Fakultät Informatik
Fachbereich Angewandte Informatik

Bachelor-Thesis

Entwicklung einer webbasierten Umfrage von typischen Personenbewegungen in öffentlichen Räumen am Beispiel Supermarkt zur Erzeugung entsprechender Wahrscheinlichkeitsverteilungen.

Name: Yue Zhang

Matrikelnummer: 297754

Erstkorrektor: Prof. Dr. Rebekka Axthelm

Zweitkorrektor: Prof. Dr. Doris Bohnet

Abgabetermin: 28.08.2022

Konstanz, 28.08.2022

Bearbeitungszeitraum: 01.05.2022 bis 28.08.2022

Ehrenwörtliche Erklärung

Hiermit erkläre ich *Yue Zhang, geboren am 10.07.1996 in Shanghai, China*,

(1) dass ich meine Bachelorarbeit mit dem Titel:

Entwicklung einer webbasierten Umfrage von typischen Personenbewegungen in öffentlichen Räumen am Beispiel Supermarkt zur Erzeugung entsprechender Wahrscheinlichkeitsverteilungen.

unter Anleitung von Prof. Dr. Rebekka Axthelm selbständig und ohne fremde Hilfe angefertigt habe und keine anderen als die angeführten Hilfen benutzt habe;

(2) dass ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

(3) dass die eingereichten Abgabe-Exemplare in Papierform und im PDF-Format vollständig übereinstimmen.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 28.08.2022

(Unterschrift)

Inhaltsverzeichnis

Abkürzungsverzeichnis

API Application programming interface

HTTP Hypertext Transfer Protocol

SSH Secure Shell

REST Representational State Transfer

CMS Content-Management-Systeme

HTML Hypertext Markup Language

CSS Cascading Style Sheets

I/O Input/Output

1 Einleitung

1.1 Motivation

Im Bereich der Innenarchitektur ist Zirkulationsdesign sehr wichtig. Besonders interessant ist dann wie sich Personen in/auf sollen Designs tatsächlich bewegen. In einem Ausstellungsraum wie einem Supermarkt oder einer Kunstgalerie trägt ein übersichtliches Zirkulationssystem nicht nur zu einer angenehmen Besuchs- oder Einkaufserfahrung bei, sondern hilft auch dem Gebäudemanager, die Einrichtung im Sinne der Sicherheit der Besuchern zu optimieren. In dieser Arbeit werden wir den Supermarkt als Beispiel verwenden. Mit Hilfe einer Umfrage können wir Informationen über die regelmäßige Routenplanung der Kunden sammeln, und mit Hilfe von Algorithmen können wir die Unterschiede in der Routenplanung zwischen verschiedenen Layouts und die Verweildauer der Nutzer vergleichen und analysieren. (Hier sehe ich Ergebnisse zweier Umfragen mit verschiedenen Raumlayouts.) Die erzielten Ergebnisse sollen zeigen, dass man auf diese Weise anonyme Daten von Personenbewegungen erhalten kann, ohne diese aufwendig und teuer zu messen.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, ein Umfragesystem zu entwerfen, um die Routenplanung und Verweildauer von Kunden in einem Supermarkt als Beispiel durch Visualisierung der Route und Zoneneinteilung umzufragen. Die Antworten auf der Umfrage werden im Hintergrund gesammelt und aufbereitet und dem Admin visuell präsentiert. Die Verwaltung kann das Layout über das Backend aktualisieren und ändern. Dies wird dem Befragten zusammen mit einem Querverweis auf den Fragebogen mitgeteilt, und beide Antworten werden erfasst und verglichen. Das System basiert auf einem Projekt, das in der Node.js-Umgebung erstellt und über Technologien wie pm2 auf einem Online-Server bereitgestellt wird. Die gesammelten Nutzerdaten werden in einer Online-Datenbank gespeichert, damit das System sie abrufen kann.

1.3 Aufbau der Arbeit

Im Anschluss an die Einleitung folgt Kapitel 2, das sich auf die theoretischen Kenntnisse konzentriert, die in dieser Arbeit behandelt werden und die in der anschließenden Darstellung des Projekts anhand von Beispielen näher erläutert werden. In Kapitel 3 werden die Gestaltungsideen des Systems vorgestellt. Dieses Kapitel befasst sich mit der Analyse der Anforderungen, dem Aufbau der Projektstruktur, dem Ansatz für die Implementierung der Funktionen und der Frage, wie das Projekt schließlich auf dem

Online-Server ausgeführt wird. In diesem Bereich liegt der Schwerpunkt auf der Implementierung der Dateninteraktion durch Kapselung der Datenbankfunktionalität. Es wird auch behandelt, wie man Routen erstellt, um zwischen den Seiten zu wechseln, und, was noch wichtiger ist, wie man die Back-End-API geschrieben und implementiert, um mit der Front-End-Schnittstelle zu interagieren. Darüber hinaus werden die Methoden zur Visualisierung der Endergebnisse vorgestellt. In Kapitel 4 werden die Endergebnisse vorgestellt und auch die im Rahmen des Projekts erreichte Funktionalität. Die bei der Durchführung des Projekts aufgetretenen Probleme und die zu ihrer Lösung angewandten Methoden. Das abschließende Kapitel fasst das gesamte Projekt zusammen und stellt einige Möglichkeiten sowie eine Zukunftsvision für einzelne Funktionen vor.

2 Grundlagen

In diesem Abschnitt werden einige der theoretischen Kenntnisse, die auf dieses System angewandt wurden, sowie die wichtige Technologien, die bei der Entwicklung dieses Systems verwendet wurden, vorgestellt.

2.1 Grundprinzipien der Webentwicklung

Webentwicklung ist der Entwurf und die Implementierung von Software für das World Wide Web. Die Entwicklung umfasst die folgenden Bereiche: Websites, Webanwendungen und Webdienste. Aus der Sicht der traditionellen Softwareentwicklung sind die Techniken zur Erstellung einer Web-Architektur, zur Verbindung mit Datenbanken und zur Implementierung von grafischen Benutzeroberflächen auch für die Webentwicklung wichtig. Die heutige Webentwicklung kann in client- und serverseitige Technologien unterteilt werden.

Die auf der Client-Seite verwendeten Programmiersprachen, Bibliotheken und Frameworks werden im Browser des Besuchers ausgeführt, während die serverseitigen Programme und die Logik auf einem vom Besucher entfernten Server laufen.

Die clientseitige Entwicklung stützt sich in der Regel auf die Auszeichnungssprache HTML (Hypertext Markup Language), die Designsprache CSS (Cascading Style Sheets) und die Skriptsprache JavaScript. HTML ist für die Entwicklung der allgemeinen Struktur der Website verantwortlich und bestimmt, welcher Inhalt wo im Browser angezeigt wird. CSS ist für die Gestaltung und das Layout der Website verantwortlich. Ein Teil der komplexen Logik und der Interaktion zwischen den Anwendungen wird von JavaScript übernommen.

Auf der Serverseite wird eine größere Vielfalt an verschiedenen Programmiersprachen und Frameworks verwendet. Unter diesen Sprachen führt PHP die Liste der am häufigsten verwendeten serverseitigen Programmiersprachen an. Das liegt daran, dass die meisten Content-Management-Systeme (CMS) auf dieser Sprache erstellt werden. Neben PHP ist es auch möglich, in Sprachen wie Java, Python zu entwickeln. Vor allem in den letzten Jahren wurde JavaScript nicht nur auf der Client-Seite, sondern auch zunehmend auf der Server-Seite eingesetzt. An dieser Stelle müssen wir über die Auswirkungen sprechen, die Node.js als in JavaScript geschriebene Laufzeitumgebung auf die Webentwicklung hatte. In den nächsten Unterabschnitten werden die Theorie von Node.js und die Funktionen des Express-Anwendungsframeworks, auf dem es basiert, vorgestellt.

2.2 Node.js

2.2.1 JavaScript-Laufzeitumgebung

Node.js wird auf der offiziellen Website als ‘eine JavaScript-Laufzeitumgebung, die auf der Chrome V8-Engine basiert‘ beschrieben. Während JavaScript oft als etwas bezeichnet wird, das im Browser ausgeführt werden muss, bietet Node.js eine serverseitige Umgebung, die es ermöglicht, JavaScript auch auf der Serverseite auszuführen. Außerdem verfügt sie über eine Reihe von Merkmalen, die sie zur bevorzugten Sprache für die Backend-Entwicklung in dieser Arbeit machen, und zwar aus Gründen, die in den folgenden Kapiteln erläutert werden.

Um Node.js zu verstehen, müssen wir zunächst wissen, was die v8-Engine ist. Node.js verdankt einen großen Teil seiner Entstehung der Einführung der v8-Engine. Es handelt sich um eine Open-Source-JavaScript-Engine, die von Google für seinen Browser Chrome entwickelt wurde und deren Quellcode ist in C++ geschrieben. Es bietet Funktionen wie ‘Embedding‘’, um den Entwickler zu helfen, JavaScript effizienter zu kompilieren, und fügt Funktionen aus C++ hinzu, wie z. B. “Read”, das an einen in C++ geschriebenen “Read-Callback” gebunden werden kann, um die JavaScript kann auch die Fähigkeit haben, zu lesen.

Zweitens ist Node.js eine serverseitige Technologie, während in der Vergangenheit viele serverseitige Implementierungen in PHP, Python und anderen Sprachen vorgenommen wurden. Die Einführung von Node.js bietet Entwicklern auch die Möglichkeit, dieselbe Sprache für Front- und Backend zu verwenden und so das Modell und Funktionen zu vereinheitlichen.

Node.js hat auch einige Funktionen, wie z. B. “Event-driven”, was bedeutet, dass Client-Anfragen zum Verbindungsaufbau, zur Übermittlung von Daten usw. Ereignisse auslösen. In Node kann immer nur eine Event-Callback Funktion ausgeführt werden. Es ist jedoch möglich, mitten in der Ausführung einer Ereignis-Callback Funktion andere “Event” zu behandeln (z. B. eine neue Benutzerverbindung) und dann zurückzukehren, um mit der Ausführung der ursprünglichen Ereignis-Callback Funktion fortzufahren. Fast die Hälfte des zugrundeliegenden Codes von Node wird zum Aufbau der Ereigniswarteschlange und der Warteschlange für die Rückruffunktionen verwendet. Die Verwendung eines ereignisgesteuerten Ansatzes für die Serverplanung bedeutet, dass ein einzelner Thread für mehrere Aufgaben verwendet wird.

Das folgende Diagramm 2.1 stellt das “Event-driven” Modell kurz dar.

Darüber hinaus ist auch die blockierungsfreie I/O(Input/Output) ein wichtiges Merkmal. Bei einem herkömmlichen Single-Thread-Verarbeitungsmechanismus würde der gesamte Thread während des Zugriffs auf die Datenbank pausieren und auf die Rückgabe der Ergebnisse durch die Datenbank warten, wodurch das gesamte Programm weniger effizient wäre. In Node.js hingegen folgt auf die Ausführung des Codes für den Datenbankzugriff unmittelbar die Ausführung des nachfolgenden Codes, wobei die Da-

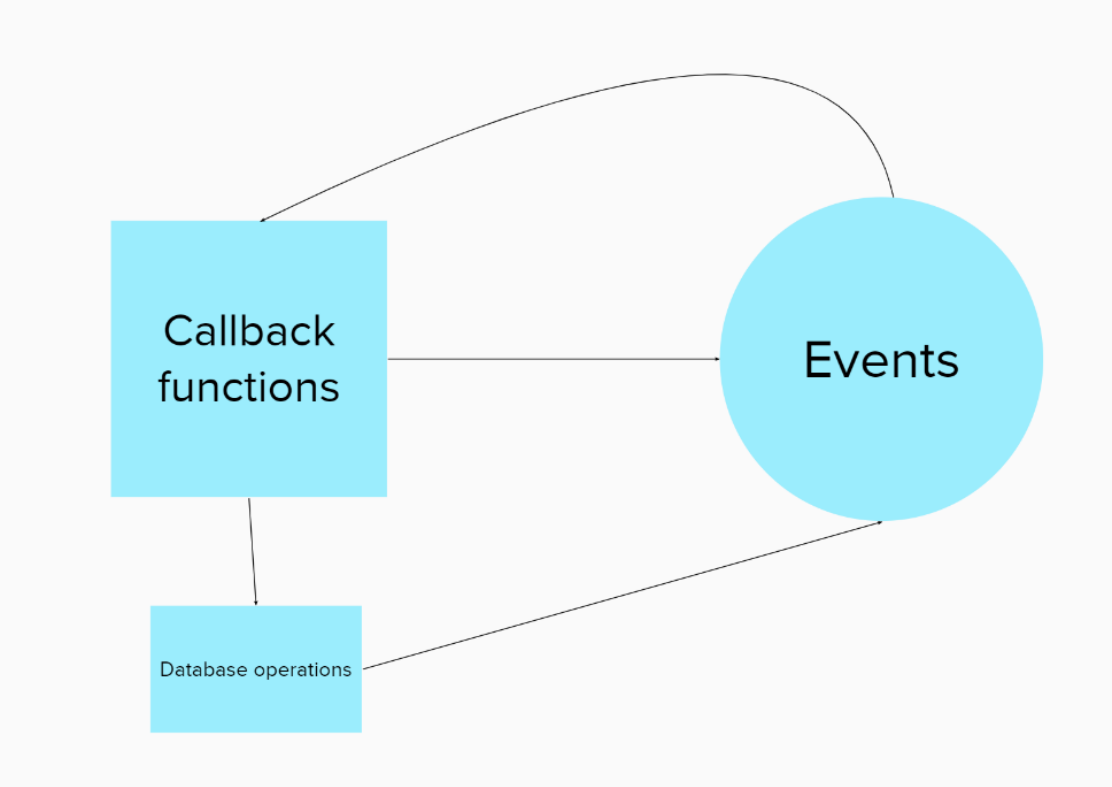

Abbildung 2.1: Event-driven Modell

tenbankrückgabe in einer Callback-Funktion platziert wird, wodurch die Effizienz der Programmausführung verbessert wird. Diese Funktion ist auch für die Gestaltung des Systems sehr wichtig, da die Datenbank häufig zum Speichern und Aufrufen von Daten verwendet wird. Diese Funktion ist auch für das System wichtig, da sie in späteren Abschnitten über die Durchführung von Datenbankabfragen und Speicheroperationen beschrieben wird.

2.2.2 Express Framework

Das Express Framework ist das bekannteste und beliebteste Backend-Entwicklungsframework in Node.js. Es bietet eine Reihe starker Funktionen und eine Fülle von HTTP-Tools, die bei der Erstellung einer Vielzahl von Webanwendungen helfen. Zu den wichtigsten Funktionen des Frameworks gehört zuerst die Möglichkeit, Middleware einzurichten, die auf HTTP-Anfragen reagiert. Die zweite ist die Möglichkeit, Routing-Tabellen zu definieren, um verschiedene HTTP-Anfragen durchzuführen. Die dritte ist die Möglichkeit, HTML-Seiten durch Übergabe von Parametern an die Vorlage zu rendern.

In Express verwenden wir den folgenden Code, um die Callback-Funktionen aufzurufen, wobei “request” und “response” die Daten sind, die das Objekt verwendet, um die Anfrage und die Antwort zu verarbeiten.

Express

```
app.get('/', function (req, res) {
    // —
})
```

Das Request stellt die HTTP-Anfrage dar, einschließlich des Strings, der Parameter usw. Das Response, die an den Client gesendet werden, wenn eine Anfrage vom Server empfangen wird.

In den folgenden Kapiteln werde ich zeigen, wie das System diese Express-Grundlagen nutzt, um Funktionen zu konstruieren, die eine serverseitige und clientseitige Datenkommunikation ermöglichen.

2.3 Interaktion mit Daten

2.3.1 RESTful API

REST (Representational State Transfer) ist eine Softwarearchitektur für das World Wide Web, die von Dr.Roy Thomas Fielding in seiner Dissertation im Jahr 2000 vorgeschlagen wurde. Sein Hauptzweck ist die Erleichterung des Informationsaustauschs zwischen verschiedenen Programmen und Softwareprogrammen über das Internet.

REST ist eine Reihe von Einschränkungen und Eigenschaften, die auf der Grundlage von HTTP festgelegt wurden, einem Software-Konstruktionsstil, der für die Bereitstellung von World Wide Web-Diensten entwickelt wurde, und Webdienste, die diesem Architekturstil entsprechen, werden als “RESTful” bezeichnet.

Wie der Name schon sagt, handelt es sich bei der RESTful API um eine Anwendungsschnittstelle, die nach dieser Architekturspezifikation geschrieben wurde und die Interaktion mit RESTful Web Services unterstützt. Wenn ein Client eine Anfrage über die RESTful-API stellt, wird eine Darstellung des Ressourcenstatus an den Serverseite, die die Daten anfragt, übergeben. Diese Informationen werden über HTTP in den Formaten JSON, HTML oder reiner Text usw. übertragen.

Die Arten von Operationen für bestimmte Ressourcen werden durch die folgenden HTTP-Funktionen angegeben, von denen einige im Folgenden aufgeführt sind.

- GET : holt eine Ressource (eine oder mehrere) vom Server.
- POST: erstellt eine neue Ressource auf dem Server.
- PUT : aktualisiert die Ressource auf dem Server (der Client stellt die vollständige Ressource nach der Änderung bereit).
- aktualisiert die Ressource auf dem Server (der Client liefert die geänderten Attribute).
- Löscht eine Ressource vom Server.

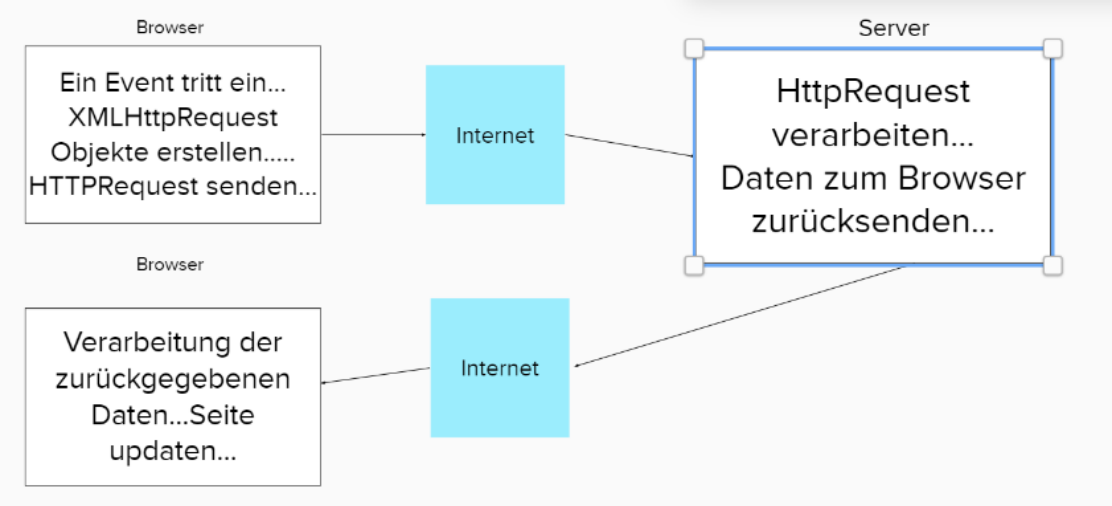

Abbildung 2.2: Ajax Funktionen

2.3.2 Ajax-Technologie

AJAX oder Äsynchronous JavaScript and XMLïst ebenfalls eine wichtige Technologie zur Erstellung schneller und dynamischer Webseiten in der Webentwicklung.

AJAX ermöglicht asynchrone Aktualisierungen einer Webseite, indem es mit dem Server im Hintergrund interagiert und die von der Serverfunktion zurückgegebenen Daten aufruft, um bestimmte Teile der Seite zu aktualisieren, ohne die gesamte Seite zu laden. Bei der Überprüfung von Benutzereingaben kann beispielsweise mit Hilfe der AJAX-Technologie festgestellt werden, ob die vom Benutzer eingegebenen Daten in der Datenbank gespeichert wurden, und das Ergebnis in Echtzeit zurückgegeben werden, um dem Benutzer mitzuteilen, dass der von ihm eingegebene Benutzername übernommen wurde.

Das folgende Diagramm 2.2erklärt kurz, wie AJAX funktioniert. Wie wir sehen, liegt der Schwerpunkt dieser Funktion auf der Interaktion zwischen dem Anfordern und dem Abrufen von Daten. Das Designkonzept in den folgenden Kapiteln wird sich wiederum darauf konzentrieren, wie dieses System die Dateninteraktion durch den Einsatz dieser Technologie bewerkstelligt.

3 Implementierung

In diesem Abschnitt werden die einzelnen Projektphasen nacheinander behandelt, vom Beginn des Entwurfsprozesses bis zur endgültigen Einführung. Dazu gehören erst die Anforderungsanalyse, der Aufbau des Backends, das Schreiben der Datenbankschnittstelle, das Design und die Verarbeitung der Seiten, die Berechnung und Analyse der Daten und die endgültige Einführung des Projekts.

3.1 Anforderungsanalyse

Im allgemeinen Systems Engineering und Software Engineering ist die Anforderungsanalyse ein primärer und sehr wichtiger Teil des Prozesses. Wenn wir ein neues Projekt erstellen oder ein existierendes Projekt oder Produkt ändern, müssen wir den Zweck, den Inhalt und die Funktionalität des Systems bestimmen. Dazu gehört es, Anforderungen aus verschiedenen Quellen zu berücksichtigen und Ausgleiche zwischen ihnen zu schaffen. Diese Softwareanforderungen werden analysiert, ermittelt und verwaltet.

Bei diesem Projekt geht es zunächst darum, die Personen zu ermitteln, für die die Umfrage macht und die Umfrage verwaltet. Was die Person, die die Umfrage ausfüllt, sieht, ist auch das, was die Umfrage auf dem Frontend darstellt. Das Frontend spielt die Rolle der Einnahme der Informationen. Die Personen, die diese Umfragen beantworten, sind also die Quelle der Informationen, die wir sammeln wollen. Gleichzeitig sind die Verwalter der Umfrage in unserem Szenario die Mitarbeiter der Supermärkte. Sie müssen die Umfragen verwalten und auf die gesammelten Informationen zugreifen.

Wir können analysieren, welche Funktionen aus der Sicht beider Benutzer erforderlich sind. Zunächst wird festgestellt, dass sie als Mitarbeiter, d.h. diejenigen, die die Umfrage leiten, die Berechtigung zum Login in das Backoffice, d.h. den Zugang zu den Backoffice-Funktionen mittels Benutzername und Passwort benötigen. Sie brauchten eine Schnittstelle, um die Karte zu verwalten, das Layout des Supermarktes zu erstellen und es in einer Datenbank zu speichern, um es später im Frontend abrufen und im Backend analysieren zu können. Um die verschiedenen Supermarktbereiche zu verwalten, die auf der Karte platziert werden müssen, benötigen wir eine Seite zum Hinzufügen oder Ändern von Supermarktbereichen.

Um die Ergebnisse des Umfragen zu erfassen, brauchen wir auch eine visuelle Seite und eine Tabelle, um diese Ergebnisse festzuhalten. Erstens gibt es eine Formularseite, auf der die in den Umfragen gesammelten Informationen in einem Formular erfasst werden, das eine textartige Darstellung bietet. Außerdem müssen wir wissen, wie sich die Zahl der Supermarktkunden zu einem bestimmten Zeitpunkt darstellt und verteilt. Wir benötigen eine Seite, die das Layout des Supermarktes zu diesem Zeitpunkt zeigt und die aktuelle Verteilung der Besucher neben jedem Supermarktregal je nach Zeitpunkt anzeigt.

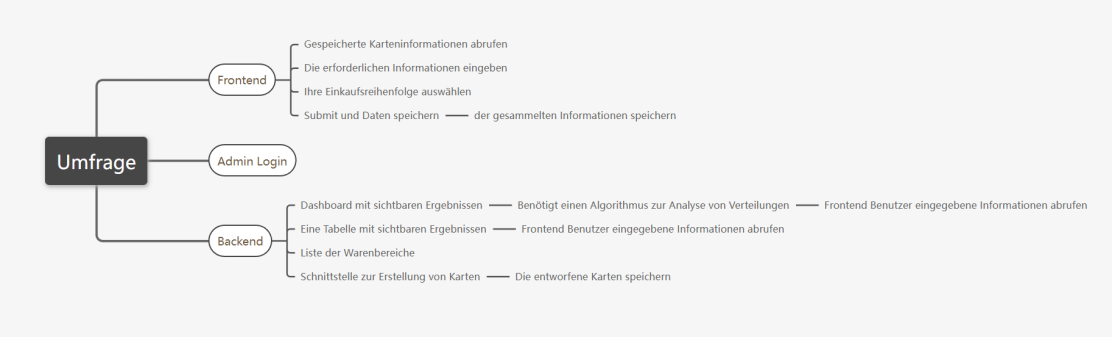

Abbildung 3.1: Anforderungsanalyse

Für diejenigen, die eine Umfrage durchführen, sind die erforderlichen Funktionen relativ einfach, und die Art und Weise der Informationserfassung ist ein sehr wichtiger Aspekt. Der Benutzer muss einige grundlegende Informationen eingeben, wie z. B. das Alter und die Tageszeit, zu der er einkaufen geht, und dann muss er vor allem die Reihenfolge der Einkäufe auswählen. Nach der Speicherung werden diese Daten in einer Datenbank für die spätere Analyse gespeichert.

Die oben genannten Anforderungen und Funktionen sind in der folgenden Mindmap dargestellt.3.1

Nachdem wir die Hauptfunktionen, die unser System erfüllen muss, identifiziert haben, haben wir auch eine Vorstellung von den Informationen, die gespeichert werden müssen, was uns auch beim Aufbau unserer Datenbank in Zukunft helfen wird.

3.2 Aufbau des Projektrahmens

3.2.1 Komponente: Backend des Projektes

3.2.1.1 Entwicklungsumgebung

Für die Umsetzung dieses Projektes wurde JavaScript mit Hilfe von Node.js gewählt. Wie im vorangegangenen Theorieteil beschrieben, ist Node.js eine Sammlung verschiedener Bibliotheken, die unabhängig voneinander entwickelt und gepflegt werden, und diese unabhängigen Bibliotheken bilden den Kern von Node.js. Dieser modulare Aufbau spiegelt sich auch in der Struktur des Projekts wider. Eine große Anzahl von Modulen kann heruntergeladen und zur Erweiterung eigener Skripte verwendet werden.

NPM, der mit Node.js gelieferte Paketmanager, enthält die weltweit größte Software-Registry. Diese bereitgestellten Module können mit dem Befehl **npm install Modulname -g** global installiert werden. Auf Linux-Systemen werden sie unter dem Pfad **/usr/local/lib** gespeichert, auf Windows-Systemen im Ordner **AppData** des jeweiligen Benutzers.

Wir geben einfach **npm install Modulname** in die Befehlsleiste ein, um es lokal im Projekt zu installieren. Die installierten Module werden wieder im Ordner **node modu-**

les gespeichert. Das Programm kann im Allgemeinen mit dem Befehl **node Skriptname.js** gestartet werden. Wird dem Knotenaufruf keine Datei übergeben, wird eine REPL gestartet. Wenn wir in der package.json-Datei den gewünschten Modulnamen, den Autor, das Git-Repository und die Lizenzeigenschaften definieren, sowie die Abhängigkeiten der Anwendung, d.h. welche Module verwendet werden. Die Datei package.json wird zuerst aufgerufen, bevor das Projekt startet, und der Quellcode wird kompiliert und das Projekt erstellt.

Der folgende Code zeigt die package.json-Datei für dieses Projekt. Sie enthält alle entwicklungsbezogenen Abhängigkeiten(devDependencies) , die für dieses Projekt erforderlich sind.

Package

```
{
  "name": "supermarkt",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "art-template": "^4.13.2",
    "babel-cli": "^6.26.0",
    "babel-core": "^6.26.3",
    "babel-preset-env": "^1.7.0",
    "body-parser": "^1.20.0",
    "consolidate": "^0.16.0",
    "core-js": "^3.23.1",
    "cors": "^2.8.5",
    "ejs": "^3.1.8",
    "express": "^4.18.1",
    "express-art-template": "^1.0.1",
    "mongoose": "^6.3.3",
    "mustache": "^4.2.0"
  },
  "engines": {
    "node": "12.1.0",
```

```
    "npm": "6.9.0"
  },
  "description": "",
  "devDependencies": {
    "@babel/core": "^7.18.5",
    "@babel/preset-env": "^7.18.2",
    "babel-loader": "^8.2.5",
    "mysql": "^2.18.1",
    "webpack": "^5.73.0"
  }
}
```

Die oben in package.json aufgeführten Modulabhängigkeiten können mit dem Befehl npm install in das Ordnerverzeichnis installiert werden.

3.2.1.2 Express Framework

Eine kurze Einführung in den Express-Rahmen wurde im vorherigen **Abschnitt 2.2.2** gegeben. Hier wird im Detail erklärt, wie der Backend-Server in diesem Projekt mit Express aufgebaut wurde.

Dies liegt daran, dass der Code normalerweise auf der Serverseite läuft und der Benutzer über die Clientseite darauf zugreift. Auf dem Server werden der Code und das Framework also in einer lokalen Serverumgebung ausgeführt. Der nachstehende Code zeigt im Grundlage einen einfachen Prozess der Erstellung einer Anwendung.

Zuerst müssen wir die Node-Bibliothek mit der **require()**-Funktion importieren. In diesem Fall werden wir das Express-Framework als Bibliothek importieren. Als Nächstes müssen wir ein Anwendungsobjekt initialisieren, das üblicherweise app genannt wird. **app.listen (3000,)** wird definiert, um den Backend-Port des Servers zu öffnen, damit der Benutzer auf den Client reagieren kann, indem er die URL [http://localhost:3000/] vom Browser aufruft.

Express

```
const express = require('express');
const app = express();

var server = app.listen(3000, function () {
var host = server.address().address;
var port = server.address().port;

console.log('App listening at localhost:3000', host, port);
});
```

```
module.exports = app;
```

3.2.1.3 Routen

Damit ist ein lokaler Zugriffspfad auf die Anwendung fertig gestellt. In der Praxis müssen wir jedoch verschiedene URLs aufrufen, um den Zugriff auf verschiedene Seiten zu ermöglichen. Einige der REST-Prinzipien und die Designphilosophie von RESTful-Schnittstellen wurden bereits im theoretischen Teil von **Kapitel 2** vorgestellt. Bei der Implementierung müssen wir also diesem Konzept folgen, um die Zugangsschnittstelle zu entwerfen. In Express wird diese Art des Zugriffs durch Routen erreicht, was den Vorteil hat, dass unsere Zugriffsanfrage blockiert wird, wenn wir auf eine Route zugreifen, die nicht angegeben ist. Die Webseite wird einen Fehlercode 404 zurückgeben, was bedeutet, dass die entsprechende Ressource nicht gefunden wurde und nicht beantwortet werden kann.

Der folgende Code zeigt die in app.js, der Hauptanwendung dieses Projekts, enthaltenen Routen. Sie sind in zwei Teile unterteilt, wobei der erste Teil die Seitenpfad in der Anzeige sind. Das heißt, der Benutzer kann den Pfad zu der Seite erhalten, indem er auf das entsprechende Suffix z.B: **'/'**, **'/login'** zugreift. Die Seiten werden separat zur Startseite oder zum Anmeldebildschirm usw. navigiert.

Routen

```
const home = require('./route/home');
const admin = require('./route/admin');
const login = require('./route/login');
const sndmap = require('./route/2ndmap')

app.use('/', home);
app.use('/2ndmap', sndmap);
app.use('/admin', admin);
app.use('/login', login);
```

Der andere Teil der Route ist die bereits im Theorieteil erwähnte REST-API, die die Verbindung zwischen der Anwendung und der Datenbank darstellt. Sie erhalten Anfragen, bearbeiten Lese- und Schreibzugriffe vom Frontend und arbeiten mit der Datenbank zusammen, um diese Anfragen zu verarbeiten. Die folgenden Pfade sind für Lesezugriffe verfügbar. Ihre Rollen werden später im entsprechenden Abschnitt ausführlich beschrieben und hier nur zu zeigen.

APIs

```
/get/classList                    /get/classListAll
```

```
/get/getPosition                    /get/admin/getCurrent
/get/updatePosition                 /get/getUserTime

/post/login                         /post/admin/insertClassList
/post/admin/updateClasslist         /post/admin/delClassList
/post/admin/insertPosition          /post/insertUserTime
/post/admin/delUserTime
```

Einige dieser APIs verwenden die get-Methode, andere die post-Methode, je nach ihrer Funktion. Die Unterschiede zwischen den beiden Express-Projekten sind die folgenden. 1. **get** holt Daten vom Server und **post** überträgt Daten an den Server. 2. **get** überträgt im Allgemeinen eine kleinere Datenmenge, während **post** eine größere Datenmenge überträgt. 3. **get** lädt die Parameter und die Daten-array in die URL, auf die im ACTION-Attribut des Formulars verwiesen wird, und die Werte müssen den Feldern des Formulars eins zu eins passen. Der Post-Mechanismus verwendet den HTTP-Post-Mechanismus, um die Felder und den Inhalt des Formulars in den HTML HEADER zusammenzustellen und sie zusammen an die URL zu übertragen, auf die das ACTION-Attribut verweist.

Basierend auf diesen Unterschieden und Funktionen verwenden wir also verschiedene get- oder post-Methoden für verschiedene Funktionen der API. Die **getPosition()**-Methode ist beispielsweise die Methode, die vom FrontEnd verwendet wird, um den Bereich zu ermitteln, den der Benutzer ausgewählt hat. Die Methode **insertPosition()** ist beispielsweise die Methode, mit der das Frontend den vom Benutzer ausgewählten Bereich ermittelt.

3.2.1.4 Middleware für Express: body-parser

Im Architekturdesign bezieht sich der Begriff 'Middleware' im Allgemeinen auf Software, die eine Verbindung zwischen der Systemsoftware (meist dem Betriebssystem) und der Anwendungssoftware herstellt, um die Kommunikation zwischen den verschiedenen Komponenten der Software zu erleichtern. Sie ist in modernen IT-Anwendungsrahmen wie Webservices und serviceorientierten Architekturen weit verwendet.

In Express hingegen besteht die Anwendung selbst aus einer Reihe von Middleware-Funktionsaufrufen. Die Middleware-Funktionen haben Zugriff auf das Anfrageobjekt (req), das Antwortobjekt (res) und die nächste Middleware-Funktion in der Anfrage- und Antwortschleife der Anwendung.

Die in diesem Projekt am häufigsten verwendete Middleware ist die Body-Parser-Middleware. Seine Aufgabe ist es, das Datenformat zu vereinheitlichen. Es gibt viele verschiedene Formate von Daten, die durch Benutzerpostanfragen(**post**) übermittelt werden. Daher benötigen wir dieses Plugin, um das Datenformat zu vereinheitlichen und es an den Server zur weiteren Datenverarbeitung zu übermitteln.

Wir führen den body-parser ein und parsen application/json mit folgendem Code.Der hier empfangene Standarddatentyp ist 'application/json'. Darüber hinaus parst express.urlencoded auch die Daten des Anfragekörpers im URL-kodierten Format.

bodyParser

```
var bodyParser = require('body-parser');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

3.2.2 Serverseitige Dantenbank

3.2.2.1 Datenbankverbindungen

Eine Datenbank(Database) ist ein Lagerhaus, das die Daten entsprechend der Datenstruktur organisiert, speichert und verwaltet. Nach dem Speichermodell kann hauptsächlich in Mesh-Datenbank, Baum-Datenbank, relationale Datenbank, objektorientierte Datenbank, etc. unterteilt werden. Die wichtigsten relationalen Datenbanken in kommerziellen Anwendungen sind Oracle, DB2, Sybase, MS SQL Server, Informax, MySQL und andere.

In diesem Projekt wird MySQL als Backend-Datenbank für die Website verwendet. Es ist zwar in Bezug auf die Sicherheitsseite nicht so gut wie andere große Datenbanken, aber da sie Daten in verschiedenen Tabellen speichert, erhöht sie die Flexibilität erheblich.

In unserer node.js-Backend-Umgebung lässt sich das native Modul nicht mit MySQL integrieren, so dass wir ein zusätzliches Modul für MySQL installieren müssen. Wenn wir auf die Datenbank zugreifen wollen, gibt es im Allgemeinen nur einen Weg, nämlich SQL-Befehle über das Netzwerk zu senden, die vom MySQL-Server ausgeführt werden und dann die Ergebnisse zurückgeben.

Der folgende Code zeigt ein Beispiel dafür, wie eine Verbindung zu einer Datenbank hergestellt werden kann. Zunächst müssen wir das Modul 'mysql' referenzieren und eine Verbindung herstellen. Erzeugt ein Verbindungsobjekt mit **mysql.createConnection**. Liefert Details über den MySQL-Server, mit dem eine Verbindung hergestellt werden soll. Akzeptiert ein Objekt als Parameter, übergibt Host, Benutzer usw. Als nächstes rufen wir die **connect-Methode** des connection-Objekts auf, um eine Verbindung zum MYSQL-Datenbankserver herzustellen.

Die Methode connection.query kann zur Abfrage und Ausführung jeder SQL-Befehl verwendet werden. Der erste akzeptierte Parameter ist ein Sql-String und der zweite Parameter ist eine Callback-Funktion, nachdem der Sql-Befehl ausgeführt worden ist.

Darüber hinaus müssen wir überlegen, wie wir mit Fehlern und Timeouts bei Datenbankverbindungen umgehen.Hier haben wir einen einfachen Prozess der Herstellung einer Datenbankverbindung im Falle eines Verbindungsfehlers oder einer Zeitüberschreitung durchgeführt.

mysql

```
const mysql = require("mysql");

var sqlConfig = {
  host: 'hostname',
  user: 'supermarkt',
  password: 'secret',
  port: 3306,
  database: 'supermarkt'
}

var connectdb = function() {
  let connection = mysql.createConnection(sqlConfig)
  connection.connect()
  connection.on('error', err=>{
    connection.log('Re-connection lost connection: ');
    connection = mysql.createConnection(sqlConfig)
  })
  return function() {
    return connection
  }
}

module.exports = connectdb()
```

3.2.2.2 CRUD: die Basis der Datenverwaltung

Das Wort CRUD ist ein Akronym, das sich aus einer Kombination der Anfangsbuchstaben einiger grundlegender Datenbankoperationen zusammensetzt.

Create (Datensatz anlegen) Read bzw. Retrieve (Datensatz lesen) Update (Datensatz aktualisieren) Delete bzw. Destroy (Datensatz löschen)

Der Verwaltungsprozess der verschiedenen Datenbankdaten basiert auf CRUD. Diese Operationen 3.2sind in der Regel unverzichtbare Werkzeuge für den Zugriff auf Datenbanken oder den Abruf von Daten aus ihnen. In diesem Projekt habe ich einige Datenbankoperationen verpackt, um den Zugriff auf die Daten des Projekts zu erleichtern. Dies bedeutet, dass ansonsten komplexe MySQL-Anweisungen nicht für jede Lese- und Schreiboperation einmal verwendet werden müssen. Stattdessen rief ich einfach die gekapselten Funktionen auf, um CRUD-Operationen in der Datenbank zu implementieren.

Nehmen wir den folgenden Operation des Einfügens neuer Daten als Beispiel. Die Parameter **'table'** bestimmt, welche Tabelle die neuen Daten erhält. datas sind die Daten,

CRUD-Operation	SQL	RESTful HTTP	XQuery
Create	INSERT	POST, PUT	insert
Read	SELECT	GET, HEAD	copy/modify/return
Update	UPDATE	PUT, PATCH	replace, rename
Delete	DELETE	DELETE	delete

Abbildung 3.2: CRUD-Operation
`https://dev.to/ijsathi/basic-crud-operations-hg0`

die als Objekt-Array übergeben werden, d.h. die hinzugefügte Daten. Ein 'callback' ist eine Rückruffunktion.

mysql insert

```
//insert data
let insertData = (table, datas, callback) => {
  var fields = '';
  var values = '';
  for (var k in datas) {
    fields += k + ',';
    values = values + "'" + datas[k] + "',"
  }
  fields = fields.slice(0, -1);
  values = values.slice(0, -1);
  console.log(fields, values);
  var sql = "INSERT INTO " + table + '(' + fields + ')
  VALUES(' + values + ')';
  console.log(sql);
  connection.query(sql, callback);
}
```

In den REST-API-Funktionen können wir das Formular, in das neue Daten eingefügt werden sollen, und die Anfrage über die Funktion ÏnsertData()ängeben. Je nachdem, ob die eingefügten Daten erfolgreich sind oder nicht, werden die entsprechenden Informationen wie Code und Nachricht eingestellt.

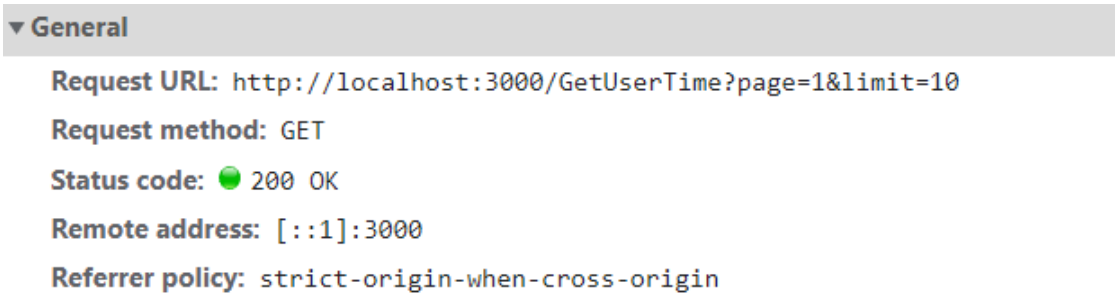

Abbildung 3.3: Browserprotokollen

insert classList

```
app.post('/admin/insertClassList', function (req, res) {
    req.body.time = new Date().getTime();
    db.insertData("classList", req.body, (e, r) => {
      if (e) {
        res.send({ code: 400, msg: 'Adding class failed' });
      };
      if (!r) {
        res.send({ code: 400, msg: 'Adding class failed' });
      } else {
        res.send({ code: 200, msg: 'Adding class successful' });
      };
    })
  });
```

Nachdem diese APIs vom Frontend aufgerufen wurden, ist es möglich, in den Browserprotokollen zu sehen3.3, wie diese Anfragen durch die APIs übertragen wurden und wie die Ergebnisse erzielt wurden. Mit der folgenden Schnittstelle als Beispiel haben wir auf die Datenbank zugegriffen, um die erforderlichen UserTime-Daten zu erhalten, und sie auf der Front-End-Seite auf dieselbe Weise dargestellt, wie im Code zuvor gezeigt. Damit ist eine vollständige Front-End-Datenabfrage 3.4 abgeschlossen, wobei das Back-End über die in der Schnittstelle gekapselten Funktionen auf die Datenbank zugreift und die gekapselten Funktionen die Operationen der Datenbankanweisungen aufrufen.

3.2.3 Frontend des Projekts

3.2.3.1 Frontend-Framework: LayUI

Im Webdesign bezieht sich ein Frontend-Framework im Allgemeinen auf einen Rahmen, der zur Vereinfachung des Webdesign-Prozesses verwendet wird. Es verwendet eine breite Palette von Frontend-Komponenten, die Funktionen wie die Bearbeitung von HTML-

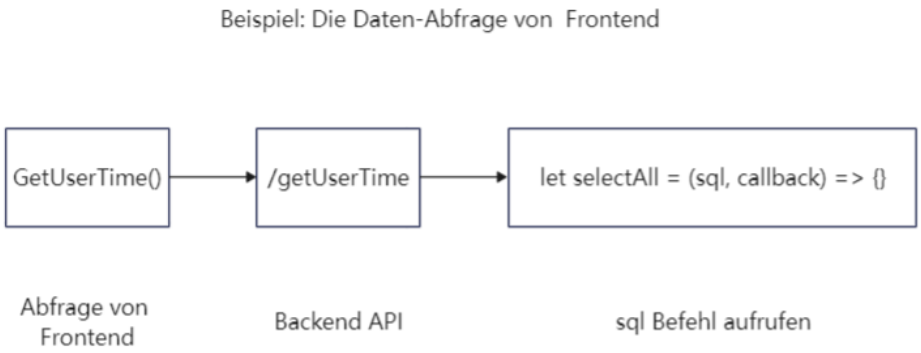

Abbildung 3.4: Front-End-Datenabfrage

Dokumenten und vorgefertigte Steuerelemente wie Buttons, Formulare usw. enthalten. Die Verwendung von Front-End-Frameworks kann uns helfen, schnell eine schöne Website zu erstellen.

Die LayUI, die in diesem Projekt verwendet wird, ist ein Frontend-UI-Framework, das mit einer eigenen Codespezifikation geschrieben wurde, die der nativen HTML/CSS/JS-Schreibweise und Organisation folgt.

Es sieht nicht nur gut aus, sondern hat auch den großen Vorteil, dass es vordefinierte Stilschnittstellen für Front- und Backend-Interaktionen, wie z. B. Formulare, bietet. Alles, was es tun kann, ist, die Schnittstelle am Frontend zu konfigurieren, und das Backend gibt die Daten entsprechend den Regeln zurück und die Seite wird angezeigt.

Das gesamte Paket wird im folgenden Verzeichnis 3.5 angezeigt, das nicht nur die erforderlichen css- und js-Dateien, sondern auch die erforderlichen Module und Schriftarten enthält.

Beim Referenzieren fügen wir einfach die folgenden zwei Dateien in das Verzeichnis und die HTML-Seite ein, um sie zu verwenden.

layui

```
./layui/css/layui.css
./layui/layui.js

<link rel="stylesheet" href="layui/css/layui.css">
<script src="layui/layui.js"></script>
```

Wenn wir bestimmte Elemente und Module verwenden und die entsprechenden Funktionen implementieren müssen, müssen wir der HTML-Seite JavaScript hinzufügen und die im Code verwendeten Module und Callback-Funktionen deklarieren. Wie das folgende Beispiel zeigt:

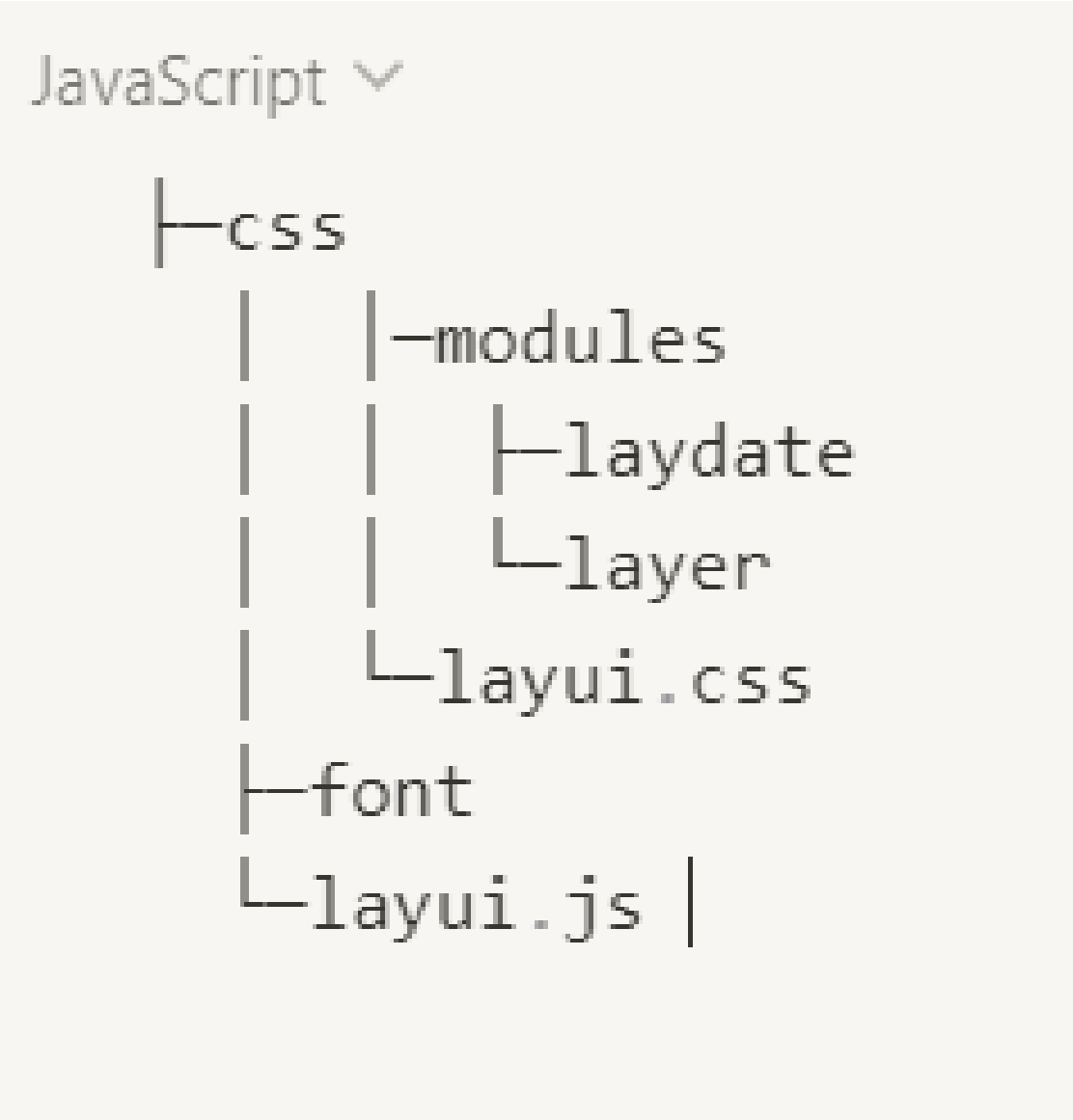

Abbildung 3.5: Verzeichnis

Hier haben wir **layer** und **form** verwendet, d. h. ein Pop-up-Fenster und ein Formular. Die Funktion **layui.use()** ermöglicht den Aufruf der verwendeten Module und das Hinzufügen von responsiven Funktionen und Callbacks. Hier haben wir ein Pop-up-Fenster mit der Nachricht 'Hello World'.

hello world

```
layui.use(['layer', 'form'], function(){
    var layer = layui.layer
    ,form = layui.form;

    layer.msg('Hello World');
});
```

Neben dem Aufruf von Modulen zur Implementierung und Bearbeitung funktionaler Aktionen können wir auch Container und Layouts verwenden, um die Oberfläche attraktiver zu gestalten.

hello world div

```
<div class="layui-container">
  <div class="layui-row">

  </div>
</div>
```

3.2.3.2 Front- und Backend-Datenkommunikation durch Ajax-Technologie

In Kapitel 2 haben wir kurz die Konzepte im Zusammenhang mit der Ajax-Technologie vorgestellt. Hier werden wir genau beschreiben, wie Ajax in diesem Projekt verwendet wird. Durch das Design und die Kapselung von Funktionen können wir es dem Frontend erleichtern, die vom Backend vorbereiteten Schnittstellen aufzurufen und mit dem Backend zu kommunizieren.

Im Abschnitt über CRUD-Operationen auf Datenbanken in Kapitel 3 haben wir erklärt, wie die Back-End-Schnittstelle die erforderlichen Daten aus der Datenbank extrahiert. Hier wird beschrieben, wie das Frontend Daten aus dem Backend entnimmt und sie schließlich dem Benutzer präsentiert.

Ermöglicht wird dieser komplette Funktionsumfang durch die Ajax-Technologie. Es funktioniert, indem ein Xmlhttp-Objekt instanziiert und dieses Objekt zur Kommunikation mit dem Backend verwendet wird. Die Rolle des XMLHTTP-Objekts besteht hier darin, dass es uns ermöglicht, Daten über Domänen hinweg ohne CORS-Fehler zu laden.

CORS ist ein Akronym für Cross-Origin Resource Sharingïm Englischen. Die Einstellungen des Servers im HTTP-Header ermöglichen es dem Browser, auf Daten aus

verschiedenen Quellen zuzugreifen.

Da der Prozess der Ajax-Kommunikation die Ausführung nachfolgender JS nicht beeinträchtigt, ermöglicht er außerdem eine asynchrone Kommunikation.

Im Folgenden wird ein einfacher Ajax-Funktionsaufruf gezeigt.

Angaben zu den Parametern.

1) url: die Adresse der Anfrage.

2) Typ: die Anfragemethode, die Default ist 'GET', die Methode 'POST' ist ebenfalls sehr verbreitet.

3) Daten: Einstellung der zu übertragenden Daten

4) dataType: legt das Format der Rückgabedaten fest, der Standard ist 'JSON'.

5) Erfolg: die Callback-Funktion nach dem Erfolg der Anfrage.

6) error: Callback-Funktion nach einem Fehler in der Anfrage.

ajax

```
$.ajax({
    url: 'request address',
    type: 'GET',      // request Type GET/POST
    data: {
        // Data
    },
    success: function(data) { // Callback Function
        // data from Backend APIs
    }
});
```

In diesem Projekt habe ich die Funktion verpackt, um es dem Frontend zu ermöglichen, die Ajax-Funktion jedes Mal aufzurufen, ohne die vollständige Aufrufgleichung schreiben zu müssen. Die detaillierten gekapselten Funktionen sind im folgenden Code dargestellt.

Die meisten Parameter sind im Wesentlichen dieselben wie in der einfachen Ajax-Funktion. In dieser Funktion geben wir url, type, data und callback als Parameter ein. Wenn die Ajax-Funktion erfolgreich ist, rufen wir auch die Layer-Komponente in layui als Pop-up-Fenster für den Hinweis auf. Außerdem setzen wir einen zusätzlichen Header-Parameter, der den Inhaltstyp äpplication/x-www-form-urlencodedënthält.

ajax

```
function postAjax(url,type,data,callback){
    $.ajax({
```

```
            url: url,
            type: type,
            headers: { 'Content-Type': '
            application/x-www-form-urlencoded' },
            data: data,
            success: function (res) {
                callback(res)
                // layer.msg(res.msg,
                { icon: res.code == 200 ? 1 : 2 })
            },
            error: function (err) {
                console.log(err)
                callback({code:500,data:[],msg:'Error'})
            }
        })
    }
```

Mit dieser umschlossenen Funktion sind die Front-End-Aufrufe viel einfacher. Wie im folgenden Code dargestellt: Wenn wir die Informationen, die der Benutzer eingegeben hat, aus der Datenbank löschen wollen, können wir einfach die postAjax()-Funktion am Frontend aufrufen und den API Namen, den Typ und die Daten als Parameter an diese Funktion übergeben. Diese Funktion ruft die delUserTime()-API von Backend unter Verwendung der eingewickelten Ajax-Funktion auf und löscht die Daten. Wenn die Daten gelöscht wird, erhält sie eine Callback-Funktion von der gekapselten Ajax-Funktion, um anzuzeigen, ob der Vorgang erfolgreich war oder nicht. Und das Front-End wird anhand des Elementaufrufs im LayUI feststellen, ob der Fehlercode in der Callback-Funktion korrekt ist. Auf der Grundlage des erhaltenen Codes wird eine Antwortmeldung angezeigt.

ajax

```
    postAjax('delUserTime', 'POST',
    { id: data.id }, function (res, err) {
        layer.msg(res.msg,
        { icon: res.code == 200 ? 1 : 2 });
  });
```

3.2.4 Bereitstellung einer Node.js Projekt

Dies liegt daran, dass die Informationen für dieses Projekt durch die Verteilung von Umfrage gesammelt wurden. Es war daher ein wichtiger Schritt, das Projekt online zu stellen und sicherzustellen, dass die Website reibungslos funktioniert. Dieser Abschnitt

befasst sich mit den verschiedenen Problemen, die bei der Bereitstellung des Projekts auftreten können, wie z. B. die Verbindung zu einem Remote-Server, die Konfiguration der Laufzeitumgebung und die Konfiguration der Datenbankumgebung.

3.2.4.1 Remote-Server verbinden mit ssh

Das als Server für dieses Projekt verwendete System ist **CentOS 8.5.2111 x86 64** CentOS, auch bekannt als Community Enterprise Operating System, ist eine der Distributionen von Linux. Um eine Umgebung auf unserem Server zu konfigurieren, die einen Betrieb von Node.js ermöglicht, müssen wir uns zunächst per SSH-Technologie beim Server anmelden.

SSH ist Secure Shell, ein sicheres Netzwerkprotokoll, das auf der Anwendungsschicht basiert und von der IETF, der Internet Engineering Task Force, entwickelt wurde.

Es handelt sich um ein Protokoll, das speziell für die Sicherheit von Fernanmeldesitzungen und anderen Netzwerkdiensten entwickelt wurde. Mit SSH können alle übertragenen Daten verschlüsselt werden, was den Übertragungsprozess sicherer und durch die Komprimierung der übertragenen Daten auch schneller macht. Es ist jetzt Standard auf Linux-Systemen.

SSH ist sicher, weil es asymmetrische Verschlüsselung (RSA) verwendet, um alle übertragenen Daten zu verschlüsseln. SSH selbst bietet zwei Authentifizierungsstufen.

Die erste ist die passwortbasierte Sicherheit, die es den Benutzern ermöglicht, sich remote beim Host anzumelden, solange sie ihr Konto und ihr Passwort kennen. Es besteht jedoch die Gefahr von Man-in-the-Middle-Angriffen.

Das zweite Verfahren ist die schlüsselbasierte Sicherheitsauthentifizierung. Die Benutzer müssen ein Schlüsselpaar für sich selbst erstellen und den öffentlichen Schlüssel auf dem Server hinterlegen, auf den sie zugreifen möchten. Wenn der Benutzer eine Verbindung zum SSH-Server herstellen muss, sendet die Client-Software eine Anforderung an den Server, den Schlüssel des Benutzers für die sichere Authentifizierung zu verwenden. Wenn der Server diese Anfrage erhält, sucht er nach dem öffentlichen Schlüssel im Dateiverzeichnis des Benutzers und vergleicht den vom Benutzer gesendeten öffentlichen Schlüssel mit dem gespeicherten öffentlichen Schlüssel. Wenn die beiden übereinstimmen, verschlüsselt der Server das 'challenge' mit dem öffentlichen Schlüssel und sendet sie an die Client-Software. Sobald die Client-Software das challengeërhalten hat, kann sie diese lokal mit dem privaten Schlüssel des Benutzers entschlüsseln und an den Server senden, um die Anmeldung abzuschließen.

Der folgende Code und die Befehlszeile zeigen einfach, wie eine SSH-Verbindung hergestellt wird. Zunächst muss der Server den ssh-Server starten. Die Passwortanmeldung ist sehr einfach und erfordert nur einen Befehl in der Form: **ssh client username@server ip address**

Der Server fragt dann nach einem Passwort, das korrekt eingegeben wird, und Sie

können sich erfolgreich anmelden. Die im Terminal eingegebenen Befehle werden nach einer erfolgreichen Anmeldung auf dem Server ausgeführt.

ssh

```
sudo /etc/init.d/ssh start
sudo /etc/init.d/ssh stop  #server stop ssh
sudo /etc/init.d/ssh restart  #server restart ssh

ssh client username@server ip address
```

Zweitens haben wir folgende Maßnahmen ergriffen, um die Sicherheit des Servers zu erhöhen. Eine ist, den Standard-Login-Port zu ändern.

ssh config

```
#edit the config file
sudo vi /etc/ssh/sshd_config
#change Port
Port xx

AllowsUsers username
ssh -p xx username@47.xx.xx.xx
```

Durch Hinzufügen eines Benutzernamens zur IP-Adresse und zum Port kann der Port weniger leicht erraten werden, wodurch die Sicherheit erhöht wird.

3.2.4.2 Erstellen einer Nodejs-Umgebung für den Server

Wir haben bereits kurz beschrieben, wie Node.js installiert wird, und die Schritte zur Installation auf dem Server sind mehr oder weniger dieselben. Wir installieren jedoch wie erforderlich nvm, das Node.js-Versionsverwaltungswerkzeug, und geben Node.js Version 16.16.0 an.

Wir brauchen auch die Hilfe von pm2, um unser Prozessmanagement zu vereinfachen. PM2 ist ein Prozessmanager für Node-Anwendungen mit Lastausgleich. Wir schon wissen, dass Node.js eine Einzelprozessausführung ist, und wenn eine Anwendung mit einem Fehler stirbt, muss sie in der Lage sein, dies zu automatisieren, und hier kommt PM2 ins Spiel. Natürlich gibt es viele Prozessmanagement-Tools, wie z. B. forever usw., aber wir haben hier das am häufigsten verwendete PM2 ausgewählt.

Er hat mehrere Vorteile, die für dieses Projekt von großer Bedeutung sein werden. Erstens unterstützt es Hot Restart und wartet automatisch auf Restarts. Dies wird das Projekt nicht sehr stören, wenn ein Thread-Fehler in der Zeile auftritt. Das zweite Merkmal ist die Unterstützung der Protokollierung, die Aufzeichnung von Systemprotokollen und Protokollen verwalteter Prozesse sowie die Protokollierung der Konsolenausgabe in

eine Protokolldatei. Dies hilft mir, die Leistung von Datenbanklesen und -aufrufen auf dem Server in gewissem Maße zu überwachen.

Der Installationsbefehl wird einfach als **npm install -g pm2** für eine globale Installation eingegeben. Das folgende Verzeichnis wird nach Abschluss der Installation automatisch auf dem Server erstellt.

pm2

```
$HOME/.pm2 will contain all PM2 related files
$HOME/.pm2/logs will contain all applications logs
$HOME/.pm2/pids will contain all applications pids
$HOME/.pm2/pm2.log PM2 logs
$HOME/.pm2/pm2.pid PM2 pid
$HOME/.pm2/rpc.sock Socket file for remote commands
$HOME/.pm2/pub.sock Socket file for publishable events
$HOME/.pm2/conf.js PM2 Configuration
```

Einer der am häufigsten verwendeten Parameter ist der Parameter –watch, der pm2 start ./bin/www –watch Befehl . Er wird verwendet, um auf Änderungen im Anwendungsverzeichnis zu warten und startet automatisch neu, wenn eine Änderung eintritt, wie z. B. eine Datei- oder Codeaktualisierung.

3.2.4.3 Nginx Konfiguration für Reverse-Proxys

Ein Reverse-Proxy ist eine Art Proxy-Server in einem Computernetz. Der Server bezieht Ressourcen von einer oder mehreren Gruppen von Back-End-Servern (z. B. Webservern), mit denen er auf der Grundlage einer Anfrage des Kunden verbunden ist, und gibt diese Ressourcen dann an den Kunden zurück, der nur die IP-Adresse des Reverse-Proxys kennt und nichts von der Existenz der Servergruppe hinter dem Proxy-Server weiß. [1]

Wenn Wir Node.js verwenden, um einen Dienst auf der Serverseite auszuführen, können wir Nginx als Reverse Proxy verwenden, da der Port nicht von mehreren Diensten gleichzeitig belegt werden kann und der Server mehrere Websites haben kann.

Nginx ist ein leichtgewichtiger Webserver und Reverse-Proxy-Server. Es ist wegen seines geringen Speicherbedarfs und seiner hohen Gleichzeitigkeitsfähigkeit weit verwendet

Wir müssen Nginx wie zuvor mit dem npm-Befehl installieren: **sudo apt-get install nginx.** Die folgende Konfigurationsdatei zeigt kurz die Nginx-Konfigurationsregeln, über die Regeln und Reverse-Proxies für den Zugriff auf die im Server geöffneten Projektportdienste über die IP-Adresse des Servers aufgestellt werden können.

proxy

```
upstream co {
    server 127.0.0.1:8081;
}
```

```
server {
    listen 80;
    server_name 47.xx.xx.xx;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-Nginx-Proxy true;

        proxy_pass http://webseite-beispiel;
        proxy_redirect off;
    }
}
```

Sobald der Benutzer http://47.xx.xx.xx aufgerufen hat, wird er vom Server auf http://127.0.0.1:8081 umgeleitet und das Projekt wird abgerufen.

3.2.4.4 Konfigurieren der Online-Datenbank

Schließlich benötigen wir auch eine Online-Datenbank, um die vom Benutzer eingegebenen Informationen zu speichern. Wie man das Node.js-Backend mit der Datenbank verbindet, wurde bereits zuvor beschrieben.

Nachdem die Datenbank installiert wurde, können wir den Befehl **service mysqld start** verwenden, um die Online-Datenbank für die Verbindung zu starten.

4 Funktionen

In den vorangegangenen Kapiteln haben wir uns eingehend mit den theoretischen Kenntnissen über das Projekt sowie mit dem architektonischen Entwurf und der Umsetzung des Projekts befasst. In diesem Kapitel wird auf das Projekt selbst eingegangen, und es wird der Prozess der Entwicklung und Implementierung einiger der Funktionen vorgestellt.

Darüber hinaus werden in diesem Kapitel auch die bei der Durchführung des Projekts aufgetretenen Schwierigkeiten und die entsprechenden Lösungen behandelt.

4.1 Einführung in die wichtigen Funktionen des Projekts

4.1.1 Funktion zur Benutzeranmeldung

Es gibt zwei Rollen im System, die erste ist der Administrator, der Zugang hat, um den Umfragen einzurichten, ihn zu ändern und das Feedback einzusehen. Der zweite ist der normale Web-Besucher, der keine anderen Rechte hat, als den Umfrage auszufüllen. Daher benötigen wir ein Login für den Administrator, um die Daten zu verfolgen. Eine reguläre Anmeldung eines Web-Benutzers besteht aus folgenden Schritten:

Zuerst werden die vom Benutzer in das Formular eingegebenen Informationen eingeholt. Es folgt ein Frontend-Aufruf an die Backend-Schnittstelle, die einen Datenbankbefehl ausführt, um festzustellen, ob der vom Benutzer eingegebene Benutzername und das Passwort in den in der Datenbank gespeicherten Benutzerinformationen vorhanden sind. Wenn sie vorhanden sind, werden sie validiert und die Identität des Benutzers wird temporär gespeichert, damit er spätere Operationen im Verwaltungssystem durchführen kann, ohne seine Identität erneut validieren zu müssen.

Der folgende Code zeigt, wie eine grundlegende Anmeldefunktion im System implementiert ist. Die Benutzereingabe ruft zunächst die Formularfunktion des **Layui**-Frameworks auf, dann wird die asynchrone Anmeldung über die bereits beschriebene vorbereitete **postAjax()** Funktion realisiert und die Backend-Anmeldeschnittstelle aufgerufen. Sobald die Authentifizierung erfolgreich war, wechselt das System zur Management-Seite und speichert die Anmeldedaten des Benutzers vorübergehend, indem es eine Sitzung einrichtet.

Jeder Benutzer erhält beim ersten Zugriff auf den Server automatisch eine Session-ID. Wenn er über einen bestimmten Zeitraum nicht auf den Server zugreift, wird die Session automatisch ungültig, und beim nächsten Zugriff auf den Server geht der Server davon aus, dass es sich um einen neuen Benutzer handelt, und weist ihm eine neue Session-ID zu, selbst wenn er die ihm zugewiesene Session-ID hat. Es wird eine neue Session-ID zugewiesen. Da das HTTP-Protokoll selbst ein zustandsloses Protokoll ist,

können Anwendungen nicht zwischen zwei Anmeldeanfragen unterscheiden, die von demselben Browser eingehen. Mit Hilfe von Session kann der Server den Status des Benutzers verfolgen.

Wenn sich ein Benutzer von der Verwaltungsseite abmelden möchte, wird das Gegenteil erreicht, indem einfach die Informationen aus dem Session entfernt werden.

login

```
//login
form.on('submit(login)', function (data) {
  postAjax('login', 'POST', data.field, (res) => {
        if (res.code == 200) {
            window.sessionStorage
            .setItem('user', res.data.username);
            layer.msg(res.msg,
            { icon: res.code == 200 ? 1 : 2, time: 1000 },
            function () {
            window.location.href = "./admin";
          })
        } else {
          layer.msg(res.msg,
          { icon: res.code == 200 ? 1 : 2, time: 1000 })
        }
  })
  return false;
});

//logout
function (index, layero) {
      window.sessionStorage.removeItem('user');
      window.location.href = "/login";
}
```

4.1.2 Anpassen und Speichern des Supermarkt-Layouts

Die zweite einzuführende Funktion ist in diesem System ebenfalls sehr wichtig, da die gesamte nachfolgende Datenverarbeitung mit ihr zusammenhängt.

Die Aufgabe des Verwalters bei diesem Projekt bestand darin, die Anzahl der Regale und Gebieten in diesem Supermarkt zu ermitteln und die Anordnung der Regale entsprechend der Situation zu gestalten. Die folgende Abbildung 4.1 zeigt eine Schnittstelle für die Gestaltung eines Supermarktes. Zunächst wird beim Laden dieser Seite die Liste der

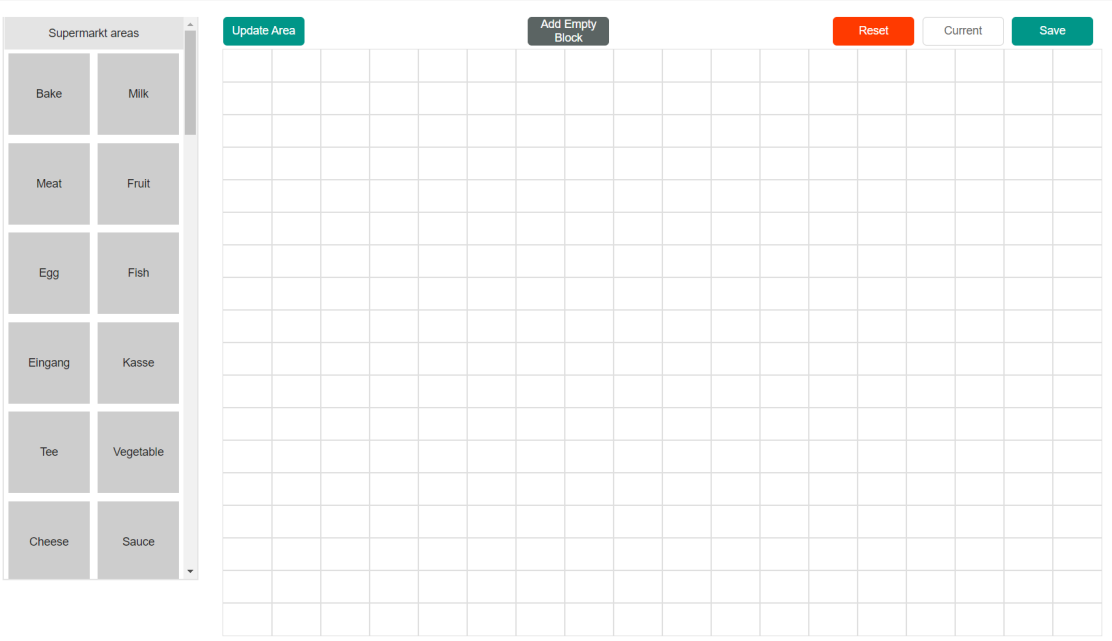

Abbildung 4.1: Map of the Supermarkt

Supermarktgebiete auf der linken Seite geladen, und Administratoren können Gebiete hinzufügen 4.2, um das Angebot an Supermarktgebieten zu erweitern. Der neu hinzugefügte Supermarktbereich wird in einem Datenbankformular heißt classList gespeichert, das später aufgerufen werden kann.

Mit Hilfe der bereits beschriebenen **getClassList** Methoden können wir die Liste der Bereiche aus der Datenbank abrufen und, wenn das Layout gleichzeitig aktualisiert wird, das Button 'Update Area' verwenden, um die neueste Liste der Bereiche zu erhalten. Auf diese Weise erhalten wir ein Array von Namen von Supermarktbereichen.

Wir möchten jedoch eine Liste dieser Supermarktbereiche in das Auswahlfeld auf der linken Seite der Seite einfügen 4.3, die ausgewählt und angeklickt werden kann. Hier haben wir in unserem HTML-Code verwendet, um jeden Supermarktbereich separat zu speichern, und wir haben ihren Stil als kleinen optionalen Kasten definiert. Wie in der Abbildung zu sehen ist, wird bei Auswahl dieses Feldes in der oberen linken Ecke der Seite ein kleines, verschiebbares schwarzes Feld angezeigt, das einen Supermarktbereich darstellt. Um diese Funktion des Hinzufügens kleiner Boxen zu einer Seite zu implementieren, verwenden wir das DOM-Baummodell in JavaScript. Wie wir wissen, ist das wichtigste Element eines HTML-Dokuments der Tag. Nach dem DOM (Document Object Model) ist jedes Tag ein Objekt, und die untergeordneten Tags, die im übergeordneten Tag verschachtelt sind, können als Zweige eines Baums betrachtet werden. Diese Objekte sind also über JavaScript zugänglich, und wir können sie verwenden, um die Seite zu ändern. Wenn wir dem Supermarktbereich auf der linken Seite einen neuen Bereich hinzufügen wollen, der ausgewählt werden kann, fügen wir dem ¡ul¿-Tag

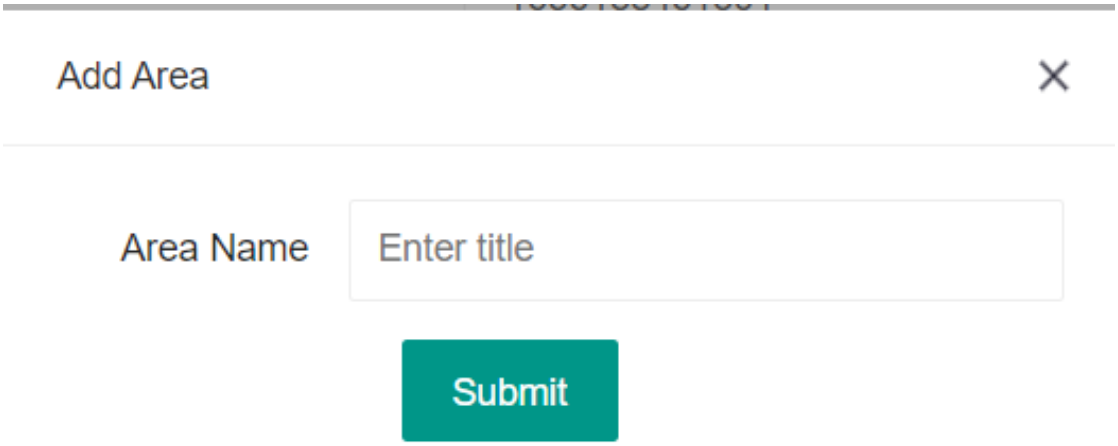

Abbildung 4.2: Add area

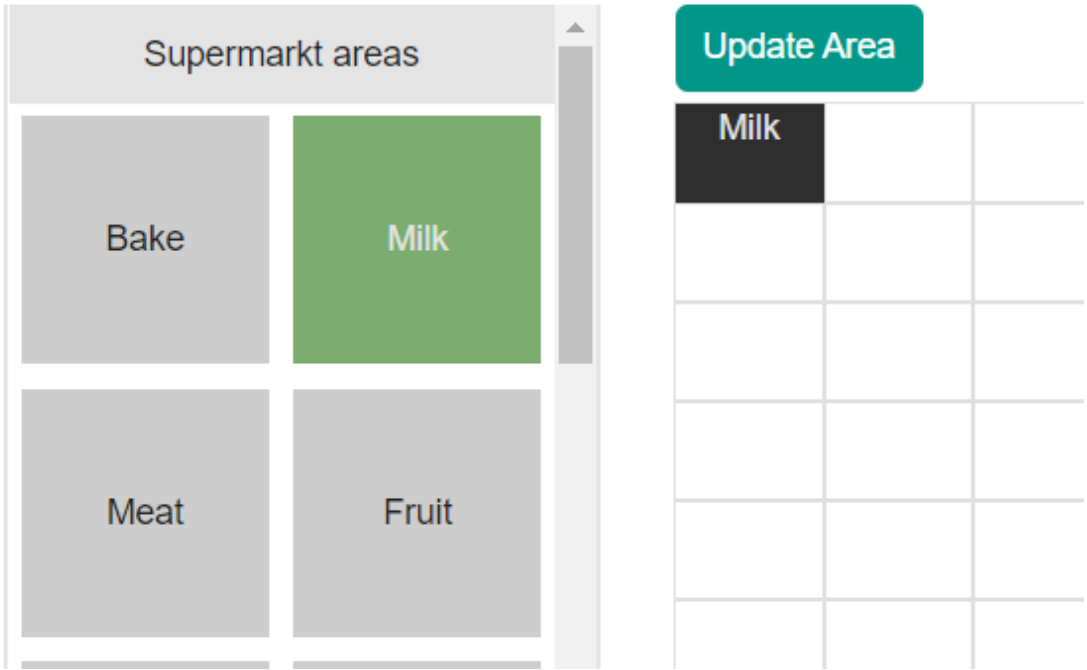

Abbildung 4.3: Drag the new added area

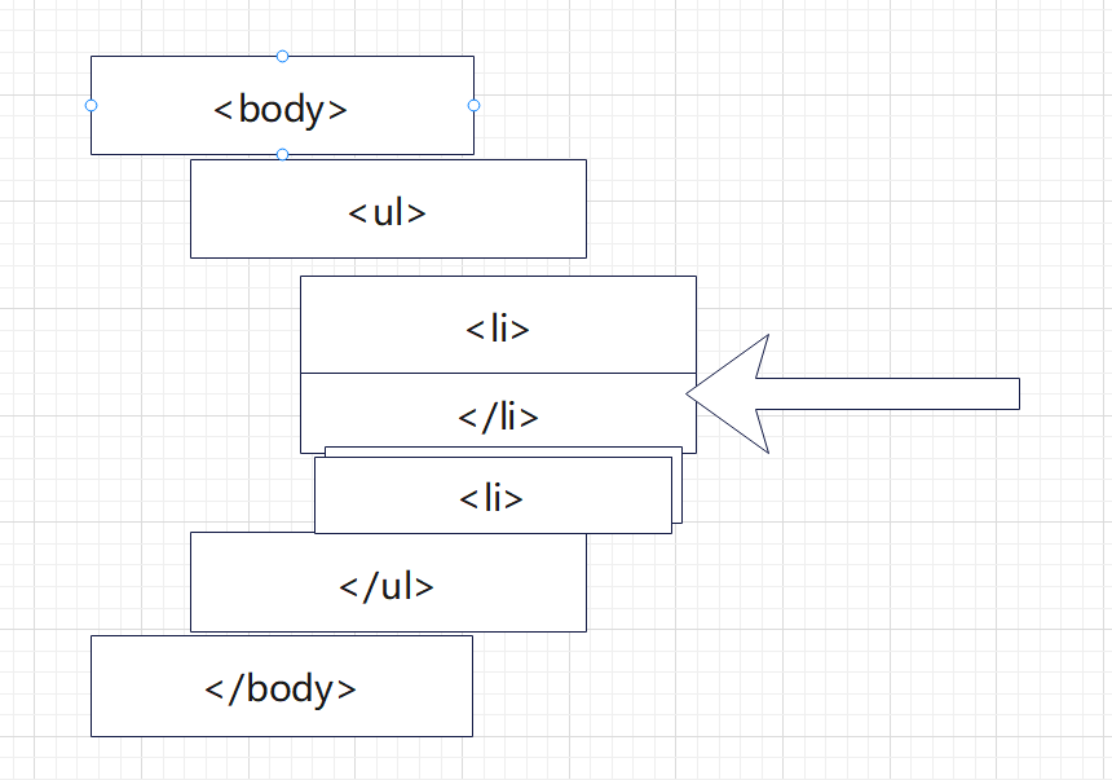

Abbildung 4.4: Struktur

einen neuen untergeordneten Knoten hinzu, der das neue ¡li¿-Tag ist. Hier ist eine Code-Demonstration des Prozesses.

aside

```
$('.aside ul').append(`
<li data-name="${item.className}" data-id="${index}"
class="asideitem aside${index}">${item.className}</li>
`)
```

Für jedes Element im Array des Supermarktbereichs wird es auf diese Weise der Liste hinzugefügt und somit in der List auf der linken Seite angezeigt. 4.4 Neben dem Menü auf der linken Seite werden auch die Kisten, aus denen sich der Seitenplan zusammensetzt, nacheinander auf diese Weise und durch die Methode **append()** unter der Eltern Knoten "box" Tag zusammengesetzt. Die verschiebbare schwarze Kisten, die die Supermarktbereiche darstellen, werden auf diese Weise ebenfalls der Seite hinzugefügt.

Als Nächstes wollen wir diese Kisten auf eine bestimmte Weise verschieben, um das Layout des Supermarktes zu vervollständigen. Wir wollen nämlich unser eigenes Supermarkt-Koordinatensystem aufbauen, indem wir ein Layout konstruieren, das die Grundlage für künftige Berechnungen und Datenverarbeitung bildet. 4.5

Der folgende Code zeigt ein einfaches Verfahren zur Erstellung von Koordinaten. Jedes

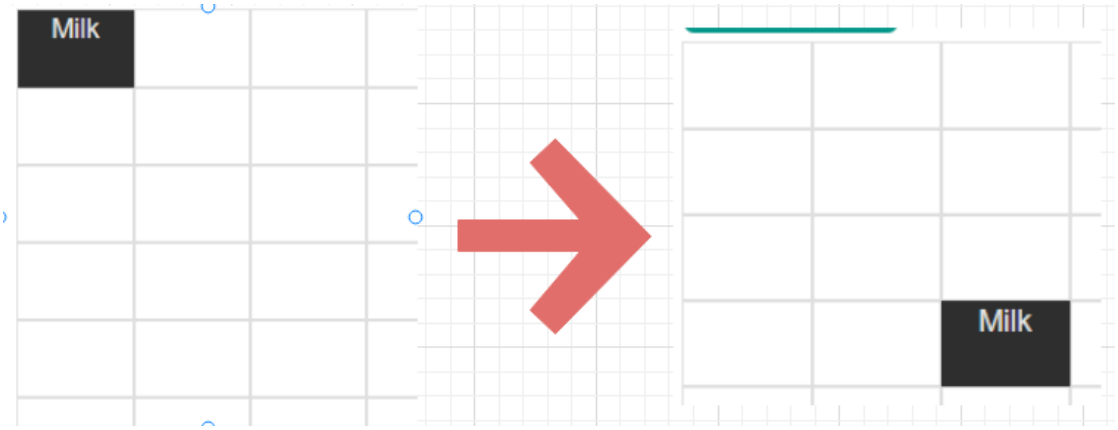

Abbildung 4.5: Move the box

Kästchen kann als ein Punkt betrachtet werden. Die gesamte Karte des Supermarkts hat die Größe 18*18, jedes Feld ist 60px lang und 40px breit. Jedes Feld wird an der linken oberen Ecke der Karte initialisiert, d.h. ihr Ausgangspunkt ist (0, 0). Wir können die Zeile und Spalte der Karte bestimmen, indem wir die Anzahl der Pixel ermitteln, über die die Box gezogen wurde, und bestimmen, wo sie enden wird.

aside

```
    this.set = {
        boxNode: "#box",
        blockW: 60,
        blockH: 40,
        block: "(18,18)",
        startPos: "(0,0)"
};

$(this.set.boxNode).css({
        width: this.col * this.set.blockW + "px",
        height: this.row * this.set.blockH + "px"
});
```

Dazu gehört ein Kalibrierungsprozess. Das heißt, es wird davon ausgegangen, dass die Maus automatisch dorthin fährt, wohin sie gehen soll, nachdem sie irgendwo während des Ziehvorgangs stehen geblieben ist und losgelassen wurde. Es wird automatisch ermittelt, welches Raster auf der Karte für seine Länge und Breite am besten geeignet ist und wo es sich befindet.

Genau wie die auf diesem Bild gezeigte.4.6 Indem es feststellt, in welchen Bereich seine Länge und Breite hauptsächlich fallen, bestimmt es, in welche Box es automatisch kommt. In einem Fall wie diesem, in dem sich der größte Teil des Bereichs dieses Kastens

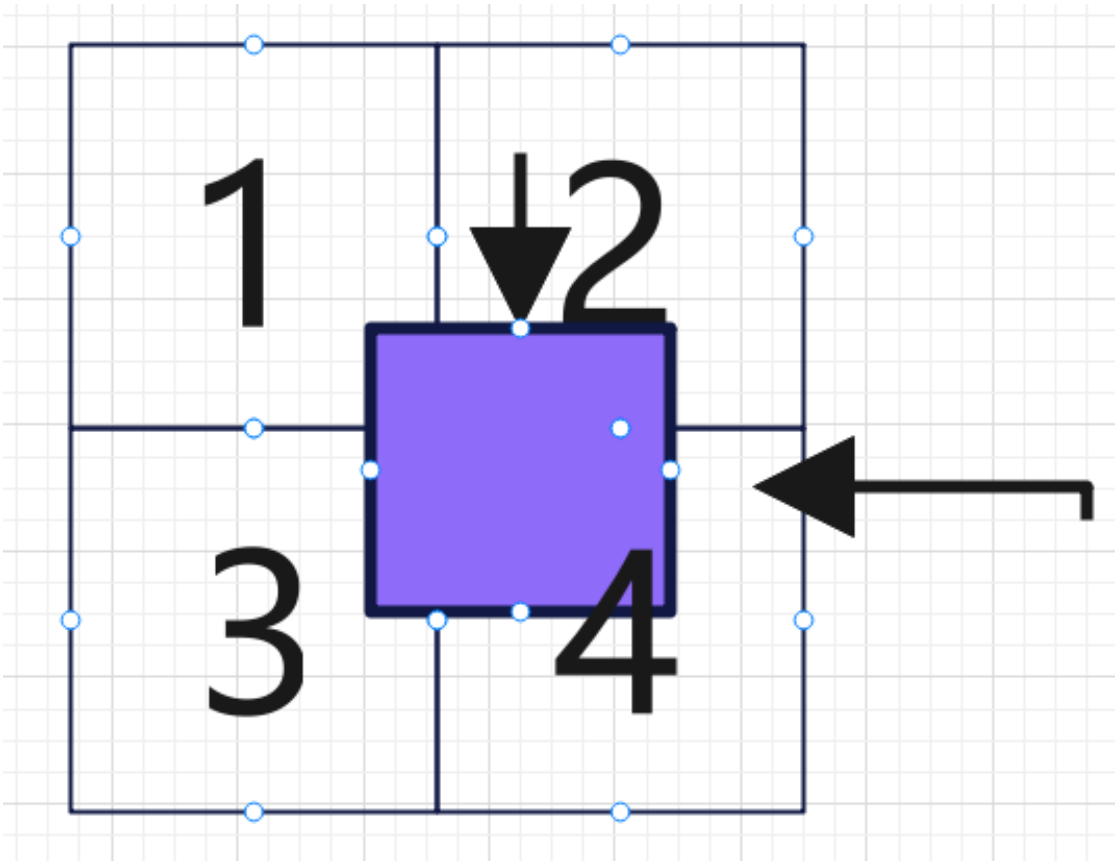

Abbildung 4.6: Box adjust

mit dem vierten Bereich überschneidet, wird der Kasten automatisch in den vierten Bereich verschoben, wenn wir die Maus loslassen. Dieser Vorgang wird durch eine sanfte Animation noch gestaltet, um den natürlicher wirken zu lassen.

Nach dem Ziehen des Kastens, um seinen Abwurfpunkt zu bestimmen, müssen wir die neue Position des Kastens in unserem Koordinatensystem in Form von Koordinaten speichern, und der entsprechende Bereich des Kastens, zusammen mit dem Namen und der ID des Kastens, wird in einem Array gespeichert. Wie die Abbildung unten zeigt 4.7, sind die Variablen pageX und pageY seine Koordinaten. Der ausgewählte Bereich wird auch in der linken Seitenleiste markiert und kann durch einen Rechtsklick auf den Bereich in der Seitenleiste oder auf ein bestehendes Feld in der Karte gelöscht werden. Sobald der Administrator die Karte entworfen hat, ruft das Front-End automatisch die Karteninformationen ab und lädt automatisch die neueste Karte in die Datenbank. Die Daten werden wie im Diagramm dargestellt transformiert, indem die X- und Y-Koordinaten jedes Kästchens genommen und das Kästchen an der entsprechenden Position hinzugefügt wird.

Sobald der Benutzer auf der Startseite die Umfrage zu den Basisinformationen ausgefüllt hat, kann er mit der rechten Maustaste auf das kleine Kästchen klicken, das jeden Supermarktbereich darstellt, und die Zeit hinzufügen, die er in diesem Bereich verbracht

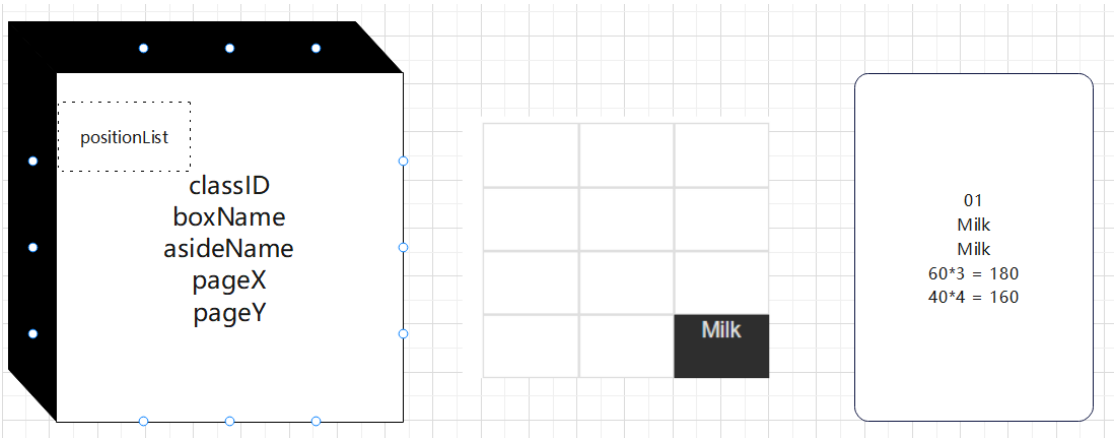

Abbildung 4.7: Box adjust

hat. Wenn sie normalerweise immer die gleichen Produkte in einem Bereich kaufen und selten vergleichen oder auswählen, wählen Sie 'short'. Wenn sie jedes Mal, wenn sie sich auf eine Information konzentrieren müssen, um eine Auswahl zu treffen, mehr als zwei Minuten bleiben, wählen Sie 'middle'. Wenn sie sich oft und lange in einem bestimmten Bereich aufhalten, wählen Sie 'long'. Die spezifischen Auswahlregeln sind in der Hinweisen auf dieser Seite beschrieben. Bevor der Benutzer eine Zone auswählen kann, muss er auch einige grundlegende Informationen eingeben, darunter sein Alter und die Tageszeit, zu der er am häufigsten einkauft, z. B. Montagnachmittag. Um zu einem allgemeinen Modell für dieses Projekt zu gelangen, wurde nur die am stärksten frequentierte Zeit der Woche untersucht; mehrere Zeiträume wurden zunächst nicht berücksichtigt.

Die Ergebnisse werden in einem Objekt-Array gespeichert, und jeder Supermarktbereich ist an eine Verweildauer gebunden, um die weitere Datenverarbeitung zu erleichtern. 4.8

4.1.3 Erfassung und Darstellung der Umfrageergebnisse

Nachdem wir das System aufgebaut und freigegeben haben, können wir die vom Benutzer bereitgestellten Daten zusammenfassen, um ein Bild von der Verteilung der Personen in den einzelnen Abteilungen des Supermarktes zu einem bestimmten Zeitpunkt zu erhalten. Die Umfrage war in vier Tageszeiträume unterteilt: morgens (von 7 bis 11 Uhr), mittags (11 bis 15 Uhr), nachmittags (15 bis 19 Uhr) und abends (19 bis 22 Uhr). Bis zum Zeitpunkt der Abfassung dieser Arbeit gingen insgesamt 157 Antworten auf die Umfrage ein, die einen gewissen Referenzwert darstellen können.

Unser Ziel ist es, anhand der verfügbaren Daten die Verteilung der Menschen in jedem Supermarktgebiet zu einem bestimmten Zeitpunkt zu berechnen. Wir wollen auch herausfinden, ob die Verteilung der Menschen in den einzelnen Bereichen des Supermarktes mit der Gestaltung des Supermarktes zusammenhängt. Um dies zu erreichen, haben wir in Umfrage zwei leicht unterschiedliche Supermarkt-Layouts eingerichtet.

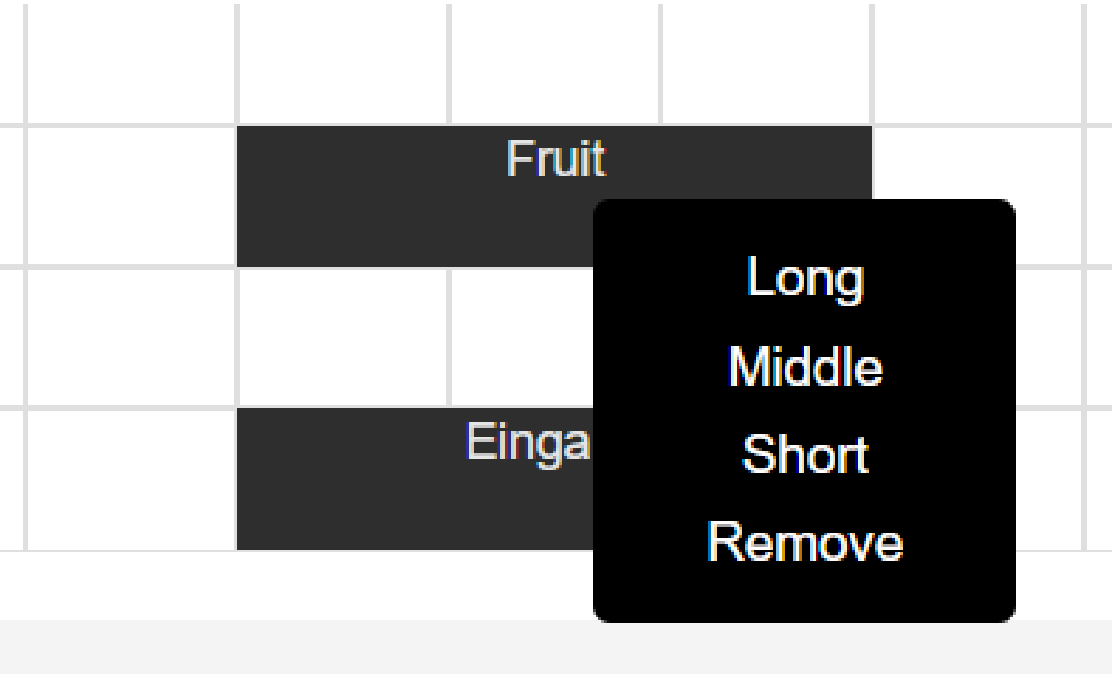

Abbildung 4.8: Gasts choose the area

Anhand bestimmter Berechnungen erhalten wir dann die Verteilung der Anzahl der Personen in jedem der beiden Supermarkt-Layouts zu einem bestimmten Zeitpunkt. In diesem Projekt gehen wir davon aus, dass die Kunden zu jedem Zeitpunkt den Supermarkt zur gleichen Zeit betreten und ihren Einkauf entsprechend ihren Einkaufsgewohnheiten getrennt durchführen. In einer realen Situation kann dieser Zeitraum detaillierter aufgeteilt werden, und die Menschen betreten den Supermarkt nicht zur gleichen Zeit, um mit dem Einkauf zu beginnen. Diese Bedingung gilt nur für die Analyse dieses Modells.

Das Endergebnis sieht so aus wie in der Abbildung gezeigt.4.9 Der Administrator kann den Zeitpunkt eingeben, den er einsehen möchte, und die Datenbank liest dann die Informationen über die Kunden aus, die zu diesem Zeitpunkt eingekauft haben, und die Reihenfolge, in der sie eingekauft haben. Durch die Erstellung eines Zeitstrahls für jeden Kunden kann der Zeitstrahl für jeden Bereich des Supermarktes ermittelt werden, um festzustellen, ob dieser Bereich zu einem bestimmten Zeitpunkt stärker frequentiert ist. Bereiche mit ganz viele Leute sind mit roten Blöcken gekennzeichnet, orangefarbene Blöcke für mittleres Menge, grüne Blöcke für geringes Menge und unmarkierte Bereiche, in denen sich fast keine aufhalten.

Da die realen Daten komplex sind und viele Bereiche und lange Zeiträume umfassen, werde ich hier ein einfacheres Modell verwenden, um zu zeigen, wie ich diese Verteilungen und Farbmarkierungen berechnet habe.

Wie bereits erwähnt, gehen wir zunächst davon aus, dass zu einem bestimmten Zeitpunkt alle Kunden gleichzeitig den Supermarkt betreten und mit dem Einkauf beginnen.

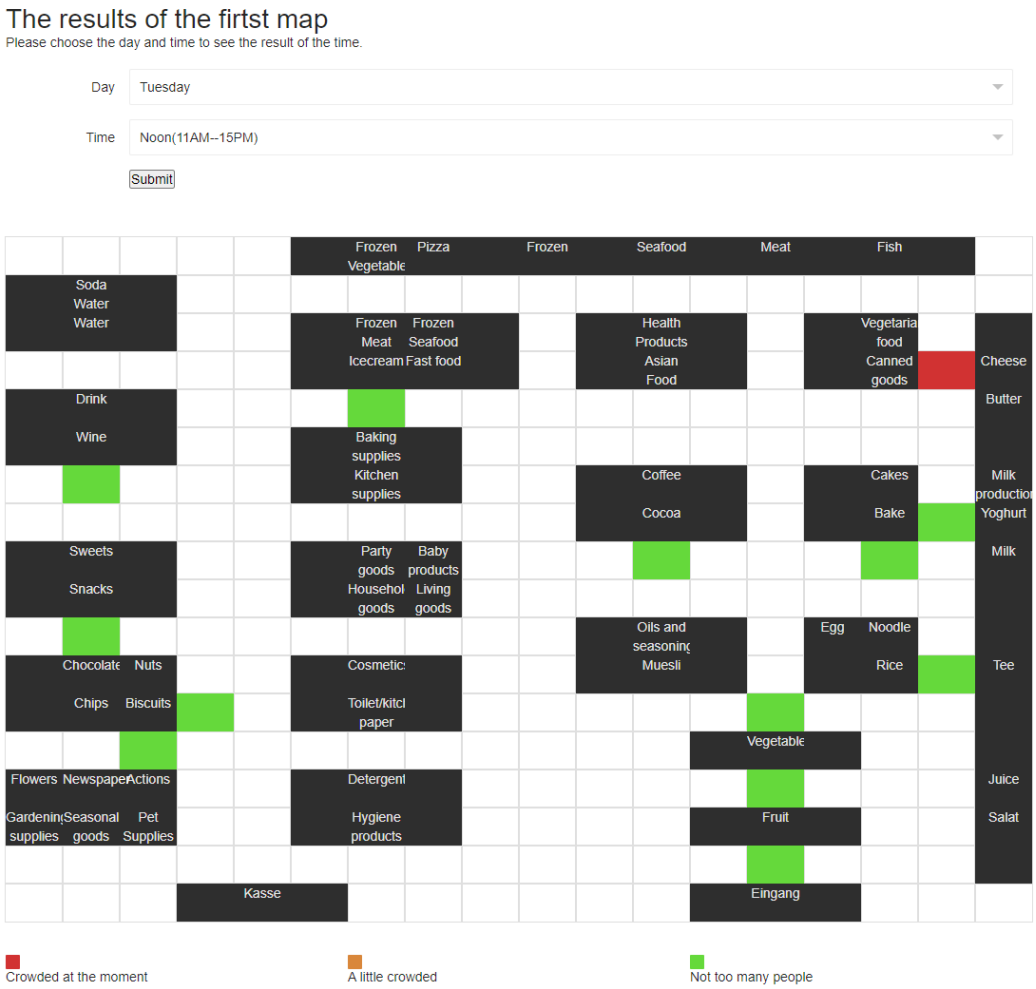

Abbildung 4.9: Evaluation of Map 1

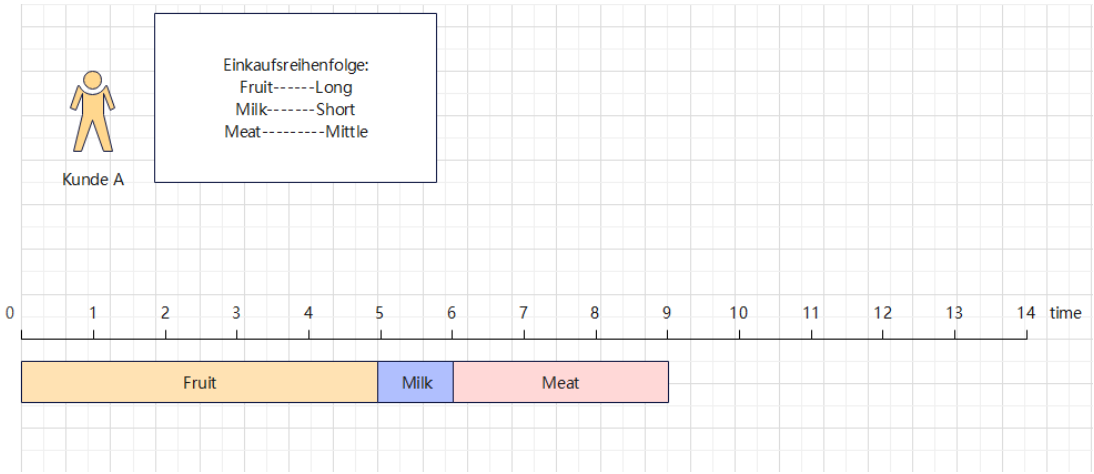

Abbildung 4.10: Gast Timeline

In unserer Umfrage geht es um ihre Einkaufsgewohnheiten und darum, wie lange sie sich in welches Bereich aufhalten werden. Alles, was wir tun müssen, ist, die eigene Zeitstrahl des Kunden für einen von ihnen zu erstellen. In dem Diagramm haben wir einen Kunden A, der eine lange Zeit in der Obstabteilung verbringt, dann eine kurze Zeit in der Milchabteilung und schließlich eine mittlere Zeitspanne in der Fleischabteilung. Wir gehen davon aus, dass jeder lange Aufenthalt 5 Zeiteinheiten, jeder mittlere Aufenthalt 3 Zeiteinheiten und jeder kurze Aufenthalt eine Zeiteinheit nimmt. 4.10

Aber nicht nur beim Einkaufen brauchen sie Zeit, auch auf dem Weg von einem Bereich zum anderen verbringen die Kunden eine gewisse Zeit, je nach den Unterschieden in den verschiedenen Layouts unserer Supermärkte.

Da unsere Gebiete aus Gittern bestehen, müssen wir nur feststellen, ob die beiden Gitter horizontal und vertikal und an jeder Stelle des Diagramms, die sie trennt, miteinander verbunden sind, um unsere Reise vom aktuellen Gebiet zum nächsten Gebiet zu berechnen. Wir gehen davon aus, dass die linke Seite die Standard-Reiserichtung für den Kunden ist. Zuerst stellen wir fest, ob der kürzeste Weg von Punkt A nach Punkt B direkt begehbar ist. Wenn nicht, ändern wir die Richtung und stellen fest, ob es Hindernisse auf dem kürzesten Weg von rechts gibt. Gibt es mehr Hindernisse auf der rechten Seite oder kommt man an den Rand, wählt man den einzigen Weg, aber die Anzahl der zu durchquerenden Quadrate wird größer, d.h. der Weg ist länger. 4.11

Die Entfernung wird als Variable zu unserem Zeitstahl hinzugefügt, nachdem wir die ungefähre Entfernung auf der Karte für jeden Kunden berechnet haben. Wir nehmen an, dass die Zeit vom Obstbereich zum Milchbereich zwei Zeiteinheiten und die Zeit vom Milchbereich zum Fleischbereich drei Zeiteinheiten beträgt. Der gesamte Zeitplan für Kunde A ist unten dargestellt. 4.12

Zur gleichen Zeit kaufen auch Kunde B und Kunde C ein. Kunde B bleibt kurz in der Obstabteilung und geht dann direkt zur Fleischabteilung. Kunde C hingegen geht zuerst

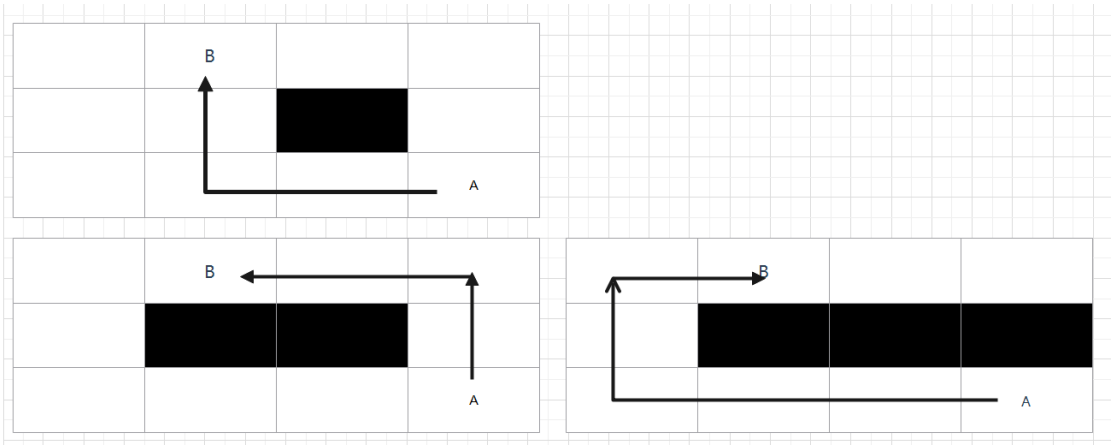

Abbildung 4.11: Calculate the distance between two areas

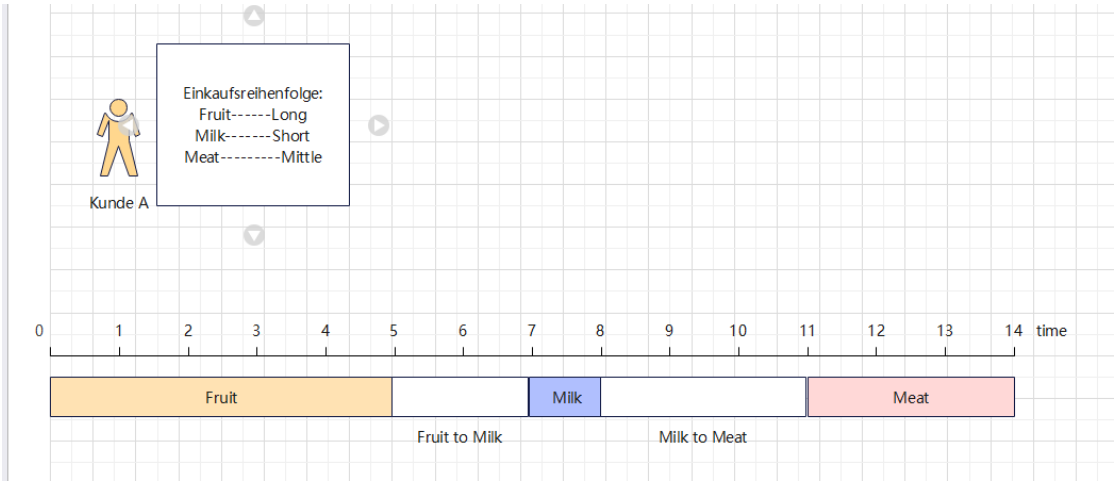

Abbildung 4.12: Gast timeline with distance

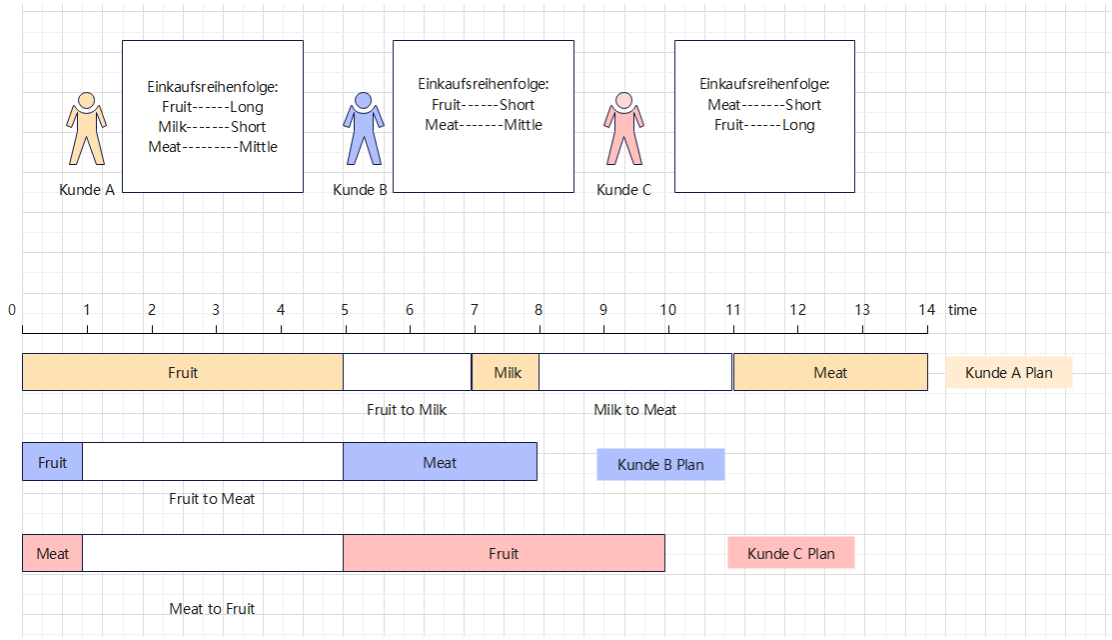

Abbildung 4.13: Area timeline with distance

in die Fleischabteilung und dann in die Obstabteilung. Wie im folgenden Diagramm dargestellt, steht Gelb für Kunde A, Blau für Kunde B und Rot für Kunde C. Wir gehen davon aus, dass der Weg von der Fleischregion zur Obstregion vier Zeiteinheiten beträgt. 4.13

Nicht nur jeder unserer Kunden hat seinen eigenen Zeitplan, sondern auch jeder Bereich des Supermarktes hat seinen eigenen Zeitplan. In diesem Fall entscheiden sich Kunde A und Kunde B dafür, gemeinsam zuerst in der Obstabteilung einzukaufen, nachdem sie den Supermarkt zur gleichen Zeit betreten haben. Aber Kunde B wird bald in den nächsten Bereich weiterziehen, und Kunde A wird lange Zeit hier bleiben, bevor er in den nächsten Bereich weiterzieht. Erst nachdem Kunde A seinen Einkauf beendet hat, kommt Kunde C aus dem vorherigen Bereich und beginnt seinen Einkauf in der Obstabteilung.

Das bedeutet, dass es für den Obstbereich eine Zeitspanne (eine Zeiteinheit lang) vom Beginn der Zeiteinheit 0 bis zum Ende der Zeiteinheit 1 gibt, wenn zwei Kunden gleichzeitig in diesem Bereich einkaufen. Die maximale Anzahl der Kunden beträgt 2. 4.14

Für den Fleischbereich jedoch, weil zuerst Kunde C kommt, dann Kunde B und schließlich Kunde A. Zu keinem Zeitpunkt befindet sich mehr als eine Person in diesem Bereich. Die maximale Anzahl von Kunden ist 1. 4.15

Im Projekt können wir die maximale Anzahl der Kunden in jedem Gebiet nach dieser Berechnungsmethode berechnen. Anhand der Gesamtdatenmenge wird ermittelt, ob diese Höchstzahl überfüllt ist oder nicht, und der Bereich wird in der Nähe des Bereichs mit einer anderen Farbe markiert.

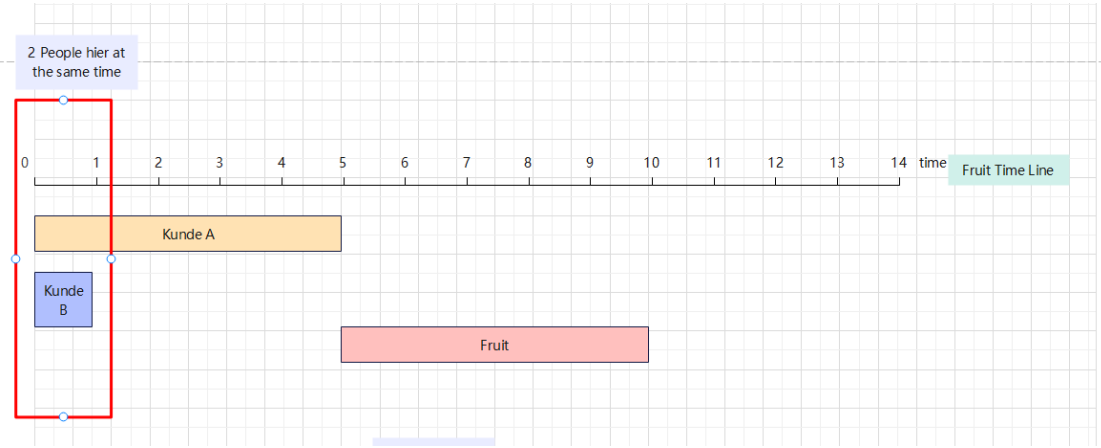

Abbildung 4.14: Area timeline

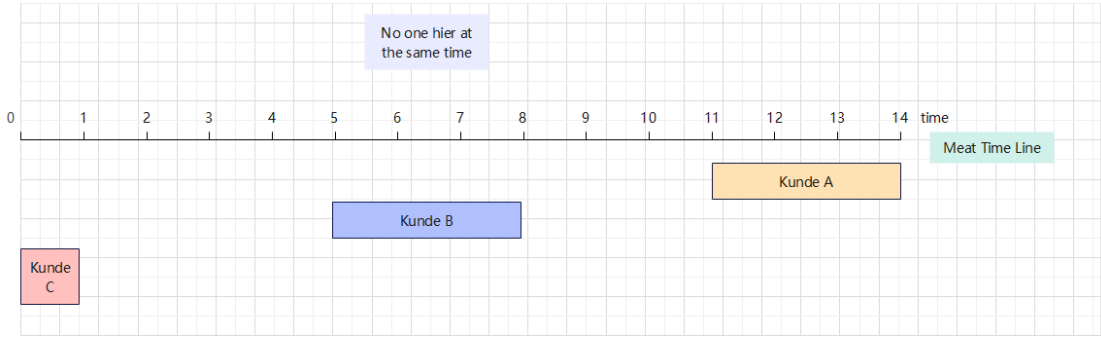

Abbildung 4.15: Area timeline

4.1.4 Vergleich der geschätzten Daten mit den tatsächlich erhaltenen Daten

Es gibt zwei Karten in dieser Umfrage, die sich im Layout unterscheiden, z. B. durch die unterschiedliche Lage der Obst- und Gemüseflächen. Auf der Grundlage des oben beschriebenen Methoden können wir die Verteilung der Kunden auf jeder der beiden Karten ableiten. Wenn der Administrator jedoch wissen möchte, wie sich die Verteilung der Kunden ändert, wenn er das Layout ändert, muss er eine Referenz verwenden, um festzustellen, ob diese Änderung notwendig ist.

Um dies zu erreichen, müssen wir zunächst die Informationen über die Einkaufsreihenfolge der Kunden in der ersten Karte und die Informationen über die Gebiete in der zweiten Karte erhalten. 4.16 Wir platzieren dann die Einkaufsreihenfolge der Kunden aus der ersten Karte in der zweiten Karte. Die Verteilung der Käufer in jedem Gebiet wird dann verwendet, um die Verteilung der Käufer in der zweiten Karte vorherzusagen. Wir vergleichen dann die Verteilung der Kunden nach Gebieten, die wir aus dem Fragebogen erhalten haben.

Zur Veranschaulichung dieses Prozesses werde ich im Folgenden ein einfaches Beispiel anführen. Das folgende Diagramm zeigt den Layout von zwei Supermärkten mit jeweils drei Bereichen A, B und C. Der Unterschied besteht darin, dass die beiden Karten unterschiedliche Layouts haben, d. h. die Entfernungen zueinander sind für die Bereiche A, B und C jeweils unterschiedlich.

Auf der ersten Karte beträgt die Entfernung von A nach B 2 Einheiten, von B nach C 2 Einheiten und von A nach C 3 Einheiten. In der zweiten Karte ist A zu B jedoch 2 Einheiten, B zu C ist 2 Einheiten und A zu C ist 1 Einheit.

An dieser Stelle haben wir zwei Kunden und können aus den Ergebnissen von Umfrage1 die Einkaufsgewohnheiten beider Kunden ableiten. Angenommen, der erste Kunde hält sich lange im Bereich A auf, geht dann für kurze Zeit in den Bereich C und bleibt schließlich für eine mittlere Zeit im Bereich B. Der zweite Kunde hält sich lange in Bereich C auf und verbringt dann jeweils eine kurze Zeit in den Bereichen A und B.

Wir können den Zeitplan für beide Kunden auf der Grundlage der zuvor beschriebenen Methode berechnen. 4.17 Wir kennen dann auch das Layout der zweiten Karte und die Entfernungen zwischen den verschiedenen Bereichen der Karte in Kapitel 2. Wir können dann die Entfernung des Kunden zum nächsten Gebiet in der ersten Zeitstrahl direkt durch die Entfernung der neuen Karte ersetzen und so der Zeitstrahl des Kunden vorhersagen.

Der gleiche Vorgang wird dann durchgeführt, um der Zeitstrahl für jeden Bereich der zweiten Karte im vorhergesagten Szenario zu berechnen.

Wir berechnen die Zeitachse für ein bestimmtes Gebiet der ersten Karte gesondert, und in der Abbildung nehmen wir das Obstgebiet als Beispiel. In den Zeiteinheiten 3-4 sind in dem Bereich die meisten Menschen versammelt. Wir ziehen also die maximale

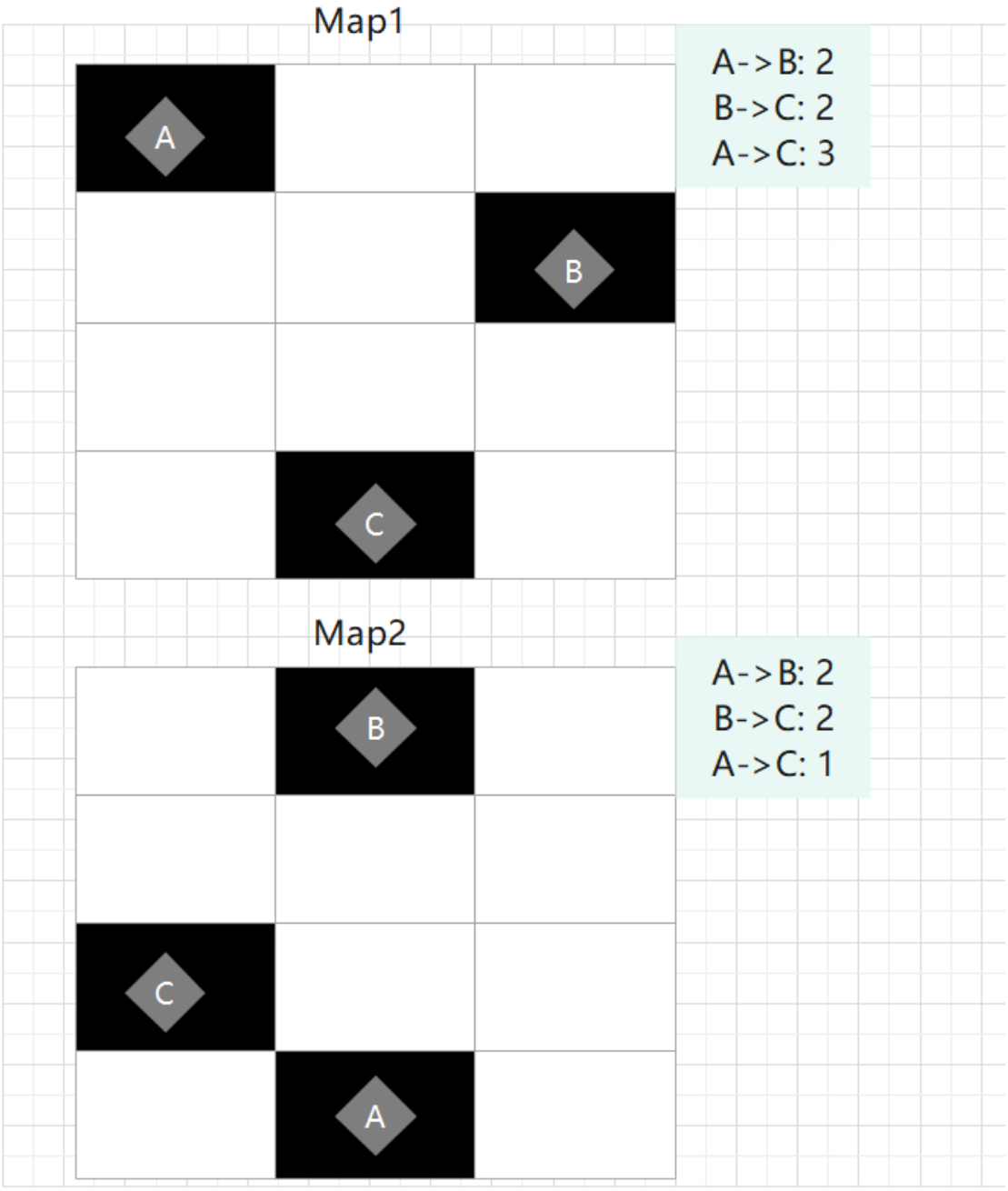

Abbildung 4.16: Two different maps

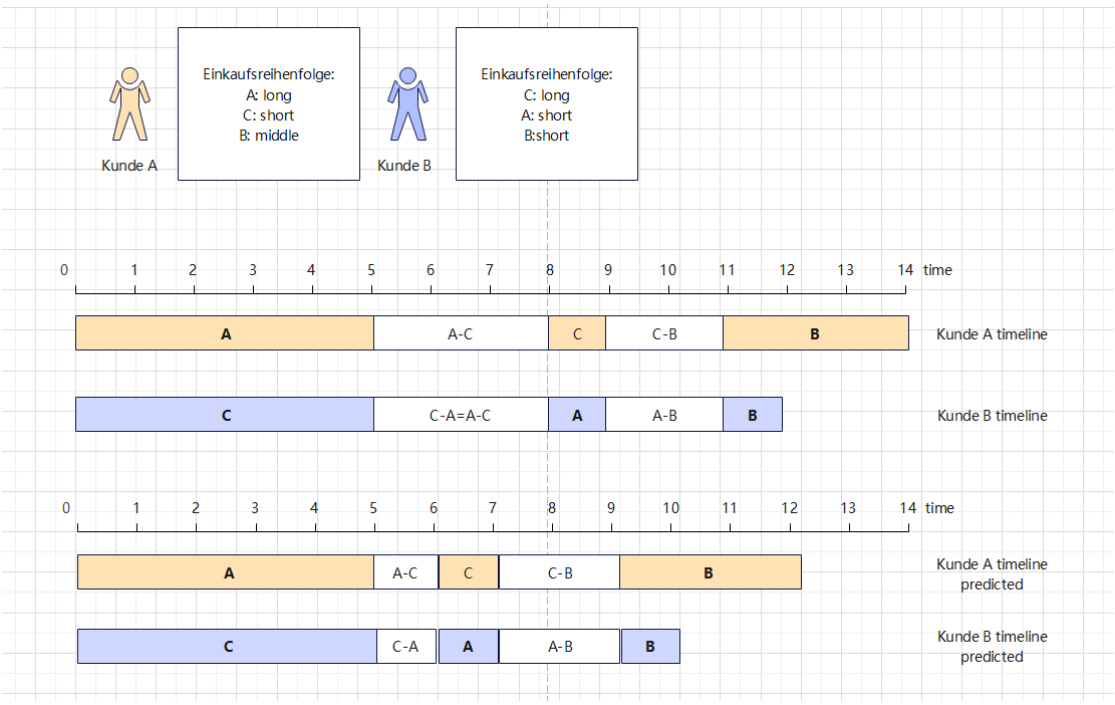

Abbildung 4.17: Two different user timeline

Anzahl von Personen heraus und tragen sie in das Balkendiagramm ein. 4.18

Damit ist das Diagramm für einen Bereich der Karte fertig. Wir verwenden dann die oben beschriebene Methode, um ein Balkendiagramm für denselben Bereich der zweiten Karte zu erhalten, der durch die Kunden-Einkaufsreihenfolge der ersten Karte vorhergesagt wurde, und vergleichen ihn mit dem Balken für denselben Bereich der zweiten Karte, den wir tatsächlich aus der Umfrage erhalten haben.

Nehmen wir als Beispiel für die realen Daten den Obstbereich an einem Samstagabend. Die folgenden drei Karten zeigen nacheinander die Verteilung der Kunden in demselben “Fruit” Bereich auf der ersten Karte (reale Informationen) 4.19, der zweiten Karte der Prognose (Prognoseinformationen) 4.20 und der zweiten Karte des Kapitels (reale Informationen) 4.21.

Anhand der erhaltenen Diagramm kann man grob erkennen, dass die tatsächliche Verteilung von der erwarteten abweicht, je nachdem, welche Informationen der Kunden in der Umfrage angibt, was unter anderem folgende Gründe haben kann:

Erstens gibt die Kunden bei zwei Umfragen nicht die gleiche Einkaufsreihenfolge an, je nachdem, wie der Supermarkt aufgebaut ist. Zweitens kann es vorkommen, dass der Benutzer während des Einkaufsprozesses aufgrund von Änderungen im Layout Bereiche zum ersten Ergebnis hinzugefügt oder entfernt hat, wodurch sich die gesamte Zeitleiste verschoben hat, was zu Fehlern führt.

Der Zweck dieser Umfrage besteht jedoch darin, den Verwaltern die Möglichkeit zu geben, grob zu spekulieren, wie die Ergebnisse einer solchen Änderung aussehen würden,

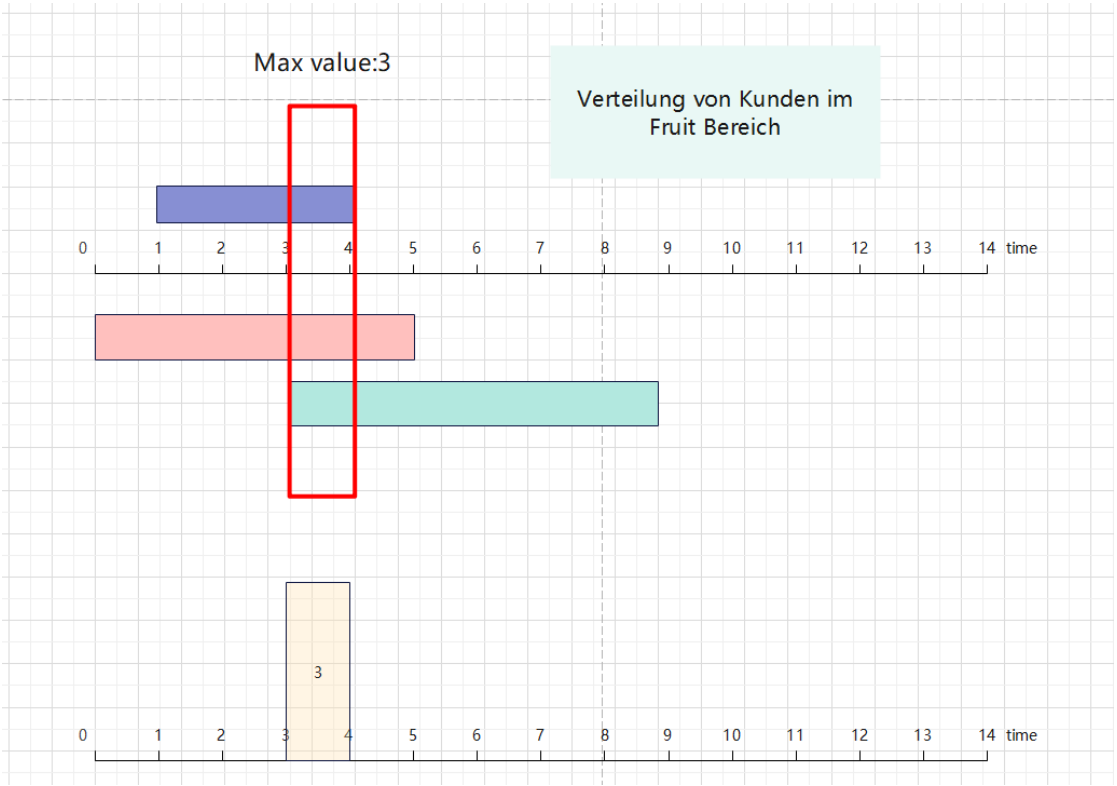

Abbildung 4.18: Calculate the bar chart

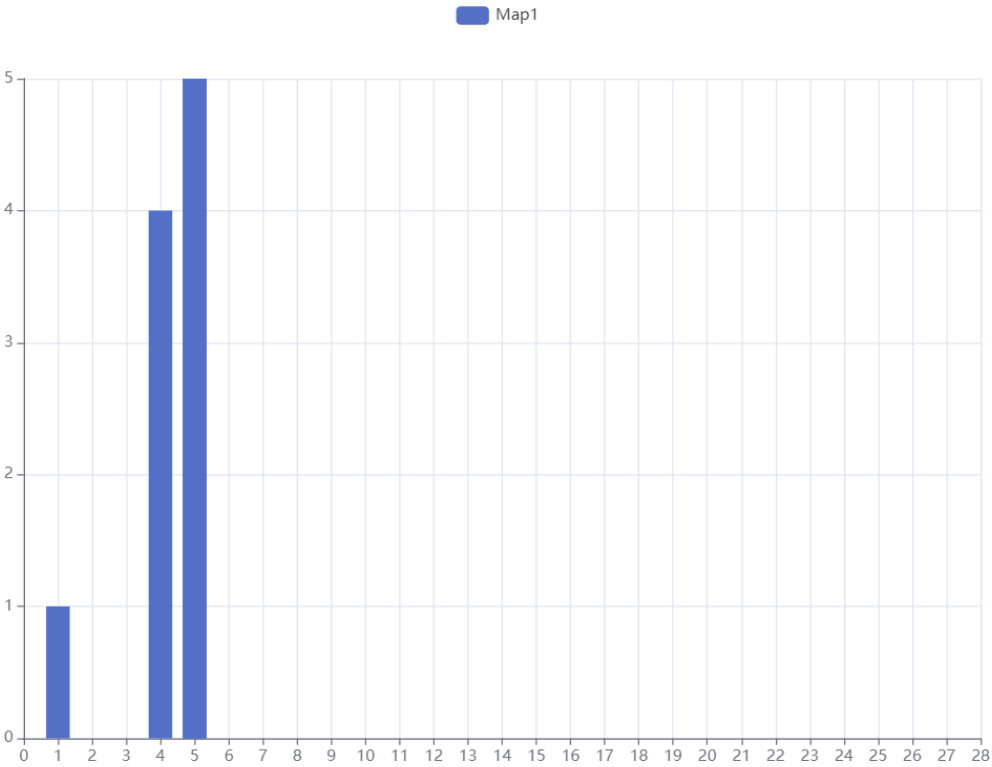

Abbildung 4.19: Bar chart of the Map1 on the area Fruit

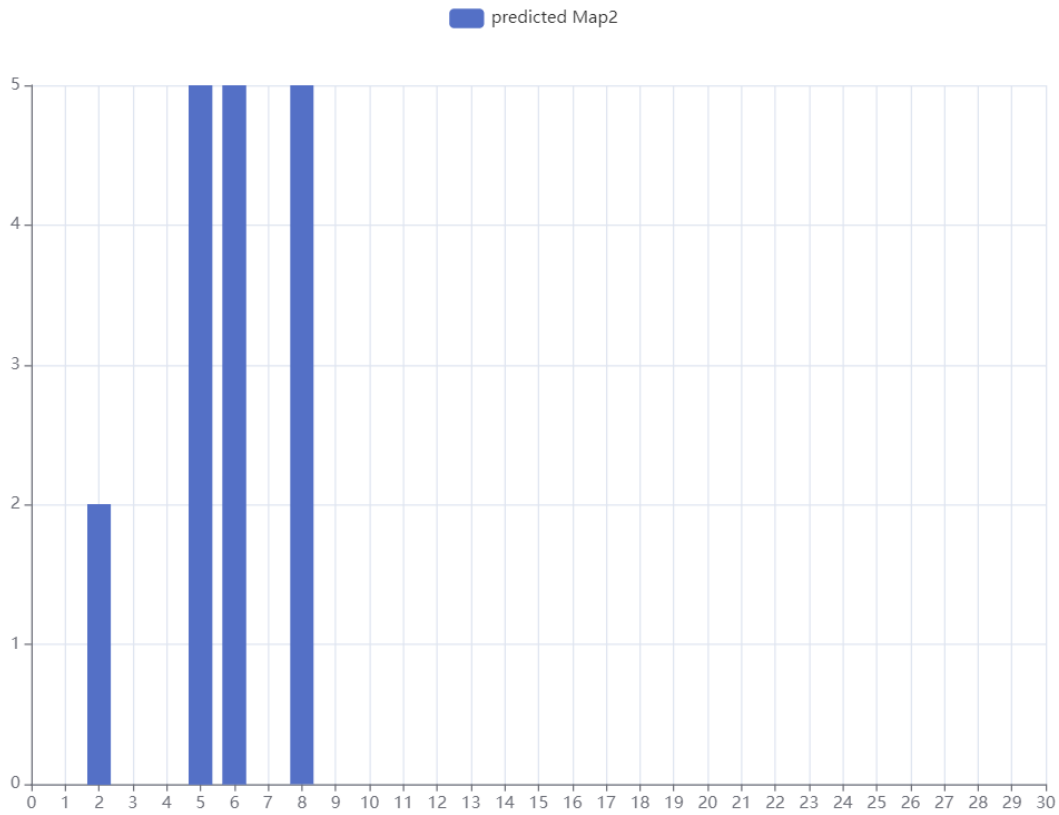

Abbildung 4.20: Bar chart of the predicted Map2 on the area Fruit

Abbildung 4.21: Bar chart of the real data Map2 on the area Fruit

wenn sie das Layout der Umfrage ändern würden. So könnte beispielsweise ein bestimmter Bereich zu einem bestimmten Zeitpunkt stärker frequentiert werden, oder der gesamte Aufbau des Supermarkts würde logischer gestaltet. Unter diesem Gesichtspunkt sind die Grundsätze der Fragebogengestaltung einigermaßen Sinn macht.

4.2 Aufgetretene Schwierigkeiten und Lösungen

In diesem Kapitel geht es um zwei der zeitaufwändigen Fehler, auf die ich bei der Erstellung und Bereitstellung meines Projekts gestoßen bin. Bei der Behebung der Fehler stellte ich fest, dass ich durch diese Fehler etwas Theorie lernen konnte, die ich zuvor übersehen hatte. Hier finden Sie also eine Aufzeichnung des Prozesses von der Entdeckung bis zur Behebung der Fehler.

4.2.1 CORS

Der erste Fehler ist einer, auf den ich bei der Entwicklung der API gestoßen bin. Beim Aufrufen einer Funktion im Backend meldete der Remote-Server, dass der Header Äccess-Control-Allow-Originïn der Quelle der Anforderung aufgrund einer CORS-Policy nicht vorhanden war.

CORS ist auch als 'Cross-origin resource sharing' bekannt. Wie wir bereits beschrieben haben, ermöglicht dieser Mechanismus den Zugriff auf Webseiten mit eingeschränkten Ressourcen durch Seiten aus anderen Bereichen. In unserem Projekt bedeutet dies, dass die Schnittstellen zu den Gleichungen und den Backend-Datenbanken durch Zugriff auf die Namen dieser Funktionen aufgerufen werden. Manchmal können diese Schnittstellen von verschiedenen Seiten aus aufgerufen werden, und CORS macht diese Aufrufe möglich.

CORS-Fehler sind jedoch in der Regel auch auf einen Sicherheitsmechanismus im Browser zurückzuführen, der als 'Same-Origin-Policy' bezeichnet wird und in der Regel dazu dient, eine sehr häufige Art von Netzangriff zu verhindern: 'Cross-Site-Request Fälschung. Bei dieser Art von Angriff werden die im Browser gespeicherten Cookies gefälscht.

Wenn unser Browser eine HTTP-Anfrage an eine Adresse sendet, enthält er ein Cookie für diese Domäne, das dazu beiträgt, die Sitzungsfunktion aufrechtzuerhalten, d. h. in unserem Fall, dass der Anmeldestatus des Benutzers immer gespeichert wird.

Wenn beim Besuch einer Website eine bösartige Seite auftaucht, wird dieser Sicherheitsmechanismus aktiviert. Diese bösartige Seite kann Cookies verwenden, um die Cookies zu erhalten, die auf der vorherigen Seite, auf der der Benutzer war, gespeichert sind. 4.22Mit Hilfe dieser Richtlinie kann die bösartige Website nicht auf die API der ursprünglichen Website zugreifen.

Die erste Lösung für dieses Problem bestand darin, die APIs, auf die Sie zugreifen

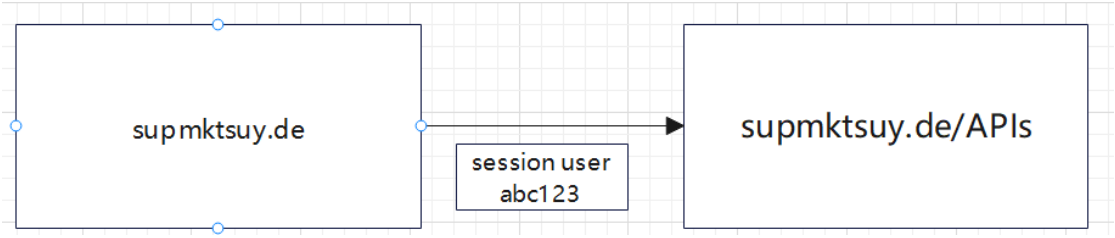

Abbildung 4.22: Session

müssen, genauer zu definieren. Je nach den Funktionen, die die APIs ausführen, wurden die APIs für die Verwaltungsschnittstelle und die Benutzerschnittstelle für die Umfrage aufgeteilt und an unterschiedliche Zugangsadressen gebunden.

Aber als die Anzahl der Backend-Verwaltungsseiten zunahm, war es sinnlos, die APIs einfach aufzuteilen, da die verschiedenen Seiten auch die APIs der anderen Seiten aufrufen würden. Ich musste einen besseren Weg finden, um dieses Problem zu lösen. Dann erfuhr ich von einer Möglichkeit, Proxys zu erstellen. Mit einem Proxy kann die Adresse, auf die ich wirklich zugreifen möchte, aufgelöst werden, so dass ich unabhängig von der Seite, der ich eine Anfrage hinzufüge, auf die gewünschte Seite zugreifen kann, ohne dass ein CORS-Fehler auftritt.

Im Folgenden finden wir ein einfaches Codebeispiel. Auch hier kommt die Middleware des Express-Frameworks zum Einsatz, die Header-Datei 'Access-Control-Allow-Origin' wird zu allen von der Serverseite zurückgegebenen Ergebnissen hinzufügt. Wenn eine Anfrage an meine API geht, geht mein Proxy-Dienst zu einem anderen Server, um die Daten anzufordern. Die '**same-origin policy**', die uns zuvor gestört hat, wird nicht funktionieren. Schließlich gibt der Proxy-Dienst eine Antwort auf der Grundlage der ursprünglichen Anfrage zurück, und diese Antwort überträgt die Daten über eine von der Middleware hinzugefügte Header-Datei.

cors

```
const express = require('express');
const request = require('request');

const app = express();

app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '');
  next();
});

app.get('/mysupermarkt/api', (req, res) => {
  request(
```

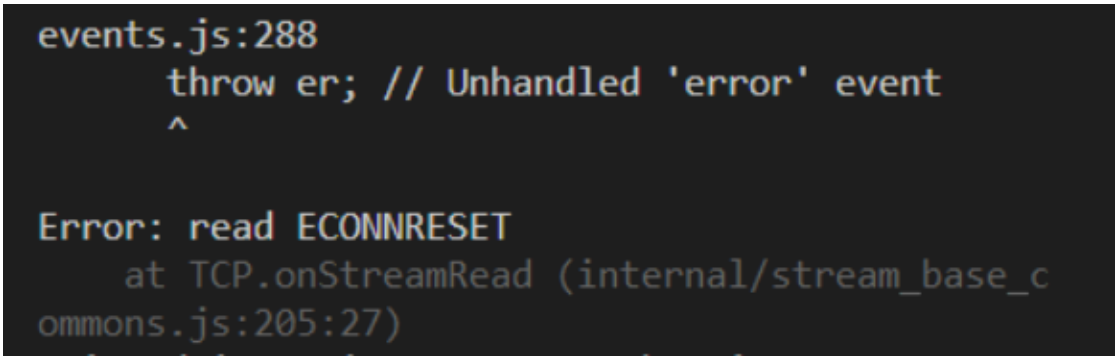

Abbildung 4.23: ECONNRESET

```
      { url: 'https://supmktsuy.de' },
      (error, response, body) => {
        if (error || response.statusCode !== 200) {
          return res.status(500).json
          ({ type: 'error', message: err.message });
        }
        res.json(JSON.parse(body));
      }
    )
  });
```

4.2.2 ECONNRESET Error

Der zweite Fehler, auf den ich stieß und der mich lange Zeit beschäftigte, war der in diesem Diagramm gezeigte ECONNRESET-Lesefehler. Dieser Fehler trat nicht nur auf, nachdem ich das Projekt auf dem Server bereitgestellt hatte, sondern auch, wenn ich localhost startete. Nachdem ich das Projekt veröffentlicht hatte, wurde das localhost-Projekt mit der Datenbank auf dem Server verbunden, um die Fragebogeninformationen von den Benutzern in Echtzeit abzurufen und zu speichern. Dieser Fehler passiert häufig, wenn der Server eine Verbindung zur Datenbank herstellt, so dass die Umfrage-Schnittstelle im Frontend die im Backend entworfenen Karten nicht abrufen kann und die Umfrageergebnisse nicht gespeichert werden können. Außerdem musste der Dienst während des lokalen Entwicklungsprozesses von Zeit zu Zeit neu gestartet werden, um die Verbindung zur Datenbank wiederherzustellen, was die Effizienz der Projektentwicklung ebenfalls stark stört. Ich starte nach dem Grund für diesen Fehler zu suchen. Aus der Beschreibung des Fehlers ECONNRESET wusste ich, dass dieser Fehler bedeutete, dass das andere Ende der TCP-Sitzung seinen Teil der Verbindung geschlossen hatte, was bedeutete, dass die Verbindung zwischen dem Backend des Projekts und der Datenbank auf dem Server plötzlich geschlossen wurde, aber es war nicht bekannt, welche Partei das

Schließen initiiert hatte. Die erste ist, dass der Knoten eine Ausnahme gefangen hat, die er nicht behandeln kann, und deshalb aktiv die Verbindung zum Server trennt. Zweitens wird die TCP-Verbindung selbst getrennt, weil sie länger aktiv war, als sie es hätte sein sollen. Die dritte Möglichkeit ist, dass der Kunde die Trennung der Verbindung nicht veranlasst hat und die Zeit für die Verbindung abgelaufen ist.

Um zu überprüfen, ob die erste Vermutung zutrifft, habe ich das Node-Handbuch und das Änderungsprotokoll durchgesehen und erfahren, dass seit Node v0.9.10 der Master-Slave-Unterdrückungsfehler ECONNRESET nicht mehr auftritt. Frühere Versionen meldeten den Fehler nicht, wenn der Client zusammenbrach, aber jetzt wird er ausgelöst, eine erwartete Funktion von Node angesichts von Fehlern. Wir müssen wirklich die Ursache dieses Fehlers finden, es sollte eine Ausnahme sein, die wir abgefangen, aber nicht behandelt haben.

Um herauszufinden, wo genau diese Ausnahme auftritt, kann sie durch Aufzeichnung des Prozesses ermittelt werden. Dieser Code zeigt grob, wie dieser Prozess läuft.

econn

```
process.on('uncaughtException', function(err) {
    console.log(err.stack);
    console.log('NOT exit...');
});
```

Nachdem ich zu dem Schluss gekommen war, dass die Ursache für die Ausnahme von der externen Datenbank ausging, die aufgrund einer Zeitüberschreitung getrennt wurde, habe ich versucht, die Dauer der aktiven Verbindung mithilfe der pm2-Einstellungen zu verlängern. Wir haben pm2 bereits als Tool beschrieben, das die kontinuierliche Ausführung von node.js auf der Serverseite ermöglicht. Bei einer Datenbank auf demselben Server sollte dieses Problem also durch die Einrichtung einer konstanten Verbindung gelöst werden. Beim Einrichten der Datenbankverbindung haben wir nur den einfachen Befehl mysql.createConnection() verwendet, um eine einzelne Verbindung zu starten, und wenn diese Verbindung ausläuft, bedeutet dies, dass alle Verbindungen getrennt werden.

In MySQL gibt es eine Variable namens wait-timeout, die die Zeit angibt, nach der die Operation abgebrochen wird, d.h. die Verbindung wird automatisch geschlossen, wenn sie eine bestimmte Zeit lang nicht reagiert hat. Zunächst habe ich eine Anweisung eingerichtet, um festzustellen, wann die Verbindung unterbrochen ist, so dass bei einer Unterbrechung automatisch eine neue Verbindung zur Datenbank hergestellt wird. Allerdings gibt es immer noch Probleme mit der Verbindung.

Neben dieser einfachen Methode gibt es auch die Methode createPool(), die einen Pool von Verbindungen erstellt und in der Regel verwendet wird, wenn eine Verbindung zu mehreren Datenbanken besteht.

econn

```
const conn = mysql.createPool(mysql_config);
const conn = mysql.createConnection(mysql_config);
```

Der folgende Code zeigt einen kurzen Prozess zur Erstellung eines Verbindungspools. Wenn eine Verbindung erstellt wird, wird sie im Verbindungspool gespeichert, wenn sie eine Zeit lang nicht verwendet wird, und wenn sie nicht mehr benötigt wird, wird sie aus dem Verbindungspool entfernt. Natürlich wird die Verbindung geschlossen, wenn sie nicht benötigt wird. Wenn eine Verbindung im Verbindungspool ein Fehlerereignis auslöst, wird das Verbindungsobjekt automatisch aus dem Verbindungspool entfernt.

econn

```
const db = mysql.createPool({
    host: 'localhost',
    port: 3306,
    user: 'admin',
    password: '123456',
    database: 'supermarkt'
  })

  db.getConnection((err, connection) => {
    if(err){
      console.log('successful')
    } else {
        console.log('failed')
      db.query('sql', (err, res) => {
        if(err) return console.log(err.message)
        console.log(res)
        connection.release()
        connection.destroy()
      })
    }
    db.end()
  })
```

5 Schlussbetrachtung

Abschließend möchte ich noch einmal die Arbeit, die ich in dieser Bachelorarbeit gemacht habe, das Projekt, das ich entworfen habe, und die Ergebnisse der Realisierung des Projekts zusammenfassen. Dann werde ich auch noch auf die Beschränkungen des Projekts und zukünftige Ausblicke erklären.

5.1 Zusammenfassung

Zunächst definierten wir den Prototyp des Systems und die Funktionen, die es erfüllen sollte. Im zweiten Kapitel dieses Papiers werden einige der theoretischen Kenntnisse vorgestellt, die für den Systementwurfsprozess erforderlich sind. In Kapitel 3 wird beschrieben, wie dieses System Schritt für Schritt vom Nichts zum Etwas umgesetzt wurde. Der Entwurfsprozess und die Gründe für die einzelnen Phasen werden beschrieben. Kapitel 4 konzentriert sich auf die Kernfunktionen, einschließlich der Implementierung der Benutzeranmeldung, der Erstellung und Speicherung von Karten und der Analyse der endgültigen Umfrageergebnisse. Darüber hinaus werden auch einige der während des Produktionsprozesses aufgetretenen Probleme und deren Lösung beschrieben.

Schließlich gibt es noch das Umfragesystem, bei dem die Kunden nach ihrer Einkaufsreihenfolge in zwei Karten mit einigen Unterschieden befragt werden und das den Verwaltern, d. h. dem Supermarktmanager, die Möglichkeit gibt, die Auswirkungen des neuen Layouts auf die Kunden vorherzusagen, wenn sie das Layout des Supermarktes ändern, indem sie sich auf die aus der Umfragen gewonnene Einkaufsreihenfolge der Kunden beziehen. Aus den Umfrageergebnissen können wir ein Balkendiagramm über die Verteilung der Kunden in einem bestimmten Gebiet zu einem bestimmten Zeitpunkt erstellen. Diese Daten werden dann in die neue Karten übernommen, um eine Vorhersage der Kundenverteilung in der neuen Karte zu erhalten.

5.2 Ausblick

Das System bietet den Verwaltern die Möglichkeit, die Auswirkungen ihrer Änderungen am Layout des Supermarktes auf die einzelnen Bereiche in einem Balkendiagramm zu sehen. Bei den Ergebnissen, die durch die konkreten Realdaten zustande kommen, können jedoch einige Abweichungen auftreten, weil die Kunden bei der Umfrage nicht immer in beiden Karten die gleichen Antworten geben.

Dies zeigt nur, dass die tatsächliche Situation komplexer und nicht so einfach ist, wie im Modell dargestellt.

Erstens betreten die Kunden den Supermarkt in der Regel nicht zur gleichen Zeit, und es kann sein, dass die vorherige Gruppe von Kunden noch nicht gegangen ist, wenn

einige Kunden mit dem Einkauf beginnen. Die Zeitkomplexität ist daher viel höher als im Modell.

Zweitens wird die Einkaufsreihenfolge des Nutzers durch das neue Layout leicht beeinflusst, und er wird je nach Layout seine Einkäufe hinzufügen oder ändern.

Drittens kann auch die Methode zur Berechnung der Fahrten der Nutzer von einem Gebiet zum nächsten optimiert werden.

Viertens kann auch das Layout des Supermarkts im System auch mehr Möglichkeiten anbieten, wie z. B. die Anpassung der Größe der Regale und die Aufstellung mehrerer identischer Regale.

5.3 Danksagung

Hiermit möchte mich bei meinen beiden Betreuern, Frau Prof. Dr. Rebekka Axthelm und Frau Prof. Dr. Doris Bohnet , für die zahlreiche hilfreiche Ratschläge und Ideen bedanken.

Schließlich möchte ich meinen Eltern für die Unterstützung und Ermutigung danken, die sie mir gegeben haben.

Literaturverzeichnis

1.layUI Dokumente: https://layui.itze.cn/
2. layUI xAdmin Doc: https://github.com/wongyuxuan/xadmin
3. Node.js https://nodejs.org/de/about/
4. Webentwicklung https://www.get-in-it.de/magazin/arbeitswelt/it-berufe/was-macht-ein-webentwickler
5. Express Framework'https://www.tutorialspoint.com/nodejs/nodejs_express_framework.html'
6. Express RESTful API Design: https://www.robinwieruch.de/node-express-server-rest-api/
7. Ajax RESTful https://silvae86.github.io/2018/12/14/differences-between-ajax-and-rest/
8. Anforderungsanalyse https://www.simplilearn.com/what-is-requirement-analysis-article
9. npm document https://docs.npmjs.com/
10. Express routen https://expressjs.com/en/guide/routing.html
11. body-parser https://github.com/expressjs/body-parser
12. Node.js Mysql https://www.mysqltutorial.org/mysql-nodejs/
13. CRUD Node.js https://zellwk.com/blog/crud-express-mongodb/
14. Ajax kapseln https://www.psd-tutorials.de/threads/ajax-aufurf-in-objekt-kapseln.131128/
15. Express API https://stackabuse.com/building-a-rest-api-with-node-and-express/
16. Forward and reverse proxies. The Apache Software Foundation. [2011-02-09].
17. CORS https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors
18. ECONNRESET [https://stackoverflow.com/questions/17245881/how-do-i-debug-error-econnreset-in-node-js]

Abbildungsverzeichnis