

Aperta - Das smarte Garagentor

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

**Höheren Abteilung für Informationstechnologie mit
Ausbildungsschwerpunkt Medientechnik**

Eingereicht von:

Benjamin Golic
David Hauser
Simon Koll

Betreuer:

Prof. Christian Aberger

Leonding, April 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022

Benjamin Golic & David Hauser & Simon Koll

Abstract

APERTA is a garage door system that was developed by Benjamin Golic, David Hauser and Simon Koll as part of their diploma thesis. The system can be retrofitted to electric garage doors regardless of the manufacturer and enables entry into the world of the Internet of Things, the IOT. Depending on the configuration, the system consists of up to 3 components, each of which represents an access option to the garage. These are a classic numpad, an RFID reader, and a camera, which is used for license plate recognition. In the future, these can be expanded to include other or further authentication options. A web dashboard is used to manage the combinations of the numpad , NFC card details or license plates. These can be provided with a time validity range. There is also an online store where the product can be configured and spare parts or expansion modules can be purchased. The dashboard, which was implemented using Angular, communicates with the Node-JS server as the backend. It acts as the core, passes the information to the MongoDB database and is available for requests from the Raspberry Pi, which is responsible for the hardware.



Abbildung 1: Rendered
Prototyp

Zusammenfassung

APERTA ist ein Garagentorsystem, welches von Benjamin Golic, David Hauser und Simon Koll im Rahmen der Diplomarbeit entwickelt wurde. Das System ist herstellerunabhängig bei elektrischen Garagentoren nachrüstbar und ermöglicht den Einstieg in die Welt des Internet der Dinge, dem IOT. Das System besteht, je nach Konfiguration, aus bis zu 3 Komponenten, welche je eine Zutrittsmöglichkeit zur Garage darstellen. Es handelt sich hierbei um ein klassisches Nummernfeld, einem RFID-Lesegerät, und einer Kamera, welche für eine Kennzeichenerkennung genutzt wird. Zukünftig können diese noch um andere oder weitere Möglichkeiten der Authentifizierung erweitert werden. Über ein Web-Dashboard werden die Nummernfeldkombinationen, NFC-Kartendetails oder Kennzeichen verwaltet. Diese können mit einem zeitlichen Gültigkeitsbereich versehen werden. Darüber hinaus gibt es einen Onlineshop, in welchem das Produkt konfiguriert werden kann, sowie Ersatzteile oder Erweiterungsmodule erworben werden können. Das Dashboard, welches mithilfe von Angular umgesetzt wurde, kommuniziert mit dem Node-JS Server als Backend. Dieser fungiert als Herzstück und reicht die Informationen an die MongoDB-Datenbank weiter und steht für Anfragen des Raspberry Pi, welcher für die Hardware zuständig ist, zur Verfügung.



Abbildung 2: Gerenderter Prototyp

Inhaltsverzeichnis

1 Einleitung	1
1.1 Problemsituation [SK]	1
1.2 Ziele [SK]	2
2 Technologien	3
2.1 Auswahl der Technologie - Client [DH] [BG]	3
2.2 Auswahl der Technologie - Datenbank [SK]	24
2.3 Auswahl der Technologie - Kennzeichenerkennung [SK]	26
2.4 Auswahl der Technologie - Backend [SK]	28
2.5 Auswahl der Technologie - Hardware [SK]	30
3 Lösungsansätze	39
3.1 Profil Management [DH]	39
3.2 Webshop [BG]	40
3.3 Automatic Number Plate Recognition (ANPR) [SK]	41
3.4 Backend	42
3.5 MongoDB	44
4 Systemarchitektur	46
4.1 Übersicht der Systemarchitektur	46
4.2 Frontend (Angular-Applikation) [DH]	46
4.3 Frontend (React-Applikation) [BG]	51
5 Umsetzung	59
5.1 Implementierung des Frontend (Angular) [DH]	59
5.2 Implementierung des Frontend (React) [BG]	68
5.3 Implementierung des Backend [SK]	74
5.4 Implementierung der Kennzeichenerkennung [SK]	76
5.5 Implementierung des Displays [SK]	85

5.6	Implementierung des Relais[SK]	86
5.7	Implementierung des Keypad [DH]	86
5.8	Implementierung des RFID-Kits [BG]	89
6	Umfeldanalyse	91
7	Persönliche Ziele	93
7.1	Projektverlauf	93
7.2	Erkenntnisse von Benjamin Golic	93
7.3	Erkenntnisse von David Hauser	94
7.4	Erkenntnisse von Simon Koll	94
8	Zusammenfassung / Ausblick	96
8.1	Zusammenfassung [DH]	96
8.2	Ausblick [DH]	96
Literaturverzeichnis		VI
Abbildungsverzeichnis		XIII
Tabellenverzeichnis		XV
Quellcodeverzeichnis		XVI
Anhang		XVIII

1 Einleitung

1.1 Problemsituation [SK]

Autos sind aus dieser Welt nicht mehr wegzudenken. Alleine in Österreich sind mehr als 5 Millionen Personenkraftwagen zugelassen. Dieser Bestand wuchs nach Angaben der Statistik Austria seit mehr als 15 Jahren kontinuierlich an.[1] Um die Sicherheit der Insassen gewährleisten zu können, wird empfohlen, das Fahrzeug in einer Garage oder einem überdachten Gebiet abzustellen. Dort sind Umwelteinflüsse wie Hagel keine große Gefahr mehr. Neue Garagen haben oftmals elektrische Tore, die mit einem Nummernfeld oder einem Handsender aus dem Auto selbst geöffnet werden können. Diese Handsender können jedoch bei der Anfahrt an die Garage in der Eile nicht gefunden werden, oder die Batterie kann sich unbemerkt entleeren. Falls zudem kein Nummernfeld oder ähnliches vorhanden ist, gibt es bei Verlust des Handsenders keine Möglichkeit, die Garage zu öffnen. Somit muss aus dem Auto ausgestiegen werden und die Notentriegelung des Systems betätigt werden. Mit der immer weiter voranschreitenden Vernetzung der nahen Umwelt, wie Haustüren mit Fingerabdruck-Zugangskontrolle oder mit dem Smartphone bedienbaren Jalousien, kommt mit APERTA das IOT in die Garage. APERTA ist ein Komplettsystem, welches bei Garagen mit elektrischem Tor nachgerüstet werden kann. Es besteht die Möglichkeit, mithilfe eines Nummernfeldes oder einer NFC-Karte das Garagentor wie gewohnt zu öffnen. Was APERTA aber auszeichnet ist eine integrierte Kennzeichenerkennung, bei der kein Handsender oder ähnliches mehr benötigt wird. Das Kennzeichen wird über das Web-Dashboard eingegeben, und das Tor kann dann bei Annäherung an das Tor dieses erkennen und steuert den Toröffnungsmechanismus an. Dazu wird ein Relais verwendet, welches wie handelsübliche Handschalter an der Innenseite der Garage den Steuerstromkreis der Garage schließt.

1.2 Ziele[SK]

Das Team hat sich vor Entwicklungsbeginn einige Ziele gesteckt, welche das Projekt erfüllen muss, um einen Mehrwert für potentielle Kunden zu bieten. Zu diesen Zielen zählen:

- *Herstellerunabhängigkeit*: APERTA soll bei jedem Garagentor mit bereits verbautem Motor nachrüstbar sein. So können mehr potentielle Käufer angesprochen werden.
- *Modularität*: Aufgrund der vielen Möglichkeiten kann APERTA für manche Käufer in der Vollausstattung nicht geeignet sein. Das Produkt soll daher im Onlineshop konfiguriert werden können, sodass bestimmte Komponenten entfernt werden können. Diese sollen nachträglich eingebaut werden können.
- *Übersichtliche Verwaltung*: Um den Benutzer oder die Benutzerin zu unterstützen, soll die Verwaltung von Nummernfeldkombinationen, NFC-Details und Kennzeichen einfach und schnell funktionieren. Der Käufer soll mithilfe von Texteingaben neue Einträge hinzufügen können und diese mit einem Knopfdruck wieder entfernen können.
- *Intuitiver Bestellvorgang*: Um die Erfahrung für den Käufer von Anfang an gut zu gestalten, soll ein optisch ansprechender Onlineshop der erste Kontakt mit dem Produkt sein.

2 Technologien

2.1 Auswahl der Technologie - Client [DH] [BG]

Da die Kennzeichenverwaltung und der Web-Shop plattformunabhängig eingesetzt werden sollen, ist die Entscheidung auf eine Web-Applikation gefallen.

Eine Web-Applikation ist eine Anwendung, die über das Internet abrufbar ist. Diese wird mittels Client-Server-Architektur umgesetzt. Bei dieser Architektur befinden sich die Daten und die Logikschicht auf einem Server und die Darstellung davon erfolgt im Webbrowser des Benutzenden.

Da Web-Applikationen lediglich einen Web-Browser erfordern, sind sie plattformübergreifend nutzbar und erfordern kein spezielles Betriebssystem. Im Vergleich dazu, werden bei klassischen Softwarelösungen oftmals Betriebssysteme oder spezielle Programme vorausgesetzt, um auf die vollständige Funktionalität zurückgreifen zu können.

Komponente: Webserver

Web-Applikationen werden auf Webservern ausgeführt. Im einfachsten Fall wird nur ein Server genutzt, allerdings handelt es sich in der Praxis meist um mehrere Systeme.

Grundsätzlich wird zwischen zwei Architekturen unterschieden, zum einen „Standalone“ und zum anderen „Integriert“. Bei der „Standalone“ – Architektur ist die Webanwendung ein Skript, das bei jeder Anfrage neu ausgeführt wird. Bei der „Integriert“ – Architektur ist die Webanwendung ein Teil des Webservers.

Für die Datenspeicherung werden Dateien oder Datenbankserver verwendet. Außerdem können benutzerbezogene Daten in Form von Cookies auf dem Clientrechner gespeichert werden.

Komponente: Client

Der Web-Client ist der Web-Browser, der mit dem Webserver mittels des Request-Response-Verfahrens kommuniziert. Hierbei sendet der Web-Browser einen HTTP-

Header mit einem Request an den Webserver, und der Webserver antwortet darauf mit einer HTTP-Response. Ein Request kann beispielsweise die Nachfrage nach einem Webseiteninhalt sein.

Vorteile einer Webapplikation:

- Plattformunabhängig
- Keine zusätzlichen Installationen notwendig
- Wartungsarbeiten finden an der Anwendung auf dem Webserver statt, dies bedeutet, dass die Nutzenden keine Updates installieren müssen, um die neuen Versionen der Web-App nutzen zu können

Nachteile einer Webapplikation:

- - Konstant funktionsfähige Internetverbindung notwendig
- - Da alle Daten auf einem Cloud-System gespeichert werden, sind bei einem Hackerangriff alle Daten in Gefahr

[2] [3] [4]

2.1.1 Unterschied Framework und Library [BG]

Generell verfolgen Frameworks und Libraries das gleiche Ziel, nämlich den Funktionsumfang eines Programmes zu erweitern. Jedoch gibt es grundlegende Unterschiede zwischen den beiden.

Library:

Bei Libraries handelt es sich um Sammlungen von Klassen und Funktionen. Diese können die Entwickelnden in den eigenen Code einbinden. Der Zugriff auf die Funktionen der Library erfolgen über die Programmierschnittstelle (API, kurz für Application Programming Interface). Dieser ist allerdings auf „öffentliche“ Funktionen begrenzt. „Private“ Funktionen arbeiten nur im Hintergrund.

Framework:

Frameworks sind eine spezielle Art von Libraries, die allerdings keine Funktionen enthalten. Frameworks liefern den Bauplan und das Grundgerüst für ein Programm.

[5]

„Das Framework stellt eine wiederverwendbare, gemeinsame Struktur für Anwendungen zur Verfügung. Entwickler binden das Framework in ihre eigenen Anwendungen ein und erweitern es so, dass es ihre bestimmten Anforderungen erfüllt.“

[6]

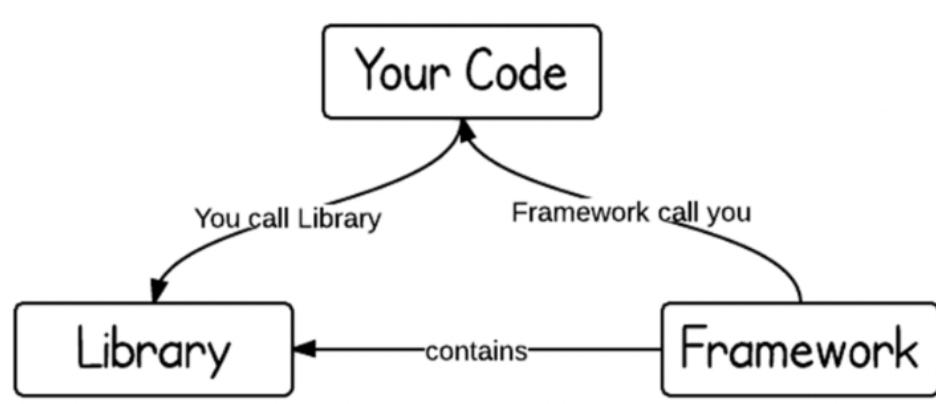


Abbildung 3: Unterschied zwischen der Funktionsweise eines Frameworks und einer Library

[7]

Um das Programmieren einer Web-Applikation zu erleichtern, wird auf JavaScript-Libraries und -Frameworks zurückgegriffen. Hier ist es möglich, zwischen vielen verschiedenen Möglichkeiten zu wählen. Die Wahl des richtigen Werkzeugs hängt von unzähligen Faktoren ab. Beispielsweise achten viele auf die Popularität oder auf die Etablierung. Für manche spielt auch die Unterstützung der Community eine wichtige Rolle bei der Entscheidung.

Die am häufigsten verwendeten JavaScript-Frameworks beziehungsweise -Libraries sind Angular, Vue.js und React. [6]

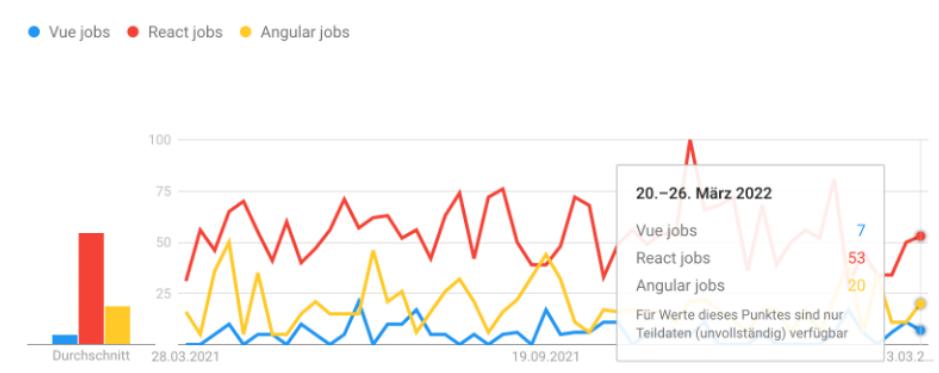


Abbildung 4: Interesse im zeitlichen Verlauf

[8]

Der größte Unterschied zwischen den drei populärsten JavaScript-Frameworks ist die Verwendung des DOMs (Document Object Model). React und Vue verwenden ein virtuelles DOM, wobei hingegen Angular auf das echte DOM setzt. Die Art des DOMs wirkt sich auf die Leistungsfähigkeit aus. Da bei dem virtuellen DOM die Nodes (Nodes-Schnittstellen sind die zentralen Objekte des DOMs) nur bei Bedarf neu geladen werden müssen, ist die Leistung deutlich besser im Vergleich zu echten DOMs.

Ein weiterer sehr großer Unterschied zwischen den Frameworks ist das Architekturnuster. Vue basiert auf dem Architekturnuster MVVM und Angular auf MVC, React hingegen basiert auf gar keinem Architekturnuster. MVC steht für „Model View Controller“ und ist ein beliebtes Architekturnuster in der Programmentwicklung. Hierbei wird das Programm in drei Abschnitte aufgeteilt, in ein Model, eine Ansicht und eine Steuerung.

Das Model wird dazu verwendet, um die Logik der Anwendung zu implementieren. Es besteht aus mehreren Model-Klassen, wobei jede eine grundlegende Einheit innerhalb der verwendeten Datenstruktur verwendet. Des Weiteren stellen diese Klassen die grundlegenden Datenoperationen zur Verfügung.

Die View-Klassen sind für die grafische Benutzeroberfläche zuständig. Sie zeigen die von den Model-Klassen zur Verfügung gestellten Daten an, ohne eine direkte Verbindung aufzubauen.

Der Controller stellt ein Verbindungsstück zwischen den View- und Model-Klassen dar. Dieser ist für die Benutzeraktionen, die von der View weitergeleitet werden, zuständig.

Vorteile von MVC:

- Ein Model kann in mehreren Views dargestellt werden
- Niedrige Kupplung
- Hohe Wiederverwendbarkeit
- Hohe Wertbarkeit

Nachteile von MVC:

- Keine klare Definition
- Zu enge Verbindung zwischen View und Control

[9] [10]

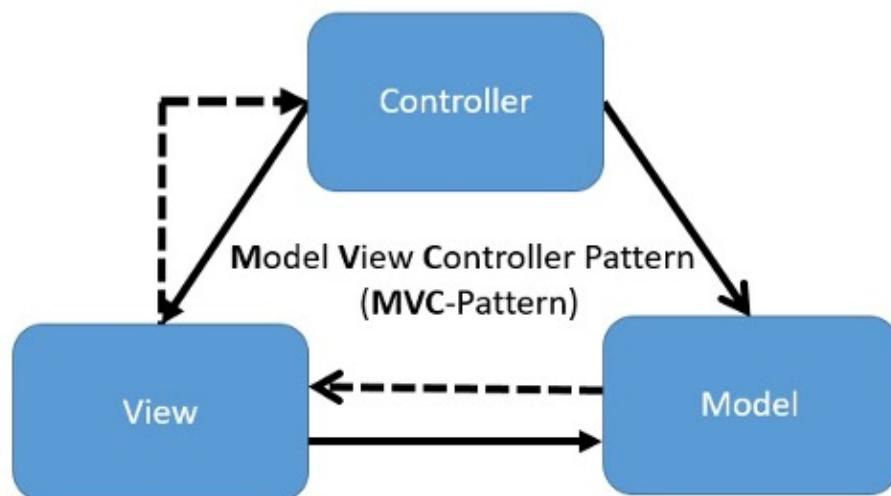


Abbildung 5: Aufbau des MVC-Patterns
[9]

MVVM steht für „Model-View-ViewModel“ und ist ebenfalls ein Software-Architekturmuster. Es besteht aus einem Model, einem ViewModel und einer View.

Wie beim MVC enthalten die Model-Klassen die Daten. Das ViewModel fungiert als Verbindung zwischen Model und View. Es wandelt die Datenobjekte des Models so um, dass die View diese leichter darstellen kann. Ebenfalls wie beim MVC ist die View hier für die Benutzeroberfläche zuständig.

Der konzeptionelle Unterschied zwischen MVC und MVVM liegt darin, dass für MVC der Einstiegspunkt der Anwendung der Controller ist, während für MVVM der Einstiegspunkt die View ist. [10]



Abbildung 6: Aufbau des MVVM-Patterns

[10]

Schlussendlich sind Angular und React für das Frontend ausgewählt worden, wobei Angular für das Frontend der Systemverwaltung und React für den Web-Shop verwendet wurde. Diese Entscheidungen wurden von mehreren Faktoren beeinflusst. Die Hauptfaktoren hierbei waren die persönlichen Erfahrungen und Kompetenzen.

Vorteile von MVVM:

- Logik kann unabhängig von der Darstellung bearbeitet werden
- Nützlich bei größeren Teamarbeiten
- Vereinfachte Testung der Komponenten

Nachteile von MVVM:

- Data-Binding erschwert das Debugging
- Das bidirektionale Data-Binding ist für Wiederverwendung von Code ungünstig

[9] [10]

2.1.2 Technologie zur Entwicklung des Frontends

2.1.3 Angular [DH]

Angular ist eines der großen Frameworks für Single-Page-Webanwendungen. Genauer genommen ist es ein sehr erfolgreiches, clientseitiges JavaScript-Web-Framework zur Erstellung von Single-Page-Webanwendungen. Mittlerweile hat sich Angular schon eher zu einer Plattform weiterentwickelt. Neben der reinen „API“ zur Anwendungsentwicklung beinhaltet Angular auch Entwicklungs-Werkzeuge, Generatoren und mitgelieferte Architektur-Konzepte. Es ist somit eine Ready-to-Rock Lösung, um Enterprise-Anwendungen zu entwickeln.

Angular wurde 2009 auf den Markt gebracht und konnte sich durch ihre Mission bei den vielen anderen Frameworks durchsetzen. Es ist ein wunderbares Ökosystem mit einer großartigen Community entstanden. Der Fokus auf Qualität und Enterprise ist dennoch zu spüren. Nach eigenen Angaben nutzt selbst Google das Framework in über 1600 Projekten.

Das Ökosystem von Angular [DH]

Die Plattform von Angular ist sehr groß. Die Basis dafür bietet das Core-Framework, in welchem fundamentale Konzepte implementiert sind, diese sind für moderne Webanwendungen essenziell. Die Angular CLI und die Verwaltung von Komponenten sind noch zwei weitere Core-Konzepte welche separat genutzt werden können. Sie bilden die Kernfunktion und werden in fast allen Anwendungen benötigt. Einige weitere Module lassen sich noch einbinden:

- Routing – Routing für Single Page Applications
- forms – Formulare und Validierung
- i18n – Mehrsprachige Anwendungen
- Animations – Animationen für Transitionen
- PWA – Offline Fähigkeiten
- HTTP – HTTP, Rest und GraphQL Kommunikation
- und noch unzählige mehr

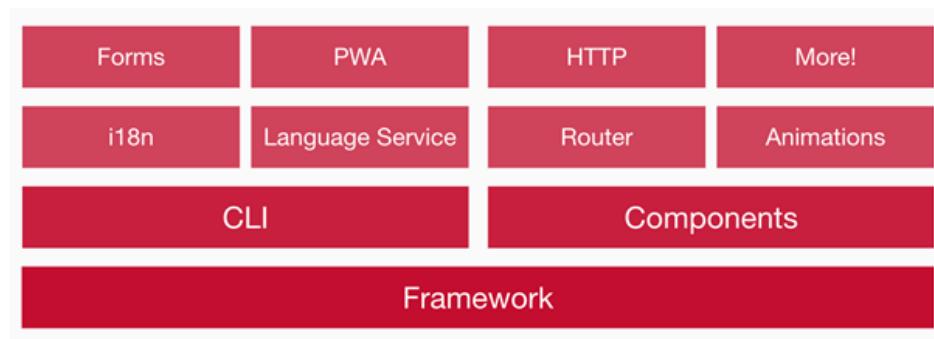


Abbildung 7: Ökosystem von Angular
[12]

Konsistenz Codekonsistenz ist ein wichtiges Ziel, welches bei der Programmierung mit Angular immer anzustreben ist. Das gesamte Framework basiert auf Komponenten und Diensten, welche in ihrem Aufbau immer gleich aussehen. Sie können sich wie ein Baustein vorgestellt werden. Alle Angular-Komponenten führen zu Beginn folgendes aus:

- Import der erforderlichen ES-Module
- Definition eines @Component-Dekorators
- Platzierung von Code in einer Komponentenklasse

What's in a Component Class?



Abbildung 8: Aufbau einer Komponente
[13]

Der Aufbau eines Bausteins ist immer gleich und unabhängig davon, welche Komponente geschrieben wird. Natürlich können einige Aspekte hinzugefügt werden, doch die Gesamtstruktur sieht immer gleich aus. Das sorgt von Anfang an für Konsistenz.

Services sind ein weiterer großer Bereich der Konsistenz in Angular. Diese sind ebenfalls wie Bausteine aufgebaut. In den Konstruktor einer Service-Klasse können alle Abhängigkeiten, die vom Service erfordert werden, eingefügt werden.

```
import { Injectable } from '@angular/core';
import { MyDependency } from './mydependency.service';

@Injectable()
class MyService {
  constructor(private myDependency: MyDependency) {}
}
```

Abbildung 9: Beispielcode einer Service-Klasse

Produktivität

Die Produktivität rückt durch die Konsistenz in den Vordergrund, denn die Entwicklerin oder der Entwickler muss sich keine Gedanken darüber machen, ob er die Dinge auf die „richtige Weise“ macht. Komponenten und Services sehen gleich aus, wiederverwendbarer Anwendungscode wird in Serviceklassen abgelegt und ES-Module organisieren zugehörige Funktionen und ermöglichen, dass der Code self-contained und self-responsible ist. Daten werden mithilfe von Eingabeeigenschaften an Komponenten übergeben und lassen sich durch Ausgabeeigenschaften weitergeben.

Wartbarkeit

Angular hat den Vorteil, dass es von einer großen Community und deren Open-source-Beiträge, regelmäßig erweitert wird. Den größten Teil der Erweiterungen trägt allerdings Google bei. Es ist nie sicher, wie lange ein bestehendes System genutzt werden kann. Außerdem kommt es Angular zugute, dass sich Google über die Auswirkungen von etwaigen Änderungen bewusst ist. Da Google Angular selbst in Projekten verwendet, gibt es die Sicherheit, dass nichts schlagartig verändert oder die Wartung eingestellt wird. Neben der Unterstützung durch das Angular Team, ist auch die beschriebene Konsistenz ein Vorteil für die einfache Wartbarkeit.

Modularität

Angular organisiert den Code in sogenannten „Buckets“. Komponenten, Services, Pipes oder Anweisungen müssen in einem oder mehreren Buckets organisieren. Die Buckets werden in Angular als Module bezeichnet. Sie bieten die Möglichkeit, die Anwendungsfunktionalität zu organisieren und in Funktionen oder wiederverwendbare Abschnitte zu unterteilen. Module bieten außerdem noch andere Vorteile.

Frühzeitige Fehlererkennung

TypeScript ist die Sprache zur Erstellung von Angular, was einige positive Aspekte mit sich bringt:

- TypeScript ist keine eigenständige Sprache und somit eine Obermenge von JavaScript. Es kann vorhandener ES JavaScript Code verwendet werden und der Code funktioniert einwandfrei.
- Außerdem unterstützt TypeScript die wichtigsten ES-Funktionen.
- Typen werden von TypeScript unterstützt, und sind für eine frühzeitige Fehlererkennung enorm wichtig. Dadurch wird leichter erkannt ob etwas falsch übergeben oder verwendet wird.
- TypeScript Code kann direkt über den Browser debuggt werden.
- Bei TypeScript können ebenso Klassen und/oder funktionale Programmiertechniken verwendet werden.

Angular wurde ebenso mit dem Hintergedanken an Testbarkeit entwickelt. Die Angular-CLI macht den Prozess des Komponententests und des End-to-End Tests sehr einfach. Standardmäßig wird Karma und Jasmine dafür verwendet. Zudem können mit dem Befehl „ng test“ alle im Projekt vorhandenen Tests ausgeführt werden.

[14]

Nachteile von Angular

- Es gibt unterschiedliche Vorgehensweisen in Angular, um das gleiche Ziel zu erreichen. Deshalb ist es für den Programmierer oder die Programmiererin oft schwer, sich für den richtigen und effizientesten Weg zu entscheiden.

- Zudem sind die Lebenszyklusmethoden komplex, was sie nur schwer verständlich macht.
- Das Nutzen von Direktiven ist für die Manipulation des DOM sehr hilfreich, jedoch ist die Erstellung nicht gerade einfach.
- Das Erlernen von Angular ist aufgrund von nicht vollständiger und ausreichender Dokumentation etwas schwieriger.

[15]

2.1.4 Angular Materials

Um das Design der Anwendung zu vereinfachen, wurde Angular Materials verwendet. Angular Materials sind moderne, schlichte Designkomponenten, mit denen unter relativ geringem Aufwand optisch ansprechende Anwendungen gestaltet werden können, welche sich auch auf allen Anzeigegeräten mit geringem Zusatzaufwand umsetzen lassen.

[16]

Objekte sollen in einem drei-dimensionalen Raum simuliert und eine natürliche, intuitive Darstellung und Interaktion ermöglichen. Bei der Entwicklung müssen dafür verschiedene Prinzipien beachtet werden.

- Materialien besitzen eine einheitliche Höhe von 1dp, können aber in x- und y-Dimension verschieden sein.
- Der geworfene Schatten von Materialien ist je nach Höhe unterschiedlich.
- Materialien sind für die Darstellung von Inhalten, welche dynamisch veränderbar sind, zuständig. Es kann nur ein Teil von Materialien damit ausgefüllt werden, wobei der Platz durch dessen Dimensionen beschränkt ist.
- Materialien sind Festkörper und können nur an freien Positionen eingesetzt werden. Ihre Transformationen sind auf den freien Platz im Raum beschränkt.
- Sie dürfen ihre Form und Größe jederzeit ändern, wobei die zuvor genannten Einschränkungen zu beachten sind. Die Materialien sind trotzdem als fest anzusehen und dürfen nicht gefalten oder verbogen werden.
- Materialien können sich verbinden oder trennen.

Es werden von Angular Material bereits viele fertige Designideen angeboten. Diese müssen oftmals nur als css-Klasse oder Direktive angefügt werden.

- Theming: Das Farbschemata kann mittels scss-Mixins einfach geändert werden. Um komplexere Änderungen vorzunehmen, können eigene Mixins erstellt werden.
- Elevation helpers: Die z-Positionen können durch css-Klassen oder Mixins einfach dargestellt werden. Es gibt auch ein Mixin für Animationen bei Höhenänderung.
- Typography: font-size, -height und -weight sind von css-Klassen vorgegeben, mithilfe von Mixins können auch diese verändert werden.

```
1  npm install --save @angular/material @angular/cdk @angular/animations
```

Listing 1: Hinzufügen von Angular Materials

Mit diesem Befehl wird Angular Material zum Projekt hinzugefügt. Material: Dieses Standardpaket enthält die Materialkomponenten. CDK (Component Dev Kit): Dieses Paket stellt dem Entwickler, komponentenunabhängige Werkzeuge zur Verfügung. Animations: Dieses Paket wird für erweiterte Animation benötigt.

Theme Das Design von Angular Materials muss durch ein Theme definiert werden. Es kann eines der vorgefertigen verwendet oder auch eine eigenes definiert werden. Zur Auswahl bei den vorgefertigten Themes stehen:

- deeppurple-amber.css
- indigo-pink.css
- pink-bluegrey.css
- purple-green.css

[16]

2.1.5 React [BG]

React, auch bekannt unter React.js oder ReactJS, ist eine Open-Source JavaScript-Library, die zur Realisierung moderner Benutzeroberflächen dient. Die Bibliothek wurde das erste Mal im Jahr 2011 in einem Facebook-Newsfeed verwendet. Im Jahr 2012 wurde sie dann bei Instagram eingesetzt und im darauffolgenden Jahr ist sie als Open-Source Projekt veröffentlicht worden.

Die Grundeigenschaft von React ist, dass es komponentenbasiert ist. Das bedeutet, dass die Anwendung in logisch trennbare Einheiten unterteilt wird, die dann unabhängig voneinander behandelt werden können. Aus diesem Grund eignet sich React besonders gut für Single-Page-Applications. [17]

```

1  function Welcome(props) {
2    return <h1>Hello, {props.name}</h1>;
3  }

```

Listing 2: Möglicher Aufbau einer Komponente in einer JavaScript-Funktion

[17]

```

1  class Welcome extends React.Component {
2    render() {
3      return <h1>Hello, {this.props.name}</h1>;
4    }
5  }

```

Listing 3: Möglicher Aufbau einer Komponente in einer JavaScript-Klasse

[17]

Single-Page-Applications:

Single-Page-Applications bestehen im Gegensatz zu Multi-Page-Applications aus einem HTML-Dokument und können neue beziehungsweise veränderte Daten dynamisch laden. Da dies zu einer client-seitigen Verarbeitung der Anwendung führt, wird die Server-Client Kommunikation verringert.

Das dynamische Laden der Daten bedeutet, dass die Seite nicht aktualisiert beziehungsweise verlassen werden muss, da sich lediglich der Inhalt der aktuellen Ansicht clientseitig verändert.

Da nur eine Abfrage an den Server erfolgt, werden gleich alle Daten übermittelt. Aus diesem Grund ist es möglich, dass Nutzende sich durch die Website navigieren.

[18] [19]

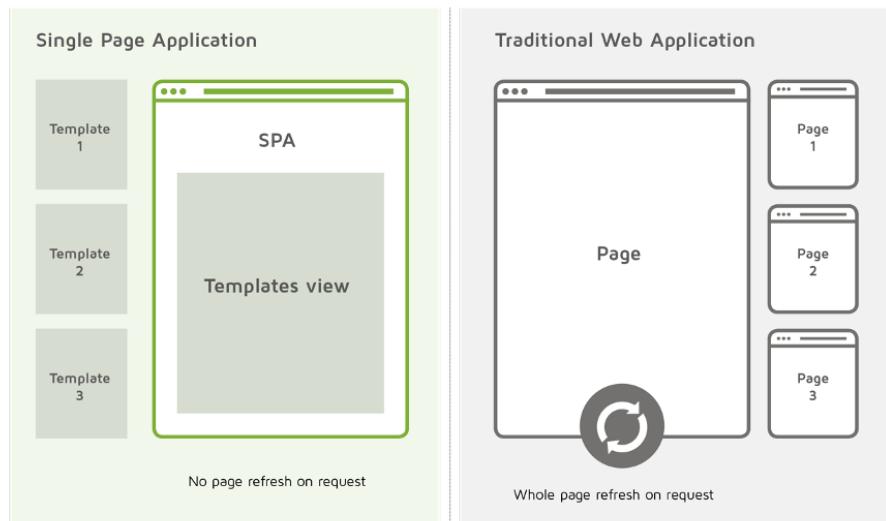


Abbildung 10: Aufbau einer Komponente
[18]

Der HTML-Source-Code einer Komponente wird in JavaScript-Klassen und -Funktionen geschrieben. Der ganze Source-Code wird hierbei mit der Syntaxerweiterung JSX geschrieben, wobei diese Erweiterung hauptsächlich für React entwickelt wurde.

Die Komponenten bestehen intern aus Properties und einem State. Außerdem werden sie in einer Baumstruktur angeordnet. [17]

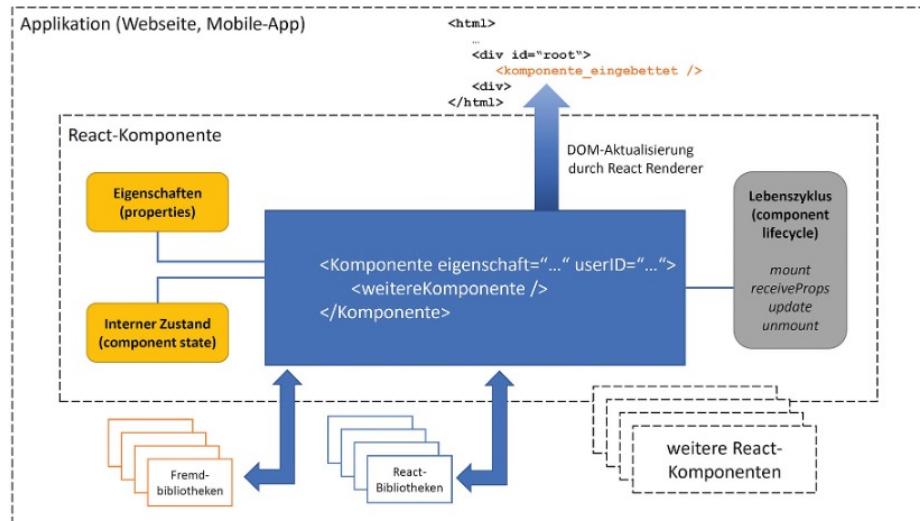


Abbildung 11: Aufbau einer Komponente
[17]

JSX:

JSX steht für JavaScript Syntax Extension beziehungsweise für JavaScript XML. Es ist eine Erweiterung der gewöhnlichen JavaScript-Grammatik für React. Bei der normalen JavaScript Grammatik wird JavaScript in Form von kleineren Scripten innerhalb einer HTML-Datei verwendet. Im Vergleich dazu ist das Ganze bei JSX genau umgekehrt, denn hier wird der HTML Code innerhalb der JavaScript-Klassen beziehungsweise Funktionen verwendet.

Damit der Browser die JSX-Dateien interpretieren kann, ist ein Präprozessor notwendig. Dieser wandelt den JSX-Code in gewöhnliches JavaScript um. Bei der Umwandlung wird aus jedem HTML-Tag in der JSX-Datei ein Element mittels `React.createElement` erzeugt.

[20]

```
1 <MyButton color="blue" shadowSize={2}>
2   Click Me
3 </MyButton>
```

Listing 4: JSX-Code

[20]

```
1 React.createElement(
2   MyButton,
3   {color: "blue", shadowSize: 2},
4   "Click Me"
5 )
```

Listing 5: JSX-Code in gewöhnliches JavaScript kompiliert

[20]

Ein erwähnenswertes Merkmal von JSX ist, dass die Möglichkeit besteht, bestehende React-Komponenten über die Punkt-Notation innerhalb des JSX-Source-Codes aufzurufen. [20]

Props:

Props sind Eigenschaften, die an Komponenten übergeben werden. Diese Eigenschaften erweitern die Funktionalität. Sie können innerhalb der Komponente wie Variablen in Verwendung genommen werden. [21]

```
1 function Welcome(props) {
2   return <h1>Hallo {props.name}</h1>;
3 }
```

Listing 6: Komponente 'Welcome' die Props 'props' erwartet

[21]

```
1 <Welcome name="Sara" />
```

Listing 7: Aufruf der Komponente 'Welcome' mit Übergabe des props 'name'

[21]

```
1 Hallo, Sara
```

Listing 8: Ausgabe der Komponente 'Welcome'

Bei der Verwendung von Props muss darauf geachtet werden, dass diese Read-Only sind. [22]

State:]

Eine weitere Besonderheit von React ist die Verwendung von States. Ein State ist die Beschreibung des Istzustands der Applikation. Beim Aufrufen der Website befindet sich diese in einem Initial State. Alle veränderbaren Daten einer Komponente, die einen Einfluss auf den Zustand der View haben, werden in der State Variable hinterlegt. Sobald sich der State innerhalb einer Komponente verändert, wird die Seite neu gerendert, somit werden die veränderten Daten direkt angezeigt. [19]

```
1 class CountButton extends React.Component {
2   constructor(props) {
3     super(props);
4     this.state = {
5       count: 0,
6       text: "Click Counter"
7     };
8   }
9
10  handleClick() {
11    this.setState({
12      count: this.state.count + 1
13    });
14  }
15
16  render() {
17    return (
18      <div>
19        <h2>{this.state.count}</h2>
20        <button onClick={this.handleClick.bind(this)}>
21          {this.state.text}
22        </button>
23      </div>
24    );
25  }
26}
```

Listing 9: Beispiel Code eines Click Counters mit der Nutzung on States

[23]

Map:

Eine Map ist ein Datentyp, der zusätzlich zum Wert auch einen eindeutigen zugehörigen Key abspeichert. Aufgrund der Einzigartigkeit jedes gespeicherten Schlüssels ist es nicht möglich, dass doppelte Paare in einer Map gespeichert werden. Die Schlüssel dienen zum Nachschlagen der zugehörigen Daten.

Die map()-Methode wird in React hauptsächlich dazu verwendet, um Arrays durchzulaufen beziehungsweise um sie an der grafischen Benutzeroberfläche anzuzeigen. Beim Durchlaufen der Daten wird ein neues Array erstellt, indem die Methode eine bereitgestellte Funktion für jedes Element im aufrufenden Array aufruft. [24]

```

1  const Users = () => {
2    const data = [
3      { id: 1, name: "John Doe" },
4      { id: 2, name: "Victor Wayne" },
5      { id: 3, name: "Jane Doe" },
6    ];
7
8    return (
9      <div className="users">
10        {data.map((user) => (
11          <div className="user">{user}</div>
12        ))}
13      </div>
14    );
15  };

```

Listing 10: Beispiel Code für die Nutzung der map()-Methode

[24]

React ohne JSX:

Es besteht die Möglichkeit, React ohne JSX zu benutzen. Dies ist für Entwickelnde, die keine Kompilierung in der eigenen Entwicklungsumgebung durchführen möchten, empfehlenswert. Da sich jedes JSX Element mittels des Aufrufes von „React.createElement(component, props, ...children)“ aufrufen lässt, ist es möglich alles im einfachen JavaScript umzusetzen. [20]

```

1  class Hello extends React.Component {
2    render() {
3      return <div>Hello {this.props.toWhat}</div>;
4    }
5  }
6
7  ReactDOM.render(
8    <Hello toWhat="World" />,
9    document.getElementById('root')
10   );

```

Listing 11: Beispiel Code 'Hello World' mit JSX

[25]

```

1  class Hello extends React.Component {
2    render() {
3      return React.createElement('div', null, 'Hello ${this.props.
4        toWhat}');
5    }
6  }
7  ReactDOM.render(
8    React.createElement(Hello, {toWhat: 'World'}, null),
9    document.getElementById('root')
10 );

```

Listing 12: 'Hello World' mit einfachem JavaScript

[25]

Des Weiteren besteht die Möglichkeit React mit TypeScript zu benutzen, falls dies dem Anwendenden lieber ist als JavaScript. Hierfür gibt es TSX, dies ist die TypeScript Variante von JSX.

```

1  import * as React from 'react';
2
3  export default class Counter extends React.Component {
4    state = {
5      count: 0
6    };
7
8    increment = () => {
9      this.setState({
10        count: (this.state.count + 1)
11      });
12    };
13
14    decrement = () => {
15      this.setState({
16        count: (this.state.count - 1)
17      });
18    };
19
20    render () {
21      return (
22        <div>
23          <h1>{this.state.count}</h1>
24          <button onClick={this.increment}>Increment</button>
25          <button onClick={this.decrement}>Decrement</button>
26        </div>
27      );
28    }
29  }

```

Listing 13: Beispiel-Code eines Click Counters in TSX label

[26]

Arbeiten im Team mit React:

Da alle Komponenten in einer eigenen Datei gespeichert werden, kann jedes Teammitglied beziehungsweise jedes kleinere Team in einem großen Projekt-Team für eine oder mehrere Komponenten verantwortlich sein. Dies verringert die Gefahr von Merge-Konflikten in Git und sorgt dafür, dass andere Teammitglieder den Code von anderen überschreiben. [22]

Single-Responsibility-Prinzip:

Bei großen React-Projekten wird in den meisten Fällen das Single-Responsibility-Prinzip angewendet. Dies bedeutet, dass die einzelnen Komponenten immer einen einzigen Existenzzweck haben sollen und nur eine Sache gleichzeitig tun sollen. Das Ganze dient dazu, dass die Komponenten wiederverwendet werden, wie zum Beispiel eine Suchleiste. [?]

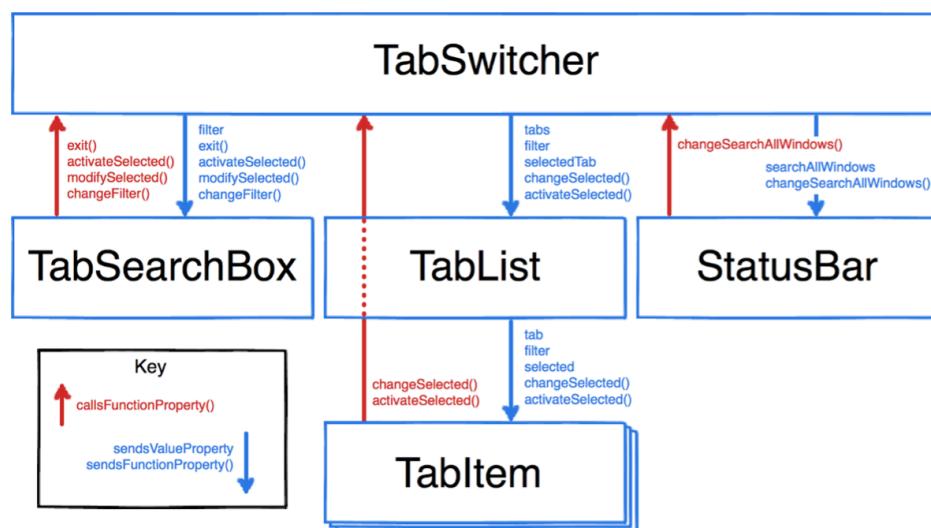


Abbildung 12: Beispiel für die Verwendung des Single-Responsibility-Prinzips

[?]

One Way Data Flow:

React überträgt die Daten nur in eine Richtung zu den Komponenten, und zwar von „oben“ nach „unten“. Es werden nur Events, die von Nutzenden der Website ausgelöst werden, in gegenläufiger Richtung weitergegeben. Durch das Ganze sind Zusammenhänge und Wechselwirkungen einer Anwendung sehr überschaubar. Dies führt dazu, dass das Debugging vereinfacht wird. [27]

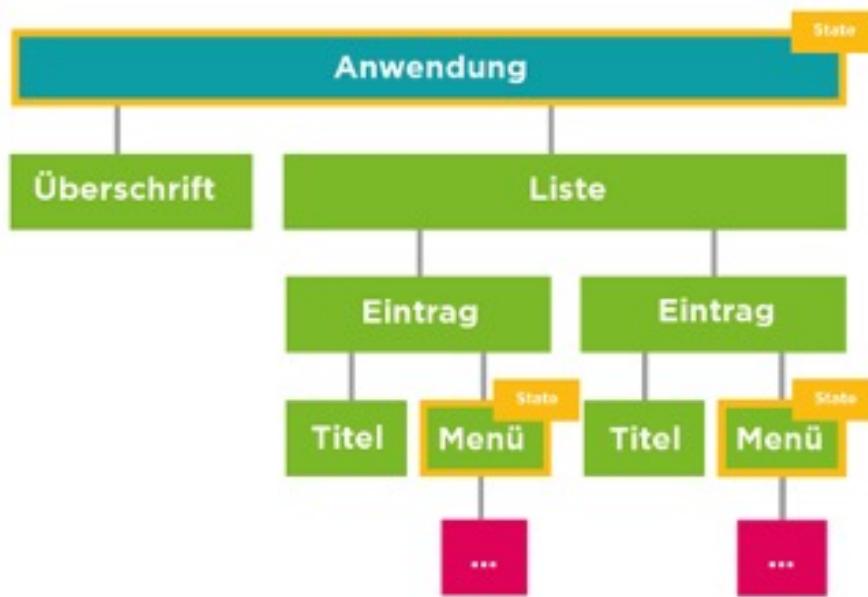


Abbildung 13: Darstellung des 'One Way Data Flow'
[27]

Hooks:

React Hooks bieten die Möglichkeit React-Funktionen, wie zum Beispiel State-Funktionen, in Function-Komponenten zu nutzen, dies war davor nur in Class- Komponenten möglich. Es gibt eine Reihe an unterschiedlichen Hooks.

Der wichtigste und am häufigsten verwendete Hook ist der State Hook. Er bietet die Möglichkeit, die State-Funktionalität in Function-Komponenten zu verwenden.

[28]

```

1 import React, { useState } from 'react';
2
3 function Example() {
4     const [count, setCount] = useState(0);
5
6     return (
7         <div>
8             <p>Du hast mich {count} mal geklickt</p>
9             <button onClick={() => setCount(count + 1)}>
10                 Klick mich
11             </button>
12         </div>
13     );
14 }

```

Listing 14: Code-Beispiel für Nutzung des State Hooks

[28]

Die Funktion 'useState' liefert ein Array zurück, das in zwei Konstanten aufgeteilt wird. Die erste Konstante ist die Repräsentation des aktuellen States. Die zweite Konstante ist eine Funktion, die es ermöglicht, den aktuellen State zu verändern. Die Benennung dieser Konstanten spielt keine Rolle, da es sich um normale JavaScript-Konstanten handelt. Die Funktion 'useState' bekommt den Initial State als Argument übergeben.

Ein weiterer sehr wichtiger Hook ist der Effect Hook. Dieser Hook wird hauptsächlich für Änderungen verwendet, die bisher in Lifecycle-Methoden ausgeführt wurden. Dies bedeutet, dass bestimmte Änderungen durchgeführt werden, nachdem die Benutzeroberfläche gerendert wurde. [28]

```

1 import React, { useState, useEffect } from 'react';
2
3 function Example() {
4     const [count, setCount] = useState(0);
5
6     // Similar to componentDidMount and componentDidUpdate:
7     useEffect(() => {
8         // Update the document title using the browser API
9         document.title = `You clicked ${count} times`;
10    });
11
12     return (
13         <div>
14             <p>You clicked {count} times</p>
15             <button onClick={() => setCount(count + 1)}>
16                 Click me
17             </button>
18         </div>
19     );
20 }

```

Listing 15: Code-Beispiel für Nutzung des Effect Hooks

[29]

Der Funktion 'useState' wird eine Funktion übergeben, die jedem neuen Rendering der Benutzeroberfläche ausgeführt wird.

Weitere Hooks:

Hook	Beschreibung
useReducer	Variation des State Hooks, hier kann der State über einen Reducer verwaltet werden (ähnlich wie bei Redux). Sinnvoll vor allem bei komplexeren State-Objekten, oder wenn der neue State abhängig vom vorherigen ist.
useMemo	Gibt einen memoisierten Wert zur Performance-Optimierung zurück. Sinnvoll für das Caching von Ergebnissen von performance-intensiven Funktionen. Läuft während des Renderings, also nicht geeignet für Side Effects.
useCallback	Variation von useMemo. Gibt ein memoisiertes Callback zurück.
useRef	Gibt ein veränderliches (mutable) Objekt zurück, das während des gesamten Komponenten-Lifecycles bestehen bleibt. Kann in Functional Components ähnlich verwendet werden wie ein Attribut in einer Class Component.
useDebugValue	Kann benutzt werden, um in den DEV-Tools ein eigenes Label für einen Custom Hook anzuzeigen.

Abbildung 14: Übersicht weiterer React-Hooks
[28]

2.2 Auswahl der Technologie - Datenbank[SK]

Die Datenbank spielt für das Projekt eine wichtige Rolle, daher wurden hier folgende Kriterien aufgestellt:

- Das Projekt besteht aus Hard- und Software, die Daten sowohl abspeichern, als auch abfragen. Das kann je nach Anwendungsfall unterschiedlich sein.
- Vom Dashboard können beispielsweise Kennzeichen mit Gültigkeitsdauer, aber auch nur einfache NFC-Codes gesendet werden.
- In der Zukunft soll die Möglichkeit bestehen, weitere Zutrittsmöglichkeit hinzuzufügen. Daher muss sich die Datenbank der sich ändernden Datenstruktur anpassen können.

2.2.1 Relationale Datenbanken

Das Team befand sich vor der Entscheidung, ein relationales Datenbanksystem zu verwenden, oder ein nicht relationales Datenbanksystem. Die größten Unterschiede hierbei sind, dass bei relationalen SQL-Datenbanken den gespeicherten Daten Tabellen

vorgegeben sind, das sogenannte Schema. Das ist bei NoSQL-Datenbanken ebenfalls möglich, jedoch optional. Relationale Datenbanken verfolgen das ACID-Prinzip. ACID steht für

- *Atomicity*
Alle Änderungen der Datenbank werden als einzige Operation verarbeitet. Entweder werden alle Änderungen wie Inserts, Updates usw. durchgeführt, oder keine davon.
- *Consistency*

Zu Beginn und zum Ende jeder Transaktion sind die Daten konsistent. Beispielsweise bei einer Geldüberweisung, ist bei "Consistency" die Gesamtsumme der Geldmittel auf beiden Konten am Anfang und am Ende jeder Transaktion gleich.

- *Isolation*
Andere Transaktionen haben keine Einsicht in die Transaktion. Isolation bedeutet also, dass parallel laufende Transaktionen sich wie serialisierte verhalten.
- *Durability*
Die Daten bleiben nach Ende der Transaktion bestehen und werden auch bei einem kompletten Systemausfall nicht revidiert.

[30]

2.2.2 MongoDB

Das Team entschied sich für eine der bekanntesten NoSQL-Datenbanken, **MongoDB**. Diese bietet einige Vorteile gegenüber den relationalen SQL-Datenbanken:

- *Skalierbarkeit*: MongoDB Datenbanken zeichnen sich durch ihre ausgezeichnete horizontale Skalierbarkeit aus. Horizontale Skalierbarkeit bedeutet, dass die Datenbank sich problemlos über mehrere Server verteilen kann, ohne die Funktionsfähigkeiten zu beeinträchtigen.
- *Verfügbarkeit*: Die Datenbank muss immer zur Verfügung stehen, da Abfrage ausgeführt wird.
- *Flexibilität*: Da das Projekt weiterentwickelt werden kann, muss die Datenbank sich anpassen können.

[31]

2.3 Auswahl der Technologie - Kennzeichenerkennung[SK]

Das Herz von APERTA ist die Kennzeichenerkennung. Dieses Alleinstellungsmerkmal separiert das Projekt von möglicher Konkurrenz. Dabei spielen Faktoren wie der Aufnahmewinkel der Kamera, die Distanz zum Kennzeichen, sowie die Lichtsituation eine entscheidende Rolle. Weiters muss aus den Einzelbildern der Kamera das Kennzeichen erkannt werden und danach die Buchstaben aus dem Bild extrahiert werden. Um dies zu vollbringen, werden 2 Libraries verwendet.

2.3.1 OpenCV:

OpenCV steht für Open Source Computer Vision und ist eine frei Zugängliche Library, welche meist in Bereichen wie Machine Learning oder Machine Vision ihren Einsatz findet. Unternehmen können kostenlos auf die Bibliothek zugreifen, sie verändern und weiterentwickeln. Die Basis bilden die mehr als 2500 klassischen und neuen Algorithmen für das maschinelle Sehen. Diese haben unterschiedliche Spezialisierungen, wie unter anderem die Erkennung von Gesichtern, Objekten und Kamerabewegungen, die Generierung von 3D-Modellen, Ähnlichkeiten in Bildern zu finden, sowie Markierungen in Augmented Reality anzuzeigen. Zu den mehr als 18 Millionen Downloads und mehr als 47.000 Benutzern oder Benutzerinnen zählen meist Unternehmen, Forschungsgruppen oder Regierungsstellen. Bekannte Namen hier sind Google, Microsoft, Intel oder Sony.[32] OpenCV hat eine modulare Struktur, was bedeutet, dass das Paket mehrere gemeinsam genutzte oder statische Bibliotheken enthält. Die folgenden Module sind verfügbar:

- Kernfunktionalität (core) - ein kompaktes Modul, das grundlegende Datenstrukturen definiert, darunter das dichte mehrdimensionale Array Mat und grundlegende Funktionen, die von allen anderen Modulen verwendet werden.
- Bildverarbeitung (imgproc) - ein Bildverarbeitungsmodul, das lineare und nichtlineare Bildfilterung, geometrische Bildtransformationen (Größenänderung, affines und perspektivisches Warping, generisches tabellenbasiertes Remapping), Farbraumkonvertierung, Histogramme usw. umfasst.

- Videoanalyse (Video) - ein Videoanalysemoodul, das Algorithmen zur Bewegungsschätzung, Hintergrundsubtraktion und Objektverfolgung umfasst.
- Kamerakalibrierung und 3D-Rekonstruktion (calib3d) - grundlegende Geometriearithmen für mehrere Ansichten, Einzel- und Stereokamerakalibrierung, Objektposenschätzung, Stereokorrespondenzalgorithmen und Elemente der 3D-Rekonstruktion.
- 2D Features Framework (features2d) - Erkennung auffälliger Merkmale, Deskritoren und Deskriptor-Matcher.
- Objekterkennung (objdetect) - Erkennung von Objekten und Instanzen der vordefinierten Klassen (z. B. Gesichter, Augen, Tassen, Menschen, Autos usw.).
- High-Level-GUI (highgui) - eine leicht zu bedienende Schnittstelle zu einfachen UI-Funktionen.
- Video I/O (videoio) - eine einfach zu bedienende Schnittstelle für Videoaufnahmen und Videocodecs.
- Einige andere Hilfsmodule, wie z.B. FLANN- und Google-Test-Wrapper, Python-Bindings, und andere.

2.3.2 Tesseract:

Tesseract ist eine Texterkennungs-Engine welche von Google entwickelt wird.

Optische Zeichenerkennung oder Optical Character Reading (OCR) ist die elektronische oder mechanische Umwandlung von Bildern mit getipptem, handgeschriebenem oder gedrucktem Text in maschinell kodierten Text, sei es aus einem gescannten Dokument, einem Foto eines Dokuments, einem Szenenfoto (z. B. der Text auf Schildern und Werbetafeln in einem Landschaftsfoto) oder aus einem Untertiteltext, der einem Bild überlagert ist (z. B. aus einer Fernsehsendung). [33]

Um besser zu verstehen, wie OCR funktioniert, hilft dieses Prozessdiagramm in der folgenden Abbildung. Aus Sicht von Endbenutzern oder Endbenutzerinnen ist der OCR-Prozess einfach: Er verarbeitet das Bild und erhält den bearbeitbaren Text.

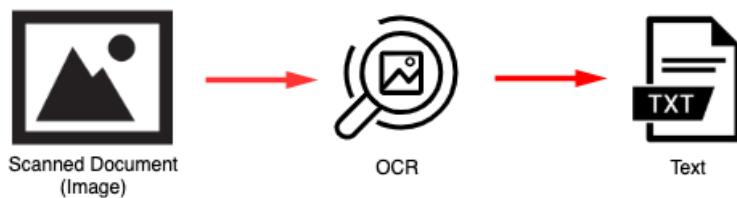


Abbildung 15: Prozessdiagramm der Optische Zeichenerkennung [34]

Tesseract bietet die Möglichkeit, Text aus Bildern zu extrahieren. Dies kann in vielen verschiedenen Programmiersprachen erfolgen, oder über eine graphische Nutzeroberfläche eines Drittanbieters.[35] Unterstützt wird Tesseract durch einen Python-Wrapper mit dem Namen **pytesseract**, welcher Bilder wie .jpeg, .png, .gif und viele mehr laden kann, sowie den gelesenen Text ausgeben kann, anstatt ihn in einer Datei abspeichern zu müssen.[36]

2.4 Auswahl der Technologie - Backend[SK]

2.4.1 Anforderungen an das Backend

Für das Backend kamen mehrere Technologien in Frage, wie unter anderem Java, JavaScript, Python, PHP, C#, und viele mehr. Um das Backend zu realisieren, muss die Technologie einige bestimmte Eigenschaften besitzen.[37]

- *Java:*

Die Vorteile von Java liegen in der Fehlerbehandlung, sowie in Bereichen wie Multithreading und Performanz. Die strikte Fehlerbehandlung führt dabei aber zum Verlust von Flexibilität und Kompaktheit des Codes.

- *JavaScript:*

Die Syntax von JavaScript ähnelt der von Java. Entwickelt als Scripting-Sprache für HTML, ist JavaScript einfach zu lernen und zu benutzen. Bei der Entwicklung von Websites kann JavaScript direkt in den Quellcode der HTML-Seite eingearbeitet werden. Aber auch im Backend-Bereich kann mit NodeJS in JavaScript entwickelt werden.

- *Python:*

Python ist eine der mit Abstand am leichtesten zu lesenden Programmiersprachen. Die flache Hierarchie ermöglicht ein einfaches Verständnis von Programmen und Codestücken. Weiters macht Python Entwickler oder Entwicklerinnen auf Fehler aufmerksam, wenn dieser nicht ausdrücklich ignoriert werden soll. Jedoch ist Python manchmal langsamer in der Ausführung, als die konkurrierenden Sprachen. Zusätzlich ist durch die Verwendung von Leerzeichen zur Einrückung ein häufiger Fehlergrund hinzugekommen.

- *PHP:*

Die PHP Syntax erinnert an eine Mischung aus C, Java und Perl. Das Ziel von PHP ist es, Entwickler oder Entwicklerinnen schnell und einfach dynamisch generierte Webpages bauen zu lassen. Die vermischtte Syntax ist jedoch etwas chaotisch, darum ist es leicht, sich in falschen Angewohnheiten zu verirren und Sicherheitslücken offen zu lassen.

Aufgrund vorhandener Vorkenntnisse standen für das Team 3 der oben genannten Technologien zur Auswahl:

- Java,
- JavaScript und
- Python

2.4.2 Verwendung von NodeJS

Von diesen konnte sich JavaScript durchsetzen. Die Gründe dafür waren:[38]

- NPM: Der NPM oder *Node Package Manager*, ist ein Paketmanager für JavaScript, welcher bei NodeJS standardmäßig mitgeliefert wird. Bei NPM werden wiederverwendbare Programmteile veröffentlicht. Diese können mittels des NPM eingenen Command Line Interfaces installiert werden. Weiters bietet der NPM eine integrierte Versionsverwaltung der Pakete, sowie eine Verwaltung der Abhängigkeiten. In diesem Projekt wurden beispielsweise die Module *express* und *mongodb* verwendet.

express ist ein Framework, welches vor allem in NodeJS Projekten verwendet wird. Die Vorteile von Express sind unter anderem die dem Team bereits bekannte Programmiersprache JavaScript, die Unterstützung der Google V8 engine für bessere Performance, die Robustheit bei einer Vielzahl an HTTP-Anfragen, sowie

die einfache Einbindung weiterer Module und Drittanbieterapplikationen. [39]

mongodb stellt Entwicklern oder Entwicklerinnen eine API zur Verfügung, welche die Nutzung einer MongoDB-Datenbank stark vereinfacht.

- Verwendung einer NoSQL Datenbank: Aufgrund des Formates, mit dem die Daten aus dem Frontend kommen, bat sich eine nicht relationale Datenbank für das Team an. Die dokumentenorientierte Datenbank MongoDB ist bekannt für ihre hohe Verfügbarkeit, sowie für die gute Skalierbarkeit. [40]
- Behandlung von JSON: NodeJS zeichnet sich durch seine einfache Verwendung von JSON-Daten aus. Diese können ohne Parsing oder andere Konvertierungen verarbeitet und darauf zugegriffen werden. Dank NodeJS können JSON Objekte mittels REST-API Anfragen direkt für den Client bereitgestellt werden. Dank NodeJS kann eine einfache Verbindung zwischen Frontend-Clients und dem Backend-Server geschaffen werden.

2.5 Auswahl der Technologie - Hardware[SK]

2.5.1 Anforderungen an die Hardware

Das Projekt sollte so vielseitig wie möglich, jedoch auch so kompakt wie möglich sein. Dazu musste auf kleine Komponenten gesetzt werden. Diese sollen dennoch leistungsfähig genug sein, um jede der drei Zugangsmöglichkeiten parallel zu verwalten.

Raspberry Pi

Die Wahl des Herzstückes fiel auf einen Raspberry Pi. Der Raspberry Pi ist ein vollwertiger Computer, welcher etwas größer als eine Kreditkarte ist. Er besitzt alle bekannten Anschlüsse eines normalgroßen PCs, wie HDMI-Ausgänge für Monitore, USB-Ports für Peripherie wie Maus, Tastatur oder Webcams, sowie einen LAN-Port für eine kabelgebundene Netzwerkverbindung. Als Betriebssystem des Raspberry Pi wurde das vom Hersteller empfohlene Raspbian OS verwendet. Dieses bietet eine grafische Benutzeroberfläche, sowie die Möglichkeit den Raspberry auch ohne angeschlossenen Monitor betreiben zu können. [41]

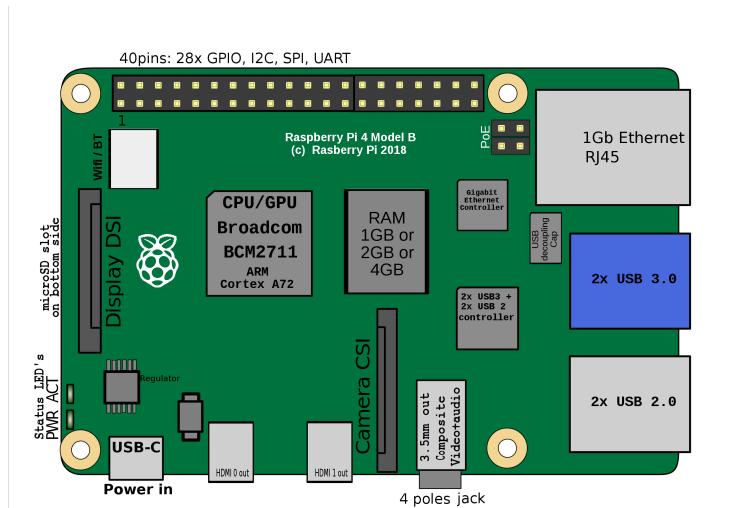


Abbildung 16: Komponenten eines Raspberry Pi
[42]

Auszeichnungsmerkmale des Raspberry Pi sind unter anderem die geringen Anschaffungskosten von ab EUR 35,00, sowie seine leistungsstarken Komponenten.

Tabelle 1: Übersicht der Komponenten des Raspberry Pi [43]

Komponente	Spezifikation	Besonderheiten
Prozessor	Broadcom BCM2711 - Quad Core Prozessor @ 1.5GHz	ARM Architektur 64-Bit SoC
RAM	1GB, 2GB, 4GB oder 8GB LPDDR4 SDRAM	Taktung von 3200MHz
USB	2 USB 3.0 Ports 2 USB 2.0 Ports	
GPIO	40 Pin Header	Abwärtskompatibel mit Vorgängermodellen
Display	2 micro-HDMI Ports	jeweils bis zu 4k60 möglich
Speicher	Micro-SD Kartenslot	Speicherplatz für Betriebssystem und Daten
Strom	5V Eingang über USB-C Port 5V Ausgang über GPIO-Header	Anforderung an Stromquelle: mindestens 3A

Der Raspberry Pi ist einer der am weitest verbreiteten Ein-Platinen-Computer der Welt. Trotz der im Verhältnis zu größeren Systemen schwachen Leistung wurde er im Jahr 2020 mehr als 7 Millionen mal verkauft worden. Daraus ergibt sich ein Marktanteil von allen PCs von 2.69%. Für ein ausgewogenes Verhältnis zwischen Kompaktheit und Leistung wurde auf einen Raspberry Pi 4 Model B in der Ausführung mit 4GB Arbeitsspeicher zurückgegriffen. Weiters waren die Anschaffungskosten von etwa 100\$ ein Grund für die Auswahl. [44]

Kernkomponenten des Raspberry Pi

GPIO-Header

Eine der Kernkomponenten, die den Raspberry Pi von anderen PC-Systemen unterscheidet, ist der "GPIO-Header". GPIO steht für General Purpose Input / Output, und kann wörtlich zu "Allzweckeingabe bzw. -ausgabe" übersetzt werden. Sie bezeichnen selbst programmierbare Ein- und Ausgänge, die auf dem Raspberry Pi als angelötete Pins zur Verfügung stehen. Der Raspberry kann über diese Schnittstellen digitale Signale von außen annehmen, sowie auch Signale abgeben. Der Raspberry Pi in der Ausführung Model 4 B hat einen 40-köpfigen GPIO-Header in Form einer Stiftleiste mit zwei Reihen. Davon gibt es einige GPIOs mit bestimmten Zusatzfunktionen, wie I2C, SPI oder einer seriellen Schnittstellen. Weiters gibt es Pins, welche vom Raspberry eine +5V-Spannung, eine +3.3V Spannung, oder die Möglichkeit der Erdung liefern.

GPIO-Belegung und elektrische Eigenschaften

Grundsätzlich kann in elektronischen Systemen auf elektrische Eigenschaften zurückgegriffen werden, die beobachtet werden müssen. Oft sind diese Eigenschaften Grenzwerte des Systems, welche nicht überschritten werden dürfen. Wird dies ignoriert, kann das System nach Start beschädigt werden und im weiteren Verlauf Defekte aufweisen. Die Eingangsspannung des Raspberry Pi beträgt zwar 5V, jedoch arbeitet der Prozessor selbst nur mit 3.3V. Daher haben auch die GPIOs nur 3.3V zur Verfügung. Dies gilt für die Ausgangsspannung, jedoch auch für die Eingangsspannung, da sonst der Chip des Raspberry Pi beschädigt werden kann. GPIOs sind empfindliche Schnittstellen, denn sie können schon bei geringen Stromstärken Schaden nehmen. Theoretisch ist eine Stromstärke von 16mA (Milliampere) möglich, wobei diese nie benötigt wird, da die GPIOs schon mit einer Stromstärke von 0.5mA geschalten werden können. Um die Langlebigkeit des Raspberry Pi zu gewährleisten, sollten nie mehr als 8mA von einem GPIO abgegeben werden. Es gibt jedoch Ausnahmen, wie die +5V-Pins. Diese bieten für externe Schaltungen eine Spannung bis zu 5V an, sind jedoch ebenfalls bei der Stromentnahme begrenzt. Hier wird mit etwa 25mA pro 5V-Pin gerechnet. Sollte dies für eine Schaltung nicht ausreichen, kann auf externe Stromquellen zurückgegriffen werden, wie eine Stromversorgung über ein separates Netzteil mit USB-Anschluss, oder die Versorgung über einen der verfügbaren USB-Ports des Raspberry Pi selbst.

[46]

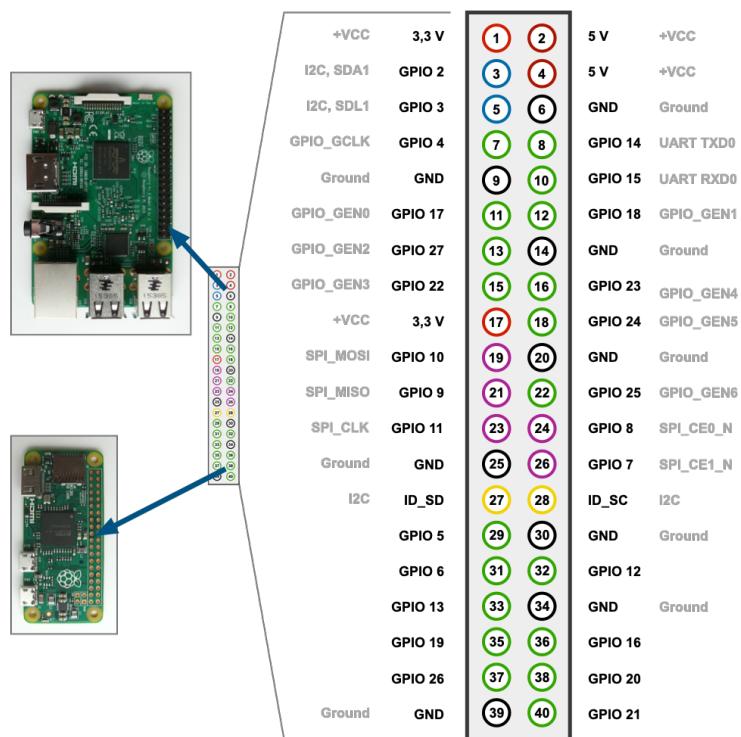


Abbildung 17: Belegung der GPIOs eines Raspberry Pi Model 4B / Raspberry Pi Zero [45]

NFC-Leser

Numpad [DH]

Das Numpad ermöglicht der Benutzerin oder dem Benutzer die Eingabe eines 6-stelligen Zahlencodes. Dazu wird bei APERTA ein übliches Nummernfeld angeschlossen. Der eingegebene Code wird dann geprüft und mit den vorhandenen Codes in der Datenbank abgeglichen. Stimmen die Zahlenkombinationen überein, so öffnet sich das Garagentor. Zur Überprüfung wird der eingegebene Code auf dem LCD angezeigt. So kann sichergestellt werden, dass keine falschen Zahlen eingegeben werden. Sollte eine falsche Kombination vorliegen, dann wird auf dem Display eine Fehlermeldung ausgegeben.

Kamera

Die Kamera ermöglicht dem Raspberry Pi, die Kennzeichen zu sehen und darauf die Kennzeichen zu erkennen. Dazu wird bei APERTA eine handelsübliche Webcam verwendet, die über einen der beiden USB 3.0 Ports am Raspberry angeschlossen wird. Um genug Auflösung für die Kennzeichenerkennung zu gewährleisten, wurde auf eine Webcam zurückgegriffen, welche mit bis zu 1920 Pixeln mal 1080 Pixeln aufnehmen kann. Alternativ wäre auch eine Raspberry Pi Camera möglich gewesen, jedoch wurde

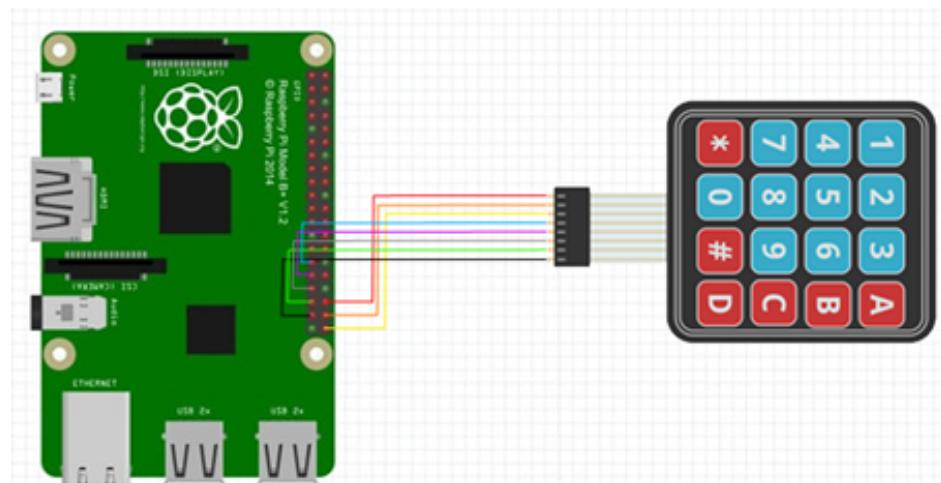


Abbildung 18: Abbildung eines Nummernfelds
[47]

die aufgrund ihres kurzen Flachbandkabels nicht verwendet, um die Kamera auch in größere Entfernung vom Raspberry Pi nutzen zu können.



Abbildung 19: Webcam mit gleichen Spezifikationen
[48]



Abbildung 20: Raspberry Pi Camera Module 2
[49]

Display

Um dem Nutzer oder der Nutzerin vor der Garage mitzuteilen, was gerade geschieht, wird ein Display verwendet, auf dem mitgeteilt wird, ob die Kombination, welche auf dem Nummernfeld eingegeben wurde korrekt ist, oder ob die NFC-Karte autorisiert ist. Da auf diesem Display nur kurze Textausgaben angezeigt werden, fiel die Entscheidung auf ein LCD-Display, welches in zwei Zeilen beschrieben werden kann. Dieses bat zudem weitere Vorteile, wie die geringen Kosten von EUR 9,00 pro Stück, die Spannungsversorgung durch den Raspberry Pi selbst, sowie die einfache Möglichkeit, Text darauf auszugeben. Im Lieferumfang des Displays war zudem ein I2C Serial Adapter, welcher durch seine 4 benötigten Ports um 8 Pins auf dem Raspberry Pi weniger braucht, als das direkt angeschlossene Display. [50]

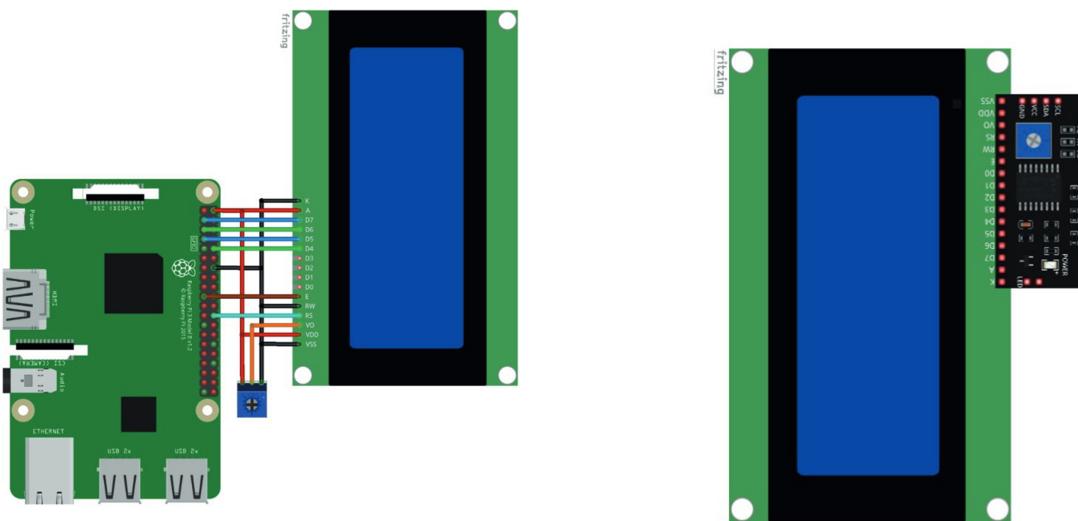


Abbildung 21: Display direkt angeschlossen / Display über I2C Adapter angeschlossen

Die technischen Daten des Displays lauten:

- 4 Zeilen zu je 20 Zeichen beschreibbar
- Blaue Hintergrundbeleuchtung
- 5V Versorgungsspannung

[51]

Relais

Handelsübliche Garagentore werden mit einer Spannung von 230 Volt betrieben. Der Raspberry Pi kann diese nicht direkt ansteuern, da die Spannung die interne Elektronik

des Pi zerstören würde. Dennoch ist es nötig, wie bei einem Schalter den Steuerstromkreis des Garagentores zu schließen, um den Öffnungsmechanismus zu aktivieren. Um dies zu erreichen, wird ein Relais verwendet, welches in den externen Stromkreis geschalten wird und wie eine Brücke den Stromkreis schließen kann. Ein Relais besteht aus einer Spule aus Draht und einem Metallkern. Wird Strom durch den Draht geschickt, wird der Kern magnetisiert. Ohne Strom verschwindet das Magnetfeld des Kerns wieder.

Das Relais ist ein Elektromagnet, welcher durch den Steuerkreis einen Eisenanker zu sich zieht und somit den Arbeitsstromkreis schließt. Der Arbeitsstromkreis kann unabhängig vom Steuerkreis aufgebaut sein und auch unterschiedliche Spannungen und Stromstärken besitzen. Wichtig ist nur, dass das richtige Relais für den Arbeitskreis verwendet wird. Im Fall dieses Projektes wird ein Relais verwendet, welches für bis zu 250 Volt Gleichstrom des Arbeitskreises verwendet werden kann. [52]

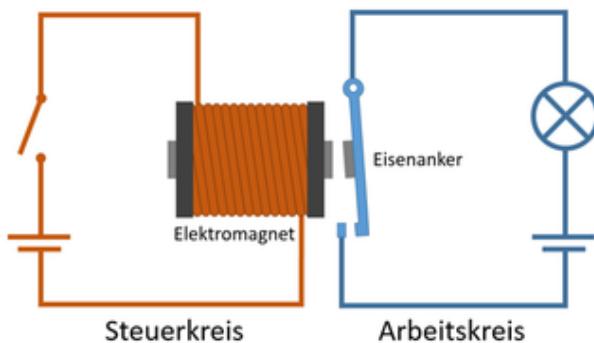


Abbildung 22: Funktionsweise eines Relais als Schema
[53]

Steuerung der Hardware

Um die Komponenten miteinander zu verbinden, war eine Programmiersprache notwendig, die mit den über GPIOs angeschlossenen Bauteilen kommunizieren kann. Dazu gab es eine Handvoll von Programmiersprachen, welche für die Entwicklung von Projekten mit dem Raspberry Pi in Frage kamen.

- *Scratch*: Scratch ist eine baukastenartige Programmiersprache, bei der Befehlsblöcke mithilfe der Maus aneinander angereiht werden. Entwickelt vom MIT Media Lab, richtet sich Scratch an Programmierneulinge und Kinder, welche es unterstützen soll, programmieren zu lernen, Code lesen und verstehen.
- *Python*: Python zählt zu den meistverwendeten Programmiersprachen in Verbindung mit dem Raspberry Pi. Es zeichnet sich durch seine einsteigerfreundliche Syntax und die riesige Community aus, welche es ermöglicht, auf eine Vielzahl

an Frameworks und Libraries zurückzugreifen. Python beschränkt sich dabei nicht auf einen bestimmten Einsatzbereich, sondern kann für die Entwicklung von graphischen Nutzeroberflächen, im Webdevelopment, zum Trainieren von Künstlichen Intelligenzen sowie für Automatisierung verwendet werden.

- *JavaScript*: JavaScript ist nicht nur eine Erweiterung von HTML als Scripting Sprache, sondern viel mehr eine eigenständige, umfangreiche Programmiersprache. Sie wird meistens mit Webentwicklung in Verbindung gebracht, ist jedoch auch fähig, bereits bestehende Applikationen zu erweitern.
- *Java*: Java ist eine der vielseitigsten Programmiersprachen, da sie erlaubt, unabhängig von Betriebssystem zu entwickeln, ohne den Code für jede Plattform verändern zu müssen. Mit mehr als 3 Milliarden Geräten, auf denen Java läuft, ist sie eine der meist verbreiteten Programmiersprachen.
- *C*: C ist einer der stärksten Konkurrenten von Java. Raspbian OS, das Betriebssystem des Raspberry Pi, wurde beispielsweise in C geschrieben. C zeichnet sich durch einen klar strukturierten Programmierstil und die Möglichkeit, Arbeitsspeicher direkt anzusprechen, aus. Die Haupteinsatzgebiete von C sind die Entwicklung von Betriebssystemen und Compilern.
- *C++*: Verglichen mit C, kann sich C++ mit objektorientierter Programmierung auszeichnen. Die Kombination aus prozeduraler und objektorientierter Programmierung machen C++ zu einer Allzwecklösung, mit welcher von Betriebssystemen über Spiele bis hin zu Webbrowsern alles entwickelt werden kann.
- *Perl*: Die Perl Org. hat mit Perl eine Sprache entwickelt, welche für fast jede Aufgabe, die mit C oder C++ Libraries zu tun hat, geeignet ist. Die große Auswahl an Libraries und Modulen sprechen trotz der geringen Bekanntheit für Perl, welche weiters für die Webentwicklung, Systemadministration, GUI-Entwicklung und vielem mehr verwendet werden kann.
- *Erlang*: Erlang ist eine relativ unbekannte Sprache, da sie meist für Industrieapplikationen verwendet wird. Auszeichnungsmerkmale von Erlang sind beispielsweise die Fähigkeit, skalierbare Echtzeitsysteme zu entwickeln. Auch bei dezentralisierten Systemen ist Erlang eine gute Wahl, da das Programm bei Ausfall eines Rechners aus dem Cluster problemlos weiter arbeiten kann. Verwender sind unter anderem Banken und Telekommunikationsunternehmen.

[54]

Durch die einfache Möglichkeit, mit den GPIOs zu arbeiten, sowie der einfach verständlichen Syntax, wurde Python verwendet. Das Team konnte somit zusätzlich auf bereits bestehende Kenntnisse aufbauen.

RFID-Karte und -Leser [BG]

Um den Zutritt zur Garage mittels RFID-Karte beziehungsweise RFID-Chip zu ermöglichen, wird ein RFID-Kit für den Raspberry Pi verwendet. Hierbei wird auf RFID-Kit der Firma AZ-Delivery gesetzt. Dieses Kit bietet den Vorteil, dass es eine eigene Library zur Programmierung der Funktionalität mit sich bringt. Für den geringen Preis von EUR 5,00 enthält das Kit einen RFID-Leser, eine RFID-Karte und einen RFID-Chip. [55]

Eine Alternative zu dem RFID-Kit von AZ-Delivery wäre das Kit der Firma Makersfactory. Der höhere Preis und die geringere Anzahl an Komponenten sprachen gegen dieses Kit. [56]



Abbildung 23: RFID-Kit der Firma AZ-Delivery
[55]

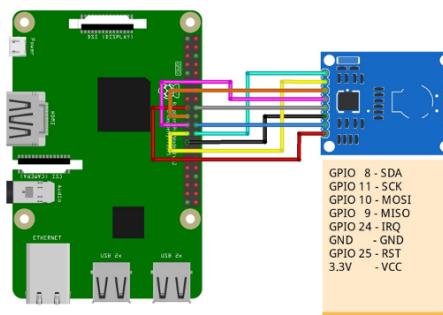


Abbildung 24: Schaltbild des RFID-Lesers
[57]

3 Lösungsansätze

3.1 Profil Management [DH]

Das Profil Management ist ein wichtiger Teil des Projekts. Es verwaltet den Benutzer oder die Benutzerin und die dazugehörigen Produkte. Im Profil Management kann nicht nur der Account verwaltet werden, sondern auch die Kennzeichen, Nummernfelder und NFC-Chips. Über die Bauteile können verschiedene Einstellungen getroffen werden.

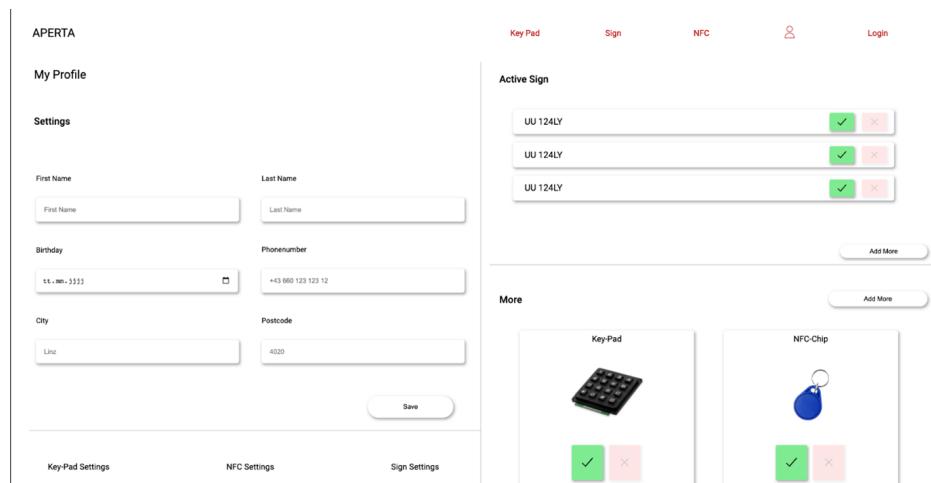


Abbildung 25: Profilmanagement

In den Unterseiten der Bauteileinstellungen haben der User und die Userin noch weitere Möglichkeiten ihre Bauteile zu verwalten. Bauteile können hinzugefügt, verändert oder entfernt werden. Außerdem kann ihr Aktivitätsstatus angepasst werden.

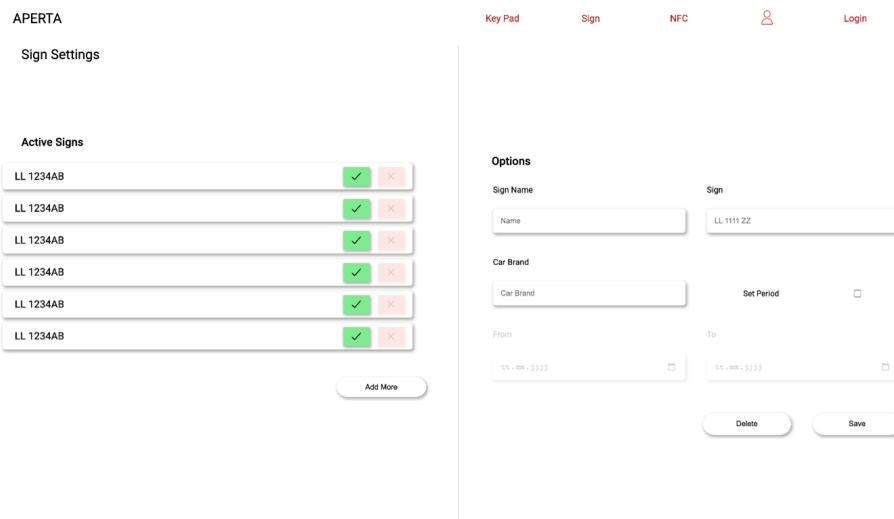


Abbildung 26: Kennzeicheneinstellungen

3.2 Webshop [BG]

Der Webshop dient zum Erwerb des APERTA-Systems. Kundinnen oder Kunden haben die Möglichkeit, Komplettpakete (Packages) und Einzelteile (Components) zu erwerben. Deshalb ist es wichtig, dass der Webshop übersichtlich und intuitiv gestaltet ist.

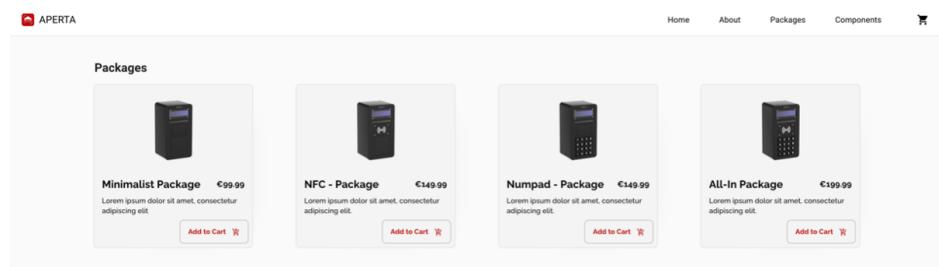


Abbildung 27: Screenshot der „Packages“-Unterseite des Webshops

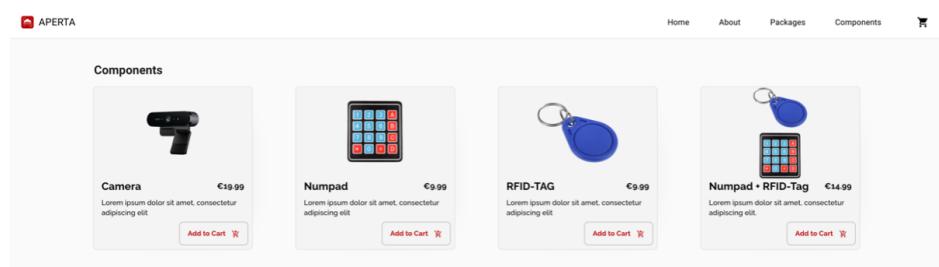


Abbildung 28: Screenshot der „Components“-Unterseite des Webshops

Des Weiteren ist es wichtig, neuen potenziellen Kundinnen oder Kunden unser Produkt zu erklären. Aus diesem Grund verfügt die Website über eine „About“-Unterseite, auf dieser wird das System in Textform und mittels eines Videos erklärt.

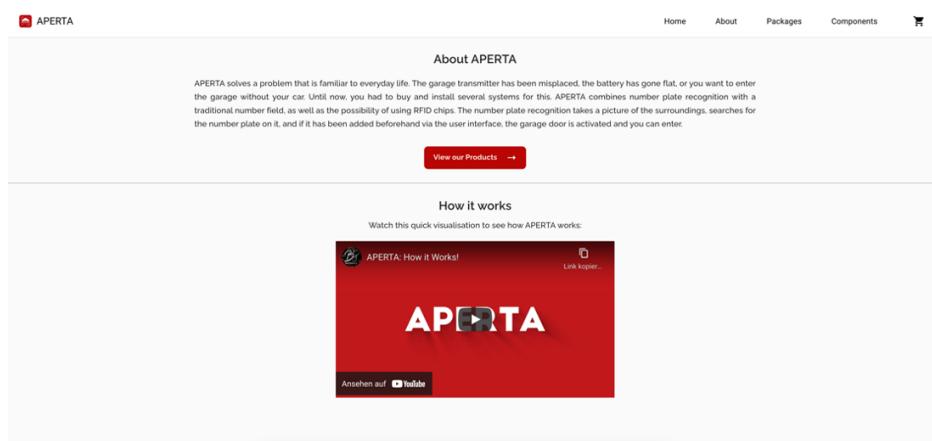


Abbildung 29: Screenshot der „About“-Unterseite des Webshops

3.3 Automatic Number Plate Recognition (ANPR)[SK]

Die Kennzeichenerkennung stellt das Alleinstellungsmerkmal des Projektes dar. Der OpenCV-Python-Code für die Nummernschilderkennung umfasst drei Hauptschritte. Der erste Schritt ist die Erkennung von Nummernschildern. Die Konturfunktion wird verwendet, um die rechteckigen Objekte im Bild zu erkennen und das Nummernschild zu finden. Der zweite Schritt ist die Zeichensegmentierung. Sobald die Kontur des Nummernschildes erkannt wurde, muss es ausgeschnitten und als neues Bild gespeichert werden. Der letzte Schritt ist die Zeichenerkennung. Die optische Zeichenerkennung wird auf dem ausgeschnittenen Bild durchgeführt, um das Kennzeichen zu erkennen.

3.3.1 Überprüfung der Kennzeichen

Die erkannten Buchstaben werden genutzt, um eine Funktion aufzurufen, welche diese mit den zugelassenen Kennzeichen vergleicht. Dazu werden die zugelassenen Kennzeichen vom Server abgefragt, welcher diese aus der Datenbank lädt. Die Abfrage erfolgt über eine REST-Abfrage. REST ist weder ein Protokoll, noch ein Standard. REST steht für REpresentational State Transfer, API für Application Programming Interface, welche gemeinsam eine Programmierschnittstelle bilden, mit der Anwendungen mit einem Server kommunizieren können. Dies wird meist mit dem HTTP-Protokoll genutzt, um Services über URLs zu erreichen. Dazu stehen die HTTP-Methoden GET, POST, PUT und DELETE zur Verfügung.

- *GET*: GET ist eine Methode, welche einen Inhalt eines Servers abruft.

- *POST*: POST ist eine Methode, um vom Client Daten an den Server zu senden, welcher diese weiter verarbeiten und in die Datenbank speichern kann.
- *PUT*: PUT bietet die Möglichkeit, bereits bestehende Daten zu ändern.
- *DELETE*: DELETE bietet die Möglichkeit, bestehende Daten zu löschen.

[58] Neben der Kennzeichenerkennung bietet das Projekt noch 2 weitere Zutrittsmöglichkeiten, nämlich das Nummernfeld sowie die NFC-Funktionalität. Damit alle Zutrittsmöglichkeiten parallel funktionieren, wird mit der Bibliothek *threading* gearbeitet. Hiermit können mehrere Funktionen parallel gestartet werden. Dies kann beispielsweise wie folgt erfolgen:

```

1 import time
2 from threading import Thread
3 def funktion_1():
4     while True:
5         print("Funktion 1")
6         time.sleep(1)
7
8
9 def funktion_2():
10    while True:
11        print("Funktion 2")
12        time.sleep(1)
13
14
15 thread_1 = Thread(target=funktion_1)
16 thread_2 = Thread(target=funktion_2)
17
18 thread_1.start()
19 thread_2.start()
```

Listing 16: Funktionsweise von Multiprocessing

`thread_1` und `thread_2` sind Threads, welche parallel ausgeführt werden. Sie bekommen als Parameter eine Funktion, welche ausgeführt werden soll, in diesem Fall `funktion_1` und `funktion_2`. Mit `start()` werden die Threads gestartet. [59]

3.4 Backend

Der Server bildet das Rückgrat des Projektes. Hier werden die Daten aus der Datenbank abgefragt. Diese werden von einem Client über einen REST-Aufruf abgefragt. Der Server kann zudem die neu zugelassenen Kennzeichen in die Datenbank speichern.

3.4.1 Oracle VM

Der Server läuft auf einer Instanz einer Oracle Virtual Machine, ebenso die MongoDB-Datenbank. Die Oracle VM bietet die Möglichkeit, eine leistungsstarke Basis für den Server sowie für die Datenbank bereitzustellen. Die technischen Daten der Instanz sind:

Tabelle 2: Technische Daten der Oracle VM Instanz

Komponente	Wert
CPU	2 OCPU
RAM	16 GB
Netzwerk	1.4 GBit Bandbreite

Im Vergleich zu anderen Cloudanbietern wie AWS, Microsoft oder Google, verwendet die Oracle Cloud Infrastructure, kurz OCI, anstelle von virtuellen CPUs sogenannte OCPUs. Jede vCPU ist als ein Hyperthread eines Intel Xeon-Kerns definiert. Ein Standard-Intel-Prozessorkern mit aktiviertem Hyperthreading hat 2 Threads.

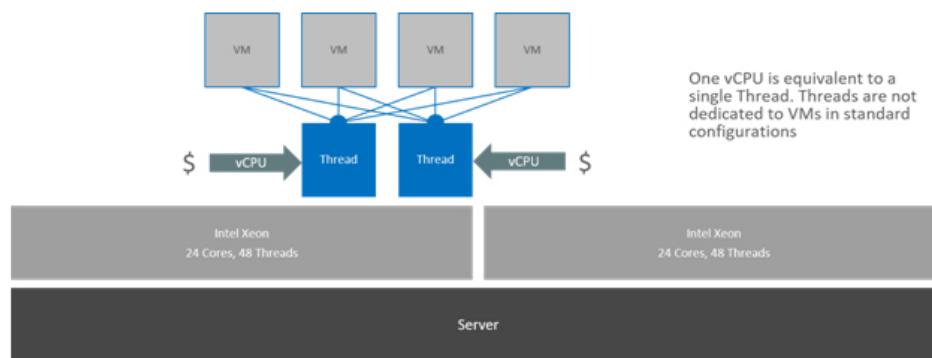


Abbildung 30: Visualisierung von vCPUs
[60]

Eine OCPU ist definiert als die CPU-Kapazität, die einem physischen Kern eines Intel Xeon-Prozessors mit aktiviertem Hyperthreading oder einem physischen Kern eines Oracle SPARC-Prozessors entspricht.

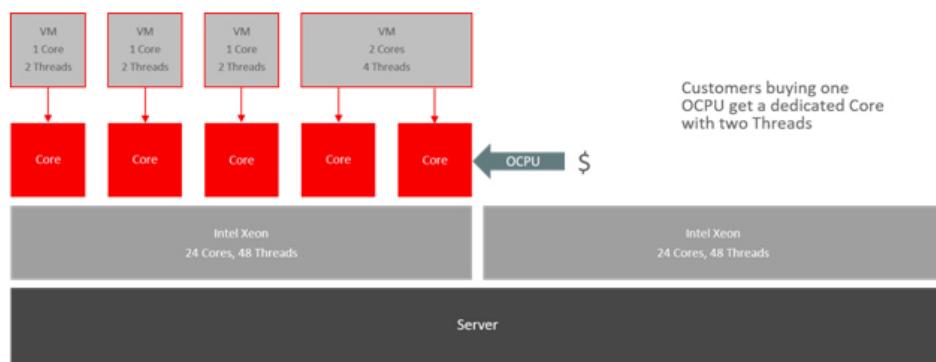


Abbildung 31: Visualisierung von OCPUs
[61]

[62]

3.5 MongoDB

Die Datenbank läuft auf der selben Oracle VM Instanz wie das Backend. Somit gibt es zwischen Server und Datenbank keine zusätzliche Latenz. Da MongoDB eine nicht relationale Datenbank ist, werden die Daten in sogenannten Collections anstelle von Tabellen abgespeichert. Je nach abzuspeichernder Zugangsmöglichkeit, verändert sich das Format der Daten. Für ein Kennzeichen müssen die Daten wie folgt strukturiert sein:

Kennzeichen

```

1   {
2       "_id": "623f449cb8c1b2f4f64c4351",
3       "licenseplate_id": 1,
4       "licenseplate": "RO 108DV",
5       "time_created": 1647252201,
6       "active": true
7   }

```

Listing 17: Aufbau eines Kennzeichen in der Datenbank

Das Attribut `_id` wird automatisch generiert und ist ein eindeutiger Schlüssel, welcher die eindeutige Identifizierung eines Datensatzes darstellt. Mithilfe des `licenseplate_id` wird eine eindeutige Kennzeichenidentifikationsnummer gespeichert. `licenseplate` ist das Kennzeichen, welches vom Client übergeben wurde. `time_created` ist die Zeit, zu der das Kennzeichen erstellt wurde. `active` gibt an, ob das Kennzeichen aktiv ist.

Nummernfeld

Bei einer Nummernfeld-Kombination sieht der Datensatz wie folgt aus:

```
1      {
2          "_id": "622f181d35a3c2ce232acad9",
3          "numpad_id": 1,
4          "numpad_code": "123456",
5          "time_created": 1647252201,
6          "active": true
7      }
```

Listing 18: Aufbau einer Nummernfeld-Kombination in der Datenbank

NFC

Jede NFC-Karte hat eine eindeutige, 10-stellige Kartenidentifikationsnummer. Diese kann mittels eines RFID-Lesers ausgelesen werden. In der Datenbank werden die NFC-Informationen wie folgt gespeichert:

```
1      {
2          "_id": "622f781cfbd4e8de471a4035",
3          "rfid_id": 3,
4          "rfid_code": "01928384756",
5          "time_created": "123456789",
6          "active": true
7      }
8
```

Listing 19: Aufbau einer NFC-Karte in der Datenbank

4 Systemarchitektur

4.1 Übersicht der Systemarchitektur

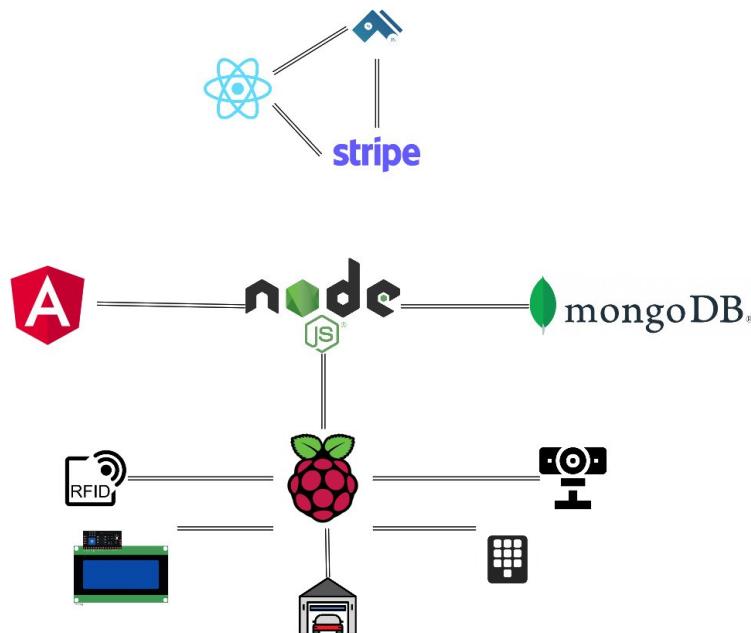


Abbildung 32: Systemarchitektur

Diese Diplomarbeit setzt sich aus zwei voneinander unabhängigen Systemen zusammen. Das Shopsystem, bestehend aus einem React-Frontend, kommuniziert mit zwei Bibliotheken, CommerceJS und Stripe, welche für die Produktverwaltung, sowie den Bezahlvorgang zuständig sind. Das Angular-Frontend des Dashboard in Verbindung mit dem NodeJS Server, der MongoDB Datenbank sowie dem Raspberry bietet das zweite System. Der Raspberry ist über GPIOs mit dem RFID-Leser, dem Nummernfeld und dem Display verbunden. Weiters wurde die Kamera an einen der USB 3.0 Ports des Raspberry angeschlossen. Wird eine der Zutrittsmöglichkeiten positiv abgeschlossen, steuert der Raspberry Pi das Relais an, welches die Garage öffnet.

4.2 Frontend (Angular-Applikation) [DH]

Die Profilseite ist ein wichtiger Teil des webbasierten Clients, jedoch gibt es für die einzelnen Bauteile auch eigene Unterseiten. Auf diesen Seiten hat die Benutzerin oder

der Benutzer noch mehr Möglichkeiten über die Komponenten zu entscheiden. Um die Daten für den Raspberry Pi bereitzustellen zu können, werden noch weitere Funktionen benötigt.

4.2.1 Aufbau von Angular

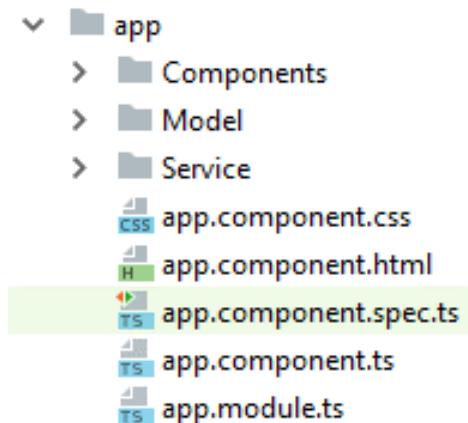


Abbildung 33: Aufbau eines Angular Projekts

Die Applikation ist, wie bei Angular-Projekten üblich, in unterschiedliche Ordner aufgeteilt. Der App-Ordner besitzt mehrere Unterordner, welche ähnliche Funktionen gruppieren.

App

Die Dateien, welche mit app.component beginnen, werden als App bezeichnet. Sie stellen die Hauptkomponente der Angular-Applikation dar. Für das Routing im Projekt könnte noch eine app-routing.module.ts Datei angelegt werden. Wird diese nicht erstellt, kann das Routing auch in der app.module.ts durchgeführt werden.

```

1  @NgModule({
2    declarations: [
3      AppComponent,
4      ProfileComponent,
5      NfcSettingsComponent,
6      KeyPadSettingsComponent,
7      SignSettingsComponent,
8      LoginComponent,
9      NavbarComponent,
10     KeypadComponent,
11     SignComponent
12   ],
13   imports: [
14     BrowserModule,
15     BrowserAnimationsModule,
16     [RouterModule.forRoot(routes)],
17     MatCardModule,
18     ReactiveFormsModule,
  
```

```

19      MatFormFieldModule ,
20      MatInputModule ,
21      MatButtonModule ,
22      MatSelectModule ,
23      MatRadioModule ,
24      MatIconModule ,
25      HttpClientModule
26 ],
27 providers: [],
28 exports: [RouterModule],
29 bootstrap: [AppComponent]
30 })
31 export class AppModule { }

```

Listing 20: app.module.ts

In dieser Datei finden die Imports der Node-Module statt. Zusätzlich werden auch die verwendeten Services importiert. Die Module werden unterteilt, um sie strukturierter zu speichern. Außerdem werden die Angular Materials Module importiert, welche der Programmiererin oder dem Programmierer helfen, die Applikation leichter zu gestalten.

```

1 const routes: Routes = [
2   { path: '', component: LoginComponent },
3   { path: 'profile', component: ProfileComponent },
4   { path: 'nfcSettings', component: NfcSettingsComponent},
5   { path: 'keyPadSettings', component: KeyPadSettingsComponent},
6   { path: 'signSettings', component: SignSettingsComponent},
7   { path: 'addSign', component: AddSignComponent },
8   { path: 'login', component: LoginComponent},
9   { path: 'nav', component: NavbarComponent}
10 ];

```

Listing 21: Routing der Komponenten in der app.module.ts

Mittels [RouterModule.forRoot(routes)] werden die Routen innerhalb des Projekts definiert. Sie navigieren die Userin oder der User durch die einzelnen Views.

Um den Pfad einer Route zu bestimmen wird der Pfadname angegeben. Darauf folgt die Komponente, welche aufgerufen werden soll.

```

1 <app-navbar></app-navbar>
2 <router-outlet></router-outlet>

```

Listing 22: app.component.html

Diese html-Datei ist der Kern der Anwendung. Mittels <app-navbar> wird die Navbar eingebunden, welche dauerhaft am Client angezeigt wird. Im Befehl <router-outlet> werden die Komponenten, die bei den Routen im app.module.ts angegeben sind, angezeigt.

Komponenten

In diesem Ordner sind alle Komponenten auffindbar, die in der Anwendung verwendet werden. Es sind also alle Seiten mit ihren Unterkomponenten, wobei eine Seite eine gesamte Komponente darstellt, zu finden.

Service

Dieser Ordner beinhaltet alle Dienste für die Applikation. Die Funktionsweise und Verwendung wird nachfolgend noch genauer beschrieben.

4.2.2 Angular-Komponente

Die Anzeigeelemente in Angular werden als Komponenten bezeichnet. Diese werden als eigene HTML-Elemente definiert. Sie stellen abhängig von ihrer Anzeige-Logik und den Daten den Zustand der Anwendung dar.

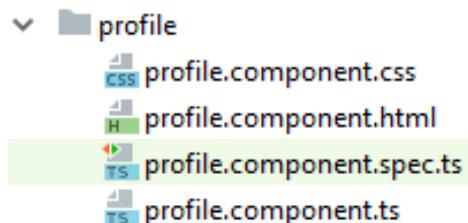


Abbildung 34: Aufbau einer Angular Komponente

Jede Komponente besteht aus:

- HTML
- CSS
- TypeScript
- spec.TypeScript

Wie sich die Komponente verhält und welche Daten sie anzeigt, wird in der TypeScript Klasse definiert. Ihr Aussehen und wo die Daten genau angezeigt werden sollen, wird in der HTML-Datei festgelegt. Ihr Design kann über das CSS-File bestimmt und beeinflusst werden. [63]

4.2.3 Services

Für die Daten und Logik, welche nicht nur in einer Komponente verwendet werden, werden Angular Services genutzt. Darin werden Attribute und Methoden definiert, welche anschließend von anderen Komponenten und Services verwendet werden können.

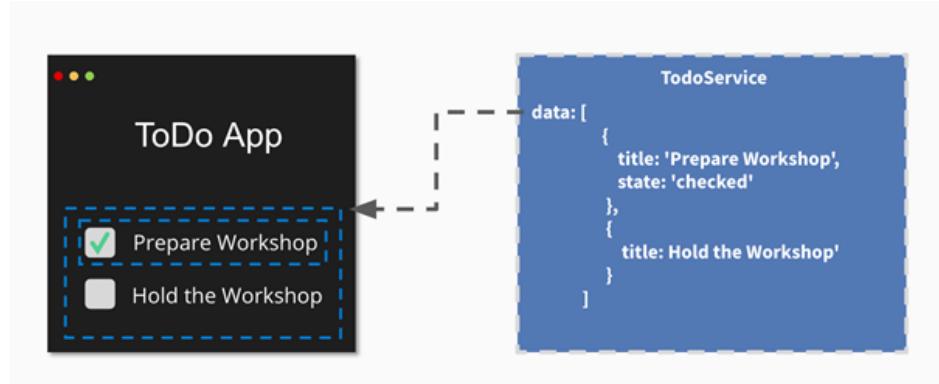


Abbildung 35: Auslagern einer Service Klasse
[64]

Das Service liefert also die eigentlichen Daten für die Komponente. Die Daten gehören nicht in die Komponente. Explizite Aufgaben sollten in einem entsprechenden Service ausgelagert werden. Um die ausgelagerten Daten dann in die Komponente zu bekommen, wird Dependency Injection verwendet.

4.2.4 Dependency Injection

Unter Dependency Injection oder auch Inversion of Control wird ein Design-Pattern verstanden. Die Dependency sollte also nicht von der aufrufenden Stelle selbst erzeugt werden. Letztere sollte die Kontrolle abgeben und nur die Abhängigkeiten definieren.

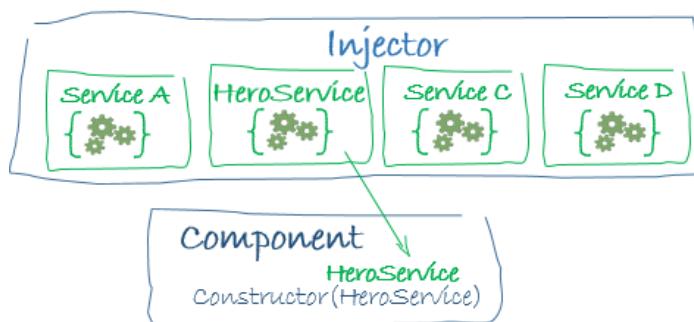


Abbildung 36: Dependency Injection

In Angular werden die verschiedenen Services vom sogenannten Injector verwaltet. Dieser gibt eine Referenz des Service und die aufrufende Stelle an, sofern dieser definiert ist. Über den Konstruktor wird die Definition der Abhängigkeit abgebildet. In einem Konstruktor können dann die Methoden des Service aufgerufen und somit auf die Daten zugegriffen werden. [65]

4.3 Frontend (React-Applikation) [BG]

Die React-Applikation besteht aus mehreren Komponenten.

Applikationsstruktur

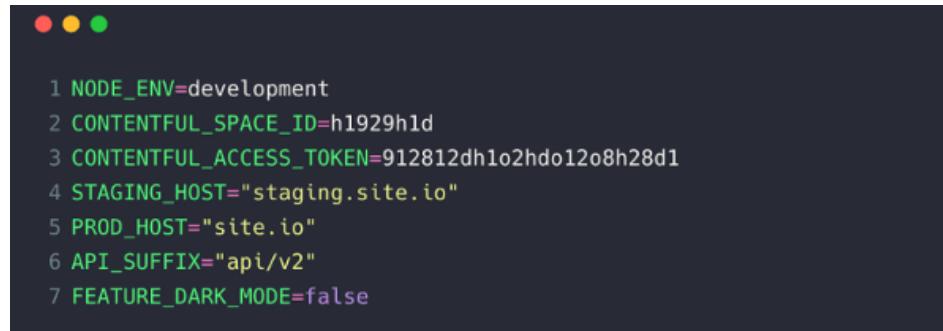
```
aperta-shop
├── node_modules
├── public
└── src
    ├── assets
    ├── components
    ├── lib
    ├── App.js
    └── index.js
├── .env
├── package-lock.json
├── package.json
└── tsconfig.json
```

Abbildung 37: Ordnerstruktur der React-Applikation

Grundsätzlich besteht eine React-App aus Node-Modules, einer TS-Config, einem Package-File und dem Source-Code. Das Package-File dient zur Verwaltung der Projekt-Abhängigkeiten und -Metadaten. Die TS-Config ist für die Kompilieroptionen des Projekts verantwortlich.

Des Weiteren verwenden Entwicklerinnen oder Entwickler eine „.env“-Datei in ihren Projekten. Da diese Datei sensible Daten, wie beispielsweise API-Schlüssel enthal-

ten, werden sie nicht versioniert. Aus diesem Grund müssen die Entwickelnden die Datei lokal gespeichert haben. Die „.env“-Datei ist eine gewöhnliche Textdatei, die Umgebungsvariablen beinhaltet. [66]



```

1 NODE_ENV=development
2 CONTENTFUL_SPACE_ID=h1929h1d
3 CONTENTFUL_ACCESS_TOKEN=912812dh1o2hdo12o8h28d1
4 STAGING_HOST="staging.site.io"
5 PROD_HOST="site.io"
6 API_SUFFIX="api/v2"
7 FEATURE_DARK_MODE=false

```

Abbildung 38: Ordnerstruktur der React-Applikation
[67]

React-Komponente [BG]

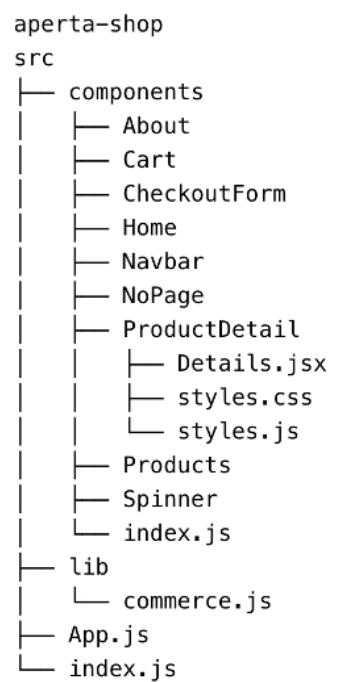


Abbildung 39: Komponenten der React-Applikation und Aufbau der „ProductDetail“-Komponente

Grundsätzlich gibt es keine Vorgaben wie eine React-Komponente aufgebaut sein soll. Es gibt lediglich gängige Ansätze, an die sich die Entwickelnden halten können. Die zwei beliebtesten Ansätze sind die Gruppierung nach Merkmalen oder Routen und die Gruppierung nach dem Dateitypen. [68]

In diesem Projekt bestehen die Komponenten aus einer „.jsx“-Datei, einer „styles.js“-Datei und meistens zusätzlich aus einer „styles.css“-Datei. Die „.jsx“-Datei ist für den Aufbau und der Logik verantwortlich. Die anderen beiden Dateien werden für die Gestaltung verwendet.

Die „App.js“-Datei ist die Haupt-Komponente der React-Applikation. Diese Komponente dient als Container für alle anderen Komponenten. Sie enthält unter anderem das Routing. [69]

Das Routing dient dazu, die Benutzeroberfläche mit dem URL zu synchronisieren. Es legt fest, welcher Inhalt bei welcher URL angezeigt werden soll. Es ermöglicht die Navigierung durch die Web-Applikation, ohne die komplette Seite neu laden zu müssen. [70]

```

1  <Router>
2    <div style={{ display: 'flex' }}>
3      <CssBaseline />
4      <Navbar totalItems={cart.total_items} handleDrawerToggle={handleDrawerToggle} />
5      <Routes>
6        <Route path="/components" element={<Products products={products} onAddToCart={handleAddToCart} handleUpdateCartQty />} />
7        <Route path="/packages" element={<Products products={pproducts} onAddToCart={handleAddToCart} handleUpdateCartQty />} />
8        <Route path="/home" element={<Home products={pproducts} onAddToCart={handleAddToCart} handleUpdateCartQty />} />
9        <Route path="/about" element={<About />} />
10       <Route path="/cart" element={<Cart cart={cart} onUpdateCartQty={handleUpdateCartQty} onRemoveFromCart={handleRemoveFromCart} onEmptyCart={handleEmptyCart} />} />
11       <Route exact path="/checkout" element={<Checkout cart={cart} order={order} onCaptureCheckout={handleCaptureCheckout} error={errorMessage} />} />
12       <Route exact path="/" element={<Home products={pproducts} onAddToCart={handleAddToCart} handleUpdateCartQty />} />
13       <Route exact path="/detail/:id" element={<Details onAddToCart={handleAddToCart} handleUpdateCartQty />} />
14       <Route path="*" element={<NoPage />} />
15     </Routes>
16   </div>
17 </Router>
```

Listing 23: Source-Code des Routers

Gestaltung der Benutzeroberfläche

Für die Gestaltung der Benutzeroberfläche kamen CSS und JSS zum Einsatz.

JSS ist eine JavaScript-Library, die es Entwickelnden ermöglicht, Stile auf konfliktfreie und wiederverwendbare Weise zu schreiben. Diese Bibliothek kann sowohl client- als auch serverseitig kompiliert werden. [71]

```

1 import jss from "jss";
2 import preset from "jss-preset-default";
3
4 jss.setup(preset());
5
6 const styles = {
7   wrapper: {
8     padding: 40,
9     background: "#f7df1e",
10    textAlign: "center"
11  },
12  title: {
13    font: {
14      size: 40,
15      weight: 900
16    },
17    color: "#24292e"
18  },
19  link: {
20    color: "#24292e",
21    "&:hover": {
22      opacity: 0.5
23    }
24  }
25};

```

Listing 24: Beispiel Source-Code für die Verwendung von JSS

[72]

Da Webapplikationen auch auf mobilen Geräten verfügbar sein sollen, muss die Benutzeroberfläche an unterschiedlichste Bildschirmgrößen angepasst werden. Hierbei wird von „Responsive Design“ gesprochen. Um dies zu realisieren, können Entwicklerinnen oder Entwickler auf bestimmte CSS-Libraries zurückgreifen. Allerdings bietet klassisches CSS „Flexbox“ und „Grid“ Möglichkeiten für die Realisierung eines Responsive Designs. Bei diesen Möglichkeiten handelt es sich um zwei signifikant unterschiedliche Layout-Technologien.

CSS-Flexbox

Dieses Modell bietet sich besonders für lineare Strukturen an, da es mit zwei Achsen, auf denen Inhalte verteilt werden können, arbeitet. Grundsätzlich wird hier mit einem Flex-Container und den darin enthaltenen Kind-Elementen (Flex-Items) gearbeitet.

[73]

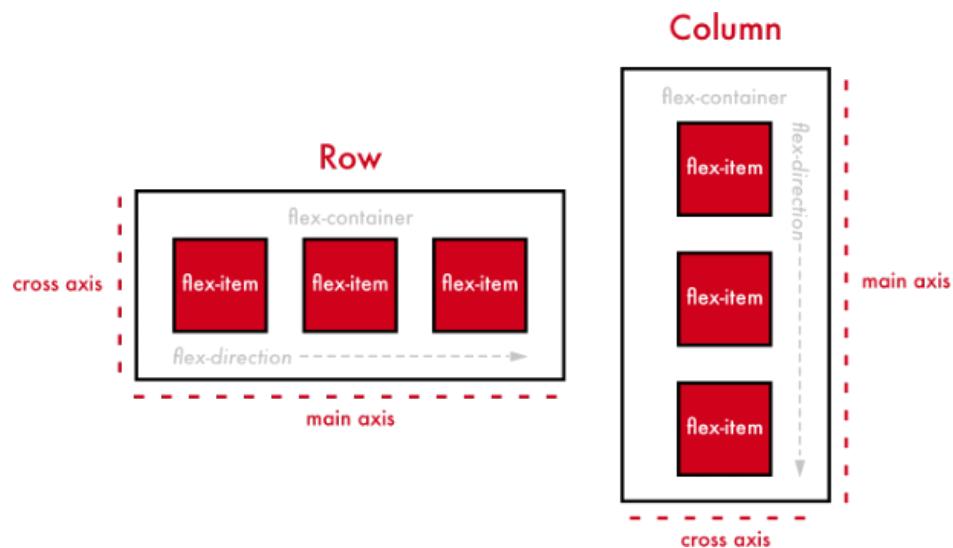


Abbildung 40: Darstellung der Flex-Direction
[74]

```

1 <div class="flex-container">
2   <div class="flex-item"></div>
3   <div class="flex-item"></div>
4   <div class="flex-item"></div>
5   <div class="flex-item"></div>
6 </div>
```

Listing 25: HTML-Code vom Aufbau einer Flexbox

[73]

Um die Flexbox auf dem Container-Element zu aktivieren, müssen wir dem „flex-container“ im CSS-Code die Eigenschaft „display:flex“ deklarieren. [73]

```

1 .flex-container {
2   display: flex;
3 }
```

Listing 26: CSS-Code zur Aktivierung der Flexbox

[73]

CSS-Grid

Mit diesem Modell werden Zeilen und Spalten für einen Bereich definiert, für welche dann Elemente zugewiesen werden. Grundsätzlich wird hier mit einem Elternelement gearbeitet, in dem das Raster definiert wird. Die darin enthaltenen Kind-Elemente werden im Raster positioniert. [75]

```

1  <div class="container">
2    <div>Element 1</div>
3    <div>Element 2</div>
4    <div>Element 11</div>
5    <div>Element 12</div>
6  </div>

```

Listing 27: HTML-Code vom Aufbau eines Grids

[75]

Dem Elternelement wird mit Hilfe der CSS-Eigenschaft „display:grid“ mitgeteilt, dass CSS-Grids verwendet werden.

```

1 .container {
2   display: grid;
3   grid-template-rows: 200px 1fr 100px;
4   grid-template-columns: 25% 25% 25% 25%;
5 }

```

Listing 28: CSS-Code zur Aktivierung des Grids

[75]

Mit den Eigenschaften „grid-template-columns“ und „grid-template-rows“ werden Rasterlinien festgelegt. [75]

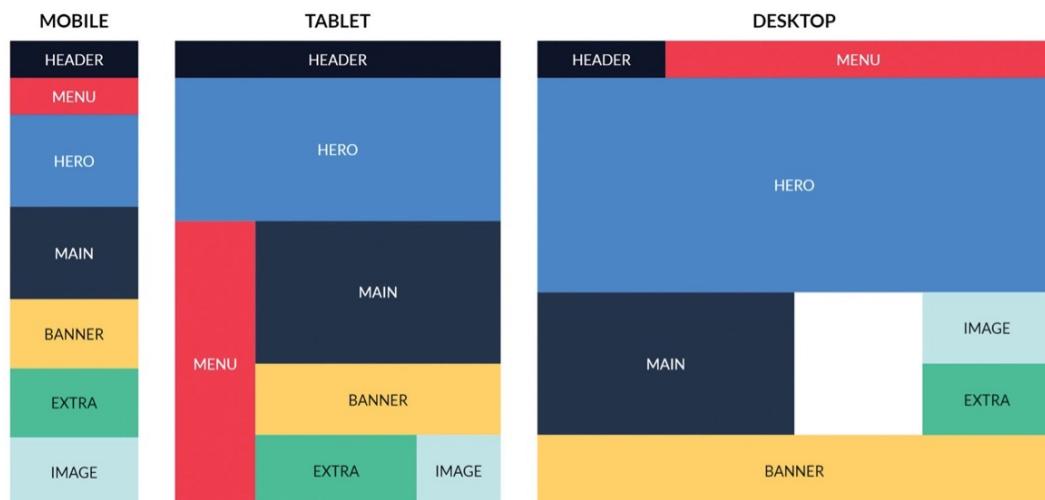


Abbildung 41: Darstellung eines Responsive-Designs durch CSS-Grids

[76]

Commerce.js

Commerce.js ist ein Headless-E-Commerce-Backend, welches die Integration von E-Commerce in jedes Web- und Mobil-Projekt beschleunigt. Die JavaScript-SDK und API-Suite wurden mit Fokus auf Entwickelnde entwickelt. Außerdem ermöglicht es Entwickelnden, dasselbe E-Commerce-Backend für eine Web-App und einer mobilen Anwendung zu verwenden. [77]

Mit Commerce.js ist es möglich eigene Produkte mit beliebigen Produktinformationen hinzuzufügen.

Produkte					+ HINZUFÜGEN
PRODUKT	VERBLEIBEND	BESTELLUNGEN	VERKÄUFE	OPTIONEN	
All-In Package	0	0	0,00 €	...	
Numpad - Package	0	0	0,00 €	...	
NFC - Package	0	0	0,00 €	...	
Minimalist Package	0	0	0,00 €	...	
Numpad + RFID-Tag	0	0	0,00 €	...	
RFID-TAG	0	0	0,00 €	...	
Numpad	0	0	0,00 €	...	
Camera	0	0	0,00 €	...	

Abbildung 42: APERTA-Produkte in Commerce.js

Da die Nutzung des Verwaltungs-Dashboards keinen Programm-Code erfordert, ermöglicht es Content-Autoren, Fulfillment-Teams und Geschäftsinhabern die Verwaltung von Produkten, Kunden und Bestellungen. [77]

Stripe

Stripe ist eine Online-Zahlungsabwicklungs- und Kreditkartenverarbeitungsplattform für Unternehmen. Diese Plattform ermöglicht eine sichere und effiziente Verarbeitung von Geldern per Kreditkarte oder Bank und überweist diese Gelder auf das Konto des Verkaufenden. Stripe umfasst sowohl eine Plattform für die Zahlungsabwicklung als auch ein Gateway für Kreditkartenzahlungen. Zusätzlich zu einmaligen Bezahlvorgängen bietet Stripe auch die Möglichkeit für Abonnements.

Bei einer korrekten Integration werden die vom Kaufenden eingetragenen Informationen beim Bezahlungsvorgang vom Web-Shop über die Stripe-Software gesendet. Diese überprüft, ob das zu zahlende Geld verfügbar ist und verarbeitet die Zahlung, bevor sie an das Händlerkonto gesendet wird. Nach einem erfolgreichem Zahlungsprozess erhalten Kaufende und Verkaufende eine Verkaufsbestätigung.

Falls bei einem Zahlungsvorgang etwas schieflaufen sollte, beispielsweise durch unzureichende Deckung auf dem Konto des Kaufenden, wird die Transaktion durch Stripe abgelehnt. Des Weiteren wird der Kaufende darüber benachrichtigt und bekommt die Möglichkeit, eine andere Zahlungsmethode zu verwenden.

Darüber hinaus überwacht Stripe auch betrügerische Transaktionen und blockiert alle verdächtig erscheinenden Transaktionen.

Bei Stripe ist man nicht auf eine Zahlungsart beschränkt. Es gibt eine große Auswahlmöglichkeit an viele gängigen Bezahlungsmethoden. Außerdem kann man mehrere davon aktivieren.

Des Weiteren liefert Stripe den Verkaufenden jede Menge an Statistiken und Berichten über die getätigten Verkäufe.

[78]

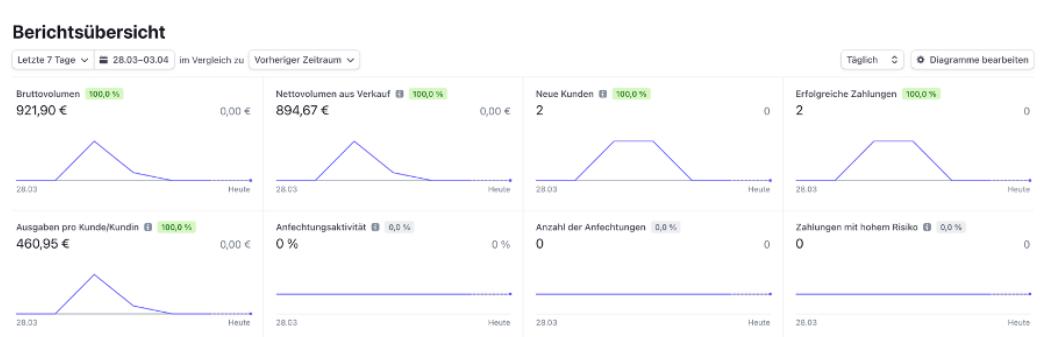


Abbildung 43: Stripe-Berichtsübersicht

5 Umsetzung

5.1 Implementierung des Frontend (Angular) [DH]

Das Frontend, genauer gesagt das Profilmanagement, wurde in Angular entwickelt. Die einzelnen Seiten sind eigene Komponenten. Die Navigationsleiste ist die einzige Komponente, welche dauerhaft in der Webanwendung zu sehen ist.

```
1      <div class="content">
2        <div class="leftSide">
3          <div class="logo">
4            <h1>APERTA</h1>
5          </div>
6        </div>
7        <div class="pageList">
8          <div class="page"><a href="keyPadSettings">Key Pad</a></div>
9          <div class="page"><a href="signSettings">Sign</a></div>
10         <div class="page"><a href="nfcSettings">NFC</a></div>
11         <div class="page"><a href="profile"> </a></div>
12         <div class="page"><a href="login">Login</a></div>
13       </div>
14     </div>
```

Listing 29: Navigationsleiste Komponente

In den Div-Containern „pageList“ befinden sich die Verlinkungen auf die einzelnen Komponenten. Mittels Tag werden die Links zu den Komponenten angegeben. Diese werden dann durch das im app.module.ts angegebene Routing in die View geladen.

DIV Container

DIV ist eine Abkürzung für den Begriff „division“ (=Bereich). Sie dienen zur Gestaltung der Website. Ein DIV-Container ist im Grunde genommen ein leerer Rahmen und begrenzt einen Bereich. [79]

<a>-Tag

Der Anchor Tag ist einer der vielen HTML-Tags. Er definiert den Anfangs- und Endpunkt eines Hyperlinks, welcher eine Website oder die Unterseiten in einem Projekt miteinander verlinkt. [80]

```

1   <div class="container">
2     <div class="divMainContent">
3       <div class="divLeftContent">
4         <h1>My Profile</h1>
5         <h2>Settings</h2>
6
7         <div class="Form">
8           <form class="formProfileSettings" name="formProfileSettings"
9             "gt;
10            <div class="divNameSettings">
11              <div class="Firstname">
12                <label name="lbFirstName">First Name</label>
13                <input name="txFirstName" type="text" placeholder="First Name">
14              </div>
15              <div class="Lastname">
16                <label name="lbLastName">Last Name</label>
17                <input name="txLastName" type="text" placeholder="Last Name">
18              </div>
19            </div>
20            <div class="divPersonalSettings">
21              <div class="Birthday">
22                <label name="lbBirthday">Birthday</label>
23                <input name="dtBirthday" type="date">
24              </div>
25              <div class="Phonenumber">
26                <label name="lbPhonenumber">Phonenumber</label>
27                <input name="nbPhonenumber" type="tel" placeholder="+43 660 123 123 12">
28              </div>
29            </div>
30            <div class="divPersonalSettings">
31              <div class="City">
32                <label name="lbCity">City</label>
33                <input name="txCity" type="text" placeholder="Linz">
34              </div>
35              <div class="Postcode">
36                <label name="lbPostcode">Postcode</label>
37                <input name="nbPostcode" type="number" placeholder="4020">
38              </div>
39            </div>
40
41            <div class="divButton">
42              <button class="btSubscribe" name="btSubscribe">Save</
43                button>
44            </div>
45            </form>
46            <hr class="horizontalLine">
47          </div>
48          <div class="divMoreSettings">
49            <a href="/keyPadSettings">Key-Pad Settings</a>
50            <a href="#">NFC Settings</a>
51            <a href="/signSettings">Sign Settings</a>
52          </div>
53        <div class="verticalLine"></div>
54        <div class="divRightContent">
55          <div class="ActiveSignh2"><h2>Active Sign</h2></div>
56          <div class="divActiveSigns">

```

```

57      <app-sign *ngFor="let a of fakeArray; let index = index" [licenseName]="licenseName"></app-sign>
58
59      </div>
60      <div class="divButton">
61          <button class="btSubscribe" name="AddMore"><a href="/addSign">Add More</a></button>
62      </div>
63
64      <hr class="horizontalLineRight">
65      <div class="Moreh2">
66          <h2>More</h2>
67          <div class="divButton">
68              <button class="btSubscribe" name="AddMore">Add More</button>
69          </div>
70      </div>
71      <div class="divMore">
72          <div class="Components">
73              <div class="ComponentsName">Key-Pad</div>
74              <div class="ComponentPic">
75                  
76              </div>
77              <div class="MoreCheckbox">
78                  <div class="CheckActive">
79                      
80                  </div>
81                  <div class="CheckInactive">
82                      
83                  </div>
84              </div>
85          </div>
86          <div class="Components">
87              <div class="ComponentsName">NFC-Chip</div>
88              <div class="ComponentPic">
89                  
90              </div>
91              <div class="MoreCheckbox">
92                  <div class="CheckActive">
93                      
94                  </div>
95                  <div class="CheckInactive">
96                      
97                  </div>
98              </div>
99          </div>
100     </div>
101   </div>
102 </div>
103 </div>
```

Listing 30: Profil Komponente

In der Profil-Komponente werden die Bauteile, die im Besitz des Users oder der Userin sind, angezeigt. Darüber können im Profil die Entscheidungen getroffen werden. Die Komponenten werden in den Tags `<app-sign>`, `<app-keypad>` und `<app-nfc>` geladen und dargestellt. Über die `<input>`-Tags kann der Benutzer oder die Benutzerin das

Konto mit persönlichen Informationen erweitern. Die Input Felder befinden sich in einem <form>-Tag. Die DIV-Boxen „Checkactive“ und „CheckInactive“ dienen dazu, ob ein Bauteil zurzeit aktiviert oder deaktiviert ist.

<form>

Der form-Tag beinhaltet die Felder des Formulars und ist für die Verarbeitung der Eingaben der Nutzerin oder des Nutzer, mittels action und method zuständig. [81]

<input>

Der Tag wird genutzt, um ein Eingabeelement zu definieren. Er ermöglicht der Nutzerin oder dem Nutzer die Eingabe von Daten. [82]

<Komponenten Tag>

Um eine Komponente von Angular einzubinden, wird der Komponenten Tag verwendet.

*ngFor

Eine For-Schleife wird verwendet, um den Code mehrmals zu durchzulaufen oder um Daten anzuzeigen. Beim ngfor wird ein Array, Element für Element durchlaufen und gibt sie als HTML-Syntax aus. Die for-Schleife wird in das Element geschrieben, welches mehrmals wiederholt werden soll. Im Falle der Profil-Seite wird es genutzt, um mehrere Kennzeichen auszugeben. Es werden immer die 3 aktuellsten Kennzeichen ausgegeben. [83]

Binding

Es gibt verschiedene Wege für das Binding in Angular. Sie erlauben die Kommunikation zwischen HTML und TypeScript. In der Profil-Komponente wurde das Property Binding verwendet. Durch das Property Binding wird eine Variable mit einer DOM-Property, HTML Attributen, Direktive oder Komponente Properties verbunden. Die Kommunikationsrichtung dabei ist von TypeScript-Datei zu HTML-Datei. [84]

```

1     export class KeyPad {
2       Id : number = 0;
3       numpad_id : number = 0;
4       numpad_code : String = "";
5       active: boolean = false;
6   }

```

Listing 31: KeyPad Model

Das KeyPad Model ist ein Beispiel für alle Models in unserem Projekt. In einer Modelklasse befinden sich alle Variablen, die auch in der Datenbank vorhanden sind. Somit kann ein Datenbankeintrag ganz einfach in eine Komponente gebracht und verarbeitet werden.

```

1      <section class="user">
2        <div class="user_options-container">
3          <div class="user_options-text">
4            <div class="user_options-unregistered">
5              <h2 class="user_unregistered-title">Don't have an account yet?</h2>
6              <button class="user_unregistered-signup" id="signup-button">Sign up</button>
7            </div>
8
9            <div class="user_options-registered">
10           <h2 class="user_registered-title">Already have an account?</h2>
11           <button class="user_registered-login" id="login-button">Login</button>
12         </div>
13       </div>
14
15      <div class="user_options-forms" id="user_options-forms">
16        <div class="user_forms-login">
17          <h2 class="forms_title">Login</h2>
18          <form class="forms_form">
19            <fieldset class="forms_fieldset">
20              <div class="forms_field">
21                <input type="email" placeholder="Email" class="forms_field-input" required autofocus />
22              </div>
23              <div class="forms_field">
24                <input type="password" placeholder="Password" class="forms_field-input" required />
25              </div>
26            </fieldset>
27            <div class="forms_buttons">
28              <button type="button" class="forms_buttons-forgot">Forgot password?</button>
29              <input type="submit" value="Log In" class="forms_buttons-action"/>
30            </div>
31          </form>
32        </div>
33        <div class="user_forms-signup">
34          <h2 class="forms_title">Sign Up</h2>
35          <form class="forms_form">
36            <fieldset class="forms_fieldset">
37              <div class="forms_field">

```

```

38          <input type="text" placeholder="Full Name" class="forms_field-input" required />
39      </div>
40      <div class="forms_field">
41          <input type="email" placeholder="Email" class="forms_field-input" required />
42      </div>
43      <div class="forms_field">
44          <input type="password" placeholder="Password" class="forms_field-input" required />
45      </div>
46      </fieldset>
47      <div class="forms_buttons">
48          <input type="submit" value="Sign up" class="forms_buttons-action">
49      </div>
50  </form>
51 </div>
52 </div>
53 </div>
54 </section>

```

Listing 32: Login Komponente

Die Login-Komponente stellt in unserem Projekt die Seite dar, auf der sich angemeldet oder registriert werden kann. Sie verwendet einfache Input Felder für die Eingabe der Daten. Diese werden dann an die Datenbank weitergeleitet und verarbeitet.

<section>

Der section-Tag stellt eine Unterteilung des Dokuments dar. Etwa eine thematische Gruppierung des Inhaltes, typischerweise mit einer Überschrift. Die section sollte identifizierbar sein. [85]

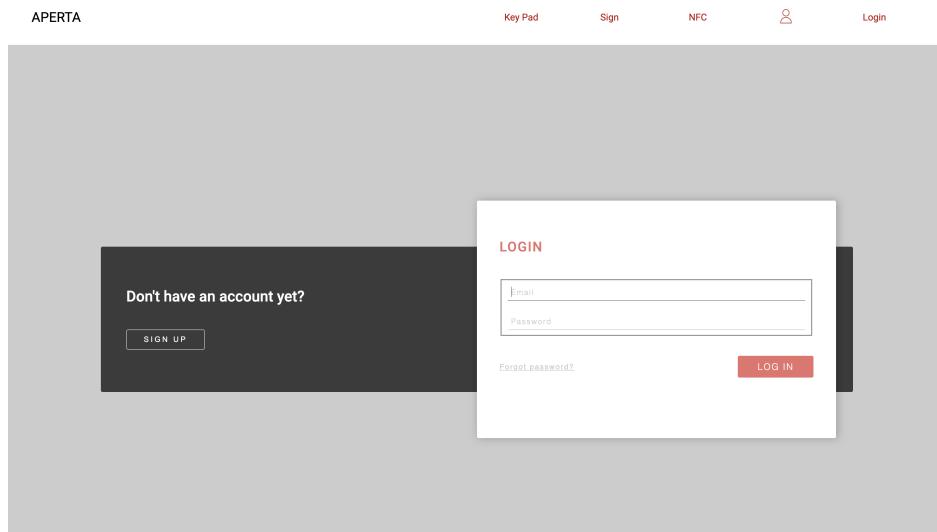


Abbildung 44: Loginseite

```

1   import { Input } from '@angular/core';
2   import { Component, OnInit } from '@angular/core';
3   import { Sign } from 'src/app/Model/sign';
4
5   import {HttpService} from "../../Service/http.service";
6   import { SignComponent } from '../sign/sign.component';
7
8   @Component({
9     selector: 'app-profile',
10    templateUrl: './profile.component.html',
11    styleUrls: ['./profile.component.css']
12 })
13 export class ProfileComponent implements OnInit {
14
15   Signs: Sign[] | undefined;
16   licensename : String = "";
17   i: number = 0;
18
19   constructor(private httpService: HttpService) { }
20
21   ngOnInit(): void {
22     this.getSigns();
23   }
24
25   fakeArray = new Array(3);
26
27
28   getSigns(){
29     this.httpService.getSigns().subscribe((data:Sign[])=>{
30       this.Signs = data;
31       console.log(data);
32       this.getSignName();
33     })
34   }
35
36   getSignName() {
37     while (this.i <= 2) {
38       this.licensenname = this.Signs![this.i].licenseplate;
39       console.log(this.licensenname);
40       this.i++;
41     }
42   }
43 }
44
45 }
```

Listing 33: profile.component.ts

Die Profil-Komponente verarbeitet die Daten für die Kennzeichen. Mit der Funktion `getSigns()` werden die Daten vom http-service verarbeitet. Es werden die gesamten Daten in einem Array von Kennzeichen gespeichert. Mit der Methode `getSignName()` wird der Array, mithilfe einer while-Schleife durchlaufen, dabei werden die Strings der Kennzeichen auf eine Variable gespeichert. Diese werden dann, in der Webapplikation dargestellt.

HttpService

Mit `this.httpService` kann auf die Service Klasse zugegriffen werden. Diese stellt die gängigsten HTTP-Methoden zur Verfügung. Somit kann in der Komponente auf die Funktion `getSigns()`, welche im http-Service definiert ist, zugegriffen und ihre Werte verarbeitet werden. [86]

```

1      import { Injectable } from '@angular/core';
2  import { HttpClient} from "@angular/common/http";
3  import {Sign} from "../Model/sign";
4  import { KeyPad } from '../Model/key-pad';
5  import { Nfc } from '../Model/nfc';
6
7
8
9  @Injectable({
10    providedIn: 'root'
11  })
12  export class HttpService {
13    baseUrl = "http://130.162.215.116";
14
15    constructor(private http: HttpClient) { }
16
17    getSigns(){
18      var licenseplate = this.http.get<Sign[]>(this.baseUrl + "/get-
19      licenseplates")
20      return licenseplate
21    }
22
23    getKeyPads(){
24      var keyPad = this.http.get<[]>(this.baseUrl + "/get-numpad-
25      codes")
26      return keyPad;
27    }
28
29    getNFC(){
30      var nfc = this.http.get<[]>(this.baseUrl + "/get-rfid-codes")
31      return nfc;
32    }
33
34    createNumpad(keypad: KeyPad){
35      return this.http.post<KeyPad>(this.baseUrl + "/add-numpad",
36      keypad)
37    }
38
39    createLicenseplate(licenseplate: Sign){
40      return this.http.post<KeyPad>(this.baseUrl + "/add-licenseplate
41      ", licenseplate)
42    }
43
44    deleteNumpad(numpad: KeyPad){
45      return this.http.delete<KeyPad>(this.baseUrl + "/delete-numpad"
46      )
47    }
48
49    deleteLicenseplate(licenseplate: Sign){
50      return this.http.delete<KeyPad>(this.baseUrl + "/delete-
51      licenseplate")

```

```

50      }
51
52      deleteRfid(nfc: Nfc){
53          return this.http.delete<KeyPad>(this.baseUrl + "/delete-rfid")
54      }
55  }

```

Listing 34: http.service.ts

Die baseUrl ist die Referenz zur Datenbank. Über diesen Link ist die Datenbank erreichbar. Mit den richtigen Endungen stellt sie einige Funktionen zur Verarbeitung der Einträge in der Datenbank zur Verfügung.

Die Funktionen `getSigns()`, `getKeyPads()` und `getNFC()` sind dazu da, um die Datenbankeinträge zu laden. Die Einträge der Datenbank kommen im return-Wert als JSON-Array zurück. Danach können sie in den Komponenten weiterverarbeitet werden. Die Funktionen `createNumpad()`, `createLicensplate()` und `createRfid()` erzeugen einen neuen Eintrag in der Datenbank. Sie bekommen jeweils ein Element mitgegeben, das die Variablen aus der Datenbank belegt hat. Die Variablen werden von den Komponenten erzeugt. `deleteNumpad()`, `deleteLicenseplate()` und `deleteRfid()` löschen jeweils den gewünschten Eintrag. Dazu bekommen sie das zu löschen Element als Parameter übergeben.

HttpClient

Unsere Front-End-Anwendung kommuniziert, so wie die meisten, über das HTTP-Protokoll mit einem Server. Dadurch werden Daten herunter- bzw. hochgeladen und auf andere Back-End-Dienste zurückgegriffen. Für Angular-Anwendungen wird eine Client-HTTP-API, die HTTP-Client Dienstklasse angeboten.

Dieser Dienst bietet folgende Hauptfunktionen:

- Es können typisierte Antwortobjekte angefordert werden
- Eine optimierte Fehlerbehandlung
- Testability features
- Anfragen und Antworten können abgefangen werden

Um den HttpClient verwenden zu können, muss das `HttpClientModule` importiert werden. Anschließend kann dieser als Abhängigkeit verwendet werden. Für Transaktionen werden `Observables` verwendet.

Fällt der Server aus, gibt der HttpClient ein Fehler-Objekt anstelle der erfolgreichen Antwort zurück. Die Fehlerprüfung, -interpretation und -behebung sollten von dem Dienst durchgeführt werden, der die Servertransaktion durchführt. Tritt ein Fehler auf, können Details zum Fehler abgerufen werden. Mit den Details können die Benutzer oder Benutzerinnen informiert werden.

Die Rückmeldung sollte für Nutzerinnen oder Nutzer möglichst nützlich sein. Das reine Fehlerobjekt ist als Rückmeldung nicht besonders nützlich. Es muss nicht nur erkannt werden, dass ein Fehler aufgetreten ist, die Fehlerdetails müssen dazu genutzt werden eine benutzerfreundliche Antwort zu verfassen. Mögliche Fehler die auftreten können:

- Der Server könnte die Anfrage ablehnen, dann würde ein Fehlercode zurückgegeben werden.
- Es kann auch etwas auf der Clientseite schief gehen. Eine Anfrage könnte nicht erfolgreich abgeschlossen werden aufgrund eines Netzwerkfehlers.

Der Service erfasst beide Arten der Fehler, die Antworten müssen untersucht werden und die Ursache des Fehlers gefunden werden. [87]

5.2 Implementierung des Frontend (React) [BG]

5.2.1 Implementierung von Commerce.js

Um die, in Commerce.js gespeicherten, Produkte im Webshop anzeigen zu können, ist eine API-Datenabfrage erforderlich. Die Commerce.js eigene Library ermöglicht dies ganz einfach mit der Methode „commerce.products.list()“. Die Abfrage sieht somit wie folgt aus:

```

1  const [products, setProducts] = useState([]);
2  const [pproducts, setPproducts] = useState([]);
3
4  const fetchComponents = async() => {
5      const { data: CProducts } = await commerce.products.list({
6          category_slug: ['components'], });
7      setProducts(CProducts);
8
9  const fetchPackages = async() => {
10     const { data: PProducts } = await commerce.products.list({
11         category_slug: ['packages'], });
12     setPproducts(PProducts);
13 }
```

Hier werden die Daten nach Kategorie gruppiert abgerufen und in States gespeichert. Damit die potenziellen Käuferinnen oder Käufer die Produkte auf der Weboberfläche sehen können, benötigen wir folgenden Source-Code:

```

1  const Product = ({ product, onAddToCart }) => {
2      const classes = useStyles();
3      const handleAddToCart = () => onAddToCart(product.id, 1);
4
5      return (

```

```

1      <div className="card">
2          <Link className="product-link" to={`/detail/${product.id}`}
3              >
4                  <img className="card__img" src={product.image.url} alt
5                      ={product.name} />
6                  </Link>
7
8          <Link className="product-link" to={`/detail/${product.id}`}>
9              <div className="card__banner">
10                 <h2 className="card__title">{product.name}</h2>
11                 <h3 className="card__price">{product.price.
12                     formatted}</h3>
13                 </div>
14                 <div className="card__content" dangerouslySetInnerHTML
15                     ={{
16                         __html: product.description
17                     }} />
18             </Link>
19             <div className="buy-btn">
20                 <div className="buyBTN-content" onClick={
21                     handleAddToCart}>
22                     <span className="buyBTN-text">Add to Cart</span>
23                     <span className="addShoppingC"><AddShoppingCart
24                         className={classes.icon} /></span>
25                 </div>
26             </div>
27         </div>
28     </div>
29 
```

Die Function-Komponente bekommt „product“ und „onAddToCart“ als Parameter übergegen. Der Parameter „onAddToCart“ wird für die Funktionalität des Warenkorbs verwendet, er ermöglicht das Hinzufügen von Produkten.

Die Produkte werden in „Cards“ dargestellt, diese wurden mit Div-Boxen aufgebaut. Um das Produktbild anzuzeigen, muss als Source des Image-Tags „product.image.url“ verwendet werden. Der Produktname wird mittels „product.name“ angezeigt. Damit der Preis mit korrektem Format dargestellt werden kann, wird die Property „product.price.formatted“ genutzt. Um die Produktbeschreibung anzeigen zu können, ist es erforderlich diese mithilfe der Eigenschaft „dangerouslySetInnerHTML“ aufzurufen. Dabei wird die Property der Eigenschaft übergeben.

Der Link-Tag wird dazu verwendet, um auf eine Detail-Ansicht des Produktes weiterzuleiten. Hierbei werden dynamische Unterseiten mithilfe von dynamischen URLs erstellt, da die Produkt-ID als URL-Parameter übergeben wird.

Warenkorb

Um einen funktionsfähigen Waren zu erhalten, sind einige States erforderlich, diese werden wie folgt gesetzt:

```

1  const [cart, setCart] = useState({});
2  const [order, setOrder] = useState({});
3
4  const fetchCart = async () => {
5      setCart(await commerce.cart.retrieve());
6  };
7
8  const handleAddToCart = async (productId, quantity) => {
9      const item = await commerce.cart.add(productId, quantity);
10     setCart(item.cart);
11 };
12
13 const handleUpdateCartQty = async (lineItemId, quantity) => {
14     const response = await commerce.cart.update(lineItemId, {
15         quantity
16     });
17     setCart(response.cart);
18 };
19
20 const handleRemoveFromCart = async (lineItemId) => {
21     const response = await commerce.cart.remove(lineItemId);
22     setCart(response.cart);
23 };
24
25 const handleEmptyCart = async () => {
26     const response = await commerce.cart.empty();
27     setCart(response.cart);
28 };
29
30 const refreshCart = async () => {
31     const newCart = await commerce.cart.refresh();
32     setCart(newCart);
33 };
34
35 const handleCaptureCheckout = async (checkoutTokenId, newOrder) => {
36     try {
37         const incomingOrder = await commerce.checkout.capture(
38             checkoutTokenId, newOrder);
39
40         setOrder(incomingOrder);
41
42         refreshCart();
43     } catch (error) {
44         setErrorMessage(error.data.error.message);
45     }
46 };
47
48 useEffect(() => {
49     fetchComponents();
50     fetchPackages();
51     fetchCart();
52 });

```

```

49
50      }, []);

```

Die „useEffect()“-Methode ermöglicht die automatische Aktualisierung der Daten, ohne die Website zu aktualisieren.

Um einen Bezahlvorgang zu ermöglichen, wird die Plattform „Stripe“ verwendet. Die Programmierung des Bezahlvorgangs erfolgt mittels Checkout-Token, diese werden durch Commerce.js generiert. Die Generierung dieses Tokens erfolgt, wie folgt:

```

1   useEffect(() => {
2     if (cart.id) {
3       const generateToken = async () => {
4         try {
5           const token = await commerce.checkout.generateToken(
6             cart.id, { type: 'cart' });
7           setCheckoutToken(token);
8         } catch {
9           if (activeStep !== steps.length) navigate('/');
10        }
11      };
12      generateToken();
13    }
14  }, [cart]);

```

Ein wichtiger Bestandteil des Bestellvorgangs ist die Angabe der Versandadresse. Hierfür wird ein Adressformular verwendet. Nach der erfolgreichen Eingabe werden Daten in einem State abgespeichert.

```

1   const test = (data) => {
2     setShippingData(data);
3
4     nextStep();
5   };

```

Mithilfe folgender Zeile wird eine Bestellbestätigung angezeigt, nachdem man alle Schritte des Bestellvorgangs durchgegangen ist:

```

1   {activeStep === steps.length ? <Confirmation /> : checkoutToken
&& <Form />}

```

Für das Adressformular werden für alle benötigten Daten bezüglich des Versandes, die Auswahlmöglichkeiten von Commerce.js abgerufen und in States abgespeichert. Dies erfolgt folgendermaßen:

```

1  const [shippingCountries, setShippingCountries] = useState([]);
2  const [shippingCountry, setShippingCountry] = useState('');
3  const [shippingSubdivisions, setShippingSubdivisions] =
  useState([]);
4  const [shippingSubdivision, setShippingSubdivision] = useState(
  '');
5  const [shippingOptions, setShippingOptions] = useState([]);
6  const [shippingOption, setShippingOption] = useState('');
7  const methods = useForm();
8
9  const fetchShippingCountries = async (checkoutTokenId) => {
  const { countries } = await commerce.services.
localeListShippingCountries(checkoutTokenId);
10
11    setShippingCountries(countries);
12    setShippingCountry(Object.keys(countries)[0]);
13  };
14
15
16  const fetchSubdivisions = async (countryCode) => {
  const { subdivisions } = await commerce.services.
localeListSubdivisions(countryCode);
17
18    setShippingSubdivisions(subdivisions);
19    setShippingSubdivision(Object.keys(subdivisions)[0]);
20  };
21
22
23  const fetchShippingOptions = async (checkoutTokenId, country,
stateProvince = null) => {
24    const options = await commerce.checkout.getShippingOptions(
checkoutTokenId, { country, region: stateProvince });
25
26    setShippingOptions(options);
27    setShippingOption(options[0].id);
28  };
29
30  useEffect(() => {
31    fetchShippingCountries(checkoutToken.id);
32  }, []);
33
34  useEffect(() => {
35    if (shippingCountry) fetchSubdivisions(shippingCountry);
36  }, [shippingCountry]);
37
38  useEffect(() => {
39    if (shippingSubdivision) fetchShippingOptions(checkoutToken
.id, shippingCountry, shippingSubdivision);
40  }, [shippingSubdivision]);

```

5.2.2 Implementierung von Stripe

Für den Bezahlungsvorgang wird mithilfe des Systems „Stripe“ ein Zahlungsmittelformular erstellt. Dieses Formular dient dazu, die Kreditkartendaten des Kaufenden zu erhalten. Die Daten werden überprüft und wenn es alle Kriterien erfüllt, wird das Geld abgebucht und auf das Geschäftskonto überwiesen. Somit ist der Bestellvorgang beendet.

Das Ganze wird mit folgendem Source-Code implementiert:

```

1   const PaymentForm = ({ checkoutToken, nextStep, backStep,
2     shippingData, onCaptureCheckout }) => {
3     const handleSubmit = async (event, elements, stripe) => {
4       event.preventDefault();
5
6       if (!stripe || !elements) return;
7
8       const cardElement = elements.getElement(CardElement);
9
10      const { error, paymentMethod } = await stripe.
11        createPaymentMethod({ type: 'card', card: cardElement });
12
13      if (error) {
14        console.log('[error]', error);
15      } else {
16        const orderData = {
17          line_items: checkoutToken.live.line_items,
18          customer: { firstname: shippingData.firstName, lastname
19            : shippingData.lastName, email: shippingData.email },
20          shipping: { name: 'International', street: shippingData.
21            address1, town_city: shippingData.city, county_state:
22            shippingData.shippingSubdivision, postal_zip_code: shippingData.
23            zip, country: shippingData.shippingCountry },
24          fulfillment: { shipping_method: shippingData.
25            shippingOption },
26          payment: {
27            gateway: 'stripe',
28            stripe: {
29              payment_method_id: paymentMethod.id,
30            },
31            },
32          },
33        };
34
35        console.log(orderData);
36
37        onCaptureCheckout(checkoutToken.id, orderData);
38
39        nextStep();
40      }
41    };
42  };

```

Stripe liefert das Formular für die Kreditkartendaten mittels „stripe.createPaymentMethod“. Alle anderen erforderlichen Daten sind bereits in States abgespeichert, diese werden

alle in das Objekt „orderData“ gespeichert. Die Zahlungsmethode wird ebenfalls dort abgespeichert.

Am Ende der Bestellung wird die Commerce.js Methode „onCaptureCheckout“ aufgerufen, hierbei werden der Checkout-Token und das Objekt „orderData“ übergeben. Diese Methode dient dazu den Bestellvorgang abzuschließen. Alles weitere wird im Hintergrund erledigt.

5.3 Implementierung des Backend [SK]

Das Backend besteht aus einer JavaScript Datei, in welcher sämtliche Endpunkte definiert werden. Weiters benötigt das Backend einige Pakete, um beispielsweise mit der Datenbank zu kommunizieren. Diese werden in `package.json` aufgeführt. Bei erstmaliger Installation wird die Datei automatisch durchlaufen, und installiert alle benötigten Dateien und Pakete. Dazu wird der Terminal-Befehl `npm install` ausgeführt. Der Node Package Manager installiert alle Pakete im Ordner `node_modules`.

Der Server selbst wird weiters eingeteilt in:

- *Imports:* Zu Beginn werden alle benötigten Pakete, wie unter anderem `express` importiert. Dies geschieht mittels `require('express')`.
- *Setup:* In weiterer Folge werden benötigte Variablen, wie die Verbindungs-URL der MongoDB Datenbank, definiert. Da die Datenbank und der Server auf der selben Maschine laufen, wird der Host auf `mongodb://127.0.0.1:27017` gesetzt. Der weitere Aufbau der URL besteht aus Parametern wie zum Beispiel der `serverSelectionTimeoutMS`.
- *Express:* Es bietet Möglichkeiten zum schreiben von Handlern für Anfragen mit verschiedenen HTTP-Verben an verschiedenen URL-Pfaden (Routen) sowie zur Festlegung allgemeiner Einstellungen für Webanwendungen, wie z. B. des Ports, der für die Verbindung verwendet werden soll, und des Speicherorts von Vorlagen, die für das Rendering der Antwort verwendet werden. Um die Express-Applikation zu erstellen, wird `const app = express();` verwendet.
- *Header:* Um den Zugriff des Raspberry auf den Server zu gewährleisten, müssen die Header-Parameter gesetzt werden. Dies geschieht mittels `app.use()` und einer als Parameter übergebenen Funktion, in der mit `res.setHeader()` die Header-Parameter gesetzt werden. Diese bestimmen welche IP-Adressen auf den

Server zugreifen können, welche Methoden verwendet werden dürfen, sowie die zugelassenen Header bei Anfragen des Clients.

- *Konfiguration der Datenbank-Verbindung:* Um den Server mit der Datenbank kommunizieren zu lassen, sind einige Parameter notwendig. Diese werden wie folgt gesetzt:

```

1      MongoClient.connect(connstring, {useUnifiedTopology: true
2      })
3          .then(client => {
4              const db = client.db('aperta')
5
6              const licenseplateCollection =
7                  db.collection('licenseplate')
8              const rfidCollection = db.collection('rfid')
9              const numpadCollection = db.collection('numpad')

```

Listing 35: Konfiguration der Datenbankanbindung

- *Handler:* Handler sind die Endpunkte, mit denen die Clients kommunizieren. Diese unterscheiden sich je nach Art der Anfrage. Die Anfragen werden mittels `app.get()`, `app.post()`, `app.put()` und `app.delete()` definiert. Als Parameter wird der Endpunkt angegeben, sowie die Funktion, welche ausgeführt wird, wenn der Endpunkt aufgerufen wird. Eine Abfrage der Kennzeichen, welche in der Datenbank gespeichert sind, sieht somit wie folgt aus:

```

1
2      app.get('/get-licenseplates', function(req, res) {
3          licenseplateCollection.find().toArray()
4              .then(results => {
5                  console.log("retrieving licenseplates")
6                  res.send(results)
7              })
8              .catch(error => console.error(error))
9                  // do something here
10         })
11

```

Listing 36: Abfrage der Kennzeichen

- *Verbindung:* Die Funktion `app.listen()` wird verwendet, um die Verbindungen an den angegebenen Host und Port zu binden und abzuhören. Diese Methode ist identisch mit der Methode `http.Server.listen()` von Node. Wenn die Portnummer weggelassen wird oder 0 ist, weist das Betriebssystem einen beliebigen unbenutzten Port zu, was für Fälle wie automatisierte Aufgaben (Tests usw.) nützlich ist. Die von `express()` zurückgegebene App ist in Wirklichkeit eine JavaScript-Funktion, die als Callback an die HTTP-Server von Node übergeben wird, um Anfragen zu bearbeiten.

5.4 Implementierung der Kennzeichenerkennung[SK]

5.4.1 Funktionsweise

Wie bereits in Kapitel 3.3 angeführt, besteht die Kennzeichenerkennung aus 3 Schritten. Bevor diese jedoch durchlaufen werden können, müssen zuerst alle notwendigen Bibliotheken installiert und importiert werden. Diese können mittels `pip3 install <name>`, oder aber auch mit `sudo apt-get install <name>` installiert werden.

Nachdem alle Bibliotheken und das Modul zur Überprüfung des Kennzeichens importiert wurden, beginnt mit `cap = cv2.VideoCapture(0)` das Aufzeichnen des Bildes. Dieses wird durch eine Schleife ausgeführt, bis das Programm beendet wird. Falls die Kamera nicht aufgenommen werden kann, wird eine Fehlermeldung ausgegeben.

Graustufenbild

Nachdem das Bild angezeigt wurde, wird der erste Filter angewendet. Dabei handelt es sich um einen Filter, welcher das Bild in ein Graustufenbild umwandelt. Dieser wird mit der Methode `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)` aufgerufen.

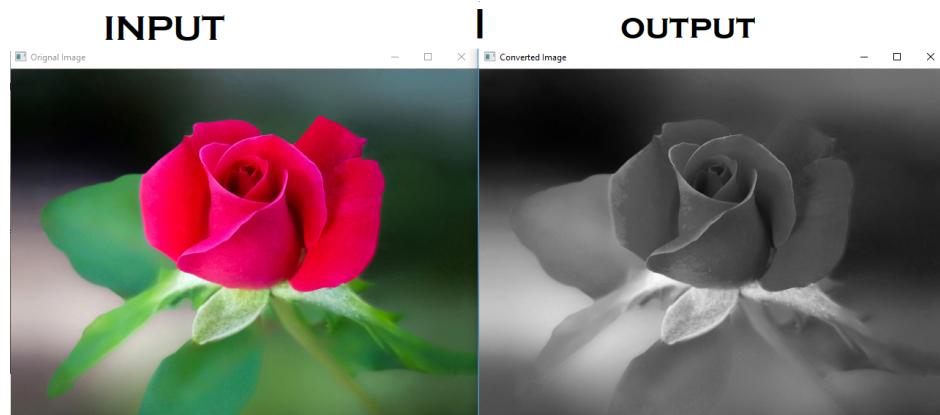


Abbildung 45: Aufgenommenes Bild vs. Graustufenbild

[88]

Filter

Als zweiter Filter fungiert ein `cv2.bilateralFilter`. Die wichtigste Eigenschaft der bilateralen Filterung ist, dass sie keine Mittelwertbildung über die Kanten vornimmt. Deshalb wird sie auch als kantenerhaltender Filter bezeichnet. Bevor die mathematischen Grundlagen des Bilateralfilters erklärt werden, ist es jedoch sinnvoll, kurz auf die Gaußsche Filterung einzugehen, da diese dem bilateralen Filter ähnlich ist. Die Gaußsche Filterung ist ein gewichteter Durchschnitt der Intensität der benachbarten

Positionen, wobei die Gewichtung mit dem räumlichen Abstand zur mittleren Position abnimmt.

Mathematisch gesehen ist ein mit Gaußscher Unschärfe gefiltertes Bild gegeben durch:

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_\sigma(\|\mathbf{p} - \mathbf{q}\|) I_{\mathbf{q}},$$

Abbildung 46: Ein mit Gaußscher Unschärfe gefiltertes Bild
[89]

Dabei sind p und q die Position der Pixel, I kennzeichnet das Bild und $G\sigma(x)$ den sogenannten zweidimensionalen Gauß-Kernel.

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

Abbildung 47: Gauß-Kernel
[90]

$G\sigma$ ist ein räumlicher Gauß, der den Einfluss der entfernten Pixel verringert. Der Abstand zwischen den Pixeln wird mit $G\sigma(\|p - q\|)$ angegeben. Hierbei ist σ die Ausdehnung der Nachbarschaft.

Der Gaußsche-Unschärfefilter untersucht nur naheliegende Pixel bei der Filterung. Es wird nicht berücksichtigt, ob die Pixel die selbe Intensität haben, oder ob die Pixel am Rand liegen oder nicht. Daher werden auch Kanten verwischt, was zu einem Verlust wichtiger Details führt. Die bilaterale Filterung verwendet ebenfalls einen Gauß-Filter im Raum, berücksichtigt aber zusätzlich einen weiteren Gauß-Filter welcher eine Funktion der Pixeldifferenz ist. Die Gaußsche Funktion des Raums sorgt dafür, dass nur nahe gelegene Pixel für die Unschärfe berücksichtigt werden, während die Gaußsche Funktion der Intensitätsdifferenz dafür sorgt, dass nur die Pixel mit ähnlichen Intensitäten wie das zentrale Pixel für die Unschärfe berücksichtigt werden. So bleiben die Kanten erhalten, da die Pixel an den Kanten große Intensitätsunterschiede aufweisen. Der wichtige Punkt, der bei der bilateralen Filterung berücksichtigt wird, ist, dass zwei Pixel nicht nur dann nahe beieinander liegen, wenn sie dies räumlich tun, sondern auch, wenn sie eine gewisse Ähnlichkeit im photometrischen Bereich aufweisen. Diese Eigenschaften der bilateralen Filterung überwinden den Nachteil anderer Techniken wie *Averaging Blur*

oder *Gaussian Blur*, da sie in der Lage ist, Kanten zu erhalten.

Der Bilaterale Filter wird wie folgt berechnet:

$$BF[I]_P = \frac{1}{W_P} \sum_{q \in S} G_{\sigma_s}(\|P - q\|) G_{\sigma_r}(I_P - I_q) I_q$$

Abbildung 48: Bilaterale Filterung Formel
[91]

W ist ein Normalisierungsfaktor, welcher mit folgender Formel berechnet wird:

$$W_P = \sum_{q \in S} G_{\sigma_s}(\|P - q\|) G_{\sigma_r}(I_P - I_q)$$

Abbildung 49: Formel des Normalisierungsfaktors W_P
[92]

Dem Gaußschen Filter werden 2 neue Terme hinzugefügt, um den bilateralen Filter zu erhalten. Diese Terme sind:

$$G_{\sigma_r}(I_P - I_q) I_q$$

Abbildung 50: Term 1
[93]

$$\frac{1}{W_P}$$

Abbildung 51: Term 2
[94]

Wie aus der Gauß-Filterung bereits bekannt, ist $G\sigma_s$ ein räumlicher Gauß, welcher den Einfluss entfernter Pixel verringert. Der zweite Term ist $G\sigma_r$, ein Bereichs-Gauß, welcher Einflüsse von Pixeln q mit anderen Intensitätswerten als I_p verringert. Bereich bedeutet in diesem Fall Größen, die sich auf Intensitäten beziehen, wohingegen der Raum die Lage der Pixel beschreibt.

σ_s ist ein Raumparameter, der für die räumliche Ausdehnung der Kernelgröße in der betrachteten Nachbarschaft beschreibt, und σ_r hingegen ein Entfernungsparameter,

welcher die minimale Amplitude einer Kante beschreibt. Zusammen ergeben die Parameter das Ausmaß der Filterung des Bildes I .

Daraus folgt:

1. Jedes Pixel erhält einen Durchschnitt der benachbarten Pixel, mit dem es überschrieben wird.
2. Jedes benachbarte Pixel wird durch eine räumliche Komponente, die entfernte Pixel bestraft, und eine Bereichskomponente, die Pixel mit unterschiedlicher Intensität bestraft, gewichtet.
3. Die Kombination der beiden Komponenten stellen sicher, dass nur nahe, ähnliche Pixel zur Berechnung des Ergebnisses berücksichtigt werden.

Die beiden Parameter σ_r und σ_s beeinflussen den bilateralen Filter, indem bei Erhöhung des Entfernungsparameter σ_r das Ergebnis dem einer Gaußschen Unschärfe ähnelt, und die Erhöhung des räumlichen Parameters σ_s zur Glättung von großen Merkmalen führt. Veranschaulicht kann dies durch folgende Grafik werden:

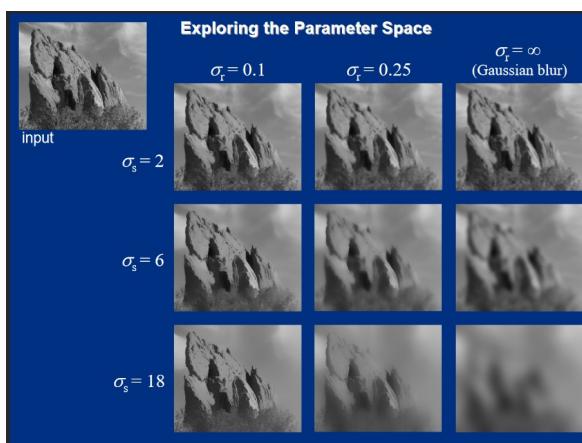


Abbildung 52: Beeinflussung durch Veränderung der Parameter σ_s und σ_r [95]

Bei der Verwendung der Funktion `cv2.bilateralFilter()` sind folgende Parameter zu berücksichtigen:

- *src*: Ein Bild, auf das die Filterung angewendet wird.
- *dst*: Das Ergebnis der Filterung. Das Zielbild hat die selbe Größe und den selben Datentyp wie das Ausgangsbild *src*.
- *d*: Durchmesser der Pixel-Nachbarschaften, welche zur Berechnung des Filters verwendet werden. Wird dieser negativ, kann auf eine Berechnung durch *sigmaSpace* zurückgegriffen werden.

- *sigmaColor*: Ein größerer Wert des Parameters bedeutet, dass weiter entfernte Farben innerhalb der Pixelnachbarschaft zusammengemischt werden, was zu größeren Bereichen mit annähernd gleicher Farbe führt.
- *sigmaSpace*: Ein größerer Wert des Parameters bedeutet, dass weiter entfernte Pixel sich gegenseitig beeinflussen, solange ihre Farben nahe genug beieinander liegen. Wenn $d > 0$ ist, gibt er die Größe der Nachbarschaft unabhängig von *sigmaSpace* an. Andernfalls ist *d* proportional zu *sigmaSpace*.
- *borderType*: Randmodus, der für die Extrapolation von Pixeln außerhalb des Bildes verwendet wird.

[96]

Kantenerkennung

Die Kantenerkennung wird mit dem Aufruf von `cv2.Canny()` durchgeführt. Diese Funktion wurde von John F. Canny entwickelt, und ist ein mehrstufiger Algorithmus, welcher in 4 Stufen aufgeteilt werden kann:

1. *Rauschunterdrückung*:
 - a. Da die Kantenerkennung anfällig für Bildrauschen ist, wird zunächst das Bildrauschen mit einem 5x5-Gauß-Filter entfernt
2. *Ermittlung des Intensitätsgradienten des Bildes*:
 - a. Das geblättete Bild wird dann mit einem Sobel-Kernel sowohl in horizontaler als auch in vertikaler Richtung gefiltert, um die erste Ableitung in horizontaler Richtung (G_x) und vertikaler Richtung (G_y) zu erhalten. Aus diesen beiden Bildern können wir den Kantengradienten und die Richtung für jedes Pixel wie folgt ermitteln:

$$\begin{aligned} \text{Kantengradient } (G) &= \sqrt{G_x^2 + G_y^2} \\ \text{Winkel } (\theta) &= \tan^{-1}\left(\frac{G_y}{G_x}\right) \end{aligned} \tag{5.1}$$

- b. Die Richtung der Steigung ist immer senkrecht zu den Kanten. Sie wird auf einen von vier Winkeln gerundet, welche die vertikale, horizontale und zwei diagonale Richtungen darstellen.
3. *Non-Maximum-Suppression*:
 - a. Nachdem Gradient und Richtung ermittelt wurden, wird das Bild vollständig gescannt, um alle unerwünschten Pixel zu entfernen, die möglicherweise nicht

die Kante bilden. Zu diesem Zweck wird bei jedem Pixel geprüft, ob es in dessen Umgebung ein lokales Maximum in der Richtung des Gradienten darstellt.

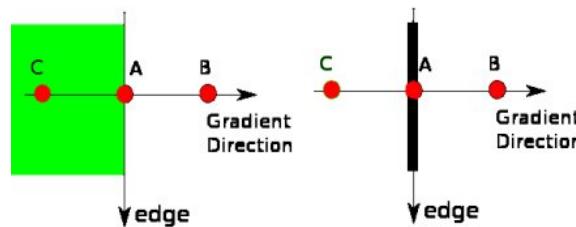


Abbildung 53: Non-Maximum-Suppression
[97]

Punkt A befindet sich auf der Kante in vertikaler Richtung. Die Richtung des Gradienten ist normal zur Kante. Die Punkte B und C liegen in Gradientenrichtung. Punkt A wird also mit den Punkten B und C daraufhin überprüft, ob er ein lokales Maximum bildet. Wenn ja, wird er für die nächste Stufe berücksichtigt, andernfalls wird er unterdrückt (auf Null gesetzt). Das Ergebnis ist ein Binärbild mit dünnen Rändern.

4. Hysterese-Schwellenwertbildung:

- In dieser Phase wird entschieden, welche Kanten wirklich Kanten sind und welche nicht. Hierfür werden zwei Schwellenwerte benötigt, `minVal` und `maxVal`. Alle Kanten, deren Intensitätsgradient größer als `maxVal` ist, sind sicher, dass es sich um Kanten handelt, und diejenigen, die unter `minVal` liegen, sind sicher, dass es sich um Nicht-Kanten handelt, und werden daher verworfen. Diejenigen, die zwischen diesen beiden Schwellenwerten liegen, werden auf der Grundlage ihrer Konnektivität als Kanten oder Nicht-Kanten klassifiziert. Wenn sie mit "sure-edgePixeln verbunden sind, werden sie als Teil von Kanten betrachtet. Andernfalls werden sie ebenfalls verworfen.

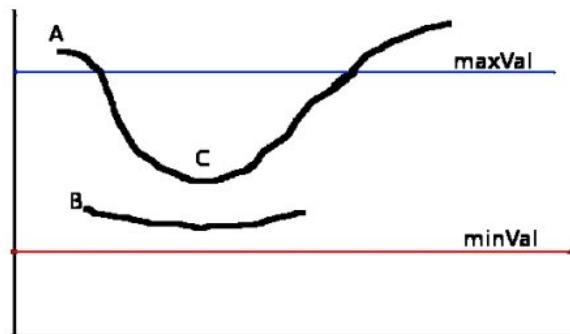


Abbildung 54: Hysterese-Schwellenwertbildung
[98]

Die Kante A liegt oberhalb von maxVal, wird also als sichere Kante betrachtet. Obwohl die Kante C unter maxVal liegt, ist sie mit der Kante A verbunden, so dass sie ebenfalls als gültige Kante betrachtet wird und wir die vollständige Kurve erhalten. Aber die Kante B, obwohl sie oberhalb von minVal liegt und in der gleichen Region wie die Kante C ist, ist nicht mit einer sicheren Kante verbunden, so dass sie verworfen wird. Es ist also sehr wichtig, dass minVal und maxVal entsprechend ausgewählt werden, um das richtige Ergebnis zu erhalten. In dieser Phase werden auch kleine Pixel entfernt, wobei davon ausgegangen wird, dass die Kanten lange Linien sind. Dadurch entstehen starke Kanten im Bild.

Angewendet auf ein Graustufenbild, sieht das Ergebnis der Kantenerkennung wie folgt aus:



Abbildung 55: Ergebnis der Kantenerkennung
[99]

Konturenfindung

Konturen können einfach als eine Kurve erklärt werden, die alle kontinuierlichen Punkte (entlang der Grenze) verbindet, die dieselbe Farbe oder Intensität haben. Die Konturen sind ein nützliches Werkzeug für die Formanalyse und die Erkennung von Objekten. Mit dem Ergebnis der Kantenerkennung aus 5.4.1 kann OpenCV Konturen eines weißen Objekts vor einem schwarzen Hintergrund finden. Dies geschieht mit dem Aufruf von `cv2.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)`. Die Parameter können wie folgt beschrieben werden:

1. *thresh*: Das Bild, auf das die Konturen gesucht werden sollen.
2. *mode*: Der Modus, der zur Konturensuche verwendet wird.
3. *method*: Die Methode, die zur Konturenähnlichkeit verwendet wird.

Als Rückgabewert gibt die Funktion die Konturen und die Hierarchie aus. Die Konturen sind eine Python-Liste mit allen Konturen im Bild. Jede einzelne Kontur ist ein Numpy-Array mit (x,y)-Koordinaten von Grenzpunkten des Objekts. Der dritte Parameter,

die Konturennäherungsmethode, bestimmt, welche Begrenzungspunkte gespeichert werden. Bei `cv2.CHAIN_APPROX_NONE` werden alle Punkte der Kontur gespeichert. Dies ist aber nicht nötig, da nur der Start- und Endpunkt der Linie benötigt werden. Daher ist es sinnvoller, `cv2.CHAIN_APPROX_SIMPLE` zu verwenden, da diese Methode alle überflüssigen Punkte entfernt und die Konturen komprimiert, um so Speicherplatz zu sparen. In Abbildung 56 wird der Unterschied der beiden Methoden erkenntlich gemacht. Anstelle von 734 Punkten bei der Verwendung von `cv2.CHAIN_APPROX_NONE` werden bei `cv2.CHAIN_APPROX_SIMPLE` nur die Start- und Endpunkte der Konturen und damit insgesamt 4 Punkte gespeichert.[100]

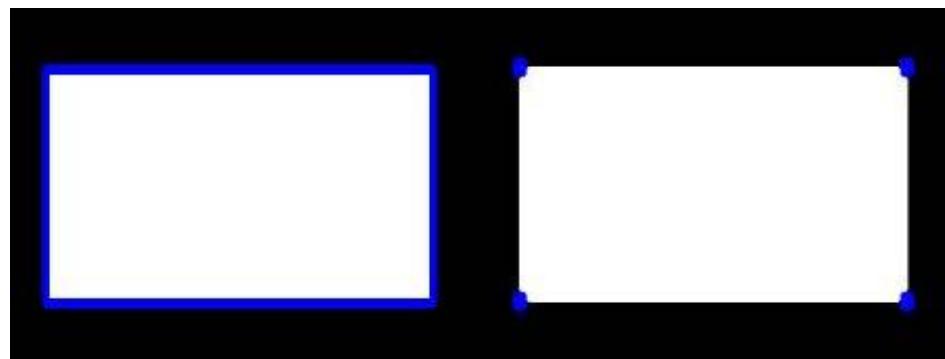


Abbildung 56: Unterschied von
[101]

Überprüfung der Konturen

Im nächsten Schritt wird mittels `imutils.grab_contours` überprüft, welche OpenCV Version verwendet wird. Dies unterscheidet sich in der Länge des Kontur-Tupels:

- Ist das Tupel 2 Elemente lang, handelt es sich um OpenCV v2.4, v4-beta oder v4-official
- Ist das Tupel 3 Elemente lang, handelt es sich um OpenCV v3, v4-pre oder v4-alpha

[102]

Nachdem die Konturen sortiert wurden, wird der Konturumfang mit `cv2.arcLength` berechnet. Danach wird die Kontur unter Verwendung des Douglas-Peucker-Algorithmus mit dem vorher berechneten Konturumfang angepasst. Kommt dieser zu keinem Ergebnis, wird die Kontur verworfen und eine Fehlermeldung ausgegeben.

Wurde jedoch eine geschlossene Kontur mit 4 Eckpunkten gefunden, beginnt die Funktion `cv2.drawContours` mit der Zeichnung der Kontur. Die Funktionsparameter sind zum Einen das Bild, auf das die Kontur gezeichnet werden soll, und zum anderen die Kontur.

Weitere Parameter, wie Farbe, Dicke und mehr können ebenfalls übergeben werden. [100]

Maskieren der relevanten Bildregion

Da für den weiteren Ablauf des Programms nur noch die Region, in der sich das Kennzeichen befindet, relevant ist, wird diese maskiert und als neues Bild gespeichert:

```

1     mask = np.zeros(gray.shape, np.uint8)
2     new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1, )
3     new_image = cv2.bitwise_and(frame, frame, mask=mask)
4     (x, y) = np.where(mask == 255)
5     (topx, topy) = (np.min(x), np.min(y))
6     (bottomx, bottomy) = (np.max(x), np.max(y))
7     Cropped = gray[topx:bottomx+1, topy:bottomy+1]

```

Listing 37: Maskieren der Kennzeichenregion und Abspeichern in neuem Bild

Texterkennung

Aus dem in 5.4.1 erzeugten Bild kann nun der Text des Kennzeichens gelesen werden.

Dies geschieht mit der Funktion :

```
1     text = pytesseract.image_to_string(Cropped, config='--psm 11')
```

Listing 38: Maskieren der Kennzeichenregion und Abspeicher in neuem Bild

Mit `config='--psm 11'` wird die Priorität der Texterkennung auf den Text selbst gesetzt. Da zuvor schon alle irrelevanten Bereiche des Bildes durch Maskieren entfernt wurden, wurde hier diese Priorität verwendet, um die Ergebnisse der Kennzeichenerkennung zu verbessern. [103]

Dadurch kann es aber auch zu Problemen kommen, da nun Sonderzeichen erkannt werden, welche nicht auf dem Kennzeichen existieren. Um alle Leerzeichen und Sonderzeichen zu entfernen, wurden mit `re.sub('[^a-zA-Z0-9 \n\.]', '', text)` alle nicht möglichen Zeichen aus dem Text entfernt.

Überprüfung des erkannten Kennzeichens

Der Text aus 5.4.1 wird mit dem Aufruf des Modules `checkPlate(text_replaced)` an ein Python-Script übergeben, welches für die weitere Überprüfung zuständig ist.

```

1     print(type(text))
2     r = requests.get('http://130.162.215.116/get-licenseplates')
3     val = json.loads(r.text)

```

Listing 39: Abfrage der Kennzeichen aus der Datenbank

Im Code-Abschnitt 40 wird die Antwort des Servers in ein JSON-Objekt geladen und mit `json.loads` in ein Python-Objekt umgewandelt, welches die Daten der gefundenen Kennzeichen enthält. Da die Formatierung der Kennzeichen aus der Datenbank sich von dem der Ergebnis der Erkennung unterscheiden kann, wird mit `str.maketrans` eine Zuordnungstabelle erstellt, die im späteren Verlauf von der Methode `str.translate` verwendet wird, um die Formatierungsunterschiede zu beseitigen.

Kennzeichen können im Dashboard deaktiviert werden und dürfen in diesem Zustand nicht die Garage öffnen. Um dies zu realisieren, wird der Boolean `active` aus der Datenbankabfrage jedes Kennzeichens überprüft. Wenn dieser Boolean auf `False` steht, wird das Kennzeichen nicht weiter überprüft.

```

1      if text.replace(" ", "").translate(table) == value["  

2          licenseplate"].translate(table).replace(" ", ""):  

3              print("licenseplate recognized, initiating opening sequence  

4          ")  

5              initiateOpeningSequence()  

6      else:  

7          print("not recognized, staying closed")
8

```

Listing 40: Überprüfung auf Gleichheit der beiden Strings

Sind beide Kennzeichen gleich, wird die Methode `initiateOpeningSequence` aufgerufen, welche das Relais ansteuert.

5.5 Implementierung des Displays[SK]

Das Display kann mithilfe von drei verschiedenen Methoden unterschiedliche Texte anzeigen.

1. Kombination des Nummernfeldes korrekt
 1. Bei korrekter Eingabe des Nummernfeldes wird das Display wie folgt beschrieben:

```

1      outString = writingString
2      lcd.text("Combination:", 1)
3      lcd.text(outString, 2)
4

```

Listing 41: Ausgabe bei korrekter Eingabe der Kombination auf dem Nummernfeld

Nach einer Wartezeit von 5 Sekunden wird das Display geleert und steht für neue Aufgaben zur Verfügung.

2. Kombination des Nummernfeldes nicht korrekt
 1. Bei inkorrekt er Eingabe des Nummernfeldes soll nicht ausgegeben werden, an welcher Stelle der Eingabe sich der Fehler befindet. Daher gibt das Display **Wrong Combination, please try again** aus.
3. NFC-Tag nicht korrekt
 1. Bei nicht autorisierten NFC-Karten sowie bei fehlerhaften Lesevorgängen muss das Garagentor verschlossen bleiben. Daher wird in diesem Fall vom Display **Error at reading, please try again** ausgegeben.

5.6 Implementierung des Relais[SK]

Das Relais wird von den drei Zutrittsmöglichkeiten angesteuert. Dazu muss der GPIO, an dem sich die Steuerung des Relais befindet, als Ausgang definiert werden. Wenn das Garagentor geöffnet werden soll, muss über diesen GPIO Spannung abgegeben werden, um den Schalter auf der Innenseite der Garage zu immitieren. Das Relais wird wie ein Schalter in den Arbeitsstromkreis der Garagentorsteuerung eingebunden und schließt mit Ansteuerung des GPIOs den eigenen Steuerkreis, aktiviert den Arbeitskreis und der Öffnungsmechanismus wird vom Motor der Garage ausgeführt. Das Relais durch den Aufruf von `initiateOpeningSequence` aus Code-Abschnitt 42 angesteuert.

```

1 import RPi.GPIO as GPIO
2 import time
3
4 in1 = 7
5 GPIO.setmode(GPIO.BCM)
6 GPIO.setup(in1, GPIO.OUT)
7
8 GPIO.output(in1, False)
9
10 def initiateOpeningSequence():
11     GPIO.output(in1, True)
12     time.sleep(0.1)
13     GPIO.output(in1, False)
14     GPIO.cleanup()

```

Listing 42: Ansteuerung des Relais

5.7 Implementierung des Keypad [DH]

Das der Raspberry Pi holt sich die gesamten Nummernfeldkombinationen von der Datenbank. Diese werden dann mit dem Eingegebenen Code abgeglichen und auf

Übereinstimmungen geprüft. Stimmt der eingegebene Code und die Kombination der Datenbank überein, so öffnet sich das Garagentor.

```

1   import requests
2   import json
3   from relais import initiateOpeningSequence
4   from testDisplay import displayText, readNumpadUnsuccessful
5
6   def checkNumpad(numpad_code):
7       inList = False
8       r = requests.get('http://130.162.215.116/get-numpad-codes')
9       val = json.loads(r.text)
10      for value in val:
11          print("value: ", value["numpad_code"])
12          print("text from reader: ", numpad_code)
13          if value["active"]:
14              if numpad_code == value["numpad_code"]:
15                  print("numpad code recognized, initiating
opening sequence")
16                  inList = True
17              else:
18                  print("not recognized, staying closed")
19                  readNumpadUnsuccessful()
20          if inList:
21              displayText(str(numpad_code))
22              initiateOpeningSequence()
```

Listing 43: checkNumpad.py

Die von der Datenbank erhaltenen Codes werden in einer Schleife durchlaufen.

Das Keypad muss mit dem Raspberry Pi über 8-pins verbunden werden. Die Pins für die Spalten sind Outputs und werden auf high gesetzt. Die Reihenpins dienen als Input, sie sind standardmäßig auf high gesetzt. Der Output wird einmal auf low gesetzt, danach werden alle Inputs in kürzester Zeit abgefragt. Sollte es einen Wert geben, welcher auf low gesetzt ist, kann genau gesagt werden welche Taste gedrückt wurde.

```

1   import RPi.GPIO as GPIO
2   import time
3   from testDisplay import displayText
4   from relais import initiateOpeningSequence
5   from testDisplay import displayText, readNumpadUnsuccessful
6   import requests
7   import json
8
9   def numpad():
10     GPIO.setmode(GPIO.BCM)
11
12     MATRIX = [[1, 2, 3, 'A'],
13                [4, 5, 6, 'B'],
14                [7, 8, 9, 'C'],
15                ['#', 0, '#', 'D']]]
16
17
18     COUNT = 0
19     COMBIN = ""
20     ROW = [18, 23, 24, 26]
21     COL = [12, 16, 20, 21]
22
```

```

23     for j in range(4):
24         GPIO.setup(COL[j], GPIO.OUT)
25         GPIO.output(COL[j], 1)
26
27     for i in range(4):
28         GPIO.setup(ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)
29
30     try:
31         while True:
32             for j in range(4):
33                 GPIO.output(COL[j], 0)
34                 for i in range(4):
35                     if GPIO.input(ROW[i]) == 0:
36                         print(MATRIX[i][j])
37                         COUNT = COUNT+1
38                         COMBIN = COMBIN + str(MATRIX[i][j])
39                         if(COUNT > 5):
40                             #displayText(COMBIN)
41                             #checkNumpad(COMBIN)
42                             r = requests.get('http
43 ://130.162.215.116/get-numpad-codes')
44                             val = json.loads(r.text)
45                             for value in val:
46                                 print("value: ", value["numpad_code
47 "])
48                                 print("text from reader: ", COMBIN)
49                                 if value["active"]:
50                                     if COMBIN == value["numpad_code
51 "]:
52                                         print("numpad code
53 recognized, initiating opening sequence")
54                                         displayText(COMBIN)
55                                         initiateOpeningSequence()
56                                     else:
57                                         print("not recognized,
58 staying closed")
59                                         readNumpadUnsuccessful()
60                                         COMBIN = ""
61                                         COUNT = 0
62                                         while(GPIO.input(ROW[i]) == 0):
63                                             pass
64                                             time.sleep(0.2)
65                                             GPIO.output(COL[j], 1)
66
67     finally:
68         #GPIO.cleanup()
69         print("end of numpad")

```

Listing 44: Numpad.py

Mittels einer For-Schleife werden die Spaltenpins auf high gesetzt, die Pins in der Reihe werden mit einem pull-up Resistor versehen. Es läuft eine Endlosschleife, diese wartet darauf das ein Feld gedrückt wird.

5.8 Implementierung des RFID-Kits [BG]

Um das RFID-Kit benützen zu können, wird die bereitgestellte Library verwendet. Da die RFID-Karten bereits IDs besitzen, ist nur das Lesen der ID erforderlich. Dies wird folgendermaßen umgesetzt:

```

1   import RPi.GPIO as GPIO
2   import os
3   from mfrc522 import SimpleMFRC522
4   from checkNFC import checkNFC
5
6   reader = SimpleMFRC522()
7   def read():
8       try:
9           id, text = reader.read()
10          print(id)
11          print(text)
12          checkNFC(id)
13      finally:
14          GPIO.cleanup()
15          os.system("python3 read.py")

```

Listing 45: Lesen der RFID-Karte

Hierbei wird die Methode „reader.read()“ verwendet. Die Variablen „id“ und „text“ werden mit den Rückgabewerten dieser Methode deklariert. Um die ID zu überprüfen, wird die Methode „checkNFC()“ aufgerufen. Diese bekommt die Variable „id“ als Parameter übergeben.

Die Methode „checkNFC()“ führt ein GET-Request aus, um alle registrierten RFID-Karten-IDs abzurufen. Die zurückgegebenen Werte werden in der Variable „r“ gespeichert. Mithilfe einer For-Schleife werden alle RFID-IDs mit dem Anfangsparameter der Methode verglichen. Sobald eine Übereinstimmung gefunden wurde, wird das Garagentor geöffnet.

Das Ganze wird wie folgt im Source-Code umgesetzt:

```

1   import requests
2   import json
3   from relais import initiateOpeningSequence
4
5   from testDisplay import displayText, readNFCUnsuccessful
6
7   def checkNFC(rfid_id):
8       print(type(rfid_id))
9       r = requests.get('http://130.162.215.116/get-rfid-codes')
10      val = json.loads(r.text)
11      for value in val:
12          print("value: ", value["rfid_code"])
13          print("text from reader: ", rfid_id)
14          if value["active"]:
15              if rfid_id == value["rfid_code"]:
16                  print("rfid code recognized, initiating opening
sequence")

```

```
17             initiateOpeningSequence()
18             displayText(str(rfid_id))
19         else:
20             print("not recognized, staying closed")
21             readNFCUnsuccessful()
```

Listing 46: Überprüfung der RFID-Karten-ID

6 Umfeldanalyse

Es gibt bereits einige Schranken mit Kennzeichenerkennung. Gerade bei Tiefgaragen, öffentlichen Parkplätzen oder Tankstellen ist die Kennzeichenerkennung eine beliebte Anwendung. Fast alle Systeme werden jedoch als Komplettset verkauft. Das heißt, es müssen die Kennzeichenerkennung sowie die Schranke gekauft werden. Bei Tankstellen dient sie hauptsächlich als Überwachungskamera. Die Firma Taurus bietet ihren Kunden oder Kundinnen die Kennzeichenerkennung als Erweiterung an. Allerdings ist sie sehr kostspielig und nicht für den Privatgebrauch vorgesehen. Immer mehr Menschen sind auf der Suche nach einem einfacheren System für ihr Eigenheim.



Abbildung 57: Umsetzung von der Firma Taurus
[104]

Unsere Idee sollte kostengünstig und einfach zu montieren sein. Jedes bereits vorhandene Garagentor sollte damit erweitert werden können. Somit fallen keine Kosten für ein neues Tor an. Die Montage sollte von jedem durchgeführt werden können. Auch die Verwaltung der Kennzeichen und sonstigen Bauteile liegt in der Hand des Endverbrauchers oder der Endverbraucherin. Außerdem kann entschieden werden, welche Funktionen die Erweiterungen bieten soll. Der Kunde oder die Kundin müssen somit nicht das Komplettset kaufen. Wird eine verkleinerte Version gekauft, kann diese ganz einfach durch die verfügbaren Module erweitert werden.



Abbildung 58: Aperta zerlegt in Module

7 Persönliche Ziele

7.1 Projektverlauf

War das Thema der Diplomarbeit gefunden, ist ein Prototyp für das Frontend erstellt worden. Damit sollte garantiert werden, dass das Design des Profilmanagement mit dem des Webshop übereinstimmt. Der Prototyp nahm viel Zeit in Anspruch, doch beim Entwickeln hat er geholfen, denn es mussten sich keine Gedanken mehr über das Design gemacht werden.

Nach den Sommerferien 2021 wurden erste Teile, sowohl im Frontend als auch im Backend implementiert. Anfang November war das Design für die Profilseite und den Webshop fertig. Am Raspberry Pi hat die Kennzeichenerkennung bereits funktioniert und auch die Kommunikation mit der Datenbank wurde umgesetzt. Die Ergebnisse wurden unserem Betreuer Professor Aberger präsentiert.

In der nachfolgenden Zeit wurde das Projekt immer weiter vorangetrieben. Anfang des Jahres 2022 war das Backend fertig implementiert und die Abfragen der Datenbank konnten im Profil verwaltet werden. Der Webshop war fertig, es fehlten nur noch ein paar Kleinigkeiten, die bis zum Ende fertig programmiert wurden.

Der Fortschritt wurde erneut präsentiert und der Raspberry Pi konnte das Öffnen der Garage simulieren. Die Kennzeichenerkennung funktionierte, Codes konnten über das Nummernfeld eingegeben und NFC Chips gelesen werden. Auch der Webshop und das Profilmanagement hatten seine Funktion erfüllt. Die restliche Zeit wurde genutzt, um den schriftlichen Teil der Diplomarbeit fertigzustellen.

7.2 Erkenntnisse von Benjamin Golic

Obwohl wir bereits im SEW- beziehungsweise ITP-Unterricht etwas größere Projekte durchgeführt haben, war dieses ganz anders. Trotz vieler unerwarteter und schwer lösbarer Fehler, konnte das Projekt zu Ende entwickelt werden.

Im Vergleich zu bisherigen Programmierprojekten hat mir die Zusammenarbeit bei diesem am meisten Freude bereitet. Wir konnten uns gegenseitig motivieren und haben uns auch gegenseitig geholfen, als jemand wo nicht mehr weitergekommen ist.

Da wir Angular im SEW-Unterricht gelernt haben und ich kein so großer Freund davon bin, habe ich mich für React bei diesem Projekt entschieden. Der Umgang mit dieser Library gefiel mir so gut, dass ich mir vorstellen könnte, mich darauf für das spätere Berufsleben zu spezialisieren.

7.3 Erkenntnisse von David Hauser

Die Umsetzung des Projektes war nicht gerade einfach. Ich bin an einigen Punkten an meine Grenzen gestoßen. Dennoch hat mir die Zusammenarbeit mit meinen Kollegen viel Freude bereitet. Die Kommunikation mit ihnen ist immer reibungslos verlaufen und wir haben uns gegenseitig unterstützt, wo es nur möglich war. Durch die Umsetzung mittels Angular konnte ich mein Wissen noch mehr vertiefen und festigen. Das Entwickeln mit einer bekannten Programmiersprache hat mir das Projekt sehr vereinfacht. Trotzdem waren einige Dinge für mich neu. Der Beruf als Programmierer wird nicht mein zukünftiger werden. Das Programmieren hat mir teilweise sehr viele Nerven gekostet. Trotzdem hat das Team die Aufgaben gut erfüllt und die Zusammenarbeit war, wie erwähnt, sehr gut.

7.4 Erkenntnisse von Simon Koll

Das Projekt nahm immer mehr Substanz an, je weiter die Entwicklung fortschritt. Es kamen neue Ideen hinzu, die Anfangs noch gar nicht im Raum standen. Die Erkenntnisse und Inhalte des ITP-, INSY- sowie SEW-Unterrichts haben mich in der Entwicklung sehr unterstützt, da ich teilweise auf bereits vorhandenes Wissen aufbauen konnte. Gleichzeitig war es eine angemessene Aufgabe, die mich genug forderte, um die Motivation hoch zu halten.

Die Zusammenarbeit im Team ist eine der wichtigsten Erfahrungen, die ich in der Laufbahn der HTL bekommen habe. Es zeigt, wie wichtig die Kommunikation zwischen den Teammitgliedern ist und wie man sich gegenseitig motivieren kann.

Ich habe viele meiner bereits vorhandenen Kenntnisse vertiefen können, jedoch auch viel Neues über Bibliotheken wie OpenCV und Tesseract gelernt. Mit bereits bekannten und

vertrauten Programmiersprachen zu arbeiten erleichterte die Entwicklung ungemein, da man bei Fehlermeldungen leichter erkennt, wo das Problem liegt und wie man es lösen kann.

Ich habe jedoch auch gelernt, dass ich in meiner Zukunft den Berufsweg eines Entwicklers vorraussichtlich nicht einschlagen werde, da diese Arbeit mich an manchen Punkten beinahe zur Verzweiflung brachte. Zusammenfassend bin ich mit dem Ergebnis sowie der Leistung des Teams sehr zufrieden.

8 Zusammenfassung / Ausblick

8.1 Zusammenfassung [DH]

Unser Projekt ermöglicht die Erweiterung eines Garagentors. Es ist möglich, das Tor mithilfe von APERTA mit einer Kennzeichenerkennung auszustatten. Das System kann selbstständig angebracht und über den Client verwaltet werden. Um sich ein Set oder Einzelteile kaufen zu können, gibt es einen Webshop. Unser Produkt ist zudem modular erweiterbar und bietet für jeden Kunden und jede Kundin das Richtige.

8.2 Ausblick [DH]

Es gibt viele Möglichkeiten, das Projekt weiterzuentwickeln. Die Verwaltung ist derzeit nur über den Browser möglich. Es könnte eine App für Android und IOS entwickelt werden, die die gesamte Verwaltung noch einfacher und bequemer macht. Auch die verfügbaren Bauteile können noch erweitert werden. Garagensysteme bieten meist eine Fernbedienung, um das Garagentor auf Knopfdruck zu öffnen. Das wäre auch für APERTA eine Möglichkeit. Natürlich könnte die Software mit einer KI ausgestattet werden. Diese könnte lernen, die Kennzeichen aus verschiedenen Winkeln zu lesen oder auch die Schemata der Kennzeichen aus anderen Ländern zu erkennen. Das wäre ein wichtiger Schritt, um das Produkt international anbieten zu können.

Literaturverzeichnis

- [1] S. Austria, „Vorläufiger Fahrzeug-Bestand am 28.02.2022,” Zuletzt besucht am 25.03.2022. Online verfügbar: https://www.statistik.at/wcm/idc/idcplg?IdcService=GET_PDF_FILE&RevisionSelectionMethod=LatestReleased&dDocName=062059
- [2] M. Dirk Srocke, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.cloudcomputing-insider.de/was-ist-eine-web-app-a-644292/>
- [3] itwissen.info, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.itwissen.info/Web-Client-web-client.html>
- [4] L. Koch, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.idesis.de/desktop-vs-web-anwendung/>
- [5] P. Böllhoff, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://kruschecompany.com/de/framework-vs-bibliothek/>
- [6] A. Domin, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://t3n.de/news/library-vs-framework-unterschiede-1022753/>
- [7] A. Cardona, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://oh-hey-dre.medium.com/what-is-the-difference-between-a-framework-and-a-library-df58cb24888>
- [8] S. Daityari, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.codeinwp.com/blog/angular-vs-vue-vs-react/>
- [9] datenbanken verstehen.de, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.datenbanken-verstehen.de/lexikon/model-view-controller-pattern/>
- [10] sawakinome.com, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.sawakinome.com/articles/technology/what-is-the-difference-between-mvc-and-mvvm.html#MVC>
- [11] R. Boehm, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://angular.de/artikel/angular-tutorial-deutsch>
- [12] ——, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://angular.de/artikel/angular-tutorial-deutsch/angular-platform-overview.png>
- [13] D. Wahlin, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://p7x7q5i4.rocketcdn.me/wp-content/uploads/2019/01/what-is-in-a-component-class.png>
- [14] ——, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://t2informatik.de/blog/softwareentwicklung/die-5-wesentlichen-vorteile-von-angular-und-typescript/>
- [15] affdu.com, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.affdu.com/de/angular-features.html>

- [16] K. M, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.lise.de/blog/artikel/angular-material-und-mehrsprachigkeit-ein-umfassendes-how-to/>
- [17] J. Österle, Zuletzt besucht am 3.04.2022. Online verfügbar: https://blog.brickmakers.de/react-eine-einführung-in-fünf-minuten?utm_campaign=Digitale%20Transformation&utm_source=Medium&utm_medium=Medium%20React%20Einführung
- [18] C. McKnight, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.digitalclaritygroup.com/single-page-application-make-sense/>
- [19] K. Stoll, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://t3n.de/news/react-facebook-623999/>
- [20] reactjs.org, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://de.reactjs.org/docs/jsx-in-depth.html#gatsby-focus-wrapper>
- [21] ——, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://de.reactjs.org/docs/components-and-props.html#gatsby-focus-wrapper>
- [22] lernen.react.js.dev, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://lernen.react.js.dev/die-grundlagen/komponenten-in-react>
- [23] B. Hytrek, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://codepen.io/bjhytrek/pen/KNOwNX?editors=1010>
- [24] G. Singhal, Zuletzt besucht am 3.04.2022. Online verfügbar: [https://www.pluralsight.com/guides/how-to-use-the-map\(\)-function-to-export-javascript-in-react](https://www.pluralsight.com/guides/how-to-use-the-map()-function-to-export-javascript-in-react)
- [25] reactjs.org, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://de.reactjs.org/docs/react-without-jsx.html>
- [26] PaulHalliday, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.digitalocean.com/community/tutorials/react-typescript-with-react>
- [27] P. Mandler, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.micromata.de/blog/react/>
- [28] mothership.de, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.mothership.de/blog/einfuehrung-react-hooks>
- [29] reactjs.org, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://reactjs.org/docs/hooks-effect.html>
- [30] IBM, „ACID-Eigenschaften für Transaktionen,” Zuletzt besucht am 28.03.2022. Online verfügbar: <https://www.ibm.com/docs/de/cics-ts/5.4?topic=processing-acid-properties-transactions>
- [31] IONOS, Zuletzt besucht am 28.03.2022. Online verfügbar: <https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysql/>
- [32] O. Team, Zuletzt besucht am 29.03.2022. Online verfügbar: <https://opencv.org/about/>
- [33] Wikipedia, Zuletzt besucht am 30.03.2022. Online verfügbar: https://en.wikipedia.org/wiki/Optical_character_recognition

- [34] B. Baruno, Zuletzt besucht am 30.03.2022. Online verfügbar: <https://medium.com/zeals-tech-blog/introduction-to-tesseract-ocr-84d3eff6f9df>
- [35] tesseract ocr, Zuletzt besucht am 29.03.2022. Online verfügbar: <https://tesseract-ocr.github.io/tessdoc/>
- [36] S. Hoffstaetter, Zuletzt besucht am 29.03.2022. Online verfügbar: <https://github.com/madmaze/pytesseract>
- [37] Learneroo, Zuletzt besucht am 25.03.2022. Online verfügbar: <https://www.learneroo.com/modules/9/nodes/620>
- [38] J. D. Tomislav Capan, Zuletzt besucht am 25.03.2022. Online verfügbar: <https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js>
- [39] S. CM, Zuletzt besucht am 25.03.2022. Online verfügbar: <https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/>
- [40] MongoDB, Zuletzt besucht am 25.03.2022. Online verfügbar: <https://www.mongodb.com/why-use-mongodb>
- [41] R. P. Foundation, Zuletzt besucht am 25.03.2022. Online verfügbar: <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>
- [42] W. Commons, Zuletzt besucht am 26.03.2022. Online verfügbar: https://upload.wikimedia.org/wikipedia/commons/thumb/e/ef/RaspberryPi_4_Model_B.svg/1200px-RaspberryPi_4_Model_B.svg.png
- [43] R. P. Foundation, Zuletzt besucht am 26.03.2022. Online verfügbar: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>
- [44] I. ARC, Zuletzt besucht am 26.03.2022. Online verfügbar: <https://www.industryarc.com/Report/19454/industrial-raspberry-pi-market.html>
- [45] E. Kopendium, „Raspberry Pi: GPIO - General Purpose Input Output - Grafik,” Zuletzt besucht am 28.03.2022. Online verfügbar: <https://www.elektronik-kopendium.de/sites/raspberry-pi/bilder/raspberry-pi-gpio.png>
- [46] E. Kompendium, „Raspberry Pi: GPIO - General Purpose Input OutputRaspberry Pi: GPIO - General Purpose Input Output,” Zuletzt besucht am 28.03.2022. Online verfügbar: <https://www.elektronik-kopendium.de/sites/raspberry-pi/2002191.htm>
- [47] L. Luppes, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://hackster.imgix.net/uploads/attachments/178906/HacksterLogo.png?auto=compress%2Cformat&w=900&h=675&fit=min>
- [48] Sandberg.world, Zuletzt besucht am 28.03.2022. Online verfügbar: https://cdn.sandberg.world/products/images/lg/133-97_lg.jpg
- [49] R. P. Foundation, Zuletzt besucht am 26.03.2022. Online verfügbar: https://images.prismic.io/rpf-products/ffa68a46-fd44-4995-9ad4-ac846a5563f1_Camera%20V2%20Hero.jpg?ixlib=gatsbyFP&auto=compress%2Cformat&fit=max&q=50&w=600&h=400
- [50] AZ-Delivery, „HD44780 2004 Blaues LCD Display Bundle 4x20 Zeichen mit I2C Schnittstelle Datenblatt,” Zuletzt besucht am 27.03.2022. Online verfügbar: https://cdn.shopify.com/s/files/1/1509/1638/files/HD44780_2004_Blaues_

LCD_Display_mit_I2C_Serielle_Schnittstelle_Bundle_Datenblatt_AZ-Delivery_Vertriebs_GmbH.pdf?v=1591270653

- [51] ——, „HD44780 2004 LCD Display Bundle 4x20 Zeichen mit I2C Schnittstelle,” Zuletzt besucht am 26.03.2022. Online verfügbar: https://cdn.shopify.com/s/files/1/1509/1638/products/1.Main_1x_HD447802004LCDDisplayBundleGrun4x20ZeichenmitI2CSchnittstelle_500x.jpg?v=1597657362
- [52] Homecomputermuseum.de, „Relais,” Zuletzt besucht am 28.03.2022. Online verfügbar: <https://www.homecomputermuseum.de/technik/rechnen-mit-strom/relais/>
- [53] ——, Zuletzt besucht am 28.03.2022. Online verfügbar: https://www.homecomputermuseum.de/fileadmin/user_upload/Technik/relais.png
- [54] S. Shawn, Zuletzt besucht am 28.03.2022. Online verfügbar: <https://www.seeedstudio.com/blog/2020/02/25/which-raspberry-pi-programming-language-should-you-use-comparison-guide/>
- [55] az delivery.de, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.az-delivery.de/en/products/rfid-set>
- [56] ——, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.az-delivery.de/en/products/rfid-set>
- [57] ——, Zuletzt besucht am 3.04.2022. Online verfügbar: https://cdn.shopify.com/s/files/1/1509/1638/files/RZE278_4.PDF?v=1642610317
- [58] M. D. S. . F. Karlstetter, Zuletzt besucht am 29.03.2022. Online verfügbar: <https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/>
- [59] P. Documentation, Zuletzt besucht am 29.03.2022. Online verfügbar: <https://docs.python.org/3/library/threading.html>
- [60] V. Nohejl, Zuletzt besucht am 30.03.2022. Online verfügbar: <https://www.industry-era.com/images/article/vCPU.jpg>
- [61] ——, Zuletzt besucht am 30.03.2022. Online verfügbar: <https://www.industry-era.com/images/article/OCPU.jpg>
- [62] ——, Zuletzt besucht am 30.03.2022. Online verfügbar: <https://www.industry-era.com/Cloud-Services-Pricing.php>
- [63] M. Alhasan, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.fellow-consulting.de/angular-komponenten/>
- [64] R. Boehm, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://angular.de/artikel/angular-tutorial-deutsch/angular-component-service-simple-example.png>
- [65] ——, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://angular.de/artikel/angular-tutorial-deutsch/#komponenten-und-services>
- [66] T. Riedl, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://hellocoding.de/blog/coding-language/allgemein/sensible-daten-env-datei>
- [67] Maxime, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://dev.to/deammer/loading-environment-variables-in-js-apps-1p7p>

- [68] reactjs.org, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://de.reactjs.org/docs/faq-structure.html>
- [69] A. Sridhar, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.freecodecamp.org/news/quick-guide-to-understanding-and-creating-reactjs-apps-8457ee8f7123/>
- [70] ichi.pro, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://ichi.pro/de/einfaches-react-js-routing-fur-anfanger-222365993746202>
- [71] cssinjs.org, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://cssinjs.org/?v=v10.9.0>
- [72] O. Isonen, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://codesandbox.io/s/z21lpmvv33?file=/index.js:0-360>
- [73] J. Hellwig, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://kulturbanause.de/blog/einfuhrung-in-das-flexbox-modell-von-css/>
- [74] S. Kapoor, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://medium.com/@sahil.kapoor440/css-flexbox-a-developers-rejoice-to-placing-elements-on-a-website-29efaa3aa9df>
- [75] J. Hellwig, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://kulturbanause.de/blog/css-grid-layout-module/>
- [76] A. Abuelgasim, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.creativebloq.com/advice/a-comprehensive-guide-to-using-css-grid>
- [77] commercejs.com, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://commercejs.com/blog/introducing-commerce-js/>
- [78] freshbooks.com, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.freshbooks.com/hub/payments/what-is-stripe>
- [79] Zeta-Producer.com, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://blog.zeta-producer.com/div-container/>
- [80] ryte.com, Zuletzt besucht am 3.04.2022. Online verfügbar: https://de.ryte.com/wiki/Anchor_Tag
- [81] mediaevent.de, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.mediaevent.de/html/form.html>
- [82] tutorialspoint.com, Zuletzt besucht am 3.04.2022. Online verfügbar: https://www.tutorialspoint.com/de/html/html_input_tag.htm
- [83] lippke.li, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://lippke.li/ngfor-provider/>
- [84] happy angular.de, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://happy-angular.de/bindings-kommunikation-zwischen-template-und-logik-folge-4/>
- [85] mozilla.org, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://developer.mozilla.org/de/docs/Web/HTML/Element/section>
- [86] B. Landmesser, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://www.cocomore.de/blog/erstellung-und-nutzung-eines-angular-2-services>

- [87] runebook.dev, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://runebook.dev/de/docs/angular/api/common/http/httpclient>
- [88] A. Srivastava, „How to convert color image to grayscale in OpenCV,” Zuletzt besucht am 1.04.2022. Online verfügbar: <https://dev-akash.github.io/images/grayscale-conversion.PNG>
- [89] S. Mohan, Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/ezoimgfmt/953894.smushcdn.com/2611031/wp-content/uploads/2020/12/Gaussian-Filter-Formula.jpg?lossy=0&strip=1&webp=1&ezoimgfmt=rs:247x68/rscb1/ng:webp/ngcb1>
- [90] —, Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/ezoimgfmt/953894.smushcdn.com/2611031/wp-content/uploads/2020/12/Gaussian-Filter-Formula-2.jpg?lossy=0&strip=1&webp=1&ezoimgfmt=rs:251x67/rscb1/ng:webp/ngcb11>
- [91] —, Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/ezoimgfmt/953894.smushcdn.com/2611031/wp-content/uploads/2020/12/Bilateral-Filtering-in-Python-OpenCV.jpg?lossy=0&strip=1&webp=1&ezoimgfmt=ng:webp/ngcb1>
- [92] —, Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/ezoimgfmt/953894.smushcdn.com/2611031/wp-content/uploads/2020/12/Bilateral-Filtering-in-Python-OpenCV-Formula2.jpg?lossy=0&strip=1&webp=1&ezoimgfmt=ng:webp/ngcb1>
- [93] —, Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/ezoimgfmt/953894.smushcdn.com/2611031/wp-content/uploads/2020/12/Bilateral-Filtering-in-Python-OpenCV-Formula3.jpg?lossy=0&strip=1&webp=1&ezoimgfmt=rs:154x59/rscb1/ng:webp/ngcb1>
- [94] —, Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/ezoimgfmt/953894.smushcdn.com/2611031/wp-content/uploads/2020/12/Bilateral-Filtering-in-Python-OpenCV-Formula4.jpg?lossy=0&strip=1&webp=1&ezoimgfmt=rs:154x59/rscb1/ng:webp/ngcb1>
- [95] S. Paris, „Fixing the Gaussian Blur”:the Bilateral Filter,” Zuletzt besucht am 1.04.2022. Online verfügbar: https://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.pdf
- [96] S. Mohan, „Bilateral Filtering in Python OpenCV with cv2.bilateralFilter(),” Zuletzt besucht am 1.04.2022. Online verfügbar: <https://machinelearningknowledge.ai/bilateral-filtering-in-python-opencv-with-cv2-bilateralfilter/>
- [97] O. Docs, „Non-maximum Suppression,” Zuletzt besucht am 2.04.2022. Online verfügbar: <https://docs.opencv.org/4.x/nms.jpg>
- [98] —, „Hysteresis Thresholding,” Zuletzt besucht am 2.04.2022. Online verfügbar: <https://docs.opencv.org/4.x/hysteresis.jpg>
- [99] —, Zuletzt besucht am 2.04.2022. Online verfügbar: <https://docs.opencv.org/4.x/canny1.jpg>
- [100] —, Zuletzt besucht am 2.04.2022. Online verfügbar: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html

- [101] ——, „Contour Approximation Method,” Zuletzt besucht am 2.04.2022. Online verfügbar: <https://docs.opencv.org/4.x/none.jpg>
- [102] A. Rosebrock, Zuletzt besucht am 2.04.2022. Online verfügbar: <https://github.com/PyImageSearch/imutils/blob/master/imutils/convenience.py>
- [103] P. Adrian Rosebrock, „PSM 11. Sparse Text: Find as Much Text as Possible in No Particular Order,” Zuletzt besucht am 2.04.2022. Online verfügbar: <https://pyimagesearch.com/2021/11/15/tesseract-page-segmentation-modes-psms-explained-how-to-improve-your-ocr-accuracy/>
- [104] taurus sicherheitstechnik.at, Zuletzt besucht am 3.04.2022. Online verfügbar: <https://image.jimcdn.com/app/cms/image/transf/none/path/sad7daa6f3a52d307/image/iaa459f5d13f4a424/version/1557222324/image.png>

Abbildungsverzeichnis

1	Rendered Prototyp	I
2	Gerenderter Prototyp	II
3	Unterschied zwischen der Funktionsweise eines Frameworks und einer Library	5
4	Interesse im zeitlichen Verlauf	6
5	Aufbau des MVC-Patterns	7
6	Aufbau des MVVM-Patterns	8
7	Ökosystem von Angular	10
8	Aufbau einer Komponente	10
9	Beispielcode einer Service-Klasse	11
10	Aufbau einer Komponente	16
11	Aufbau einer Komponente	16
12	Beispiel für die Verwendung des Single-Responsibility-Prinzips	21
13	Darstellung des 'One Way Data Flow'	22
14	Übersicht weiterer React-Hooks	24
15	Prozessdiagramm der Optische Zeichenerkennung	28
16	Komponenten eines Raspberry Pi	31
17	Belegung der GPIOs eines Raspberry Pi Model 4B / Raspberry Pi Zero	33
18	Abbildung eines Nummernfelds	34
19	Webcam mit gleichen Spezifikationen	34
20	Raspberry Pi Camera Module 2	34
21	Display direkt angeschlossen / Display über I2C Adapter angeschlossen	35
22	Funktionsweise eines Relais als Schema	36
23	RFID-Kit der Firma AZ-Delivery	38
24	Schaltbild des RFID-Lesers	38
25	Profilmanagement	39
26	Kennzeicheneinstellungen	40
27	Screenshot der „Packages“-Unterseite des Webshops	40
28	Screenshot der „Components“-Unterseite des Webshops	40
29	Screenshot der „About“-Unterseite des Webshops	41
30	Visualisierung von vCPUs	43
31	Visualisierung von OCPUs	44
32	Systemarchitektur	46
33	Aufbau eines Angular Projekts	47
34	Aufbau einer Angular Komponente	49
35	Auslagern einer Service Klasse	50
36	Dependency Injection	50
37	Ordnerstruktur der React-Applikation	51
38	Ordnerstruktur der React-Applikation	52
39	Komponenten der React-Applikation und Aufbau der „ProductDetail“-Komponente	52
40	Darstellung der Flex-Direction	55

41	Darstellung eines Responsive-Designs durch CSS-Grids	56
42	APERTA-Produkte in Commerce.js	57
43	Stripe-Berichtsübersicht	58
44	Loginseite	64
45	Aufgenommenes Bild vs. Graustufenbild	76
46	Ein mit Gaußscher Unschärfe gefiltertes Bild	77
47	Gauß-Kernel	77
48	Bilaterale Filterung Formel	78
49	Formel des Normalisierungsfaktors W_p	78
50	Term 1	78
51	Term 2	78
52	Beeinflussung durch Veränderung der Parameter σ_s und σ_r	79
53	Non-Maximum-Suppression	81
54	Hysterese-Schwellenwertbildung	81
55	Ergebnis der Kantenerkennung	82
56	Unterschied von	83
57	Umsetzung von der Firma Taurus	91
58	Aperta zerlegt in Module	92

Tabellenverzeichnis

1	Übersicht der Komponenten des Raspberry Pi [43]	31
2	Technische Daten der Oracle VM Instanz	43

Quellcodeverzeichnis

1	Hinzufügen von Angular Materials	14
2	Möglicher Aufbau einer Komponente in einer JavaScript-Funktion	15
3	Möglicher Aufbau einer Komponente in einer JavaScript-Klasse	15
4	JSX-Code	17
5	JSX-Code in gewöhnliches JavaScript kompiliert	17
6	Komponente 'Welcome' die Props 'props' erwartet	17
7	Aufruf der Komponente 'Welcome' mit Übergabe des props 'name'	18
8	Ausgabe der Komponente 'Welcome'	18
9	Beispiel Code eines Click Counters mit der Nutzung on States	18
10	Beispiel Code für die Nutzung der map()-Methode	19
11	Beispiel Code 'Hello World' mit JSX	19
12	'Hello World' mit einfachem JavaScript	20
13	Beispiel-Code eines Click Counters in TSX label	20
14	Code-Beispiel für Nutzung des State Hooks	23
15	Code-Beispiel für Nutzung des Effect Hooks	23
16	Funktionsweise von Multiprocessing	42
17	Aufbau eines Kennzeichen in der Datenbank	44
18	Aufbau einer Nummernfeld-Kombination in der Datenbank	45
19	Aufbau einer NFC-Karte in der Datenbank	45
20	app.module.ts	47
21	Routing der Komponenten in der app.module.ts	48
22	app.component.html	48
23	Source-Code des Routers	53
24	Beispiel Source-Code für die Verwendung von JSS	54
25	HTML-Code vom Aufbau einer Flexbox	55
26	CSS-Code zur Aktivierung der Flexbox	55
27	HTML-Code vom Aufbau eines Grids	56
28	CSS-Code zur Aktivierung des Grids	56
29	Navigationsleiste Komponente	59
30	Profil Komponente	60
31	KeyPad Model	63
32	Login Komponente	63
33	profile.component.ts	65
34	http.service.ts	66
35	Konfiguration der Datenbankanbindung	75
36	Abfrage der Kennzeichen	75
37	Maskieren der Kennzeichenregion und Abspeichern in neuem Bild	84
38	Maskieren der Kennzeichenregion und Abspeicher in neuem Bild	84
39	Abfrage der Kennzeichen aus der Datenbank	84
40	Überprüfung auf Gleichheit der beiden Strings	85
41	Ausgabe bei korrekter Eingabe der Kombination auf dem Nummernfeld	85
42	Ansteuerung des Relais	86
43	checkNumpad.py	87

44	Numpad.py	87
45	Lesen der RFID-Karte	89
46	Überprüfung der RFID-Karten-ID	89

Anhang