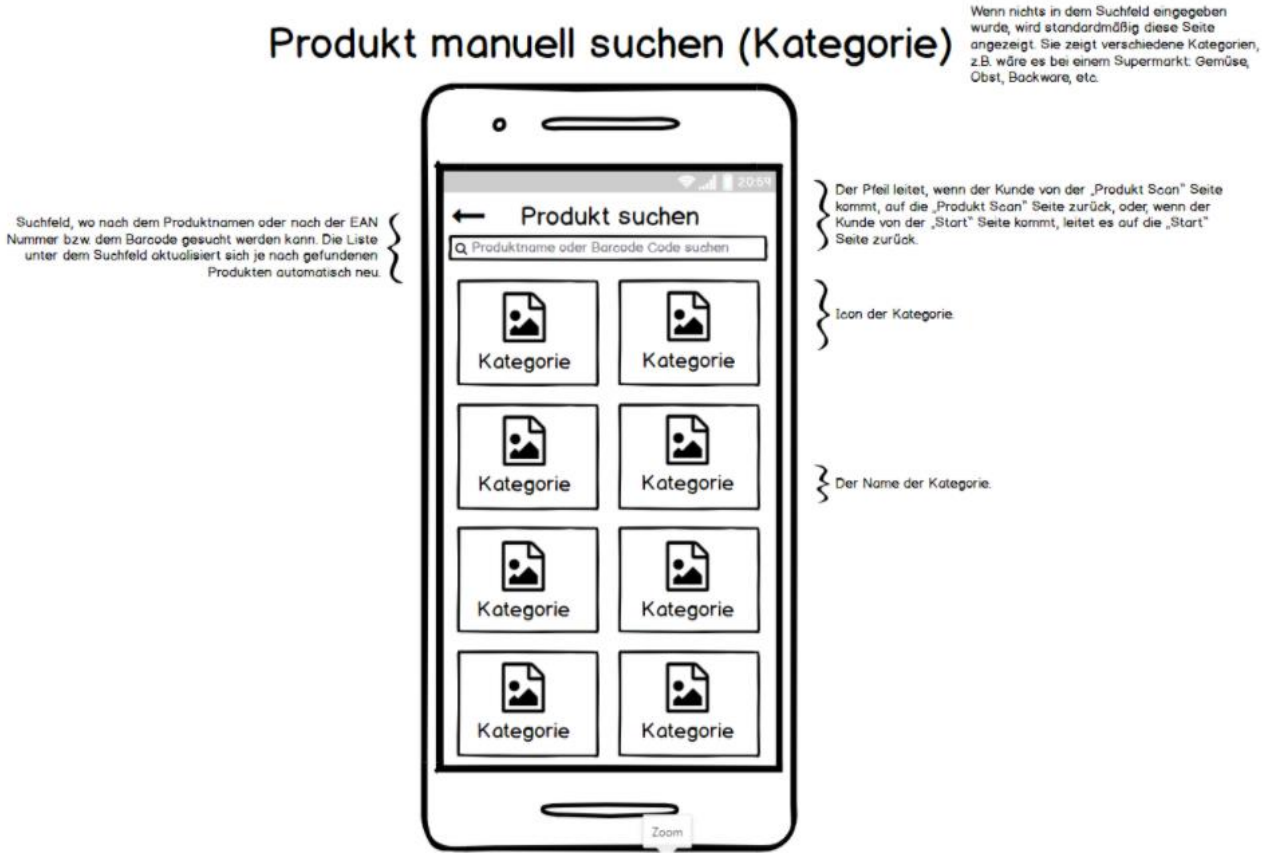
- Shopanmeldung vorhanden

9.2 Userstory "Produkt per Kategorie hinzufügen"

9.2.1 Beschreibung

„Als Kunde kann ich durch eine Liste ein Produkt suchen und hinzufügen, sodass ich das Produkt, bevor es meinem virtuellen Warenkorb hinzugefügt wird, nochmals reviewen kann.“

9.2.2 Screendesign



9.2.3 Funktionen der Masken

Funktion	Beschreibung
Button „Kategorie“	Öffnet eine Liste von Produkten in der angezeigten Kategorie
Button „Zurück“	Leitet den Benutzer zurück zur Scanseite

9.2.4 Grafische Prozessdarstellung

Workflow, möglichst detailliert

9.2.5 Schnittstellen

9.2.5.1 "GetCategories" Funktion im Webservice des Back End

Request:
Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShoppingSessionToken: 33fdd1b6-b496-4b33-9f7d-df96679d32fe

Body:

```
{
    "requestDate": "2020-09-07T13:10:59"
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "categories": [
        {
            "image": "https://scanbuygo.example.com/images/schuhe.jpg",
```

```
            "name": "Schuhe"
        }
    ]
}
```

9.2.5.2 "GetCategories" Funktion im Webservice des Back End

9.2.6 Akzeptanzkriterien

-

9.2.7 Abgrenzung

Zum Warenkorb hinzufügen

9.2.8 Voraussetzungen

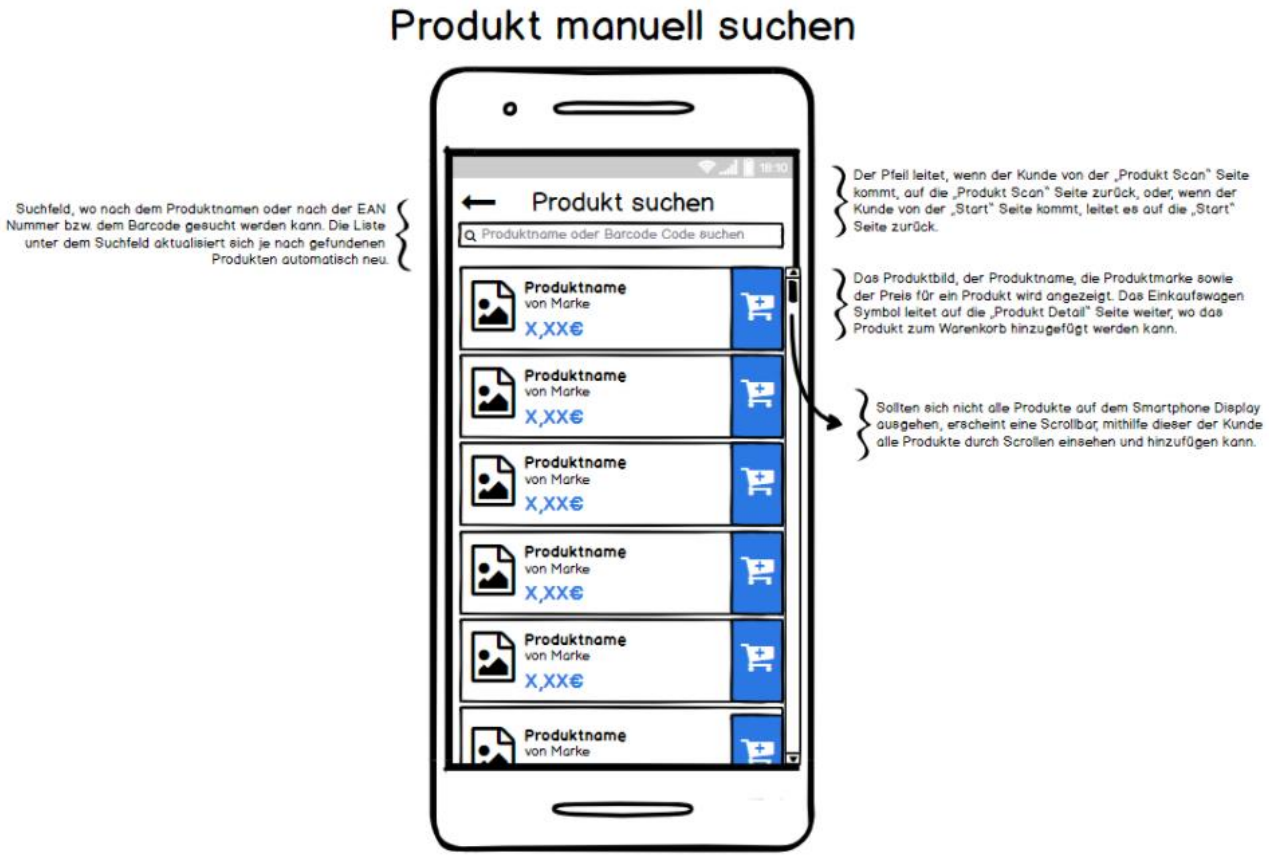
- Shopanmeldung vorhanden

9.3 Userstory "Produkt per Liste hinzufügen"

9.3.1 Beschreibung

"Als Kunde kann ich optional durch eine Liste ein Produkt suchen und hinzufügen, sodass ich das Produkt, bevor es meinem virtuellen Warenkorb hinzugefügt wird, nochmals reviewen kann."

9.3.2 Screendesign

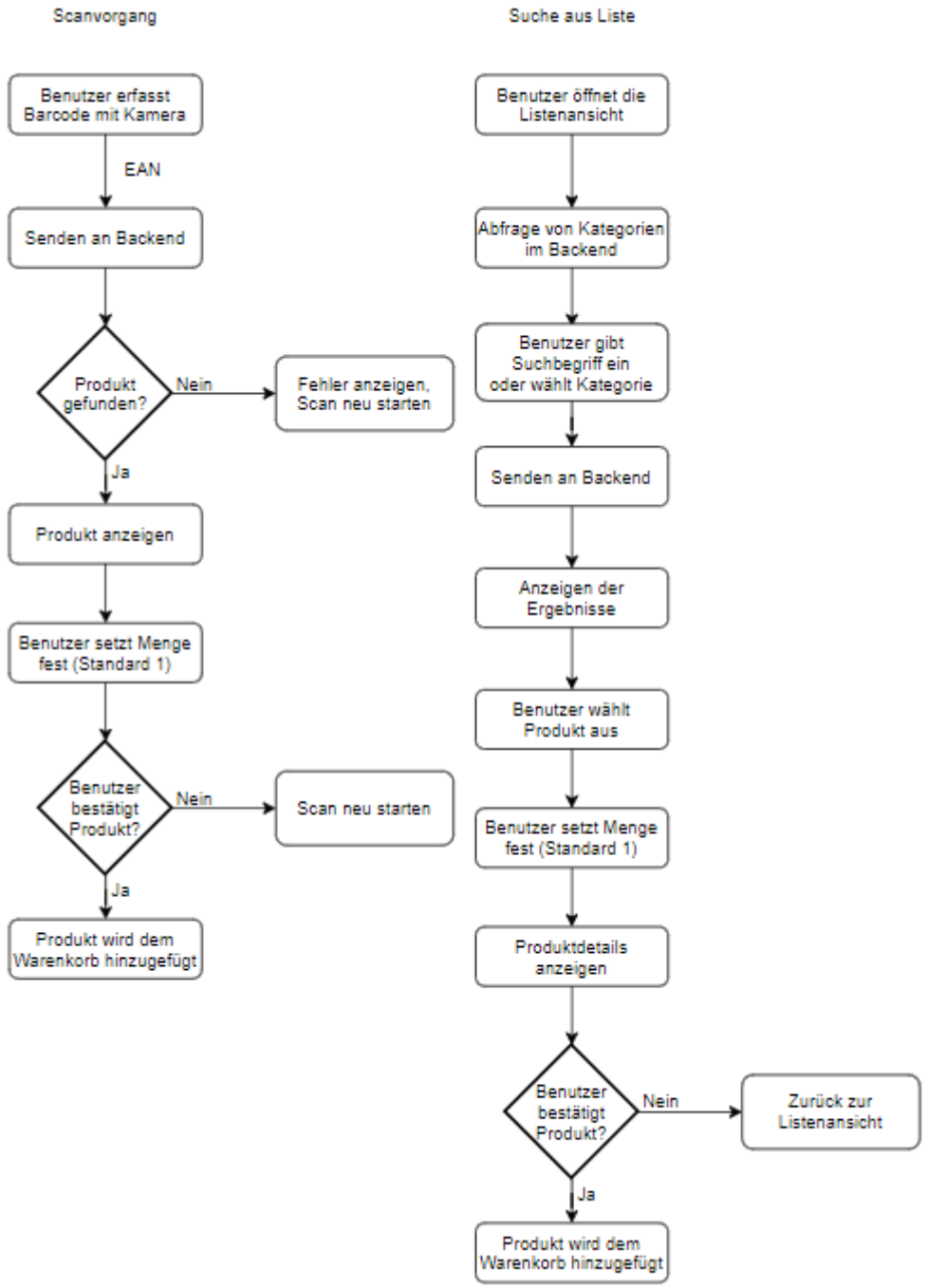


9.3.3 Funktionen der Masken

Funktion	Beschreibung

Button „Zurück“	Leitet den Benutzer zur Produktliste/-suche weiter
Textfeld „Suche“	Lässt den Benutzer einen Suchbegriff (Barcode, Produktnummer, Name, Kategorie, Marke) eingeben und sendet diesen ans Backend. Alle Begriffe können teilweise eingegeben werden.
Button „Zum Warenkorb hinzufügen“	Fügt das Produkt dem Warenkorb hinzu
Produktfeld	Öffnet die Detailansicht des Produkts

9.3.4 Grafische Prozessdarstellung



9.3.5 Schnittstellen

9.3.5.1 "FindProducts" Funktion im Webservice des Back End

Request:
Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShoppingSessionToken: 33fdd1b6-b496-4b33-9f7d-df96679d32fe

Body:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "query": "Nike" // Suchbegriff
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "results": [
        {
            "articleCode": "57392048220",
            "image": "https://scanbuygo.example.com/images/cola.jpg",
            "name": "Coca Cola",
            "brand": "The Coca Cola Company",
            "info": "0,5 Liter Flasche",
            "price": 300,
            "quantityEditable": true
        }
    ]
}
```

9.3.6 Akzeptanzkriterien

- Die Suchergebnisse werden dargestellt

9.3.7 Abgrenzung

Zum Warenkorb hinzufügen

9.3.8 Voraussetzungen

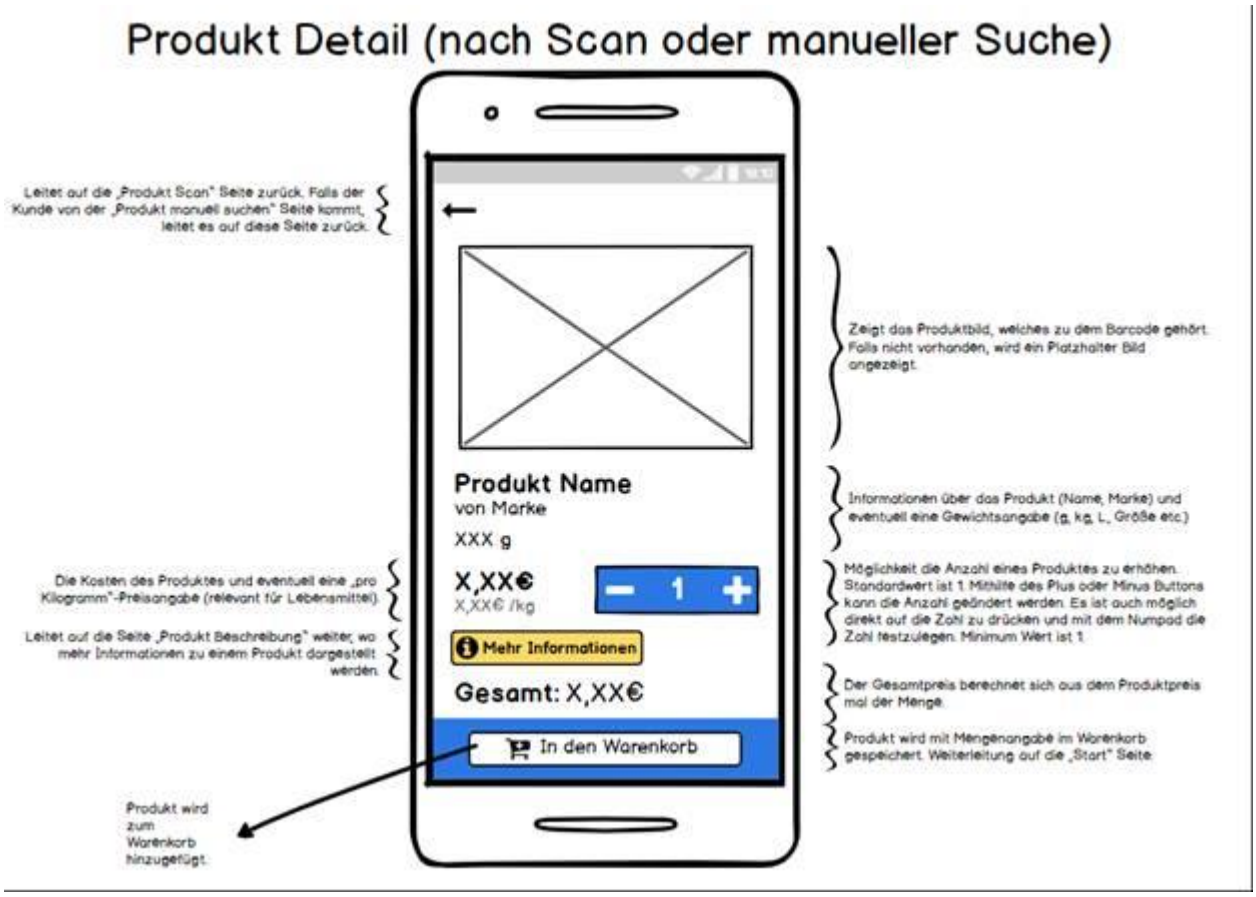
- Shopanmeldung vorhanden

9.4 Userstory "Produkt hinzufügen/Stückzahl"

9.4.1 Beschreibung

"Als Kunde kann ich, nachdem ich das Produkt gescannt, in einer Liste gefunden habe oder durch die Kategorien entdeckt habe, auswählen, welche Stückzahl ich von diesem Produkt kaufen möchte, sodass ich das Produkt nicht mehrmals suchen brauche und die jeweilige Produktanzahl meinem Warenkorb hinzugefügt wird."

9.4.2 Screendesign



9.4.3 Funktionen der Masken

Funktion	Beschreibung
Button „Plus“	Erhöht die Anzahl des Produkts und errechnet die Gesamtsumme
Button „Minus“	Verringert die Anzahl des Produkts und errechnet die Gesamtsumme
Textfeld „Anzahl“	Setzt die Anzahl des Produkts auf die eingegebene Zahl und errechnet die Gesamtsumme

9.4.4 Lokale Funktionen

Menge im Warenkorb festlegen

9.4.5 Akzeptanzkriterien

- Die Anzahl eines Produkts kann nicht kleiner als 1 sein
- Die Menge des Produkts wird bei Klick auf „Plus“ erhöht
- Die Menge des Produkts wird bei Klick auf „Minus“ verringert
- Die Menge des Produkts wird bei Eingabe in „Anzahl“ gesetzt

9.4.6 Abgrenzung

Warenkorb

9.4.7 Voraussetzungen

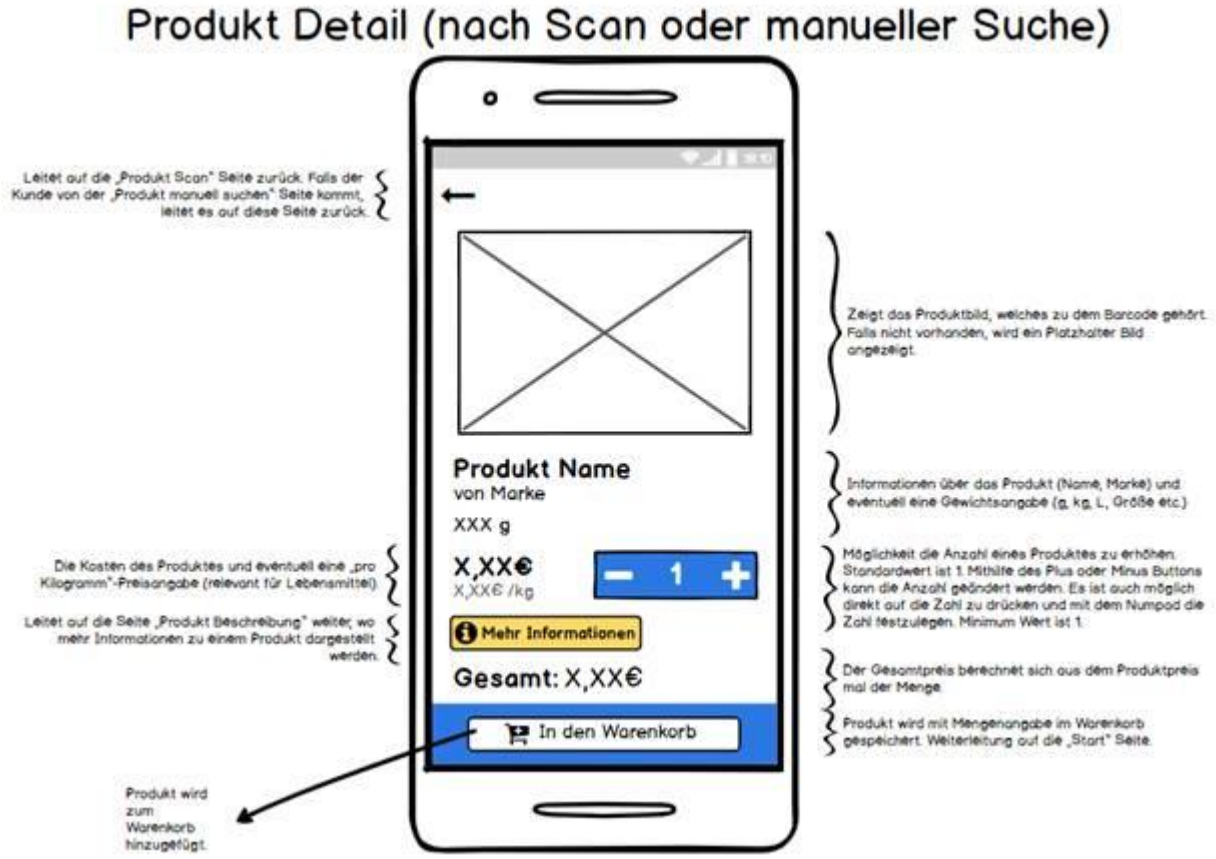
Warenkorb, Produkt hinzufügen

9.5 Userstory "Produktdetails nach Scan"

9.5.1 Beschreibung

"Als Kunde kann ich, nachdem ich ein Produkt eingescannt, manuell in der Produktliste oder in den Kategorien gefunden habe und bevor ich es in den Warenkorb hinzufüge, die Produktdetails inkl. Bild ansehen, sodass ich mir vorher noch die Produktdetails durchlesen kann."

9.5.2 Screendesign



9.5.3 Funktionen der Masken

Funktion	Beschreibung
Button „Zurück“	Leitet den Benutzer zur Scanseite zurück
Label „Einzelpreis“	Zeigt den Einzelpreis des Produkts an
Button „Mehr Informationen“	Öffnet die Seite „Mehr Informationen“
Label „Gesamtpreis“	Zeigt den Gesamtpreis (Einzelpreis * Menge) brutto an

9.5.4 Grafische Prozessdarstellung

Workflow, möglichst detailliert

9.5.5 Lokale Funktionen

Die Gesamtsumme wird bei Veränderung der Menge angepasst

9.5.6 Akzeptanzkriterien

- Gesamtpreis wird korrekt errechnet
- Produktinformationen werden falls vorhanden angezeigt

9.5.7 Abgrenzung

Menge erhöhen/verringern

9.5.8 Voraussetzungen

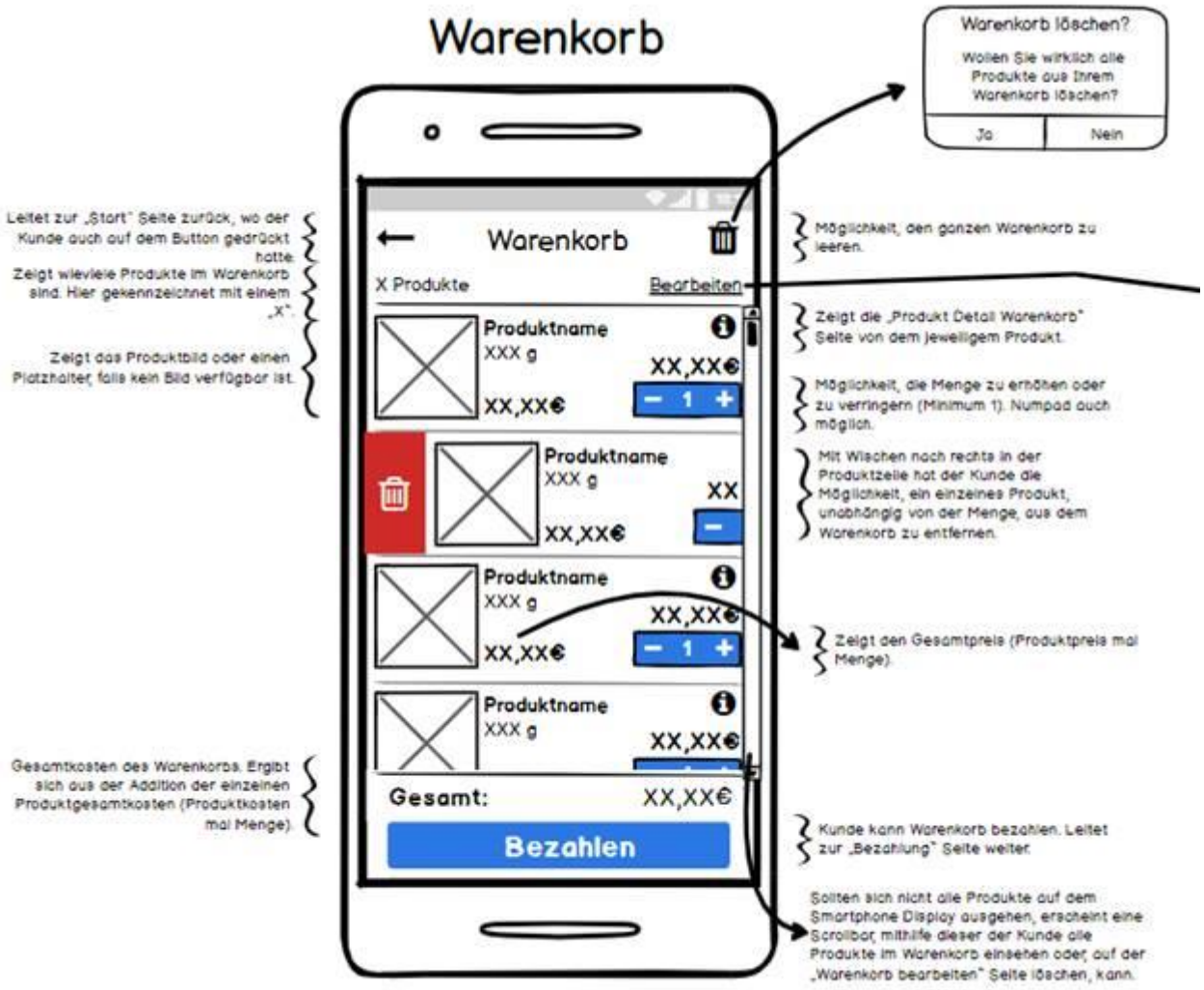
Produkt hinzufügen, Warenkorblogik

9.6 Userstory "Warenkorb ansehen"

9.6.1 Beschreibung

"Als Kunde kann ich während des Einkaufens jederzeit meinen virtuellen Warenkorb ansehen, sodass mir alle bereits gescannten Produkte inkl. Bild und Stückzahl angezeigt werden und was die aktuelle Zwischensumme in Euro beträgt."

9.6.2 Screendesign



9.6.3 Funktionen der Masken

Funktion	Beschreibung
Liste	Liste der Produkte im Warenkorb mit Bild, Name, Einzelpreis und Gesamtpreis
Button „Zurück“	Leitet den Benutzer zur Startseite zurück
Label „Gesamt“	Zeigt den Brutto-Gesamtpreis an
Button „Bezahlen“	Leitet den Benutzer zur Bezahlung weiter
Button „Bearbeiten“	Leitet den Benutzer zur Seite „Warenkorb bearbeiten“ weiter

Button „Produktinformationen“	Leitet den Benutzer zur Seite „Produktdetails (Warenkorb)“ weiter

9.6.4 Grafische Prozessdarstellung

Workflow, möglichst detailliert

9.6.5 Lokale Funktionen

Gesamtpreis pro Produkt ausrechnen
Gesamtpreis des Warenkorbs ausrechnen

9.6.6 Akzeptanzkriterien

- Preise werden errechnet und angezeigt
- Produkte und Produktinformationen werden angezeigt

9.6.7 Abgrenzung

- Warenkorb Produkt entfernen
- Warenkorb Produktanzahl ändern
- Produktdetails
- Warenkorb löschen
- Bezahlen

9.6.8 Voraussetzungen

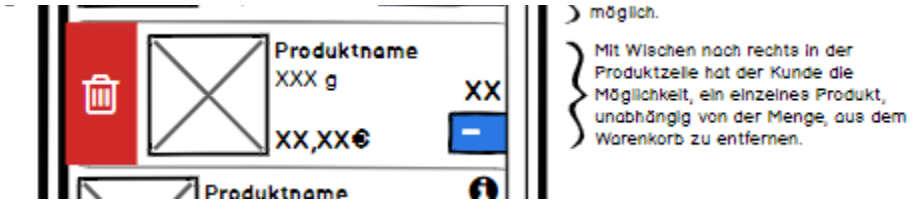
Produkt hinzufügen, Warenkorblogik

9.7 Userstory "Produkt im Warenkorb entfernen"

9.7.1 Beschreibung

"Als Kunde kann ich während des Einkaufens jederzeit in meinem virtuellen Warenkorb Produkte entfernen, sodass diese nicht mehr im Warenkorb vorhanden sind und ich somit das entfernte Produkt nicht bezahlen muss."

9.7.2 Screendesign



9.7.3 Funktionen der Masken

Funktion	Beschreibung
Wischfunktion „Löschen“	Löscht das Produkt aus dem Warenkorb

9.7.4 Lokale Funktionen

Das Produkt wird aus dem Warenkorb entfernt

9.7.5 Akzeptanzkriterien

- Das Produkt ist aus dem Warenkorb entfernt

9.7.6 Abgrenzung

- Warenkorb Produktanzahl ändern
- Produktdetails
- Warenkorb anzeigen

- Warenkorb löschen
- Bezahlen

9.7.7 Voraussetzungen

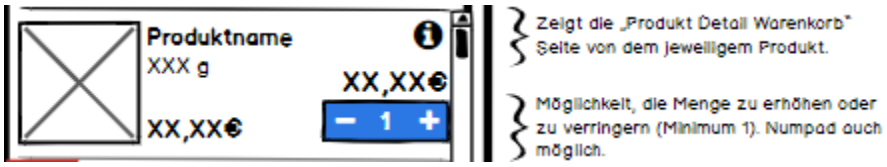
Warenkorblogik

9.8 Userstory "Warenkorb Produktstückzahl erhöhen/verringern"

9.8.1 Beschreibung

" Als Kunde kann ich während des Einkaufens jederzeit in meinem virtuellen Warenkorb die Stückzahl eines bereits eingescannten Produktes erhöhen oder verringern, sodass ich das Produkt im Nachhinein nicht erneut einscannen muss."

9.8.2 Screendesign



9.8.3 Funktionen der Masken

Funktion	Beschreibung
Button „Plus“	Erhöht die Anzahl des Produkts und errechnet die Gesamtsumme
Button „Minus“	Verringert die Anzahl des Produkts und errechnet die Gesamtsumme
Textfeld „Anzahl“	Setzt die Anzahl des Produkts auf die eingegebene Zahl und errechnet die Gesamtsumme

- Lokale Funktionen

Menge im Warenkorb festlegen, Berechnung des neuen Gesamtpreises

9.8.4 Akzeptanzkriterien

- Die Anzahl eines Produkts kann nicht kleiner als 1 sein
- Die Menge des Produkts wird bei Klick auf „Plus“ erhöht
- Die Menge des Produkts wird bei Klick auf „Minus“ verringert
- Die Menge des Produkts wird bei Eingabe in „Anzahl“ gesetzt

9.8.5 Abgrenzung

Warenkorb

9.8.6 Voraussetzungen

Warenkorb, Produkt hinzufügen

9.9 Userstory "Produktdetails (Warenkorb)"

9.9.1 Beschreibung

"Als Kunde kann ich während des Einkaufens jederzeit in meinem virtuellen Warenkorb von allen hinzugefügten Produkten die Details inkl. Bild anzeigen lassen, sodass ich auch im Nachhinein Inhaltsstoffe etc. durchlesen kann."

9.9.2 Funktionen der Masken

Funktion	Beschreibung
Button „Plus“	Erhöht die Anzahl des Produkts und errechnet die Gesamtsumme
Button „Minus“	Verringert die Anzahl des Produkts und errechnet die Gesamtsumme
Textfeld „Anzahl“	Setzt die Anzahl des Produkts auf die eingegebene Zahl und errechnet die Gesamtsumme

Button „Mehr Informationen“	Öffnet die Seite „Mehr Informationen“
Label „Gesamtpreis“	Der errechnete Gesamtpreis
Button „Zurück“	Leitet den Benutzer zur Warenkorbseite zurück
Label „Netto“	Der errechnete Nettopreis
Label „MWSt“	Die gespeicherte Mehrwertsteuer von GetProductDetails

9.9.3 Lokale Funktionen

Nettopreis, Bruttopreis werden aus der Anzahl und den Werten von GetProductDetails errechnet

9.9.4 Akzeptanzkriterien

- Gesamtpreis und Nettopreis werden korrekt errechnet
- Produktinformationen werden falls vorhanden angezeigt

9.9.5 Abgrenzung

Warenkorb

9.9.6 Voraussetzungen

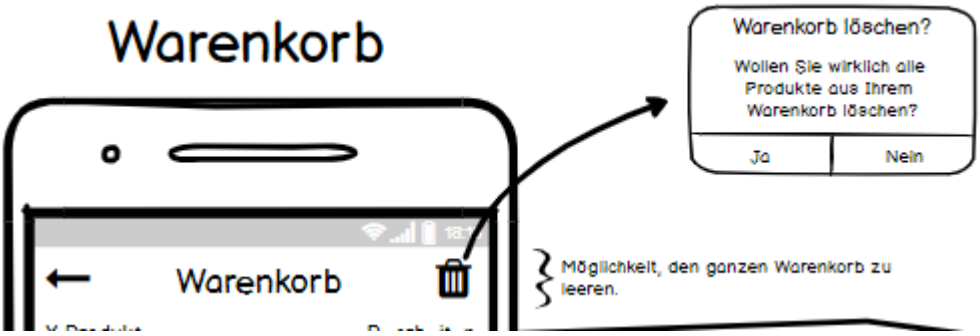
Warenkorblogik, Produkt hinzufügen

9.10 Userstory "Warenkorb löschen"

9.10.1 Beschreibung

"Als Kunde kann ich während des Einkaufens jederzeit den gesamten Warenkorb löschen, sodass ich mit dem Einkaufen im selben Geschäft neu beginnen kann."

9.10.2 Screendesign



9.10.3 Funktionen der Masken

Funktion	Beschreibung
Button „Warenkorb löschen“	Öffnet einen Dialog, in dem der Benutzer bestätigt, dass er den Warenkorb löschen will

9.10.4 Lokale Funktionen

Der Warenkorb wird nach der Bestätigung im Dialogfenster geleert

9.10.5 Akzeptanzkriterien

- Das Dialogfenster mit Ja/Nein wird angezeigt
- Bei Klick auf „Ja“ wird der Warenkorb gelöscht

9.10.6 Abgrenzung

- Warenkorb Produktanzahl ändern
- Produktdetails

- Warenkorb anzeigen
- Produkt aus Warenkorb löschen
- Bezahlen

9.10.7 Voraussetzungen

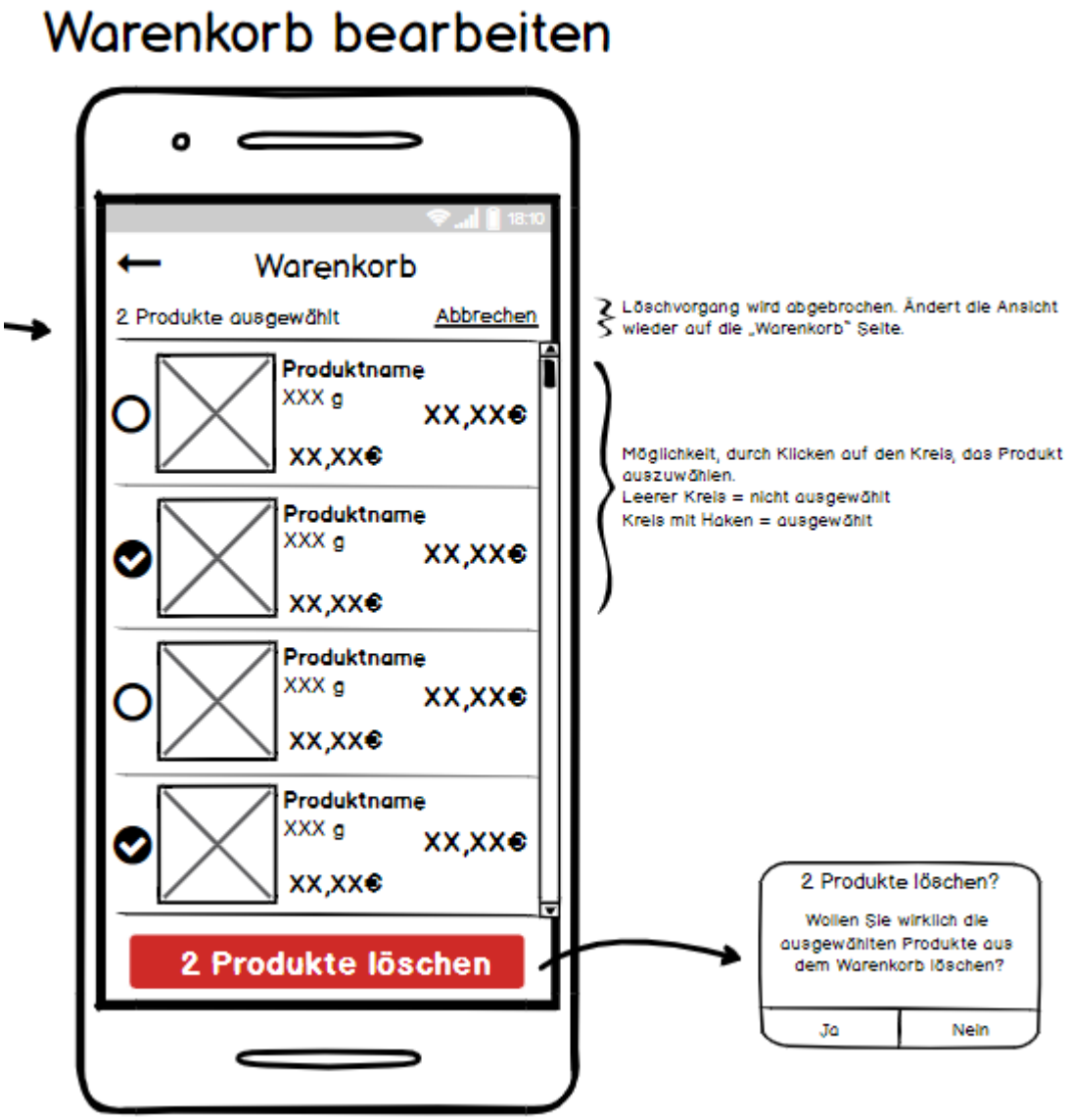
Warenkorblogik

9.11 Userstory "Warenkorb bearbeiten"

9.11.1 Beschreibung

„Als Kunde kann ich im Warenkorb mehrere Produkte gleichzeitig auswählen, sodass ich diese gemeinsam aus dem Warenkorb entfernen kann.“

9.11.2 Screendesign



9.11.3 Funktionen der Masken

Funktion	Beschreibung
Button „Zurück“	Leitet den Benutzer zur Startseite zurück

Label „Produkte ausgewählt“	Zeigt die Anzahl der ausgewählten Produkte an
Button „Abbrechen“	Leitet den Benutzer zum Warenkorb zurück
Button „Löschen“	Löscht die ausgewählten Produkte vom Warenkorb, ausgegraut, wenn nichts ausgewählt ist

9.11.4 Lokale Funktionen

Der Button ist bei einer Auswahl nicht mehr ausgegraut, die Anzahl der ausgewählten Produkte wird angepasst.

9.11.5 Akzeptanzkriterien

- Die ausgewählten Produkte werden aus dem Warenkorb gelöscht
- Der Gesamtpreis des Warenkorbs wird angepasst

9.11.6 Abgrenzung

- Warenkorb Produktanzahl ändern
- Produktdetails
- Warenkorb anzeigen
- Bezahlen

9.11.7 Voraussetzungen

Warenkorblogik

9.12 Userstory "Biometrische Daten"

9.12.1 Beschreibung

"Als Kunde kann ich mit meinen am Smartphone gespeicherten biometrischen Daten die Zahlung starten, sodass sichergestellt wird, dass der Besitzer des Smartphones einkauft und das Smartphone nicht gestohlen ist."

9.12.2 Grafische Prozessdarstellung

Workflow, möglichst detailliert

9.12.3 Lokale Funktionen

Die API für biometrische Verifikation wird aufgerufen und es wird auf das Ergebnis gewartet.

9.12.4 Akzeptanzkriterien

- Es kann auf iOS und Android überprüft werden, ob der Benutzer sich authentifizieren kann.

9.12.5 Abgrenzung

Bezahlung

10 Epic „Start Page“

10.1 Userstory "Home"

10.1.1 Beschreibung

"Als Benutzer werde ich nach Eingabe meines Benutzernamens und meines Passwortes in der Applikation angemeldet und es stehen die Funktionen, die registrierten Benutzern angeboten werden, zur Verfügung"

10.1.2 Screendesign

Bilddatei des Screendesigns & Mockups

10.1.3 Funktionen der Masken

Tabellarische Darstellung: Button oder Feld mit der entsprechenden Funktionsbeschreibung auf Niveau der User Story!

Zum Beispiel:

Funktion	Beschreibung
Button „Anmelden“	Schickt die vom Anwender eingegeben Credentials an die Funktion „Login“ vom NQP Webservice.

10.1.4 Grafische Prozessdarstellung

Workflow, möglichst detailliert

10.1.5 Schnittstellen

10.1.5.1 "Login" Funktion im Webservice des Back End

Welche Parameter werden übergeben? Was sind die Daten, die sich die Applikation als Antwort erwartet.

Oder auch:

10.1.5.2 Authenzifizierung bei Google

10.1.6 Lokale Funktionen

Werden Daten nur in der Applikation verarbeitet (ohne Schnittstellen?) - dann müssen diese hier definiert werden (z.B. Dateien, die geschrieben werden)

10.1.7 Akzeptanzkriterien

Verbale Formulierung, wann die Userstory erfolgreich umgesetzt wurde, z.B.: "Wenn der Kunde sich in ScanBuyGo mit den Back End Funktionen anmelden kann" oder auch "Eine Anmeldung über OAuth bei Google/Facebook durchgeführt werden kann."

10.1.8 Abgrenzung

Was ist in dieser Userstory nicht enthalten (z.B. Registierung, Passwort vergessen) - was ist wichtig zu erwähnen

10.1.9 Voraussetzungen

Was muss zuvor implementiert werden, z.B. Grundservices oder Test-Services im Back End oder notwendige DEV Accounts bei Drittanbietern usw.

11 Epic „Zahlung“

11.1 Userstory "Zur Kasse"

11.1.1 Beschreibung

Als Benutzer wird mir beim Ende des Einkaufs die Möglichkeit gegeben auf die Zahlungsseite zu gehen, um die Summe des Belegs zahlen zu können.

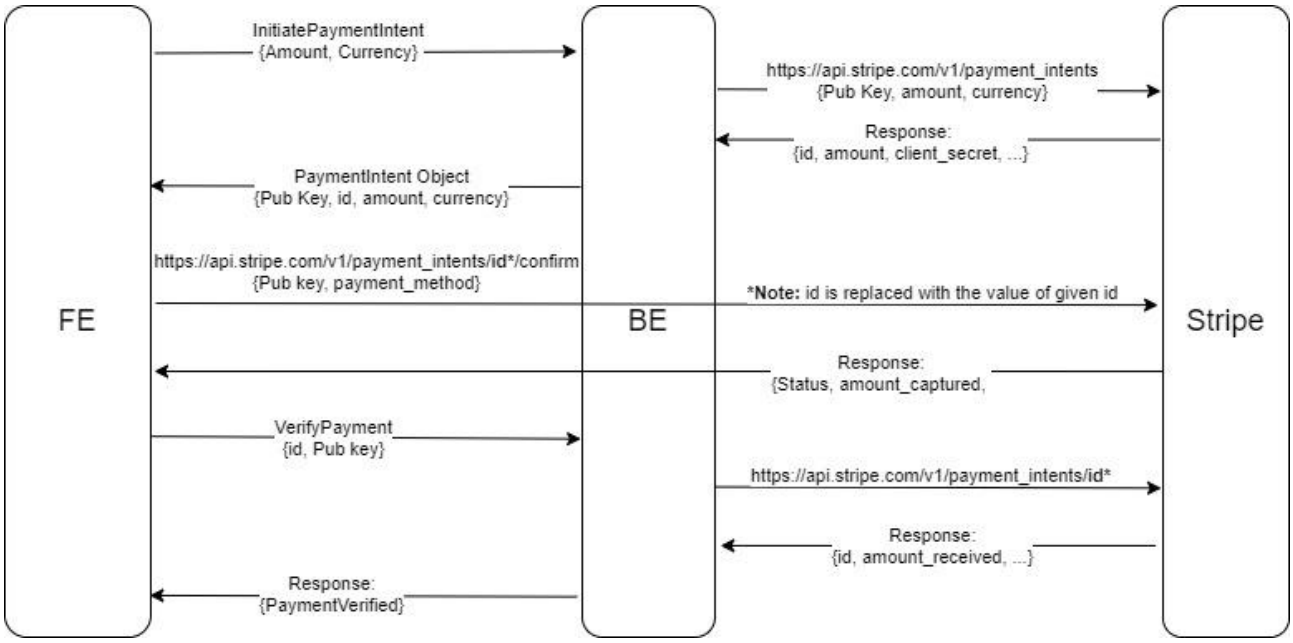
11.1.2 Screendesign

Siehe 9.6.2

11.1.3 Funktionen der Masken

Funktion	Beschreibung
Button „Bezahlen“	Schickt ein Request an Back End für Initiieren eines „Payment Intent“, beim Response von Backend wird der PaymentIntent Object erhalten und auf die Zahlungsseite weitergeleitet

11.1.4 Grafische Prozessdarstellung



11.1.5 Schnittstellen

11.1.5.1 "InitiatePaymenIntent" Funktion im Webservice des Back End

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShoppingSessionToken: 33fdd1b6-b496-4b33-9f7d-df96679d32fe
UDID: 7946DA4E-8429-423C-B405-B3FC77914E3E

Body:

```
{
    "requestDate": "2020-09-07T13:10:59",
    "amount": "9450", //in Cent, entspricht 94,50
    "currency": "Eur" //ISO currency code (3 stellig)
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "paymentIntent ": {
        "PublishableKey": "sk_test_4eC39HqLyjWDarjtT1zdp7dc:",
        "id": "pi_1DigqB2eZvKYlo2CttE9Bif8",
        "amount": 9450,
        "currency": "Eur"
    }
}
```

11.1.6 Lokale Funktionen

11.1.7 Akzeptanzkriterien

Das Backend bekommt das Request für InitiatePaymentIntent und das Frontend springt auf die Zahlungsseite.

11.1.8 Abgrenzung

Zahlungsprozess

11.1.9 Voraussetzungen

Warenkorb voll

11.2 Userstory "Bezahlen"

11.2.1 Beschreibung

Als Benutzer kann ich meine Kartendaten eingeben und eine Zahlung durchführen. Als registrierter Nutzer werden meine hinterlegten Zahlungsdaten (Kreditkarten) vorab ausgefüllt angezeigt.

11.2.2 Screendesign

11.2.3 Funktionen der Masken

Funktion	Beschreibung
Textfeld „Kartennummer“	16 Stellige Nummer, wo der Benutzer seine Kartennummer eingeben kann
Textfeld „Gültig bis“	Gültigkeitsdatum der Karte im Format MM/YY
Textfeld „CVV“	3 Stellige Nummer für den Sicherheitscode
Textfeld „Karteninhaber“	Vollständiger Name der Kartenbesitzer
Slide Checkbox „Kartendaten speichern“	Um die Kartendaten lokal auf dem Gerät zu speichern, damit der Benutzer bei nächster Zahlung nicht mehr die Kartendaten eingeben muss.
Button „Jetzt zahlen“	Bestätigen der Zahlung bei Stripe (Aufruf des Confirm Service von Stripe). Wenn das Response von Stripe positive ist („paid“: true), wird das Backendservice „VerifyPayment“ aufgerufen und das BE anfordern die Zahlung mit Stripe zu prüfen.

11.2.4 Grafische Prozessdarstellung

Siehe 11.1.4

11.2.5 Schnittstellen

11.2.5.1 "Confirm" Endpoint von Stripe

Body:

```
{
Siehe Dokumentation der Stripe API
}
```

Response:

```
{
Siehe Dokumentation der Stripe API
}
```

11.2.5.2 "VerifyPayment" Funktion im Webservice des Back End

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShoppingSessionToken: 33fdd1b6-b496-4b33-9f7d-df96679d32fe
UDID: 7946DA4E-8429-423C-B405-B3FC77914E3E

Body:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "PublishableKey": "sk_test_4eC39HqLyjWDarjtT1zdp7dc:",
    "id": "pi_1DigqB2eZvKYlo2CttE9Bif8"
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
}
```

11.2.6 Lokale Funktionen

Es wird geprüft im vorab, ob die eingegebenen Daten in Textfelder der definierten Richtlinien entsprechen.

Beim Aktivieren der Slide Checkbox, werden die eingegebenen Kartendaten lokal auf dem Gerät gespeichert.

Eine Bestätigung von Zahlung wird dem Benutzer angezeigt, erst wenn das Response von VerifyPayment positiv ist („success“: true)

11.2.7 Akzeptanzkriterien

Die Zahlung bei Stripe erfolgreich ist.

Die Zahlungsbestätigung ist auf FE ersichtlich

11.2.8 Abgrenzung

Anlegen eines Handlerkonto bei Stripe

11.2.9 Voraussetzungen

Der Handler muss ein gültiges Konto bei Stripe angelegt haben.

12 Spezifikationsaufbau für Back End

Der Aufbau der Struktur für die Spezifikation der Back End Prozesse soll wie folgt aufgebaut werden:

- Epic
    - Userstory 1

            - Schnittstellen
                - WS Funktion A ... Z
            - Datenstrukturen DB
                - Tabellen
                - Felder (= Feld + Datentypen + PK + FK + Index + Mandatory/Defaults)
            - Lokale Funktionen
            - Akzeptanzkriterien
            - Abgrenzung zu anderen Prozessen
            - Voraussetzungen
        - Userstory 2
            - …
        - Userstory N
- Epic 2
- …
- Epic N

13 Epic Registrierung

13.1 Userstory Registrierung

13.1.1 Beschreibung

"Als Webservice können wir Registrierungsrequests verarbeiten, sodass ein Account mit persönlichen Daten* in die Customer Tabelle eingefügt wird. Es wird eine E-Mail-Verifikation versendet und der Code wird in die Email_Verification Tabelle eingefügt.“

* Vorname, Zuname, E-Mail, Geburtsdatum, Straße, Hausnummer, …"

13.1.2 Schnittstellen

Siehe Kapitel 8.1.5 für Request und Response Daten

13.1.3 Datenbankstrukturen DB

Tabelle: Customer					
Feldname	Datentyp	PK	FK	Nullable	Default
CustomerID	**VARCHAR(200)**	**true**	**false**	**false**	**AUTOINC**
Firstname	VARCHAR(200)	false	false	false	
Lastname	VARCHAR(200)	false	false	false	
Hash	VARCHAR(200)	false	false	false	
Email	VARCHAR(200)	false	false	false	
Birthdate	DATE	false	false	false	
Street	VARCHAR(200)	false	false	true	null
House_number	VARCHAR(50)	false	false	true	null
ZIP_code	VARCHAR(30)	false	false	true	null
Country	VARCHAR(200)	false	false	true	null
City	VARCHAR(200)	false	false	true	null
VAT_ID	VARCHAR(20)	false	false	true	null
Gender	VARCHAR(1)	false	false	true	null
Phone	VARCHAR(50)	false	false	true	null
Lockout_TS	TIMESTAMP	false	false	true	null
Registration_TS	TIMESTAMP	false	false	false	Current_TS
Two_FA_secret	VARCHAR(200)	false	false	true	null
Verification_TS	TIMESTAMP	false	false	true	null
Provider	VARCHAR(200)	false	false	true	null

Abbildung 1: Customer Tabelle DB-Struktur

Feldname	Datentyp	Nullable	FK	PK	Default
SendID	**INT**	**false**	**false**	**true**	**AutoINC**
CustomerID	INT	false	true	false	
VerifyCode	INT	false	false	false	
Email	VARCHAR(200	false	false	false	
Send_TS	TIMESTAMP	false	false	false	Current TS
Result	BOOL	false	false	false	false

Abbildung 2: Email_Verification Tabelle DB-Struktur

13.1.4 Lokale Funktionen

13.1.4.1 generateVerificationCode

Hierbei wird ein Bestätigungscode generiert.

13.1.4.2 sendVerificationEmail

Hierbei wird eine E-Mail versendet, welche der Nutzer in der App, mit dem in der E-Mail enthaltenen Code, bestätigen muss. Dieser Prozess wird samt Ergebnis (OK, nicht OK) in der Email_Verification Tabelle protokolliert.

13.1.4.3 deleteUnverifiedCustomers

Wenn der Nutzer seine angegebene E-Mail nicht innerhalb einer Woche verifiziert, wird der Datensatz aus der Customer Tabelle gelöscht.

13.1.4.4 logRequest

Alle empfangenen Requests* werden in einer Logdatei (*.log) protokolliert.

* Timestamp, IP, RequestBody

13.1.4.5 logError

Alle möglichen Fehler* werden in einer Logdatei (*.log) protokolliert.

13.1.5 Akzeptanzkriterien

Nutzer wird in die Customer Tabelle eingefügt; Verifikationsemail wird versendet; Response wird an den Client gesendet.

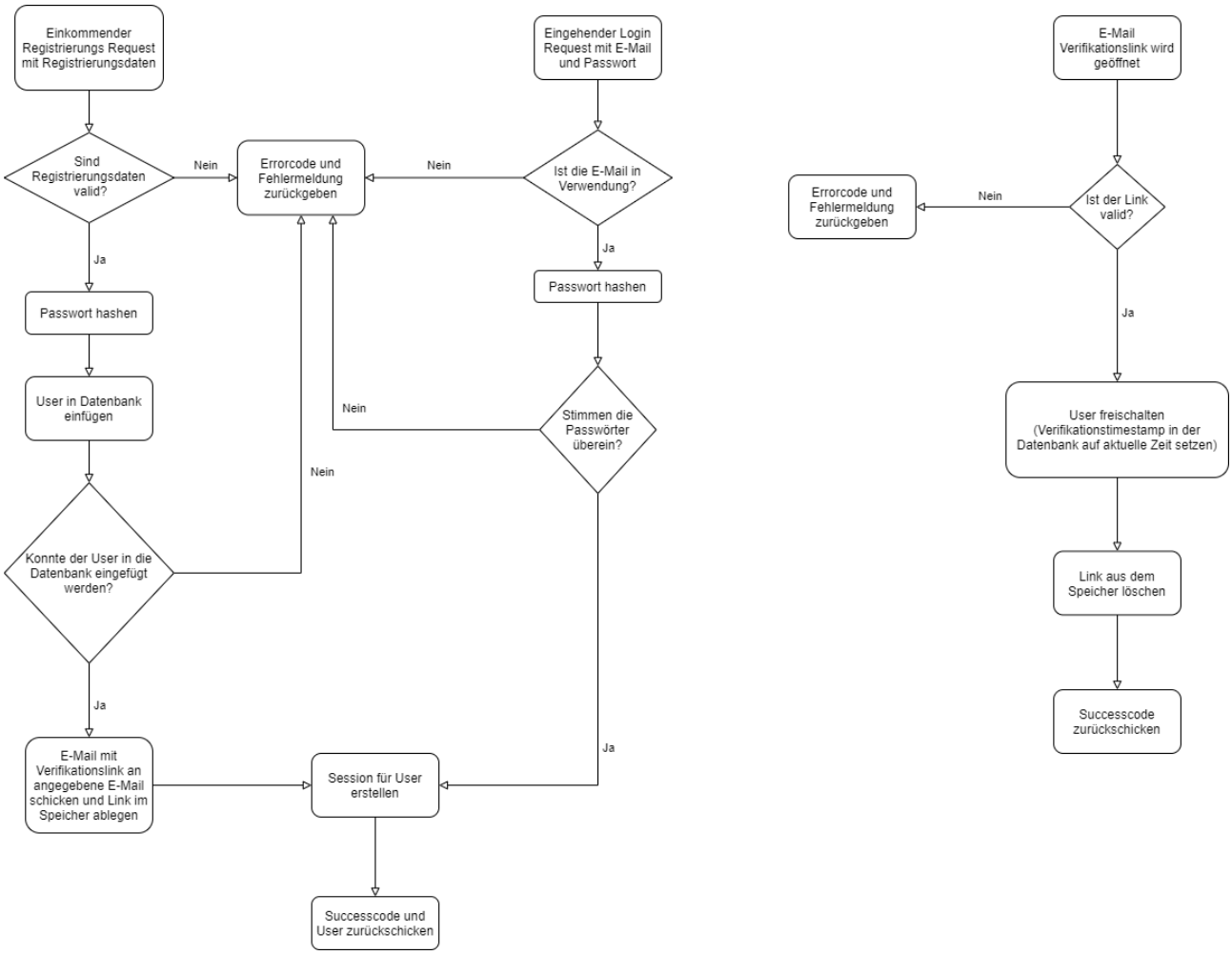
13.1.6 Abgrenzung zu anderen Prozessen

Abgrenzung anderer Userstories (wie zB Login)

Bei fehlgeschlagenem E-Mail Versand wird keine weitere Rückmeldung an den User gesendet.

13.1.7 Voraussetzungen

13.1.8 Grafische Prozessdarstellung



13.2 Userstory "Registrierung mit externem Anbieter"

13.2.1 Beschreibung

„Als Webservice ermöglichen wir es einem Nutzer sich mit externen Anbietern, wie Google oder Facebook, zu registrieren. Es wird überprüft ob ein Account mit der E-Mail bereits vorhanden ist. Der Nutzer muss zusätzliche nötige Daten angeben, um die Registrierung abzuschließen. Nach Abschluss der Registrierung wird noch einmal der Token überprüft und Verification_TS auf den aktuellen Zeiptunkt gesetzt. Danach wird das Feld provider auf den Provider gesetzt.“

13.2.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Token ungültig	401
Request timeout	Externer Anbieter nicht erreichbar	408
Konflikt	E-Mail oder Account ist bereits in Verwendung.	409
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

13.2.3 Lokale Funktionen

13.2.3.1 getExternalUserInformation

Holt die Informationen mit dem Token vom externen Anbieter und liefert ein Objekt mit relevanten Informationen, Name, Adresse, E-Mail-Adresse, Telefonnummer, falls vorhanden, zurück.

13.2.3.2 checkIfUserAlreadyExists

Überprüft, ob die E-Mail des externen Accounts bereits in der Customer Tabelle eingetragen ist.

13.2.4 Akzeptanzkriterien

Nutzer kann sich mit externen Anbietern registrieren; Nutzer wird in der Customer Tabelle mit samt Provider gespeichert

13.2.5 Abgrenzung zu anderen Prozessen

- Login
- Registrierung

13.2.6 Grafische Prozessdarstellung

13.3 Userstory "E-Mail bestätigen"

13.3.1 Beschreibung

„Als Webservice verifizieren wir einen Benutzer, indem wir den von ihm eingegebenen Verifikationscode mit dem aus der Email_Verification Tabelle abgleichen. Der eingegebene Code muss in mindestens einer E-Mail, welche vor nicht mehr als 24 Stunden gesendet wurde, enthalten gewesen sein.“

13.3.2 Schnittstellen

Siehe Kapitel 8.3.5.1 (VerifyEmail), 8.3.5.2 (ResendEmail) für Request und Response Daten.

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Falscher Bestätigungscode	401
Konflikt	E-Mail ist bereits verifiziert	409
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

13.3.3 Datenbankstrukturen DB

Siehe Kapitel 11.1.3

13.3.4 Lokale Funktionen

13.3.4.1 deleteFormerVerificationCodes

Zuvor generierte Verifikationscodes, die dem bestätigten Nutzer zugeordnet sind, werden nach erfolgreicher Verifizierung gelöscht.

13.3.4.2 verifyEmail

Prüft anhand der Email_Verification Tabelle, ob der Code einer innerhalb von 24 Stunden an den Customer gesendeten Email zugeordnet ist und setzt dann den Verification_TS in der Customer Tabelle auf den Zeitpunkt der Bestätigung.

13.3.4.3 resendEmail

Ein neuer Bestätigungscode wird generiert, in die Email_Verification Tabelle geschrieben, und anschließend per E-Mail an den Nutzer versendet. Dieser Bestätigungscode ist 24 Stunden gültig.

13.3.5 Akzeptanzkriterien

13.3.5.1 verifyEmail:

Verification_TS wird gesetzt; Alte Verifikationscodes werden gelöscht; Statuscode wird zurückgegeben; Benutzer kann sich nach erfolgreicher Verifizierung anmelden

13.3.5.2 resendEmail:

Neue E-Mail wird an richtige E-Mail versendet; neue Zeile wird in der Email_Verification Tabelle angelegt

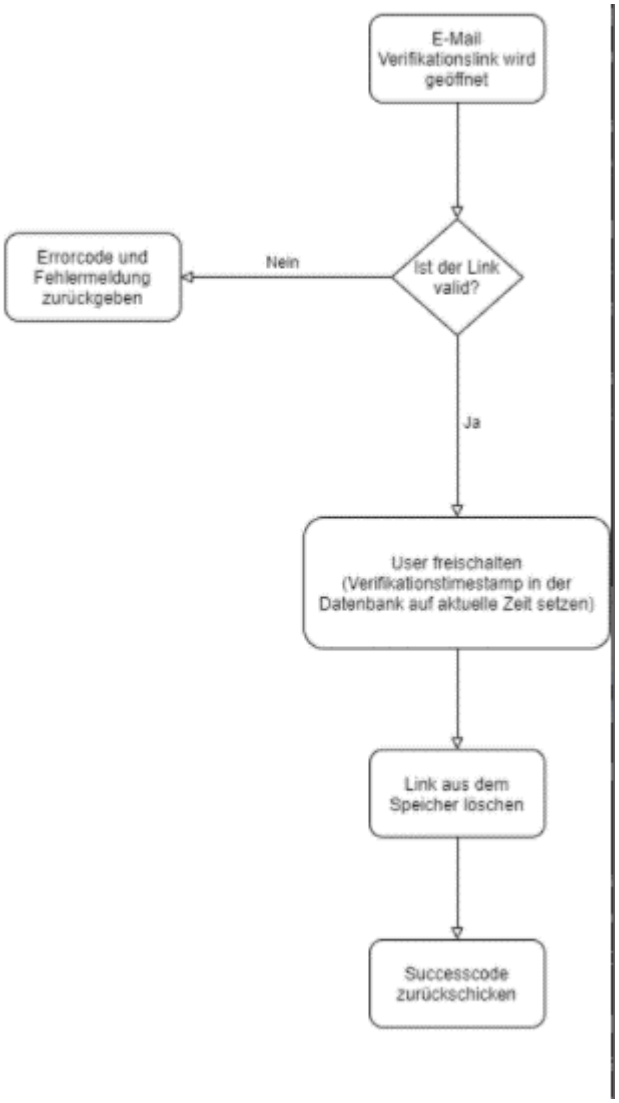
13.3.6 Abgrenzung zu anderen Prozessen

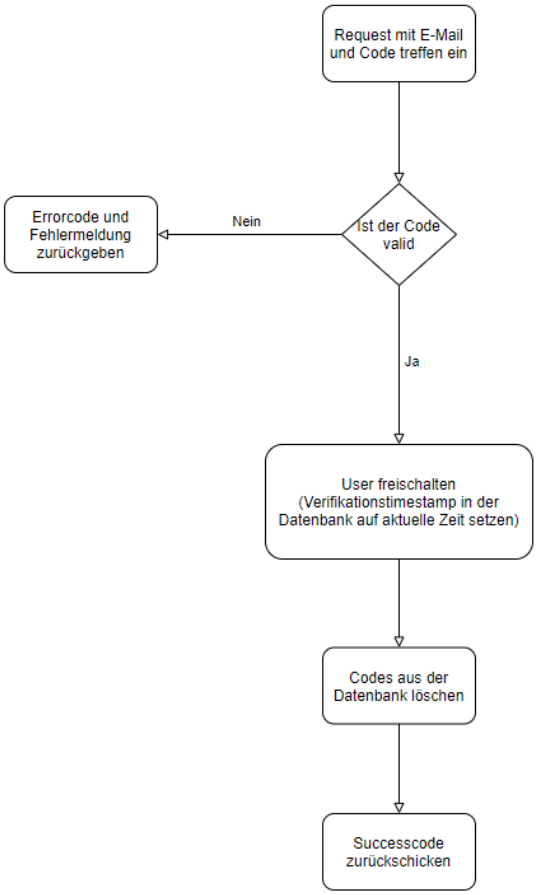
- Login
- Registrierung

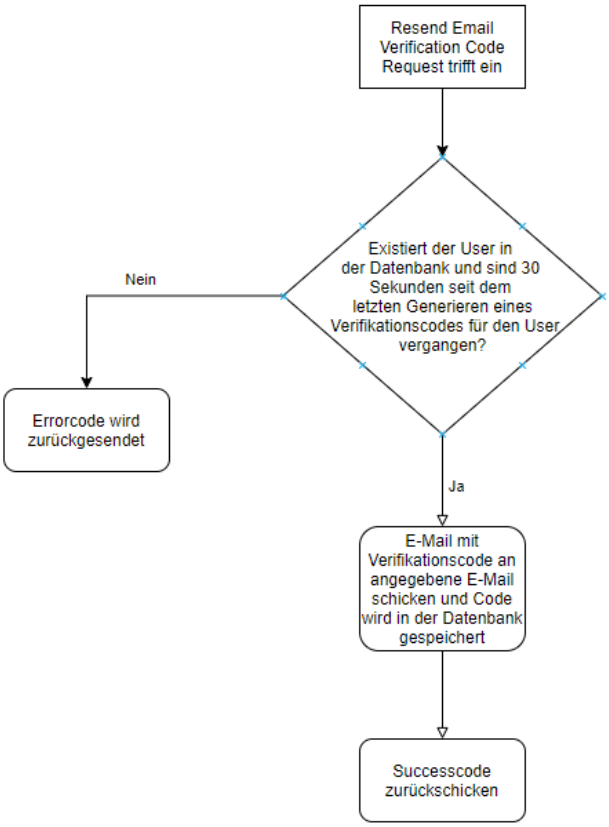
13.3.7 Voraussetzungen

Verifizierungscode wurde gesendet und Customer existiert in der Datenbank.

13.3.8 Grafische Prozessdarstellung




Request mit E-Mail und Code treffen ein
Ist der Code valid
Nein
Errorcode und Fehlermeldung zurückgeben
Ja
User freischalten (Verifikationstimestamp in der Datenbank auf aktuelle Zeit setzen)
Codes aus der Datenbank löschen
Successcode zurückschicken



14 Epic “Login”

14.1 Userstory "Login"

14.1.1 Beschreibung

"Als Webservice können wir Loginrequests verarbeiten, sodass ein registrierter Benutzer mit verifizierter E-Mail-Adresse sich anmelden kann und eine SessionID zurückbekommt. Die SessionID wird in der Tabelle Customer_Sessions gespeichert.“

14.1.2 Schnittstellen

POST Request:

```
{
    "email": "email@example.com",
    "password": "P@ssw0rd",
    "requestDate": "2020-09-07T13:10:59"
}
```

Response:

```
{
    "responseDate": "2020-09-07T13:10:59",
    "success": true,
    "sessionId": "33fdd1b6-b496-4b33-9f7d-df96679d32fe"
}
```

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Das Login war erfolgreich	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Falsche Zugangsdaten	401
Forbidden	Unbestätigter Nutzer versucht sich mit korrektem Kennwort anzumelden. E-Mail-Bestätigung ausständig.	403
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423
Nutzer gesperrt	Gesperrter Nutzer versucht sich mit korrektem Kennwort anzumelden.	480

14.1.3 Datenbankstrukturen DB

Feldname	Typ	PK	FK	Null	Default
SessionID	**VARCHAR(200)**	**true**	**false**	**false**	
CustomerID	INT	false	true	false	
Init_TS	timestamp	false	false	false	Current TS
Lastconnection	timestamp	false	false	false	Current TS

14.1.4 Lokale Funktionen

14.1.4.1 generateSessionID

Zufällige Session ID wird generiert und in die Customer_Sessions Tabelle eingetragen.

14.1.4.2 verifyCredentials

Wenn Credentials gültig sind und der Customer verifiziert ist und nicht gesperrt ist, gibt er eine neue Session zurück, welche ein erfolgreiches Login impliziert. Bei fehlerhaftem Login wird keine SessionID zurück geschickt sondern ein Statuscode.

14.1.5 Akzeptanzkriterien

Session wird in der Customer_Sessions Tabelle angelegt; Nutzer kann sich nur mit gültigen Zugansdaten einloggen;

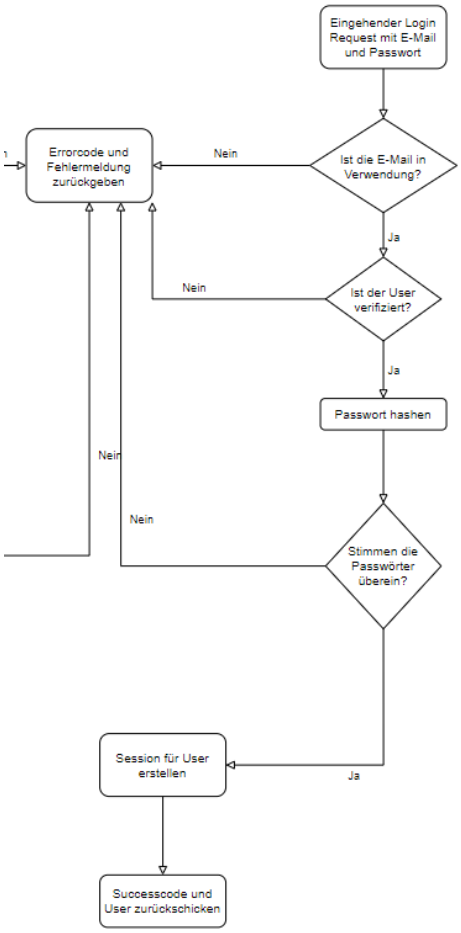
14.1.6 Abgrenzung zu anderen Prozessen

- Registrieren
- Passwort ändern
- Passwort vergessen
- E-Mail bestätigen

14.1.7 Voraussetzungen

Es gibt einen gültigen Customer der zur Anmeldung berechtigt ist.

14.1.8 Grafische Prozessdarstellung



14.2 Userstory „Login mit externem Anbieter“

14.2.1 Beschreibung

„Als Webservice ermöglichen wir es einem Nutzer sich mit externen Anbietern, wie Google oder Facebook, einzuloggen.“

14.2.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Falsche Nutzerdaten	401
Forbidden	Gesperrter Nutzer versucht sich mit korrektem Kennwort anzumelden.	403
Request timeout	Provider Request abgelaufen	408
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

14.2.3 Lokale Funktionen

14.2.3.1 checkIfUserIsRegisteredWithAccordingProvider

Überprüft, ob die E-Mail des externen Accounts mit dem Provider verknüpft ist.

14.2.4 Akzeptanzkriterien

Nutzer kann sich mit externen Anbietern anmelden und es wird eine Session in der Customer_Sessions Tabelle angelegt

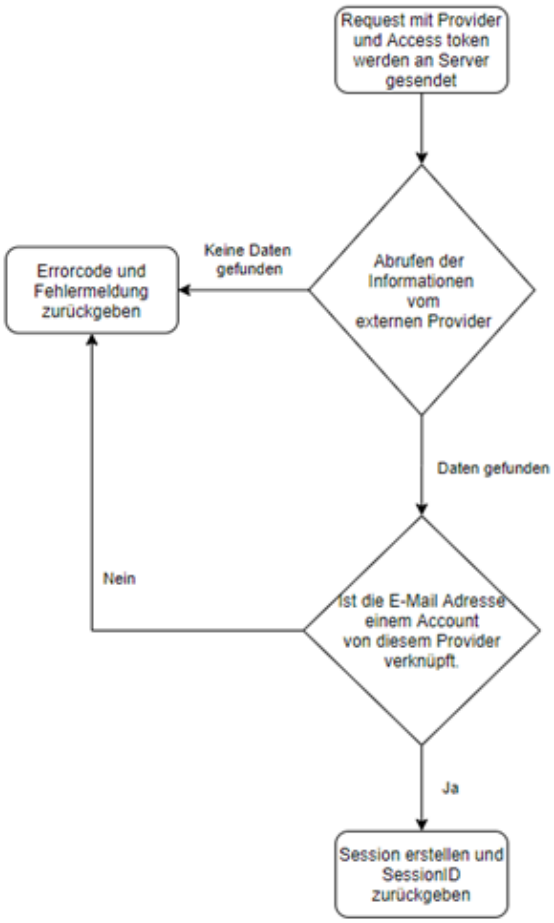
14.2.5 Abgrenzung zu anderen Prozessen

- Login
- Registrierung

14.2.6 Voraussetzungen

Gültiger Nutzer muss in Customer Tabelle gespeichert sein

14.2.7 Grafische Prozessdarstellung



14.3 Userstory "Gastmodus"

14.3.1 Beschreibung

"Als Webservice stellen wir Nutzern, die sich keinen Account erstellen wollen, die Möglichkeit zur Verfügung, ohne Account Registrierung eine Session starten zu können.“

14.3.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500

Invalid	Invalide (JSON) Daten.	400
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

14.3.3 Datenbankstrukturen DB

Siehe 12.1.3

14.3.4 Lokale Funktionen

14.3.4.1 generateSessionID

Zufällige Session ID wird generiert und mit dem Gast Account verknüpft in die Customer_Sessions Tabelle eingetragen.

14.3.5 Akzeptanzkriterien

Session wird in der Customer_Sessions Tabelle angelegt; Nutzer kann die App im Gast Modus benutzen

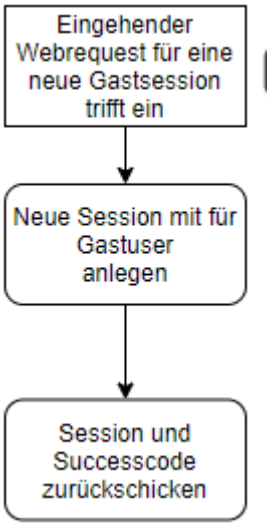
14.3.6 Abgrenzung zu anderen Prozessen

- Registrieren
- Login

14.3.7 Voraussetzungen

Es gibt einen Gast Account in der Customer Tabelle (ID = 0815).

14.3.8 Grafische Prozessdarstellung



14.4 Userstory "Passwort vergessen"

14.4.1 Beschreibung

"Als Webservice stellen wir bereits registrierten Nutzern die Möglichkeit zur Verfügung, ihr Passwort mithilfe eines 6-stelligen Codes, welcher an die Nutzer E-Mail gesendet wird, zurückzusetzen und ein neues Passwort zu wählen. Um eine unautorisierten Zugriff zu verhindern, mit der das Passwort nach der Verifizierung des Codes zurückgesetzt werden könnte, schicken wir dem Client eine temporäre VerificationSessionID, mit der sichergestellt werden kann, dass es sich auch beim Setzen des neuen Passworts um den gleichen Client handelt.“

14.4.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Der Request war erfolgreich	200
Nutzer nicht gefunden	Der Nutzer wurde nicht gefunden.	204

Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Falscher Verifizierungscode oder falsche Session	401
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

14.4.3 Datenbankstrukturen DB

Feldname	Datentyp	PK	FK	Nullable	Default
SendID	**INT**	**true**	**false**	**false**	**AutoINC**
Verification_Session_ID	VARCHAR(200)	false	false	false	
CustomerID	INT	false	true	false	
Confirmation_TS	timestamp	false	false	true	null
Send_TS	timestamp	false	false	false	Current TS
Code	INT	false	false	false	
Email	VARCHAR(200)	false	false	false	

Abbildung 3: Tabelle PasswordResetVerificationCode

14.4.4 Lokale Funktionen

14.4.4.1 generateVerificationSessionID

Zufällige temporäre Session ID wird generiert und in die PasswordResetVerification Tabelle eingetragen.

14.4.4.2 sendPasswordResetEmail

E-Mail zur Passwortzurücksetzung wird mitsamt Code an die E-Mail-Adresse des Customers versendet.

14.4.4.3 generatePasswordResetVerificationCode

Zufälliger sechsstelliger Code wird generiert und in die PasswordResetVerificationCode Tabelle geschrieben.

14.4.4.4 invalidateOutdatedUserSessions

Alte, mit diesem User Usersessions in der Customer_Session Tabelle werden durch das setzen von “Invalidation_TS” beendet, da diese noch mit dem alten Passwort angemeldet sind.

14.4.4.5 deleteOutdatedPasswordResetVerificationCodes

Zuvor generierte Verifikationscodes, die dem bestätigten Nutzer zugeordnet sind, werden nach erfolgreicher Verifizierung beziehungsweise nach frühestens 24 Stunden und spätestens 48 Stunden aus der PasswordResetVerification Tabelle gelöscht.

14.4.5 Akzeptanzkriterien

Passwortzurücksetzungsemail mitsamt Passwortzurücksetzungscode wird versendet; Nutzer kann das Passwort mithilfe des Codes ändern; Alte (outdated) Sessions werden beendet

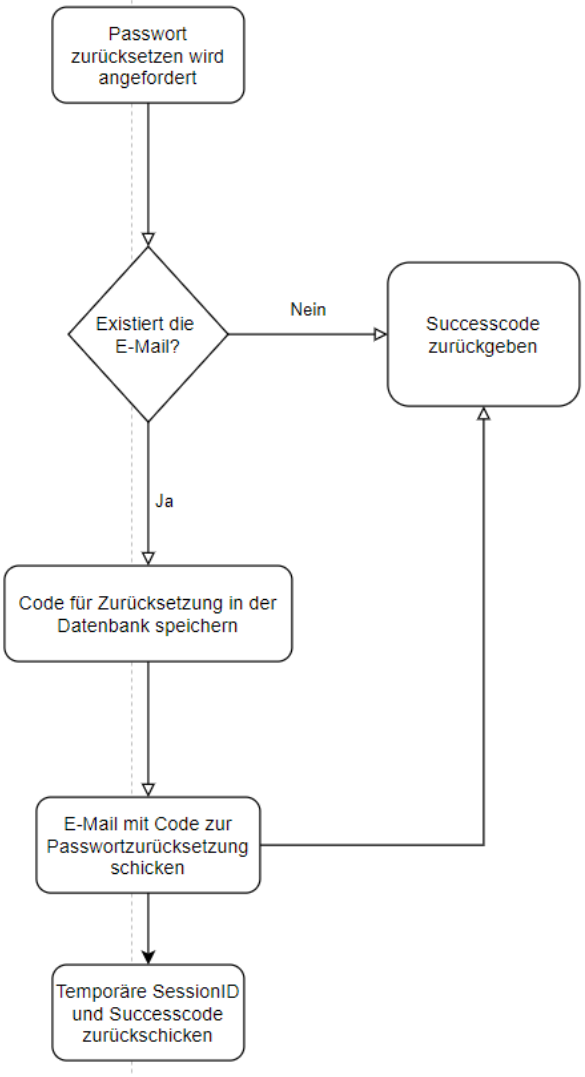
14.4.6 Abgrenzung zu anderen Prozessen

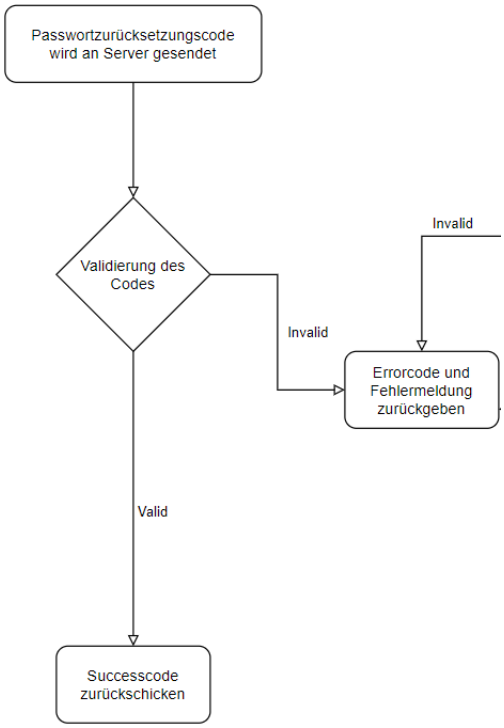
- Passwort ändern
- Login

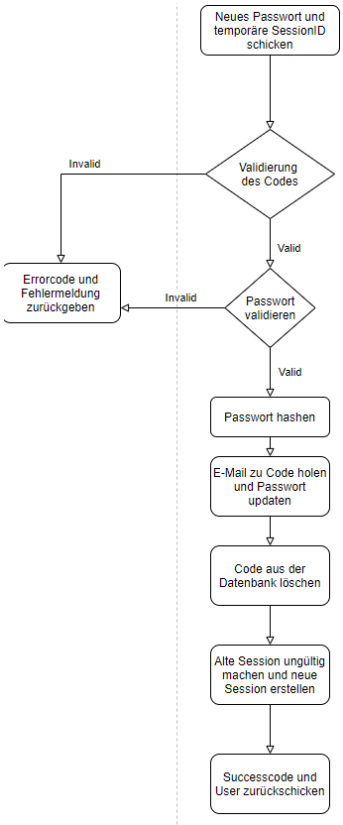
14.4.7 Voraussetzungen

Es gibt einen gültigen Customer der zur Anmeldung berechtigt ist.

14.4.8 Grafische Prozessdarstellung







15 Epic „Shop Assignment“

15.1 Userstory "QR-Code scannen"

15.1.1 Beschreibung

"Als Webservice ermöglichen wir es dem Nutzer einen QR-Code eines Shops zu scannen, um in diesem den Einkauf zu beginnen.”

15.1.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Eine Session wurde für den User im Shop angelegt.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Ungültige SessionID	401
Forbidden	Shop ID existiert nicht	403
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

15.1.3 Datenbankstrukturen DB

Feldname	Typ	PK	Null	Default
StyleID	**INT**	**true**	**false**	**AutoINC**
Primary_Color	BYTEA(6)	false	false	
Secondary_Color	BYTEA(6)	false	false	
Primary_Text_Color	BYTEA(6)	false	false	
Secondary_Text_Color	BYTEA(6)	false	false	

Abbildung 1: Style Tabelle DB-Struktur

Feldname	Typ	PK	FK	Nullable	Default
MerchantID	**INT**	**true**	**false**	**false**	**AutoINC**
UID_Nr	VARCHAR(200)	false	false	false	
Phone	VARCHAR(50)	false	false	false	
Email	VARCHAR(200)	false	false	false	
Fax	VARCHAR(200)	false	false	true	null
Logo	VARCHAR(1024)	false	false	false	
Commercial_register_court	VARCHAR(200)	false	false	true	null
Commercial_register_number	VARCHAR(200)	false	false	true	null
VAT_ID	VARCHAR(200)	false	false	false	
Style	INT	false	true	false	

Abbildung 2: Merchant Tabelle DB-Struktur

Feldname	Typ	PK	FK	Nullable	Default
ShopID	**INT**	**true**	**false**	**false**	**AutoINC**
MerchantID	**INT**	**true**	**true**	**false**	
Phone	VARCHAR(50)	false	false	false	
Email	VARCHAR(200)	false	false	false	
Fax	VARCHAR(200)	false	false	true	null
Opening_hour	INT	false	false	true	null
Closing_hour	INT	false	false	true	null
Street	VARCHAR(200)	false	false	false	
House_number	VARCHAR(50)	false	false	false	
ZIP_code	VARCHAR(20)	false	false	false	
Country	VARCHAR(200)	false	false	false	
City	VARCHAR(200)	false	false	false	
Longitude	DECIMAL(4,4)	false	false	true	null
Latitude	DECIMAL(4,4)	false	false	true	null

Abbildung 1 Shop Tabelle DB-Struktur

Feldname	Typ	PK	FK	Nullable	Default
ShopID	**INT**	**true**	**true**	**false**	
MerchantID	**INT**	**true**	**true**	**false**	
SessionID	**VARCHAR(200)**	**true**	**true**	**false**	
Start_TS	TIMESTAMP	false	false	false	Current ts
End_TS	TIMESTAMP	false	false	true	null

Abbildung 2 Shop_Sessions Tabelle DB-Struktur

15.1.4 Lokale Funktionen

15.1.4.1 createSessionForUser

Eine Session für den User wird im zum Code zugehörigen Shop angelegt, alle noch offenen alten Einträge in der Shop_Sessions Tabelle werden beendet (indem der End_TS gesetzt wird), sofern sie der SessionID zugewiesen sind.

15.1.4.2 getStoreByCode

Ein Shop kann anhand seiner ID + MerchantID gefunden werden.

15.1.5 Akzeptanzkriterien

User kann sich mit dem QR Code im Shop anmelden; Session wird in der Shop_Sessions Tabelle generiert und Start_TS wird auf aktuellen Zeitpunkt gesetzt; Alte Sessions werden gelöscht

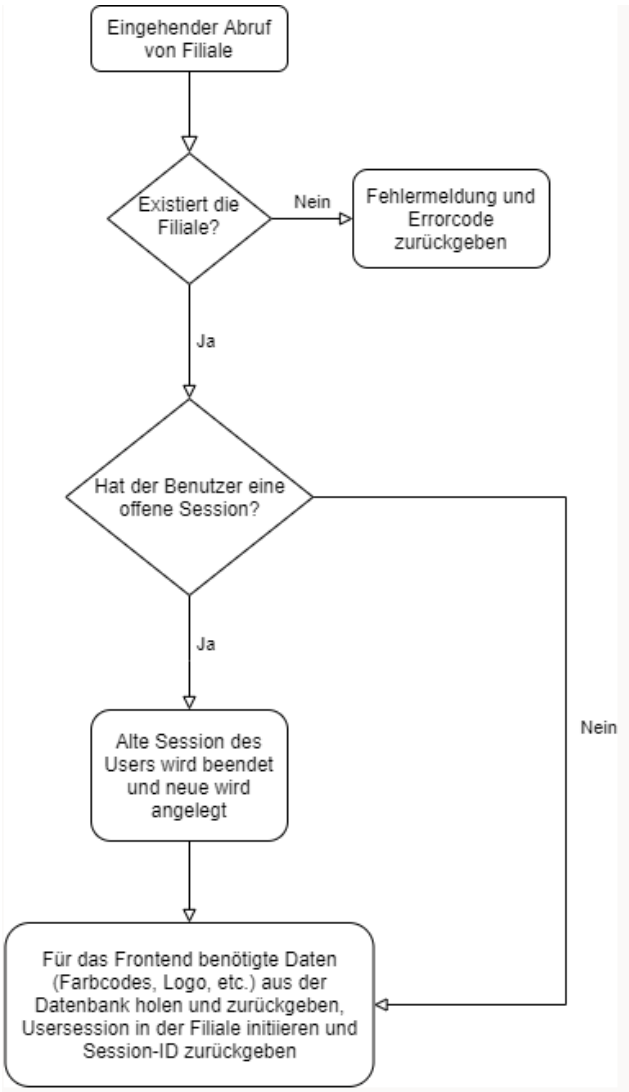
15.1.6 Abgrenzung zu anderen Prozessen

- Shop verlassen
- Shops in meiner Nähe

15.1.7 Voraussetzungen

Gültiger User existiert; Shop und Merchant sind in der Datenbank vorhanden;

15.1.8 Grafische Prozessdarstellung



15.2 Userstory "Shops in meiner Nähe"

15.2.1 Beschreibung

"Als Webservice kann ich anhand von Geokoordinaten eines Benutzers herausfinden, welche Shops in der Nähe sind und diese gereiht nach Distanz zurückgeben."

15.2.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request	401
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

15.2.3 Lokale Funktionen

15.2.3.1 calculateDistanceToShop

Berechnet Entfernung von User zu Shop.

15.2.3.2 sortShopsByDistance

Sortiert eine Liste an Shops nach Entfernung.

15.2.3.3 findNearbyShopsByLocation

Findet Shops in der Nähe der gesendeten Location innerhalb eines gewissen konfigurierbaren Radius.

15.2.4 Akzeptanzkriterien

Shops in der Nähe werden korrekt angezeigt und in der richtigen Reihenfolge (der Shop, der am nächsten ist, zuerst) zurückgegeben.

15.2.5 Abgrenzung zu anderen Prozessen

- Shop manuell verlassen
- QR-Code scannen

15.2.6 Voraussetzungen

Es gibt einen Shop mit Longitude und Latitude, es gibt einen bestätigten Customer.

15.2.7 Grafische Prozessdarstellung

15.3 Userstory "Shop manuell verlassen"

15.3.1 Beschreibung

„Als Webservice stellen wir Nutzern, welche bereits in einem Shop eingeloggt sind, die Möglichkeit zur Verfügung, den Shop manuell verlassen zu können, ohne den Einkauf abschließen zu müssen.“

15.3.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request.	401
Request timeout	Request timeout	408
Konflikt	Shop wurde bereits verlassen	409
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

15.3.3 Datenbankstrukturen DB

15.3.4 Lokale Funktionen

15.3.4.1 endCurrentSession

Beim Verlassen des Shops wird die aktuelle Session beendet und somit inaktiv, indem der End_TS in der Shop_Sessions Tabelle auf den aktuellen Timestamp gesetzt wird.

15.3.5 Akzeptanzkriterien

Nutzer kann den Shop manuell verlassen; Aktuelle Session wird beendet Session wird in der Customer_Sessions Tabelle angelegt

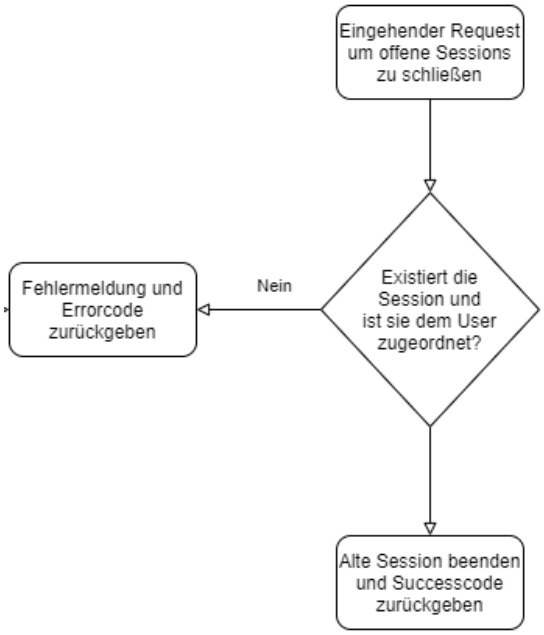
15.3.6 Abgrenzung zu anderen Prozessen

- Shop betreten
- Shop verlassen

15.3.7 Voraussetzungen

Customer muss angelegt sein; Nutzer muss Shop betreten können

15.3.8 Grafische Prozessdarstellung



16 Epic „Shopping Cart“

16.1 Userstory "Produkt per Scan hinzufügen"

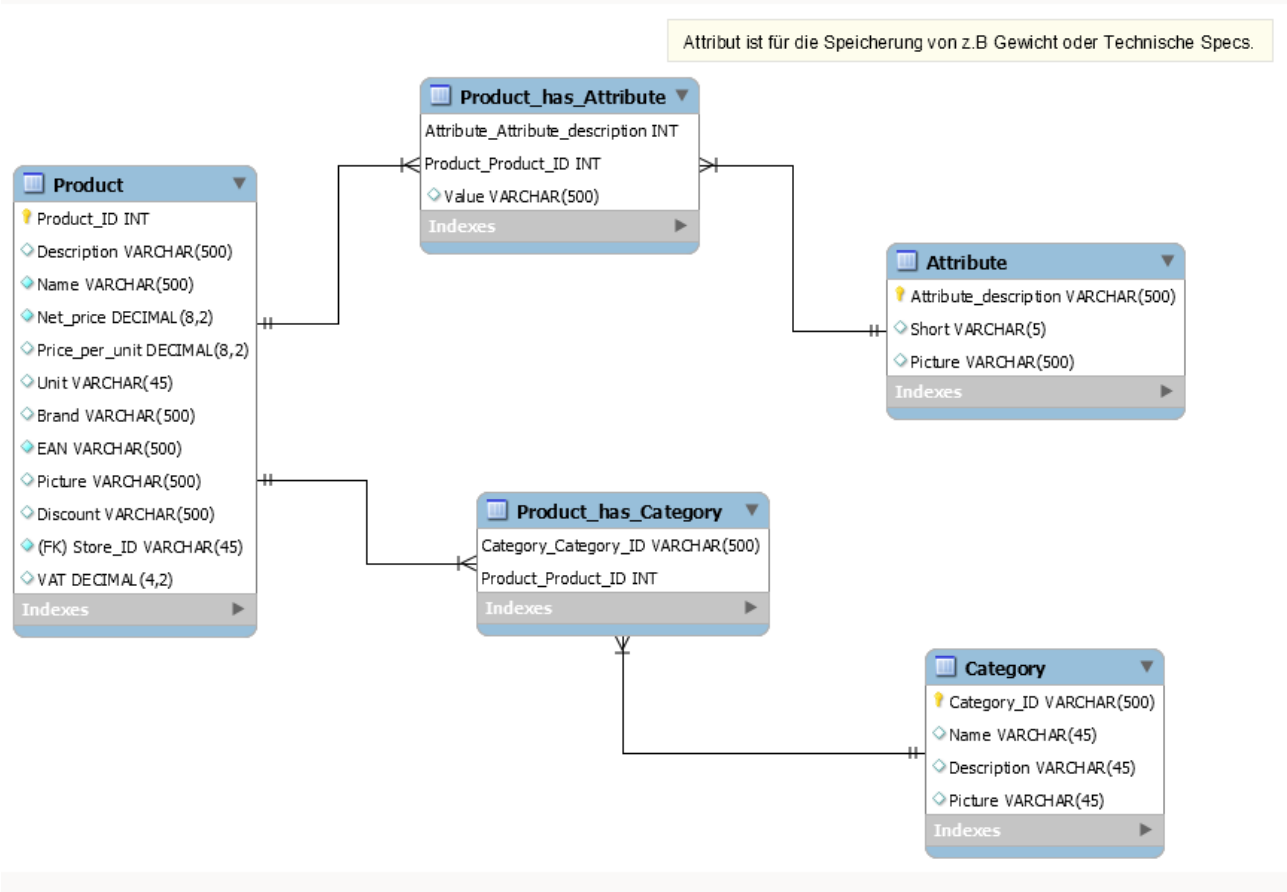
16.1.1 Beschreibung

"Als Webservice ermöglichen wir es dem Nutzer, über den Barcode ein Produkt zu finden und dieses samt Details zurückzugeben.“

16.1.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request	401
Forbidden	Falscher Barcode	403
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

16.1.3 Datenbankstrukturen DB



16.1.4 Lokale Funktionen

16.1.4.1 findProductByBarcode

Liefert ein Produkt zurück, das einen gewissen Barcode hat.

16.1.5 Akzeptanzkriterien

Produkt wird anhand eines gültigen Barcodes samt Details zurückgegeben

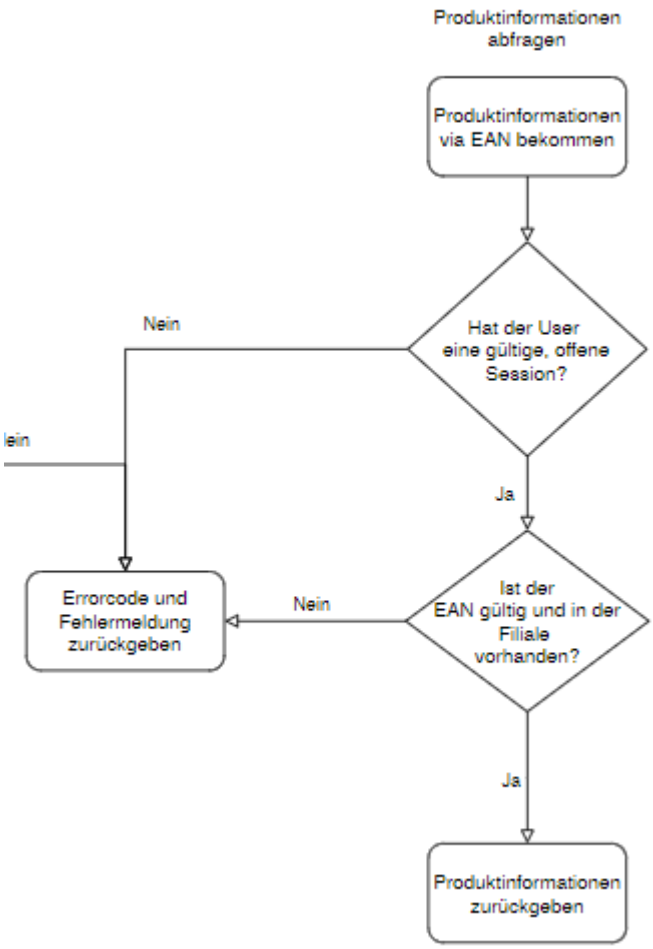
16.1.6 Abgrenzung zu anderen Prozessen

- Produkt per Kategorie hinzufügen
- Produkt per Liste hinzufügen

16.1.7 Voraussetzungen

Produkte müssen samt Barcode in der Product Tabelle gespeichert sein.

16.1.8 Grafische Prozessdarstellung



16.2 Userstory "Produkt per Kategorie hinzufügen"

16.2.1 Beschreibung

"Als Webservice ermöglichen wir es, dass der Client eine Liste von Kategorien eines spezifischen Stores bekommt. Auch ermöglichen wir das Abfragen der Details einer Kategorie."

16.2.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Keine Kategorien vorhanden	Der Request war erfolgreich, allerdings waren keine Kategorien vorhanden.	204
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request.	401
Forbidden	Kategoriename wurde in dem Shop nicht gefunden.	403
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

16.2.3 Datenbankstrukturen DB

Siehe 15.1.3

16.2.4 Lokale Funktionen

16.2.4.1 getAllCategories

Gibt alle Kategorien samt zugehörigen Produkten in einem Store zurück.

16.2.4.2 getProductsByCategory

Gibt alle Produkte einer Kategorie in einem Store zurück.

16.2.5 Akzeptanzkriterien

Die Kategorien samt Produkte von einem spezifischen Store werden zurückgegeben

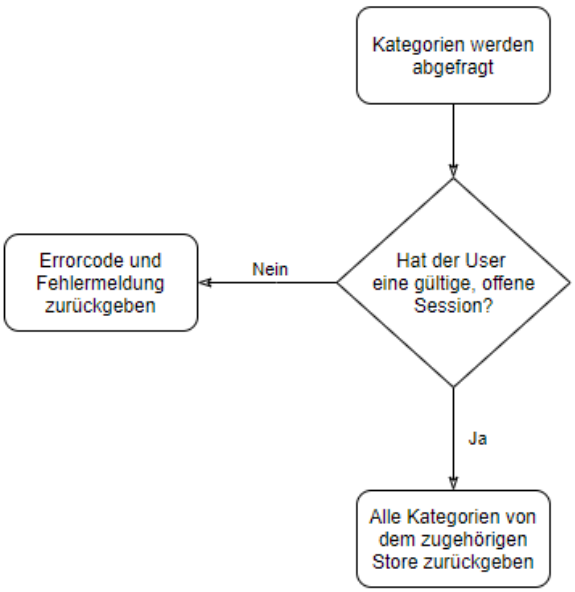
16.2.6 Abgrenzung zu anderen Prozessen

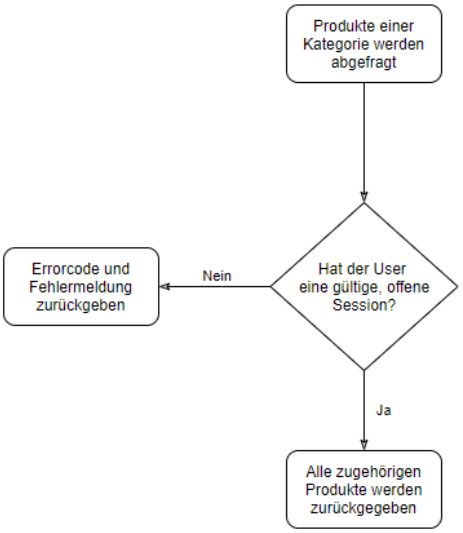
- Produkt per Liste hinzufügen
- Produkt per Scan hinzufügen

16.2.7 Voraussetzungen

Kategorien sind in der Category Tabelle mit den zugehörigen Produkten gespeichert.

16.2.8 Grafische Prozessdarstellung





16.3 Userstory "Produkt per Liste hinzufügen"

16.3.1 Beschreibung

"Als Webservice können wir anhand von einem Search-String Produkte zurückliefern, die den Suchkriterien entsprechen.“

16.3.2 Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request.	401

Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

16.3.3 Datenbankstrukturen DB

Siehe 15.1.3

16.3.4 Lokale Funktionen

findProductByQueryString
Sucht Produkt anhand von einem Search String. Dieser wird gesplittet (z.B. nach Spaces) und mit einem Suchalgorithmus ausgewertet.

16.3.5 Akzeptanzkriterien

Produkte eines Shops werden anhand eines Search-Strings korrekt nach Suchkriterien zurückgeliefert

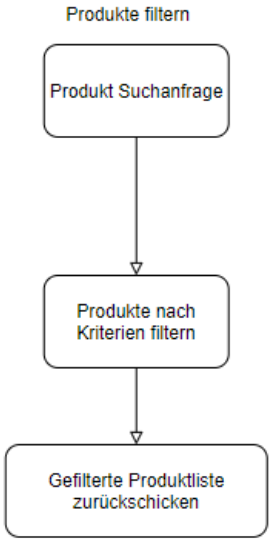
16.3.6 Abgrenzung zu anderen Prozessen

- Produkt per Kategorie hinzufügen
- Produkt per Scan hinzufügen

16.3.7 Voraussetzungen

Bestehende Produkte; Bestehender Shop; Bestehender Customer

16.3.8 Grafische Prozessdarstellung



16.4 Userstory „Produkt hinzufügen/Stückanzahl“

Verwendet die findProductByBarcode Methode, die in 4.1 definiert wurde.

16.5 Userstory „Produktdetails nach Scan“

Verwendet die findProductByBarcode Methode, die in 4.1 definiert wurde.

16.6 Userstory „Warenkorb ansehen“

Verwendet die findProductByBarcode Methode, die in 4.1 definiert wurde.

16.7 Userstory „Produkt im Warenkorb ansehen“

Verwendet die findProductByBarcode Methode, die in 4.1 definiert wurde.

16.8 Userstory „Produkt im Warenkorb entfernen“

Keine Backend Implementierung notwendig.

16.9 Userstory „Warenkorb Produktstückzahl erhöhen/verringern“

Keine Backend Implementierung notwendig.

16.10 Userstory „Produktdetails (Warenkorb)

Verwendet die findProductByBarcode Methode, die in 4.1 definiert wurde.

16.11 Userstory „Warenkorb löschen“

Keine Backend Implementierung notwendig

16.12 Userstory „Warenkorb bearbeiten“

Keine Backend Implementierung notwendig

16.13 Userstory „Biometrische Daten“

Keine Backend Implementierung notwendig

17 Epic „Start Page“

17.1 Userstory „Home“

Keine Backend Implementierung notwendig.

18 Epic „Zahlung“

18.1 Userstory "Zur Kasse"

18.1.1 Beschreibung

"Als Webservice wird die Intention des Benutzers angenommen und anhand die Zahlungshöhe, Währung und Handler Key einen PaymentIntent Objekt generiert und an FE zurück geliefert.

Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200
Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request	401
Forbidden	Falscher Barcode	403
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

18.1.2 Datenbankstrukturen DB

Tabelle: StripeAccounts					
Feldname	Datentyp	PK	FK	Nullable	Default
ID	**INT**	**True**	**False**	**False**	**AutoINC**
CustomerID	VARCHAR(200)	False	True	False	
PublishableKey	VARCHAR(100)	False	False	False	
SecretKey	VARCHAR(100)	False	False	False	
ValidFrom	DateTime	False	False	False	Current_TS
ValidThru	DateTime	False	False	False	

18.1.3 Lokale Funktionen

"Sobald ein Benutzer mit einer Zahlung anfangen will, wird der Webservice „InitiatePaymentInit“ mit zwei Parameters aufgerufen (Amount, Currency). Anhand dem mitgeschickten ShoppingSessionToken wird der Handler gemappt und desen SessionID wird gefunden, nachher wird das Endopint das Backend https://api.stripe.com/v1/payment_intents aufgerufen um ein PaymentIntent Objekt zu bekommen.

18.1.4 Externe Schnittstellen

18.1.4.1 "Create" Endpoint von Stripe: https://api.stripe.com/v1/payment_intents

Body:

```
{
Siehe Dokumentation der Stripe API
}
```

Response:

```
{
Siehe Dokumentation der Stripe API
}
```

18.1.5 Akzeptanzkriterien

PaymentIntent Objekt mit id wird generiert und an FE geliefert

18.1.6 Abgrenzung zu anderen Prozessen

Zahlungsprozess

18.1.7 Voraussetzungen

Publishable Key vom Handler muss in DB vorhanden sein

18.1.8 Grafische Prozessdarstellung

Siehe 11.1.4

18.2 Userstory "Bezahlen"

18.2.1 Beschreibung

Wenn der Benutzer behauptet dass er eine Zahlung erfolgreich durchgeführt hat, wird sich beim FE melden um die Zahlung mit dem genannten ID bei Stripe zu verifizieren.

Schnittstellen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	Die Daten sind vollständig und valid.	200

Internal Server Error	Der Server hat einen internen Fehler.	500
Invalid	Invalide (JSON) Daten.	400
Unautorisiert	Unautorisierter Request	401
Forbidden	Falscher Barcode	403
Payload too large	Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.	413
Gesperrt	Bestimmte Resources sind gesperrt.	423

18.2.2 Datenbankstrukturen DB

Siehe 18.1.2

18.2.3 Lokale Funktionen

"Sobald ein Benutzer behauptet die Zahlung durchgeführt zu haben, wird der Webservice „VerifyPayment“ mit zwei Parameters aufgerufen (id, Publishable Key). Unser Backend wird das Endopint https://api.stripe.com/v1/payment_intents/**id*** aufrufen um den Zahlungs Status abzufragen.

18.2.4 Externe Schnittstellen

*18.2.4.1 "Retrieve" Endpoint von Stripe: https://api.stripe.com/v1/payment_intents/id**

Body:

```
{
Siehe Dokumentation der Stripe API
}
```

Response:

```
{
Siehe Dokumentation der Stripe API
}
```

18.2.5 Akzeptanzkriterien

Zahlungsstatus wird bei Stripe abgefragt und der Benutzer bekommt die Zahlung verifiziert.

18.2.6 Abgrenzung zu anderen Prozessen

18.2.7 Voraussetzungen

Publishable Key vom Handler muss in DB vorhanden sein

18.2.8 Grafische Prozessdarstellung

Siehe 11.1.4

Userstory “Zahlen”

- **Beschreibung**

Als Webservice ermöglichen wir es Nutzern, welche bereits einen Einkauf durchgeführt haben und diesen nun zahlen wollen, diesen über eine ausgewählte Zahlungsart zu zahlen und somit den Shop physisch verlassen können. Die Abwicklung erfolgt über einen simulierten Zahlungsanbieter.

- **Schnittstellen**

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	**Die Daten sind vollständig und valid, alle Produkte sind im Store vorhanden.**	**200**
Internal Server Error	**Der Server hat einen internen Fehler.**	**500**
Invalid	**Invalide (JSON) Daten. Die übergebenen Daten stimmen nicht mit den Datenanforderungen überein.**	**400**
Unauthorized	**Der ShopSessionToken oder der SessionToken ist ungültig.**	**401**
Konflikt	**Ein oder mehrere Produkte wurden nicht in dem Shop gefunden**	**409**
Payload too large	**Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.**	**413**
Payment failed	**Die Zahlung ist fehlgeschlagen**	**430**

- **Request**

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0
ShopSessionToken: 62fe31252-5512-1234-fef6-f44f2c2512515

```
{
    "requestDate": "2020-09-07T13:10:59",
    "stripeToken": "token",
    "products": [
        {
            "articleCode": "5192851255151515",
            "quantity": 1
        },
        {
            "articleCode": "513593513895889",
            "quantity": 10
        },
        {
```

```
            "articleCode": "121241242141244124",
            "quantity": 1
        }
    ],
    "paymentType": "PayPal",
      “language”: “EN” //Possible: EN, DE, FR
}
```

- **Response**

```
{
    "responseDate": "2020-09-07T13:10:59",
    "receiptID": "1552012509512",
    "success": true
}
```

- **Akzeptanzkriterien**

- Nutzer kann eine Zahlung durchführen
- Gesamtpreis aller Produkte wird berechnet (Quantität, MwSt.)
- Rechnungs PDF wird zurückgegeben

- **Abgrenzung**
  - Rechnungserstellung

Userstory “App/Rechnung/PDF + Rechnung erstellen”

- **Beschreibung**

Als Webservice erstellen wir nach durchgeführter Zahlung eine Rechnung mit allen gesetzlich verbindlichen Bestandteilen. Das PDF zu dieser Rechnung kann über einen Endpunkt mit zugehöriger Rechnungsnummer abgerufen werden. Auf Fiskalisierung auf der Rechnung wird aufgrund von mangelndem Wissen verzichtet und nur ein Beispiel davon verwendet.

- **Datenbank (ReceiptID = STRING!)**

Tabelle: Receipt

Feldname	Datentyp	PK	FK	Nullable	Default
ReceiptID	**INT**	**true**	**false**	**false**	**AUTOINC**
CustomerID	INT	false	true	false	
ShopID	INT	false	true	false	
MerchantID	INT	false	true	false	
CreationTS	TIMESTAMP	false	false	false	
BrutPrice	DECIMAL(10,2)	false	false	false	
NetPrice	DECIMAL(10,2)	false	false	false	
Filename	VARCHAR(1024)	false	false	false	

- **Schnittstellen**

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	**Die Daten sind vollständig und valid, der Customer hat Zugriff auf die Rechnung.**	**200**
Internal Server Error	**Der Server hat einen internen Fehler.**	**500**
Invalid	**Invalide (JSON) Daten. Die übergebenen Daten stimmen nicht mit den Datenanforderungen überein.**	**400**
Unauthorized	**Der SessionToken ist ungültig.**	**401**
Forbidden	**Der Customer hat keinen Zugriff auf die Rechnung.**	**403**
Payload too large	**Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.**	**413**

- **Request**

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

```
{
    "requestDate": "2020-09-07T13:10:59",
    "receiptId": "1552012509512"
}
```

- **Response**

```
[PDF-FILE content]
```

- **Akzeptanzkriterien**
  - Rechnung wird gesetzlich korrekt als PDF erstellt (ohne echter Fiskalisierung)
  - Rechnung kann über einen Endpunkt heruntergeladen werden

- **Abgrenzung**
  - Zahlen

US “Alte Rechnungen ansehen” - Rechnungshistory

- **Beschreibung**

Als Webservice erstellen ermöglichen wir es den Kunden, dass sie die bisher getätigten Rechnungen anschauen können.

- **Schnittstellen**

Fall	Beschreibung	Statuscode

Daten vollständig (OK)	**Die Daten sind vollständig und valid, der Customer hat Zugriff auf die Rechnung.**	**200**
Internal Server Error	**Der Server hat einen internen Fehler.**	**500**
Invalid	**Invalide (JSON) Daten. Die übergebenen Daten stimmen nicht mit den Datenanforderungen überein.**	**400**
Unauthorized	**Der SessionToken ist ungültig.**	**401**
Payload too large	**Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.**	**413**

- **Request**

Header:SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

Request:
{Endpunkt}?size=3&page=0

- Response:


```
{
    "responseDate": "2021-01-25T20:40:08.272",
    "success": true,
    "receipts": [
        {
            "receiptId": "f16ba1bd-b78d-4f8b-a5da-36ba6b1213a6",
            "date": "2021-01-25T20:22:19.731",
            "shop": {
                "merchantId": 4444,
                "name": "Billa",
                "phone": "+43123123123",
                "vatId": "AT231212441",
                "email": "info@billa.at",
                "fax": "+43123123123",
                "openingHour": null,
                "closingHour": null,
                "street": "Billastraße",
                "houseNumber": "12",
                "zipCode": "1010",
                "country": "Austria",
                "city": "Vienna",
                "longitude": 51.4124124,
                "latitude": -12.42112
            },
            "merchant": {
                "name": "Billa",
                "uidNr": "Test123",
                "phone": "+43123123123",
                "email": "billa@gmail.com",
                "fax": "+431231231232",
```

```
                "logo": "https://assets.billa.at/v1/assets/6197966",
                "commercialRegisterCourt": "Testcourt",
                "commercialRegisterNumber": "Testcourt",
                "vatId": "Test123",
                "privateStripeKey": "",
                "styleId": 1
            },
            "netPrice": 49660.0,
            "brutPrice": 59592.0
        },
        {
            "receiptId": "b9fa0ae7-a6b8-4eec-bfbc-b26379e54d8a",
            "date": "2021-01-25T20:23:35.734",
            "shop": {
                "merchantId": 4444,
                "name": "Billa",
                "phone": "+43123123123",
                "vatId": "AT231212441",
                "email": "info@billa.at",
                "fax": "+43123123123",
                "openingHour": null,
                "closingHour": null,
                "street": "Billastraße",
                "houseNumber": "12",
                "zipCode": "1010",
                "country": "Austria",
                "city": "Vienna",
                "longitude": 51.4124124,
                "latitude": -12.42112
            },
            "merchant": {
                "name": "Billa",
                "uidNr": "Test123",
                "phone": "+43123123123",
                "email": "billa@gmail.com",
                "fax": "+431231231232",
                "logo": "https://assets.billa.at/v1/assets/6197966",
                "commercialRegisterCourt": "Testcourt",
                "commercialRegisterNumber": "Testcourt",
                "vatId": "Test123",
                "privateStripeKey": "",
                "styleId": 1
            },
            "netPrice": 3000.0,
            "brutPrice": 3600.0
        }
    ],
    "total": 2
}
```

- **Akzeptanzkriterien**
  - Alle Rechnungen können dem Kunden korrekt zugeordnet werden
  - Rechnungen werden richtig zurückgegeben
  - Pagination funktioniert korrekt

- **Abgrenzung**
  - Zahlen

US “Registrierung” - Daten ändern

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	**Die Daten sind vollständig und valid, der Customer hat Zugriff auf die Rechnung.**	**200**
Internal Server Error	**Der Server hat einen internen Fehler.**	**500**
Invalid	**Invalide (JSON) Daten. Die übergebenen Daten stimmen nicht mit den Datenanforderungen überein.**	**400**
Unauthorized	**Der SessionToken ist ungültig.**	**401**
Payload too large	**Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.**	**413**

- **Request**

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

```
{
      "requestDate": "2020-09-07T13:10:59",
      "gender": "M",
      "firstname": "Max",
      "lastname": "Mustermann",
      "address": "Straße 1, 1020 Wien",
      "birthdate": "2020-09-07",
      "phone": "068120160041",
      "vatid": "ATU 57615634"
}
```

- **Response**

```
{
	"success": true,
	"responseDate": "2020-09-07T13:10:59"
}
```

US “Registrierung” - Passwort ändern

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	**Die Daten sind vollständig und valid, der Customer hat Zugriff auf die Rechnung.**	**200**
Internal Server Error	**Der Server hat einen internen Fehler.**	**500**
Invalid	**Invalide (JSON) Daten. Die übergebenen Daten stimmen nicht mit den Datenanforderungen überein.**	**400**
Unauthorized	**Der Session Token ist ungültig.**	**401**
Forbidden	**Das alte Passwort ist ungültig.**	**403**
Payload too large	**Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.**	**413**

- **Request**

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

```
{
	"requestDate": "2020-09-07T13:10:59",
	"oldPassword": "P@ssw0rd",
	"newPassword": "newP@ssw0rd"
}
```

- **Response**

```
{
	"success": true,
	"responseDate": "2020-09-07T13:10:59"
}
```

US “Registrierung” - E-Mail-Adresse ändern

- **Ablaufbeschreibung**

E-Mail Change Request → neuer Eintrag in VerifyEmail Tabelle mit Verificationcode, UserID und E-Mail-Adresse | Verifizierung: Wenn der Code stimmt, wird die E-Mail-Adresse aus der Tabelle in die Customertabelle übernommen

Fall	Beschreibung	Statuscode
Daten vollständig (OK)	**Die Daten sind vollständig und valid, der Customer hat Zugriff auf die Rechnung.**	**200**
Internal Server Error	**Der Server hat einen internen Fehler.**	**500**
Invalid	**Invalide (JSON) Daten. Die übergebenen Daten stimmen nicht mit den Datenanforderungen überein.**	**400**
Unauthorized	**Der SessionToken ist ungültig.**	**401**
Konflikt	**Die E-Mail-Adresse ist bereits in Verwendung.**	**409**
Payload too large	**Das JSON/Feld ist zu lange. Zu viele Zeichen gesendet.**	**413**

- **Request**

Header:
SessionToken: 49fd3176-570d-4238-aed6-f44f2c2533c0

```
{
	"requestDate": "2020-09-07T13:10:59",
	"newEmail": "newEmail@localhost"
}
```

- **Response**

```
{
	"success": true,
	"responseDate": "2020-09-07T13:10:59",
	"email": "newEmail@localhost",
	"countdownTime": 30
}
```

Appendix I: Statusberichte

Scan Buy Go

Statusbericht 21.01.2021

Alexander Wiener
Florian Freimüller
Paul Maurovich
Lucas Walter
Dennis Sima

Inhaltsverzeichnis

1 Programmierung + Know How 3

2 Meilensteinplan 4

3 Projektstatus-Linie 5

4 Earned Value Analyse 6

5 Risiko Analyse 7

6 Zusatz 8

Tabellenverzeichnis

Tabelle 1: Meilensteinplan 4
Tabelle 2: Risikoanalyse 7

Abbildungsverzeichnis

Abbildung 1: Statuslinie 5
Abbildung 2: Earned Value Analyse 6

1 Programmierung + Know How

Derzeit Frontend:

- Knowhow in Dart/Flutter angeeignet
- Programmierphase 1 abgeschlossen (>50% der User Stories)
- Tests am Handy wurden durchgeführt

Derzeit Backend:

- Gelerntes im Bereich Microservices konnte erfolgreich im Projekt umgesetzt werden
- Backendstruktur wurde auf Azure aufgebaut (Server, Datenbank)
- Programmierphase 1 wurde abgeschlossen (>50% der User Stories)

Was ist seit dem letzten Statusbericht passiert?

- Die Spezifikation wurde abgeschlossen
- Die User Stories wurden in Azure-Boards erstellt
- Meetingtermine wurden festgelegt (Daily Standup, Sprint Planning)
- Die Programmierung wurde begonnen
- Die Programmierphase 1 wurde erfolgreich abgeschlossen
- Die Backendinfrastruktur läuft
- Bugs wurden in Azure-Boards erfasst

Was wird bis zum nächsten Statusbericht passieren?

- Die Programmierphase 2 wird abgeschlossen sein
- Die Diplomarbeit wird zum Großteil fertig sein
- Die App wird getestet

2 Meilensteinplan

PROJEKT-MEILENSTEINPLAN			
Meilenstein	**Basis-termine**	**neue Plantermine**	**Ist Termine**
Lastenheft übergeben	**15.09.2020**		**15.09.2020**
Vorstudien abgeschlossen	**30.09.2020**		**30.09.2020**
Spezifikation abgenommen	**23.10.2020**		**23.10.2020**
Programmierphase 1 abgeschlossen	**01.12.2020**		**01.12.2020**
Programmierphase 2 abgeschlossen	**04.02.2020**		**-**
Prototyp fertiggestellt	**01.03.2021**		**-**
Projektdokumentation abgeschlossen	**15.03.2020**		**-**
Projekt durch Schule abgenommen	**31.03.2020**		**-**

Tabelle 1: Meilensteinplan

3 Projektstatus-Linie

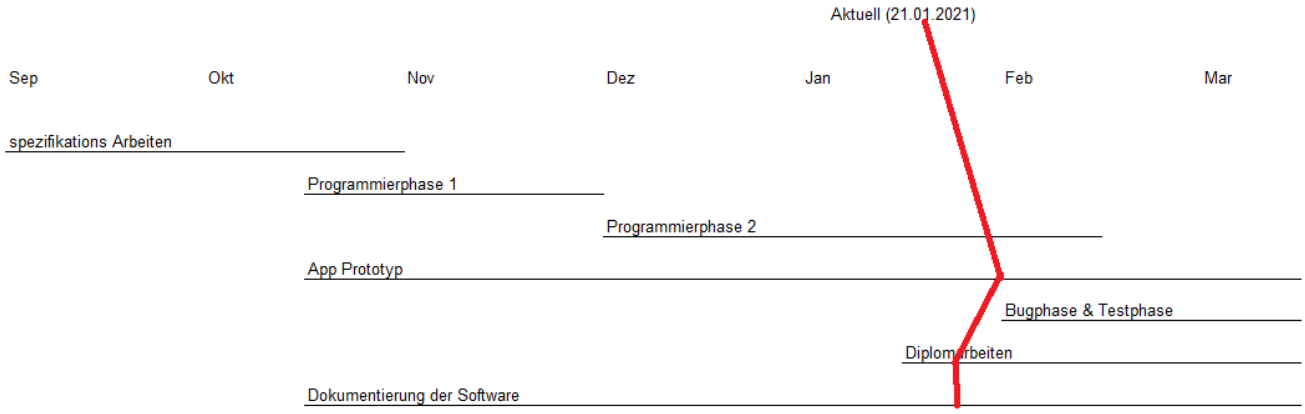


Abbildung 1: Statuslinie

Anmerkungen:

- Diplomarbeiten: die individuellen Themenstellungen
- Bugphase & Testphase: natürlich wird laufend getestet und ebenso Bugs erfasst, am Ende wird aber nochmals alles genau geprüft und auftretende Bugs behoben
- Spezifikationsarbeiten: Lastenheft, Spezifikation, User Stories und Einarbeitsungszeit für neue Technologien

4 Earned Value Analyse

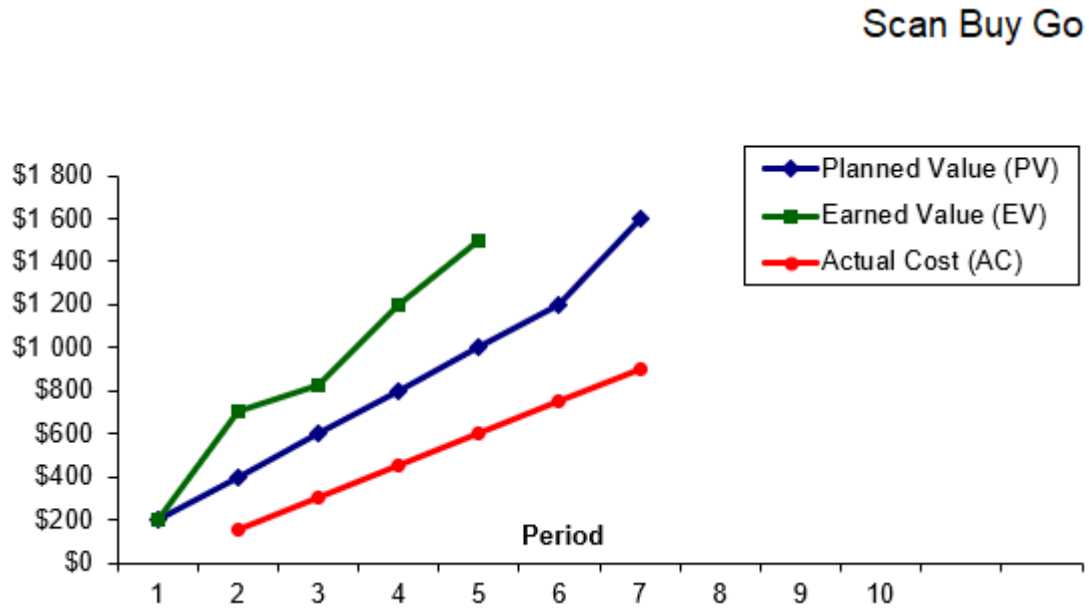


Abbildung 2: Earned Value Analyse

Anmerkungen:

- Kosten: 150$ pro Monat für Backend-Struktur
- Period: Jeweils 1 Monat beginnend mit September 2020
- EV: Starker Anstieg aufgrund des Streites mit dem Partner (hier konnten wertvolle Erfahrungen gesammelt werden)
- In der nächsten Periode wird ein starker Anstieg in der EV kommen, da das Projekt jetzt zum Teil auch noch für ein anderes Fach verwendet wird

5 Risiko Analyse

ID	Risikobezeichnung (Ereignis)	Risiko-klasse	Mögliche Risikoursache	EW %	Schadenshöhe (€)	Risiko-wert (€)
R1	Personalausfall	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€

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

Tabelle 2: Risikoanalyse

6 Zusatz

Anmerkungen der Projektleitung:

Seitens der Projektleitung gibt es derzeit keine groben Mängel im Bezug auf das Projektteam sowie die Projektarbeit. Fristen werden eingehalten, die Kommunikation funktioniert. Jeder leistet seinen Beitrag und wir sind durch die enorme Anstrengung sehr weit vorne und konnten es uns dadurch in der schwierigen Schulphase im Dezember leisten, das Projekt ein wenig hintenanzustellen und uns auf die Schularbeiten und Tests sowie Hausaufgabe zu konzentrieren, ohne dass der Projekterfolg dadurch gefährdet wird.

Externe Projekt-Nutzung

Die Projektmitglieder Lucas Walter und Alexander Wiener sind derzeit dabei, das Projekt auch für OPS zum Einsatz zu bringen. In dem Schulfach wird unser Backend herangezogen, um ein Lasttest-Projekt durchzuführen. Hier sind wir der vollen Überzeugung, dass dies einen großen Mehrwert für das Team bringt, da hier auch eine Skalierung der Ressourcen für Apps stattfindet. Das bedeutet, dass wir lernen werden, welche Infrastruktur man für eine gewisse Bandbreite sowie Anzahl an Usern benötigt.

Aktueller Stand der App

Wir würden den aktuellen Stand der App dann bei einer Statuspräsentation vorführen.

Scan Buy Go

Statusbericht 16.03.2021

Alexander Wiener
Florian Freimüller
Paul Maurovich
Lucas Walter
Dennis Sima

Inhaltsverzeichnis

1 Programmierung + Know How **4**

2 Meilensteinplan **5**

3 Projektstatus-Linie **6**

4 Earned Value Analyse **7**

5 Risiko Analyse **8**

6 Zusatz ..... Fehler! Textmarke nicht definiert.

Tabellenverzeichnis

Tabelle 1: Meilensteinplan 5
Tabelle 2: Risikoanalyse 8

Abbildungsverzeichnis

Abbildung 1: Statuslinie 6
Abbildung 2: Earned Value Analyse 7

1 Programmierung + Know How

Derzeit Frontend:

- Programmierphase 2 ist abgeschlossen
- Tests auf den Smartphones wurden durchgeführt

Derzeit Backend:

- Programmierphase 2 ist abgeschlossen

Was ist seit dem letzten Statusbericht passiert?

- Frontend Programmierphase 2 wurde abgeschlossen

Was passiert noch?

- Abschluss der individuellen Themenstellungen
- Projektabnahme durch Betreuer

2 Meilensteinplan

PROJEKT-MEILENSTEINPLAN			
Meilenstein	**Basis-termine**	**neue Plantermine**	**Ist Termine**
Lastenheft übergeben	**15.09.2020**		**15.09.2020**
Vorstudien abgeschlossen	**30.09.2020**		**30.09.2020**
Spezifikation abgenommen	**23.10.2020**		**23.10.2020**
Programmierphase 1 abgeschlossen	**01.12.2020**		**01.12.2020**
Programmierphase 2 abgeschlossen	**04.03.2021**		**04.03.2021**
Prototyp fertiggestellt	**25.03.2021**		**-**
Projektdokumentation abgeschlossen	**29.03.2021**		**-**
Projekt durch Schule abgenommen	**31.03.2021**		**-**

Tabelle 1: Meilensteinplan

3 Projektstatus-Linie

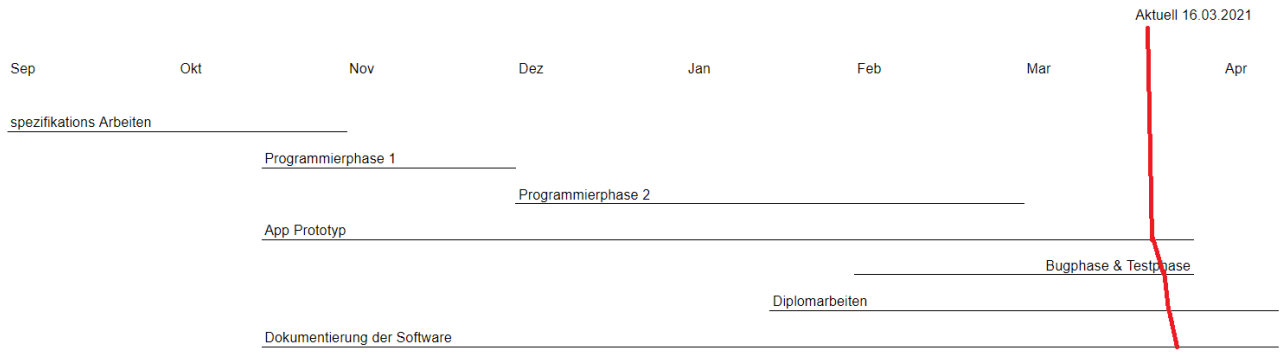


Abbildung 1: Statuslinie

Anmerkungen:

- Diplomarbeiten: die individuellen Themenstellungen
- Bugphase & Testphase: natürlich wird laufend getestet und ebenso Bugs erfasst, am Ende wird aber nochmals alles genau geprüft und auftretende Bugs behoben
- Spezifikationsarbeiten: Lastenheft, Spezifikation, User Stories und Einarbeitsungszeit für neue Technologien
- Die Diplomarbeiten werden noch fertiggestellt

4 Earned Value Analyse

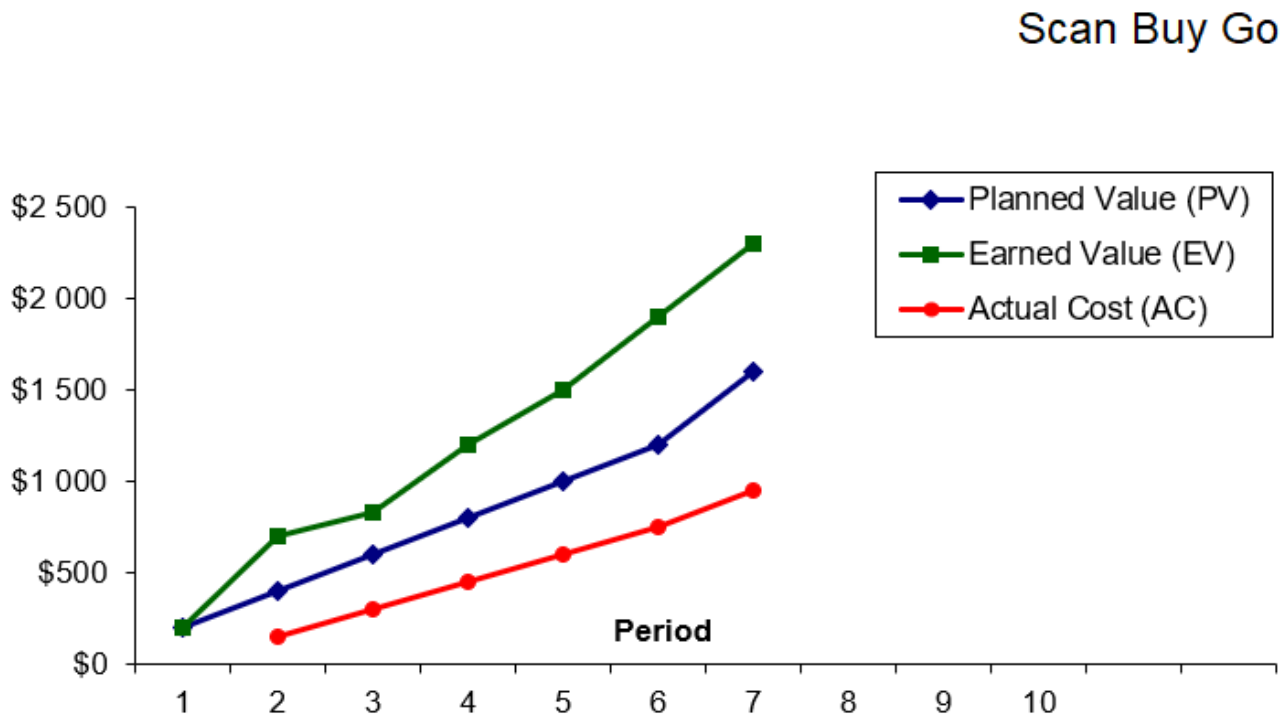


Abbildung 2: Earned Value Analyse

Anmerkungen:

- Period: Jeweils 1 Monat beginnend mit September 2020
- In der Periode 2 gibt es einen starken Anstieg in der EV aufgrund des Streites mit dem Partner (hier konnten wertvolle Erfahrungen gesammelt werden)
- In der Periode 6 gibt es einen starken Anstieg in der EV, da das Projekt jetzt zum Teil auch noch für ein anderes Fach verwendet wird

5 Risiko Analyse

ID	Risikobezeichnung (Ereignis)	Risiko-klasse	Mögliche Risikoursache	EW %	Schadenshöhe (€)	Risiko-wert (€)
R1	Personalausfall	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€

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

Tabelle 2: Risikoanalyse

6 Abschluss

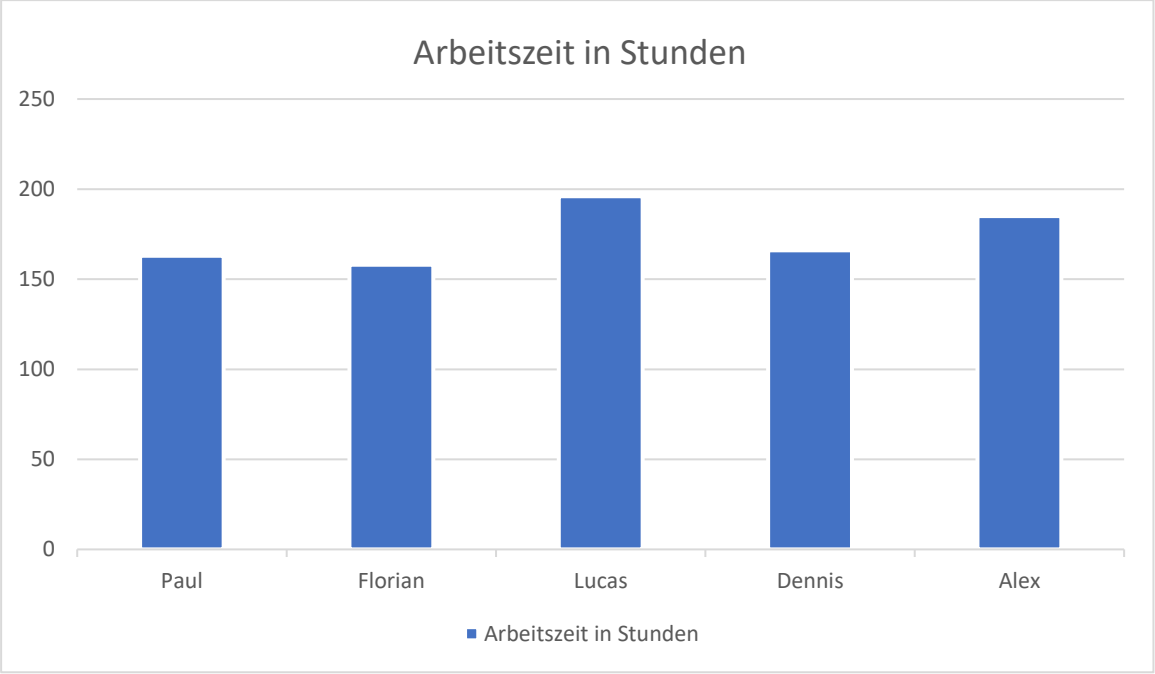
Abschluss des Projektes

Derzeit befindet sich das Projekt in der finalen Phase und wird bald planmäßig abgenommen.

Der Betreuungslehrer hat sich das Projekt bereits laufend immer wieder angesehen und den Fortschritt verfolgt. Ebenso werden wir Ende März / Anfang April die individuellen Themenstellungen erarbeiten und Feedback einholen, bevor wir diese einreichen.

Die Einreichung selbst ist am 7.4.2021 geplant und wir zusammen als Team durchgeführt.

7 Zeit


Arbeitszeit in Stunden
250
200
150
100
50
0
Paul
Florian
Lucas
Dennis
Alex
Arbeitszeit in Stunden

Appendix J: Change Request

ScanBuyGo

Change Request

Alexander Wiener

Florian Freimüller

Paul Maurovich

Lucas Walter

Dennis Sima

Change Request	
Project: ScanBuyGo	**Date:** 01.12.2020
Change Requestor: Alexander Wiener	**Change No:** 1

Change Category (Check all that apply):

☐ Schedule ☐ Cost ☐ Scope ☐ Requirements/Deliverables

☐ Testing/Quality ☐ Resources X Partner

Does this Change Affect (Check all that apply):

☐ Corrective Action ☐ Preventative Action ☐ Defect Repair ☐ Updates

☐ Other

Describe the Change Being Requested:

Der Kooperationspartner soll von Syncore EDV-Systems auf die HTL Spengergasse geändert werden.
Der Name des Projektes soll von NoQuePOS auf ScanBuyGo geändert werden.

Describe the Reason for the Change:

Nach einem Streit mit dem Partnerunternehmen Syncore EDV-Systems, haben wir uns einvernehmlich getrennt und die Schule als neuen Partner gewählt.
Aufgrund der Streitigkeiten, war ein Fortführen der Arbeit mit dem Projektpartner nicht mehr möglich.

Da der Projektname vom Projektpartner vorgeschlagen wurde, müssen wir diesen nun ändern, da wir diesen nicht weiterverwenden dürfen.

Describe all Alternatives Considered:

Es wurde lange mit dem Partner verhandelt und diskutiert, doch es konnte keine Einigung, die für beide Seiten gepasst hat, erzielt werden.

Describe any Technical Changes Required to Implement this Change:

In der Applikation und in den Dokumenten muss „NoQuePOS“ durch „ScanBuyGo“ ersetzt werden

Describe Risks to be Considered for this Change:

-

Estimate Resources and Costs Needed to Implement this Change:

5 Stunden Arbeit für neues Branding und das umändern des Namens in der Applikation und in den Dokumenten

Scan Buy Go

Describe the Implications to Quality: -
Disposition: X Approve ☐ Reject ☐ Defer
Justification of Approval, Rejection, or Deferral: Approved

Change Board Approval:		
Name	**Signature**	**Date**
Alexander Wiener		01.12.2020
Paul Maurovich		01.12.2020
Dennis Sima		01.12.2020
Florian Freimüller		01.12.2020
Lucas Walter		01.12.2020

Scan Buy Go