EazyMenu

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:
Benjamin Besic
Bozidar Spasenovic
David Ignjatovic

Betreuer:
Prof. DDI Clemens EISSERER

Projektpartner:
Oberösterreichische Versicherung AG, Linz

Leonding, April 2022

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

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

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022 Benjamin Besic & Bozidar Spasenovic & David Ignjatovic

Zur Verbesserung der Lesbarkeit wurde in diesem Dokument auf eine geschlechtsneutrale Ausdrucksweise verzichtet. Alle verwendeten Formulierungen richten sich jedoch an alle Geschlechter.

Abstract

The Company Oberösterreichische Versicherung uses IBM Notes to process orders. As the name says, the desktop workflow application is only optimized for the desktop and unfortunately it is also outdated. The company wanted to replace the outdated app with a new web application that meets today's standards. With the help of EazyMenu, it is much easier for employees to place an order from their desktop at work or from their own mobile phone. The canteen now also has a much easier interface. EazyMenu has made it easier for the canteen to enter and manage new dishes.

Zusammenfassung

Die Firma Oberösterreichische Versicherung benützt IBM Notes um Bestellungen durchzuführen. Wie der Name schon sagt, ist die desktop workflow application nur für den Desktop optimiert und auch leider veraltet. Die Firma wollte die veraltete App ersetzten mit einer neuen Webanwendung, die den heutigen Standards entspricht. Mithilfe von EazyMenu ist es den Mitarbeiter viel einfacher eine Bestellung zu tätigen und das entweder von dem Desktop auf dem Arbeitsplatz oder auch von dem eigenen Handy. Auch die Kantine hat jetzt eine viel einfach zu Bedienende Oberfläche. Durch EazyMenu ist es nämlich der Kantine einfacher geworden, neue Gerichte einzutragen zu lassen und sie auch zu verwalten.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangsituation

Die Oberösterreichische Versicherung ist der Marktführer im Versicherungsbereich in Oberösterreich und beschäftigt in ihrer Zentrale in Linz über 500 Mitarbeiter. Das Unternehmen besitzt eine Kantine, wo täglich 3 Hauptspeisen serviert werden. Dazu noch eine Vorspeise und eine Nachspeise.

1.2 Ist-Zustand

Die derzeitige Bestellmöglichkeit funktioniert über eine simple Datenbankanwendung mit IBM Notes. Dieses Programm ist auf jedem Rechner installiert und jeder Mitarbeiter ist mit seinen Daten bereits eingeloggt. Auf einem simplen Interface kann man zwischen den heutigen und zukünftigen Mahlzeiten wählen und die Uhrzeit, wann man diese konsumiert. Nach erfolgreicher Bestellung hat man eine kleine Übersicht über die vergangenen Bestellungen.

1.3 Problemstellung

Das derzeitige Programm ist sehr veraltet und nicht besonders benutzerfreundlich. Außerdem läuft das Programm lokal auf jedem Rechner und eine mobile Bestellung ist ausgeschlossen. Außerdem ist zu erwähnen ist, dass der Benutzer nur eine beschränkte Möglichkeit hat sein Bestellverhalten zu visualisieren bzw. zu analysieren.

1.4 Aufgabenstellung

Die Aufgabenstellung der Firma ist eine Webanwendung zu entwickeln, die den Vorgänger ablöst und ein modernes und benutzerfreundliches Interface hat. Zudem soll eine Bestellung über das Smartphone möglich sein. Zusätzlich soll ein Empfehlungssytem entwickelt werden, dass einem ermöglicht Mahlzeiten zu bestellen, die auf einen

abgestimmt sind. Außerdem soll man eine Einsicht über seine Bestellhistorie haben mit diversen Statistikelementen.

1.5 Ziel(e)

- Erleichterung des Bestellprozesses für den Benutzer
- Flexible Bestellungsmöglichkeiten
- Der Benutzer hat mehr Verständnis über sein Bestellverhalten

1.5.1 Zielgruppe

Mitarbeiter der OÖ Versicherung AG

1.6 Use Cases

1.6.1 Kantinenarbeiter

- Neue Menüs anlegen
  - Ein Kantinenmitarbeiter kann für jeden Tag neue Menüs mit drei Hauptspeisen und deren Kategorien, einer Vorspeise und einer Nachspeise anlegen.
- Vorhandene Menüs editieren
  - Die Bezeichnungen der bereits erstellten Menüs sollen verändert werden können.
- Übersicht der täglichen Bestellungen
  - Die Kantinenmitarbeiter sollen eine Übersicht, der an einem bestimmten Tag bestellten Menüs haben. Diese inkludiert die zusammengefasste Bestellanzahl der verschiedenen Menüs und eine Liste von allen Bestellungen.
- Bestellungsübersicht drucken
  - Die Übersicht wie vorher beschrieben soll zu einem PDF-Objekt konvertiert werden und dementsprechend ausgedruckt werden können.

1.6.2 Mitarbeiter

- Menüs bestellen

  - Ein Mitarbeiter hat eine Auswahl aller Menüs und kann für jeden Tag eine der drei Hauptspeisen auswählen. Nach der Auswahl kann er die Essenszeit auswählen, die Anzahl und nötige Kommentare hinzufügen.
- Menüs für andere Mitarbeiter bestellen
  - Ein Mitarbeiter kann den obrigen Bestellvorgang für einen anderen Mitarbeiter ausführen.
- Übersicht aller Bestellungen
  - Als Mitarbeiter soll man alle seine vergangenen Bestellungen und deren Informationen in einer Übersicht einsehen können. Diese Übersicht kann filtriert werden.
- Bestellungen stornieren
  - In der oben genannten Übersicht soll man die Möglichkeit haben eine Bestellung auszuwählen und zu stornieren, wenn dies möglich ist.
- Bestellstatistiken einsehen
  - Ein Mitarbeiter soll Diagramme zur Verfügung haben, wo er sein Bestellverhalten einsehen kann.

2 State of Art

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. Citing [**?**] properly.

Was ist eine Globally Unique Identifier (GUID)? Eine GUID kollidiert nicht gerne.

Kabellose Technologien sind in abgelegenen Gebieten wichtig [**?**].

3 Verwendete Technologien

3.1 IntelliJ IDEA

IntelliJ IDEA ist eine der führenden Entwicklungsumgebungen für die Programmiersprache Java. Sie wurde vom Unternehmen Jetbrains im Jahre 2000 entwickelt.
Außerdem bietet sie ebenfalls Entwicklungsmöglichkeiten für Kotlin, Groovy, Scala und auch Android. Sie ist immer auf dem neusten Entwicklungsstand, wird laufend mit Updates versorgt und unterstützt die derzeit gängigen Programmiertools wie Docker, Kubernetes, Maven, Datenbank-Tools, Git, Jakarta EE und viele weitere. Es gibt eine kostenpflichtige Ultimate Version und eine Community Version, die kostenfrei zur Verfügung gestellt wird.
IntelliJ zeichnet auch die Anzahl an Erweiterungen mittels Plugins aus. Die Umgebung besitzt auch eine sehr intuitive Intelligenz, die es dem Entwickler sehr einfach macht damit zu programmieren. [**?**]
Wir haben uns dafür entschieden, da wir damit viel Erfahrung hatten und die oben genannten Punkte unterstützten unsere Entscheidung enorm.

3.2 Android Studio

Android Studio ist eine Entwicklungsumgebung für Android-Anwendungen.
Neben dem leistungsstarken Code Editor werden noch mehr Funktionen, für die Produktivität, bereitgestellt. Android Studio funktioniert auf Windows, GNU/Linux, macOS und Chrome OS. Die erste Version kam, nach zwei Jahren Entwicklungsdauer, am 8. Dezember 2014 raus. Weiters beinhaltet es ein Gradle-build-system, viele verschiedene Android Emulatoren, ein Vorschaufenster mit der Anwendung was bedeutet, dass es konstant aktualisiert wird. Es funktioniert mit den Programmiersprachen Java, Kotlin und C++.

3.2.1 Merkmale

Apply Changes

Das ist eine sehr wichtiges Merkmal für alle Programmierer. Ab der Android Studio 3.5 Version ist es möglich seine Programmierfortschritte zur Laufzeit des Programmes zu ändern ohne die Applikation zu schließen. Um dieses Merkmal benutzen zu können muss man ein so genanntes *debug build variant* benutzen und der Emulator muss die Version 8.0 oder höher haben.

Wenn man die Änderungen verwenden möchte hat man drei Möglichkeit.

- Apply Changes and Restart Activity
- Apply Code Changes
- Run

Falls ein Problem auftaucht nachdem man *Apply Changes and Restart Activity* oder *Apply Code Changes* betätigt, wird sofort vorgeschlagen die App mittels *Run* zu starten.

3.3 Git

Git ist ein Versionskontrollsystem (oft abgekürzt durch VCS) für Entwickler. Es ist ein Open-Source System, das im Jahre 2005 von Linus Torvald entwickelt wurde. Laut einer Stack Overflow-Umfrage von Entwicklern nutzen über 87 % der Entwickler Git. [**?**]
Zu aller erst muss man den Begriff Versionskontrolle erklären, um Git zu verstehen.

3.3.1 Versionskontrolle

Diese dient dazu, um den originalen Quellcode effizient mit mehreren Personen editieren bzw. entwickeln zu können.
Die Entwickler arbeiten mit Verzweigungen und Zusammenführungen. Jeder Entwickler kann Änderungen sicher durchführen, ohne seine Kollegen dabei zu behindern. Diese Änderungen können dann, sobald sie funktionsfähig sind, wieder in den Hauptquellcode eingebunden werden. Alle Änderungen sind zurückzuverfolgen und bei Bedarf kann man sie dann wieder zurücksetzen. [**?**]

3.3.2 Git Funktionsweise

Jeder Entwickler hat seine eigene Version des Projekts (Working Directory), die er frei bearbeiten kann. Diese bekommt man durch einen Klon des Projekts (Clone).
Diese Änderungen kann man aufteilen und in Paketen bereitstellen, nach dem man diese durch Commits trennt. Einen Commit kann man benennen.
Diese Commits kann man dann online veröffentlichen durch einen Push. Ein Push ist nur möglich, wenn man die aktuellste Version des Projekts auf seinen Rechner gezogen hat (Pull). Einem Push kann man einen bestimmten Zweig (Branch) zuordnen. Diese

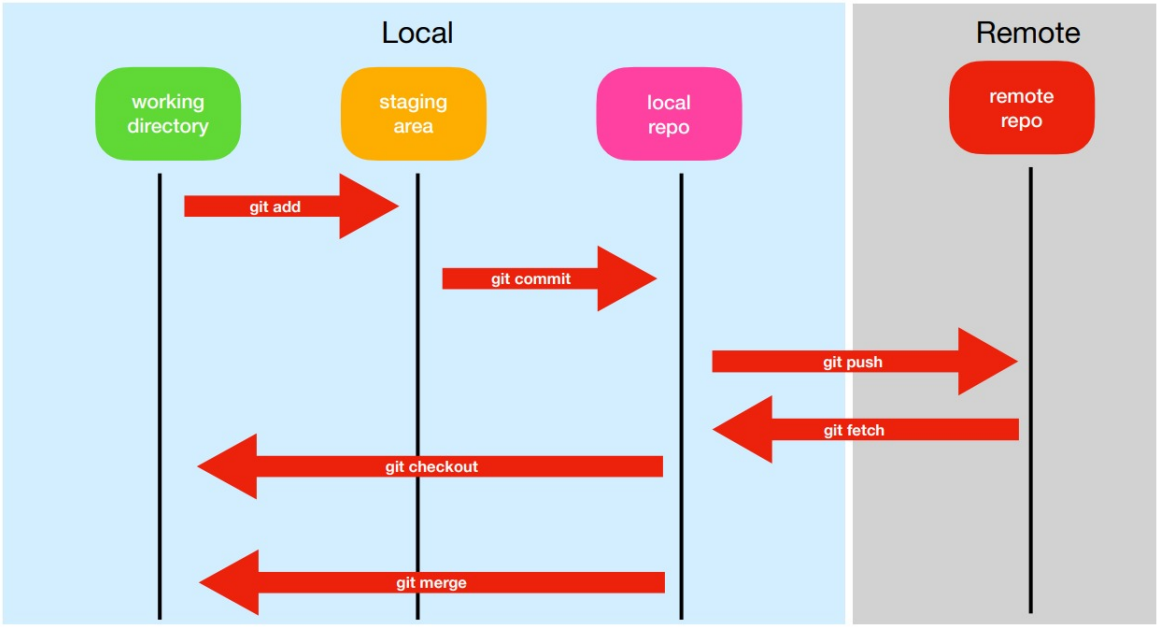

Abbildung 1: Darstellung des Git Workflows

Zweige dienen dazu, um das Projekt in verschiedene unabhängige Teile zu trennen, falls man zum Beispiel an einer Demo Version weiterschreiben möchte.
Nach einem erfolgreichen Push sind die Änderungen online auf dem Repository zu finden. Ein Repository ist der "Ordner", wo alle Dateien online gespeichert zu finden sind.
Ein Vorteil, den Git ebenfalls bietet ist das jeder Commit eine Version des Projekts ist, die man bis zum Zeitpunkt vom Commit herunterladen oder klonen kann. Jedes Projekt kann privat oder öffentlich gemacht werden, sodass auch Personen, die keine Entwickler sind, darauf zugreifen können. [?]

3.3.3 Git Befehle

Clone

Es initialisiert ein Git Repository auf dem Rechner und ladet die zugehörigen Dateien runter. Wenn man es nicht spezifisch angibt, klont es den Master Branch. Der Master Branch ist der Hauptzweig, eines jeden Git Projekts. Innerhalb des erstellten Ordners können alle weiteren Git Befehle ausgeführt werden. [**?**]

Commit

Ein Commit beschreibt Änderungen, die man im Projekt gemacht hat. Jeder Commit hat eine Bezeichnung, mit der der Entwickler die Änderungen, die er gemacht hat beschreiben kann. Zu jedem Commit gehören auch die Dateien die dabei geändert bzw. hinzugefügt wurden.
Es speichert den Zustand des gesamten Projekts bis zu dem Zeitpunkt und kann danach jederzeit abgerufen oder rückgängig gemacht werden. Diese Änderungen bleiben aber zunächst nur lokal auf dem Rechner. [**?**]

Push

Ein Push dient dazu, um die lokalen Änderungen (Commits) zu veröffentlichen. Es kopiert den aktuellen, lokalen Stand und speichert diesen auf das vom Internet erreichbare Repository.
Einem Push kann ein Zweig (Branch) zugeordnet werden um die Änderungen zuzuordnen. Ansonsten wird der Master Branch genommen. [**?**]

Pull

Der Pull Befehl kopiert die Inhalte vom öffentlichen Repository und fasst diese mit den lokalen Zustand auf dem rechner zusammen (merge). Es dient dazu die aktuelle Version des Projekts auf den Rechner herunterzuladen.
Falls es Konflikte zwischen der derzeitigen und neusten Version gibt, werden die Änderungen zusammengeführt. [**?**]

Branch

Die Branch Befehle dienen dazu um eine neue Abzweigung des Projekts (Branch) zu erstellen.
Branches erstellt man, wenn man an einer neuen Version des Projekts arbeitet und diese vom Hauptteil trennen will. Meistens werden dadurch neue Funktionen programmiert, die später wieder in den Master Branch eingebunden werden.
Der Vorteil daran ist, dass man an neuen Funktionalitäten experimentieren kann, ohne den Hauptentwicklungsstand zu beeinflussen. Branches können jederzeit gelöscht oder wieder ins Hauptprojekt integriert werden. Es kann simultan am Master Branch und an Nebenzweigen gearbeitet werden durch trennen mit dem Push Befehl. [?]

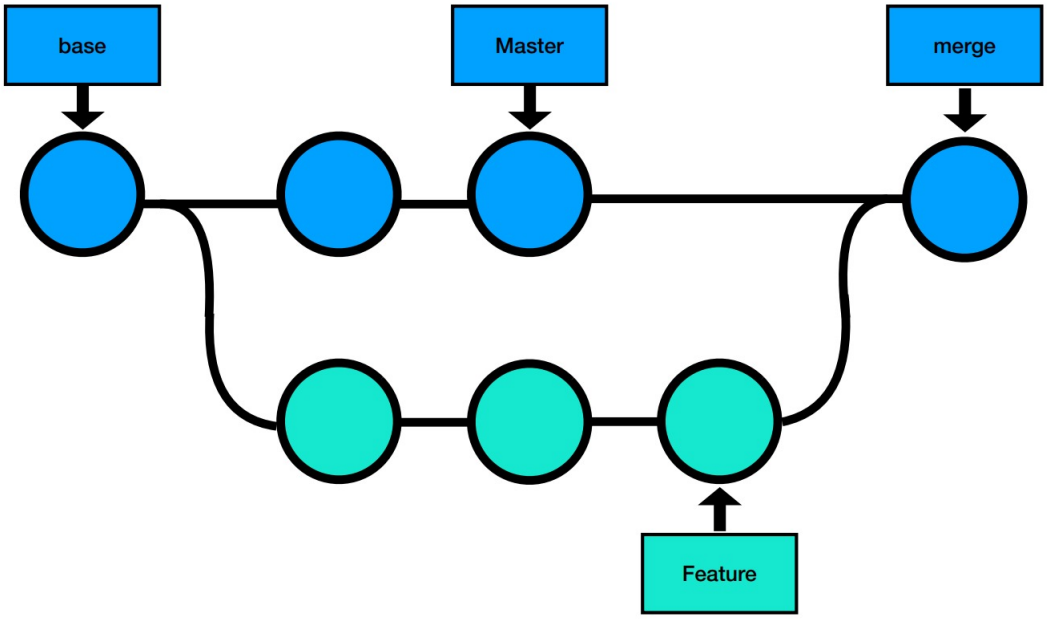

Abbildung 2: Darstellung von Branches in einem Repository

3.3.4 GitHub

GitHub ist ein gewinnorientiertes Unternehmen, dass einen auf Cloud basierten Git Repository Hosting-Service anbietet. Es wurde im Februar 2008 gestartet und von Chris Wanstrath, PJ Hyett, Scott Chacon und Tom Preston-Werner entwickelt. Es ist das beliebteste Tool um Softwareprojekte zu verwalten und wird von über 73 Millionen Entwicklern und über 4 Millionen Organisationen benutzt. Dazu ist es das größte und am meisten fortgeschrittene Entwicklungssystem, das es gibt. [?]
Es vereinfacht die Nutzung von Git für Teams und auch Einzelpersonen. Jeder kann sich einen GitHub Account erstellen und direkt loslegen und seine Arbeiten auf Reposi-

tories veröffentlichen. Es ist nicht nur zwingend für Code-basierte Projekte verwendbar sondern auch Websiten erstellen und das Schreiben von Büchern ist möglich.
Was GitHub ausmacht ist die Benutzerfreundlichkeit und die Integration von Git. Außerdem bietet Github viele andere Funktionen wie zum Beispiel ein Projekt Board an, was es erleichtert innerhalb eines Teams, Probleme besser lösen zu können.
Es gibt ebenfalls bezahlte Pläne, die es vor allem Organisationen und Unternehmen leichter macht Unternehmensprojekte zu verwalten durch zusätzliche Funktionen. [?]

3.4 GitHub Actions

GitHub Actions ist ein von GitHub entwickelte Plattform, die das Automatisieren von verschiedenen Aspekten eines Softwareprojekts in Form von Pipelines ermöglicht. Dabei handelt es sich um das Kompilieren, Testen und das Deployment eines Projekts.
Es implementiert die beiden Methoden CI und CD. [?]

3.4.1 CI und CD

CI und CD sind zwei zusammenhängende Methoden, um Entwickler das Programmieren an einem Projekt zu vereinfachen. Ebenfalls wird die Bereitstellung an den Kunden vereinfacht.

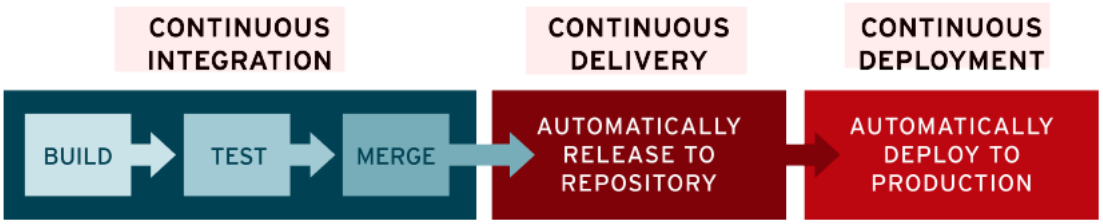

Abbildung 3: Zusammenhang zwischen CI und CD
https://www.redhat.com/cms/managed-files/ci-cd-flow-desktop_edited_0.png

CI (Continuous Integration)

CI steht für "Continuous Integration", dies stellt den Prozess der Automatisierung in der Sicht der Programmierers da.
Wenn mehrere Entwickler gleichzeitig an einem neuen Feature arbeiten, kann es passieren, dass der Code beim Zusammenführen oft die Funktionsfähigkeit des Programms

beeinträchtigt. CI soll dies verhindern, indem es einen automatisierten Prozess bereitstellt, der das Programm auf bestimmte Validierungen (Tests) prüft nach jeder Zusammenführung.
Dies erlaubt es den Entwicklern, den Code öfter zusammenzuführen, sogar eine tägliche Zusammenführung ist dadurch manchmal möglich [**?**]

CD (Continuous Delivery/Deployment)

CD steht für "Continuous Delivery" bzw. "Continuous Deployment", dies sind zwei Konzepte die immer gemeinsam genutzt werden.
Continuous Delivery sorgt dafür, dass Änderungen eines Entwicklers automatisch in ein Repository veröffentlicht werden. Danach können diese Änderungen sofort in einer Produktionsumgebung dargestellt werden.
Nach der Delivery kommt das Continuous Deployment, dies stellt dem Kunden finale Änderungen am Programm, in einem automatisiertem Prozess, zur Verfügung. Das heißt, nachdem die Entwickler zufrieden sind mit einer Version, können sie diese veröffentlichen und der Kunde kann das Programm dann benutzen.[**?**]

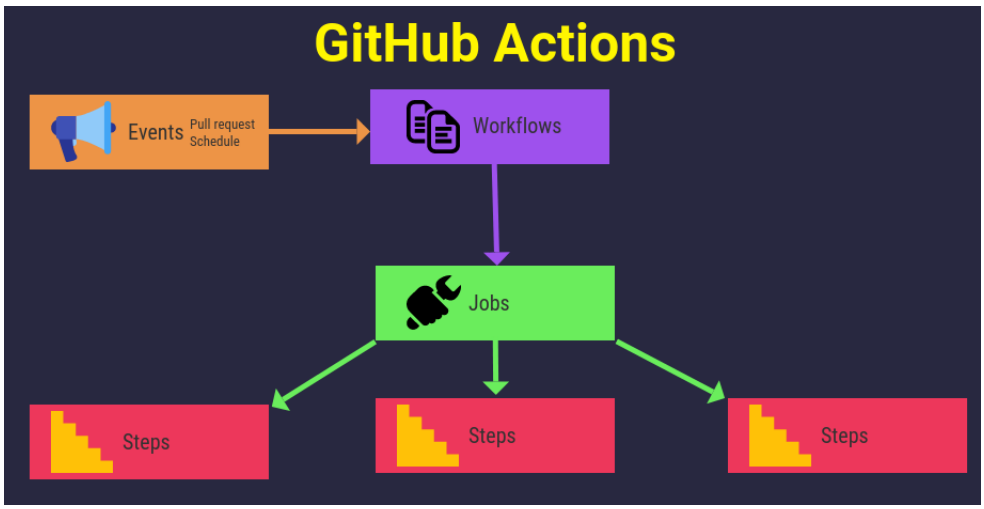

Abbildung 4: Ablauf von Github Actions
https://miro.medium.com/max/1400/1*8Agtv-SaM-zw_GLzPI7p5w.png

3.4.2 Workflows

GitHub Actions benutzt Workflows, um Arbeitsschritte, die zusammenhängen, gemeinsam auszuführen. Zum Beispiel gibt es Workflows zum Deployen, nur zum Testen oder um eine Dokumentation hochzuladen.

Ein Workflow ist ein automatisierter Prozess, der einen oder mehrere Jobs ausführt. Diese werden mittels eines .yaml-File beschrieben. Ein Workflow wird durch ein bestimmtes Event ausgeführt, z.B. ein Push auf die main-Branch. Mann kann es aber auch manuell ausführen oder nach einem bestimmten Zeitplan. [?]

Listing 1: Ausschnitt eines Workflow Files

```
1  # Workflow Bezeichnung
2  name: Publish EazyMenue Frontend and Backend
3
4  # Definiert wann der Workflow ausgefuehrt wird
5  on:
6    # Trigger wenn ein Push auf den main Branch
       erfolgt
7    push:
8      branches: ['main']
9
10 # Jobs die nacheinander oder parallel ausgefuehrt
     werden
11 jobs:
12   build_backend:
13     name: Build backend
```

3.4.3 Event

Ein Event ist eine Aktivität, die einen Workflow ausführt. Er dient als sogenannter Trigger. Ein Event könnte z.B. ein Push auf eine Branch sein oder ein Pull Request etc.[?]

3.4.4 Jobs

Ein Job hat mehrere Steps (Stufen), die nacheinander ausgeführt werden. Ein Step kann z.B. ein Shellscript File sein, ein Login in ein anderes Repository oder eine vordefinierte Action. Der Jobs fasst diese Steps zusammen und die Steps können untereinander die selben Ressourcen nutzen.[?]

Listing 2: Ausschnitt eines Jobs

```
1       build_backend:
2       name: Build backend
3       # Der Job wird in einem Ubuntu Container
           ausgefuehrt
4       runs-on: ubuntu-latest
5       env:
6       # Unter env kann man sich Variablen definieren
7         IMAGE_NAME: eazy-menue-backend
8       defaults:
9         run:
10        # Es wird der backend-Ordner aus dem Repo
             genommen
11          working-directory: ./eazy-menue-backend
12      steps:
13        - name: Check out the repo
14          uses: actions/checkout@v2
15        - name: Package
16          run: mvn package -DskipTests
17        - name: Build image
18          run: docker build . -f
               src/main/docker/Dockerfile.jvm --tag
               $IMAGE_NAME
```

3.4.5 Actions

Die GitHub Actions Plattform stellt verschiedene vordefinierte Actions zur Verfügung. Eine Action führt eine bestimmte komplexe Aufgabe aus, die auch oft benutzt wird in Workflows. Diese Actions helfen dabei, um die Wiederverwendung von bestimmtem Code bzw. Codeblöcken zu vermeiden. Actions umfassen z.B.: ein Repository pullen

oder sich bei einem Dienst zu authentifizieren.
Es ist auch möglich selber Actions zu schreiben, falls man keine passende auf dem GitHub Marketplace findet.[?]

3.5 Github Packages

GitHub Packages, entwickelt von GitHub, ist ein Dienst, der es ermöglicht Software-Pakete hochzuladen und diese privat oder öffentlich zu nutzen. Dies beinhaltet Softwarecontainer und Dependencies.
Das Ziel von GitHub Packages ist es den Code und die Software selbst auf einer Plattform zusammen zu Verfügung stellen zu können. Der ganze Entwicklungsprozess ist somit auf GitHub zentralisiert. Zusammen mit GitHub Actions kann man die Veröffentlichung komplett automatisieren.
Es stellt Package Registries (Sammlungen) für die meist benutzten Package Manager wie npm, RubyGems, Apache Maven, Docker (Docker Images), NuGet und Gradle zur Verfügung. [?]

Nützliche Funktionen von GitHub Packages sind:

- Rechte: Man kann bestimmen, wer die Packages ändern bzw. wer diese sehen und benutzen darf.
- Sichtbarkeit: Man kann einstellen ob das Package nur privat oder öffentlich nutzbar ist.

3.5.1 Github Container Registry (ghcr.io)

Das GitHub Container Registry (ghcr) ist aus den Packages entstanden, um eine eigene Plattform für Softwarecontainer zu haben, die zu GitHub gehört. Somit wird GitHub noch mehr in die Entwicklung integriert.
Man kann es mit DockerHub vergleichen, eine Plattform um Docker Images hochzuladen und an einem Ort zu sammeln. Es besteht ebenfalls die Möglichkeit Docker Images auf das ghcr hochzuladen, anstatt DockerHub zu benutzen.
Es hat die selben Rechte- und Sichtbarkeitsfunktionen wie oben genannt. Darüber hinaus kann man auch verschiedene Versionen eines Images und Statistiken über die Nutzung einsehen.[?]

3.6 Java

Die relativ junge Programmiersprache Java, welche 1995 entwickelt wurde, weckte schon von Beginn an das Interesse der Programmiergemeinschaft auf. [?]
James Gosling, welcher ursprünglich aus Kanada abstammt, entwickelte die objektorientierten, plattformunabhängigen Programmiersprache Java. [?]
Die Programmiersprache Java ist Teil der Java-Technologie. Sie besteht im Wesentlichen aus dem Java Development Kit (JDK) zum Erstellen von Java-Programmen und dem Java Runtime Environment (JRE) für deren Ausführung. Die Programmiersprache selbst sollte aber nicht gleichgesetzt werden mit der Java-Technologie. [?]

3.6.1 Hello World

Listing 3: Java File HelloWorld

```
1 public class HelloWorld {
2     public static void main (String[] args){
3             // Ausgabe Hello World!
4             System.out.println("Hello World!");
5     }
6 }
```

3.6.2 Java Runtime Environment

Java Runtime Environment (JRE) führt Software aus, die in der objektorientierten Programmiersprache Java geschrieben ist. Denn bei Java benötigt der Computer im Vergleich zur Programmiersprache C eine Laufzeitumgebung „Java Runtime Environment“ für das Betriebssystem, um Java-Programme ausführen zu können. [?]

3.6.3 Java Virtual Machine

Java Virtual Machine (JVM) ist eine virtuelle Maschine, welche plattformunabhängige Anwendungen ermöglicht. Somit lassen sich Systemweit Java-Programme ausführen. Die Plattformunabhängigkeit wird in Java durch das Zusammenspiel zweier Programme gelöst:
Der Compiler wandelt den Quelltext, also die .java-Dateien, in einen sogenannten Java-Bytecode. Der Interpreter (Virtual Machine) führt dann den Java-Bytecode aus. [?]

3.6.4 Java Byte-Code

Bytecode in Java ist der Grund dafür, dass Java plattformunabhängig ist. Sobald ein Java-Programm kompiliert wird, wird der Bytecode generiert. Genauer gesagt ist ein Java-Bytecode welcher Maschinencode in Form einer .class-Datei.

3.7 JUnit

Um ganz einfach einen beliebigen Java Code zu überprüfen und zu testen, verwenden wir JUnit. JUnit ist ein Test Framework welches von Erich Gamma und Kent Beck entwickelt wurde. [**?**]

Um JUnit zu verwenden, brauchen wir, wie schon erwähnt, einen beliebigen Java Code, welchen der Programmierer geschrieben hat. Mithilfe von sogenannten Unit Tests werden dann einzelne Testfälle von verschiedenen Klassen getestet. In JUnit verwendet man ***assertEquals(Parameter1,Parameter2)***. ***Parameter1*** gibt den Aktuellen Wert, welchen zum Beispiel ein Objekt hat, an. ***Parameter2*** gibt den Wert an, welcher erwartet wird. Natürlich gibt es dann auch noch ***assertNotEquals(Parameter1,Parameter2)*** welches schaut das die übergebenen Parameter nicht übereinstimmen.

Die Akuelle JUnit Version, welche wir verwenden, heißt JUnit 5. Diese Version setzt sich auch mehreren Modulen zusammen. JUnit 5 besteht aus JUnit Platform + JUnit Jupiter + JUnit Vintage.

3.8 Karate

Karate wird verwendet, um Enpoints testen zu können. Die Syntax ist sehr einfach und leserlich für Menschen, die wenig bis keine Programmierkenntnisse haben. Karate selbst enthält nicht nur Testen von Enpoints, sondern auch HTML Reports, welche alle Testfälle dokumentiert und anzeigt.

Einzelne Testfälle bezeichnet man in Karate als Scenarios. In diesen Scenarios werden dann CRUD Operationen getestet. Den Path der Abfrage und die einzelnen Parameter

werden in den Scenarios mitgegeben, aber auch was für ein Response code zurückkommen soll.

3.9 Java EE

Die Enterprise Edition der Java Plattform (JEE), eine Spezifikation der Java-Plattform. Der wichtigste Bestandteil von Java EE sind die Webanwendungen. Um solch eine Anwendung ausführen zu können, wird eine Applikationsserver verwendet, welcher in mehrere Systeme unterteilt wird, die auch Container genannt werden. Für eine Webanwendung reicht meistens reicht ein einzelner Web-Container aus.

Die erste Version von Java EE, welche damals noch Java2EE genannt wurde, erschien im Jahr 1990. Im Jahr 2003 wurde der Name der Plattform auf Java EE geändert. Die erste richtige Version von Java EE, welches wir bis heute noch kennen, erschien im Mai 2006 und trug die Versionsnummer 5. 2009 folgte die Version 6 und 2013 die Version 7.

Wie schon bereit erwähnt, wird zum Ausführen einer Java EE Webanwendung ein Applikationsserver verwendet. Die am meisten verwendeten Applikationsserver sind so genannte Open-Source-Server wie zum Beispiel Apache Geronimo und GlassFish. Als Alternative zu einem vollständigen Applikationsserver können auch Programme verwendet werden, welche nur einen Web-Container implementieren. Diese Web-Container werden oft auch als Servlet- oder JSP-Container bezeichnet und sind meistens dafür ausrecheichend. Der bekannteste Web-Container ist Apache Tomcat.

[**?**]

3.10 Quarkus

Quarkus ist ein Cloud-nativer Stack welcher von Red Hat entwickelt wurde. Es basiert auf den Jakarta EE- und Micro Profile-Standards. Eine Quarkus-Applikation hat gegenüber ein Konkurrenzprodukt eine deutlich kürzere Startzeit und einen erheblich geringeren Speicherbedarf, was ein großer Vorteil ist für das Serverless Computing. [**?**]

Quarkus wurde für entwickelt um de Entwicklern die Möglichkeit zu geben, Applikationen für die moderne, Cloud-native Welt zu erstellen. Es ist ein Kubernetes-natives Java-Framework, das auf GraalVM und HotSpot angepasst wurde. Noch dazu wurde es aus erstklassigen Java-Bibliotheken und Standards erstellt.

Im März 2019 wurde Quarkus nach mehr als einem Jahr interner Entwicklung der Open-Source-Community vorgestellt

3.10.1 Java EE vs. Quarkus

Der größte Unterschied zwischen Java EE und Quarkus ist, dass man bei Quarkus keinen Application Server benötigt. Ein weiter Vorteil wäre, dass bei Quarkus keine war-File erstellt wird, sondern eine jar-File, welche dem entsprechend sehr klein ist. In diesem jar-File befinden sich auch noch die einzelnen Libaries, welche für die Webanwendung verwendet werden.

Im Gegensatz zu Java EE wird eine war-File erstellt. Noch dazu befinden sich auf dem JEE-Application Server schon viele Libraries die schon vorinstalliert sind. Der größte Nachteil von Java EE ist, dass man auch für kleine Anwendung einen ganzen Application Server braucht.

Im Großen und Ganzen kann man sagen, dass Quarkus hier dominanter ist, da es sehr praktisch und schnell ist. Noch dazu ist Quarkus relativ neu und kann sich im Laufe der Jahre noch umso mehr verbessern.

Am Anfang haben wir Java EE gewählt, da es so vorgesehen war. Mit Java EE haten wir einige Probleme wie zum Beispiel die Zeit zum deployen. Mit Quarkus aber haben wir uns sehr viel zeit gespart und konnten dadurch auch mehr Zeit in andere Sachen investieren wie zum Beispiel in das Testen.

Die Konfiguration bei Quarkus war auch um einiges einfacher als bei Java EE. Bei Quarkus findet die Konfiguration in den sogenannten **Application.Properties** statt. Bei Java EE war die Konfiguration auf dem Application Server. Hier war nicht nur eine File sondern mehrere .xml Files. Das war ebenso noch ein Grund warum wir uns zum Schluss doch noch für Quarkus entschieden haben.

3.10.2 JPA

Java Persistence API (JPA) ist eine API welche für Datenbankzugriffe und für das objektrelationales Mapping verwendet wird. Objektrelationales Mapping (ORM) zeigt eine objektorientierte Sicht auf Tabellen und Beziehungen in relationalen Datenbank-Management-Systemen an.

Enterprise Applikationen Führern üblicherweise viele Operationen auf Datenbanken durch. Um aber eine Datenbankanbindung zu haben, müsste man sehr viel Code schreiben. Durch die Hilfe von JPA (Java Persistance API) wird der Aufwand reduziert, welcher benötigt wird, um mit der Datenbank zu kommunizieren.
[?]

JPA wurde von JSR 220 Expert Group entwickelt und im Mai 2006 erstmals veröffentlicht.

3.10.3 Hibernate

Hibernate wird oft als Object Relational Mapping Tool bezeichnet. Es ist ein Framework zur Abbildung von Objekten auf relationaler Datenbank für die Programmiersprache Java. Hibernate implementiert die Java Persistence API (JPA)und bietet Funktionalitäten wie zum Beispiel Datenbank abfragen mit der sogenannten Hibernate Query Language (HQL).

Es wurde von JBoss im Jahr 2001 entwickelt und veröffentlicht. Somit ist auch Hibernate in dem Boss Applikations-Server integriert .[?]

3.11 JBoss

JBoss ist ein Tochterunternehmen von Red Hat, welches Unterstützung für die gleichnamige Server-Plattform (JBoss Application Server) und zugehörige Services unter der Marke JBoss Enterprise Middleware Suite (JEMS) bietet. [?]

3.11.1 JBoss Application Server

Der JBoss Application Server ist eine J2EE-Plattform. Mithilfe von J2EE werden standardisierte, modulare Komponenten verwendet. JBoss ist ebenfalls innerhalb des Amazon Cloud-Services EC2 verfügbar. [**?**]

3.11.2 Panache

Panache ist eine Quarkus-spezifische Bibliothek, welche die Entwicklung einer Hibernate-basierten Persistenz Schicht vereinfacht. Panache ersetzt den größten Teil des Boilerplate-Codes mit einfachen Methoden die man in die Quarkus Anwendung implementieren kann. Methoden wie create, update, and remove aber auch eigene Abfragen auf der Datenbank werden von Panache zur Verfügung gestellt.
[**?**]

3.12 Maven

Apache Maven ist ein leistungsfähiges Werkzeug, welches immer wieder vorkommende Prozeduren automatisiert und vereinfacht. Es wird oft als "Build Management System" bezeichnet und ist Teil vom "Software Configuration Management (SCM)".
[**?**]

Neben Maven gibt es auch noch ANT was eher kommandoorientiert arbeitet. Maven ist eher strategisch orientiert, realisiert mehr Abstraktionen, wird deklarativer gesteuert, berücksichtigt Abhängigkeiten besser und ist besonders für aufwändigere Multimodulprojekte geeignet. [**?**]

3.13 Cypress

3.14 Keycloak

Keycloak ist eine Open-Source Software, die Single-Sign On mit Identity und Access Management für moderne Applikationen bereitstellt. Der erste Release war 2014 als Wildfly Community Projekt, seit 2018 aber steht es unter der Verwaltung von RedHat.

[?]
Es hat mehrere Distributionen und ist mit einer Vielzahl von Frameworks und Tools kompatibel bzw. integriert wie z.B.: Quarkus, Angular, Vue.js, Spring, usw. [?]

3.14.1 Distributionen

Server

Die Standalone Applikation ist downloadbar als .zip auf der Keycloak Seite. Es gibt zwei Versionen vom Server, zu einem die Wildy Application Server Version und auch eine Version, die über Quarkus läuft.

Docker Image

Genau so wie beim Server gibt es auch hier zwei verschiedene, offizielle Docker-Images, eins, dass auf dem Wildfly Server basiert und eins, dass auf Quarkus basiert.

Operator

Dies ist eine Distribution für Kubernetes und OpenShift, basierend auf der Operator SDK. [?]

3.14.2 IAM (Identity Access Management)

Eine der Hauptfunktionen von Keycloak ist das IAM.
IAM oder auch IdM (Identity Management) ist ein Framework, das zum Authentifizieren von Benutzern und deren Rechten genutzt wird.
Es prüft ob der Benutzer Zugang zu bestimmten Bereichen, Dateien und anderen Ressourcen hat, auf die er zugreifen will. Es prüft auch wer diese Rechte verändern darf. IAM-Systeme stellen auch nützliche Admin-Tools zur Verfügung, um z.B. Nutzerrechte zu ändern oder die Aktivität von Benutzern zu verfolgen. [?]
IAM hat vier Hauptfunktionen:

- **Die Identitäts-Funktion:** Es umfasst die Erstellung von Identitäten (Benutzern), das Managen und Löschen von diesen.
- **Die User Zugriff-Funktion (log-on):** Dies umfasst ein Interface, indem der Benutzer seine Zugriffsdaten eingeben (übermitteln) kann.

- **Die Service-Funktion:** Für Benutzer und deren Geräte stellt das System personalisierte, rollenabhängige, multimedia, online und on-demand Services zur Verfügung. [**?**]

3.14.3 Single Sign-On (SSO)

SSO ist eine Variante der Zugangskontrolle, deren sich Keycloak bedient.
Es ist für mehrere und zusammenhängende Softwaresysteme gedacht, wo der Benutzer nur einmal sein Passwort und Benutzernamen eingeben muss, um auf alle Systeme zugreifen zu können, ohne sich dazwischen neu identifizieren zu müssen.
SSO ist typischerweise zu ermöglichen mit LDAP (Lightweight Directory Access Protocol). Bei IP-Netzwerken funktioniert das Ganze über Cookies, die gespeichert bleiben und die Seiten müssen eine gleiche, übergeordnete DNS Domain haben.
Authentifizierungsschemas, die dies unterstützen sind: OAuth, OpenID, OpenID Connect und Facebook Connect. Diese ermöglichen das Einloggen über mehrere verschiedene Websites mit den selben Anmeldedaten.[**?**]

Vorteile von SSO

- Reduziert das Risiko beim Verwenden von Drittanbieter-Websites
- Reduziert die Passwortschwäche von den verschiedenen User und Passwort Kombinationen.
- Reduziert die Zeit, die man für das Eingeben der verschiedenen Identitäten braucht
- Reduziert IT-Helpdesk Anrufe wegen Passwörtern, daraus werden auch die IT-Kosten reduziert [**?**]

3.14.4 OAuth 2.0

Es ist möglich mit Keycloak den Standard OAuth 2.0 umzusetzen.
OAuth 2.0 ist ein offener Standard, der im Oktober 2012 als Überarbeitung des Standards OAuth veröffentlicht wurde.
Er behandelt ausschließlich den autorisierten Zugriff eines Benutzers auf eine Ressource. Wichtig zu beachten ist, dass er die Identität nicht überprüft.
Ein Anwendungsfall von OAuth wäre einen Client zu verwenden, um Neuigkeiten auf

einem Social-Media Profil zu veröffentlichen. Der Client holt die Autorisierung dazu aus der Social-Media Plattform. [**?**] [**?**]

Ablauf von OAuth 2.0

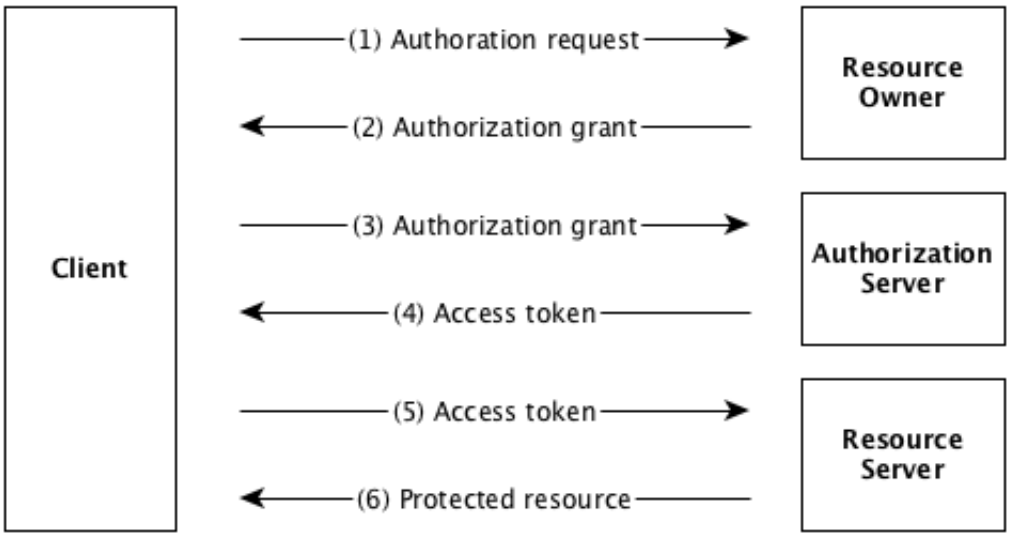

Abbildung 5: Ablauf des OAuth 2.0 Protokolls
https://miro.medium.com/max/1400/1*1McvnvrW6wh37ECYpmTSxw.png

Diese Grafik beschreibt den Weg, den der Client geht, um sich die Rechte auf die bestimmte Ressource (z.B. Veröffentlichung eines Beitrags) zu sichern.
Wie man an der Grafik erkennen kann kommuniziert OAuth 2.0 nicht mit dem Benutzernamen und Passwort des Benutzers. Als Kommunikationsgrundlage dient ein Access-Token, dieser kann jeder Zeit ungültig gemacht werden, somit verliert der Client auch die Rechte auf die Ressource. [**?**]

3.14.5 OpenID Connect (OIDC)

Mit Keycloak ist es ebenfalls möglich OIDC umzusetzen bzw. zu benutzen.
OpenID Connect ist ein Authentifizierungs- und Autorisierungsprotokoll, dass im Februar 2014 von der OpenID Foundation veröffentlicht wurde. Es erweitert die Funktionalität von OAuth 2.0 und benutzt dazu auch JWT (JSON Web Tokens).
OpenId Connect soll die zentrale Stelle zur Verwaltung eines Benutzerprofils sein. Somit läuft auch jeder Authentifizierungsprozess über diesen ab. Der Browser soll quasi die zentrale Komponente im Internet darstellen und jede Website benutzt die selbe OIDC Authentifizierung. Somit werden einem mehrere Benutzeraccounts gespart. [**?**]

Ablauf von OIDC

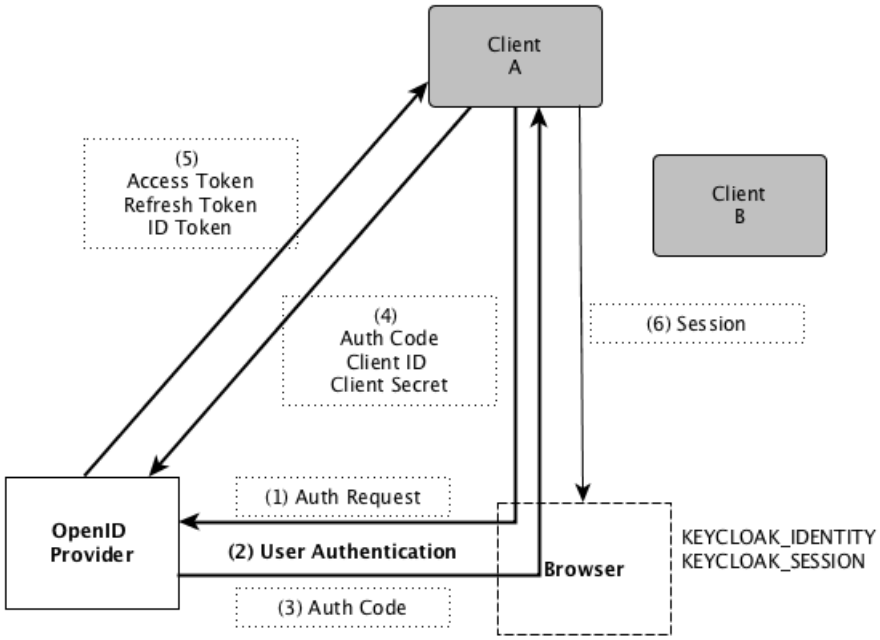

Abbildung 6: Ablauf des OIDC Protokolls
https://blog.codecentric.de/files/2016/08/openidconnect1.png

Der Browser leitet anfangs den Benutzer zum OpenID Provider hin, wo man sich beispielsweise mit Benutzername und Passwort authentifiziert. Danach bekommt der Client einen Auth-Code, dieser wird dann zum Access, Identity und Refresh Token umgewandelt. Der Client wird durch Keycloak verifiziert, indem er vorher bei Keycloak mit einem Secret und einer ID konfiguriert wurde. Dies dient dazu, damit kein willkürlicher Client diese Information erhalten kann. In dem ganzen Prozess muss aber sichergestellt sein, dass jede Komponente das Secret sicher bewahren kann. Zu allerletzt setzt der Browser dann die Session vom Benutzer und diese bleibt auch. [**?**]

Unterschied zu OAuth 2.0

Der initiale Ablauf ähnelt dem von OAuth 2.0, doch der große Unterschied liegt in den Tokens. Bei OIDC wird ein JWT (Json Web Token) übergeben. Dieser enthält Informationen zur Identität und zu den Attributen des Benutzers. Diese können dann von der Applikation benutzt werden. [**?**]

3.14.6 Features

Multiple Protocols Support

Gerade unterstützt Keycloak drei verschiedene Authentifizierungsprotokolle: OpenID Connect, OAuth 2.0 und SAML 2.0 [**?**]

SSO (Single Sign-On)

Siehe hier

Admin Konsole

Keycloak stellt eine web-basiertes Interface zur Verfügung, wo der Admin seine Konfigurationen intuitiv vornehmen kann. [**?**]

User Identity und Accesses

Siehe hier

External Identity Source Sync

Wenn man ein eigene User-Datenbank hat, kann man diese mit Keycloak verbinden und synchronisieren. Standardmäßig unterstützt es LDAP und Active Directory, aber diese kann man durch selbstgeschriebene Extensions erweitern. Diese kann man mit der Keycloak User-Storage API machen. Es garantiert aber nicht, dass Keycloak alle Funktionen aufweist, die auch die Datenbank hat. [**?**]

Identity Brokering

Keycloak kann auch als Proxy zwischen den Usern und einem oder mehreren externen Identity Provider fungieren. Diese können im Admin Panel editiert werden. [**?**]

Social Identity Providers

Zusätzlich erlaubt Keycloak die Benutzung von Social Identity Providern. Es hat eine eingebaute Unterstützung für Google, Twitter, Facebook, Stack Overflow, diese müssen aber im Admin Panel manuell konfiguriert werden. Die volle Liste der kompatiblen Social Identity Provider kann in der Keycloak Dokumentation gefunden werden. [**?**]

Anpassung von Seiten

Keycloak erlaubt eine Anpassung aller Seiten, die dem User angezeigt werden (z.B. Login-Page). Diese sind im .ftl Format, somit kann man klassisches HTML und CSS zum editieren verwenden. Auch Javascript steht zur Verfügung, damit ist der Anpassungsspielraum unendlich. [**?**]

3.14.7 Funktionsweise

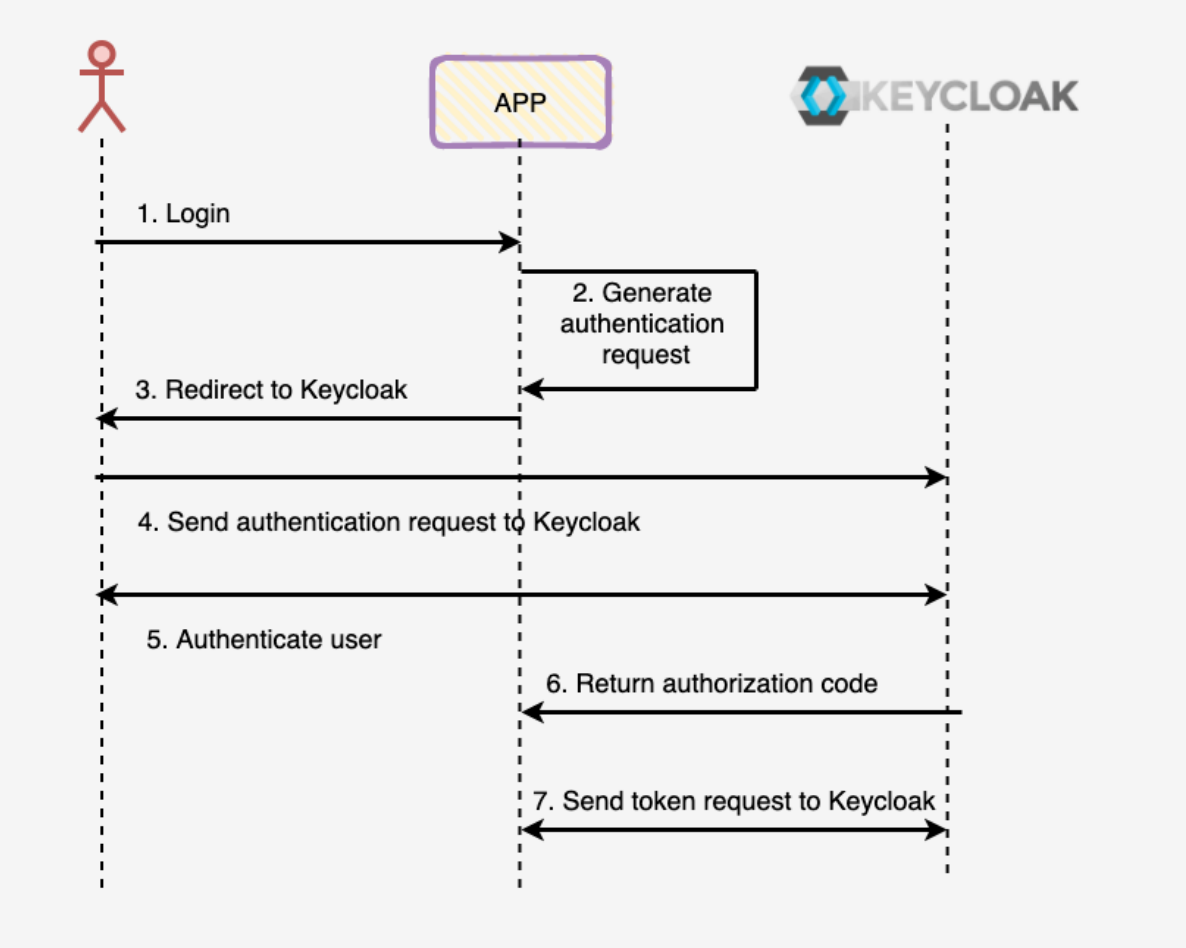

Abbildung 7: Keycloak Funktionsweise
https://miro.medium.com/max/1400/1*1McvnvrW6wh37ECYpmTSxw.png

Keycloak speichert sich einen Public Link der Applikation. Wenn dieser Link von einem User geöffnet wird, leitet Keycloak diesen zu einer Keycloak Authentication Page weiter. Nachdem erfolgreichen Einloggen leitet Keycloak den User dann zu dem eigentlich gewünschtem Link weiter und gibt einen Token mit Zeitstempel mit. Die Applikation kann dann den Token verwenden um den User und seine Rechte zu identifizieren. [**?**]
Bei der Funktionsweise sind wichtige Begriffe von Nöten:

Realm

Ein Realm kann man sich als Mieter vorstellen, der einen Bereich zur Verfügung stellt. Dieser sogenannte Realm ist vollkommen isoliert (im Sinne von Usern, Konfigurationen, Rollen, etc.) von anderen Realms.
Aus diesem Grund kann man einen internen Realm für z.B. Mitarbeiter machen und einen externen für z.B. Kunden. Somit sind die beiden von einander getrennt. [**?**]

Client

Clients sind die Anwendungen, die eine Authentifizierung von Keycloak fordern, also z.B. eine Webapp. Diese können aber auch mobile oder native Applikationen sein. Sie können jeder Servicetyp sein wie REST API's, gRPC oder WebSockets, die nur eine simple Authentifizierung und Rollenvergabe mittels Access Tokens benötigen. [**?**]

Rollen

Eine Rolle repräsentiert eine Rolle in der Organisation bzw. in der Applikation. Ein User kann z.B. eine Admin-Rolle bekommen und somit alles an der Applikation konfigurieren. Wenn man dies nicht will kann man Rollen und deren Spielraum eingrenzen, sodass die Applikation nicht willkürlich verändert wird.
Keycloak unterstützt ebenfalls zusammengesetzte Rollen. Diese Funktion sollte man aber vorsichtig behandeln, da diese die Komplexität der Applikation erhöht und diese somit schwerer zu pflegen ist. [**?**]

User

Der User ist der eigentliche Benutzer der Applikation. Dieser kann dementsprechend zu einem Realm gehören und seine eigenen Rechte haben. Dieser identifiziert sich durch den Client und seine Rollen im Realm werden der Applikation mitgegeben.

3.15 Oracle Datenbank

Oracle Database ist ein relationales Datenbankmanagementsystem, auch genannt RDBMS, welche von der Firma Oracle hergestellt wurde. Oracle ist einer der bekanntesten und auch meist verwendet Datenbanken. Die Firma Oracle wurde am 17. August 1977 Lawrence Joseph Ellison in der Bronx, New York City, gegründet. Die erste Version von Oracle Database kam 1979 auf den Markt. [**?**]

Die Oracle Database nutzt wie die meisten relationale Datenbanken, SQL als Programmiersprache (Structured Query Language). SQL wird in der Oracle Database verwendet, um Aktionen durchzuführen, Datenbankstrukturen zu schaffen, Datensätze zu verwalten oder enthaltene Daten abzurufen. PL/SQL, welche eine Oracle-eigen Sprache ist, ist wiederum eng verknüpft mit SQL und bietet die Möglichkeit, SQL um

Oracle-Programmiererweiterungen zu ergänzen. Oracle verwendet Zeilen- und Spaltentabellen zur Strukturierung der Datenbanken, in denen Datenpunkte über Attribute verbunden sind [?]

Wichtige Oracle-Database-Werkzeuge währen Oracle SQL Developer und Oracle Data Modeler.

Mit dem Oracle SQL Developer kann man leicht Datenbankabfragen auf einer Graphischen oberfläche tätigen. Noch dazu kann man PL/SQL Prozeduren erstellen oder generieren lassen. Der Oracle Data Modeler wird hauptsächlich zum Erstellen von Datenbankmodellen oder Entity-Relationship-Modellen verwendet. Zum Schluss hat man dann die Möglichkeit ein Skript zu generieren welches dann alle Tabellen erstellt und auch die dazu modellierten Beziehungen erstellt.

3.16 SQL

SQL steht für Structured Query Language und ist eine Datenbanksprache zur Erstellung von Datenbankstrukturen in relationalen Datenbanken aber auch zum Bearbeiten und Abfragen von Daten. Die Datenbanksprache basiert auf der relationalen Algebra. Die Syntax ist einfach und an die englische Sprach angelehnt.

Mit abfragen werden Daten in einer Datenbank abgerufen. [?]

3.16.1 SQL - Select

Der Select Befehl in der Datenbanksprache, ist sozusagen der Grundstein für SQL-Abfragen.

Listing 4: Sql Select

```
1  SELECT t.Spaltenname1, t.Spaltenname2 FROM Tabellenname t
```

Wenn man den Select Befehl mit einem "WHERE" erweiter, kann man die Datenbank abfrage umso genauer gestallten. Es werden sommit nur sezielle Daten abgefragt.

Listing 5: Sql Select Where

```
1  SELECT t.Spaltenname1, t.Spaltenname2 FROM Tabellenname t WHERE t.Spaltenname1 = 1
```

Die wichtigsten SQL-Comands lauten:

- SELECT - extrahiert Daten aus der Datenbank
- UPDATE - aktualisiert Daten in der Datenbank
- DELETE - löscht Daten aus einer Datenbank
- INSERT INTO - fügt neue Daten in die Datenbank ein
- CREATE DATABASE - erstellt eine neue Datenbank
- ALTER DATABASE - modifizert eine neue Datenbank
- CREATE TABLE - erstelt eine neue Tabelle
- ALTER TABLE - modifiziert eine Tabelle
- DROP TABLE - löscht eine Tabelle
- CREATE INDEX - erstellt einen Index
- DROP INDEX - löscht den Index

3.17 Vue.js

Vue.js ist ein JavaScript-Webframework, das zum Erstellen von Single-Page-Webanwendungen dient. Es wurde von einem kleinen Team im Jahre 2014 entwickelt mit dem ursprünglichem Autor Evan You.
Vue ist relativ neu und die große Stärke von Vue ist die einfache Lernkurve, die Vielseitigkeit und die Leichtgewichtigkeit. Man benötigt Kenntnisse in JavaScript, HTML, CSS und schon kann man loslegen mit deren ausführlich dokumentierten Guide[**?**]. [**?**] [**?**]

3.17.1 Vue Funktionsweise

Model-View-Viewmodel (MVVM) Pattern

Vue.js benutzt das MVVM Pattern. Das Pattern trennt die Darstellung von der Logik der Benutzer-UI's. Dazu ist ein Datenbindungsmechanismus vorausgesetzt. Dadurch können sich Entwickler und Interfacedesigner trennen und ihre Aufgaben im Projekt

aufteilen.

Dieses Pattern wurde 2005 von John Gossman veröffentlicht. [?]

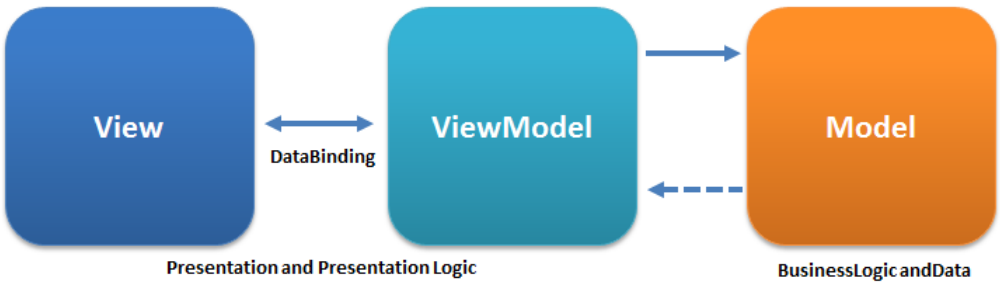

Abbildung 8: Darstellung von Branches in einem Repository
https://upload.wikimedia.org/wikipedia/commons/8/87/MVVMPattern.png

- **View:** Enthält alle Elemente die durch die Benutzeroberfläche angezeigt werden. Es bindet sich an das ViewModel, welches die Eigenschaften der View bestimmt.
- **ViewModel:** Es enthält die Logik des UI's. Es tauscht sich mit dem Model aus und benützt seine Methoden und Dienste. Gleichzeitig gibt es der View Eigenschaften, die dem Model entsprechen. Es bindet Daten mit der View und sich selbst (DataBinding).
- **Model:** Diese Schicht enthält alle Daten die der Benutzer manipuliert oder aufruft. Es enthält die gesamte Geschäftslogik.[?]

Vue Instance

Jede Vue Applikation beginnt mit der Erstellung einer Vue Instanz.

Listing 6: Vue Instanz

```
1       var vm = new Vue({
2           // options
3       })
```

Die Variable vm steht für ViewModel, was unsere Vue Instanz darstellt. Man kann jeder Instanz Optionen zuweisen, um sie zu konfigurieren. Diese Instanz wird auch als Root Instanz bezeichnet und bildet den Stamm eines Baumes mit Komponenten.

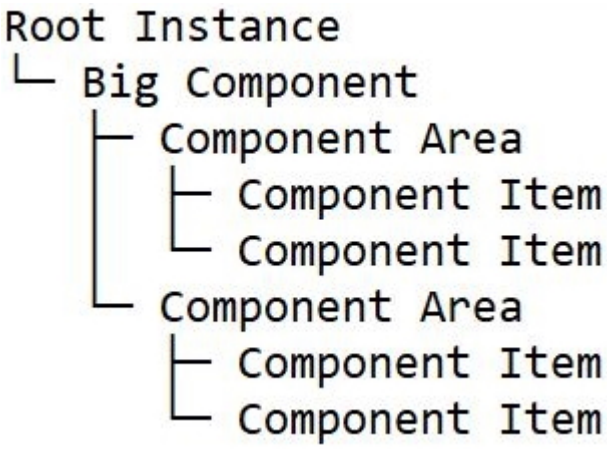

Abbildung 9: Der Stammbaum einer Root Instanz

Zu einer Instanz gehört auch der data-Bereich. Dieser beherbergt alle Properties einer Instanz und diese Properties reagieren auf Veränderungen im Code. Noch dazu kann jede Instanz Methoden haben.
Jede Instanz hat auch seine Lifecycle Hooks, dies sind Methoden, die zu bestimmten Zeitpunkten einer Instanz ausgeführt werden. [**?**] Diese sind:

- **created**
- **mounted**
- **updated**
- **destroyed**

Lifecycle Diagram

Das Diagramm hier stellt den Ablauf einer Erstellung einer neuen Vue Instanz dar.

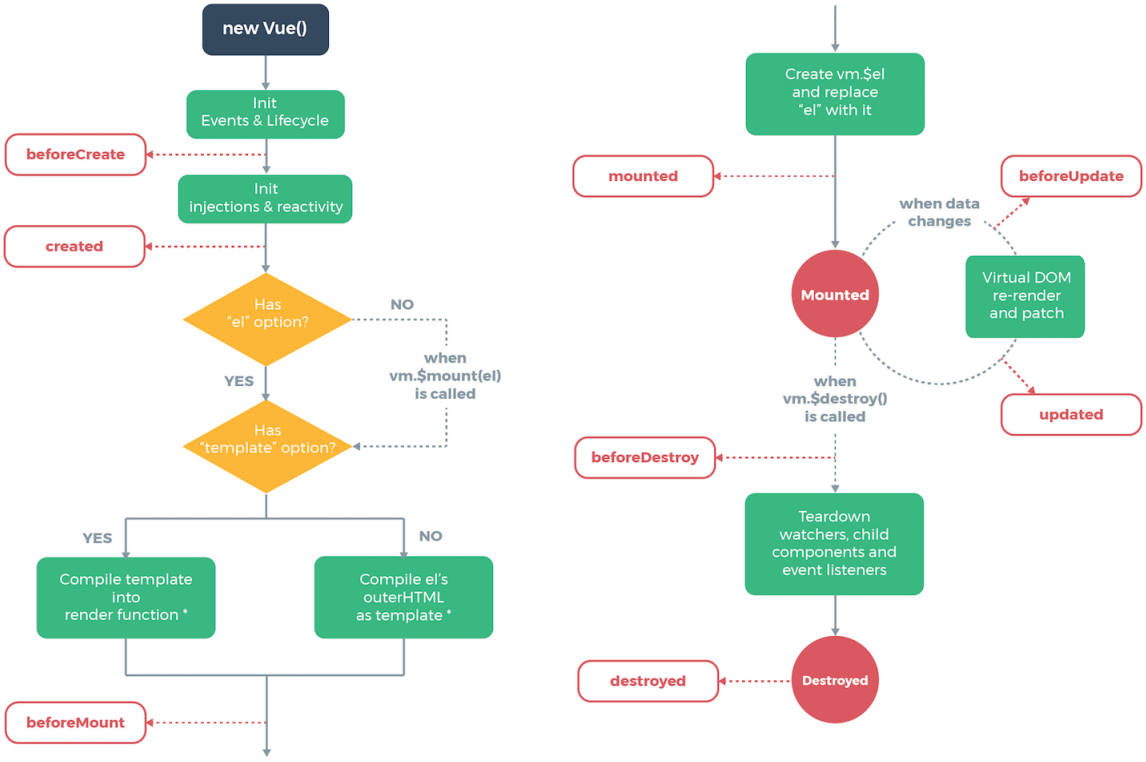

Abbildung 10: Diagramm des Ablaufs einer Vue Instanz
https://www.oreilly.com/library/view/full-stack-vuejs-2/9781788299589/assets/9f308e86-bbbe-489c-9f93-06abe2675081.png

Vue Components

Komponenten sind wiederverwendbare Vue Instanzen mit einem eigenen Namen. Diese Komponenten können mittels HTML-Tags in anderen Komponenten verwendet werden. Eine Vue.js Seite ist meistens in mehrere Komponenten aufgeteilt, um größere Bereiche auf der Seite zu trennen und übersichtlicher zu gestalten. Diese können miteinander kommunizieren und Daten austauschen, um z.B. Daten, die ständig verändert werden ordentlich darzustellen.[?]

Die folgende Grafik veranschaulicht den Component Tree, der zeigt wie die Single-Page Anwendung umgesetzt ist. Die grünen Boxen zeigen die miteinander vernetzten Komponenten, jeder Komponent kann mehrere Unterkomponenten haben, je nach Einteilung der Webseite.

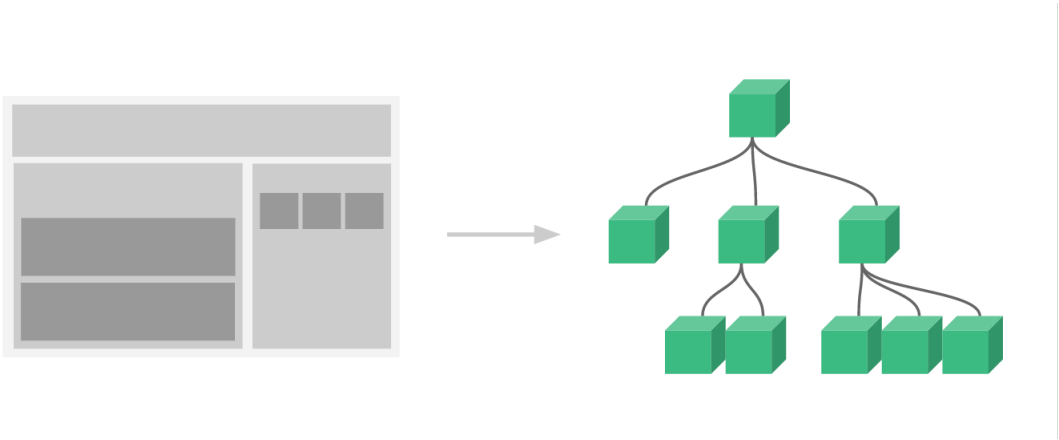

Abbildung 11: Darstellung von vernetzten Komponenten innerhalb einer Webpage
https://vuejs.org/images/components.png?_sw-precache=b5c08269dfc26ae6d7db3801e9efd296

3.17.2 Angular vs. Vue

Marktstatistik

Angular:

- Angular wird eher für Seiten benutzt mit hohen Aufrufzahlen
- Nachdem was bekannt ist benutzen unter 0.4% aller Webseiten Angular
- Angular wird von 16.1% Entwicklern weltweit verwendet [**?**]

Vue:

- Es gibt mehr als 1.523.449 Millionen Webseiten, die Vue benutzen
- Der Marktanteil von Vue beträgt nicht mehr als 0.5% [**?**]

Vor- und Nachteile

Von der Lernkurve her ist Vue deutlich im Vorteil, da es einfacher zu verstehen ist als Angular. Angular benötigt viel Einarbeitungszeit, bis man die Funktionsweise verstanden hat. Angular ist eher für umfangreiche Projekte gedacht, währenddessen Vue auf geringe Größe und hohe Performance abzielt.
In Angular sind die Logik und das Aussehen strikt getrennt, währenddessen in Vue alles in einem File zu finden ist und mehr an HTML erinnert durch die Scripts, die man schreibt. Ein Vorteil von Angular ist die Implementierung von Typescript, was eine Weiterentwicklung von JavaScript ist, die versucht die Macken von JavaScript zu verbessern.
Was noch zu erwägen ist ist, dass Angular weiter verbreitet ist als Vue und es oft

Features oder Plugins gibt, die in Angular selbstverständlich sind, aber in Vue nicht aufzufinden sind. Dies sollte sich aber im Laufe der Zeit verbessern.
Generell kann man sagen, dass das Programmieren mit Angular eher and die Programmierung mit Java erinnert mit den Objekten, Abhängigkeiten, Konstruktoren, usw. Während Vue an das Programmieren von Websites mittels HTML und JavaScript erinnert. [?]

Warum Vue?

Unsere Entscheidung Vue zu nehmen ist einerseits von der Firma beeinflusst worden, da sie es vorgeschlagen haben und es bei ihnen das gängige Framework ist.
Andere Faktoren waren noch, dass wir Angular in der Schule oft verwendet haben und wollten durch Vue eine andere Methode ausprobieren, um Webanwendungen zu erstellen. Vue ist die bessere Wahl für den Umfang unserer Webanwendung gewesen und die bessere Performance in Vue ist wichtig, damit die Bestellungen reibungslos ablaufen können im Echtbetrieb.

3.18 HTML

HTML genannt HyperText Markup Language, ist eine einheitliche, textbasierte, Auszeichnungssprache für Webdokumente. HTML definiert ganz allgemein gesehen die Struktur eines Dokuments. Am 13.März 1989 wurde am CERN in Genf das Konzept HTML von Tim Berners-Lee vorgeschlagen, um eine einheitliche Methode zu finden Dokumente öffentlich zu übermitteln. Seitdem ist HTML einer Grundbausteine des World Wide Webs geworden.
HTML basiert auf sogenannten Tags, die verschiedene Inhalte auf der Website definieren sollten. Diese basieren auf einer bestimmten Syntax, um das Schreiben zu vereinheitlichen.
Es ist im Grunde eigentlich keine Programmiersprache, sondern eine statische Sprache, die zur Definierung benutzt wird. Das HTML Gerüst wird von einem Browser eingelesen und dieser generiert dann auf der Basis des HTML Files eine Website. [?] [?]

Listing 7: HTML File Grundgerüst

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title> Titel der Webseite </title>
5     </head>
6     <body>
```

```
7           <h1> Ueberschrift </h1>
8       </body>
9   </html>
```

Wie man im Beispiel sieht gibt bei Tags ein Beginn und ein Ende und der Inhalt dazwischen wird dargestellt. Es gibt auch Tags ohne Beginn und Ende, sogenannte inhaltslose Tags. Doch HTML wird meist nie allein verwendet, erst in der Kombination mit CSS, Bootstrap und JavaScript kann man eine gute Website erstellen.

3.19 CSS

CSS (Cascading Style Sheets) ist die Sprache, die benutzt wird eine HTML Seite visuell zu gestalten. CSS ist wie HTML keine Programmiersprache, sie wurde dafür entwickelt um das Aussehen von HTML Seiten einheitlich zu verändern.
Eine CSS-Datei wird durch einen Tag mit dem zugehörigen HTML Dokument verbunden und dadurch werden die Änderungen angewendet. [?]

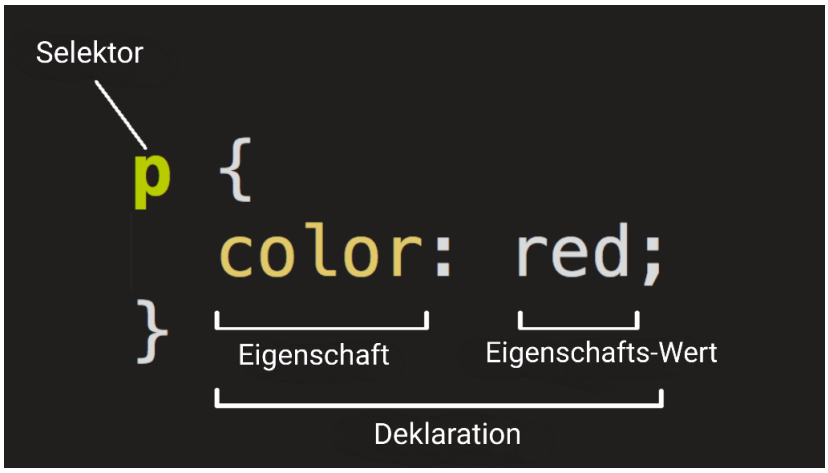

Abbildung 12: Aufbau einer CSS-Regel
https://media.prod.mdn.mozit.cloud/attachments/2017/09/27/15467/3889d04d90c10b27e863c6850d588c43/css-example.png

Die Struktur von CSS-Dateien wird durch Regeln beschrieben. Jede Regel hat einen Selektor, der auf ein zugehöriges HTML Tag zugreift. Innerhalb der Deklaration wird dann der Wert einer bestimmten Eigenschaft gesetzt (Mehrere Eigenschaften sind möglich). Es gibt eine große Auswahl von Eigenschaften wie z.B.: Schriftfarbe, Textart, Positionierung, etc. [?]

3.20 JavaScript

JavaScript ist eine leichtgewichtige Skriptsprache, die 1995 vom Softwareunternehmen Netscape entwickelt geworden ist, um die Möglichkeiten von HTML und CSS zu erweitern. Bekannt ist sie hauptsächlich als Sprache für Webseiten geworden, jedoch wird sie auch in vielen Umgebungen außerhalb des Browsers oft benutzt, wie z.B. Servern.
Sie unterstützt objektorientierte, imperative als auch deklarative Programmierung. JavaScript folgt dem Standard ECMAScript, welche alle Browser unterstützen. JavaScript dient dazu auf Webseiten Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren. Sie bietet normalen HTML Webseiten eine Großzahl an Verbesserung, durch Hinzufügen von Programmierelementen.
Jedoch sollte man JavaScript nicht mit der Programmiersprache Java verwechseln. Beide sind verschiedene Handelsmarken der Firma Oracle und ähneln einander höchstens mit der Syntax. [**?**] [**?**]

3.21 JSON Web Token (JWT)

JSON Web Token ist ein nach RFC 7519 genormter Standard, um Daten sicher zwischen zwei Parteien auszutauschen. Es wird in der Form eines JSON-Objektes übertragen. Die Information, die der Token enthält kann verifiziert werden, weil es eine digitale Unterschrift enthält. Der Token selbst kann entweder mit einem geheimen Schlüssel (HMAC-Algorithmus) oder einem private/public Schlüsselpaar (RSA oder ECDSA) verschlüsselt werden.
Beliebte Anwendungsfälle des Tokens sind Authentifizierung und Datenaustausch. Der Token selbst enthält Informationen über den Absender und ob er die nötigen Zugriffsrechte hat. [**?**] [**?**]

3.21.1 Aufbau eines JWT

Ein signierter JWT besteht aus 3 Teilen, getrennt durch einen Punkt. Jeder dieser Teile wird mit Base64 kodiert.
Jeder JWT hat auch eine Gültigkeitsdauer, wenn diese abgelaufen ist, gilt ein Token als ungültig.

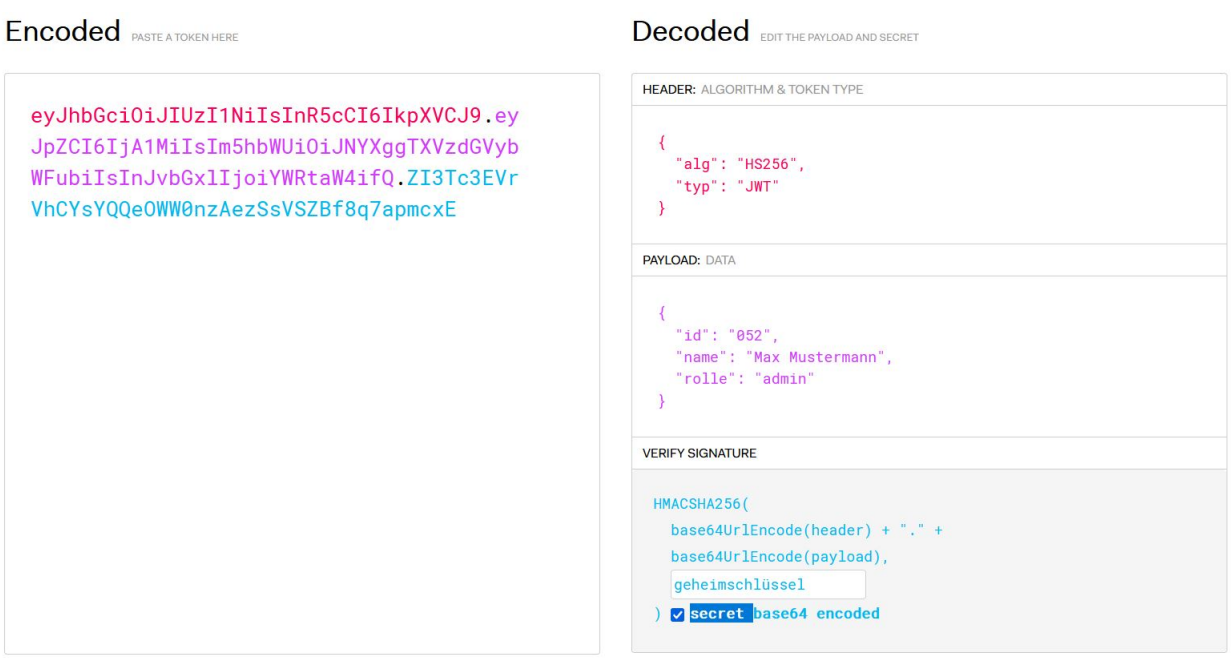

Abbildung 13: Aufbau eines JSON Web Tokens (Ausschnitt von jwt.io)
https://jwt.io/

Header

Der Header besteht meist aus zwei Teilen und liefert Informationen über den Typ des Tokens und den verwendeten Signatur- bzw. Verschlüsselungsalgorithmus.

- Der "typ"-Wert beschreibt den IANA Medientypen des Tokens, oft wird "application/jwt" verwendet.
- Der "alg"-Wert gibt an welcher Algorithmus zum Signieren des Tokens verwendet wurde. Es ist möglich auch keine Verschlüsselung anzugeben, was jedoch nicht zu empfehlen ist. [?]

Payload

Der Payload ist der Teil des Tokens, der die tatsächlichen Daten bzw. Informationen enthält, die übermittelt werden sollen. Sie werden als Key-/Value-Paare bereitgestellt, diese Werte werden bei JWT als Claims bezeichnet. [?] Davon gibt es drei verschiedene Arten:

- **Registrierte Claims** sind standardisiert und im JWT Claim Register festgelegt. Es wird empfohlen diese zu verwenden.
- **Öffentliche Claims** sind nach belieben definierbar.

- **Private Claims** sind für Informationen gedacht, die speziell auf unsere Anwendung angepasst sind wie z.B. "Benutzer-ID". [**?**]

Signature

Diese wird durch Base64-Kodierung des Headers, des Payloads und der angegebenen Signaturmethode erzeugt. Der Aufbau ist definiert nach dem RFC 7515 Standard, auch genannt JWS (JSON Web Signature). Damit die Signatur funktioniert, muss man einen geheimen Schlüssel verwenden, der nur dem Ursprung bekannt ist. [**?**]

3.21.2 Sicherungsverfahren

Keine Sicherung

Wenn die Daten keiner Verschlüsselung bedürfen, kann im Header "none" angegeben werden. In diesem Fall wird keine Signatur generiert, dadurch fällt auch der Signature-Teil weg.
Ohne Sicherung lässt sich die Nachricht nach einer Base64-Entschlüsselung klar und deutlich lesen. Der Absender oder ob die Nachricht im Laufe verändert worden ist, ist nicht mehr verifizierbar.[**?**]

Signatur (JWS)

Im Normalfall reicht es nur zu prüfen, ob die Daten vom richtigen Absender kommen und ob Veränderungen geschehen sind. Da kommt die JWS (JSON Web Signature) zum Einsatz, die genau die vorher genannten Sachen überprüft.
Bei diesem Verfahren lässt sich die Payload nach Base64-Entschlüsselung klar und deutlich lesen. [**?**]

Signatur (JWS) und Verschlüsselung (JWE)

Es ist möglich zusätzlich zum JWS noch eine JWE (JSON Web Encryption) zu benützen. JWE verschlüsselt den Inhalt des Payloads, diese werden danach mit JWS signiert. Um die Inhalte dann zu entschlüsseln wird noch ein Kennwort oder ein privater Schlüssel angegeben.
Damit ist der Absender verifiziert, die Nachricht authentisch und der Payload ist nicht lesbar nach einer Base64-Entschlüsselung. [**?**]

3.22 Progressive Web App(PWA)

Webanwendung werden mit Hilfe von HTML, CSS und Javascript wntwickelt. Das in Verbindung mit dem Progressive Enhancement, was für die Lauffähigkeit einer Webseite in jedem Browser verantwortlich ist, wird eine PWA genannt.

Der Service-Worker arbeitet mit HTTPS und führt konstant einen Web-Browser im Hintergrund. Dank ihm hat die PWA Zugriff auf Caching und kann Offline verwendet werden. Weiter noch stellt der Service-Workerr die Funktionalität der Push-Notification bereit.

Vorteile sind die Kostenreduktion in der Entwicklung da man nur eine PWA benötigt, die man auf Android, IOS und Windows benützen kann. Weiters kann man die PWA so konfigurieren, dass sie wie eine echte Applikation ausschaut.

3.23 Google Charts

3.24 Jetpack Compose

Jetpack Compose ist seit Juli 2021 in der ersten stabilen Version verfügbar. Es ist ein "Werkzeugkasten" zum Bauen von Android Applikationen. Es vereinfacht und beschleunigt den Anwendungsentwicklungsprozess. Da man mit wenig Code schnell viele Ansichten erstellen kann, ist die Fehlerqoute dementsprechend niedrig. Jetpack Compose basiert 100 Prozent auf Kotlin. Man kann zwischen drei Ansichten auswählen, die da Code, Split oder Design wären die das parallele arbeiten sehr vereinfacht. Das App wird aber nicht nach jeder änderung neu gebaut, was wiederum die Schnelligkeit des Arbeiten beeinflusst. Damit die Applikation eine schöne Oberfläche hat, bietet Jetpack Compose mehrere Themes zu Verfügung. Noch dazu kann man diese dann auch selbst konfigurieren indem man Items hinzufügt oder löscht. Jetbrains hat in Planung eine plattformübergreifendes Open-source Projekt zu starten, damit man Desktop Anwendungen genau so schnell und einfach erstellen kann.

3.25 Kotlin

Kotlin ist eine universelle und statisch typisierte Open-Source-Programmiersprache, die ursprünglich für die Java Virtual Machine und Android entwickelt wurde. Es konzentriert

sich auf Interoperabilität, Sicherheit, Übersichtlichkeit und Werkzeugunterstützung. Als build-script wird Gradle verwendet.

2017 wurde sie zu einer offiziellen Sprache zur Entwicklung von Android-Applikationen. Deswegen wird Kotlin auch Programmiersprache der Zukunft betitelt wenn man sie mit Java vergleicht. Lambdafunktionen werden deutlich reduziert und somit auch leichter in der Praxis als in Java. Da Kotlin sehr gut mit Java arbeiten kann, können beim Testen die Frameworks und Bibliotheken von Java sehr weiterhelfen. Ursprünglich hat man den Kotlin Code in Bytecode übersetzt und diesen dann in edr JVM laufen gelassen. Aber den Code kann man aber auch in Javascript umschreiben lassen und somit für Web Anwendungen verwenden.

3.26 XML

XML steht für eXtensible Markup Language und ist ein textformbasiertes Datenformat. Das Wort eXtensible beschreibt die erweiterungsmöglichkeiten der Sprache. Sie wurde im Jahre 1996 entwickelt und wurde zwe Jahre später zum W3C-Standart. Sie ist sehr leicht einsetzbar in verschiedene Anwendungen da sie keine Lizenz benötigt. Viele Tools heutzutage erleichtern aber auch die bearbeitung von XML-Dateien. Ein Vorteil ist, dass man die Daten im XML als Textformat abspeichert, was heisßt das XML auch plattformunabhängig ist. Wie in HTML, gibt es auch hier sogenannte Tags, die aber selbst zu kofigurieren sind. Der große Vorteil von XML ist das es sehr leicht lesbar ist, sowohl von Mensch als auch von Machine. Es wird in Android verwendet um das Layout jeder Aktivität des Android zu entwerfen und erleichtert die Datenübertragung zwischen Anwendungen.

3.27 Docker

Docker ist eine in GO programmierte Softwareplattform, die den Prozess des Erstellens, Ausführens und Verwaltens von Apps umfasst. Dies geschieht durch Virtualisierung des Betriebssystems, auf dem es installiert ist und ausgeführt wird.

3.27.1 Docker Compose

3.28 OKHttp

OkHttp ist ein HTTP-Client, der standardmäßig effizient ist. Dank der HTTP/2-Unterstützung ist es möglich alle Anfragen an einen Host zu schicken. Response Caching hilft bei vermeidung von wiederholten Anfragen. OkHttp hilft bei Verbindungsproblemen indem man alternative Adressen benutzt und unterstürzt weiters noch TLS-Funktionen. Es kann so konfiguriert werden, dass es auf eine breite Konnektivität zurückgreift. Der Client ist sehr gut für die Verwendung von blockierte synchrone und nicht blockierte asynchrone Aufrufe

3.29 Gradle

Gradle ist ein, in 2007 entwickeltes, Build-Management-Automatisierungs-Tool welches auf Java basiert. Man kann es mit Apache Maven und Ant vergleichen. Es hat die Flexibilität von Apache Ant und die simple Bediunungsmöglichkeit von Maven. Der unterschied zu den Maven-Projektdefinitionen ist, dass man Gradle-Scripts gleich ausführen kann. Gradle unterstütz neben Java noch die Programmiersprachen Groovy und Kotlin. DAG wird verwendetr um die Tasks in richtiger Reihenfolge zu dirigieren. Heutzutage hat Gradle eine eigene Maschiene für das verwalten der einzelnen Abhängigkeiten, aber es begann mit Apache Ivy. Seit Mitte 2013 ist Android hinzugekommen. Seitdem wurde das Tool weitgehend erweitert, um den Aufbau sogenannter „nativer“ Systeme zu unterstützen, welche nicht auf Java basieren. Gradle wird von sehr großen Unternehmen verwendet wie zum Beispiel LinkedIn, Netflix, Adobe und vielen weiteren.

3.30 Retrofit

Über einen REST-basierten Webdienst wird es sehr einfach Daten hochzuladen oder auch abrufen. Retrofit ist ein REST-Client für Java und Android. Es wird verwendet, um Requests zu Empfangen, Senden und Erstellen von HTTP-Anforderungen sowie Antworten. Noch dazu wird die IP-Adressen ausgewechselt, wenn eine Verbindung zu einem Webdienstfehler besteht.

3.31 Asciidoc

AsciiDoc ist ein Textformat, welcher hauptsächlich für Dokumentationen, Webpages, Blog und vieles mehr verwendet mit. Da AsciiDoc ein plain-text Format ist und nicht wie die meisten Anwendungen für Textverarbeitungen Binärformate sind, wird es auch als Docs-as-Code bezeichnet.

AsciiDoc wurde entwickelt, um Dokument so zu schreiben als währen sie normale Textdokumente. Da das Design von AsciiDoc schon inkludiert ist, muss man nicht einstellen wie die einzelnen Schriftgrößen für Überschriften sind. Somit ist man mit AsciDoc viel schneller als mit einer Anwendung, welche in Binärform ist.

3.32 Asciidoctor

Asciidoctor ist eine open source Textverarbeitung, welche AsciiDoc Text in Formate wie HTML 5, PDF und weiter, umwandelt.

Asciidoctor selbst ist in Ruby entwickelt worden. Um aber Asciidoctor zu verwenden braucht man kein Ruby. Mittels AsciidoctorJ kann man ganz einfach Asciidoctor auf einer JVM ausführen.

3.33 Swagger

Swagger ist ein Open-Source Werrkzeug welches verwendet wird um HTTP-Webservices zu entwerfen, dokumentierten aber auch zu benutzen. Um einen leichten überblick über seine ganzen Endpoints zu haben, wird Swagger verwendet, da es ganz schnell und einfach zum verwenden ist.

Damit man Swagger in einem Quarkus Backend verwedet, muss mann in den Application.Properties folgende Zeile eintragen:

quarkus.swagger-ui.always-include=true

Die Sagger Page wird dann unter folgender URL abgerufen:

http://localhost:8080/q/swagger-ui/

Wie schon erwähnt, kann man bei der Swagger Page alle Requests sehen die verwendet werden.

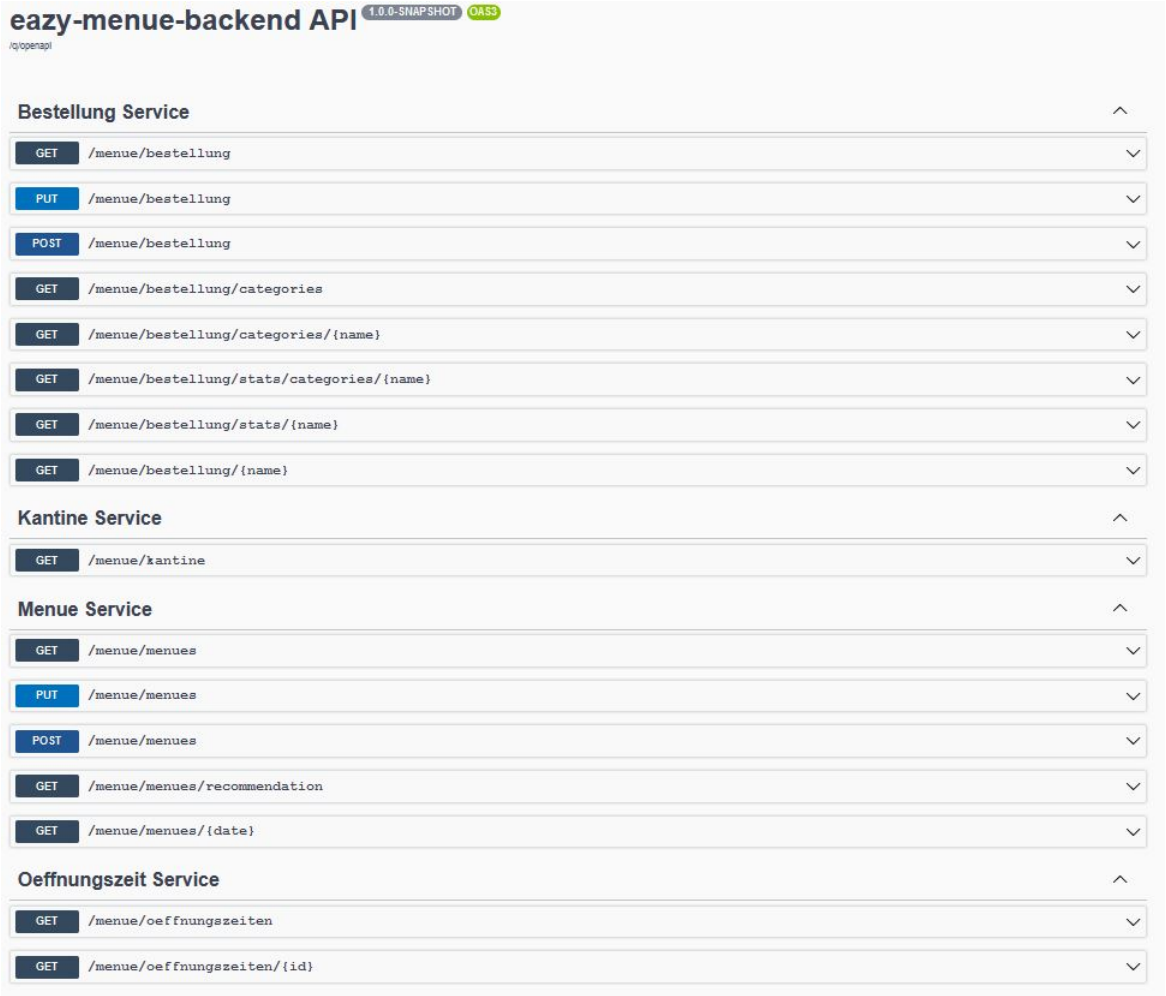

Abbildung 14: SwaggerUI

Noch dazu kann man mit Swagger auch einzelne Data Transfer Objects veranschaulichen, welche in den Requests verwendet wird.

Abbildung 15: SwaggerUI

4 Implementierung

4.1 Systemarchitektur

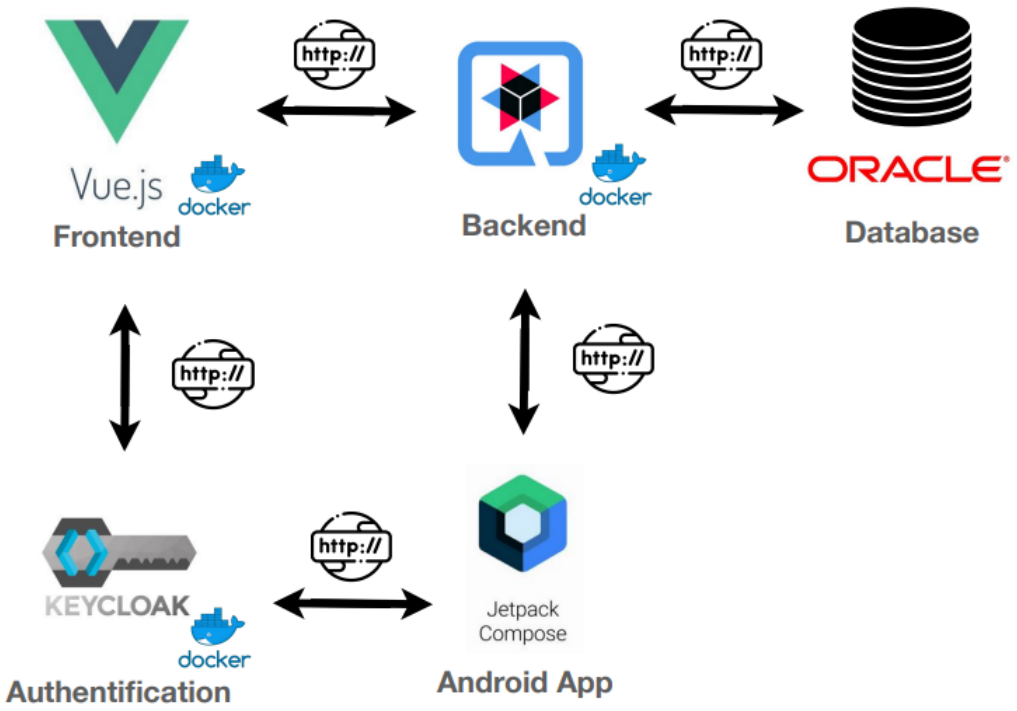

Abbildung 16: Systemarchitektur des Programms

Der Systemarchitektur kann man entnehmen, dass das Quarkus Backend die zentrale Stelle für alle anderen Technologien bildet. Das Backend stellt mit Hilfe der Datenbank alle Daten zur Verfügung mittels REST.
Der Keycloak Server sichert die beiden Frontends ab und sorgt dafür, dass keine ungewollten Zugriffe entstehen. Vue.js benützt dafür ein Keycloak Package, dass eine leichte Integration erlaubt. Die Android App hingegen arbeitet mittels REST Requests mit Keycloak zusammen, da es zum jetzigen Zeitpunkt keine offizielle Erweiterung für Jetpack Compose gibt.

4.1.1 Technologien

Beim Entwickeln wurden folgende Technologien verwendet:

- docker 3.1
- Vue.js 2.6.14
- quarkus 2.5.0.Final
- Jetpack Compose 1.0.1
- Keycloak 14.0.0
- Java OpenJDK-11
- Java EE 8
- JBoss Wildfly 7.3.4.GA

4.2 Datenmodell

Ein Datenmodell wird als Darstellung der relevanten Objekte eines Projektes verwendet. Unser Datenmodell ist im Großen und Ganzen immer gleichgeblieben. Für den Algorithmus aber haben wir es aber erweitern müssen.

4.2.1 Planung

Einer der ersten Arbeitsschritte war die Entwicklung eines Datenmodells, welches die Basis der Programmlogik sein soll. Dieses wurde mit einem ERD-Diagramm erstellt.

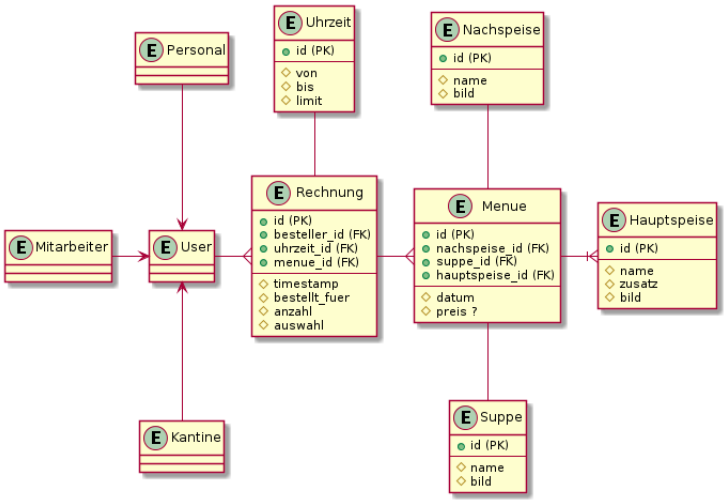

Abbildung 17: Erste Version des Datenmodells

Die erste Version wurde von unserem Team entwickelt, aus den Erfahrungen und Informationen, die wir im Unterricht gesammelt haben.
An dieser Version merkt man, dass die meisten Teile in einzelne Tabellen aufgeteilt wurden. Dies sorgt für Flexibilität und Wiederverwendung von einem Menü, da die einzelnen Speisen aufgeteilt sind. Am Anfang war geplant, dass jede Speise auch ein Bild hat, damit der Mitarbeiter weiß, wie die Speise auch wirklich aussieht.
Ebenfalls erkennt man, dass es eine User-Klasse gibt, da Mitarbeiter, Personal und Kantine gleiche Eigenschaften miteinander teilen wie zum Beispiel Vor- und Nachname.

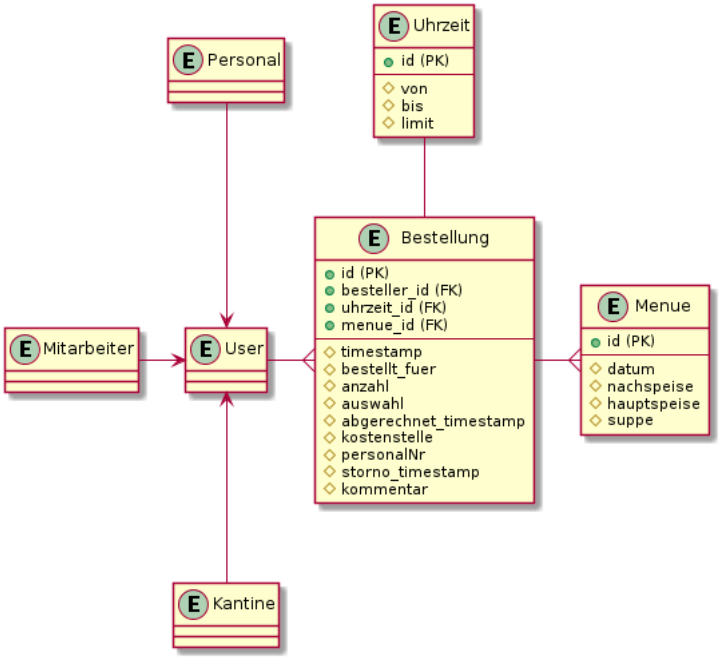

Abbildung 18: Finale Version des Datenmodells

Die finale Version des Datenmodells übernimmt das meiste der ersten Version, jedoch wurde es nach Absprache mit unserer Partnerfirma überarbeitet, um deren Anforderungen mehr zu entsprechen.
Eine wichtige Änderung ist, dass die Menue-Klasse alle Speisen enthält und diese als einfacher Text gespeichert werden. Der Grund dafür ist, dass die Kantine immer die Menüs händisch eingibt und eine Auswahl von vorhandenen Speisen würde nur den Speicheraufwand unnötig erhöhen. Es kommt eher selten vor, dass die selben Speisen nacheinander kommen.
Was vorher gefehlt hat war der Kommentar und die Möglichkeit die Stornierung eines Menüs nachzuvollziehen. Wenn ein Menü storniert wird, wird es nicht gelöscht, sondern es wird nur das Stornodatum gesetzt. Es ist wichtig, dass auch stornierte Bestellungen

in der Datenbank bleiben.
Ebenfalls wurde die Abrechnung eines Menüs in der ersten Version nicht berücksichtigt. Die Kosten für das Menü werden dem Mitarbeiter vom Gehalt abgezogen und dies erfolgt anhand von den Daten in der Datenbank.

4.2.2 Entitäten

Eine Entität ist ein bestimmtes Objekt mit den jeweiligen Atributen. Atribute sind die Eigenschaften eins Objektes. Unser Projektes beinhaltet 5 Entitäten:

- Bestellung
- Categories
- Kantine
- Menue
- Oeffungszeit

4.2.3 Bestellung

Die Entität **Bestellung** enthält die wichtigsten Informationen über einen Mitarbeiter und die dazu ausgewählte Mahlzeit. Zur Identifikation einer Bestellung verwenden wir eine Id welche generiert wird.

In der Bestellung wird angegeben von wem die Bestellung bestellt wurde und ob der Mitarbeiter es auch für sich selbst bestellt hat oder für jemand anderen. Jeder Mitarbeiter der Firma hat auch eine Personalnummer. Bei jeder Bestellung gibt es die Möglichkeit einen Kommentar abzugeben. Dieser wird an die Kantine mitgegeben. Natürlich hat man auch die Möglichkeit eine Mahlzeit öfters zu bestellen. Somit hat jeder Mitarbeiter die Möglichkeit eine Bestellanzahl mitzugeben. Für das Bestellen, bearbeiten aber auch das Stornieren wird immer die jetzige Uhrzeit mitgegeben.

Das wichtigste in der Bestellung ist die Mahlzeit und für wann es bestellt wurde. Dafür verwenden wir die zwei Klassen, Oeffnungszeit und Menue.

```
    private Long id;

    private String orderedBy;

    private String orderedFor;

    private String comment;

    private Long personalNumber;

    private int menueCounter;

    private Timestamp canceledAt;

    private Timestamp settledAt;

    private Integer moneyPool;

    private Timestamp createdAt;

    private String createdBy;

    private Timestamp changedAt;

    private String changedBy;

    private Oeffnungszeit oeffnungszeit;

    private Menue menue;
```

4.2.4 Categories

Die Entität **Categories** ist ein Enum, welches wir Hauptsächlich für unseren Algorithmus verwenden. Es besteht aus 4 Einträgen:

- Vegetarisch
- Vegan
- Schwein
- Rind
- Huhn
- Pute
- Salat
- Nudel
- Süß
- Fisch
- Sonstiges

Die einzelnen Kategorien beschreiben eine Mahlezeit. Sommit kann man ganz einfach zwischen einzelnen Mahlezeiten unterscheiden und sie auch gruppieren.

4.2.5 Kantine

In der Entitäten **Kantine** befinden sich die wichtigsten Informationen über eine Kantine wie zum Beispiel ob eine Kantine offen hat oder zurzeit geschlossen ist. Eine kleine Beschreibung über die Kantine und über das Service ist ebenso enthalten.

```
    private Long id;

    private String canteenDesc;

    private String serviceDesc;

    private char status;

    private Timestamp createdAt;

    private String createdBy;

   private Timestamp changedAt;

    private String changedBy;
```

4.2.6 Menue

In der Entität **Menue** stehen die wichtigsten Informationen über ein Menü. Ein Menü beinhaltet eine Vorspeise, Hauptspeise, Nachspeise und ein Dessert. Noch dazu findet man in der Entität ein Datum, um zuzuordnen wann das Menü bestellt worden ist.

Neben dem Bestelldatum beinhaltet die Entität Menue auch noch weiter Timestamps. Die verwendeten Timestamps werden verwendet, um zu speichern, wann ein Menü erstellt wurde oder wann ein Menü geändert wurde. Natürlich wird auch mitgespeichert, von wem diese Änderungen durchgeführt worden sind.

Die bereits oben genannten Kategorien finden wir in der Entität **Menue**. Diese werden verwendet, um einzelne Menüs zu unterscheiden und mit Hilfe der Kategorien und unserem Algorithmus kann man einfach hervorwagen welche Mahlzeit zu welchem Mitarbeiter am besten passt.

Für jedes Menü wird auch die dazu ausgewählte Kantine mitgespeichert. So weiß man, zu welcher Kantine der Mitarbeiter gehen muss, um sein Menü zu bekommen.

```
private Long id;
private LocalDate date;
private char code;
private String appetizer;
private String mainDish;
private String desert;
private Timestamp createdAt;
private String createdBy;
private Timestamp changedAt;
private String changedBy;
private String categories;
private Kantine kantine;
```

4.2.7 Oeffnungszeit

Die Entitäten **Oeffnungszeit** zeigt an, ob ein Kantinenraum in Verwendung ist und ob die Maximale Anzahl an Sitzplätzen schon belegt ist. Das Zeitfenster welches beschreibt wann gegessen wird, wird ebenso mitgespeichert.

Um zu wissen welcher Kantinenraum gemeint wird, wird auch die verwendete Kantine mitgespeichert.

```
private Long id;
private char status;
private String timeWindowFrom;
private String timeWindowTo;
private Integer maxPositions;
private Timestamp createdAt;
private String createdBy;
private Timestamp changedAt;
private String changedBy;
private Kantine kantine;
```

4.3 REST-Schnittstellen

4.3.1 Allgemein

In den folgenden Beispielen läuft unser Server lokal auf den Port 8080. Die URL uber die unser Server erreicht werden kann, lautet: ***http://localhost:8080/menue***.

4.3.2 Bestellung

Bestellungen von einem Mitarbeiter

Um zu sehen welche Bestellungen ein Mitarbeiter gemacht hat, muss der Pfad ***/bestellung/<username>*** aufgerufen werden. Hierbei handelt es sich um eine GET-Methode die mittels Path parameter die Bestellungen eines bestimmten Mitarbeiters zurück gibt. Wenn ein Mitarbeiter gefunden wurde und dieser auch mindestens eine Bestellung hat wird der Status 200 zurückgegeben.

Das Reslutat für den Mitarbeiter **spabo** sieht folgendermassen aus:

URL: GET `http://localhost:8080/menue/bestellung/spabo`

Output:

```
{
    "createdAt": "2021-11-26T15:33:33.62898Z[UTC]",
    "id": 1315,
    "menueDate": "2021-11-29",
    "menueName": "Spaghetti",
    "orderedFor": "spabo",
    "timeWindow": "11:15 - 11:45"
}
```

Die Bestellungen wird nach dem Datum, an dem eine Bestellungen angelegt wurde, sortiert. Somit werden immer die neusten Bestellungen gleich am Anfgang angezeigt.

Anzahl der Bestellungen pro Wochentag eines Mitarbeitern

Wenn man sehen will wie oft ein Mitarbeiter pro Wochentag ein Menue bestellt hat, muss der Pfad ***/bestellung/stats/<username>*** aufgerufen werden. Es werden alle Wochentage angezeigt an denen ein Mitarbeiter eine bestellung getätigt hat. Wenn mindestens ein Wochentag mit einer bestellung zur verfügung steht, wird der Status 200 zurückgegeben.

Das Reslutat für den Mitarbeiter **besbe** sieht folgendermassen aus:

URL: GET `http://localhost:8080/menue/bestellung/stats/besbe`

Output:

```
[
    {
    "amount": 2,
    "weekday": "Montag"
    },
    {
    "amount": 1,
    "weekday": "Mittwoch"
    },
    {
    "amount": 1,
    "weekday": "Donnerstag"
    },
    {
    "amount": 0,
    "weekday": "Dienstag"
    }
]
```

Alle Kategorien

Damit man alle Kategorien sehen kann, die eine Mahlzeit beschreiben kann, muss der Pfad ***/bestellung/categories*** aufgerufen werden. Wichtig ist es, dass man hier keinen namen als Parameter übergibt.

Anhand der Kategrien, wird mit hilfe eines Algorithmus enschieden, welche Mahlzeit am besten zu einem Mitarbeiter passt.

Die verwendbaren Kategorien sehen folgendermassen aus:

URL: GET `http://localhost:8080/menue/bestellung/categories`

Output:

```
[
    "Vegetarisch",
    "Vegan",
    "Schwein",
    "Rind",
    "Huhn",
    "Pute",
    "Salat",
    "Nudel"
    "Fisch",
    "Sonstiges"
]
```

Das wird hauptsächlich verwendet, um bei dem Frontend alle Kategorien anzuzeigen die für eine bestimmte Mahlezeit dazugehören.

Alle Kategorien von einem bestimmten Mitarbeiter

Um zu sehen welche Kategorien ein Mitarbeiter bestellt hat, muss der Pfad ***/bestellung/categories/<username>*** aufgerufen werden.

Wenn ein User keine Bestellungen getätigt hast, werden auch keine Kategorien zurückgegeben. In dem folgenden Beispiel kann man aber sehen, wie das Reslutat für den Mitarbeiter **spabo** aussieht:

URL: GET `http://localhost:8080/menue/bestellung/categories/spabo`

Output:

```
[
    "Vegetarisch",
    "Nudel",
    "Vegan",
    "Schwein",
    "Salat",
    "Pute"
]
```

Alle Bestellungen an einem bestimmten Tag

Damit die Kantine sehen kann, welche Bestellungen an welchen Tagen zugewissen worden sind, muss der Pfad ***/bestellung?date=datum*** aufgerufen werden.

Es kann aber auch sein, das es an bestimmten Tagen keine bestellungen gibt. Deshalb werden auch keine Ergebnisse zurückgegeben.

Um aber zu veranschaulichen, wie manche Bestellungen an einem bestimmten Zeitpunkt zurückgegeben werden, verwenden wir das den **16.03.2022**.

URL: GET `http://localhost:8080/menue/bestellung?date=2022-03-16`

Output:

```
[
    {
        "code": "A",
        "date": "2022-03-16",
        "menue": "Schnitzel",
        "menueCounter": 1,
        "orderedFor": "spabo",
        "personalNumber": 1023,
        "timewindow": "11:15 - 11:45"
    },
    {
```

```
        "code": "C",
        "date": "2022-03-16",
        "menue": "Eisbergsalat",
        "menueCounter": 1,
        "orderedFor": "spabo",
        "personalNumber": 1023,
        "timewindow": "12:00 - 12:30"
    },
    ...
    ]
```

Bestellung erstellen

Wenn ein User eine Bestellung bestellen will, muss der Pfad ***/menue/bestellung*** aufgerufen werden. Mithilfe der id, die man von dem vorherigen Request bekommen hat, kann man ganz einfach seine bestellung abgeben.

Ein Beispiel für eine Bestellung sieht folgendermassen aus:

```
    {
        "orderedBy":"spabo",
        "amount":"1",
        "comment":"",
        "menueId":707,
        "orderedFor":"spabo",
        "timeId":6,
        "personalNummer":1023
    }
```

Wenn die betellung erfolgreich war, wird der Status 200 zurückgegeben.

Bestellung stornieren

Um eine bestellung zu stornieren, muss der Pfad ***/menue/bestellung*** aufgerufen werden. Wie auch bei dem erstellen einer Bestellung wird mit hilfe einer id die Bestellung zugeordnet.

In unserer Datenbank wird die Bestellung nicht direkt gelöscht sondern überschrieben. Sie wird also auf gelöscht gesetzt, jedoch nicht aus der Datenbank gelöscht.

Wenn die Bestellung erfolgreich storniert wurde, wird die Antwort **Order with id <is> was cancelled!** zurückgegeben.

4.3.3 Menue

Alle Menüs

Um sehen zu können, welche Menüs es überhapt schon gab, muss der Pfad ***/menues*** aufgerufen werden.

Ein Beispiel für ein Menue sieht folgendermassen aus:

URL: GET `http://localhost:8080/menue/menues`

```
[
    {
        "appetizer": "Suppe",
        "categories": "Vegan;Vegetarisch;Salat",
        "code": "C",
        "date": "2022-03-22",
        "dessert": "Milka",
        "id": 706,
        "mainDish": "Kartofell Salat"
    },
    {
        "appetizer": "Suppe",
        "categories": "Vegan;Vegetarisch;Salat",
        "code": "C",
        "date": "2022-03-22",
        "dessert": "Milka",
        "id": 705,
        "mainDish": "Kartofell Salat"
    },
    ...
]
```

Menü erstellen

Damit die Kantine ein Menü erstellen kann, wird ein POST-Request an den Pfad ***/menue/menues*** geschickt.

Da eine Bestellung aus genau 3 Menues besteht, werden auch 3 POST-Requests abgeschickt. Die werte werden in Form eines JSON Objekt abgeschickt.

Ein Beispiel für eine Bestellung sieht folgendermassen aus:

URL: POST `http://localhost:8080/menue/menues/`

JSON Objekt für Menue A:

```
{
    "id":null,
```

```
    "date":"2022-03-22",
    "code":"A",
    "appetizer":"Suppe",
    "mainDish":"Pasta",
    "dessert":"Milka",
    "categories":"Rind;Nudel"
}
```

JSON Objekt für Menue B:

```
{
    "id":null,
    "date":"2022-03-22",
    "code":"B",
    "appetizer":"Suppe",
    "mainDish":"Lachs",
    "dessert":"Milka",
    "categories":"Fisch"
}
```

JSON Objekt für Menue C:

```
{
    "id":null,
    "date":"2022-03-22",
    "code":"C",
    "appetizer":"Suppe",
    "mainDish":"Kartofell Salat",
    "dessert":"Milka",
    "categories":"Vegan;Vegetarisch;Salat"
}
```

Wie man sehen kann, werden die Kategorien in einem String übergeben. Die Kategorien werden aber dann in getrennt und als Enum gespeichert.

Menü verändern

Wie auch bei dem erstellen eines Menüs, wird der Pfad ***/menue/menues*** aufgerufen.

Wie schon erwähnt, besteht eine Bestellung aus genau 3 Menüs, deswegen werden auch 3 PUT-Requests abgeschickt.

Ein Beispiel für eine Bestellung sieht folgendermassen aus:

URL: PUT `http://localhost:8080/menue/menues/`

JSON Objekt für Menue A:

```
{
    "id":null,
    "date":"2022-03-22",
    "code":"A",
    "appetizer":"Suppe",
    "mainDish":"Pasta",
    "dessert":"Milka",
    "categories":"Rind;Nudel"
}
```

JSON Objekt für Menue B:

```
{
    "id":null,
    "date":"2022-03-22",
    "code":"B",
    "appetizer":"Suppe",
    "mainDish":"Lachs",
    "dessert":"Milka",
    "categories":"Fisch"
}
```

JSON Objekt für Menue C:

```
{
    "id":null,
    "date":"2022-03-22",
    "code":"C",
    "appetizer":"Suppe",
    "mainDish":"Kartofell Salat",
    "dessert":"Milka",
    "categories":"Vegan;Vegetarisch;Salat"
}
```

Menü-Vorschlag (Recommendation)

Wenn ein Mitarbeiter nicht weiß, welche Mahlezeit er wählen soll, wird mithilfe eines Recommender entschieden, welche Mahlzeit am besten dem User passt. Mithilfe eines Expertensystem wird entschieden welche Mahlzeit am besten wäre.

Um die passende Mahlezeit zu bekommen, muss der Pfad ***/menue/menues/recommendation*** aufgerufen werden.

Der Algorithmus sieht folgendermassen aus:

URL: PUT `http://localhost:8080/menue/menues/recommendation`

```
public Menue getRecommendation(String name, String date){
    List<String> categories =
        bestellungRepository.getALlCategoriesByUsername(name);
    List<Menue> menues = getMenuesByDate(date);
    Menue recommendedMenue = null;

    for (String category : categories){
        if(recommendedMenue != null){
            break;
        }
        for (Menue m : menues){
            if(recommendedMenue != null){
                break;
            }

            String[] categorieOfMenue = m.getCategories().split(";");

            for (String c : categorieOfMenue){
                if (category.equals(c)){
                    recommendedMenue = m ;
                    break;
                }
            }
        }
    }
    return recommendedMenue;
}
```

Um herauszufinden zu können welche Mahlezeit optimal wäre, werden mit Hilfe des Algorithmus alle Kategorien des Users gehollt. Jeder User der eine Bestellung getätigt hat, hatte die höglichkeit zu sehen, welche Kategorien die Mahlzeit auch besitzt. Sommit kann man leicht herauszufinden welche Kategorien auch am häufigsten in der Bestellhistorie vorkommen.

Unser Algorithmus würde nicht funktionieren, gäbe es keine Bestellungen. In anderen Worten, desto mehr Bestellungen man hat, umso genauer wird die recommendation.

Die Kategorien werden in der Datenbank als string gespeichert die mit einem Strichpunkt geteilt werden. Hat man also bei einer Mahlzeit mehr als eine Kategorie, so wird dieser als string mit einem oder mehreren Strichpunkten gespeichert.

Hier ein paar Beispiele:

Schnitzel mit Pommes = **"schwein"**

Eisbergsalat = **"Vegan;Vegetarisch"**

Nudelsalat mit Hühnerfleisch = **"Nudel;Salat;Huhn"**

Alle Menüs an einem bestimmten Tag

Um alle Menüs an einem bestimmten Tag zu bekommen, wird der Pfad ***/menue/menues/<date>*** aufgerufen.

Ein Beispiel für paar Menüs, welche am 12.08.2021 bestellt worden sind, sieht folgendermassen aus:

URL: PUT `http://localhost:8080/menue/menues/2021-12-08`


```
{
    [
        {
            "appetizer": "Nudelsuppe",
            "categories": "Schwein",
            "changedAt": "2021-12-07T19:29:08.840936Z[UTC]",
            "changedBy": "IF170009:IF170009:beni",
            "code": "A",
            "createdAt": "2021-12-07T18:37:30.322514Z[UTC]",
            "createdBy": "IF170009:beni",
            "date": "2021-12-08",
            "desert": "Obst",
            "id": 499,
            "kantine": {
            "canteenDesc": "Betriebskueche",
            "changedAt": "2021-11-25T21:29:47.471379Z[UTC]",
            "changedBy": "IF170009:beni",
            "createdAt": "2021-11-25T21:29:47.471196Z[UTC]",
            "createdBy": "IF170009:beni",
            "id": 3,
            "serviceDesc": "Mittagstisch - CORONA LIGHT",
            "status": "A"
            },
            "mainDish": "Schweineschnitzel"
        },
        {
            "appetizer": "Nudelsuppe",
            "categories": "Rind;Nudel",
            "changedAt": "2021-12-07T19:29:08.995487Z[UTC]",
            "changedBy": "IF170009:IF170009:beni",
            "code": "B",
            "createdAt": "2021-12-07T18:37:30.490055Z[UTC]",
            "createdBy": "IF170009:beni",
            "date": "2021-12-08",
            "desert": "Obst",
```

```
            "id": 500,
            "kantine": {
            "canteenDesc": "Betriebskueche",
            "changedAt": "2021-11-25T21:29:47.471379Z[UTC]",
            "changedBy": "IF170009:beni",
            "createdAt": "2021-11-25T21:29:47.471196Z[UTC]",
            "createdBy": "IF170009:beni",
            "id": 3,
            "serviceDesc": "Mittagstisch - CORONA LIGHT",
            "status": "A"
            },
            "mainDish": "Lasagne"
        },
        ...
    ]
}
```


Hier sieht man die wichtigsten Daten wie zum Beispiel die Vorspeise, Hauptspeise, Nachspeise und Desert welche von dem jeweiligem Mitarbeiter bestellt worden sind. Auch Daten wie die Bestellzeit und in welcher Kantine gegessen wird wird auch mitgegeben.

Alle Aktiven Essenszeiten

Um zu sehen ob noch freie Sitzplätze zur verfügung stehen, wird der Pfad ***/menue/oeffnungszeiten*** aufgerufen.

In unserer Kantine gibt es 4 Zeiten an denen man essen gehen kann. Diese dauern immer eine halbe Stunde und haben einen zwischenpause vo n 15 min. Pro Essenszeit können maximal 20 Personen in dem Saal essen.

Ein Beispiel für die Öffnungszeiten und für die freien Plätze sieht folgendermassen aus:

URL: PUT `http://localhost:8080/menue/menues/2021-12-08`


```
{
    [
        {
            "chosen": false,
            "freeSeats": -1,
            "id": 6,
            "maxSeats": 20,
            "time": "11:15 - 11:45"
        },
        {
            "chosen": false,
```

```
12              "freeSeats": -1,
13              "id": 7,
14              "maxSeats": 20,
15              "time": "12:00 - 12:30"
16          },
17          {
18              "chosen": false,
19              "freeSeats": -1,
20              "id": 8,
21              "maxSeats": 20,
22              "time": "12:45 - 13:15"
23          },
24          {
25              "chosen": false,
26              "freeSeats": -1,
27              "id": 9,
28              "maxSeats": 20,
29              "time": "13:30 - 14:00"
30          }
31      ]
32  }
```

4.4 Authentifizierung

4.5 Interface Webapp

4.5.1 Planung

Nachdem das Datenmodell feststand wurden UI-Prototypen entwickelt, die das Aussehen der Vue-App darstellen sollen.

Bestellvorgang

Beim Entwickeln stehen die Benutzerfreundlichkeit und das Aussehen auf mobilen Geräten an erster Stelle. Anhand dessen ist die Navigationsleiste am Wichtigsten. Diese soll den Benutzer auf alle Sichten führen können und dem Benutzer ermöglichen, sich auszuloggen.

Der Kalender muss übersichtlich sein und die Tage, an denen Bestellungen nicht möglich sind, sollten ausgeblendet sein. Ebenfalls soll auch in die Vergangenheit geschaut werden können.

Um den Rest des Platzes auszunutzen wurde eine Menüauswahlt geplant, die soviel wie möglich Information darstellen kann, ohne den Benutzer zu überfordern. Nach langem

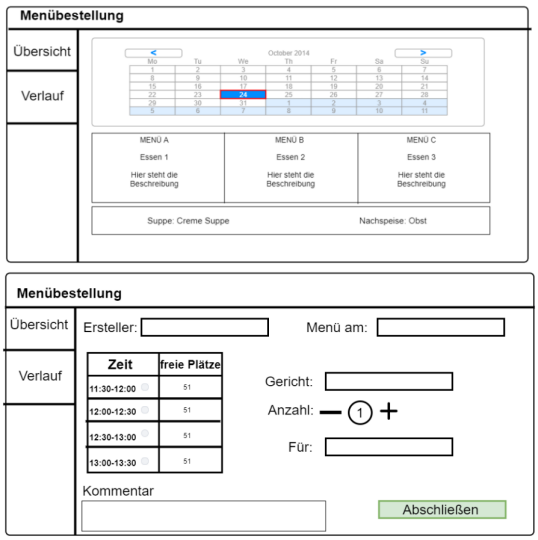

Abbildung 19: UI-Prototypen für den Bestellvorgang

Überlegen wurden die drei Menüs groß als Kästen dargestellt, um soviel wie möglich innerhalb dieser darstellen zu können. Die Vor- und Nachspeise ist kleiner dargestellt, da man keine Entscheidung darüber machen kann.
Die Sicht nachdem man sich ein Menü ausgewählt hat, soll so kompakt wie möglich sein, da auch die mobile Ansicht ohne Probleme nutzbar sein soll. Alle Informationen, die vom Benutzer eingegeben werden müssen, sollen in der Form eines Formulars angezeigt werden und andere Informationen wie Ersteller und Gericht werden automatisch eingesetzt.

Abbildung 20: UI-Prototypen für den den Login und die Übersicht

Login und Übersicht

Der Login soll einladend sein, hat aber keinen besonderen Anforderungen zu entsprechen Die Übersicht über die bereits bestellten Menüs soll dem Benutzer alle wichtigen Informationen auf einen Blick geben. Das Stornieren soll ebenfalls simpel gehalten werden, damit der Benutzer nicht darüber nachdenken muss. Der Benutzer soll auch eine Möglichkeit haben die Bestellungen zu filtern, um bestimmte Zeitpunkte bzw. Menüs einfacher zu finden.

4.5.2 Login

Beim Aufrufen der Webapp wird man zunächst zu der Login-Seite weitergeleitet. Wie diese fungiert, sieht man in der nächsten Abbildung.
Der Login dient in erster Hinsicht dazu, um festzustellen ob der Benutzer ein Mitarbeiter oder Kantinenmitarbeiter ist. Denn nach dem Login wird man entweder zur Mitarbeiter-Ansicht oder Kantinenmitarbeiter-Ansicht weitergeleitet.

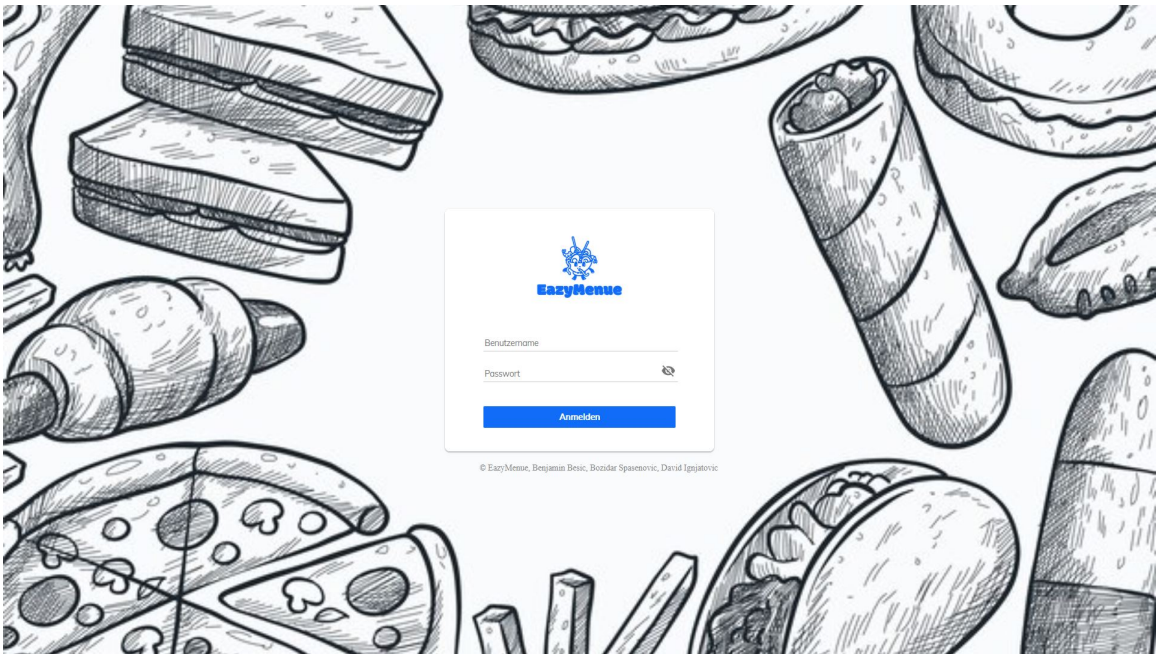

Abbildung 21: Login Screen

4.5.3 Mitarbeiter-Ansicht

Home

Das Erste, was ein Mitarbeiter zu sehen bekommt, ist die Home Ansicht. In dieser Ansicht ist eine Kalender und die Menüauswahl enthalten.
Der Benutzer kann im Kalender das gewünschte Bestelldatum anklicken, dadurch wird automatisch die Menüauswahl aktualisiert. Man kann sich in die Zukunft, sowohl auch in die Vergangenheit klicken, um sich Auskunft über die vergangenen/kommenden Menüs zu beschaffen. Die Ansicht ist nur auf die Bestelltage beschränkt, an denen Menüs angeboten werden.
Nach der erfolgreichen Datumswahl hat man unten drei Menüs zur Auswahl, sowohl wie die dazugehörige Vor- und Nachspeise. Das dem Benutzer empfohlene Menü (Analyse aus seinem Bestellverlauf) wird grün hinterlegt. Neben den Bestellköpfen befindet sich ein Fragezeichen-Knopf, wenn man über diesen geht werden einem die Kategorien des Menüs angezeigt.
Wenn die Bedingungen für eine Bestellung erfüllt sind, kann man auf einen der drei Bestellknöpfe drücken, um zur Bestellansicht weitergeleitet zu werden. Sind diese Bedingungen nicht erfüllt, sind die Knöpfe ausgeschaltet.
Weiters werden unter den Menüs noch relevante Informationen angezeigt. Durch das Klicken des Fragezeichen-Knopfs wird ein Bild der 14 Allergene geöffnet.

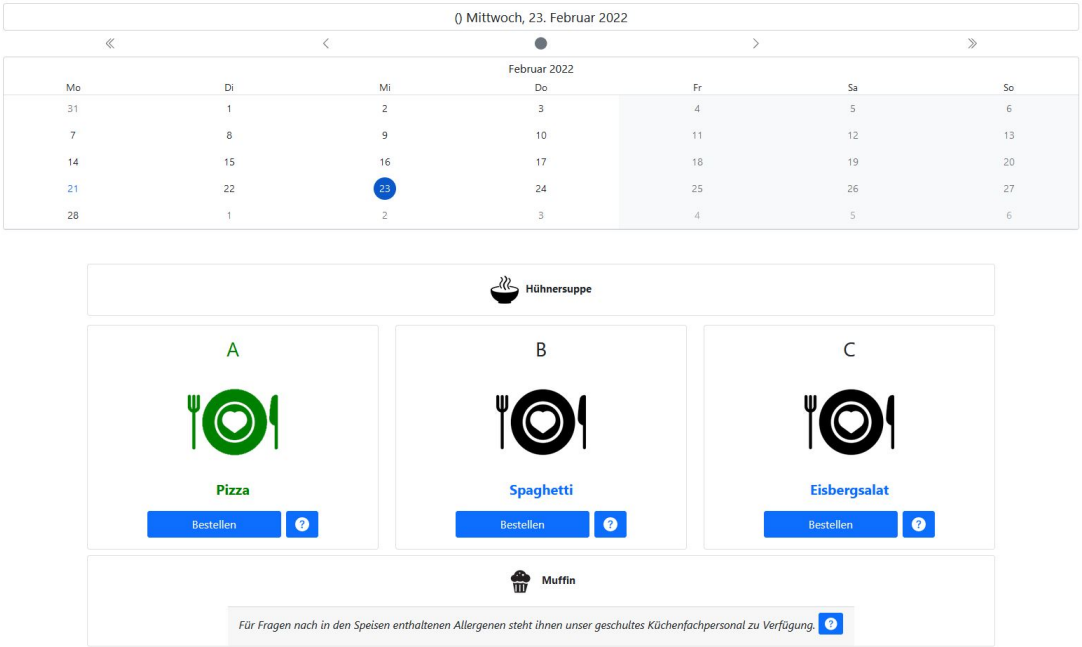

Abbildung 22: Home Ansicht eines Mitarbeiters

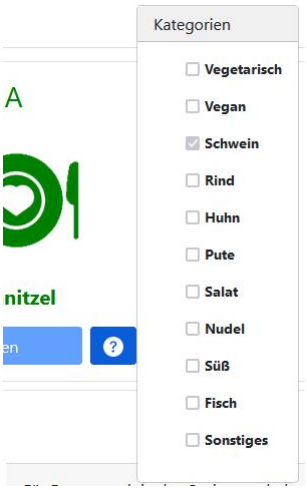

Abbildung 23: Kategorieanzeige eines Menüs

Bestellansicht

In der Bestellansicht werden die nötigen Daten für die Bestellung ausgefüllt. Die ersten drei Felder sind automatisch ausgefüllt, aufgrund der vorherigen Auswahl.
Die Tabelle auf der rechten Seite enthält alle Bestellzeiträume, die es gibt. Man kann nur eine gleichzeitig auswählen. Außerdem stehen die freien Plätze dabei, die aus der Datenbank geladen werden.
Die Anzahl der Menüs kann durchs Klicken des Plus- und Minusknopfs angepasst werden.
Darunter steht voreingestellt der Benutzer, doch dies kann verändert werden, um das Menü für einen anderen Mitarbeiter bestellen zu können.
Abschließend kann man noch einen Kommentar an die Kantine mitgeben, falls es etwaige Extrawünsche geben sollte.
Die Bestellung kann durch den Abschließen-Knopf durchgeführt werden und Abbrechen kann man jederzeit mit dem Abbrechen-Knopf. Die Bestellung ist erst ausführbar, sobald alle Felder außer des Kommentars ausgefüllt wurden.

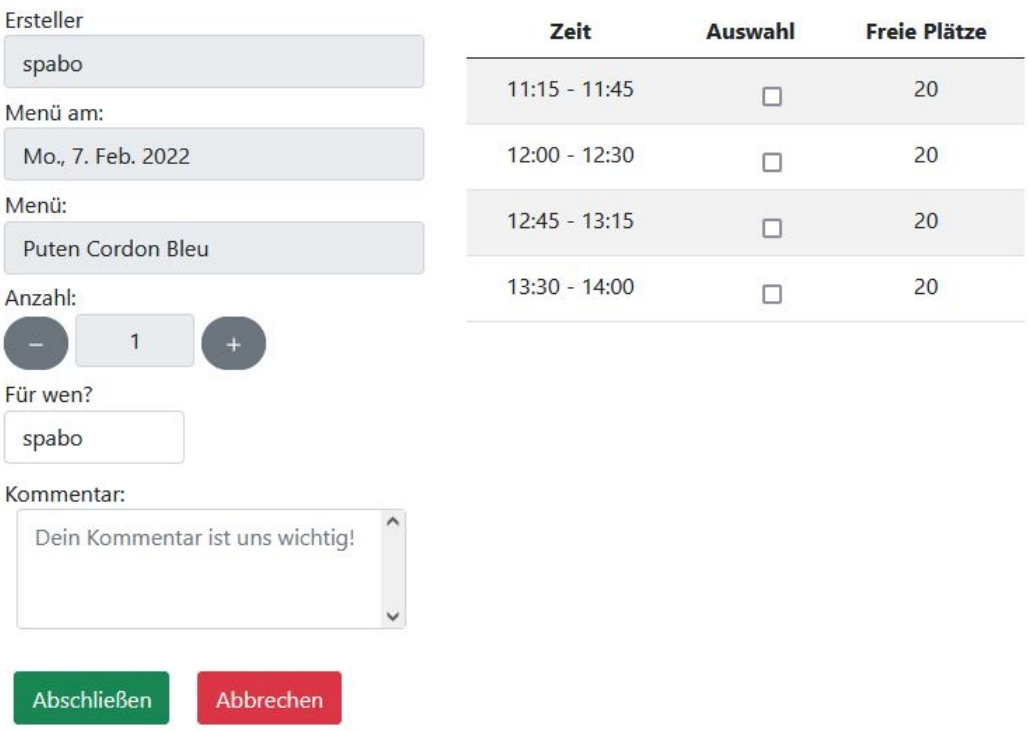

Abbildung 24: Bestellansicht

Bestellübersicht

Die Bestellübersicht dient dem Benutzer dazu seine Bestellhistorie nachzuvollziehen. Zu jeder Bestellung ist der Name des bestellten Menüs, das Menüdatum, der Bestellzeitpunkt und die Essenszeit zugeordnet.

Der Benutzer hat die Möglichkeit, oben in der Suchleiste, die Bestellungen nach Name oder Menüdatum zu filtern. Das Filtern erfolgt direkt nach der Eingabe.

Man kann jede Bestellung anklicken und falls eine Bestellung die Stornierbedingungen erfüllt kann diese mit dem unten gelegen Storno-Button storniert werden. Nach einer erfolgreichen Stornierung verschwindet die Bestellung aus dem Verlauf, doch in der Datenbank wird nur das Stornierdatum gesetzt und somit wird die Bestellung ungültig gemacht.

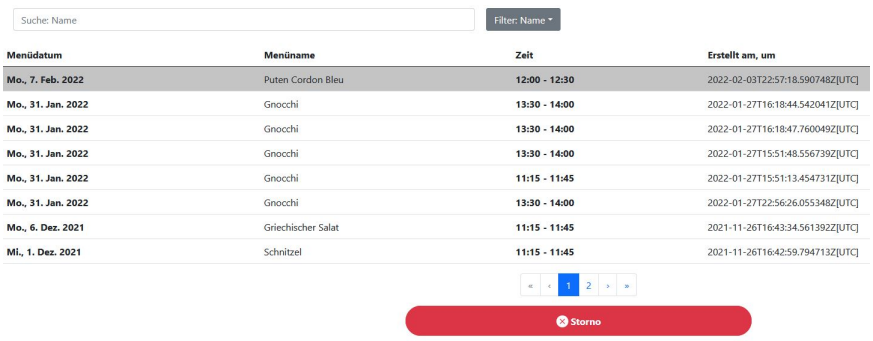

Abbildung 25: Bestellungsübersicht

Statistiken

Der Benutzer kann in dieser Ansicht mehr Informationen über seine vergangenen Bestellungen bekommen. Durch das Wechseln des Tabs oben links wird entweder eine Statistik über die Kategorien oder über die Wochentage angezeigt.
Die Informationen der Statistiken beziehen sich auf alle Bestellungen, die der Nutzer bereits getätigt hat.

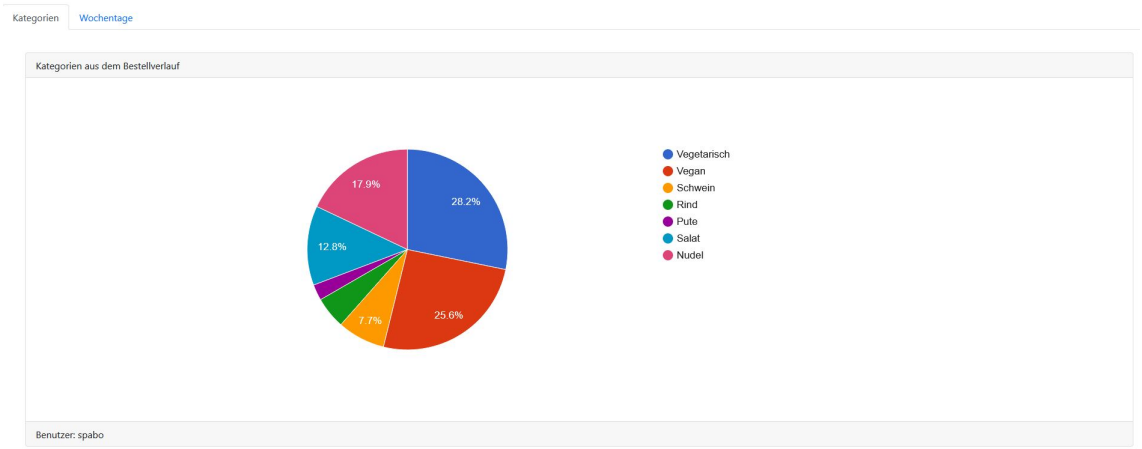

Abbildung 26: Kategorie Statistiken

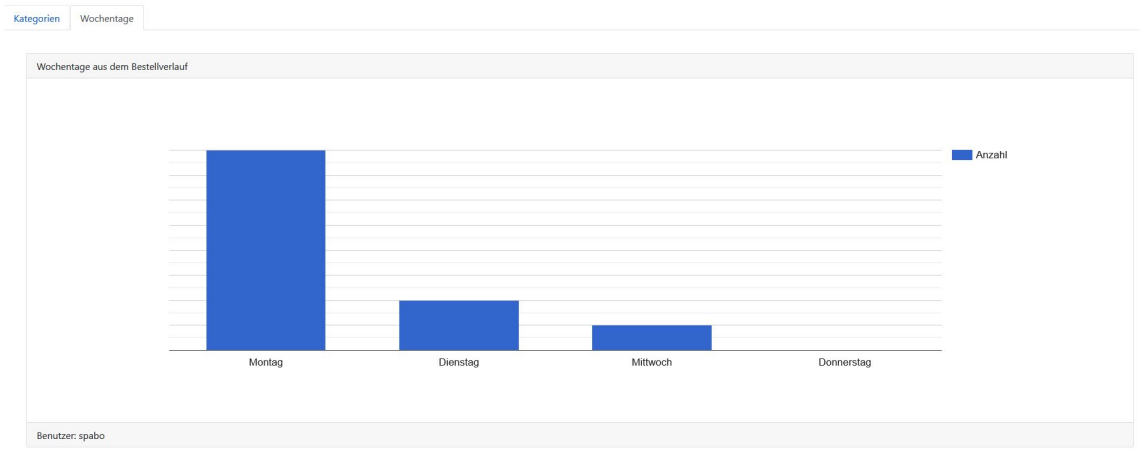

Abbildung 27: Wochentage Statistiken

4.5.4 Kantinen-Ansicht

Home

Die Startseite der Kantinenmitarbeiter ähnelt der Ansicht der Mitarbeiter. Siehe hier. Die Unterschiede sind, dass die Kantine Textfelder für jedes Menü und dessen Vor- und Nachspeise hat. Ebenfalls erscheint beim Kategorien-Knopf eine Liste von Kategorien zum Anhaken, um dem Menü die entsprechenden Kategorien zuzuordnen.
Die Textfelder und Kategorien können bearbeitet werden, um ein vorhandenes Menü zu aktualisieren oder um ein neues zu erstellen. Die Erstellung bzw. die Änderung kann durch den Speichern-Knopf durchgeführt werden. Je nach Fall werden neue Menüs an das Backend geschickt oder ein vorhandenes wird aktualisiert.

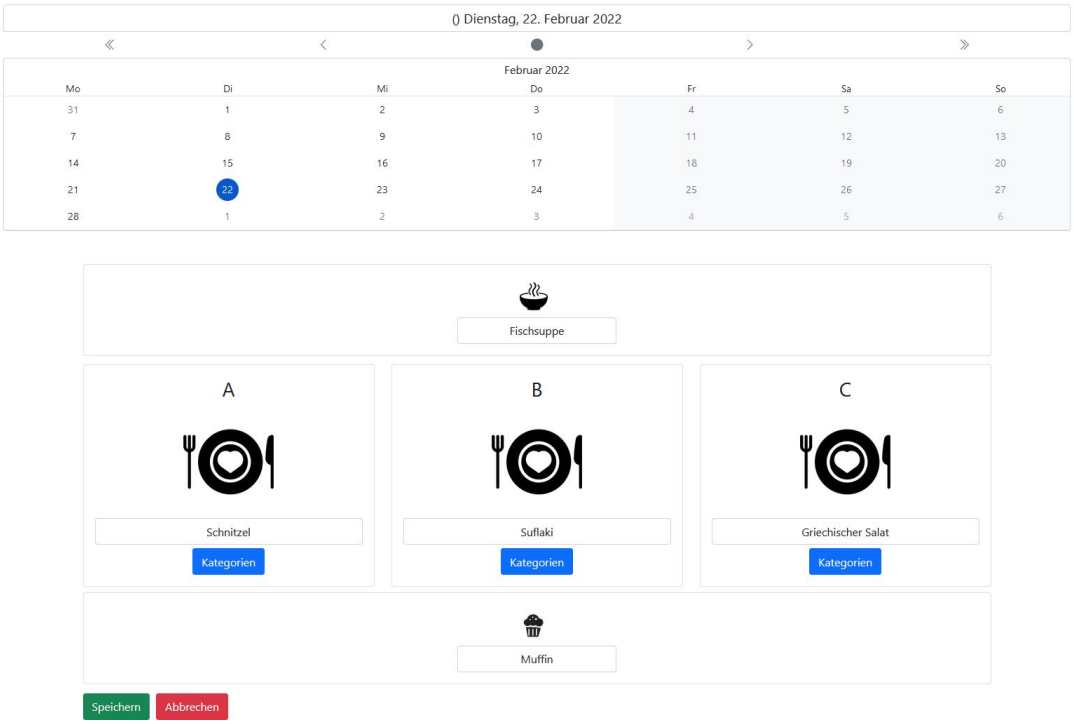

Abbildung 28: Kantine Homeansicht

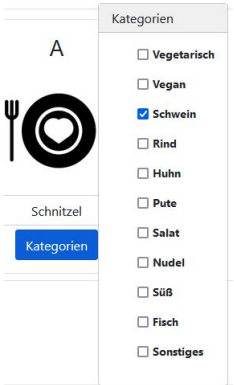

Abbildung 29: Kategorieauswahl

Drucken

Die Druckansicht dient den Kantinenmitarbeitern dazu, um die Bestellungen des jeweiligen Tages auch durchführen zu können. Der Tag kann durch eine Datumsauswahl bestimmt werden. Die Ansicht zeigt alle nötigen Informationen, damit die Kantine einen Überblick über die Bestellungen hat und dementsprechend das Essen vorbereiten kann. Es sind die Essenszeiten sowie Anzahl der jeweiligen Menücodes summiert für einen schnellen Überblick.
Durch das Drücken des Drucken-Knopfs hat die Kantine die Möglichkeit ein pdf-Dokument aus der kompletten Ansicht, die unter dem Knopf zu sehen ist, zu drucken. Dies dient dazu, dass sie einen Ausdruck auf Papier haben, der für die Kantine praktischer ist.

Abbildung 30: Druckansicht der Kantine

Abbildung 31: PDF der Ansicht

4.6 Android-App

Die Android App ist in verschieden Packages unterteilt.

- api
- composable
- dataClasses

Im api Package sind gibt es ein ApiObject, welcher alle URL's beinhaltet. Dies erleichtert die Konfiguration der Android Anwendung im Bezug auf das Kommunizieren mit dem Backend. Noch dazu gibt es die ApiService Klasse, die alle Requests erledigt.

ApiObject:

```
    object ApiObject {
    @JvmField
    var initMenuesUrl = "http://10.0.2.2:8080/menue/menues/"

    @JvmField
    var bestellungUrl = "http://10.0.2.2:8080/menue/bestellung/"

    @JvmField
    var oeffnungszeiten = "http://10.0.2.2:8080/menue/oeffnungszeiten"

    @JvmField
    var keycloak =
        "http://10.0.2.2:8082/auth/realms/menuRealm/protocol/openid-connect/token"
}
```

Im composable Package sind jene Klassen mit einer View drinnen. Diese werden jeweils mit @Composable annotiert. Diese benutzen sich dann entweder gegenseitig oder die Methoden der ApiService Klasse.
Im dataClasses Package sind die gebrauchten Entitäten und DTO's (Data Transfer Object). Hier werden den Menüs, Bestellungen, Öffnungszeiten und dem Acces Token die einzelnen Felder zugewiesen.

Die MainActivity Klasse ist die Hauptklasse über die die ganze Oberfläche der Android Applikation aufgerufen wird. Sie zeigt auch noch die Bottom Navigation Bar, welche zur navigierung zwischen Übersicht, Bestellansicht und Verlauf verwendet wird.

4.7 Login

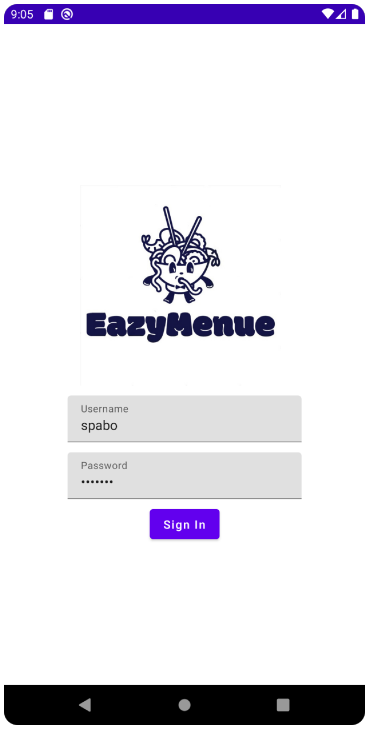

Abbildung 32: Loginscreen Android

4.7.1 Authentification

Die Authentifizierung erfolgt mit der Methode checkAccessToken, welche nichts anderes macht wie den Keycloak nach einem Token für den User zu fragen. Kriegt man einen Token zurück, so besteht der User.

```
Button(onClick = {
    if (!name.isEmpty() && !password.isEmpty() && checkAccessToken())
        {

        InitMenues()
        InitBestellungen(name)
        Toast.makeText(
            context,
            "Logged in successfully",
            Toast.LENGTH_SHORT
        ).show()
        isLoggedIn.value = true
        navController.navigate("uebersicht")

    }
}) {
    Text(text = "Sign In")
}
```

checkAccessToken()

Man schickt einen Request an http://10.0.2.2:8082/auth/realms/menuRealm/protocol/openid-connect/token mit einem passendem body. Im body stehen die Felder die für den Request nötig sind wie zum Beispiel:

- die Client ID
- der Grant Type
- der Client Secret
- der Scope
- das Passwort
- der Username

Als Client wird ein OkHttpClient verwendet. Dieser erstellt einen neuen Aufruf mit dem erstelltem Request. Er wartet auf einen Response und speichert bei einem erfolgreichem Response den Token. Da es zu Internet Problemen kommen kann haben wir einen countDownLatch eingebaut, der vor der Rückgabe der Methode darauf wartet, dass der Token gespeichert worden ist.

```
client.newCall(request).enqueue(object : Callback {
    override fun onResponse(call: Call, response: Response) {
        response.use {
            if (response.isSuccessful) {
                accessToken.value = response.body!!.string()
            }
            else{
                throw IOException("Unexpected code $response")
            }
        }
        countDownLatch.countDown()
    }
    override fun onFailure(call: Call, e: IOException) {
        e.printStackTrace()
        countDownLatch.countDown()

    }
})
countDownLatch.await()
return accessToken.value != ""
```

4.8 Übersicht

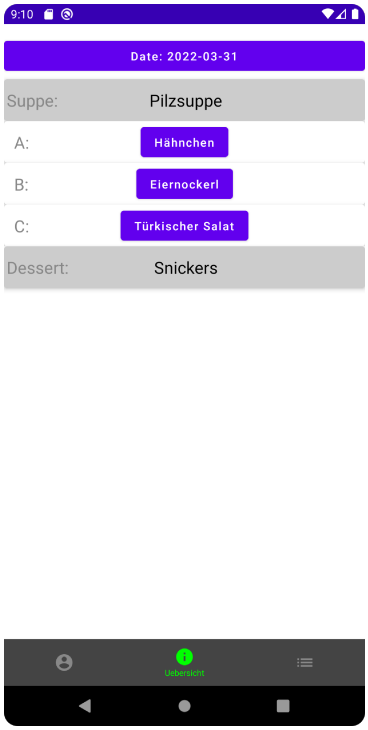

Abbildung 33: Übersicht Android

In der Übersicht kann man mit einem Datepicker ein Datum auswählen, was dazu führt das die jeweiligen Menüs an dem Tag als Cards angezeigt werden. Noch dazu wird die jeweilige Vorspeise und Nachspeise angezeigt.

4.8.1 Datepicker

Der Datepicker in Kotlin wird als DatePickerDialog verwendet. Im Dialog wird das ausgewählte datum in eine temporäre Variable gespeichert und der Methode getMenuesForDate(date: String) übergeben. Anschließend wird der Wert in der Übersicht angezeigt und man wird wieder zurücknavigiert.

```
val datePickerDialog = DatePickerDialog(
    context,
    { _: DatePicker, year: Int, month: Int, dayOfMonth: Int ->
        var temp = month.inc()
        if (dayOfMonth >= 1 && dayOfMonth <= 9) {

            if (temp.toString().length == 1) {
                date.value = "$year-0${temp}-0$dayOfMonth"
            } else {
                date.value = "$year-${temp}-0$dayOfMonth"
            }
        } else {
            if (temp.toString().length == 1) {
                date.value = "$year-0${temp}-$dayOfMonth"
```

```
            } else {
                date.value = "$year-${temp}-$dayOfMonth"
            }
        }
        getMenuesForDate(date.value)
        uebersichtDate.value = date.value
        navController.navigate("uebersicht")
    }, year, month, day
)
```

getMenuesForDate

Die Methode beinhaltet den Filteralgorhytmus von Menüs für einen Tag. Es wird erstmals überprüft ob das jetzige Datum voor dem Datum des Menüs ist und ob die jetzige Zeit, falls man am selben Tag bestellt, vor neun Uhr ist. Sind die beiden Bedingungen erfüllt so wird es dem Mitarbeiter ermöglicht ein Menü auszuwählen.

```
if (date <= LocalDateTime.now().toString().take(10) &&
    LocalDateTime.now().hour >= 9) {
    menueIsInThePast.value = true
} else {
    menueIsInThePast.value = false
}
menuesFilteredByDate = menuesFilteredByDate - menuesFilteredByDate
menues.sortedBy { menue -> menue.date }.sortedBy { menue ->
    menue.code }.forEach { menu ->
    if (menu.date == date) {
        menuesFilteredByDate = menuesFilteredByDate.plusElement(menu)
    }
}
```

Dazu wird die Methode getOeffnungszeiten() aufgerufen die die notwendigen Zeiten mit freien Sitzplätzen aus dem Backend holt und anzeigt.

```
client.newCall(request).enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {
        e.printStackTrace()
    }

    override fun onResponse(call: Call, response: Response) {
        response.use {
            if (!response.isSuccessful) throw IOException("Unexpected
                code $response")

            val gson = GsonBuilder().create()

            val collectionType: Type =
                object : TypeToken<Collection<Oeffnungszeiten?>?>()
                    {}.type
```

```
                oeffnungszeiten = gson.fromJson(response.body!!.string(),
                    collectionType)
            }
        }
    })
```

4.8.2 Card

Eine Card wird verwendet um mehrere Items schön beisammen zu halten. Man kann die Card konfigurieren indem man die erhöhung und den Schatten erhöht. Außerdem kann man Formen definieren wie zum Beispiel:

- Rectangle Shape
- Circle Shape
- Rounded Corner Shape
- Cut Corner Shape

Wie bei fast allen anderen Items, ist es auch hier möglich sowohl die Hintergrund- als auch die Vordergrundfarbe zu ändern.

4.8.3 Log out

Der Mitarbeiter hat eine Möglichkeit sich auch aus der App auszuloggen. Das erfolgt durch den Sign out Button.

Nach dem einloggen wird die Methode InitBestellungen() aufgerufen. Ihre Aufgabe ist es den derzeitigen Bestellungsverlauf des Mitarbeiters zu initialisieren. Den kann man dann auch in der Verlaufsansicht sehen. Noch dazu wird die Methode InitMenues() aufgerufen und somit alle erstellten Menüs geholt.

```
client.newCall(request).enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {
        e.printStackTrace()
    }

    override fun onResponse(call: Call, response: Response) {
        response.use {
            if (!response.isSuccessful) throw IOException("Unexpected
                code $response")

            val gson = GsonBuilder().create()

            val collectionType: Type =
```

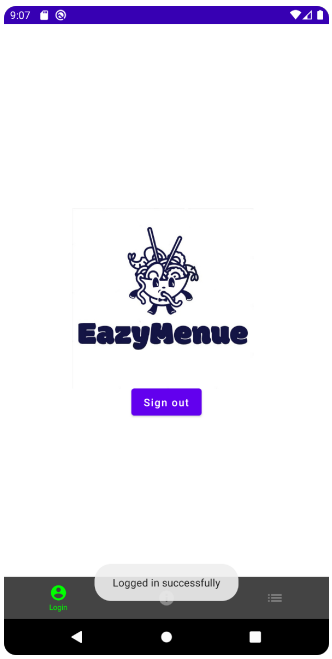

Abbildung 34: Signout Android

```
                object : TypeToken<Collection<Menue?>?>() {}.type


            menues = gson.fromJson(response.body!!.string(),
                collectionType)
        }
    }
})
```

Der Response wird so benutzt, dass man als erstes mit dem GsonBuilder ein Gson-Objekt erstellt was das umwandeln von JSON-Objekten zu Kotlin Objekten ermöglicht. Dieses Objekt wird verwendet um ein JSON-Objekt in ein Kotlin-Objekt umzuwandeln. Damit man Gson verwendetn kann, benötigt man die abhängigkeit com.google.code.gson:gson:2.8.9. Mit dem collectionType wird definiert in was für ein Typ das JSON-Objekt umgewandelt werden soll. Abschliesend wird vom Responsebody eine Collection von allen mitgegebenen Bestellungen erstellt und gespeichert.

4.9 Verlauf

Der Verlauf ist eine einfache Liste (LazyColumn) von den ganzen Bestellungen des Users. Es wird der Menüname, das Bestelldatum und das Datum des Menüs angezeigt. Ist die Bestellung stornierbar, so wird eine Mülltonne als Icon angezeigt. Durchs draufklicken auf die Mülltonne, wird die Bestellung storniert.

Abbildung 35: VerlaufAndroid Android

Es wird ein PUT auf die URL http://10.0.2.2:8080/menue/bestellung?id=$menuId was im backend das löschen zur jeweiligen Menü Id ausübt. Falls ein erfolgreicher Response zurückkommt, so wird die InitBestellungen()-Methode zum neu laden des Verlaufes aufgerufen.

```
client.newCall(request).enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {
        e.printStackTrace()
    }

    override fun onResponse(call: Call, response: Response) {
        response.use {
            if (!response.isSuccessful){
                throw IOException("Unexpected code $response")
            }

            InitBestellungen(currUser.value)
        }
    }
})
```

4.10 Bestellansicht

In der Bestellansicht werden die Felder Menü, Von und das Datum automatisch in ein Readonly Text Field gespeichert. Man kann dann die Anzahl angeben, für wen man es bestellen möchte und auch zwischen vier Zeiten aussuchen.

Abbildung 36: Bestellansicht Android

4.10.1 postBestellung()

Nach dem Abschließen wird die postBestellung()-Methode aufgerufen. Sie erstellt einen Requestbody mit den eingegebenen Werten und schickt es mittels POST an die "http://10.0.2.2:8080/menue/bestellung/" URL. Dieser Endpoint führt dann die Anfrage durch und schickt einen Response zurück. Wichtig dabei ist es den richtigen Media-Type zu definieren.

Die Anfrage und Antwort:

```
client.newCall(requestDto).enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {
        e.printStackTrace()
    }

    override fun onResponse(call: Call, response: Response) {
        response.use {
            if (!response.isSuccessful) throw IOException("Unexpected
                code $response")

            response.headers.get("orderedBy")?.let { it1 ->
                InitBestellungen(it1) }
```

```
            }
        }
    })
```

5 Zusammenfassung

Die Oberösterreichische Versicherung benutzt ein veraltetes Bestellsystem. Die vorliegende Diplomarbeit beschäftigt sich mit einer Kantinenwebapp und einem zusätzlichen Android Client, die zur Essensbestellung verwendet werden.
Ziel dieser Diplomarbeit ist es allen Mitarbeitern der Oberösterreichischen Versicherung ein Benutzerfreundliches Bestellsystem bereitzustellen und somit auf keine veraltete Technologie beschränkt zu sein.

Die Webapp bietet zwei Arten von Benutzern an. Ein Kantinen-Benutzer, welcher die einzelnen Menüs erstellt und verarbeitet und noch ein Mitarbeiter-Benutzer, der die jeweiligen Menüs bestellen kann. Die Daten für die beiden Benutzerinterfaces liefert das Quarkus-Backend zusammen mit einer Oracle Datenbank. Durch die Statistikansicht hat der Mitarbeiter eine übersicht seiner Bestellungen. Dem Benutzer wird aber auch für die einzelnen Tagen Menüs vorgeschlagen welche anhand dem Bestellverlauf ermittelt werden. Die einzelnen Menüs werden von der Kantine in verschiedene Kategorien kategorisiert. Somit kann man leicht sein passendes Menü finden.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang