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

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)

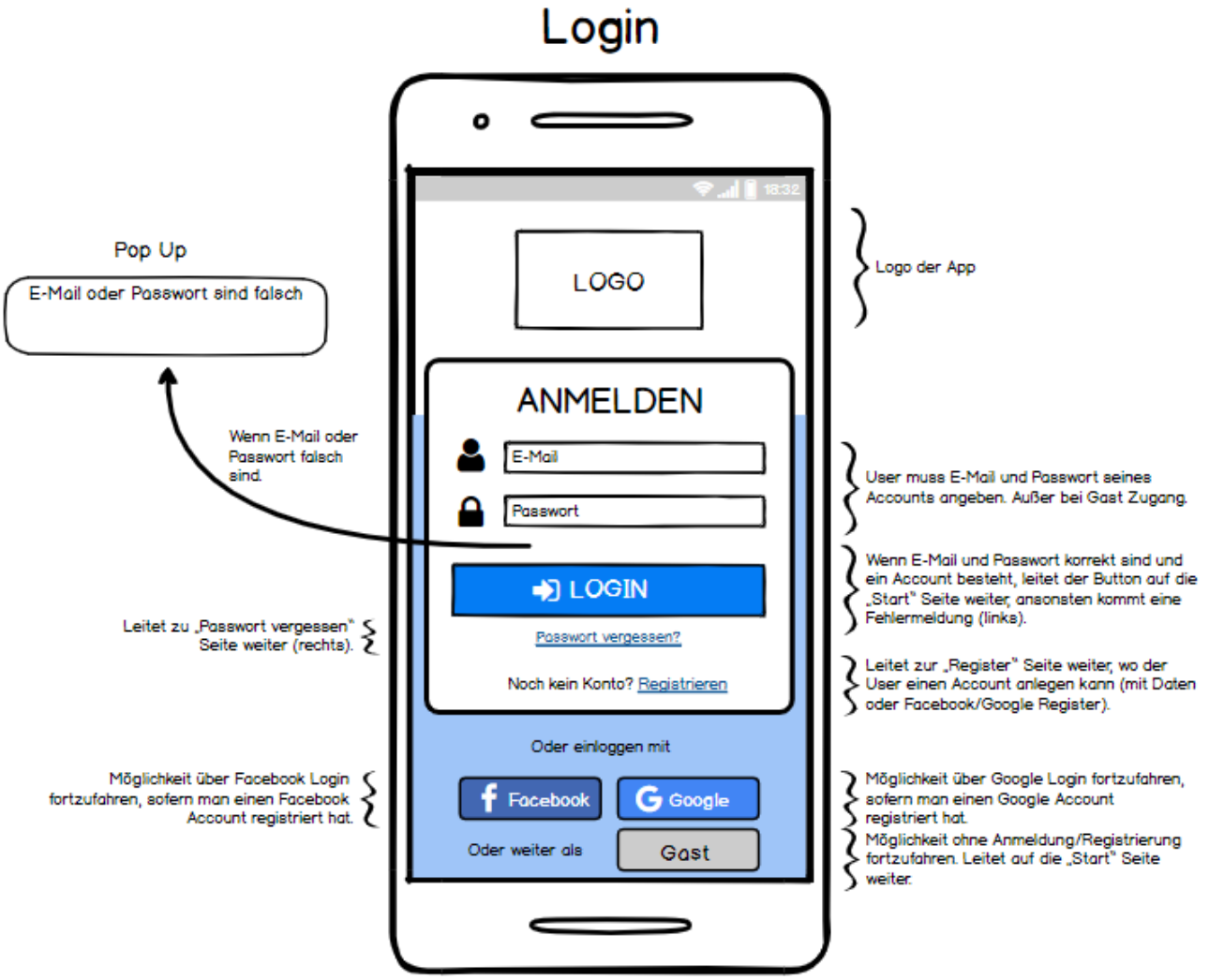
6 Epic "Login"

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



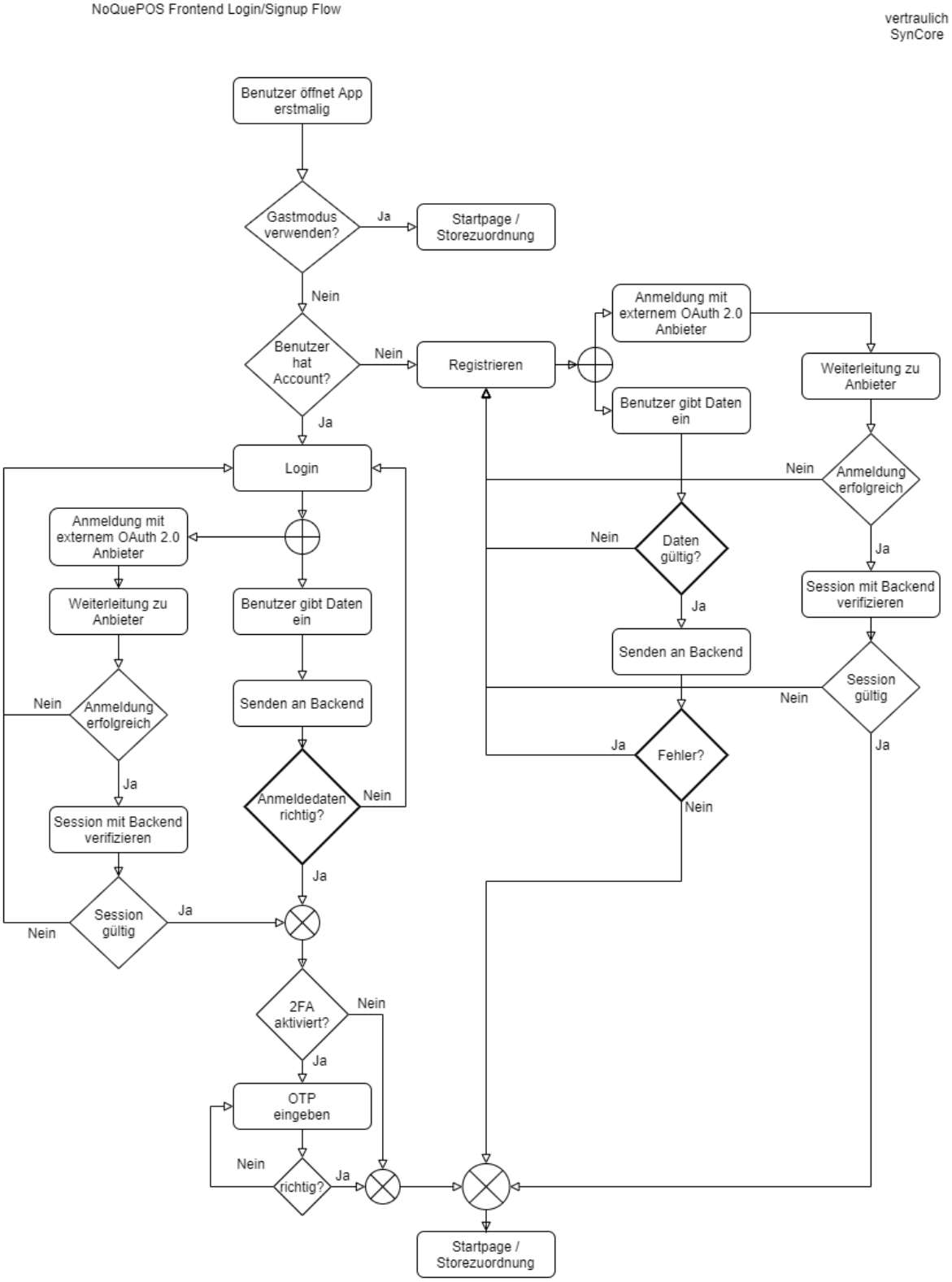
6.1.3 Funktionen der Masken

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

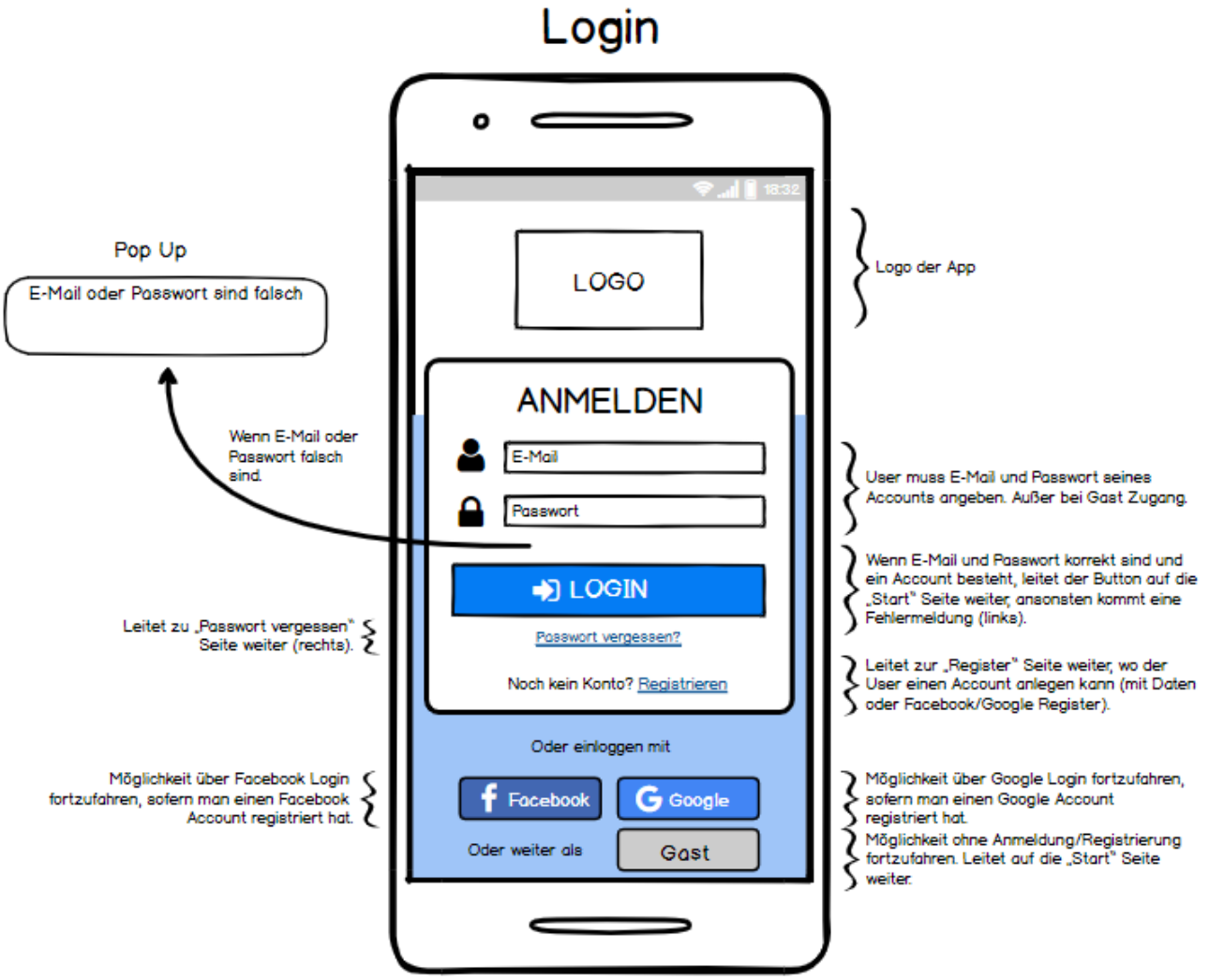
Registrierung

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



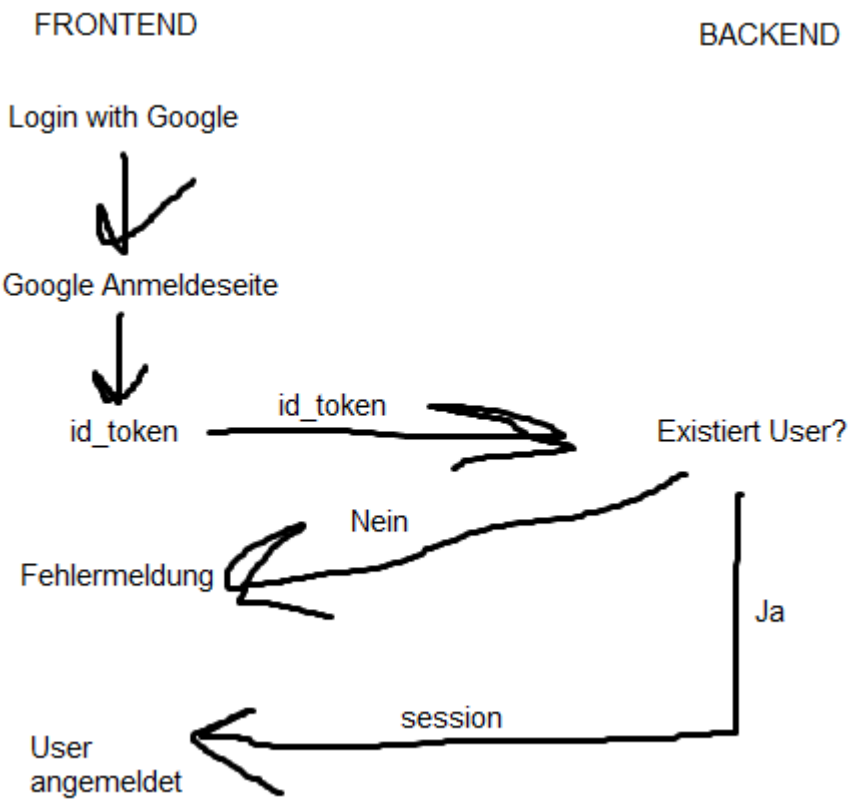
6.2.3 Funktionen der Masken

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"

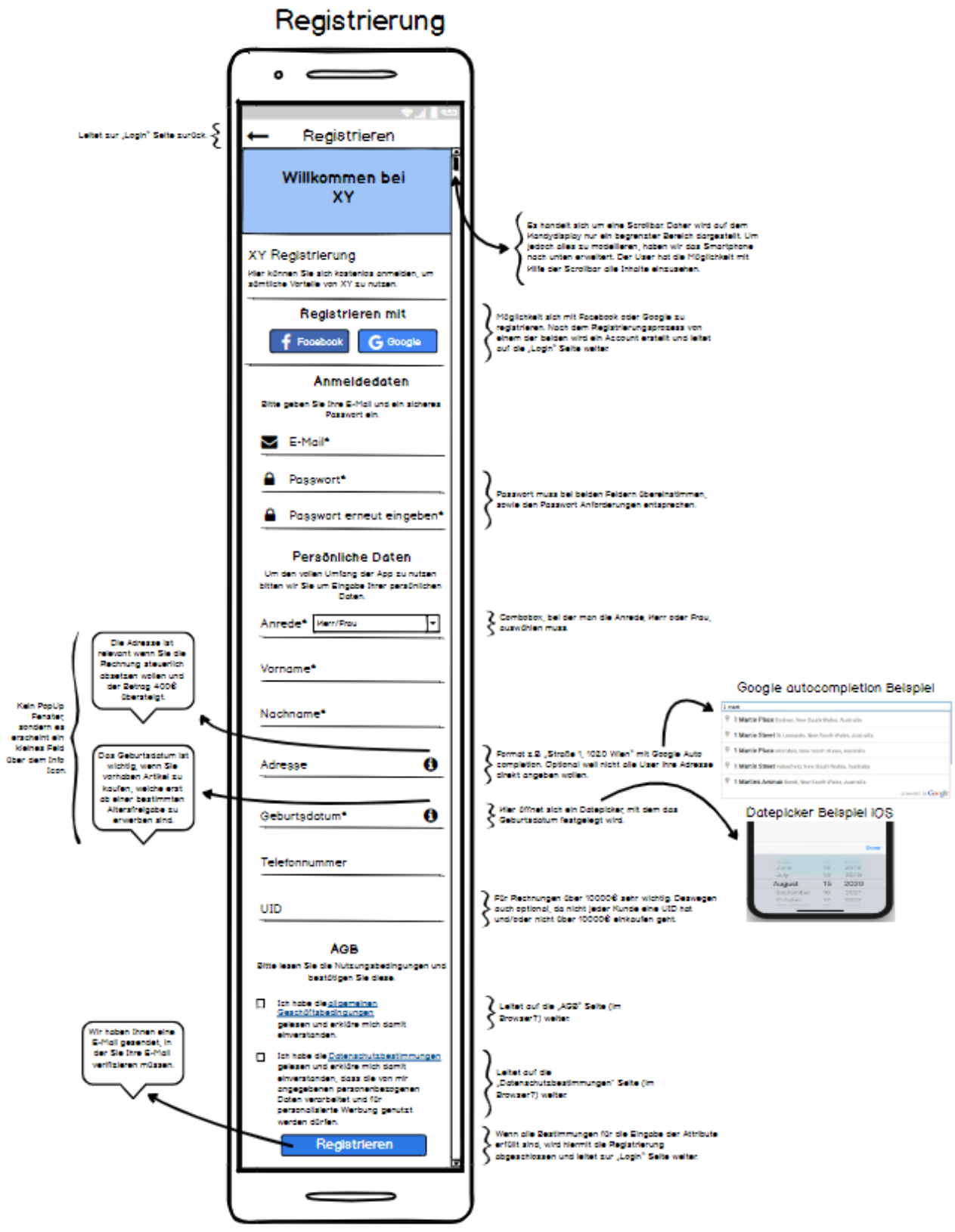
7.1 Userstory "Registrierung"

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



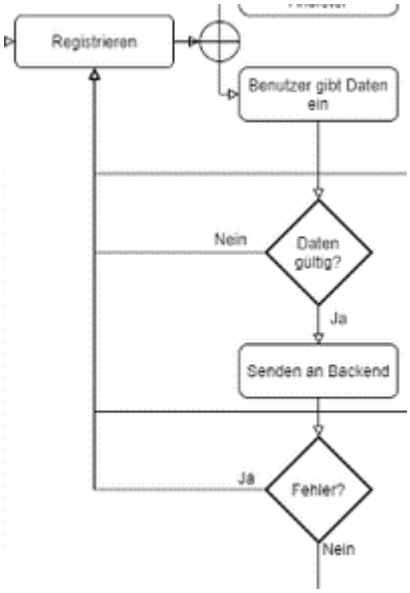
7.1.3 Funktionen der Masken

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

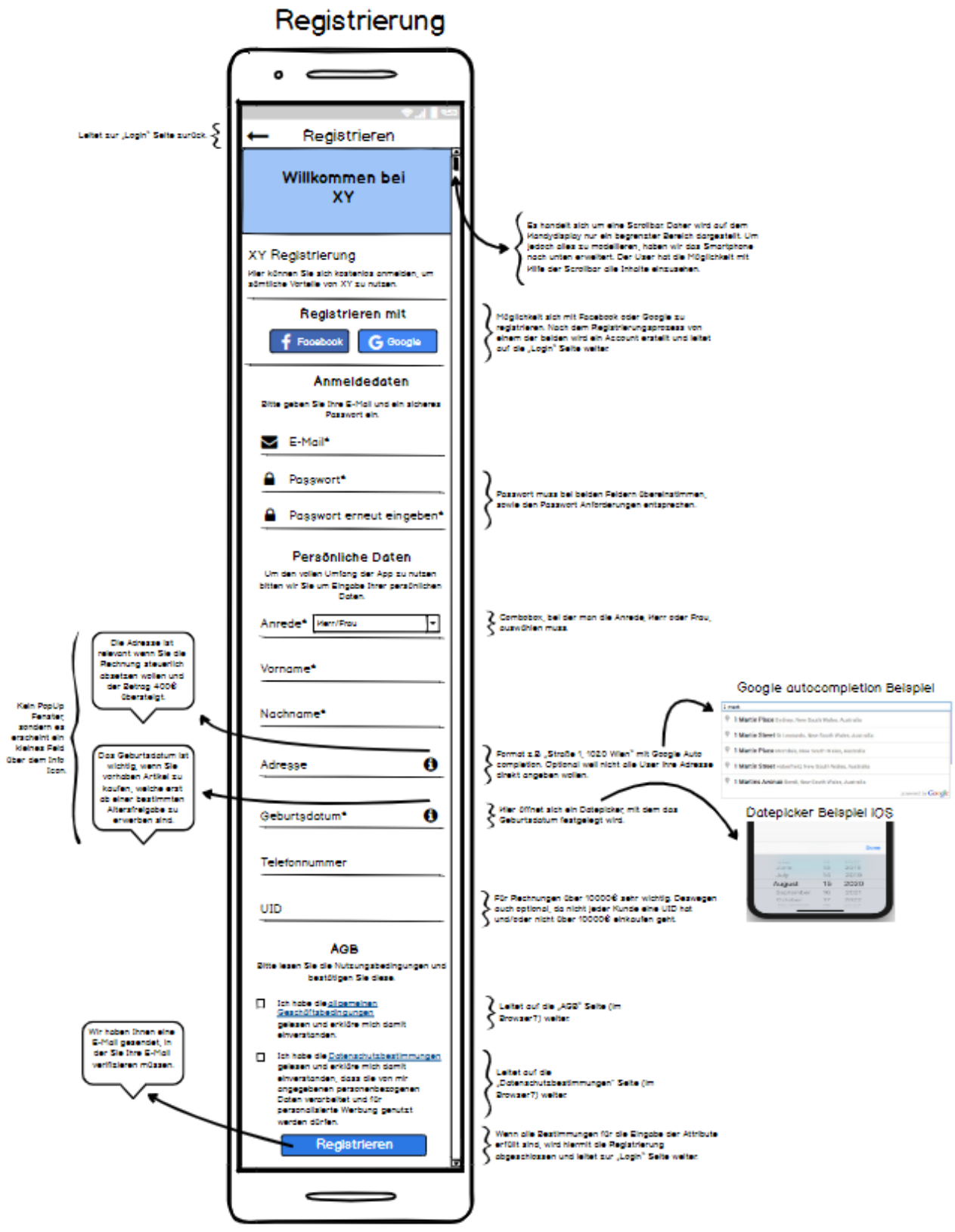
- E-Mail bestätigen
- Anmeldung
- Registrierung mit externem Anbieter

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



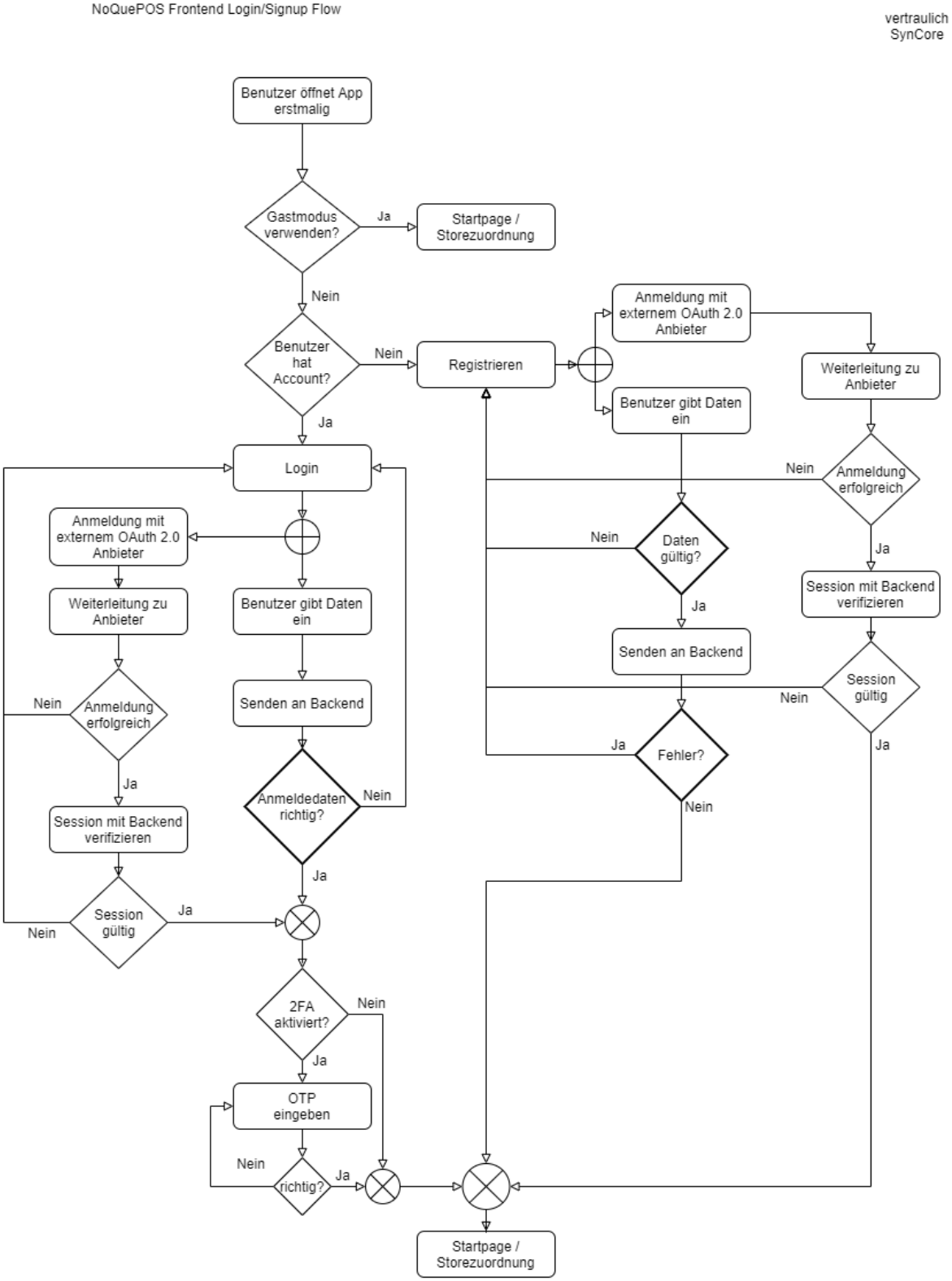
7.2.3 Funktionen der Masken

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

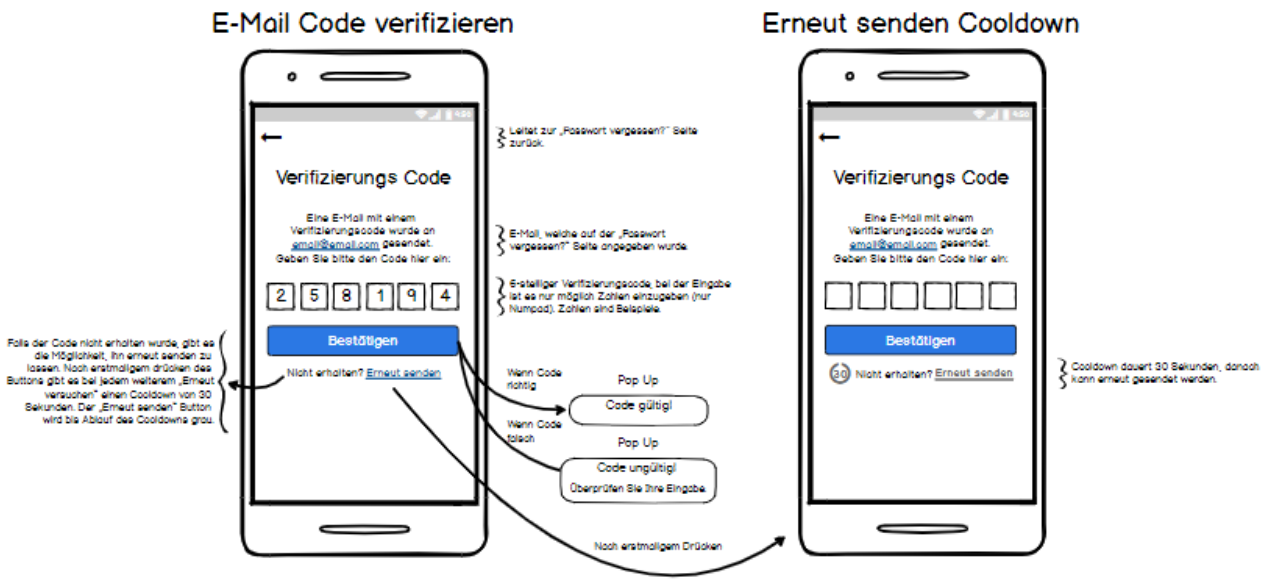
- E-Mail bestätigen
- Anmeldung
- Registrierung mit externem Anbieter

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

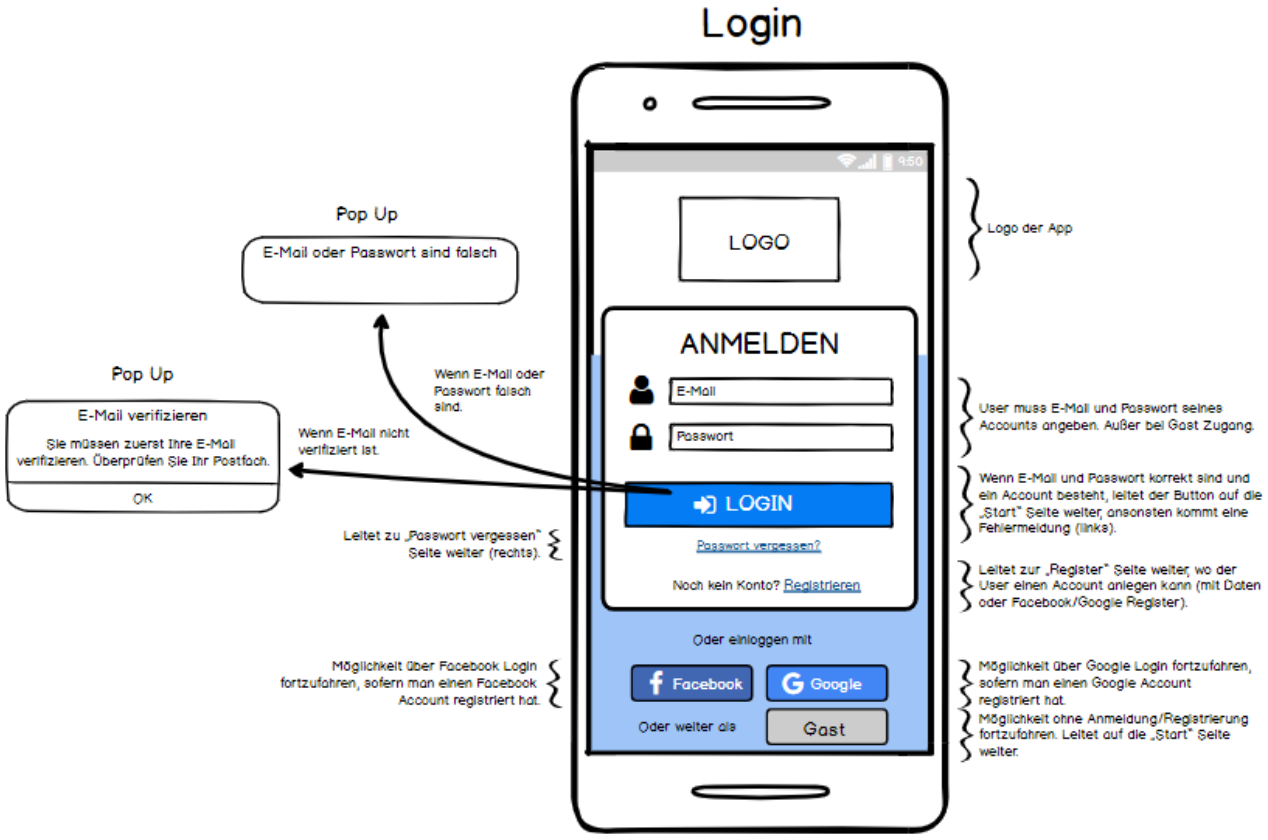
- Login
- Registrierung

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

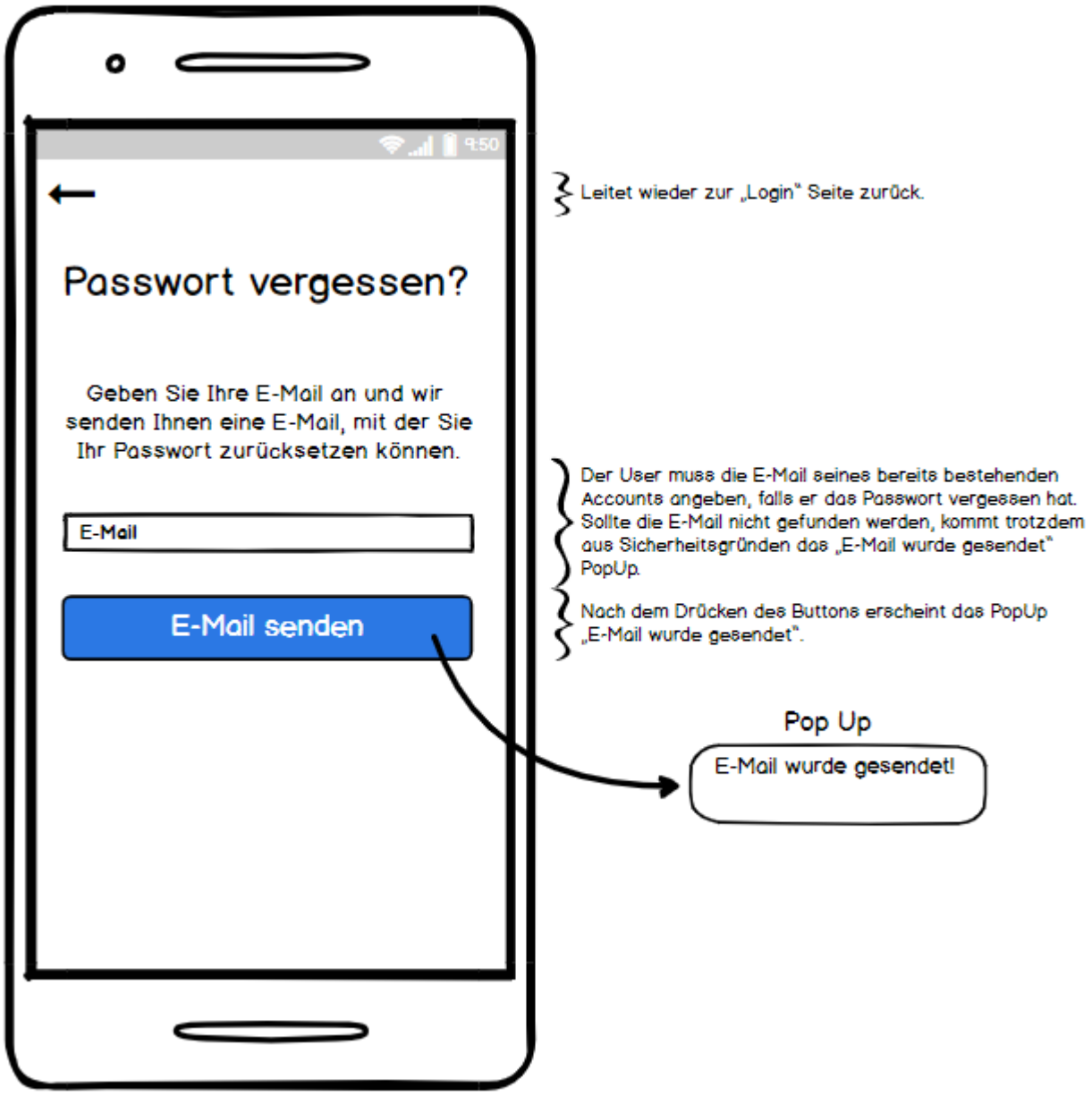
7.5 Userstory "Passwort vergessen"

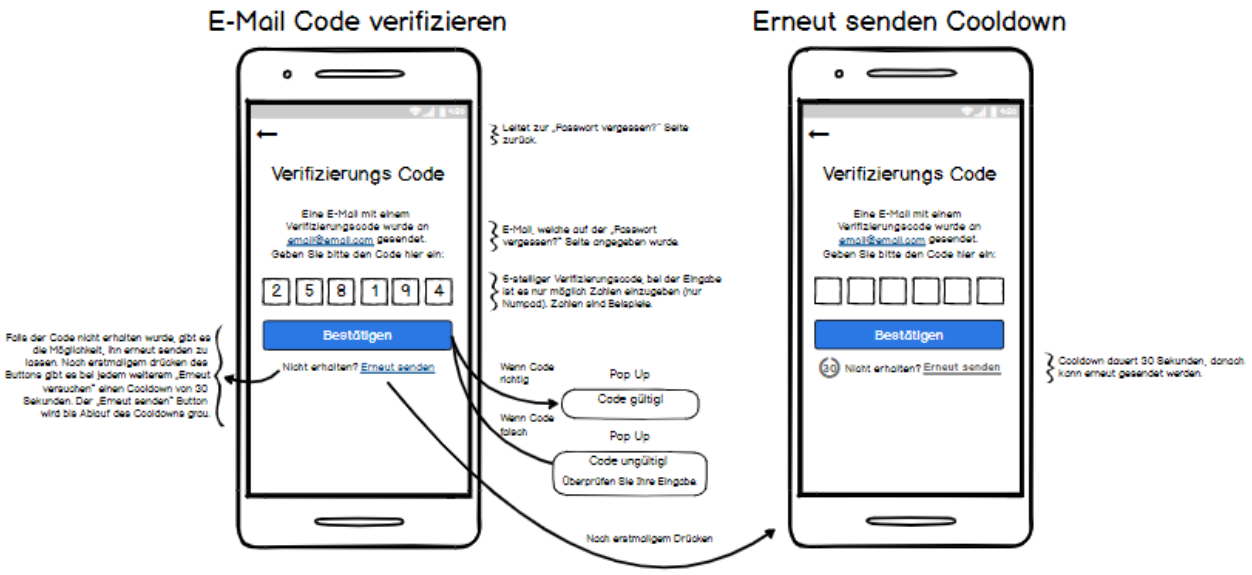
7.5.1 Beschreibung

" Als Kunde kann ich mein Passwort mit meiner E-Mail zurücksetzen, sodass ich ein neues Passwort für meinen bestehenden Account festlegen kann."

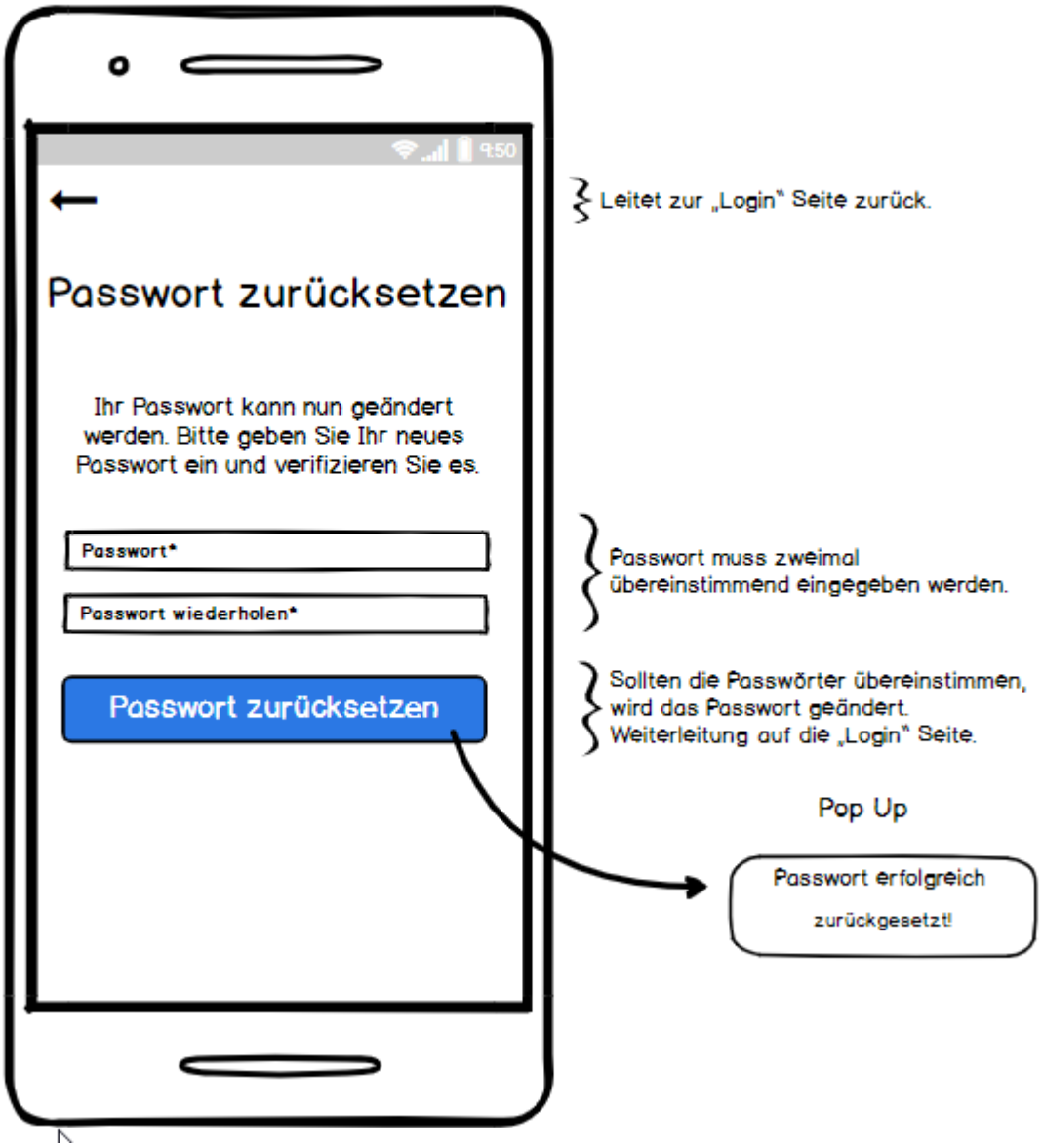
7.5.2 Screendesign

Passwort vergessen





Passwort zurücksetzen/ändern



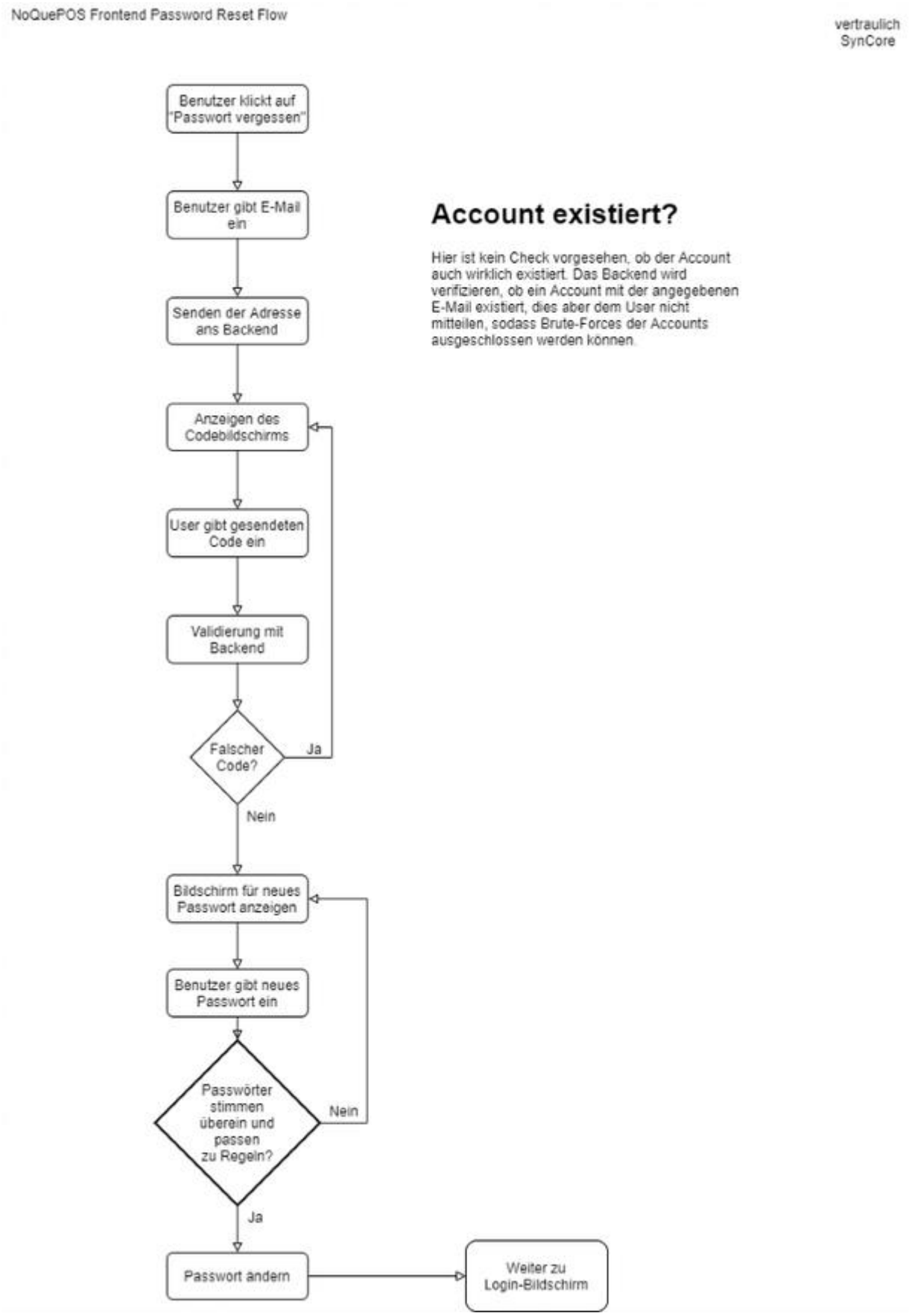
7.5.3 Funktionen der Masken

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

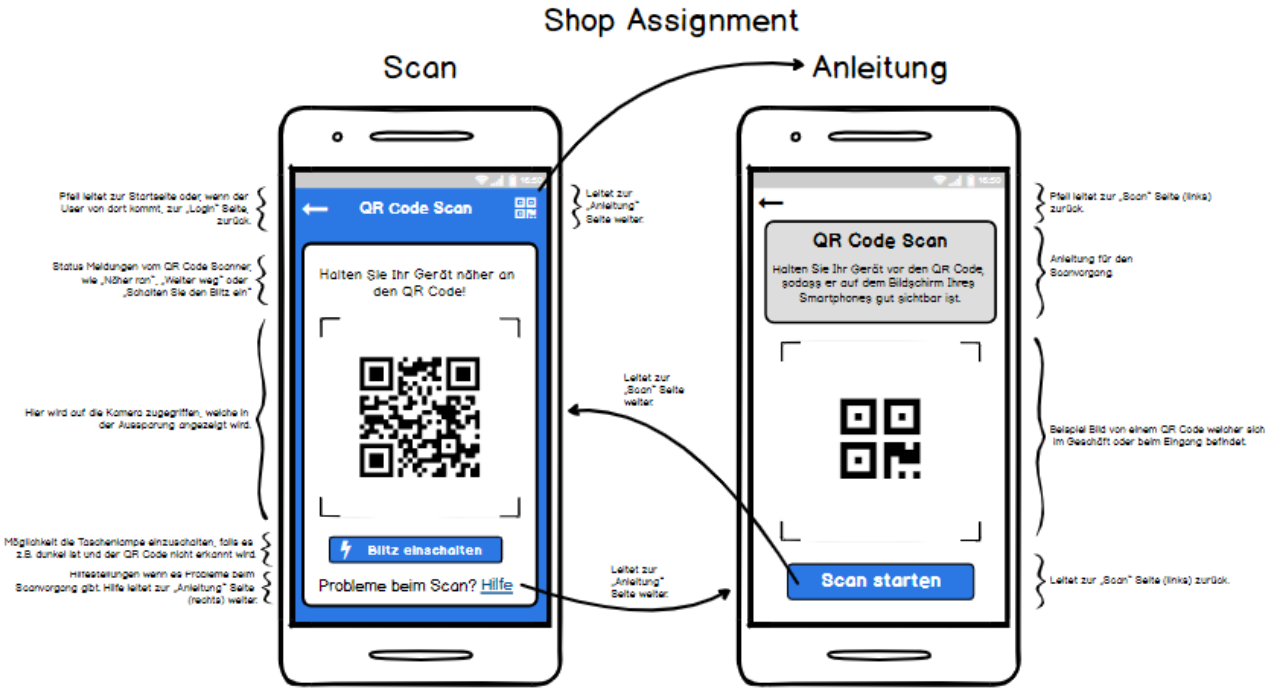
8 Epic „Shop Assignment“

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



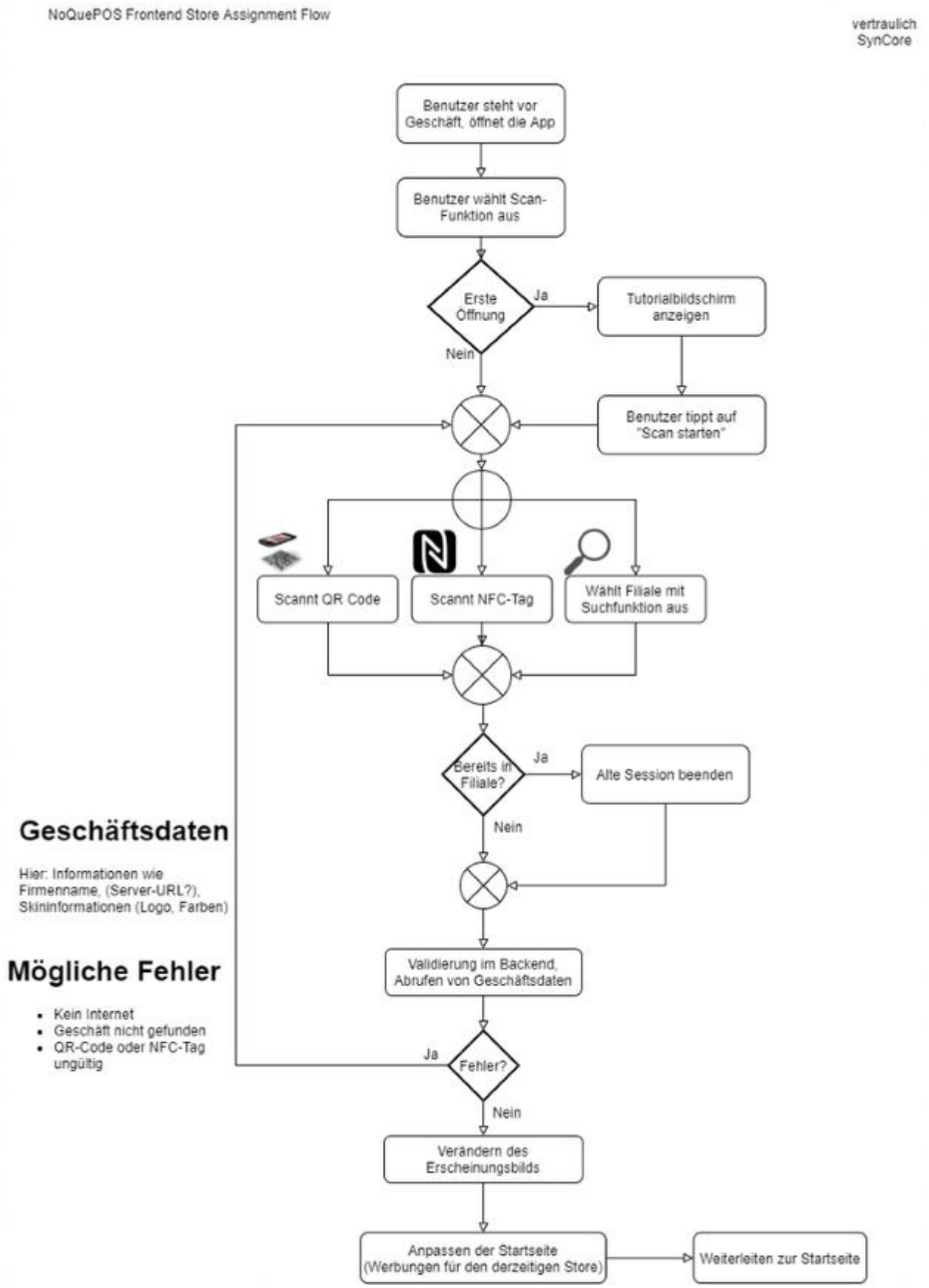
8.1.3 Funktionen der Masken

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.

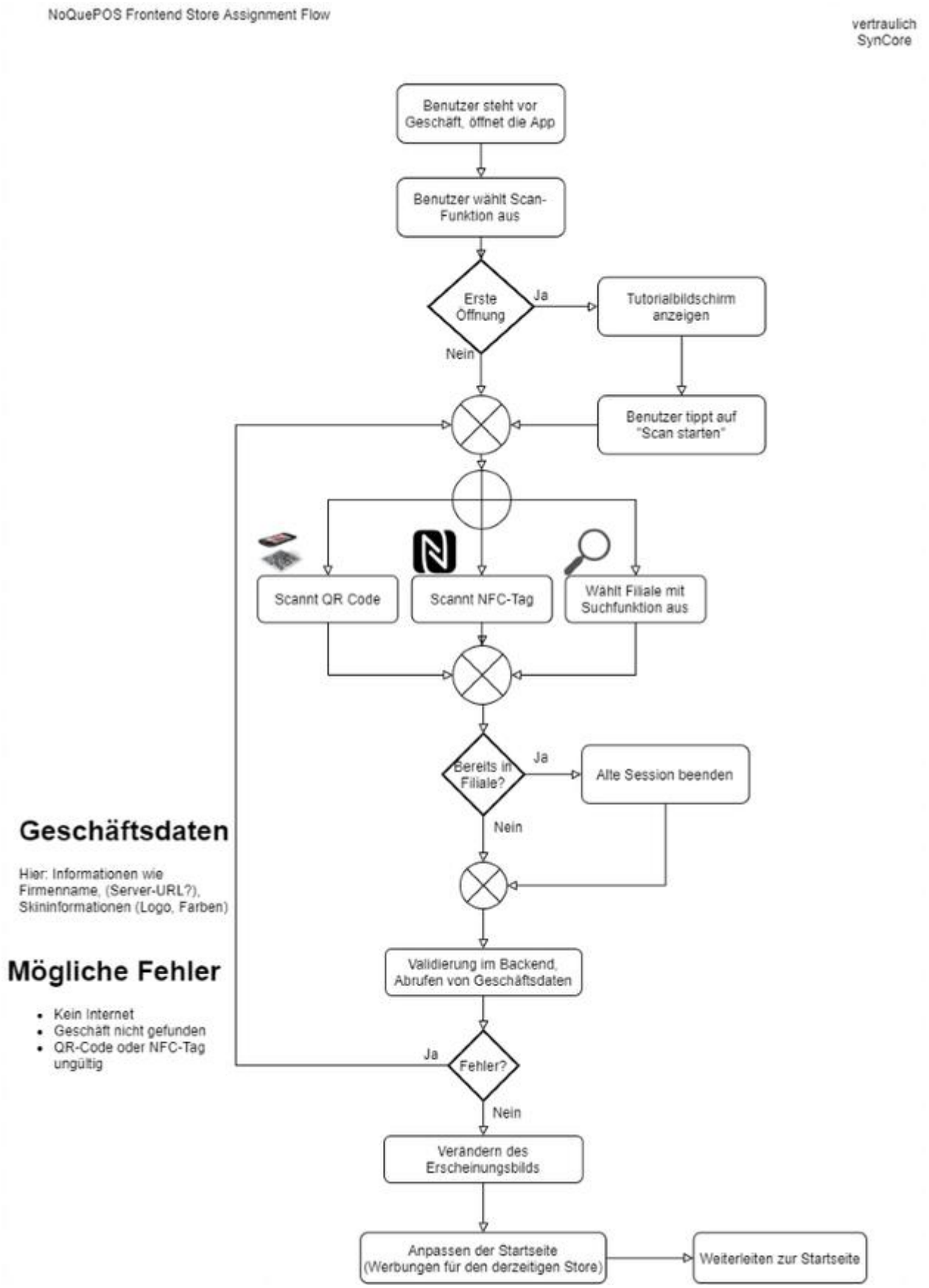
Als Kunde kann ich während des Einkaufens jederzeit den Einkauf abbrechen, sodass der gesamte Warenkorb gelöscht wird und ich aus dem Shop ausgeloggt werde.

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"

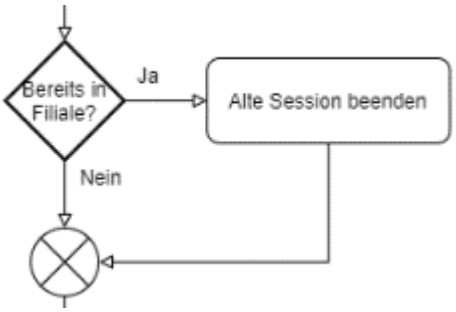
8.4.1 Beschreibung

" 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

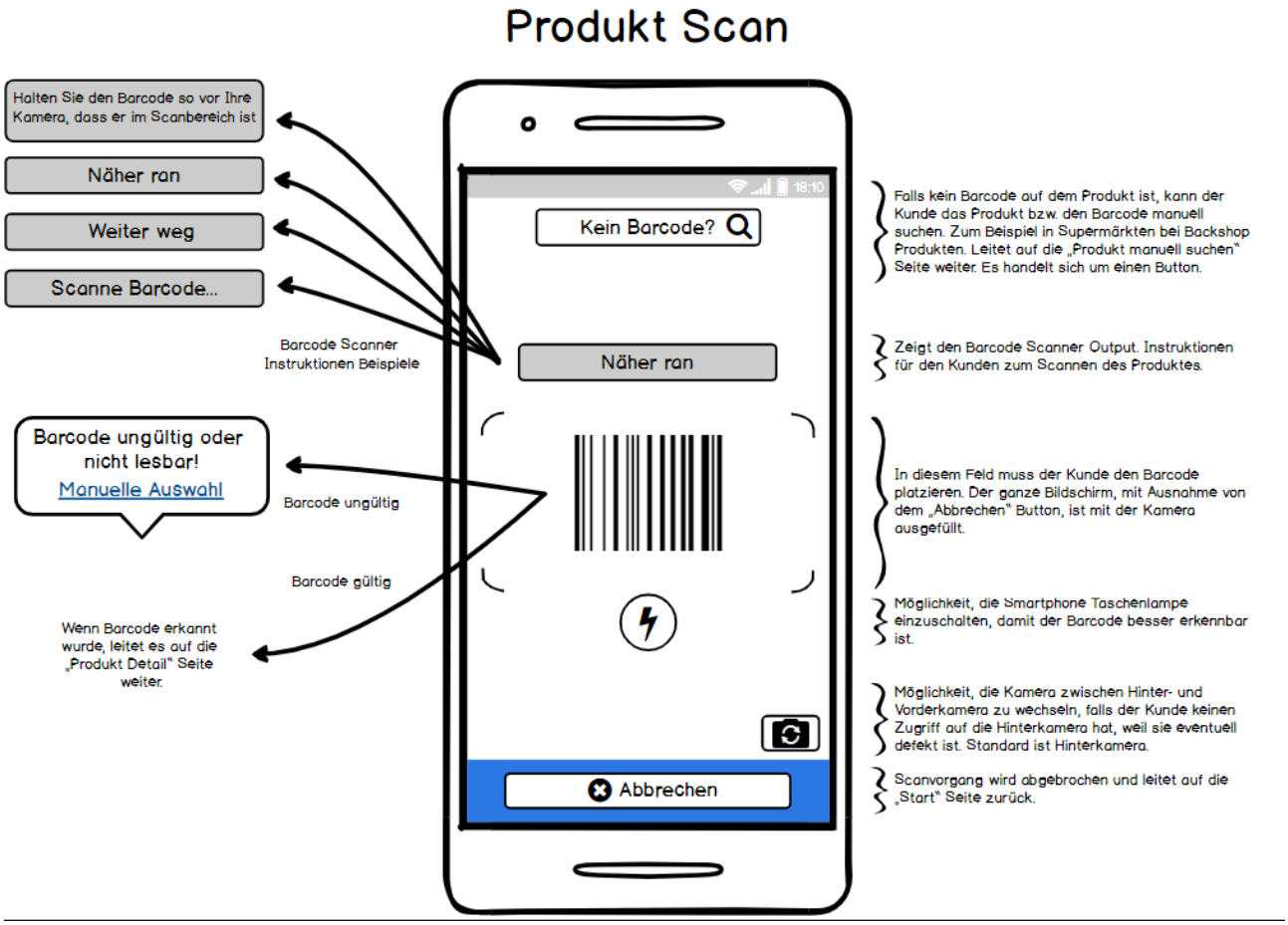
9 Epic „Shopping Cart“

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."

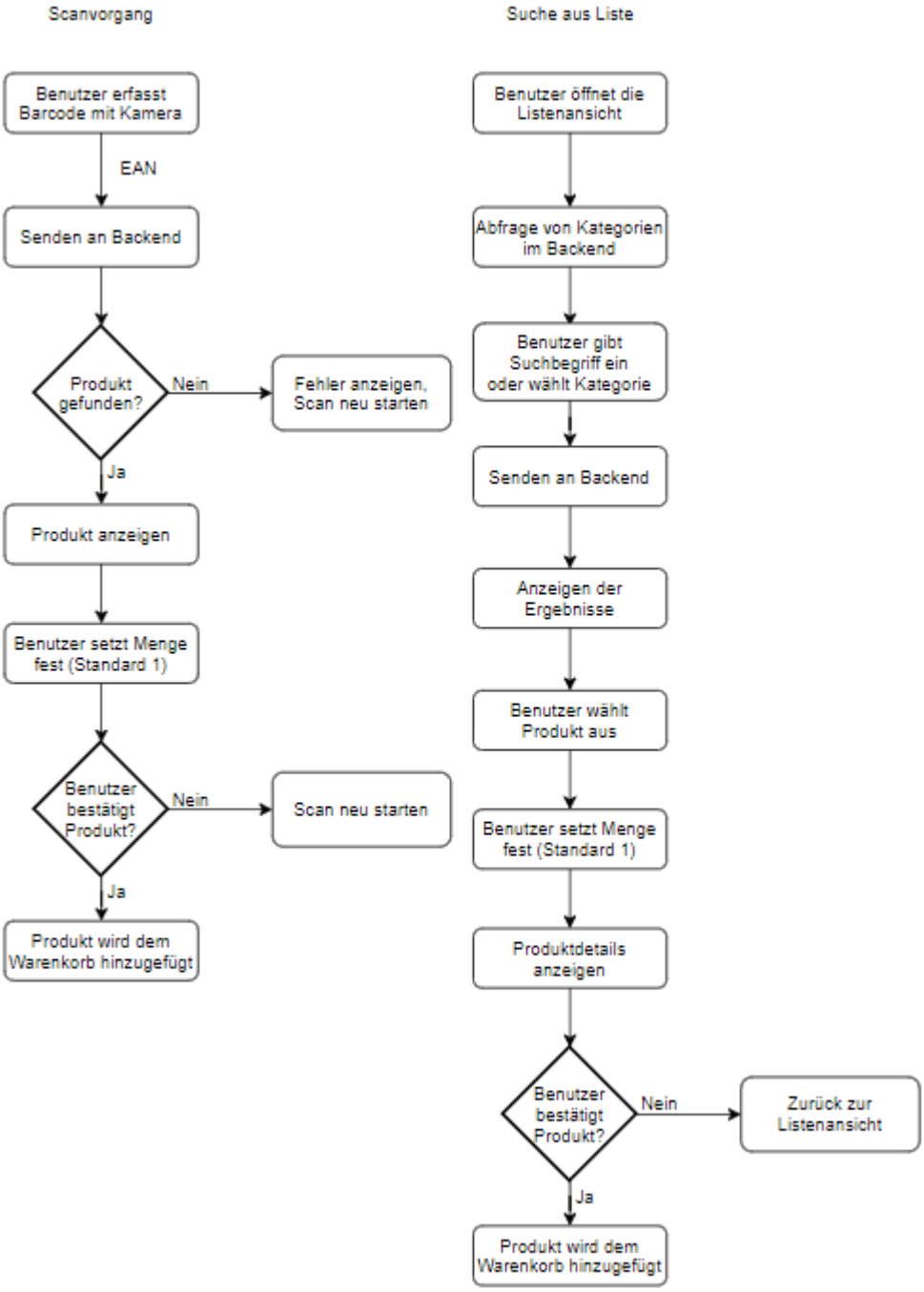
9.1.2 Screendesign



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

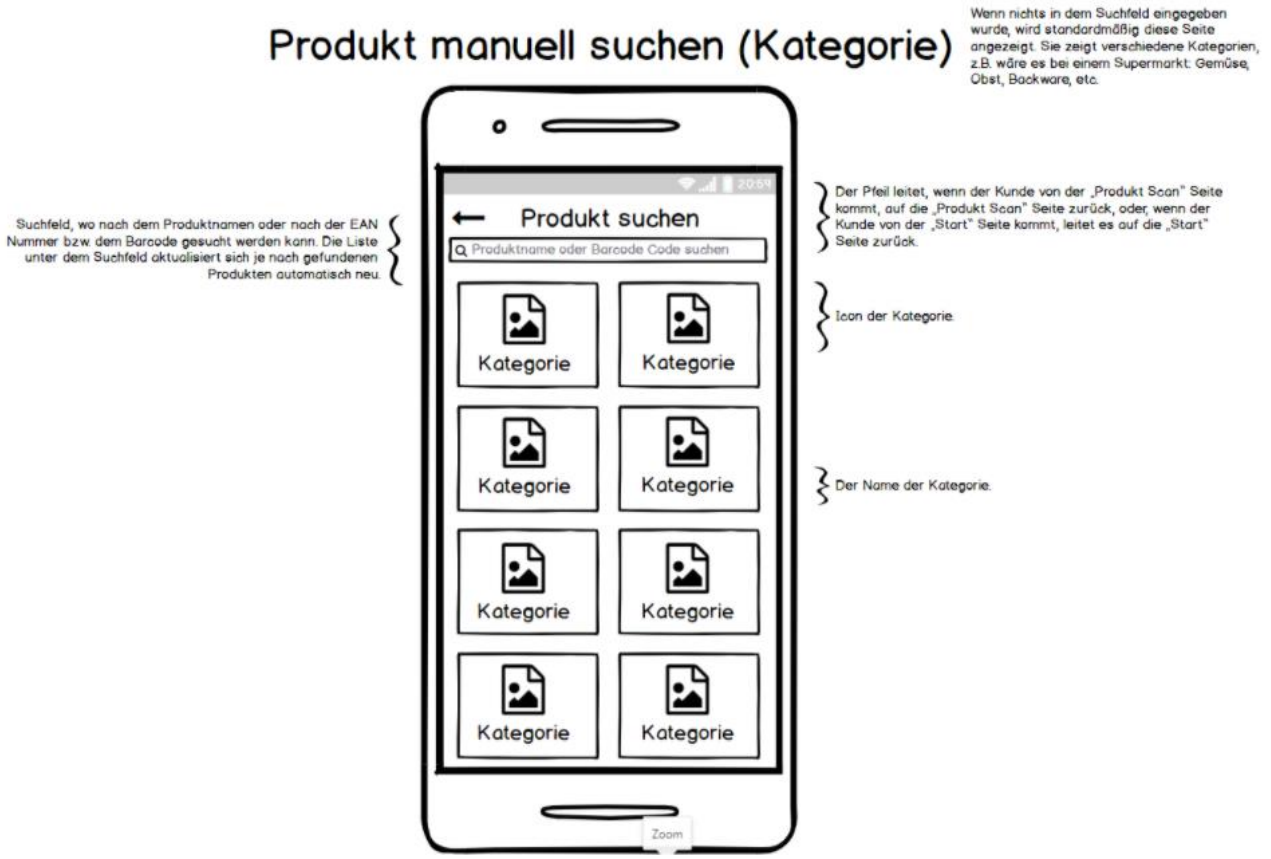
- Shopanmeldung vorhanden

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

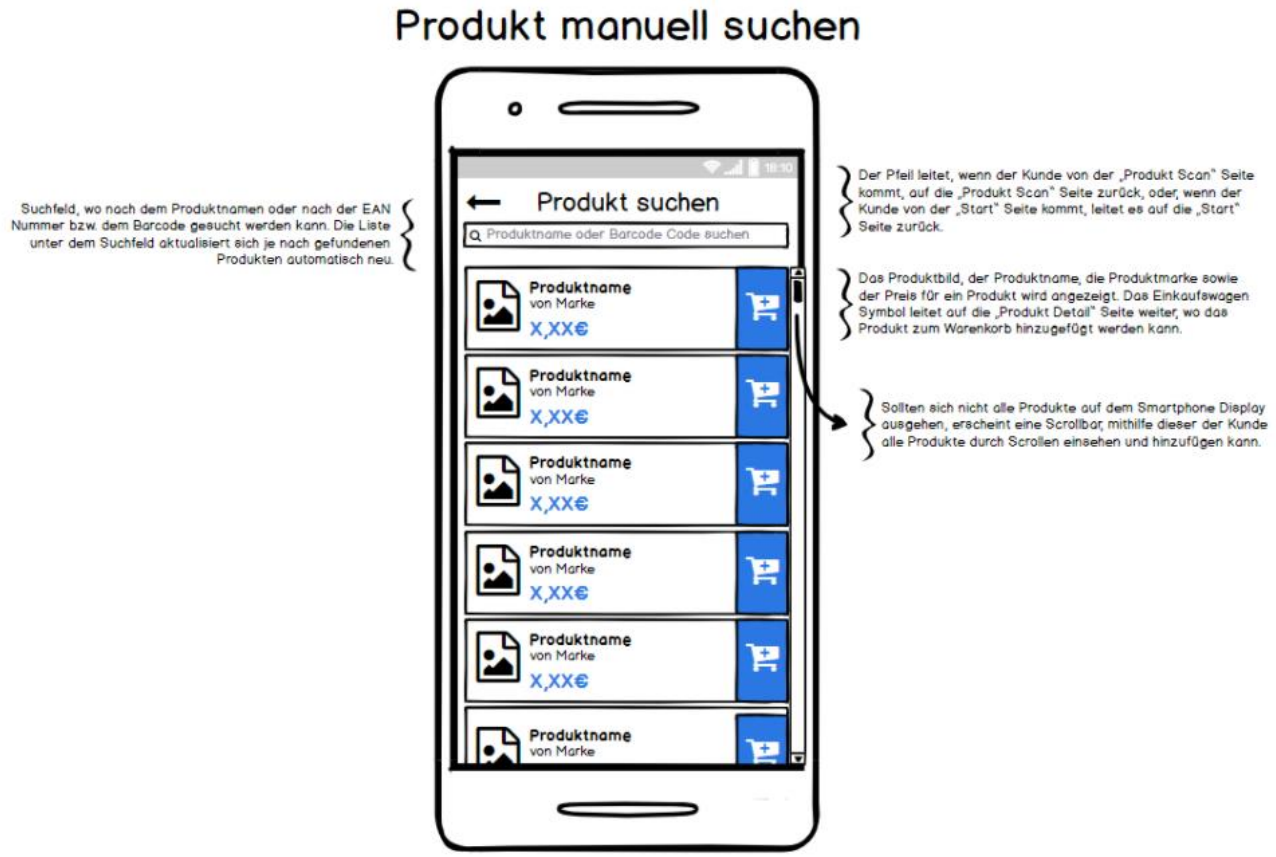
- Shopanmeldung vorhanden

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

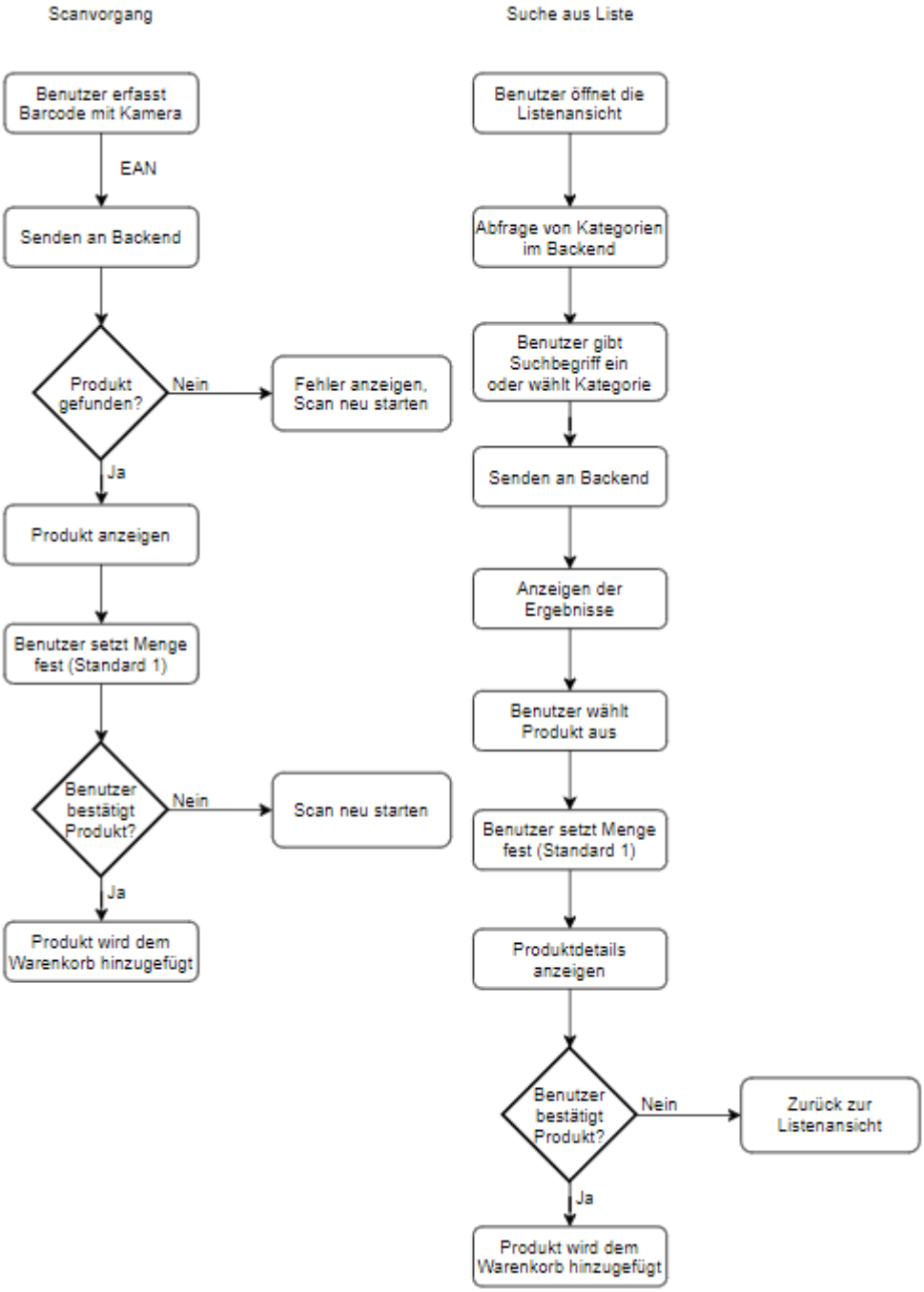


9.3.3 Funktionen der Masken

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

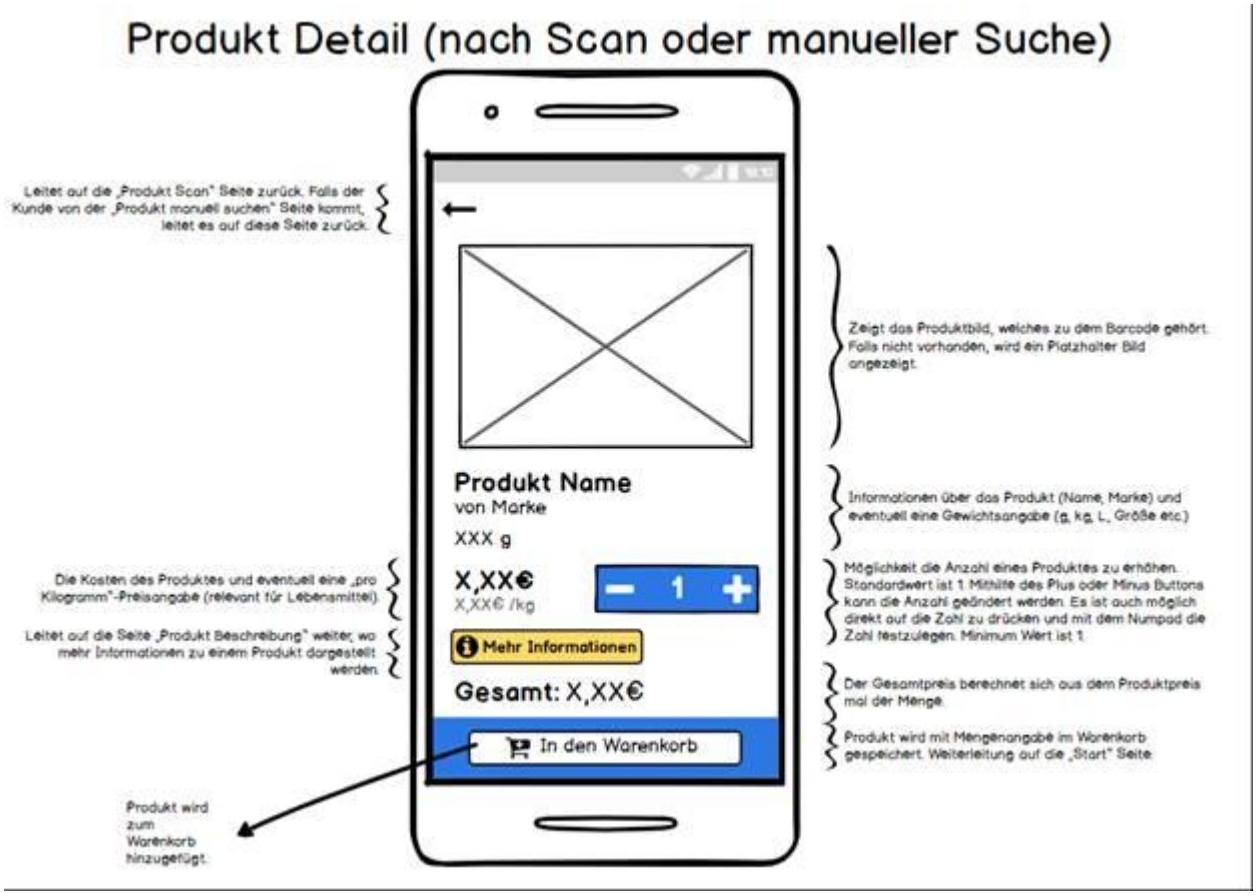
- Shopanmeldung vorhanden

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

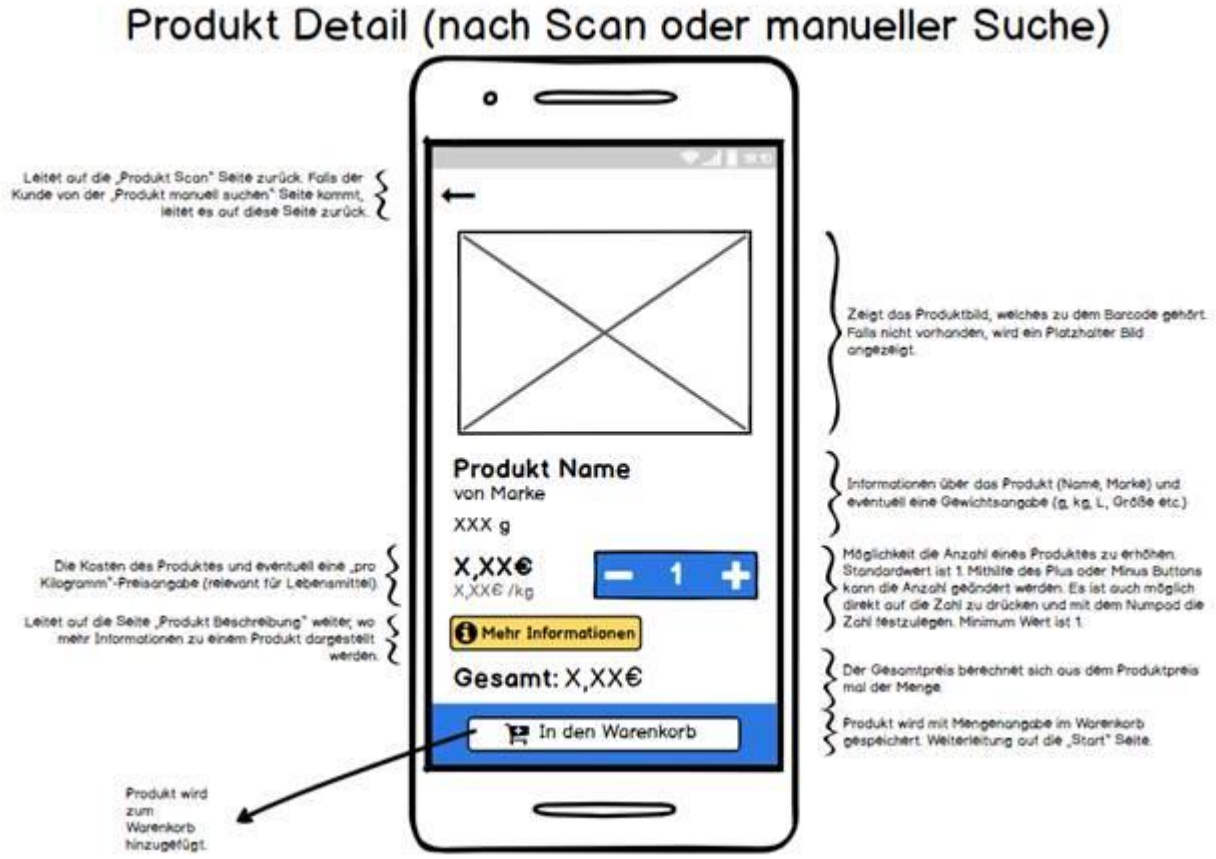
Warenkorb, Produkt hinzufügen

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

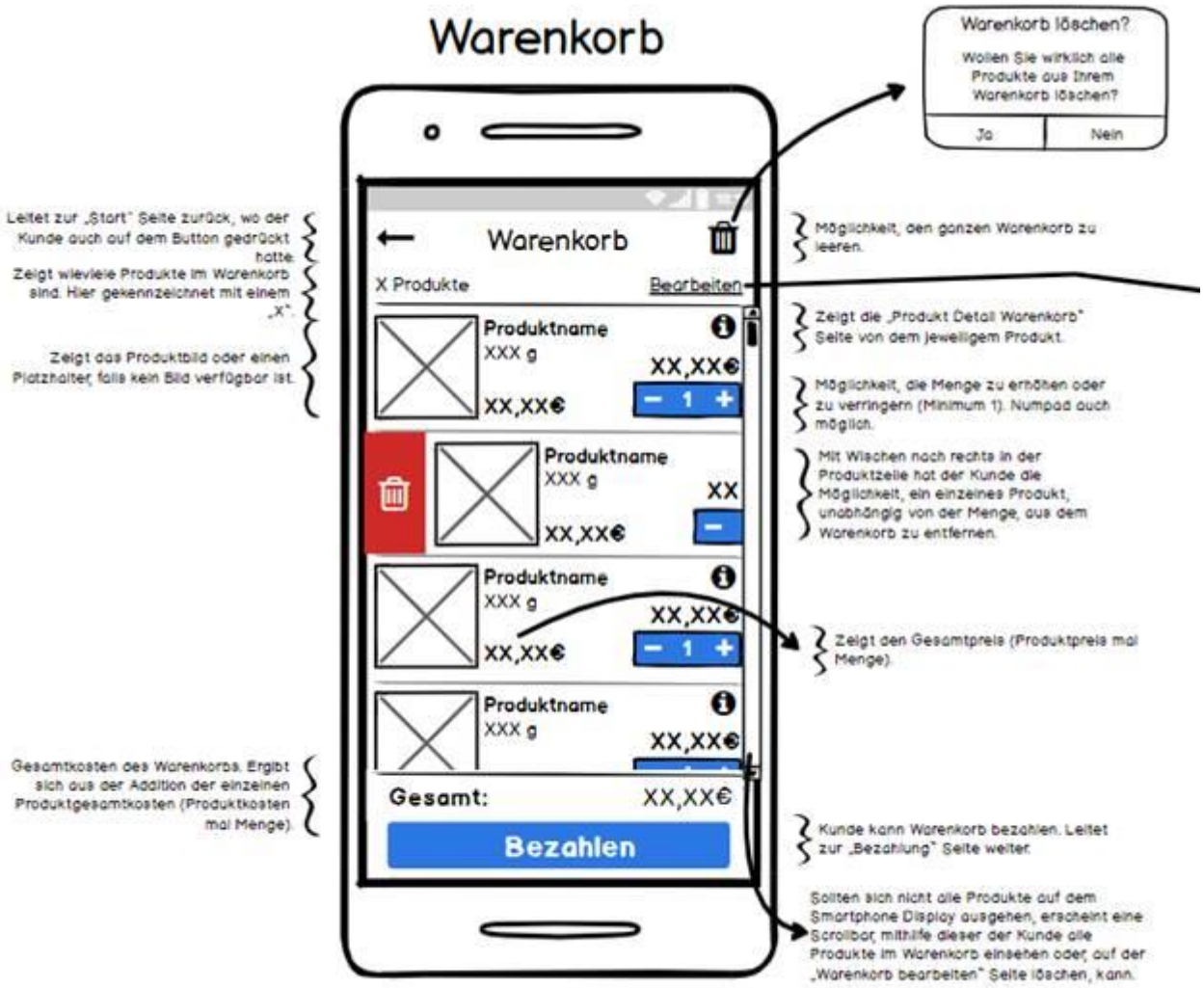
Produkt hinzufügen, Warenkorblogik

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

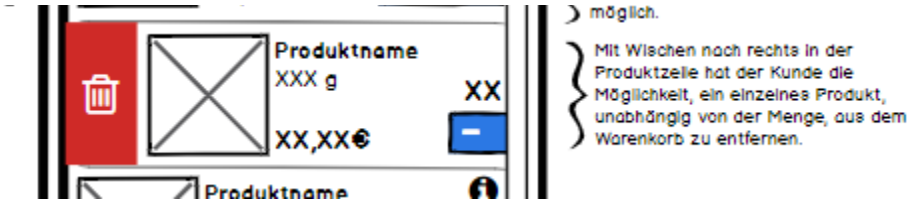
Produkt hinzufügen, Warenkorblogik

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

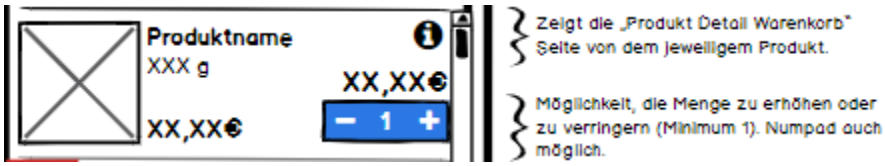
Warenkorblogik

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

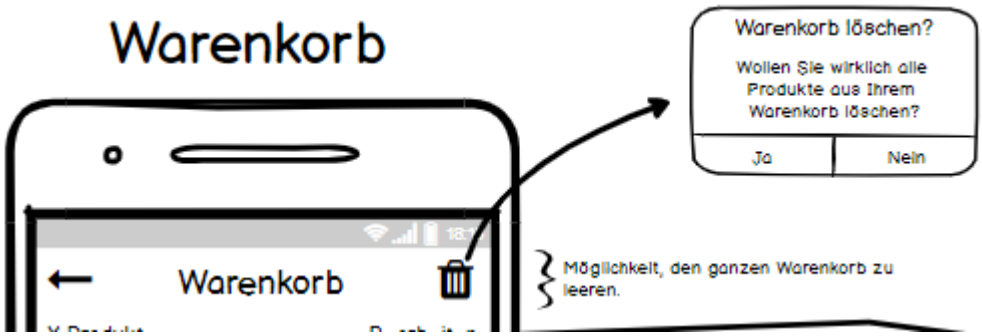
Warenkorblogik, Produkt hinzufügen

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

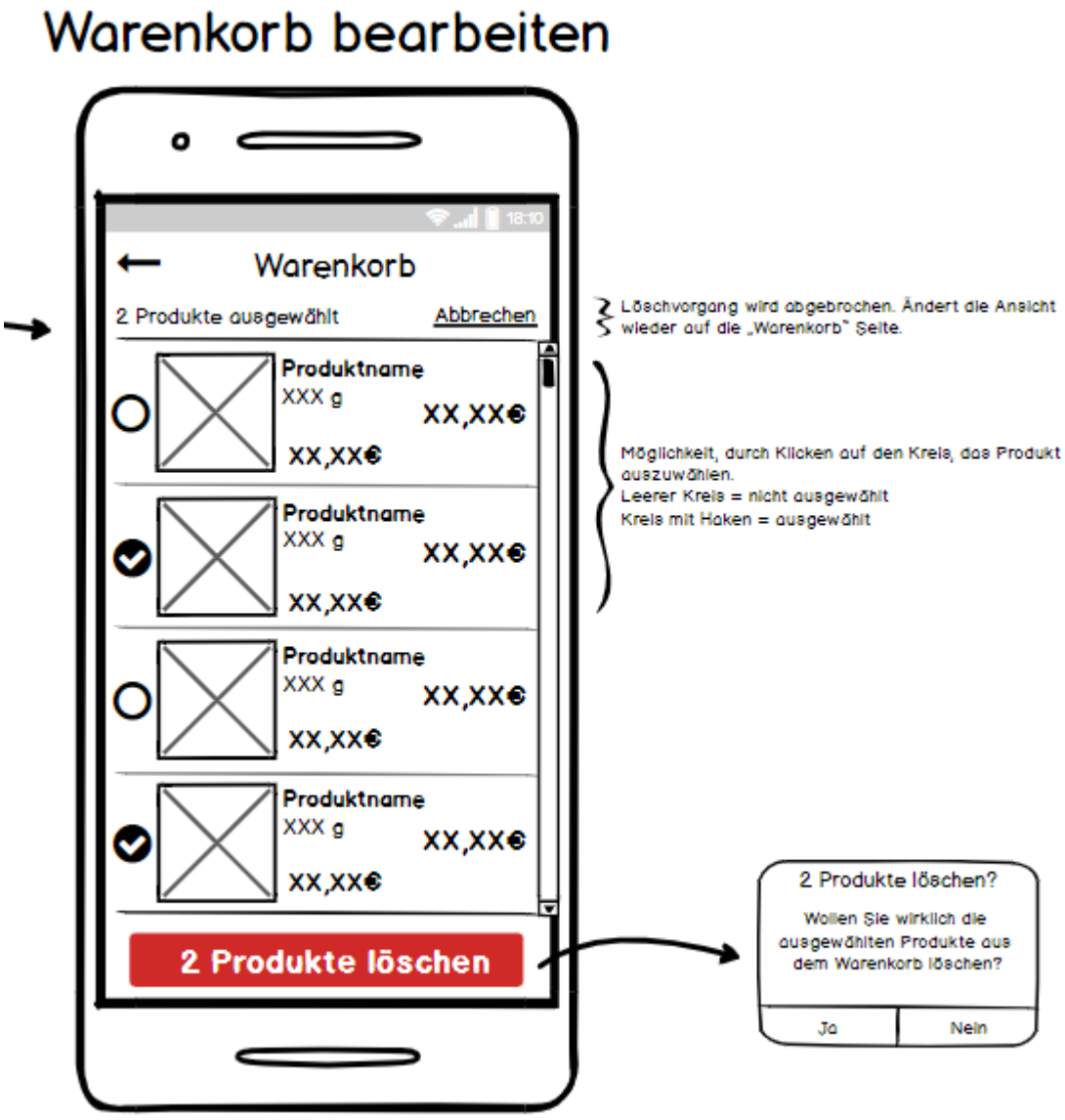
Warenkorblogik

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.

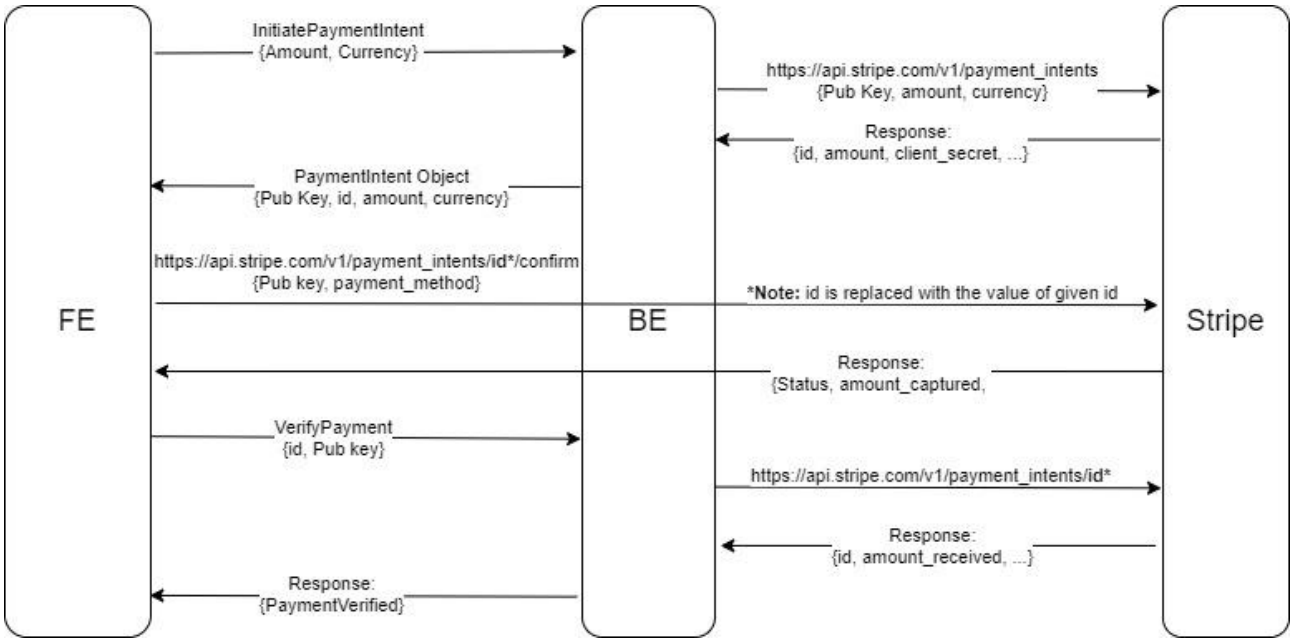
11.1.2 Screendesign

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.

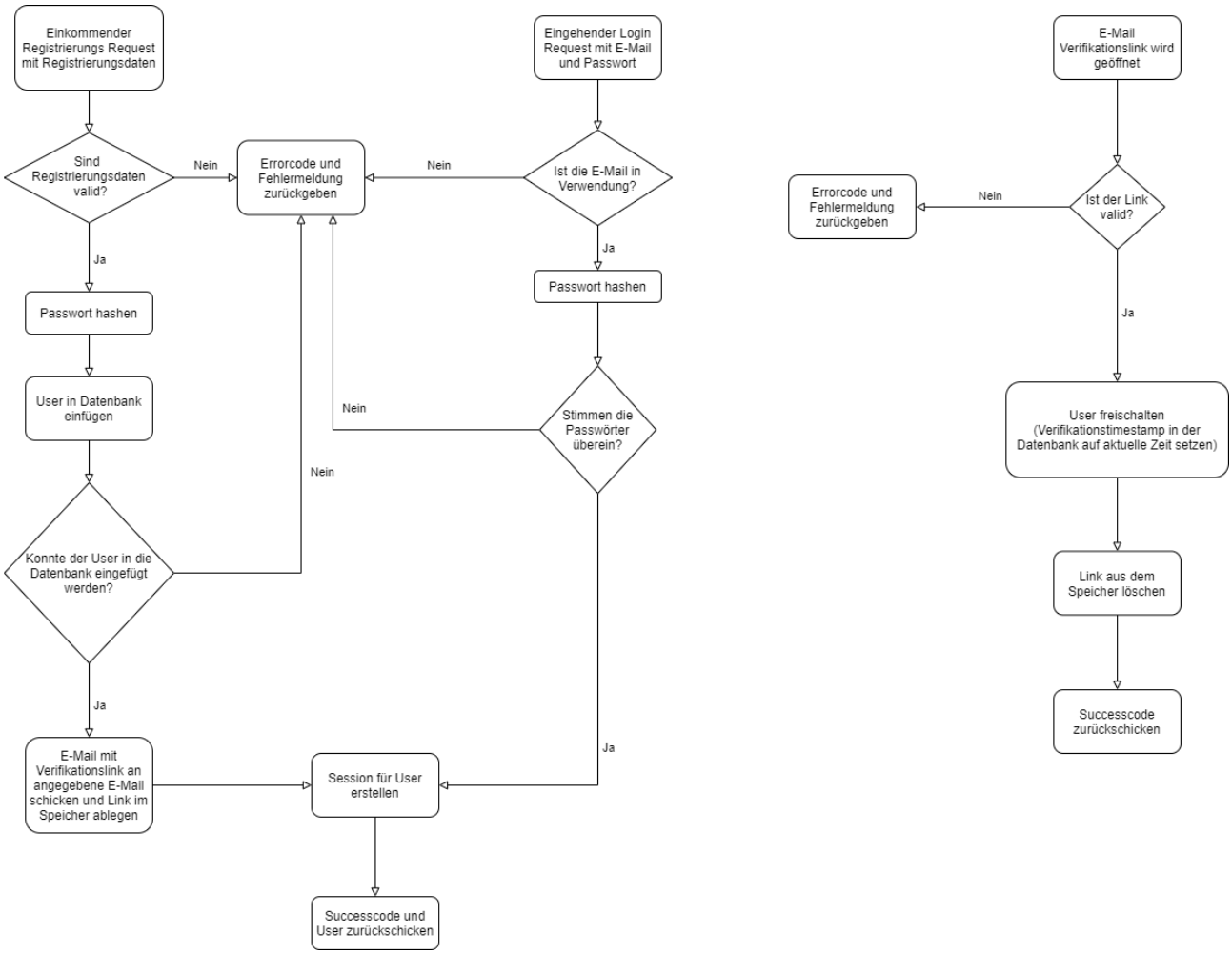
13.1.6 Abgrenzung zu anderen Prozessen

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

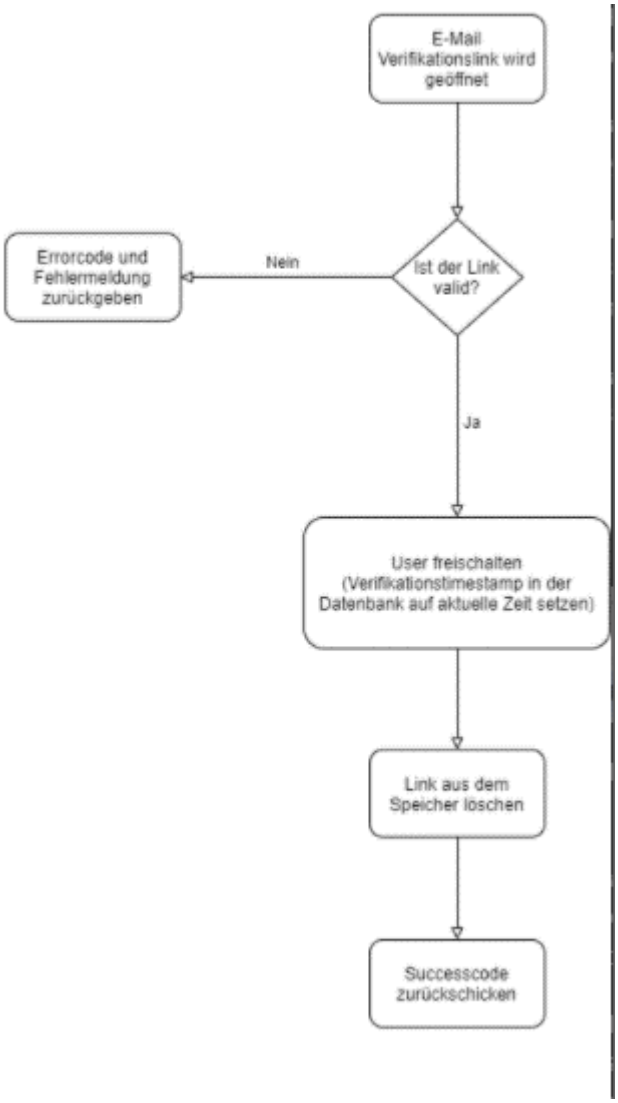
13.3.6 Abgrenzung zu anderen Prozessen

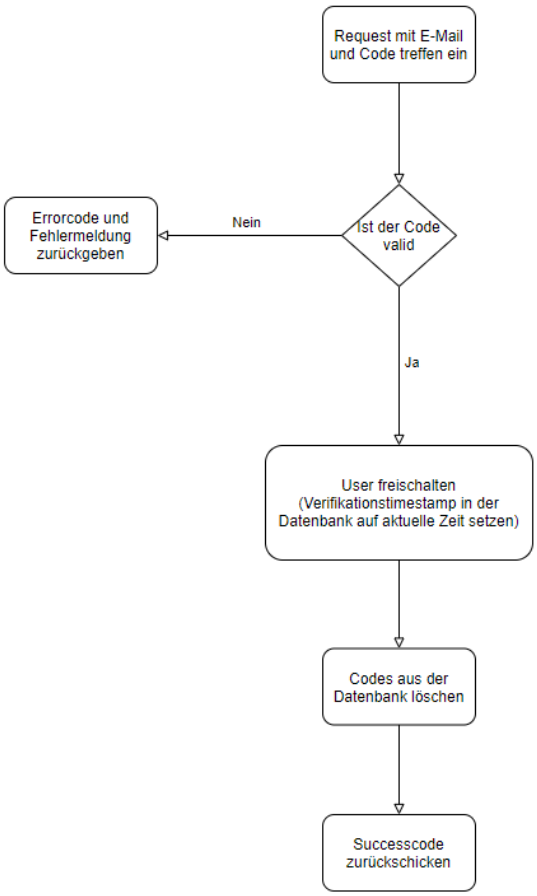
- Login
- Registrierung

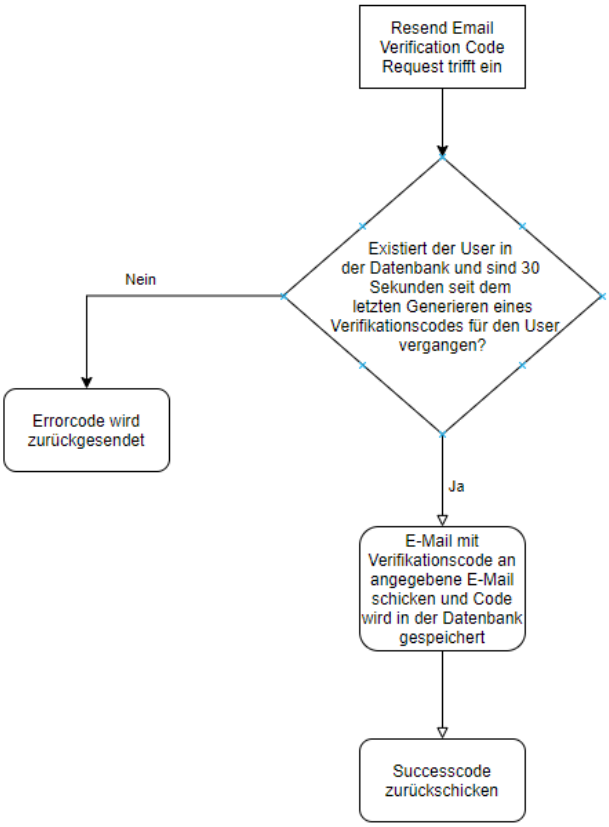
13.3.7 Voraussetzungen

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;

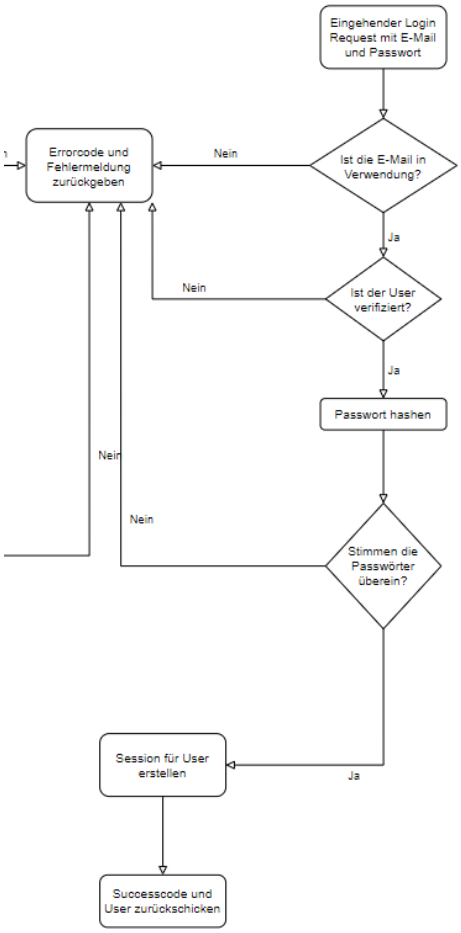
14.1.6 Abgrenzung zu anderen Prozessen

- 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

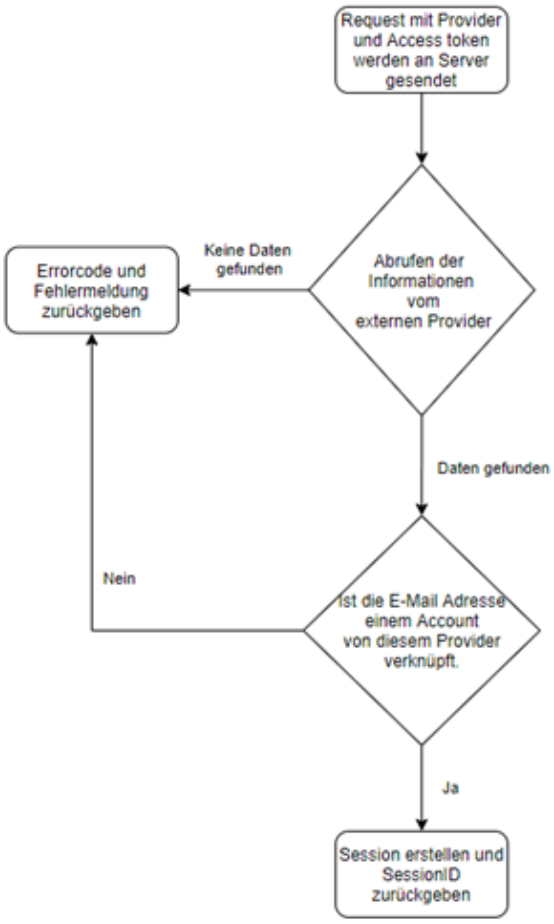
14.2.5 Abgrenzung zu anderen Prozessen

- 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

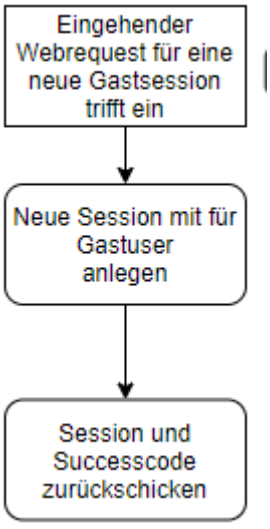
14.3.6 Abgrenzung zu anderen Prozessen

- 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

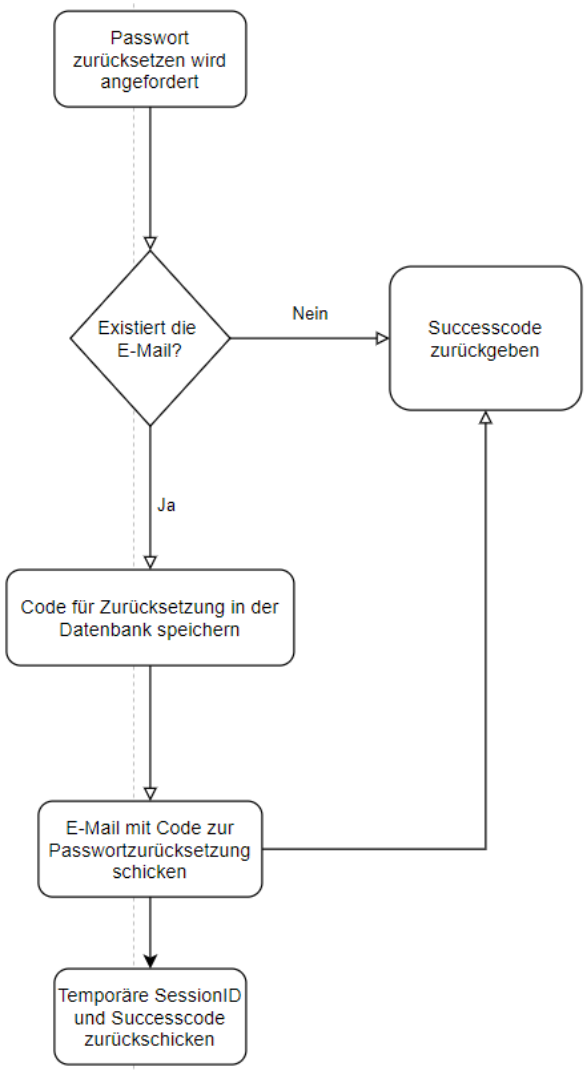
14.4.6 Abgrenzung zu anderen Prozessen

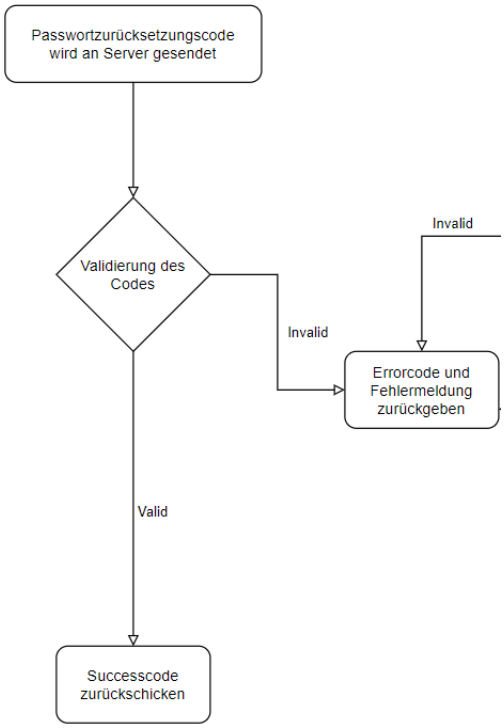
- Passwort ändern
- Login

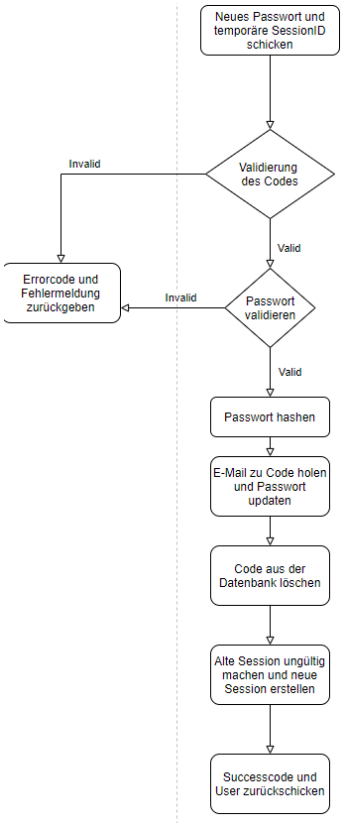
14.4.7 Voraussetzungen

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

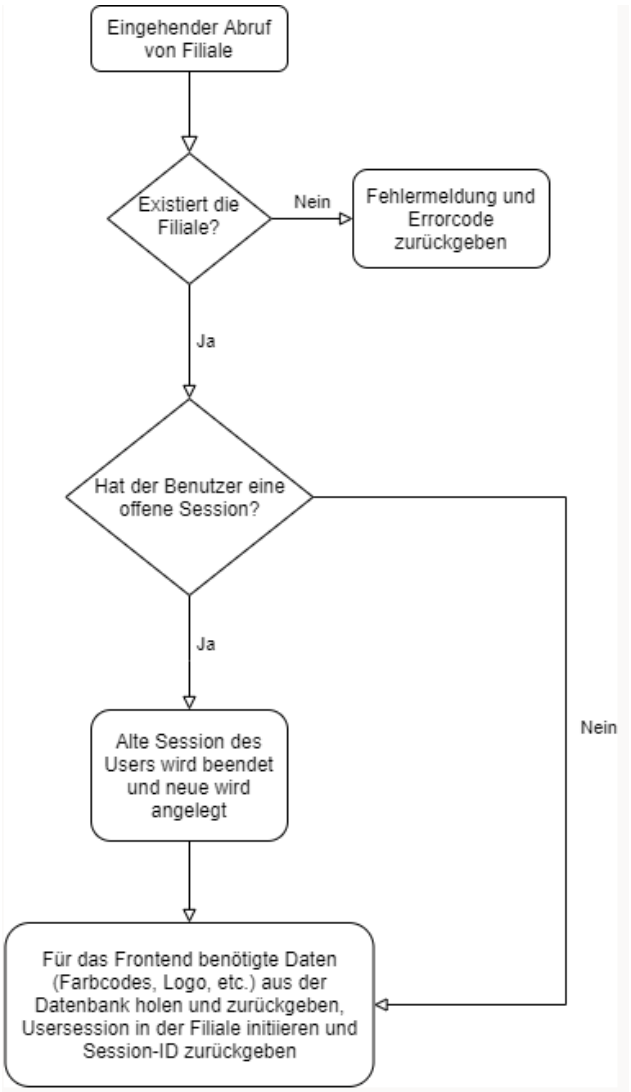
15.1.6 Abgrenzung zu anderen Prozessen

- 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

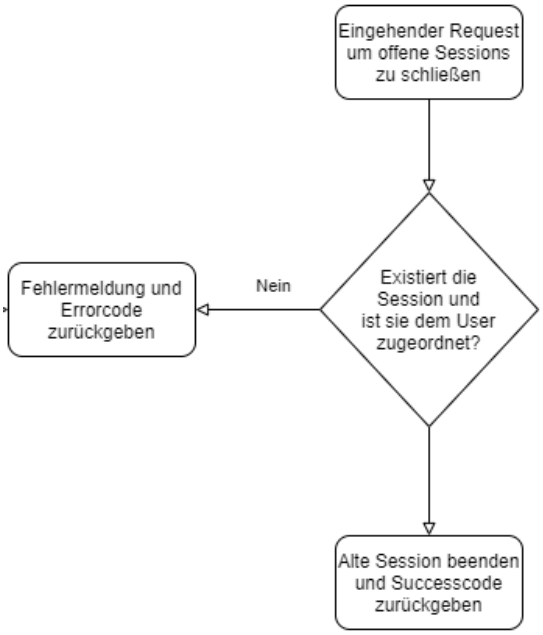
15.3.6 Abgrenzung zu anderen Prozessen

- 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"

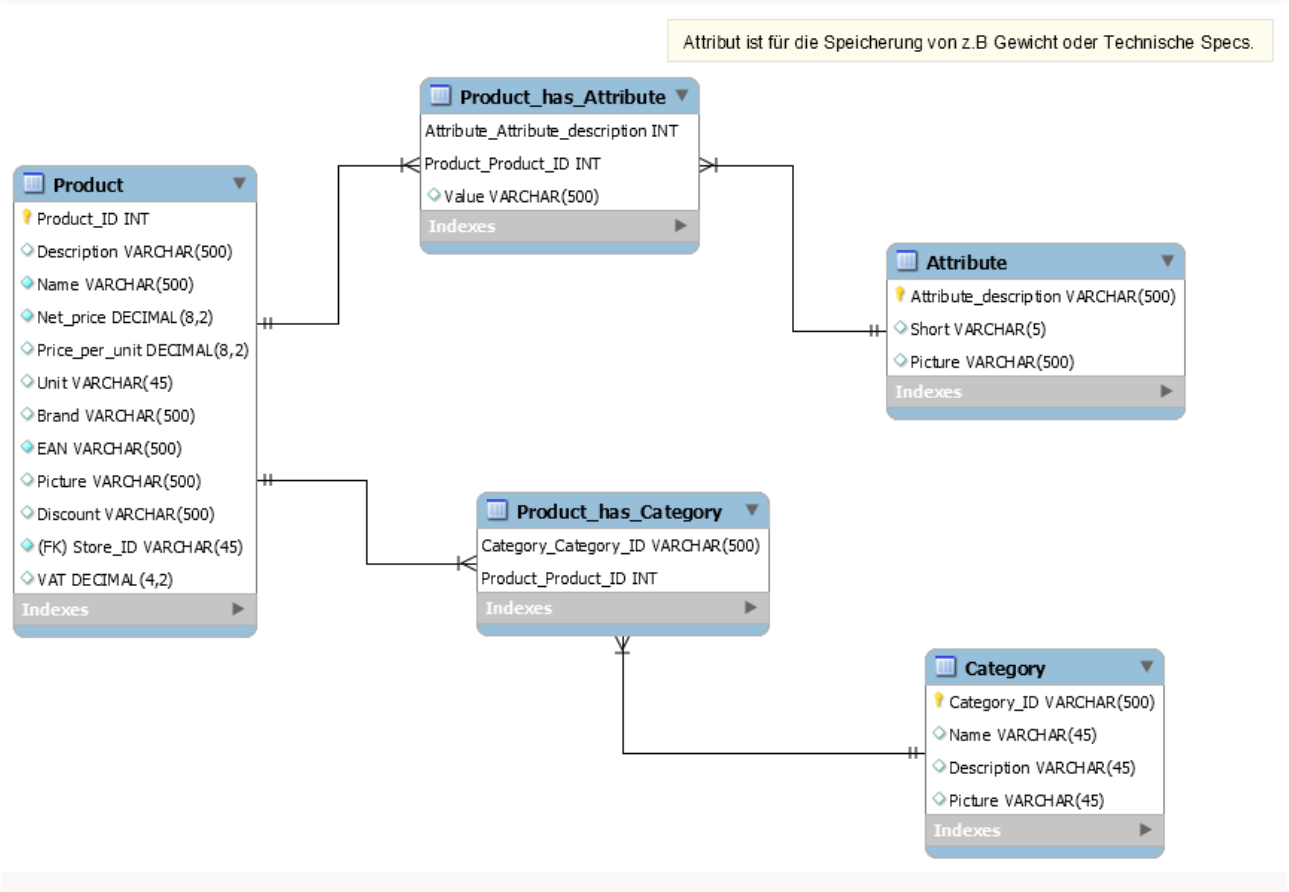
16.1.1 Beschreibung

"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

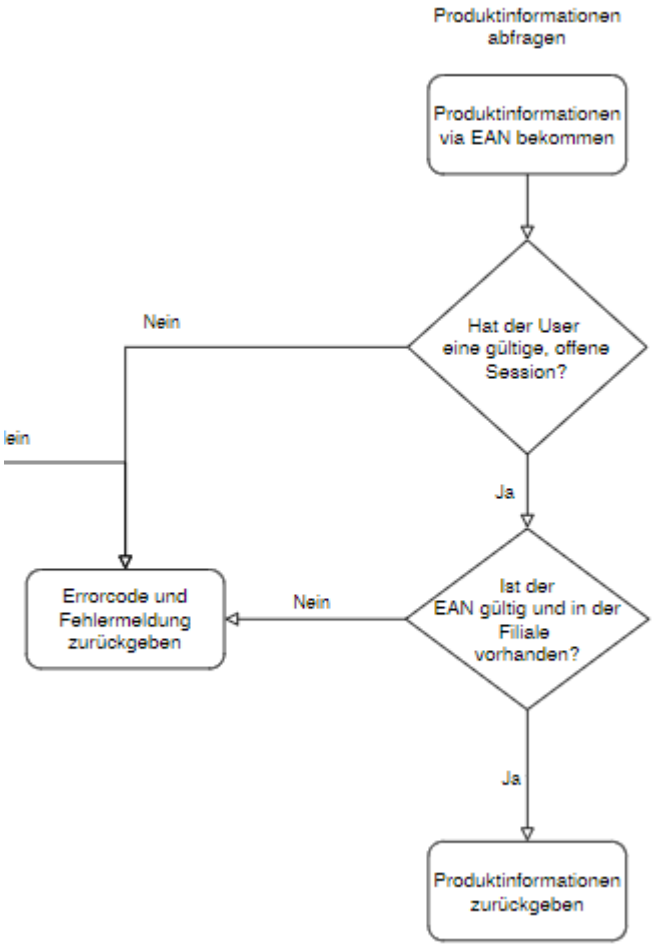
16.1.6 Abgrenzung zu anderen Prozessen

- 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

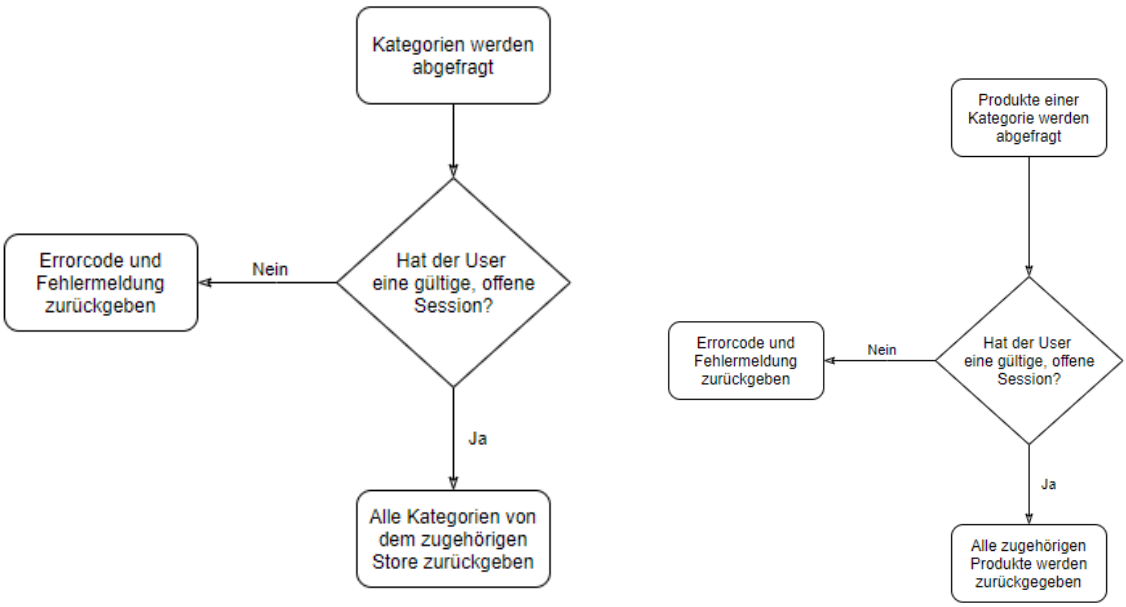
16.2.6 Abgrenzung zu anderen Prozessen

- 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

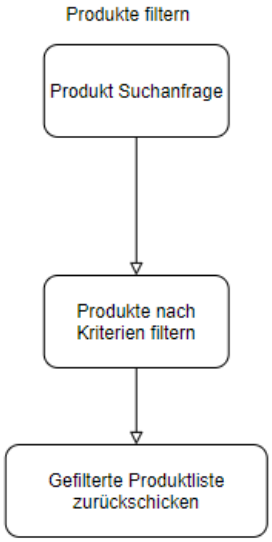
16.3.6 Abgrenzung zu anderen Prozessen

- 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