Zentrales Kassensystem mit RFID-basierter Identifikation

Technikerarbeit 2022 – IT-Schule Stuttgart

Florian Werner

24.03.2022

Zentrales Kassensystem mit RFID-basierter Identifikation

Technikerarbeit 2022 – IT-Schule Stuttgart

Hiermit erkläre ich, dass die vorliegende Technikerarbeit eine eigenständige Leistung darstellt und nicht auf der Basis einer bereits vorhandenen Techniker-, Diplom- oder ähnlicher Arbeit erstellt wurde. Verwendete Quellen habe ich vollständig und nachprüfbar aufgeführt. Bei der Durchführung und Ausarbeitung wurden nur die zulässigen Hilfsmittel verwendet.

Mir ist bewusst, dass bei einem Verstoß gegen diese Erklärung innerhalb der gesetzlichen Einspruchsfristen auch im Nachhinein die Leistungsbewertung aberkannt werden kann. Damit erlischt die Berechtigung zum Tragen der Berufsbezeichnung „staatlich geprüfter Techniker“.

Bedanken möchte ich mich bei Herrn Selwan Gorel-Brwari und meinem Verein der DLRG OG Lauffen für die Unterstützung und Betreuung bei der Erstellung der Technikerarbeit.

Lauffen am Neckar, den 24.03.2022

_____

Unterschrift

Inhaltsverzeichnis

1. Überblick

Diese Dokumentation beschreibt die Planung, Durchführung und Auswertung meiner Technikerarbeit. Die angesprochene Zielgruppe dieses Dokumentes ist der Betreiber des digitalen Kassensystems. Es werden hier alle durchgeführten Maßnahmen beschrieben und erläutert, welche Gedanken in der Entwicklung aufgekommen sind und wie diese technisch umgesetzt wurden.

2. Vorwort

Die Projektarbeit hat mit dem Vorschlag begonnen innerhalb der DLRG Ortsgruppe Lauffen das manuelle Kassensystem zu digitalisieren. Diese waren mit der Nutzung von Stempelkarten und den dazugehörigen Ausdrucken nicht zufrieden. Ein digitales System soll hier zum Einsatz kommen, das für einen Verein auch bezahlbar ist. Die Beauftragung erfolgte durch den damaligen Kassierer an den Technikerschüler Florian Werner der hier das Potential sah seine Technikerarbeit zu gestalten. Florian ist selbst Mitglied der Ortsgruppe, so dass er die Vorrausetzungen genau kannte und die Planung selbstständig durchführen konnte. Für die Nutzer ist es wichtig jeglichen erstellten Code öffentlich zu sehen und das Verfahren zu analysieren. Deshalb wird die komplette Umsetzung als Open-Source Projekt der Öffentlichkeit auf GitHub zur Verfügung stehen.

3. Einführung

Zugunsten der Übersichtlichkeit des Projektes wird hier die Einführung für einen allgemeinen Überblick genutzt. Die wichtigste Funktion der Technikerarbeit ist es Waren / Artikel über ein Terminal am Kühlschrank zu erkaufen. Da dies aber jeder selbstständig durchführen soll, war der Gedanke schnell auf personalisierte RFID-Chips gefallen. Diese soll die Speicherung der Benutzerdaten, Guthaben und Chips ermöglichen. Zusätzlich soll es dem Verwalter ermöglicht werden Datensätze zu erstellen / bearbeiten und zu löschen.

4. IST-Zustand

Wie im Punkt 2 „Vorwort“ schon angedeutet, ist die Verwaltung der Guthaben mit Guthabenkarten realisiert worden. Hier konnten sich die Mitglieder Karten im Wert von 10, 20 und 50 € beim Kassierer erwerben.

Die Karten wurden immer manuell erstellt und gedruckt. Dies war bei der Beschaffung ein zu hoher Akt. Eine Historie der Käufe von Artikeln und Guthabenkarten ist nie geführt und dokumentiert worden.

Der Kauf erfolgt durch die Entnahme des gewünschten Artikels oder Getränk und die Quittierung auf der Guthabenkarte.

Da die manuellen Tätigkeiten dieses Prozesses abgeschafft und ein einheitliches digitales System erstellt werden muss, ist die Durchführung des Projektes von hoher Relevanz.

5. Soll-Zustand

Für die Benutzung des digitalen Kassensystem wird den Nutzern ein Terminal zur Verfügung stehen. An diesem soll es ihnen ermöglicht werden, mithilfe eines RFID-Chip und Lesegerät, sich zu autorisieren. Weiterhin ist für die Benutzung das LCD-Touchdisplay vorhanden, an dem die Käufe von Artikeln und Getränke durchgeführt werden.

Die Speicherung aller Daten soll auf einem zentralen Backend in der Cloud zur Verfügung stehen. Dies soll mithilfe einer MySQL Datenbank erfolgen und bei dem Cloud Provider Hetzner erfolgen.

Da zusätzlich eine Zentrale Verwaltungsschnittstelle benötigt wird, um die administrativen Tätigkeiten durchzuführen, wird ein Webservice benötigt, dass Änderungsmaßnahmen durchführen lässt. Dies soll als Frontend ebenfalls auf der Hetzner Cloud gehostet werden und den Verwaltern zur Verfügung stehen.

Alle zur Verfügung stehenden Artikel werden im Rahmen der Technikerarbeit nur einen Euro kosten. Eine Möglichkeit der Preiserhöhung oder Änderung ist aktuell nicht vorgesehen.

6. Rahmenbedingungen

Die Technikerarbeit ist ein eigenentwickeltes Projekt von dem Autor der Dokumentation. Für die Entwicklung aller Applikationen werden unterschiedliche Programmiersprachen mit dazugehörigen Frameworks zum Einsatz gebracht. Diese werden in dieser Dokumentation vorgestellt und auch wie sie innerhalb des Projektes genutzt werden.

Für das Erlernen der unterschiedlichen Programmiersprachen, wurden Online-Kurse, Fachliteraturen oder die Produktspezifischen Dokumentationen durchgearbeitet. Generell ist die Graphische „Schönheit“ der unterschiedlichen Applikationen kein ausschlaggebendes Ziel der Technikerarbeit. Die Dokumentation wurde in einem natürlichen Schreibstil beschrieben. Des Weiteren wurde der erstellte Programmiercode auf einem USB-Stick im Anhang beigefügt. Der Autor ist selbst auch der vollumfangende Entwickler.

7. Terminplanung

Die Zeitliche Terminplanung ist ebenfalls anhänglich der Dokumentation beigefügt.

8. Kostenplanung

Die Kostenanalyse des Gesamtprojektes wird folgend aufgelistet. Alle Änderungen zum Pflichtenheft werden detailliert beschrieben.

8.1 Hardwarekosten

Name	Anzahl	Kosten
Raspberry Pi 4 Computer Modell B, 2GB RAM	1	49,80 €
Raspberry Pi USB-C Netzteil 5,1V / 3,0A, EU	1	7,90 €
SanDisk Extreme micro SDHC A1 UHS-I U3 Speicherkarte	1	8,85 €
7 Inch IPS Touch Screen für Raspberry Pi	1	49,38 €
OPEN-SMART RC522 RFID Kartenleser	1	1,92 €

Tabelle 1 Hardwarekosten

Im Laufe der Projektphasen wurde das Model der Raspberry Pi auf die Version 4 mit 2 GB Arbeitsspeicher aufgestockt.

8.2 Softwarekosten:

Name	Anzahl	Kosten
JetBrains - IntelliJ IDEA Ultimate	1	0,00 €
JetBrains - PyCharm	1	0,00€
Qt Creator – Community	1	0,00€

Tabelle 2 Softwarekosten

Da die Technikerarbeit nicht kommerziell verkauft wird, kann bei den Softwareprodukten auch die Education Version von JetBrains eingesetzt werden. Diese hat den gleichen Funktionsumfang wie die im Pflichtenheft beschriebene Ultimate Edition.

8.3 Infrastruktur

Domain:

Provider	Anzahl	Jährliche Kosten
INWX – „.cloud“ Domain	1	1,50€

Tabelle 3 Domainkosten

vServer:

Cloud Server – Hetzner	Anzahl	Monatliche Kosten
CPX11	1	4,75 €

Tabelle 4 Serverkosten

Die Kosten des vServer und der Domain starten im Monat 09/2021.

8.4 Kostensatz

Einmalige Kosten	= 117,95
Hardware	117,95 €
Software	0,00 €
Jährliche Kosten	= 58,50
vServer	57,00 €
Domain	1,5

Tabelle 5 Kostensatz

9. Benutzerprofile / Ausführungsrichtlinien

Das Projekt unterscheidet in drei Benutzerprofile. Diese kapseln die Funktionen und Aufgaben des Systems ab. Folgend werden die Benutzer und Funktionen gruppiert. Das Use-Case Diagramm zeigt die dazugehörigen Ausführungsrichtlinien der Einzelnen Profile mit auf.

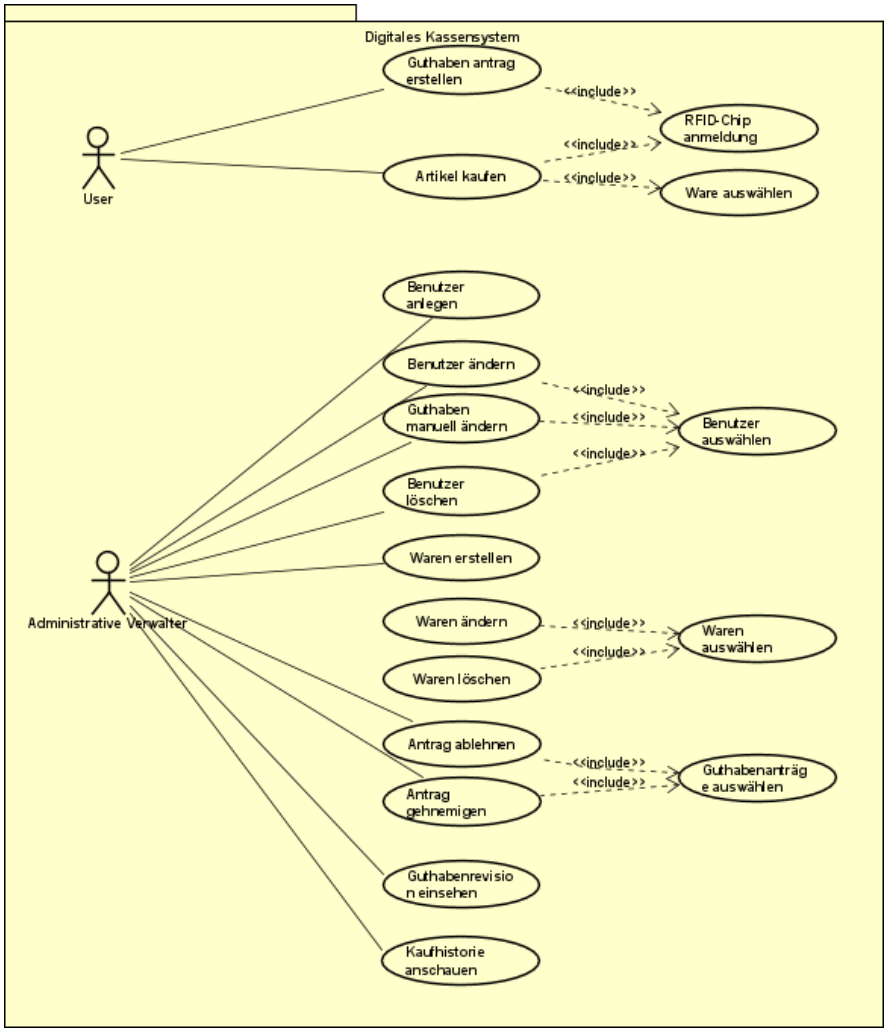

Abbildung 1 USE-Case Diagramm

Developer:

Der Developer ist als Entwickler des Gesamtprojektes definiert. Er erhält die Berechtigung auf jeglichen Systemen administrative Maßnahmen zu treffen. Zusätzlich ist er über den kompletten Aufbau und den Gedankengängen der Entwicklung informiert. Er kann selbstständig Änderungen im Gesamtprojekt vornehmen und diese an die benutzten Systeme ausrollen.

Administrative Verwalter:

Der administrative Verwalter verfügt über die Funktionen, Benutzer, Waren und Guthaben zu administrieren und zu ändern. Zusätzlich steht ihm die Möglichkeiten zu, Historien von Käufen und Guthaben Aufträge einzusehen. Folgende Funktionen stehen im zur Verfügung.

Benutzer:

- Erstellen
- Ändern
- Löschen

Ware:

- Erstellen
- Ändern
- Löschen

Guthaben Anträge

- Genehmigen
- Ablehnen

Guthaben Revision

- Ansehen

Kauf Historie

- Ansehen

User/Benutzer:

Den Benutzern ist es möglich, sich am Terminal mithilfe ihres RFID-Chips anzumelden. Anschließend können sie sich in den Bereichen „Käufe“ und „Guthaben“ aufhalten. Bei den Käufen erhalten Sie die Funktion, der Auswahl der Waren und den abschließenden Kauf der Ware. Des Weiteren kann im Bereich Guthaben das aktuelle Guthaben eingesehen werden und ein Antrag auf eine Erhöhung gestellt werden. Die Anträge der Guthaben müssen aber durch die Administrative Verwalter genehmigt werden.

Dem User stehen folgende Funktionen zur Verfügung:

Anmeldung RFID

- Waren auswählen / kaufen
- Guthabenantrag stellen

10. Planung und BIG Picture

Die technische Planung des digitalen Kassensystems wurde in drei Fachbereiche aufgeteilt: „Backend“, „Terminal“ und das „Frontend“. Jedes der Fachbereiche ist für die Nutzung unabdingbar.

Das Backend ist die zentrale Speicherung aller Benutzer, Waren, Guthabenanträge und Käufe. Es besteht aus einer MySQL Datenbank. Da die Entwicklung der anderen Fachbereiche aber weitestgehend unabhängig sein sollte, wurde die Realisierung der Anbindung in einer zweiten Komponente ausgelagert. Alle Anfragen auf die Datenbank erfolgen über einen sogenannten Rest Controller. Dieser ermöglicht es mit „HTTP-Anfragen“, wie Beispielsweise „GET“, „POST“, „PATCH“ und „Delete“, Daten in die Datenbank zu speichern. Der Gedanke zu dieser Entwicklung lag daran, dass weiterführende Programmiersprachen keinen MySQL-Connector benötigen, sondern alles über HTTP-Anfragen abbilden. Vor allem in der Webentwicklung ist es ein gängiger Prozess den Datenverkehr über eine REST-API (11.8 Definition Rest-API) abzubilden. Zusätzlich werden alle Steuerungen auf dem Backend durchgeführt. Der Rest-Controller bildet zusätzlich die Logik des Kassensystem ab. Prozesse wie das Verhalten von Guthaben Anträge und Waren Anzahl sind hier ausschlaggebend. Da das Backend überall erreichbar sein muss, wird es eine zentrale Platzierung in der Cloud bekommen.

Der zweite Fachbereich umfasst das administrative Frontend. Da die Administration aller Accounts nur den Administrativen Benutzern zusteht, soll diese Aufgabe separiert ausgelagert werden. Benötigt ist ein Webservice der alle definierten Werte Sehen/Bearbeiten und Löschen kann. Zusätzlich ist die Einsicht der Historien ein wichtiger Punkt.

Der dritte und letzte Fachbereich der Planung ist das „Terminal“ (Client). Dies muss für alle Benutzer eine Übersichtliche Oberfläche besitzen und eine gute Bedienung darstellen. Zusätzlich ist die Authentifizierung am Terminal mithilfe eines RFID-Chips notwendig. Geplant war hier eine Raspberry PI 3 Model B mit einem 7 Inch Touch Display. Die Anbindung des RFID-Chip Readers ist über die GPIOs des Raspberry PIs realisierbar. Alle auszuführenden Operationen werden am Terminal über HTTP-Aufrufe an das Backend erfolgen. Für die Nutzung wird aber an der Raspberry PI eine durchgehende Verbindung vorausgesetzt.

Anhand der Planung wurde folgendes BIG Picture gestaltet:

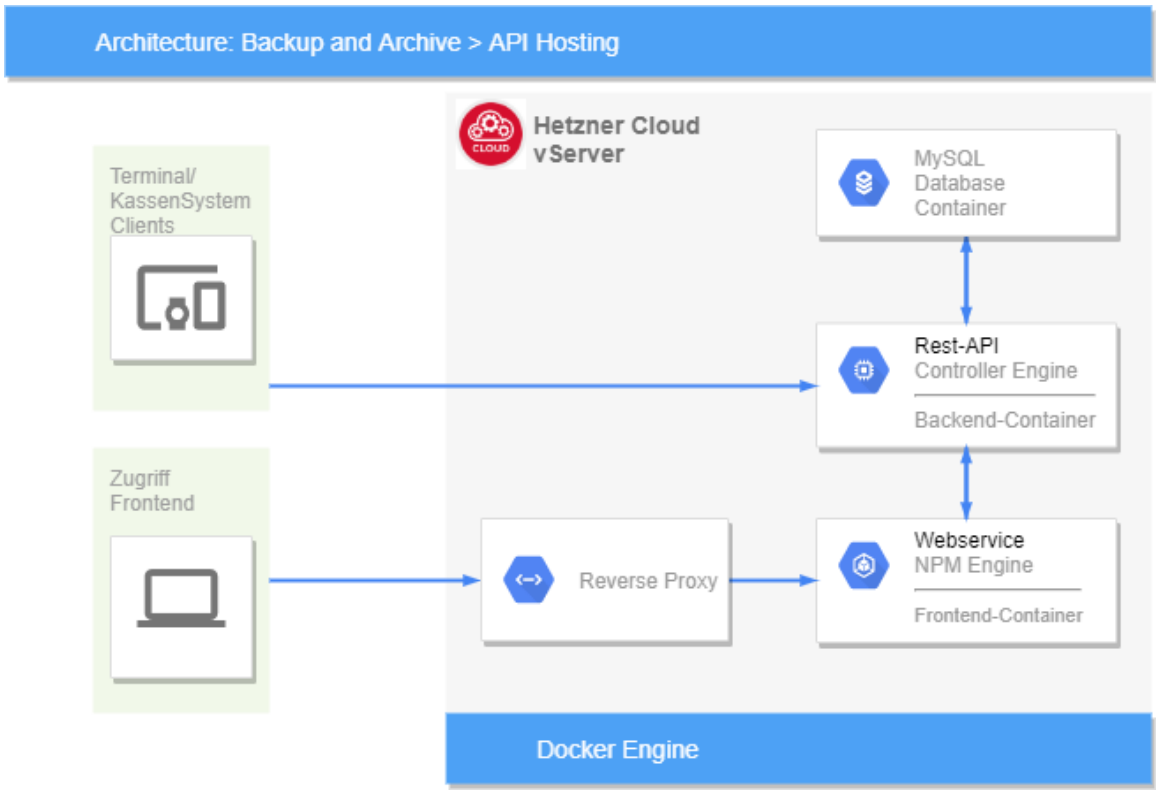

Abbildung 2 BIG Picture

11. Technische Umsetzung Backend

11.1 Auswahl der Software

Die Technische Umsetzung des Backend erfordert die Speicherung von Daten im Hintergrund. Schnell ist hier der Entschluss gefallen, eine MySQL Datenbank zu verwenden. MySQL ist ein relationales Datenbanksystem, das weltweit bekannt ist und eingesetzt wird. Entwickler der Server Applikation ist die Firma Oracle (Literaturverzeichnis Lfd. Nr. 1). Alle Daten werden auf diesem System gesichert. Da die Verbindung zu der MySQL Datenbank über einen Rest Controller erfolgt, war hier die Auswahl des benötigten Frameworks unschlüssig. Folgende Frameworks und dazugehörige Programmiersprachen sind dort zur Auswahl gestanden:

- Spring Boot – basierend auf Java
- FastAPI – basierend auf Python

Da ich als Entwickler in den verschiedenen Bereichen aber unterschiedliche Programmiersprachen bevorzuge, ist die Auswahl in diesem Falle auf Spring Boot gefallen. Diese Aussage resultiert daraus, dass Fehler und Bugs, die innerhalb einer Programmiersprache vorhanden sind, nicht auf die Gesamtbreite des Projektes Auswirkungen zeigen. Zusätzlich bringt Spring Boot Erweiterungen mit, die eine Implementierung von Sicherheitsrelevanten Features enorm vereinfachen.

Spring Boot im Bezug zu Java:

Das Spring Framework ist ein quelloffenes Framework, das sich einfach in Java Projekte implementieren lässt. Das Spring Framework bietet ein weites Spektrum an Funktionalitäten, dass zur Entwicklung unterschiedlicher Geschäftsanwendungen dient. Entwickelt wurde es von der Firma VMware im Jahr 2002 (Literaturverzeichnis Lfd. Nr. 2). Spring ermöglicht es die Programmierung von Enterprise Applikationen in Java deutlich zu vereinfachen. Für die einfache Einbindung aller benötigten Bibliotheken und dazugehörigen Abhängigkeiten wird das Spring Framework im Zusammenhang mit Spring Boot verwendet. Spring Boot erstellt die Grundkonfiguration eines neuen Java Projektes, mit allen benötigten Abhängigkeiten, die eine Applikation benötigt. Beispielsweise die Einbindung von Konnektoren unterschiedlicher Datenbanken oder die Integration relevanter Sicherheitsfeatures. Diese lassen sich über folgende URL grundkonfigurieren: (URL Verzeichnis 1).Ein Weiterer Vorteil für die Einsetzung von Spring Boot ist das Zusammenspiel mit der Jakarta Persistence API. Dies wird im Kapitel 11.3 Objekt Relationale Abbildungen genauer beschrieben.

11.2 Softwareversionen und Abhängigkeiten

Folgende Softwareversionen werden im Backend genutzt:

Datenbank:

MySQL-Server

Version: 8.0.28

Rest Controller:

Projekt:

- Spring Boot

Version: 2.6.1

Framework:

- Spring

Ausgewählte Abhängigkeiten:

- Spring WEB (spring-boot-starter-web)
- Rest Repositories (spring-boot-starter-data-rest)
- MySQL Driver (mysql-connector-java)
- Spring Data JPA (spring-boot-starter-data-jpa)

11.3 Objekt Relationale Abbildungen

Objekt Relationale Abbildungen ist eine gängige Methode für die Erstellung von Objektorientierten Klassen innerhalb Javas, die als relationales Modell auf einer Datenbank abgebildet werden. Für die Entwicklung werden alle Datenbankmodelle in Java Klassen abgebildet und festgelegt. Änderungen an Tabellen und Strukturen erfolgt ausschließlich über die erstellten Klassen im Java Projekt.

Eine Java Klasse die als Datenbank Tabelle angelegt werden soll, muss in sich mit der Annotation „@Entity“ definiert sein. Die Jakarta Persistence API weiß durch die Definition von Annotationen, welche Eigenschaften dahinterstecken und innerhalb der Datenbank modelliert werden müssen. Zusätzlich wird der Public Key der Tabelle mit der folgenden Annotation festgelegt:

„@ID“
„@GeneratedValue(strategy = GenerationType.Identity)“

Jakarta definiert den Public mit dem Typ: Identity. Dies ist das Prinzip des „Auto Increment“. Die Erstellung der Datenbank Attribute erfolgt durch die Hinterlegung von Attributen innerhalb der Java Klasse und dem dazugehörigen Datentyp.

11.3 JSON String

Die Rückgabe der http-Anfragen auf das Backend sind JSON-Strings. JSON ist die „JavaScript Object Notation“. Dies ist ein kompaktes Datenformat für den Austausch von Datenobjekten zwischen Applikationen (Literaturverzeichnis Lfd. Nr. 3). Da JSON in allen gängigen Programmiersprachen als Datenaustauschformat unterstützt wird, ist der Einsatzgrad enorm hoch. Gespeicherte Variablen können in folgende Datentypen innerhalb eines JSON Strings gespeichert werden: Boolean, Zeichenketten, Zahlen und Nullwerte. Zusätzlich lassen sich JSON Strings innerhalb der Programmiersprachen in Objekte oder Objektlisten umwandeln (Literaturverzeichnis Lfd. Nr. 4). Das erleichtert die Arbeit beim Austausch von Daten enorm, da Objekte anhand der Struktur erzeugt werden und nicht durch Logikbausteine konvertiert werden.

11.4 Aufbau JAVA Projekt „Backend“

Da ein strukturierter Aufbau eines Java Projektes relevant für das Verständnis und die Übersicht ist, wurde hier auf Trennung der einzelnen Funktion und Eigenschaft der Klassen geachtet.

Folgende Strukturen(Verzeichnisse) wurde hier gewählt:

- de.dlrg.backend.
  - Controller
  - Entity
  - ENUM
  - Repository
  - Time
  - Ressourcen

Detailliert werden die einzelnen Verzeichnisse in den folgenden Kapitel beschrieben.

Da der Ordner „Time“ nur eine Klasse beinhaltet ist eine separate Erläuterung aber nicht notwendig. Dieser erzeugt nur bei Initialisierung des Objektes eine Timestamp des Systems. Dieser wird beim Aufruf der GET-Methode als Objekt von SQL-Date zurückgegeben.

11.5 Entity

In der Struktur „Entity“ sind alle Einheiten des digitalen Kassensystems hinterlegt. Diese sind die Benutzer, Waren, Käufe, Guthaben Aufträge und die Guthaben Revision als Java Klassen. Die Klassen definieren die Struktur wie die Tabellen innerhalb der MySQL Datenbank erstellt und gespeichert werden.

Für diesen Prozess kommt die im Kapitel 11.3 Objekt Relationale Abbildungen beschriebene Technologie zum Einsatz.

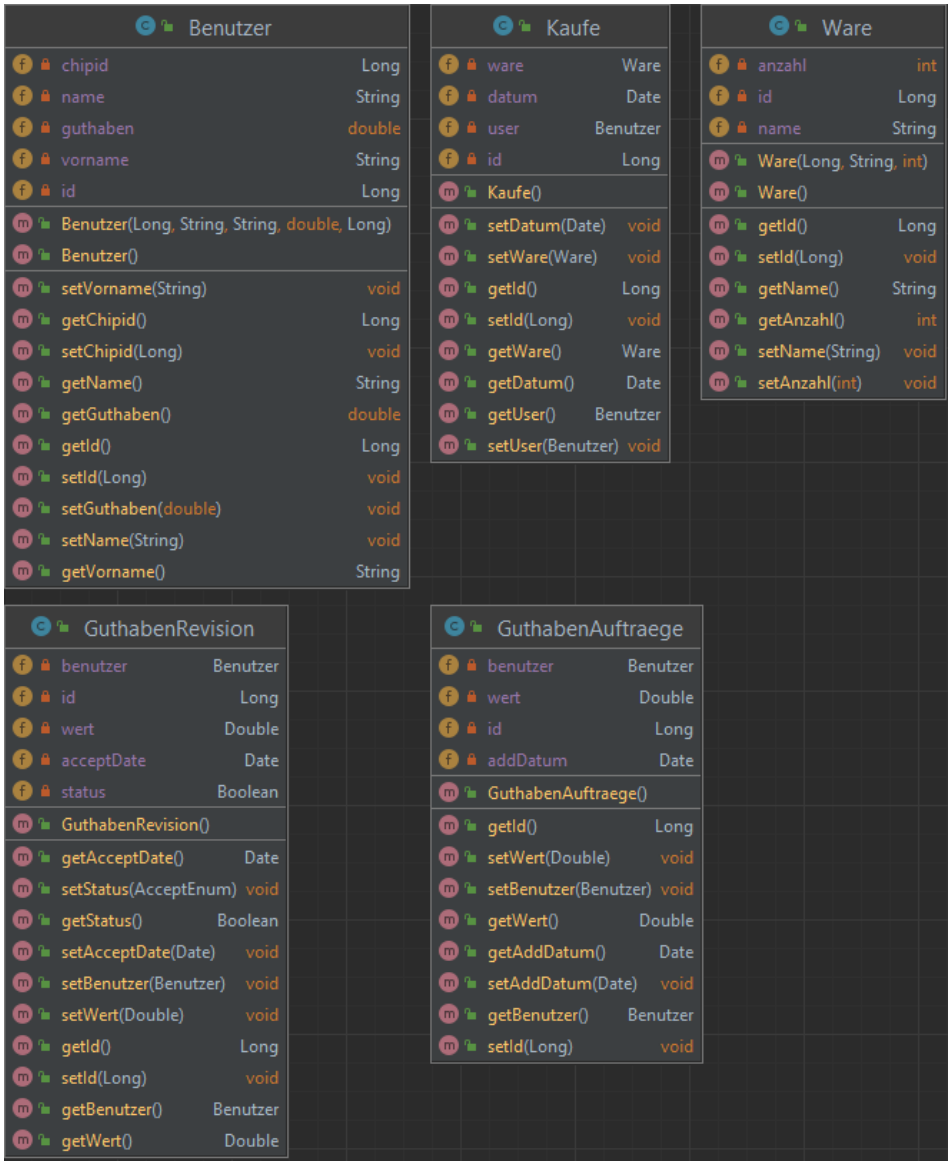

Abbildung 3 Klassendiagramm

Wie in der Abbildung zu sehen sind alle Klassen ähnlich aufgebaut. Sie bestehen aus Attributen, Konstruktoren und den GET/SET-Methoden.

Folgende Strategien verfolgen die Klassen:

KLASSENNAME	STRATEGIE
BENUTZER	Definition der Benutzerobjekt
KAUFE	Speicherung aller Käufe / Historie
WARE	Definition der Warenobjekte
GUTHABENREVISION	Speicherung aller Guthaben Aufträge / Historie
GUTHABENAUFTRÄGE	Aktuell zu bearbeitende Guthabenaufträge

Tabelle 6 Backendklassen

Da alle Attribute bei der Initialisierung des Projektes auf der Datenbank erzeugt werden, ist die Speicherung der genauen Verbindungskonfiguration notwendig. Diese werden innerhalb folgender Datei „application.properties“ im Java Projekt hinterlegt.

Diese Datei definiert die folgenden Parameter:

- URL -> MySQL Server URL
- Username -> MySQL Benutzer
- Passwort -> MySQL Passwort
- Treiber Klasse -> JDBC Treiberklassenname
- Datenbank Initialisierungstyp -> Create/Drop/Update

11.6 Repository

Für den Zugriff auf die Datenobjekte innerhalb der Spring Boot Applikation werden für jede Klassen der Entitäten, „Repository Interfaces“ angelegt. Diese Interfaces erben von dem Interface „JpaRepository“. Bei der Implementierung des Interfaces „JpaRepository“ wird die Klasse der jeweiligen Entität und dem Attributen typ des „Primary Key“ angegeben. Die „JPA Repositories“ ermöglichen es Abfragen auf die Datenbank durchzuführen. Diese werden benötigt, um dem Projekt Abfragen zu definieren, aus denen die jeweiligen Controller den Zugriff auf die Datenbank erhalten. Bei der Implementierung der „Repositories“ analysiert Spring die Methoden der Entitäten und generiert automatische Abfragen aus den Methodennamen (Literaturverzeichnis Lfd. Nr. 5). Folgende Interfaces für die „Repositories“ wurden dafür angelegt:

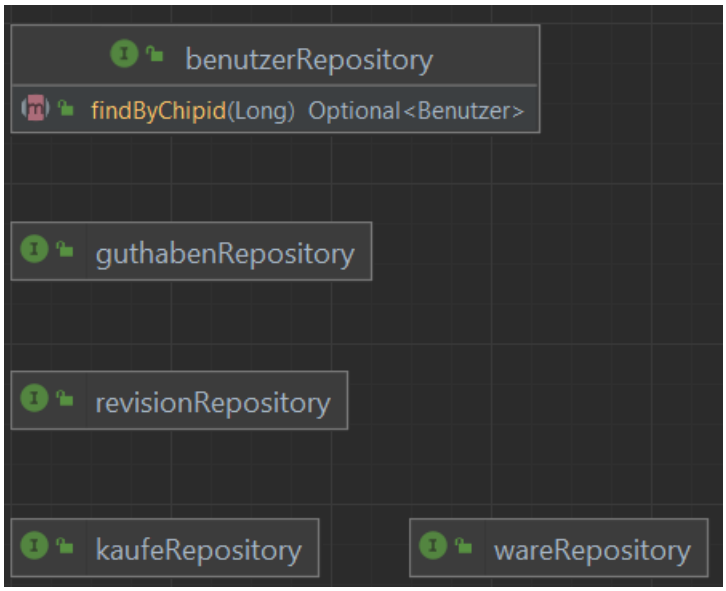

Abbildung 4 Klassendiagramm Repository

Für das Interface „benutzerRepository“ wurde die Methode „findByChipid“ zusätzlich implementiert, dass Abfragen eines Benutzers anhand der „ChipID“ ermöglicht.

11.7 Rest Controller Klassen

11.7.1 Definition

Alle definierten http-Request Methoden werden in den Controller Klassen definiert. Jeder in der Tabelle definierte Zeile ist eine eigenständige Methode der Klassen.

Da beim Aufruf der http-Request die hinterlegte Methode ausgeführt wird, gibt es als Rückgabe Operator ein Objekt vom Typ „ResponseEntity“ zurück. Eine „ResponseEntity“ ist eine Klasse vom Framework Spring die eine komplette „HTTP“ Antwort darstellt. Diese beinhaltet HTTP-Status Code und HTTP-Body (Literaturverzeichnis Lfd. Nr. 6). Der HTTP-Body besteht aus den jeweiligen Rückgaben der einzelnen Methoden als JSON Strings. Der HTTP-Status Code gibt Antwort, ob die Anfrage an das Backend erfolgreich war oder ob es innerhalb des Methodenaufrufes zu Fehlern kam.

Jede Controllerklasse beinhaltet Objektereferenzen auf die definierten „Repositories“. Dadurch wird es innerhalb der Methode möglich, Änderungen an von Werten an die Datenbank zu senden.

11.7.2 Angewendete Klassen / Besonderheiten

Folgende Klassen wurden angelegt und mit den beschriebenen Methoden definiert.

benutzerController:

http-Methode	Methoden-Name	URL	Definition
PUT	changeUser()	/benutzer/change	Änderung von Benutzereigenschaften
POST	addBenutzer()	/benutzer/add	Anlegen neuer Benutzer
PATCH	addGuthaben()	/benutzer/guthaben/add	Benutzerguthaben addieren
PATCH	subGuthaben()	/benutzer/guthaben/sub	Benutzerguthaben subtrahieren
GET	get()	/benutzer/get	Rückgabe aller Benutzer
GET	getBenutzer()	/benutzer/get/id	Rückgabe Benutzer anhand des Primary Keys
GET	getBenutzerby ChipID()	/benutzer/get/chipid	Rückgabe des Benutzers anhand der CHIPID
DELETE	deleteBenutzer	/benutzer/del/id	Benutzer löschen

Tabelle 7 benutzerController

Die Klasse „benutzerController“ implementiert eine Objektreferenz auf das Interface „BenutzerRepository“. Dadurch ist es möglich innerhalb der definierten Methoden Benutzerobjekte in die Datenbank zu speichern / ändern und Informationen von Benutzerwerten zu erhalten.

wareController:

http-Methode	Methoden-Name	URL	Definition
POST	addWare()	/ware/add	Anlegen neuer Waren
PATCH	subAnzahl()	/ware/anzahl/sub	Anzahl der Ware Subtrahieren
PATCH	addAnzahl()	/ware/anzahl/add	Anzahl der Ware Addieren
GET	get()	/ware/get	Rückgabe aller Waren
GET	getWare()	/ware/get/id	Rückgabe der Ware anhand des Primary Keys
DELETE	deleteWare()	/ware/del	Ware löschen

Tabelle 8 wareController

Die Klasse „wareController“ implementiert eine Objektreferenz auf das Interface „wareRepository“. Dadurch ist es möglich innerhalb der definierten Methoden Warenobjekte in die Datenbank zu speichern / ändern und Informationen den Waren zu erhalten.

kaufeController:

http-Methode	Methoden-Name	URL	Definition
POST	add()	/kaeufe/add	Kauf erstellen
GET	Get()	/kaeufe/get	Rückgabe aller Käufe

Tabelle 9 kaufeController

Die Klasse „kaufeController“ implementieren die Objektreferenzen von den „Benutzer- und WarenRepositories“. Diese werden benötigt, da bei der Durchführung eines Kaufes immer auf ein Objekt vom Benutzer (anhand des Primary Keys des Benutzers) und eines Warenobjektes (anhand des Primary Keys der Ware) referenziert wird. Ein Kauf wird durch den Methodenaufruf „add“ durchgeführt und dem jeweiligen Benutzer wird das Guthaben und die Anzahl des Warenobjektes um 1 verringert.

guthabenController:

http-Methode	Methoden-Name	URL	Definition
POST	denyAntrag()	/guthaben/antrag/deny	Guthabenantrag Ablehnen
POST	addAntrag()	/guthaben/antrag/add	Anlegen Guthabenantrag
POST	acceptAntrag()	/guthaben/antrag/accept	Guthabenantrag Genehmigen
GET	getRevision()	/guthaben/revision/get	Rückgabe der erledigten Guthabenanträge
GET	Get()	/guthaben/get	Rückgabe der offenen Guthabenanträge

Tabelle 10 guthabenController

Für die Erstellung von Guthabenanträgen gibt es die Methode „addAntrag()“. Sie implementiert das genutzte Benutzerobjekt und übergibt den Wert (Geldbetrag) der beantragt wird. Anschließend kann über die Methode „acceptAntrag()“ und „denyAntrag()“ der erstellte Guthabenantrag genehmigt oder abgelehnt werden. Bei Genehmigung wird das Guthaben auf das Aktuelle addiert. Zusätzlich wird eine Objektreferenz des Interfaces „GuthabenRevision“ mit hinterlegt. Um die detaillierte Ansicht der aktuellen Anträge und der bearbeiteten zu sehen, stehen die Methodenaufrufe „Get()“ und „getRevision()“ zur Verfügung.

11.7.3 Ablauf backend

Das folgende Sequenzdiagramm zeigt den Ablauf einer Methode innerhalb des „BenutzerControllers“ auf.

Der Actor stellt den http-Request von Extern dar.

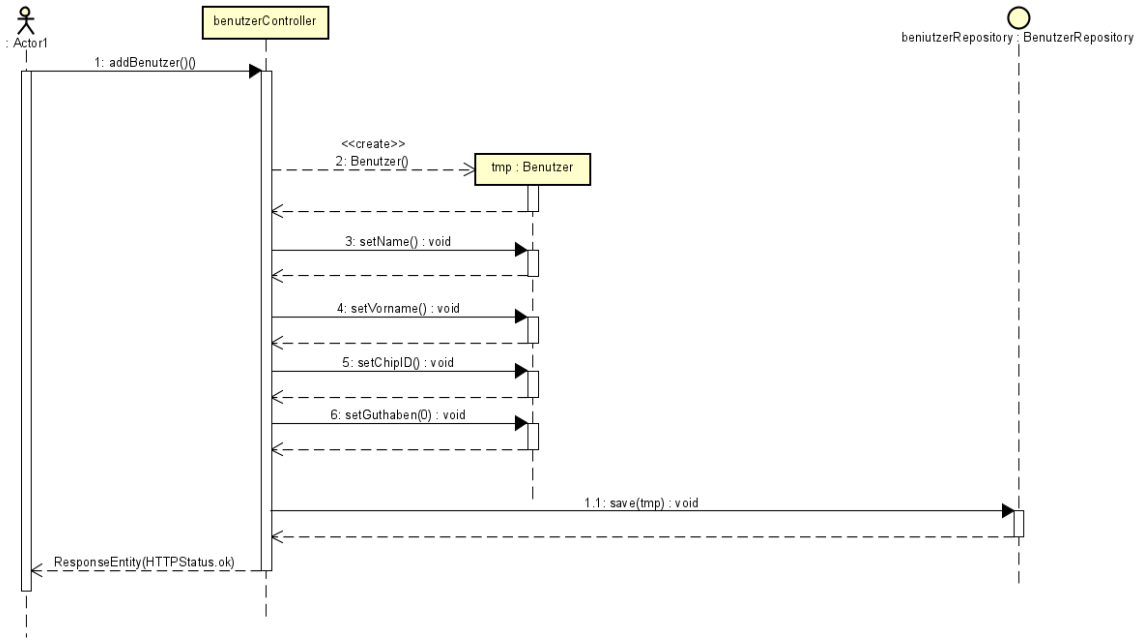

Abbildung 5 Sequenz Diagramm Ablauf

11.8 Definition Rest-API

Das beschrieben Vorgehen und umgesetzte Projekt ist der allgemeine Begriff für einen Rest-API Controller. Dieser ermöglicht den Informationsaustausch von Daten anhand von JSON Strings über die „http-Requests“. Die Unterstützen Methoden für den Datenaustausch sind „GET“, „POST“, „PUT“ und „DELETE“. Bei einem Request ist die Angabe der jeweiligen Methode zwingend mit anzugeben.

Das Backend ermöglicht es mit den Anfragen zu arbeiten und die jeweiligen Informationen zurückzugeben.

14. Technische Umsetzung Frontend

Für die Technische Umsetzung des Frontend war zuerst die Frage, welche Frameworks und technischen Spezifikationen eingesetzt werden. Die Einbindung von nativem HTML und JavaScript Code hat sich schnell als nicht mehr modern herausgestellt. Es gibt im 21. Jahrhundert modernere Frameworks und Libraries, die performanter sind und einen einfacheren Aufbau in der Entwicklung darstellen.

Die Entscheidung hier ging schnell in die Richtung, das „React“ Framework einzusetzen. React bietet eine einfache Entwicklungskomplexität und ist für den Einsatz der Technikerarbeit enorm flexibel und skalierbar. Weiterer Vorteil von React ist die Erstellung von Komponenten. Ein „component“ ist eine JavaScript Klasse, welche die Gesamte Logik zur Darstellung eines Bausteines vorhält. Diese components werden einmal erstellt und können mehrfach innerhalb des Projektes genutzt werden. Das spart deutlich Programmieraufwand, und erhöht die Flexibilität. Des Weiteren verfügt es über die sogenannte „JSX“ Syntax Erweiterung. Diese repräsentiert innerhalb einer JavaScript Klasse, den Code von HTML und CSS. Dieser kann beim Aufruf eines components von JSX auf der Website dargestellt werden.

12.1 NPM und Node.js

Für die Erstellung des React Projektes „Frontend“, der Entwicklung und das spätere Deployment auf dem Webserver, kommt der Packetmanager NPM für die JavaScript-Laufzeitumgebung Node.js zum Einsatz (Literaturverzeichnis Lfd. Nr. 7).

NPM wurde bei der Erstellung eines Standard React Projektes genutzt. Dies erstellt alle Grundkonfigurationen und deren Abhängigkeiten für die weitere Verwendung des Projektes. Zusätzlich ermöglicht NPM, alle erweiterten Pakete/Frameworks/Module bereitzustellen und zu implementieren.

Node.js ist die plattformübergreifende JavaScript Laufzeitumgebung. Diese ermöglicht es JavaScript-Code in einem Webbrowser darzustellen und kommt als Webservice zum Einsatz. Die Entwickler von Node.JS legten enormen Fokus auf die Performance des Projektes. Dies bietet den Vorteil, dass die Ausführung von Quellcode performant abgebildet wird (Literaturverzeichnis Lfd. Nr. 8).

12.2 Softwareversionen und Abhängigkeiten

Folgende Versionen wurden für die Entwicklung und Veröffentlichung:

NPM Version: 8.3.0

React / React-DOM: 17.0.2

React-Bootstrap: 2.1.0

12.3 Datentransfer / Verbindung

Der komplette Datenaustausch erfolgt über das „Backend“ mit sogenannten „http-Requests“. React bringt von Haus aus die Methode „Fetch“ mit, die uns die erforderlichen Verbindungen ermöglicht.

Prinzipiell arbeitet das Frontend als http-Client und das Backend als http-Server. Das Frontend schickt bei gewissen Aktionen die definierten Requests über „Fetch“ an das „Backend“ heraus.

12.4 Programmierung

12.4.1 Projekt Initialisierung und Struktur

Die Erstellung des React Projektes erfolgt über den NPM Dienst. Er erstellt ein Standard React Projekt mit den dazugehörigen Abhängigkeiten. Das Standardprojekt ist der Grundaufbau des „Frontend“.

Bei der Erstellung wird folgender Befehl ausgeführt:

```
npx create-react-app frontend
```

Abbildung 6 React Projekt Erstellung

Die Ordner Struktur des Projektes wurde folgend ausgewählt:

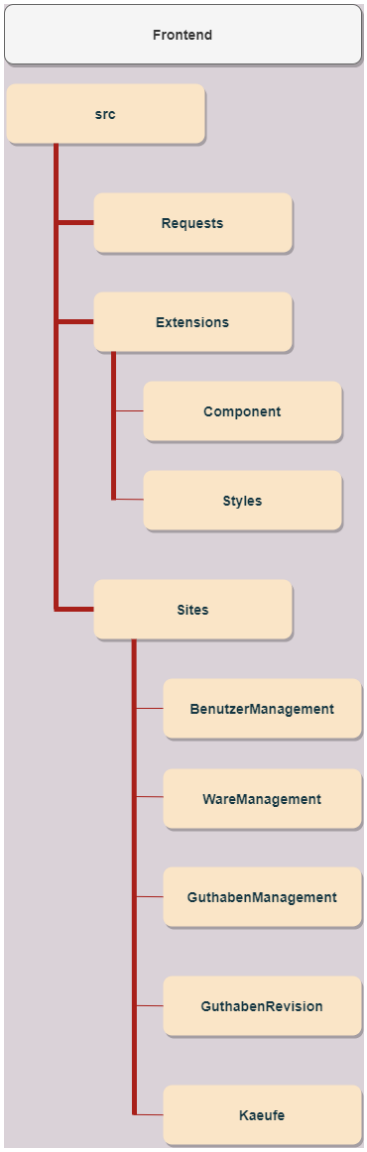

Abbildung 7 Ordnerstruktur Frontend

12.4.2 Funktionen der Klassen

Die Funktionen der Klassen werden anhand der dokumentierten Ordnerstrukturen beschrieben.

Der generelle Aufbau einer JavaScript/React Klasse/Component ist folgend:

```
import React, {Component} from "react";

class Test extends Component{
    constructor(props) {
        super(props);
        this.state = {
            beispiel: null
        }
    }

    render() {
        return(
            <div>
                Test
            </div>
        )
    }
}

export default Test
```

Abbildung 8 React Komponent

Im ersten Bereich werden die React spezifischen „Components“ importiert. Anschließend wird die Klasse „Test“ definiert, die von „Components“ erbt. Dadurch werden die JavaScript Klassen auch als React Component bezeichnet.

Des Weiteren wird der „Constructor“ und der Übergabetyp „Props“ definiert. Zusätzlich werden definierte Zustandsobjekte innerhalb des „States“ beschrieben. In dieser Klasse ist es das Zustandsobjekt „Beispiel“. Ein „State“ kann sich im Laufe der Zeit ändern und wird in diesem Zuge neu gerendert. Ein Status kann aufgrund von Netzwerkänderungen oder Benutzeraktionen geändert werden (Literaturverzeichnis Lfd. Nr. 9).

„Props“ unterstützt „States“, in andere React Komponenten zu übergeben.

Die „Render“ Methode nimmt die Eingabedaten entgegen und definiert was auf der Website angezeigt wird. Diese beinhaltet die Syntaxerweiterung für JavaScript, „JSX“.

12.4.2.1 Requests

Innerhalb des Ordners Requests sind statische Funktionen für den Aufruf des Backend erstellt. Diese ermöglicht es Änderungen im Backend zu Speichern. Aufgeteilt ist dies in Benutzer / Guthaben / Ware / Kaeufe und zusätzlich in ein Component zur Überprüfung der Verbindung zum Backend.

Folgende Methoden wurden innerhalb der JavaScript Dateien erstellt:

RequestBenutzer.js	RequestGuthaben.js	RequestWaren.js	RequestKaeufe.js
BenutzerGET()	GuthabenGet()	WareGet()	KaeufeGet()
BenutzerAdd()	GuthabenRevisionGet()	WareAnzahl()	
BenutzerChange()	GuthabenAntragAccept()	WareCreate()	
BenutzerGuthaben()	GuthabenAntragDeny()	WareDelete()	
BenutzerLoeschen()			

Tabelle 11 Requests Frontend

Alle beschriebenen Methoden werden asynchron angesprochen. Die Asynchronität schützt unsere Applikation davor, dass eine Anfrage unsere Applikation vor weiteren Aufgaben blockiert. Asynchrone Methoden laufen in diesem Falle neben dem Hauptprogramm. Zur weiteren Sicherung sind alle „Fetch“ (http_Requests) aufrufe mit dem „Try“ und „Catch“ Prinzip gesichert. Bei allen Fehlern erhalten wir „NULL“ Werte zurück.

12.4.2.2 Extensions

Innerhalb des „Extensions“ Ordner wird unterteilt in „Component“ und „Styles“

Innerhalb des Ordners „Style“ befinden sich alle CSS-Dateien. CSS ist eine Sprache, die den Stylesheet der Website definiert. Diese werden innerhalb der Render Methoden der React Komponenten mit dem React-Deklarationsattribut „className“ zugewiesen.

Des Weiteren wurden die globalen React-Components im Ordner Component definiert. Folgende wurden erstellt:

- CheckBackendConnection:

  Regelmäßige Prüfung der Verbindung zum Backend. Bei Fehlerhaften Verbindungen wird dies auf dem Dashboard visualisiert.

- Clock:

  Darstellung einer durchlaufenden Uhr.

- SiteHead:

  Der Kopf aller Websites. Dieser beinhaltet ein Zurück „Button“ und den Component „Clock“.

Die Extensions sind generell nur Komponenten und Stylesheets, die das „Frontend“ erweitern und die Nutzung vereinfachen. Für die generelle Funktion ist der Bereich aber nicht notwendig.

12.4.2.3 Sites

Die Hauptlogik des Frontend befindet sich generalisiert in den „Sites“. Eine Aufteilung erfolgt hier funktionell und aufgabengetrennt. Jede Funktion ist in eigenständigen Components abgebildet.

Die Hauptsite ist hier das Dashboard. Dieses stellt alle Verlinkungen zu den SUB-Websites dar. Über Link Aufrufe kann in die folgenden Sites gesprungen werden. Zusätzlich ist die Überprüfung der „Backendconnection“ implementiert. Diese werden mithilfe des React-Routers aufgerufen und veröffentlicht.

Die folgenden Sites erwähnen öfters einzelne Aufrufe der Methoden für den http-Requests. Diese Methodenaufrufe werden innerhalb der einzelnen React-Components importiert und aufgerufen.

Die definierten Buttons und Textfelder werden mithilfe des CSS-Frameworks Bootstrap zeitgemäßer dargestellt.

Folgende Aufteilung wurde gewählt (siehe 12.3 Projekt Initialisierung und Struktur):

Benutzermanagement:

Für die Benutzerverwaltung wurden drei React-Components erstellt. Das Component „BenutzerList“ beinhaltet eine Übersicht aller Benutzer in einer tabellarischen Ansicht. Die Benutzerobjekte werden über den Request von „BenutzerGET()“ innerhalb des Component geladen und dargestellt.

Des Weiteren wird für die Veränderung von Benutzerwerten, das React-Component „BenutzerItemChange“ hinzugefügt. Es ermöglicht die Änderung von Benutzerwerten innerhalb von Textfeldern. Mit dem Abschluss der Speicherung werden die veränderten Werten dem Backend mittels der „BenutzerChange()“ Methode übergeben. Zusätzlich lassen sich die Guthaben einzelner Benutzer mittels der „BenutzerGuthaben()“ Methode im backend ändern und per „BenutzerLoeschen()“ auch einzelne Benutzer aus der Datenbank entfernen.

Für das Erstellen von neuen Benutzern wird das dritte React-Component „BenutzerADD“ genutzt. Durch die Eingabe der Parameter „Name“, „Vorname“ und „CHIPID“ wird es nach Bestätigung im Backend erstellt.

Warenmanagement:

Für die Bearbeitung der Waren/Artikel wurden wie beim Vorgänger ebenfalls 3 React-Componenten erstellt. Das Component „WareList“ stellt tabellarisch alle Waren aus dem backend heraus dar. Die Objekte werden über den Methodenaufruf „WareGet()“ innerhalb des Component geladen und dargestellt.

Für die Bearbeitung der Werte eines Warenobjektes wurde das Component „WareChange“ erstellt und implementiert. Folgende Möglichkeiten können durchgeführt werden mit dazugehörigem http-Request:

- Änderung der Anzahl -> WareAnzahl()
- Löschen einer Ware -> WareDelete()

Für die Erstellung neuer Waren wird das Component „WareADD“ genutzt. Durch die Eingabe der Parameter „Name“ und Anzahl“ wird es nach Bestätigung im Backend erstellt.

Die Änderung des Namens wurde nicht implementiert, da hierfür ein neues Objekt generiert werden muss.

GuthabenManagement:

Für die Auflistung aller Guthaben Anträge wurde das Component „GuthabenAuftragList“ erstellt. Es listet alle Anträge tabellarisch dar, indem es die Information aus dem Backend über die Methode „GuthabenGet()“ erhält. Wenn aktuell keine Anträge vorliegen, wird ein Infotext dargestellt.

Des Weiteren wurde das Component „GuthabenAuftragRequest“ erstellt und implementiert. Dieses fügt nach der Tabellarischer Auflistung ein Textfeld mit den Buttons „Ablehnen“ und „Genehmigen“ hinzu. Durch die Eingabe der AuftragsID in das Textfeld kann über die Buttons der Prozess durchgeführt werden. Das Betätigen der jeweiligen Buttons führt im Hintergrund die jeweilige Methoden „GuthabenAntragAccept()“ oder „GuthabenAntragDeny()“ aus.

GuthabenRevison:

Die Darstellung der Guthabenrevision, wird durch das React-Component „GuthabenRevision“ tabellarisch dargestellt. Über den Methodenaufruf „GuthabenRevisionGet()“ werden die Objekte geladen und an das Component übergeben. Die Site stellt lediglich nur Objekte aus dem Backend dar.

Kaeufe:

Die Darstellung aller Käufe, wird durch das React-Component „KaeufeList“ tabellarisch dargestellt. Über den Methodenaufruf „KaeufeGet()“ innerhalb des Component, werden die Objekte geladen und dargestellt.

Abschließend zu dem Funktionsüberblick ist zu sagen, dass Funktionen der Anwendung in einzelne React-Components aufgeteilt sind. Diese Components rufen nur die programmierten Requests auf, die für das Speichern im Backend verantwortlich sind. Innerhalb der Components sind aber Überprüfungen vorhanden, die fehlerhafte Übermittlungen ausschließen.

12.5 Problemstellung

Da beim Aufruf des Frontend im Hintergrund Ressourcen einer anderen Domäne skriptseitig bezogen werden, wird dies standartgemäß durch die Same Origin Policy (SOP) untersagt. SOP ist eine Funktion, die das Laden von Ressourcen nur auf der eigenen Quelle erlaubt. Dies Technologie ist in den meisten modernen Browser implementiert (Literaturverzeichnis Lfd. Nr. 10).

Dieses Sicherheitsfeature führte zu Fehlerhaften http-Requests. Die Lösung dahinter ist die Einrichtung von CORS auf dem Backend.

Cross-Origin Resource Sharing (CORS) ermöglicht es, die Beschränkungen durch SOP entstanden sind, zu umgehen. Hierzu werden sogenannte CORS-Header im Backend definiert. Diese geben an von welcher Adresse die Anfragen genehmigt werden und welche http-Methoden erlaubt sind (Literaturverzeichnis Lfd. Nr. 10).

Das Backend wurde mit einer Methode erweitert, welche die erlaubten Zugriffe auf alle(„*“) genehmigt. Des Weiteren wurden die http-Request Methoden „GET“, „PUT“, „PATCH“, „POST“, „DELETE“, und „Options“ genehmigt. Durch die Implementierung der CORS Richtlinien im Backend ist der Aufruf auf dem Frontend ohne Fehler möglich.

Im Pflichtenheft wurde der Einsatz von „TypeScript“ mit React beschrieben und auch als eingesetzte Software definiert. „TypeScript“ bietet eigentlich den Vorteil die Applikation/Website Typen sicher zu programmieren. Da die definierten Typen/Klassen schon aus dem Backend kommen und das Frontend mit den JSON-Objekten arbeitet, war die Nutzung von „TypeScript“ nicht mehr notwendig.

Zusätzlich wurde der Login Bereich über die sogenannte „Basic-Authentication“ realisiert. Der Aufwand für die Implementierung eines eigenständigen Login Bereiches konnte zeitlich nicht umgesetzt werden.

13. Rollout Backend/Frontend

Für die Bereitstellung der beiden Services wurde ein vServer bei der Firma Hetzner gebucht. Dieser verfügt über folgende Ressourcen

CPU: 2 vCPU

Arbeitsspeicher: 4 GB

Speicher: 50 GB

Ipv4: 1x Öffentliche Adresse

Betriebssystem: Debian 10.11

Hostname: Backend

Der Server ist mit der Container-Virtualisierung Docker ausgerüstet. Docker ermöglicht es die jeweiligen Applikationen aus dem Backend und Frontend in eigenständige Container zu generieren.

Container sind eigenständige und abgekapselte Einheiten, in denen die Applikation gestartet werden können. Vorerst mussten aber Docker-Images erstellt werden, die mit den Werten der programmierten Applikation gefüllt werden.

Hierfür wurden innerhalb der Projekte von Backend und Frontend ein Docker File erstellt(Projekt: „Dockerfile“). Dieses beinhaltet notwendige Parameter, die für den Start der Applikation innerhalb des Containers notwendig sind.

Anschließend konnten die Projekte auf den vServer verschoben und die Images daraus generiert werden.

Mit der Multi-Container Applikation „Docker-Compose“ wurde ein Skript erstellt, das genau definiert, welche Eigenschaften die jeweiligen Container erhalten. Dies ist für eine reibungslose Zusammenarbeit der Applikation notwendig.

„Docker-Compose“ ermöglicht zusätzlich das zentrale Starten und Stoppen der Container.

Details zum Erstellen der Images sind im FAQ genau definiert.

13.1 Container Backend

Für das Backend wurden folgende Container erstellt:

Name	opt_db_1
Applikation	MySQL
Freigegeben Port	3000

Tabelle 12 Datenbank Container

Name	opt_springproxy_1
Applikation	Backend Image
Freigegeben Port	8080

Tabelle 13 Backend Container

Diese zwei Container stellen das Backend für das Digitale Kassensystem im Internet zur Verfügung. Der Start des Backend ist nur möglich, wenn davor der Container „opt_db_1“ gestartet wurde. Eine Absicherung durch einen Reverse Proxy wurde in diesem Projekt nicht durchgeführt.

13.2 Container Frontend

Für das Frontend wurden folgende Container erstellt:

Name	opt_frontend_1
Applikation	Frontend Image
Freigegeben Port	3306

Tabelle 14 Frontend Container

Name	Production_ngingx
Applikation	NGINX Reverse Proxy
Freigegeben Port	80 , 443

Tabelle 15 Reverse Proxy Container

Die zwei Container stellen das Frontend für das Digitale Kassensystem dar. Für den Aufruf aus dem Internet wurde dem „Frontend“ ein Reverse Proxy von NGINX bereitgestellt. Dieser ermöglicht eine Basic-Authentifizierung, sodass die Website vom Frontend nur mit einem User und Password aufgerufen werden kann.

Der Container „Production_nginx“ hat zusätzlich eine Verlinkung auf einem definierten Ordner des vServers. Dadurch kann die Konfigurationsdatei des NGINX außerhalb des Containers bearbeitet werden.

Folgende Konfigurationsdatei wurde dem Container „Production_nginx“ übergeben:

```
http {
server {
    server_name backen.werner-intern.cloud;
    listen 80;
    auth_basic "Administrator's Area";
    auth_basic_user_file /etc/nginx/conf.d/.htpasswd;

    location / {
      proxy_pass http://opt_frontend_1:3000/;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
      proxy_set_header X-Forwarded-Port $server_port;
      proxy_set_header Host      $host:$server_port;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      }
   }
}
```

Abbildung 9 NGINX Konfiguration

Diese Konfigurationsdatei definiert die Weiterleitung zu dem Container „opt_frontend_1“. Des Weiteren wurde das Benutzerfile für die Basic-Authentifizierung mitgegeben. Die Datei „.htpasswd“ beinhaltet den Benutzer und das dazugehörige Passwort als Hash-Wert.

13.3 Technischer Überblick

Das folgende Bild zeigt die Container innerhalb des vServers und die dazugehörigen Verbindungen.

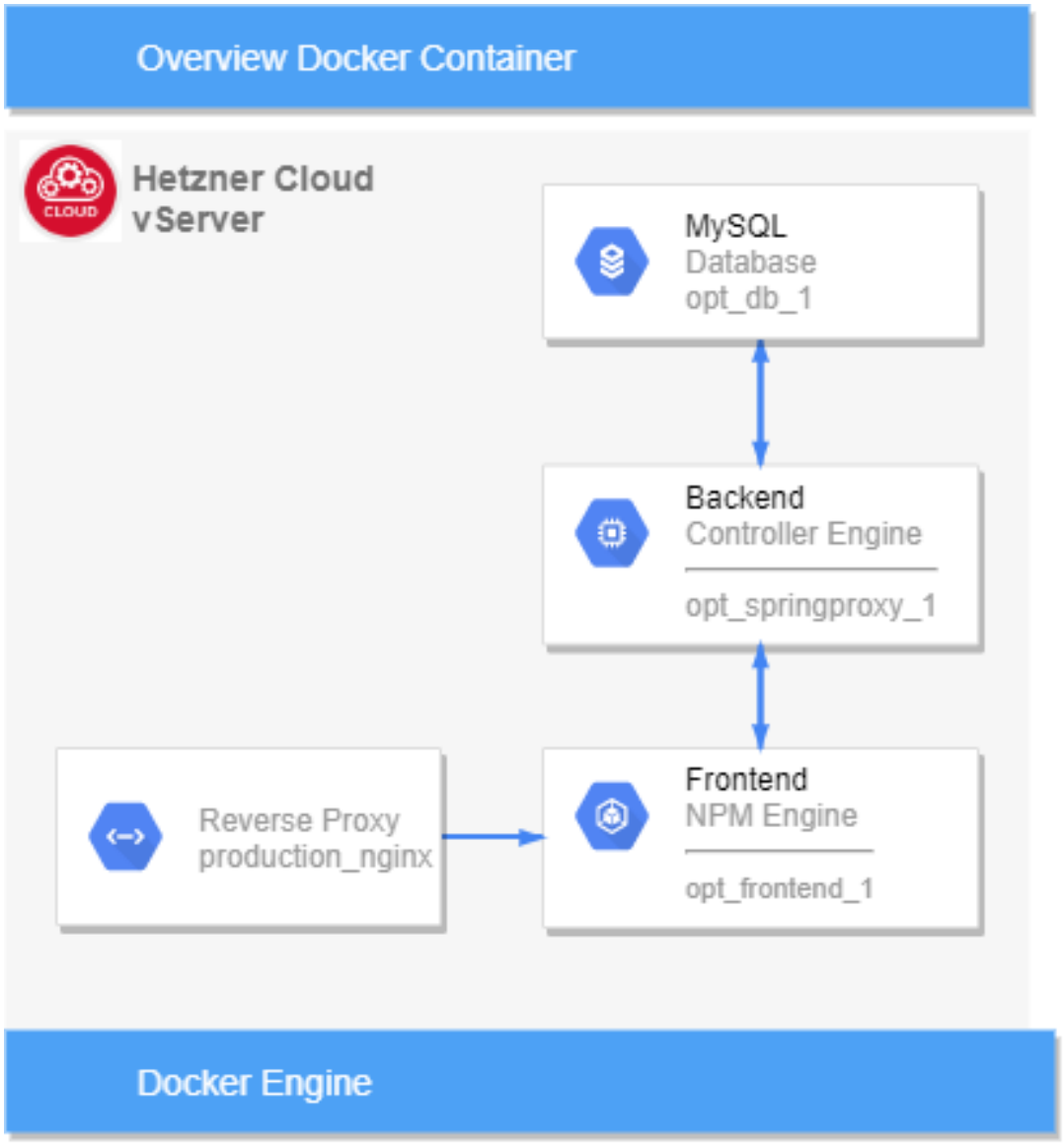

Abbildung 10 **Technischer Überblick Backend/Frontend**

Der Server erhielt die Domain „backend.werner-intern.cloud“. Über diese ist der Server aus dem Internet erreichbar. Die Domain ist beim Provider INWX registriert und wurde dort auch käuflich erworben.

14. TERMINAL

Das Terminal muss für die Benutzer eine einfache Handhabung haben und der Anschluss für einen RFID-Reader erfüllen. Des Weiteren musste eine graphische Oberfläche für eine standardisierte Bedienung vorhanden sein.

Aufgrund der geringen Kosten und der einfachen Implementierung, ist hier die Entscheidung auf eine Raspberry Pi 4 gefallen. Die Raspberry bietet den Anschluss von Externen HDMI Touch Panel und die Integration digitaler Kontaktstifte für die Nutzung integrierter Schaltkreise (GPIOs). (Literaturverzeichnis Lfd. Nr. 11)

Die Auswahl des Betriebssystems fiel dadurch auf das Debian-basierte „Raspbian OS“. Das zusätzlich eine graphische Oberfläche mitliefert.

Das Auslesen der RFID-Chips wurde mithilfe des MFRC522 RFID Chip-Reader realisiert. Dieser wird durch die GPIOs der Raspberry Pi und dem SPI Bus System angesprochen.

Für die Programmierung der Terminal Software in Abhängigkeit der beschriebenen Komponenten, musste eine einheitliche Software definiert werden, die unterschiedliche Frameworks unterstützt. Da das System auf Linux basiert, war schnell der Einsatz von Python3 klar. Diese unterstützt die benötigten Frameworks für das Auslesen der RFID-Chips und das Erstellen der graphischen Oberfläche.

Folgende Produkte werden für das Terminal ausgewählt:

Hardware:

Raspberry Pi 4 Model B

- 2 GB Ram
- 32 GB SanDisk Extreme micro SDHC A1 UHS-I U3 Speicherkarte
- Raspberry Pi USB-C Netzteil 5,1V / 3,0A, EU schwarz
- Micro HDMI Adapterkabel D-Stecker - A-Buchse 15cm schwarz

7 Inch IPS Touch Screen für Raspberry Pi

- Native Auflösung: 1024x600

OPEN-SMART RC522 RFID Kartenleser

- Betriebs Frequenz: 13,56 MHz

Software:

- Raspbian OS
- Python3
- QT

14.1 Aufbau der Komponenten

Folgend der Überblick über den Aufbau der Komponenten.

14.1.1 Aufbau Touch Display

Die Verbindung des Touchdisplays zur Raspberry PI erfolgt durch HDMI. Da die Raspberry ein Micro HDMI Anschluss hat, wird ein Adapter auf normales HDMI verwendet. Zusätzlich benötigt das Touch-Display noch zwei USB-Anschlüsse an der Raspberry, jeweils für den Strom und die Touch-Funktion

14.1.2 Aufbau RFID-Reader

Die Anbindung des RFID Reader an die Raspberry PI4 erfolgt über die GPIOs an der Raspberry und den jeweiligen PINS an dem RFID-Reader.

Die Grafik und die Tabelle zeigen den Aufbau der Verbindung der beiden Komponenten.

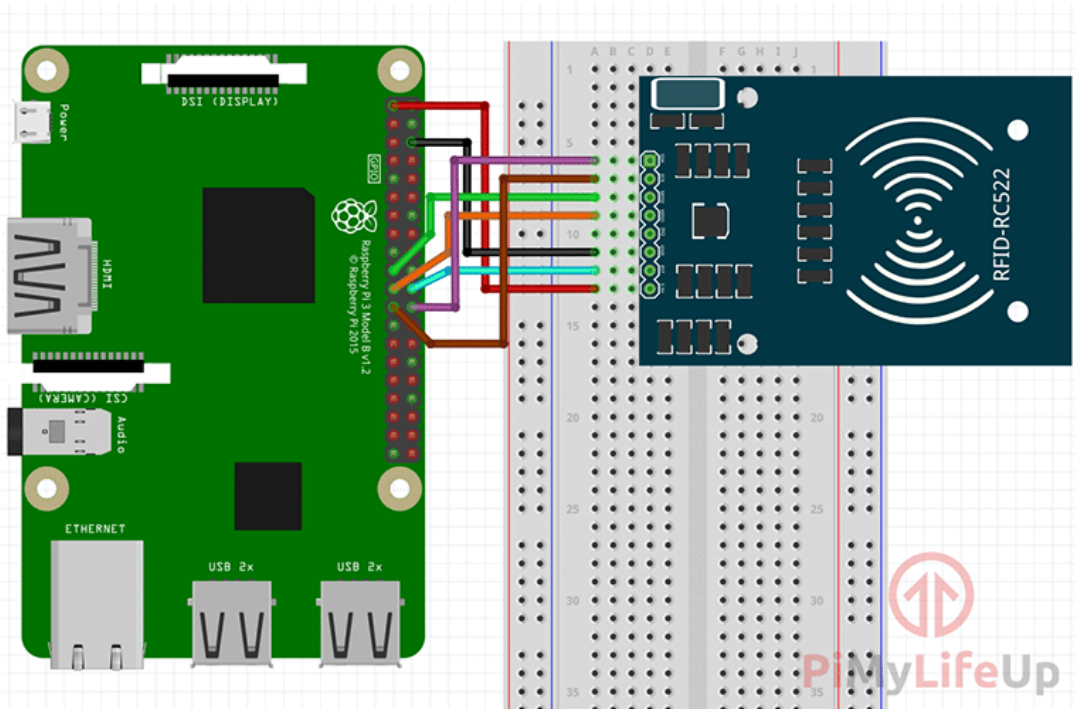

Abbildung 11 RFID Reader Aufbau

Bild von der Website: URL Verzeichnis Lfd. Nr. 2

Tabellarisch dargestellt:

Raspberry GPIO	RFID - PIN
3,3 V	3,3 V
Ground	GND
GPIO 10	Mosi
GPIO 09	Miso
GPIO 11	SCK
GPIO 25	RST
GPIO 8	SDA

Tabelle 16 GPIO Anschluss

14.2 SPI Schnittstelle

Die Kommunikation zwischen den beiden Komponenten erfolgt über die SPI-Schnittstelle das bedeutet Serial Peripheral Interface und ist ein BUS-System mit einer synchronen Übertragung. Das BUS-System wurde im Jahr 1987 von der Entwicklerin Susan C. Hill und weiteren Halbleiterherstellern entwickelt.

Die SPI Schnittstelle kommt dann zum Einsatz, wenn unterschiedliche Mikrocontroller wie Speicher, Sensoren oder USB-Controller usw. miteinander kommunizieren müssen. Das Prinzip für den Verbindungsaufbau erfolgt über den Master/Slave Prinzip. Wobei hier der Raspberry Pi 4 der Master (Taktgeber) ist und der Reader den Slave Status annimmt (Literaturverzeichnis Lfd. Nr. 12). Über die genutzten MOSI und MISO GPIO Pins, können dann die geforderten Daten vom Reader gesendet/empfangen werden.

14.3 Python3 mit QT

Für die Erstellung der graphischen Oberfläche des Terminals wurde mit dem Anwendungsframework Qt und dem „QtCreator“ gearbeitet. Mit dem „QtCreator“ können graphische Oberflächen gestaltet und anschließend in C++ Code ausgegeben werden.

Für die Darstellung/Programmierung der graphischen Oberfläche in Python3 wird die Bibliothek PyQT5 verwendet. Diese ist kompatibel mit dem Anwendungsframework „Qt“. Da der Erstellte C++ Code nicht mit Python3 Anwendungen direkt nutzbar ist, muss dieser erst in Python3-Format überführt werden. Hierfür kommt die Bibliothek „QtPY“ zum Einsatz. Diese konvertiert den Code innerhalb von C++ in das Python3 Format mit beinhalteten Klassenbezug auf PyQT5.

Nach der Konvertierung kann innerhalb des Python3 Projektes die graphische Oberfläche implementiert werden.

14.4 Programmierung

14.4.1 Projekt Struktur

Die Projektstruktur wurde in Abhängigkeit der Tätigkeiten gewählt.

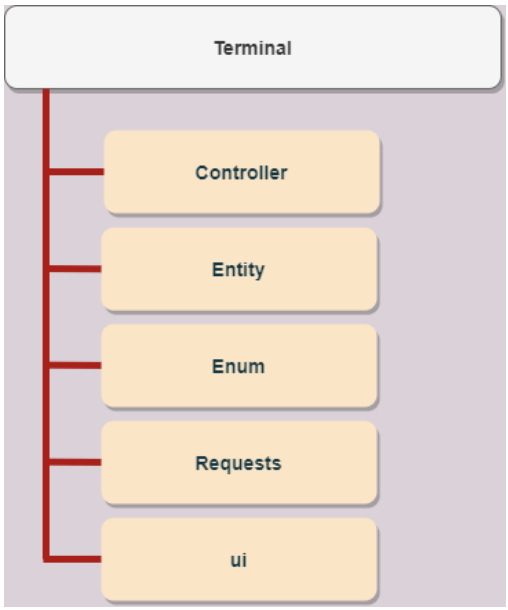

Abbildung 12 Ordnerstruktur Terminal

14.4.1.1 Requests

Der Ordner „Requests“ beinhaltet statische Methoden für den Aufruf der http-Requests zum Backend. Folgende wurden hierzu erstellt:

REQUESTFILE	BESCHREIBUNG
WAREGET()	Alle Waren Abfragen
KAEUFEADD()	Kauf durchführen
USERGETBYCHIPID()	Benutzerinformationen anhand der CHIPID abfragen
GUTHABENANTRAGADD()	Guthaben Antrag stellen

Tabelle 17 Terminal Requests

Die jeweiligen Methoden sind mit dem „Try“ und „Catch“ Prinzip gesichert. Dadurch wird bei fehlerhaften http-Requests die Anwendung nicht abstürzen, sondern zeigt Fehlerinformationen an.

Die Methoden können aber nur genutzt werden, wenn ein Benutzer Objekt anhand der CHIPID vorliegt.

14.4.1.2 Entity

Innerhalb des Ordners „Entity“ befinden sich die Klassen von Benutzer und Ware. Diese halten das Grundgerüst eines Benutzers und einer Ware. Sie verfügen über die gleichen Attribute, wie im Kapitel 11.5 Entity von Benutzer und Ware schon dargestellt.

14.4.1.2 Enum

Innerhalb des Ordners wurde eine ENUM-Klasse erstellt, die feste Werte für das Guthaben beinhaltet. Das ENUM schützt so das Projekt, damit innerhalb immer mit gleichen Werten gearbeitet wird.

Folgende Werte sind definiert:

- Fünf
- Zehn
- Zwanzig
- Fünfzig

14.4.1.3 Controller

Für die Durchführung der http-Requests wurden die Controller Klassen implementiert. Diese überprüfen, ob die übergebenen Parameter nicht „Null“ oder leer sind, sodass Fehlerhafte abfragen schon innerhalb des Terminals geprüft werden. Zusätzlich wird geprüft, ob die Rückgabe der http-Requests Fehler Codes enthalten. Dadurch werden Fehler oder Exceptions abgefangen, ohne dass sie die Anwendung zum Abstürzen bringen.

Die Aufteilung erfolgt hier in vier statische Methoden.

Die Controllerklassen geben bei der Rückgabe des Benutzers oder der Waren nie ein Objekt des JSON Strings zurück, sondern Objekt / Objektlisten von Benutzer oder Ware anhand der Entitätenklasse zurück.

14.4.1.4 UI

Innerhalb des Ordners UI sind die zwei graphischen Oberflächen des digitalen Kassensystems hinterlegt. Unterteilung ist hier jeweils das „loginwindows“ und das „kassenwindow“. Die Python Klassen für das jeweilige Fenster definieren das genaue Aussehen der jeweiligen Oberflächen und deren Positionierung der implementierten Items. Operationen oder Aufrufe werden daraus nicht durchgeführt, da sie wie im Kapitel 14.3 Python3 mit QT generiert wurden. Die Aufrufe der jeweiligen Items werden zentral aus dem Main Programm gesteuert.

14.4.1.5 Main / Hauptprogramm

Der Komplette Aufruf des Terminals, befindet sich in der „main.py“. Diese unterteilt sich innerhalb in zwei Python Klassen auf. Die sind für den Aufruf der GUI Klassen aus dem Ordner UI zuständig und bringen die Logik für das Betätigen der unterschiedlichen Items. Zusätzlich findet der Aufruf des RFID Readers innerhalb der Klasse des „LoginWindows“ statt. Für eine einfachere Übersicht folgend das Klassendiagramm:

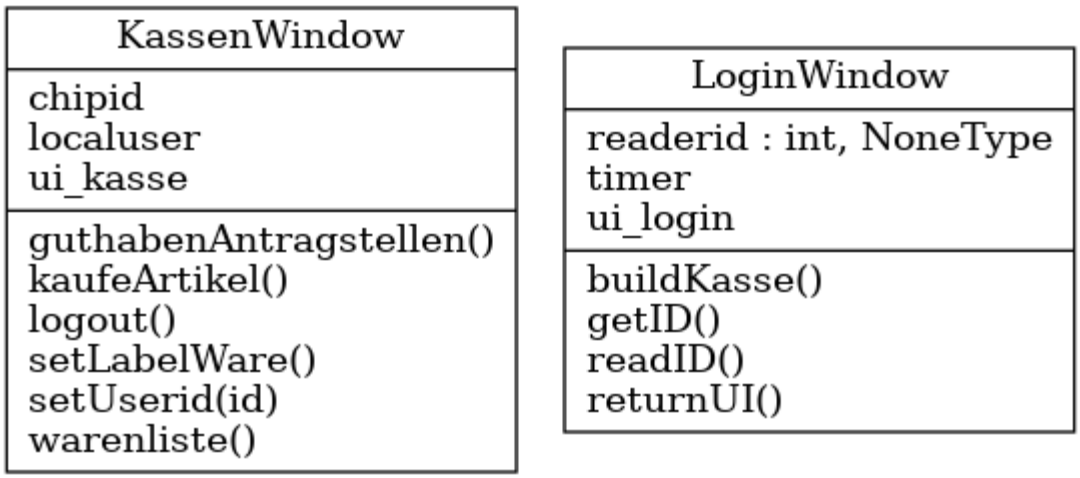

Abbildung 13 Klassendiagramm Hauptprogramm

14.5 Terminal Oberflächen

LoginWindow:

Für die Anmeldung der Benutzer wird das „Loginwindow“ als erstes gestartet. Durch das Vorhalten des RFID-Chips, wird die CHIPID angezeigt und der Benutzer kann sich anmelden.

Abbildung 14 Terminal LoginWindows

KassenWindow:

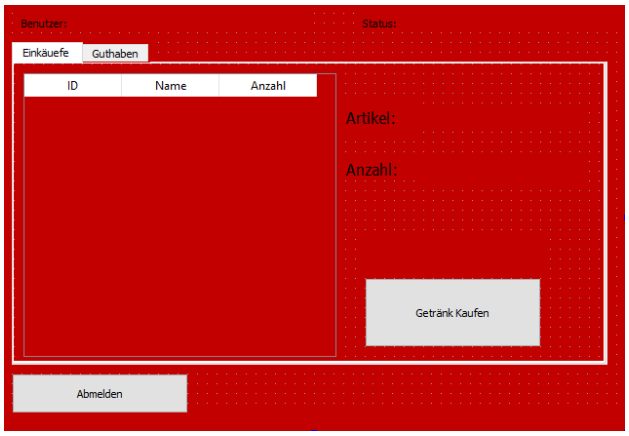

Abbildung 15 Terminal KassenWindow-1

Abbildung 16 Terminal KassenWindows-2

Die Durchführung der Transaktionen erfolgt über das „KassenWindow“. In der linken Grafik ist der Kauf von Waren/Artikel gegeben. Hier werden in der Tabelle alle Waren angezeigt und durch Auswahl einer kann diese erworben werden.

Im Zweiten Reiter „Guthaben“ ist das Aufladen des eigenen Kontostandes durch die Stellung eines Antrages möglich.

14.6 Problemstellung

Leider ist es im Rahmen der Technikerarbeit nur möglich eine Anmeldung durchzuführen. Anschließend muss die Anwendung geschlossen und neu ausgeführt werden. Vorgesehen war das Abmelden eines angemeldeten Benutzers und das erneute Anmelden anderer Benutzer, ohne die Anwendung zu schließen. Da dieser Fehler aber nicht behoben werden konnte, wurde der Workaround gewählt nach jeder Nutzung der Applikation das Programm zu beenden und manuell zu Starten.

Dadurch kann das Terminal problemlos benutzt werden.

15 FAQ

Das FAQ beschreibt einfache Handhabungsmöglichkeiten, wie mit dem System gearbeitet werden kann.

15.1 Generierung Docker-Image

Wenn die Programmierprojekte von Backend und Frontend erweitert werden, muss ein neues Image innerhalb des vServers von Docker generiert werden.

Folgende Schritte sind notwendig:

1. Das Gesamte Projektverzeichnis von Backend oder Frontend muss auf den vServer kopiert werden. Empfehlenswert ist hier das Tool „WinSCP“. Wenn das Projekt als ZIP Archive komprimiert wird, erleichtert es das Kopieren auf den vServer enorm.
2. Das Verschobene Projekt muss anschließend dekomprimiert werden. „unzip“ ermöglicht dies vollständig durchzuführen.
3. Abschließend kann das Projekt zum Docker-Image hinzugefügt werden. Im Projekt Ordner auf dem Server wird folgender Befehl:
   # docker build -t spring-boot:latest .

   Für das Frontend Projekt wird der Name des Images angepasst:
   # docker build -t frontend:latest .

15.2 Administrative Unterstützung Backend

Da im Fehlerfall vom Frontend, das Backend weiterhin noch administrativ erreichbar sein muss, wurde hierfür die Swagger-UI dem Backend hinzugefügt. Swagger-UI ist eine Applikation, die es ermöglicht API-Controller zu visualisieren und mit ihr zu interagieren.

Swagger-UI stellt hier alle erstellten Methoden die im Kapitel 11.7.2 Angewendete Klassen / Besonderheiten programmiert wurden visuell dar. Des Weiteren kann über Swagger alle http-Request betätigt werden. Dadurch können alle Werte der Datenbank bearbeitet werden.

Der Aufruf erfolgt über die gesetzte folgende URL:

http://backend.werner-intern.cloud:8080/swagger-ui.html.

16. QUALITÄTSSICHERUNG

Folgende Prüfungen wurden für die Qualitätssicherung durchgeführt:

Backend (anhand der Swagger-UI):

Durchgeführte Maßnahmen	Testergebnis
Anlegen / Bearbeiten / Löschen von Benutzer	Erfolgreich
Benutzereigenschaften Laden anhand der CHIPID	Erfolgreich
Guthaben eines Benutzers Erhöhen / Verringern	Erfolgreich
Anlegen / Löschen von Waren	Erfolgreich
Warenanzahl erhöhen / verringern	Erfolgreich
Kauf Anlegen	Erfolgreich
Einsicht aller Käufe	Erfolgreich
Guthabenanträge erstellen	Erfolgreich
Antrag Akzeptieren, Prüfung ob das Guthaben des Benutzers aktualisiert wurde	Erfolgreich
Antrag ablehnen, Prüfung ob das Guthaben des Benutzers gleich geblieben ist	Erfolgreich

Tabelle 18 Qualitätssicherung Backend

Frontend:

Durchgeführte Maßnahmen	Testergebnis
Anlegen / Bearbeiten / Löschen von Benutzer	Erfolgreich
Guthaben eines Benutzers Erhöhen / Verringern	Erfolgreich
Anlegen / Löschen von Waren	Erfolgreich
Warenanzahl erhöhen / verringern	Erfolgreich
Einsicht aller Käufe	Erfolgreich
Antrag Akzeptieren, Prüfung ob das Guthaben des Benutzers aktualisiert wurde	Erfolgreich
Antrag ablehnen, Prüfung ob das Guthaben des Benutzers gleich geblieben ist	Erfolgreich
Einsicht in die Revision der Guthabenanträge	Erfolgreich

Tabelle 19 Qualitätssicherung Frontend

Terminal:

Durchgeführte Maßnahmen	Testergebnis
Benutzerinformation durch die CHIPID erhalten	Erfolgreich
Darstellung der Warenliste	Erfolgreich
Durchführung eines Kaufes	Erfolgreich
Erstellung Guthabenantrag	Erfolgreich

Tabelle 20 Qualitätssicherung Terminal

17. Fazit

Mit dem Gesamtergebnis des Projektes bin ich sehr zufrieden. Leider musste ich in den Bereichen der Sicherheit Abstriche machen. Diese hätten aufgrund des Zeitfaktors nicht mehr in den Rahmen der Technikerarbeit gepasst.

Des Weiteren bin ich sehr erfreut sagen zu können, eine Full Stack Anwendung durchgeplant und programmiert zu haben. Für die DLRG OG Lauffen steht ab sofort ein neues Kassensystem zur Verfügung.

18. Literaturverzeichnis

Laufende Nummer	URL	Kapitel	Letzter Aufruf
1	https://de.wikipedia.org/wiki/MySQL	Überblick	08.02.2022
2	https://de.wikipedia.org/wiki/Spring_(Framework)	Überblick	08.02.2022
3	https://de.wikipedia.org/wiki/JavaScript_Object_Notation	Überblick	11.02.2022
4	https://www.opc-router.de/was-ist-json/#Zahlen-Zeichenketten	-	11.02.2022
5	https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa	3.1	13.02.2022
6	https://www.baeldung.com/spring-response-entity	-	17.02.2022
7	https://de.wikipedia.org/wiki/Npm_(Software)	-	25.02.2022
8	https://de.wikipedia.org/wiki/Node.js	-	25.02.2022
9	https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-state	-	01.03.2022
10	https://www.securai.de/veroeffentlichungen/blog/was-ist-cors-und-welche-sicherheitsauswirkungen-hat-es-auf-web-applikationen/	-	03.03.2022
11	https://de.wikipedia.org/wiki/GPIO	-	04.03.2022
12	https://www.kunbus.de/die-spi-schnittstelle.html#:~:text=SPI%20bedeutet%20Serial%20Peripheral%20Interface,einer%20Norm%20oder%20einem%20Standard.	-	07.03.2022

19. URL Verzeichnis

Laufende Nummer	URL	Letzter Aufruf
1	https://start.spring.io/	08.02.2022
2	https://pimylifeup.com/raspberry-pi-rfid-rc522/	07.03.2022

20. Tabellenverzeichnis

21. ABBILDUNGSVERZEICHNIS