

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 EISSEMER

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.

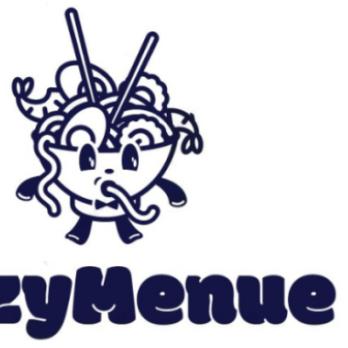
Leonding, April 2022

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

EazyMenu is an application for managing food orders in company canteens and was developed to replace an existing application used by Oberösterreichische Versicherung based on IBM Notes. This application was only accessible on the desktop and did not meet today's standards. EazyMenu instead allows employees to order a meal quickly and easily via their mobile phone or from a desktop. Meals can be categorized by tags. An algorithm then uses the tags to decide what meal should be suggested to an employee based on his previous orders. The cafeteria also has the ability to create new meals and manage them as well. It is also possible to print out an order overview with the most important information for a particular day.



Zusammenfassung

EazyMenu ist eine Anwendung zur Verwaltung von Essensbestellungen in Firmenkantinen und wurde entwickelt, um eine Altapplikation der Oberösterreichische Versicherung auf Basis von IBM Notes abzulösen. Diese Altapplikation war nur auf dem Desktop zugänglich und entsprach nicht den heutigen Standards. Im Gegensatz dazu ermöglicht EazyMenu Mitarbeiterinnen und Mitarbeitern per Handy oder von einem Desktop aus schnell und einfach eine Mahlzeit zu bestellen. Dabei können Gerichte durch Tags kategorisiert werden. Ein Algorithmus entscheidet dann anhand der Tags, welche Mahlzeit einem Mitarbeiter auf Basis seiner bisherigen Bestellungen vorgeschlagen werden soll. Auch die Kantine hat die Möglichkeit neue Mahlzeiten zu erstellen und diese auch zu verwalten. Es besteht auch die Möglichkeit eine Bestellübersicht mit den wichtigsten Informationen für einen bestimmten Tag auszudrucken.



EazyMenue

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangssituation

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. Zu jeder Hauptspeise gehört zusätzlich eine Vor- und 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 Mitarbeitende ist mit seinen bzw. ihren Daten bereits eingeloggt. Auf einem simplen Interface kann man zwischen den heutigen und zukünftigen Mahlzeiten wählen. Dazu kann man einen Zeitpunkt auswählen, wann man eine Mahlzeit konsumieren will. 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 Bestellung unter Benutzung von mobilen Endgeräten ist nicht möglich. 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 war daher, 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 Empfehlungssystem

entwickelt werden, dass einem ermöglicht Mahlzeiten zu bestellen, die auf einen abgestimmt sind. Außerdem soll es eine Einsicht über die Bestellhistorie geben mit diversen Statistikelementen.

1.5 Ziel(e)

- Erleichterung des Bestellprozesses für den Benutzenden
- Flexible Bestellmöglichkeiten
- Der Benutzende hat bessere Einsicht über sein Bestellverhalten

1.5.1 Zielgruppe

Grundsätzlich ist das Programm an die Mitarbeiter der OÖ Versicherung AG gerichtet. Doch das Projekt könnte durchaus unter anderen Bedingungen für andere Firmen bzw. Einrichtungen umgesetzt werden.

2 State of Art

Die Oberösterreichische Versicherung hat das veraltete IBM-Notes Bestellsystem verwendet. Das hatte keine Möglichkeit Kategorien auszuwählen, die Beestellübersicht mit Graphen und dem Vorschlagen des *perfekten* Menüs für den Benutzer.

Die Webapp sollte nicht mit *Lieferando*, *Mjam*, *Delivery Hero* oder *Lieferheld* und weiteren verglichenn werden da sie nur auf Gerichte der Kantine des Unternehemens spezifiziert ist.

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 eine Programmierumgebung für Kotlin, Groovy, Scala und auch Android. Sie ist immer auf dem neusten Entwicklungsstand, wird laufend mit Updates versorgt und unterstützt die derzeit gängigen Programmierools 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. Diese Intelligenz beinhaltet z.B. Codevorschläge oder Vorschläge, um die Codequalität zu verbessern. [?]

Wir haben uns für die Verwendung entschieden, da wir damit viel Erfahrung hatten und die oben genannten Punkte unterstützten unsere Entscheidung dabei.

3.2 Android Studio

[?] Android Studio ist eine Entwicklungsumgebung für Android-Anwendungen. Es basiert auf IntelliJ.

Neben dem leistungsstarken Code Editor werden noch mehr Funktionen bereitgestellt, um für die Produktivität zu verbessern. Android Studio funktioniert auf Windows, GNU/Linux, macOS und Chrome OS. Die erste Version kam nach zwei Jahren Entwicklungszeit am 8. Dezember 2014 heraus. Weiters beinhaltet es ein Gradle-build-system, viele verschiedene Androidemulatoren und ein Vorschaufenster mit der Anwendung.

3.2.1 Apply Changes

[?] Das ist eine sehr wichtiges Merkmal für alle Programmierer. Ab der Android Studio 3.5 Version ist es möglich, seinen Code zur Laufzeit des Programmes zu ändern, ohne die Applikation zu schließen. Um dieses Merkmal benutzen zu können, muss man ein sogenanntes *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.

3.2.2 Emulator

Um eine App sehen zu können, benötigt man ein Emulator oder ein echtes Gerät. Der Android Emulator simuliert also nicht nur ein einzelnes Gerät sondern viele andere Versionen. Somit kann man seine App auf mehreren Devices problemlos testen, auch wenn man nicht jedes besitzt.

Den Emulator kann genauso viel wie ein normales Handy. Anrufen, SMS, Standort sowie rotieren des Emulators ist kein Problem. Einer der Vorteile zu einem echten Gerät ist

die Schnelligkeit der Übertragungen von Daten. Android Studio bietet eine Vielzahl von Emulatoren für normale Handys, Tablets, Smart-Uhren und Android Fernseher.

3.3 Git

Git ist ein Versionskontrollsysteem (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. [?]

Zuallererst 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 nachvollziehbar 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).

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 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

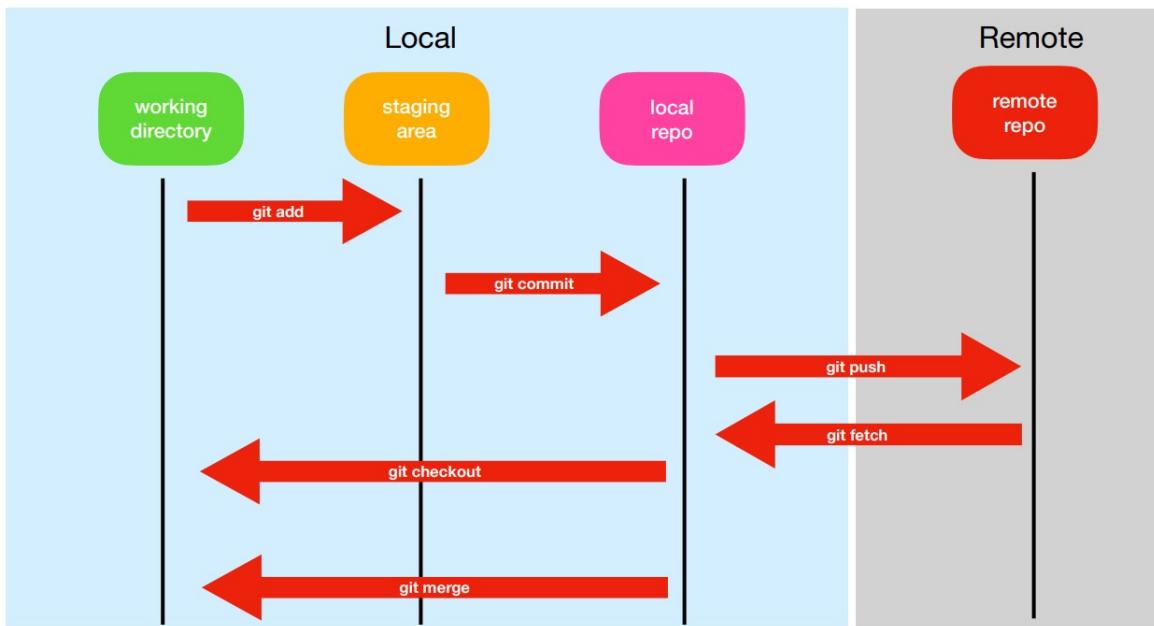


Abbildung 1: Darstellung des Git Workflows

sind.

Ein Vorteil, den Git ebenfalls bietet ist, dass 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.

Der Commit speichert die Veränderungen, die seit dem letzten Commit gemacht wurden

und dieser 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. [?]

Pull

Der Pull Befehl kopiert die Inhalte vom öffentlichen Repository und fasst diese mit den lokalen Zustand auf dem Rechner zusammen (merge). Er 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 wird an neuen Funktionen gearbeitet, 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. [?]

3.3.4 GitHub

GitHub ist ein gewinnorientiertes Unternehmen, das 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

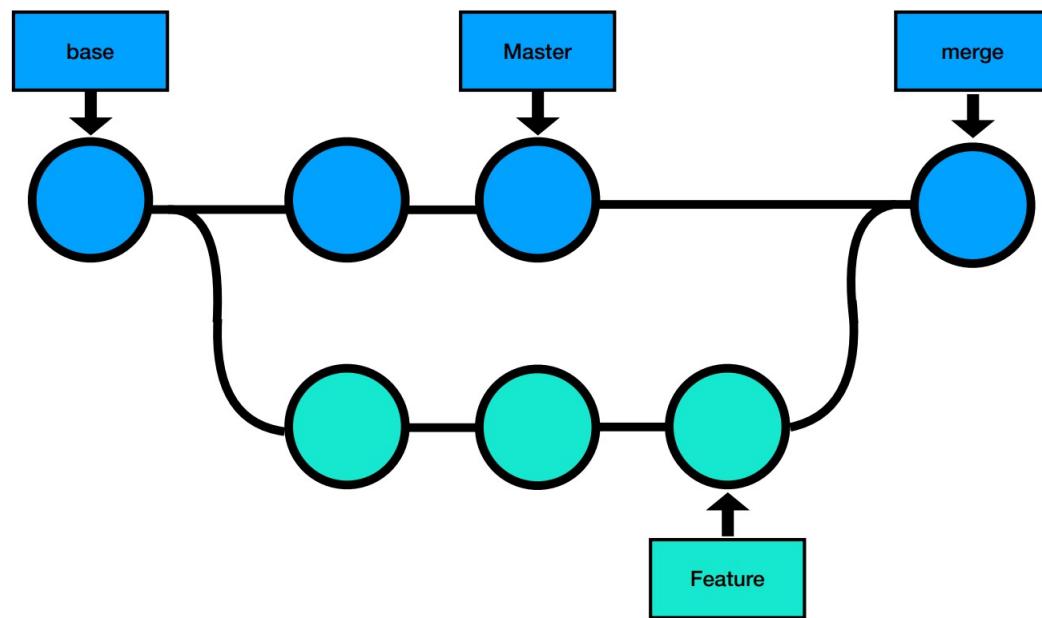


Abbildung 2: Darstellung von Branches in einem Repository

Entwicklern und über 4 Millionen Organisationen benutzt. Es ist 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 Repositories veröffentlichen. Es ist nicht nur für Code-basierte Projekte verwendbar sondern auch Websites 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. In diesem Board können alle Probleme aufgelistet werden und dementsprechend können Teammitglieder zu bestimmten Problemen zugewiesen werden.

Es gibt ebenfalls Finanzierungspläne, die es vor allem Organisationen und Unternehmen leichter machen Unternehmensprojekte zu verwalten durch zusätzliche Funktionen. [?] Diese Funktionen wären zum Beispiel:

- Mehr Speicherplatz für das Repository
- Branches können bestimmten Nutzern zugewiesen werden
- Es kann eine Prüfung durch Dritte bei Branchänderungen erforderlich gemacht werden

- Bessere Möglichkeit nachzuvollziehen, wer, was, wann gemacht hat durch detaillierte Logs
- Und viele weitere ...

3.4 GitHub Actions

GitHub Actions ist eine 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 Entwicklern das Programmieren an einem Projekt zu vereinfachen. Ebenfalls wird die Bereitstellung an den Kunden vereinfacht.

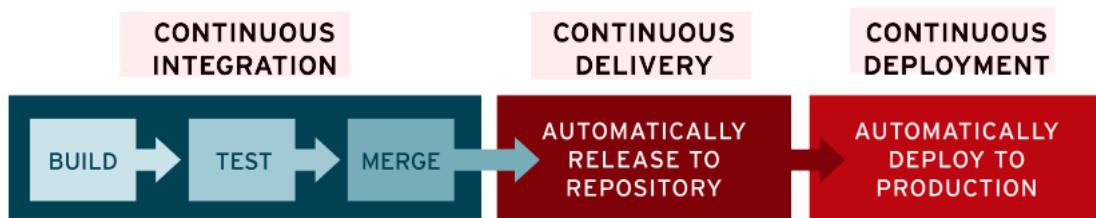


Abbildung 3: Zusammenhang zwischen CI und CD
https://www.redhat.com/cms/managed-files/ci-cd-flow-desktop_edited_0.png

CI (Continuous Integration)

CI steht für "Continuous Integration", dies beschreibt den Prozess der Automatisierung aus der Sicht eines Programmierers.

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 die Funktionsfähigkeit eines Programms validiert mittels Tests, 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, durch einen automatisierten 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.[?]

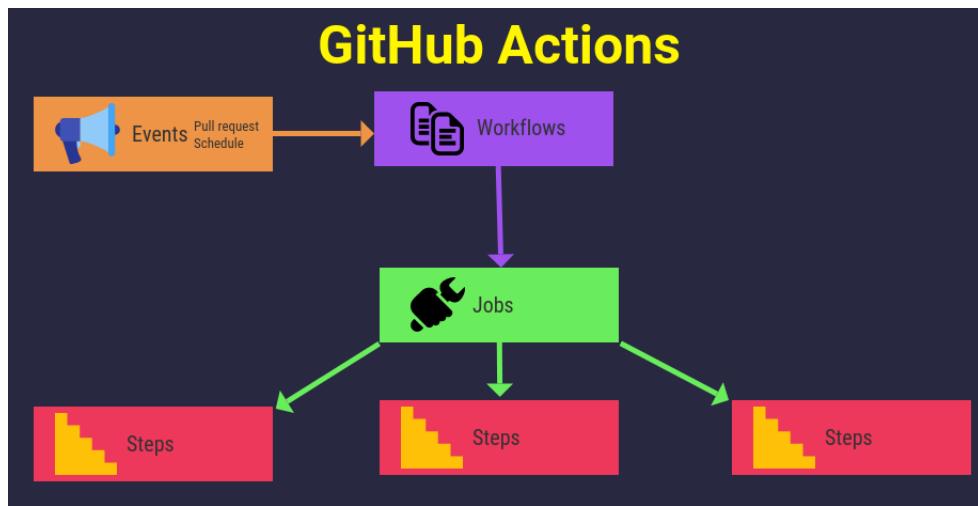


Abbildung 4: Ablauf von Github Actions
https://miro.medium.com/max/1400/1*8Agtv-SaM-zw_GLzPl7p5w.png

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 den main-Branch. Man kann es aber auch manuell oder nach einem bestimmten Zeitplan ausführen. [?]

Listing 1: Ausschnitt eines Workflow Files

```

1 # Workflow Bezeichnung
2 name: Publish EazyMenu 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 einen 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
4          # ausgefuehrt
5          runs-on: ubuntu-latest
6          env:
7              # Unter env kann man sich Variablen definieren
8              IMAGE_NAME: eazy-menue-backend
9          defaults:
10             run:
11                 # Es wird der backend-Ordner aus dem Repo
12                 # genommen
13                 working-directory: ./eazy-menue-backend
14             steps:
15                 - name: Check out the repo
16                     uses: actions/checkout@v2
17                 - name: Package
18                     run: mvn package -DskipTests
19                 - name: Build image
20                     run: docker build . -f
21                         src/main/docker/Dockerfile.jvm --tag
22                         $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 Redundanz 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 gängigsten 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)

Die 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 dieselben 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. [?]

Die Programmiersprache Java ist Teil der Java-Technologie. Sie besteht aus dem sogennaten Java Development Kit (JDK). Die JDK wird dann verwendet um Java-Programme zu erstellen und die Java Runtime Environment (JRE) um diese auszuführen. 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. 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 um. 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 ein Maschinencode in Form einer .class-Datei.

3.7 JUnit

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

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.

Beispiel für einen Unit Test (assertEquals):

Listing 4: Java File JUnit 1

```

1      @Test
2      public void t02_MenueTest_CreateMenueShouldBeSchnitzel(){
3          Menue menue = new Menue();
4          menue.setMainDish("Schnitzel");
5
6          assertEquals(menue.getMainDish(), "Schnitzel");
7      }

```

Beispiel für einen Unit Test (assertNotEquals):

Listing 5: Java File JUnit 2

```

1      @Test
2      public void t03_MenueTest_CreateMenueShouldNotBeSuppe(){
3          Menue menue = new Menue();
4          menue.setMainDish("Schnitzel");
5
6          assertNotEquals(menue.getMainDish(), "Suppe");
7      }

```

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

3.8 Karate

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

Ein HTML Report sieht folgendermaßen aus:

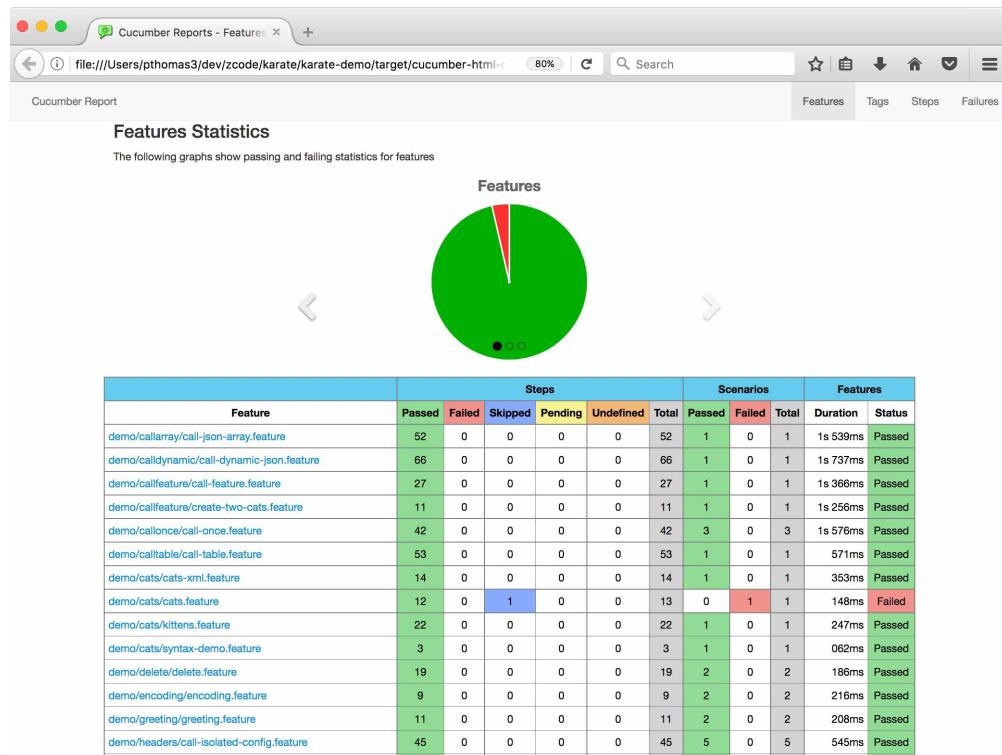


Abbildung 5: Karate Report
<https://karatelabs.github.io/karate/>

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), ist 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 ein einzelner Web-Container aus.

Die erste Version von Java EE, welche damals noch Java2EE genannt wurde, erschien 1999. Im Jahr 2003 wurde der Name der Plattform auf Java EE geändert. Die erste völlig ausgebauten Version von Java EE, welche 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 bereits erwähnt, wird zum Ausführen einer Java EE Webanwendung ein Applikationsserver verwendet. Die am häufigsten verwendeten Applikations-Server sind unter einer freien Lizenz verfügbar (sogenannte Open-Source-Software). Beispiele für solche Servere währen 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 ausreichend. 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 einem Konkurrenzprodukt eine deutlich kürzere Startzeit und einen erheblich geringeren Speicherbedarf, was ein großer Vorteil ist für das Serverless Computing. [?]

Mit Quarkus hat man die Möglichkeit, Applikationen für die moderne, Cloud-native Welt zu erstellen. Es ist ein Kubernetes-natives Java-Framework, welches auf GraalVM und HotSpot angepasst wurde.

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 sehr klein ist. In diesem jar-File befinden sich auch noch die einzelnen Libraries, welche für die Webanwendung verwendet werden.

Bei 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 besser 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 g haben wir Java EE gewählt, da es vom Auftraggeber vorgesehen war. Mit Java EE hatten 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 gab es nicht nur eine File zum konfigurieren sondern mehrere .xml Files. Das war ebenso noch ein Grund weshalb wir uns zum Schluss doch noch für Quarkus entschieden haben.

3.11 JPA

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

Enterprise Applikationen führen üblicherweise viele Operationen auf Datenbanken durch. Durch die Hilfe von JPA (Java Persistence 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.

3.11.1 Entity

Das Grundobjekt ist die so genannte Entity Klasse, die sich aus einem Standardobjekt-Java-Objekt, auch genannt POJO, ableitet. In einer relationalen Datenbank werden Entities in form einer Tabelle angezeigt. Einzelne Klassen Attribute sind in einer Tabelle die zeilen. [?]

In JPA arbeitet man viel mit Annotationen. Die wichtigsten Annotationen kann man im folgenden Beispiel leicht ersehen:

Listing 6: JPA Entity

```
1  @Entity
2  public class Oeffnungszeit {
3      @Id
4      @Column(name = "OEFFNUNGSZEIT_ID")
5      private Long id;
6
7      @Column(name = "STATUSKZ", length = 1)
8      private char status;
9
10     @ManyToOne
11     @JoinColumn(name = "KANTINE_ID")
12     private Kantine kantine;
13 }
```

Mithilfe der Annotation @Entity wird die Klasse als Entity erkannt. Jede relationalen Datenbank braucht einen Primarschlüssel beziehungsweise eine Id. In JPA wird das mit der Annotation @ID gemacht.

3.11.2 Beziehungen

Um Beziehungen darzustellen gibt es 3 Annotionen.

- @OneToOne
- @OneToMany
- @ManyToMany

Diese Beziehungen können dann auf zwei Arten umgesetzt werden.

- Unidirektional
- Bidirektional

Unidirektional bedeutet, dass die Beziehung nur in einer Richtung zeigt. Bidirektional zeigt in beide Richtungen.

Durch folgender Grafik werden die zwei Arten deutlich gemacht.

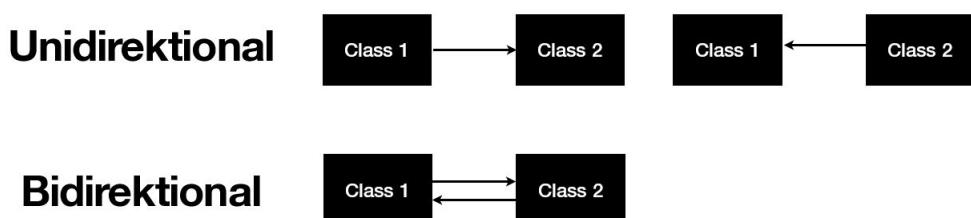


Abbildung 6: Ablauf des OAuth 2.0 Protokolls

3.11.3 Java Persistence Query Language

JPQL ist die Abfragesprache von JPA2 und differenziert sich von SQL in den folgenden Punkten.

- JPQL greift auf Entities und Objekte zu.
- B SQL greift auf Tabellen, Spalten und Zeilen zu.

Hier sind Beispiele für einige Queries:

TypedQuery

TypedQuery ist sozusagen die Standard Abfrage. Es besteht aber auch die Möglichkeit Parameter mitzugeben welche mit einem doppelten Punkt gekennzeichnet werden. Hier ein Beispiel für eine TypedQuery:

Listing 7: JPA Entity

```
1     TypedQuery<PersistenceOrderItem> allMatchesQuery = em.createQuery("SELECT u  
2         FROM " + getEntityClass().getName()  
3         + " u WHERE u.product = :prod", getEntityClass());
```

NamedQuery

NamedQuery ist deutlich übersichtlicher indem sie als Annotation an eine Entity dazugeschrieben werden und somit an verschiedenen Stellen aufgerufen werden. Hier ein Beispiel für eine NamedQuery:

Listing 8: JPA Entity

```
1     @Entity  
2     @Table(name = "Bestellung")  
3     @NamedQuery(  
4         name = "Bestellung.getAllOrders",  
5         query = "SELECT b FROM Bestellung b")  
6     public class Bestellung {  
7         ...  
8     }
```

3.11.4 Hibernate

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

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

3.12 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. [?]

3.12.1 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. [?]

3.12.2 Panache

Panache ist eine Quarkus-spezifische Bibliothek, welche die Entwicklung einer Hibernate-basierten Persistenzschicht 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.

[?]

3.13 Maven

Apache Maven ist ein Abhängigkeitsverwaltung und ein Automatisierungstool welches hauptsächlich nur für Java Applikationen verwendet wird. Wie auch Ant verwendet Maven verwendet genauso wie Ant *XML-Files* nur das sie viel besser zum verwalten sind. Während Ant Flexibilität bietet und erfordert, dass alles von Grund auf neu geschrieben wird, verlässt sich Maven auf Konventionen und stellt vordefinierte Befehle (Ziele) bereit.

[?]

Ein Beispiel für eine *pom.xml* File sieht folgendermaßen aus.

Listing 9: maven file

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4     http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6   <groupId>baeldung</groupId>
7   <artifactId>mavenExample</artifactId>
8   <version>0.0.1-SNAPSHOT</version>
9   <description>Maven example</description>
10
11  <dependencies>
12    <dependency>
13      <groupId>junit</groupId>
14      <artifactId>junit</artifactId>
15      <version>4.12</version>
16      <scope>test</scope>
17    </dependency>
18  </dependencies>
19 </project>
```

3.13.1 Ant vs. Gradle vs. Maven

Ant

Am Anfang war Make das einzige Werkzeug für automatisierte Prozeduren. Make wurde 1976 hauptsächlich für Java Applikationen vernwendet. Ant ist in vielen Aspekten gleich wie Make. Der einzige Unterschied ist, dass Ant nicht nur für Java Applikationen verwendet werden kann.

Die sogenannten Ant build files sind in *.xml* geschrieben und werden *build.xml* genannt.

[?]

Gradle

Gradle wurde nach den Konzepten von Ant. Der große Unterschied ist das Gradle keine XML files benutzt sondern in einer Domänenpezifische Sprache verwendet. Der Vorteil

dabei ist, dass die Konfigurations Files deutlich kleiner sind. Die Konfigurations File wird als *build.gradle* bezeichnet.

[?]

Maven

Der Grund warum wir Maven verwenden ist, dass wir viel Erfahrung im Unterricht gesammelt haben. Noch dazu arbeitet Maven gut mit Quarkus zusammen.

3.14 Cypress

[?] Cypress ist eine JavaScript Framework, welches 2020 auf den Markt kam. Der CEO von Cypress ist Drew Lanham. Da es kostenlos für jeden ist, hat Cypress somit eine gute Position unter den Entwicklern eingenommen. Mit Cypress ist es möglich, dass man die Möglichkeiten seiner Webapplikation zu bestimmten Anwendungsfällen zu testen. Ein Anwendungsfall wäre eine Bestellung aufzugeben, eine vorhandene Bestellung löschen oder einzelne Menüs zu erstellen und kategorisieren.

Mit Cypress ist es möglich solche Beispiele aufzuzeichnen und es mehrere Male abspielen zu lassen. Somit wird der Input vom User von Cypress übernommen.

Die Vorteile davon sind:

- schnelles Testen
- Inputfehler vermeiden
- leicht konfigurierbar
- Fehleranalyse leichter

Cypress bietet aber noch mehr Funktionen an. Man kann sehen was der nächste Schritt ist, somit kann man es auch Debuggen. Es ist aber auch oft von Vorteil wenn man eine Bildschirmaufnahme oder Fotos hat. Somit ist man ständig in der Lage einen Fehler nachzuvollziehen.

Es gibt drei Arten von Tests:

- End-To-End
- Integration
- Unit

3.14 Cypress

Cypress wurde in dieser Diplomarbeit nicht verwendet, da die Anwendungsfälle minimal sind. Als Testframework haben wir stattdessen Karate-Tests verwendet.

3.15 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. bietet es eine Unterstützung dafür an. Zu diesen Frameworks zählen z.B.: Quarkus, Angular, Vue.js, Spring, usw. [?]

3.15.1 Distributionen

Server

Die Standalone-Applikation ist downloadbar als .zip auf der Keycloak Seite. Es gibt zwei Versionen vom Server, zu einem die Wildfly 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, eines, das auf dem Wildfly Server basiert und eines, das auf Quarkus basiert.

Operator

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

3.15.2 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) sowie das Managen und Löschen von diesen.
- **Die User Zugriff-Funktion (log-on):** Dies umfasst ein Interface, in dem 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. [?]

3.15.3 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 wird typischerweise ermöglicht mit LDAP (Lightweight Directory Access Protocol). Bei Websites 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, dadurch werden auch die IT-Kosten reduziert [?]

3.15.4 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

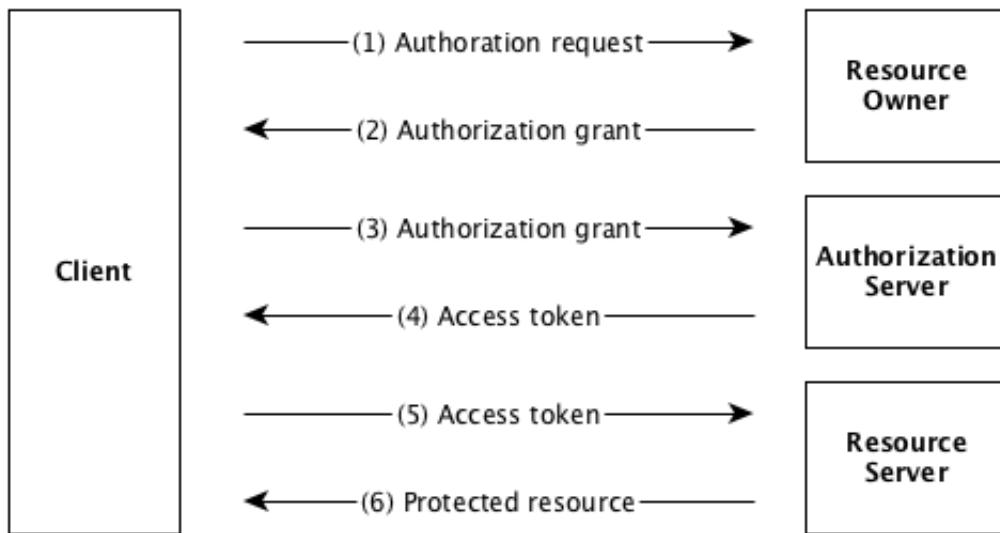


Abbildung 7: Ablauf des OAuth 2.0 Protokolls
https://miro.medium.com/max/1400/1*1McvnvrW6wh37ECYpmTSxw.png

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, übergibt OAuth 2.0 beim Kommunizieren nicht direkt den Benutzernamen und das 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. [?]

3.15.5 OpenID Connect (OIDC)

Mit Keycloak ist es ebenfalls möglich OIDC umzusetzen bzw. zu benutzen.

OpenID Connect ist ein Authentifizierungs- und Autorisierungsprotokoll, das 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 bleiben einem mehrere Benutzeraccounts erspart. [?]

Ablauf von OIDC

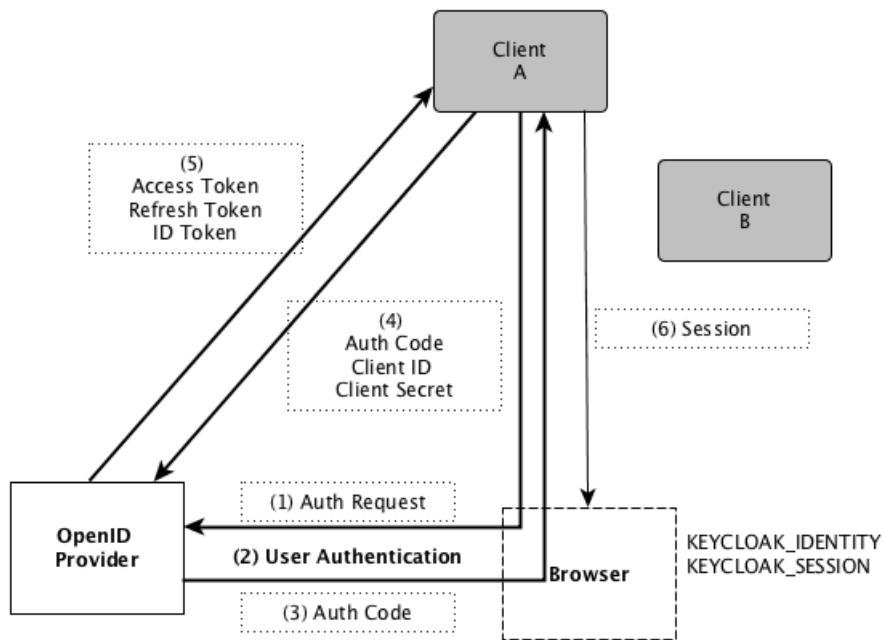


Abbildung 8: Ablauf des OIDC Protokolls
<https://blog.codecentric.de/files/2016/08/openidconnect1.png>

Der Browser leitet den Benutzer anfangs zum OpenID Provider hin. Der Benutzer authentifiziert sich dann beispielsweise mit Name und Passwort. 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 verhindert, dass 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. [?]

3.15.6 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, diese sind aber erweiterbar durch selbstgeschriebene Extensions. Dies 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). Die Seiten sind im FTL Format geschrieben, somit kann man klassisches HTML und CSS zum Editieren verwenden. Auch Javascript steht zur Verfügung, damit ist der Anpassungsspielraum sehr groß. [?]

3.15.7 Funktionsweise

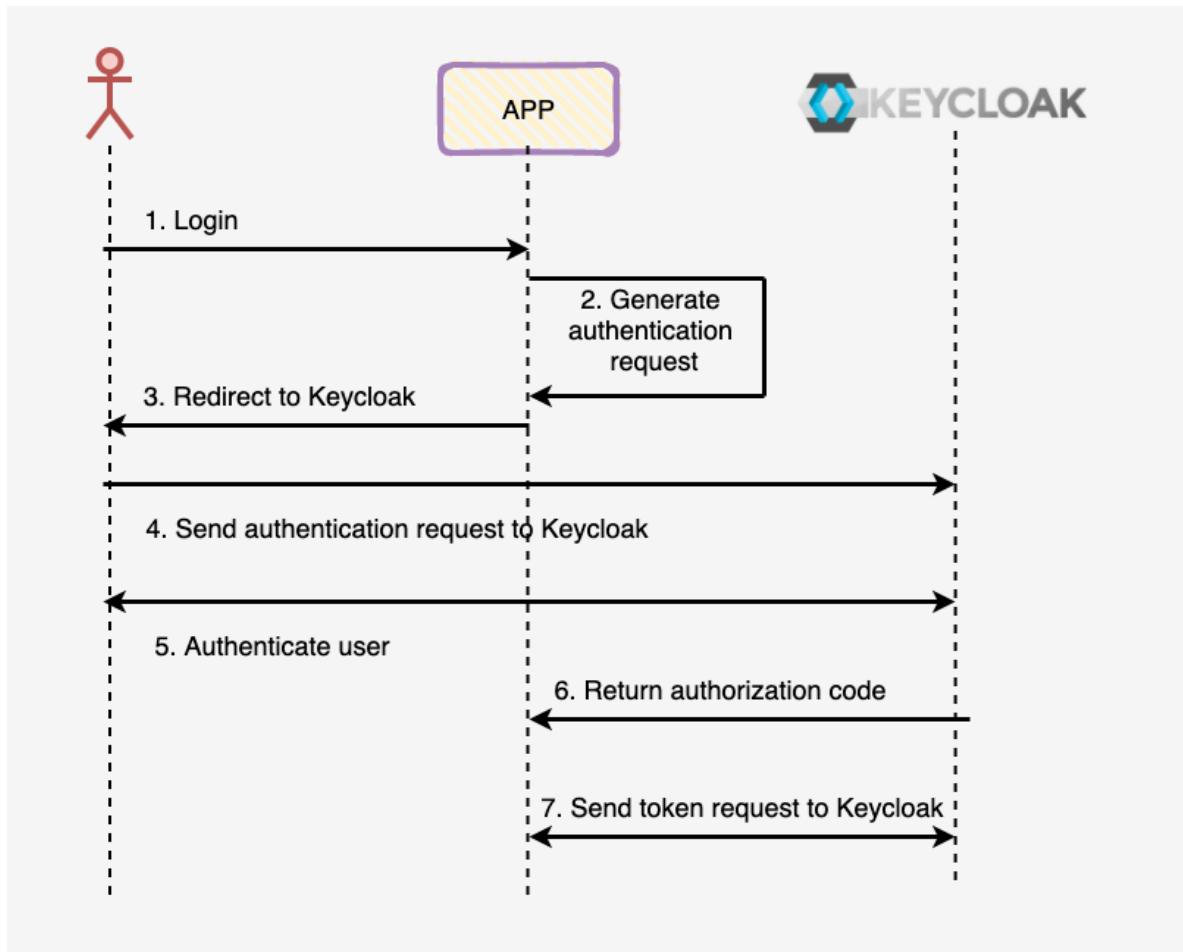


Abbildung 9: Keycloak Funktionsweise
https://miro.medium.com/max/1400/1*1McvnvrW6wh37ECYpmTSxw.png

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. Nach dem 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 nötig:

Realm

Ein Realm kann man sich als Vermieter 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.

3.16 Oracle Datenbank

Oracle Database ist ein relationales Datenbankmanagementsystem, auch genannt RDBMS, welches von der Firma Oracle hergestellt worden ist. Oracle ist einer der bekanntesten und auch meist verwendeten 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 relationalen 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-eigene 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.

3.17 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. [?]

3.17.1 SQL - Select

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

Listing 10: 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 spezielle Daten abgefragt.

Listing 11: Sql Select Where

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

Die wichtigsten SQL-Commands 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 - modifiziert eine Datenbank
- CREATE TABLE - erstellt eine neue Tabelle
- ALTER TABLE - modifiziert eine Tabelle
- DROP TABLE - löscht eine Tabelle
- CREATE INDEX - erstellt einen Index
- DROP INDEX - löscht den Index

3.18 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 ursprünglichen 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[?]. [?] [?]

3.18.1 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. [?]

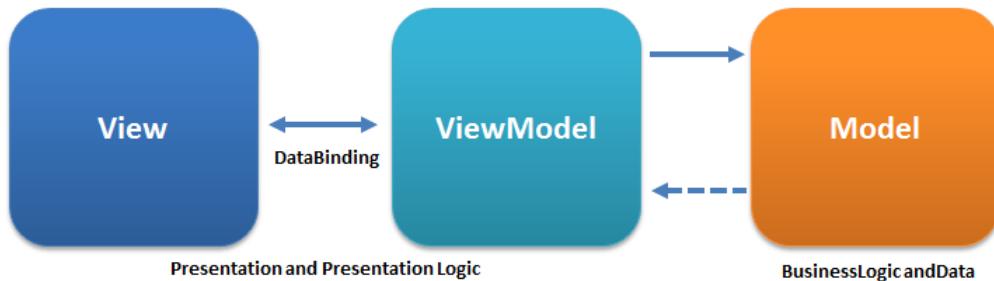


Abbildung 10: Darstellung von Branches in einem Repository
<https://upload.wikimedia.org/wikipedia/commons/8/87/MVVMPattern.png>

- **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 12: 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.

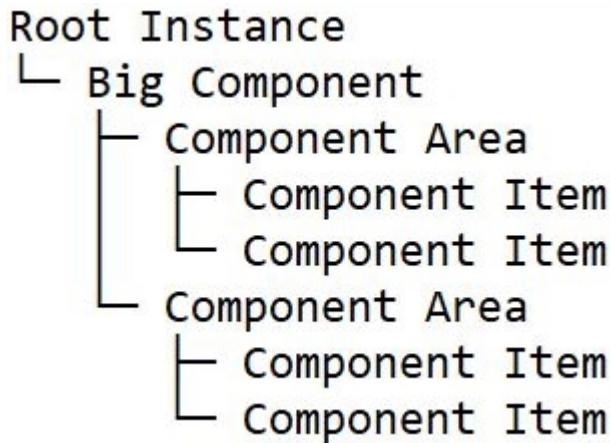


Abbildung 11: Der Stammbaum einer Root Instanz

Zu einer Instanz gehört auch der data-Bereich. Dieser beherbergt alle Properties einer Instanz und diese Properties reagieren auf Veränderungen von Werten. 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.

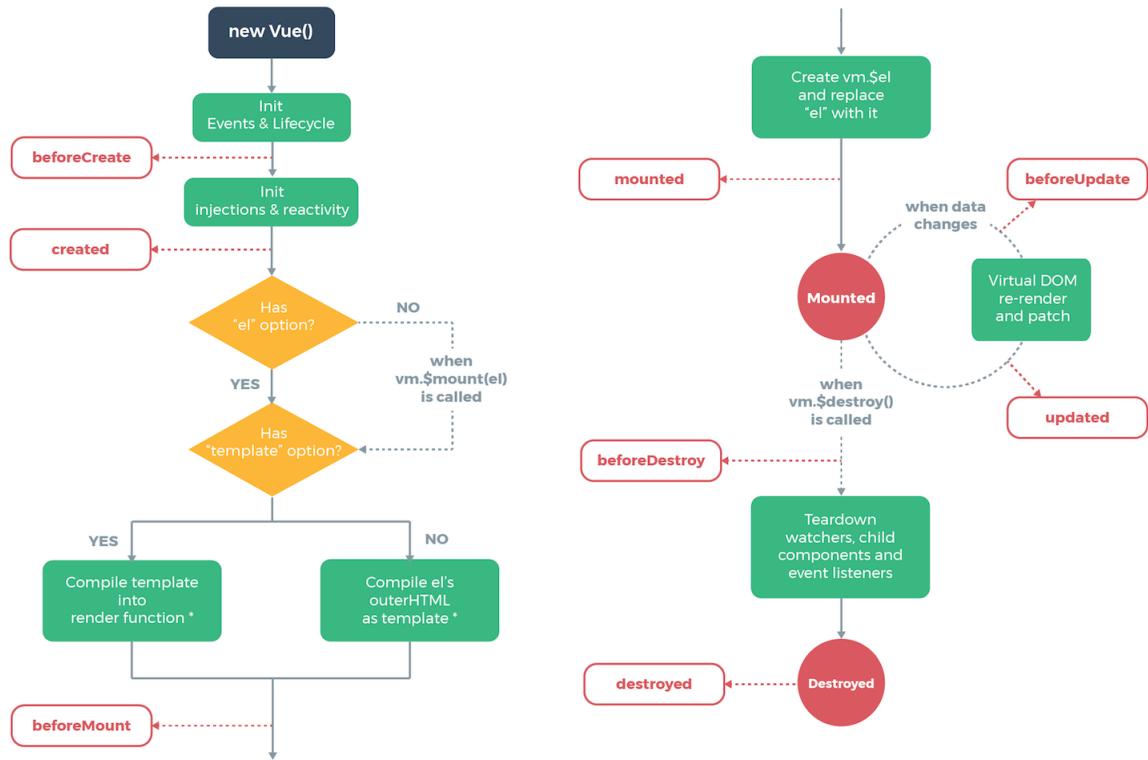


Abbildung 12: Diagramm des Ablaufs einer Vue Instanz

<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.

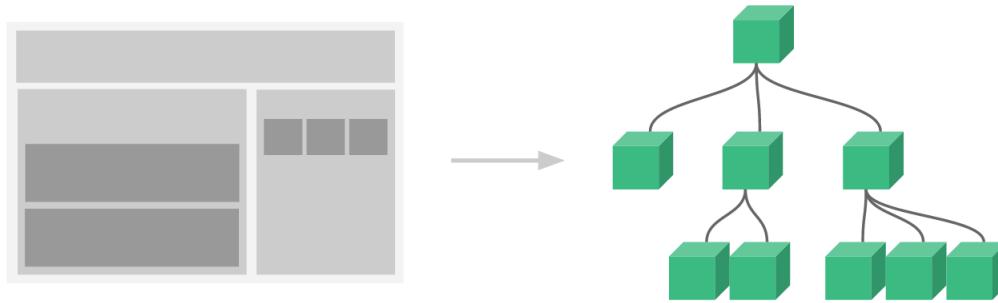


Abbildung 13: Darstellung von vernetzten Komponenten innerhalb einer Webseite
https://vuejs.org/images/components.png?_sw-precache=b5c08269dfc26ae6d7db3801e9efd296

3.18.2 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 22.96% Entwicklern weltweit verwendet [?]

Vue:

- Es gibt mehr als 2.071.882 Millionen Webseiten, die Vue benutzen
- Der Marktanteil von Vue bei Webseiten mit hohen Aufrufzahlen, beträgt nicht mehr als 0.8% [?]

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 länger existiert 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 an die Programmierung mit Java erinnert mit den Objekten, Abhängigkeiten, Konstruktoren und so weiter. 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.

3.19 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 klassische Programmiersprache, sondern eine statische Sprache, die zur Definierung benutzt wird, auch genannt deklarative Programmiersprache. Das HTML Gerüst wird von einem Browser eingelesen und dieser generiert dann auf der Basis des HTML Files eine Website. [?] [?]

Listing 13: HTML File Grundgerüst

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title> Titel der Webseite </title>
5      </head>
6      <body>
7          <h1> Überschrift </h1>
8      </body>
9  </html>

```

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

3.20 CSS

CSS (Cascading Style Sheets) ist die Sprache, die benutzt wird, um 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. [?]

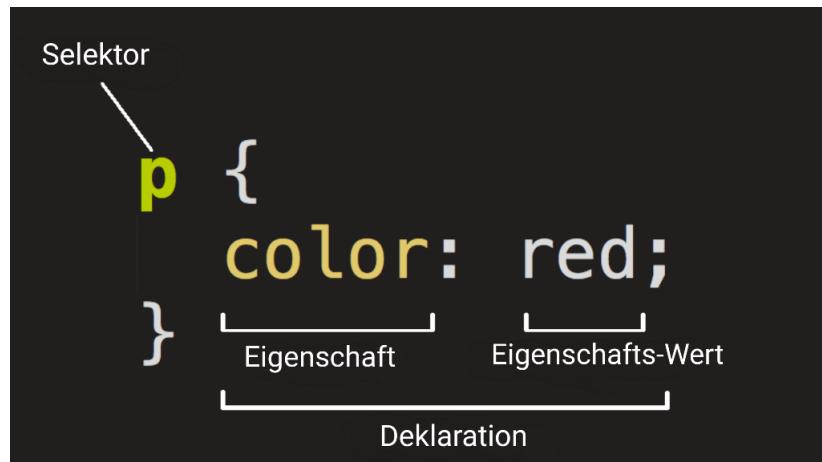


Abbildung 14: Aufbau einer CSS-Regel
<https://media.prod.mdn.mozilla.net/media/cache/2017/09/27/15467/3889d04d90c10b27e863c6850d588c43/css-example.png>

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. [?]

3.21 JavaScript

JavaScript ist eine leichtgewichtige Skriptsprache, die 1995 vom Softwareunternehmen Netscape entwickelt worden ist, um die Möglichkeiten von CSS und HTML zu erweitern. Bekannt ist sie insbesondere als Sprache für Webseiten geworden, jedoch wird sie auch in Bereichen abseits des Browsers oft benutzt, wie z.B. Servern.

Sie unterstützt imperative, objektorientierte als auch deklarative Programmierung (funktionales Programmieren). JavaScript folgt dem Standard ECMAScript, welchen alle Browser unterstützen. JavaScript dient dazu auf Webseiten Interaktionen von Benutzern auszuwerten. Veränderung, Nachladen oder Generierung von Inhalten gehört auch zu den Funktionen.

Jedoch sollte man JavaScript nicht mit Java, der Programmiersprache, vertauschen. Beide sind verschiedene Handelsmarken der Firma Oracle und ähneln einander höchstens, was die Syntax betrifft. [?] [?]

3.22 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. [?] [?]

3.22.1 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.

The screenshot shows the jwt.io interface. On the left, under 'Encoded' (PASTE A TOKEN HERE), there is a large text area containing a long string of base64-encoded data. On the right, under 'Decoded' (EDIT THE PAYLOAD AND SECRET), there are three sections: 'HEADER: ALGORITHM & TOKEN TYPE' containing the JSON header { "alg": "HS256", "typ": "JWT" }; 'PAYLOAD: DATA' containing the JSON payload { "id": "052", "name": "Max Mustermann", "rolle": "admin" }; and 'VERIFY SIGNATURE' containing the HMACSHA256 verification code: HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), geheimschlüssel). A checkbox for 'secret' is checked, and 'base64 encoded' is noted.

Abbildung 15: 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 auch möglich 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 die 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. [?]

3.22.2 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. Die JWS (JSON Web Signature) führt diese Überprüfungen aus.

Bei diesem Verfahren lässt sich die Payload nach Base64-Entschlüsselung klar und deutlich lesen. [?]

Signatur (JWS) und Verschlüsselung (JWE)

Es besteht die Möglichkeit als Zusatz zum JWS noch eine JWE (JSON Web Encryption) zu benutzen. JWE verschlüsselt den Inhalt des Payloads, diese werden danach mit JWS signiert. Damit die Entschlüsselung des Inhalts auch möglich ist, wird ein privater Schlüssel oder ein Kennwort mitgegeben.

Damit ist der Absender verifiziert, die Nachricht authentisch und der Payload ist nicht lesbar nach einer Base64-Entschlüsselung. [?]

3.23 Progressive Web App(PWA)

Webanwendung werden mit Hilfe von HTML, CSS und Javascript entwickelt. 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 aus. Dank ihm hat die PWA Zugriff auf Caching und kann Offline verwendet werden. Weiter noch stellt der Service-Worker die Funktionalität der Push-Notification bereit.

3.23.1 Vorteile

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.

3.23.2 Nachteile

Die Nachteile sind minimal. Doch einer der wichtigsten ist, dass die PWAs im Vergleich zu nativen Applikationen viel langsamer auf die Hardware Ressourcen zugreift.

Noch dazu haben zurzeit (Stand April 2022) PWAs keine volle Kompatibilität mit Apple. Im Gegensatz zu Chrome, Edge und Firefox kann der Browser von Apple, namens Safari, keine Push Notifikationen sowie Hintergrundprozesse bearbeiten.

3.23.3 Bekannte PWAs

Jeder hat schon einmal eine PWA benutzt, auch wenn er sich nicht bewusst war. Viele bekannte Firmen wie zum Beispiel Starbucks, BMW, Spotify und Pinterest verwenden solch eine.

Dabei sieht man wie unterschiedlich eine PWA verwendet werden kann. Zum Teil als Bestellapp, Informationsapp oder auch als Streaming Plattform.

3.24 Google Charts

[?] Google Charts ist eine leicht verwendbare Möglichkeit, Werte in verschiedenen Darstellungen anzuzeigen. Es gibt 18 verschiedene Arten von Diagrammen. Aber die meist benützen sind:

- Histogram
- Bar Charts
- Donut Chart
- Column Chart
- Pie Chart
- Area Chart

Zu jedem Diagramm gibt es auch eine *options*-Variable, welche das modifizieren ermöglicht. Man kann somit die Höhe, Breite, den Titel und die einzelnen Farben angeben. Viele solcher Features sind kostenpflichtige Angelegenheiten. Google Charts sind kostenfrei.

3.25 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 Fehlerquote dementsprechend niedrig. Man kann zwischen drei Ansichten auswählen, welche Code, Split oder Design wären. Diese vereinfachen parallele Arbeiten sehr. Die App wird aber nicht nach jeder Änderung neu gebaut, was wiederum die Schnelligkeit der Arbeit beeinflusst. Damit die Applikation eine schöne Oberfläche hat, stellt Jetpack Compose mehrere Themes zur 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.

3.25.1 Wichtig zu wissen!

[?] Um mit Jetpack Compose coole Applikationen entwerfen zu können, muss man die Anfangsschritte verstanden haben.

- Jetpack Compose wird mithilfe von *Android Studio* programmiert.
- Jetpack Compose basiert 100 Prozent auf Kotlin.
- Laufzeit und Speichernutzung der App verbessern.

3.25.2 Applikationsgröße

Jede Applikation benötigt Speicher. Aber wie schafft man es den Speicher einer App zu minimieren. Auf den Speicher haben Bibliotheken einen großen Einfluss.

3.25.3 Applikationsbauzeit

[?] Man kann mit einem einfachem Kommando die *Build Time* herausfinden.

```
1      ./gradlew --profile --offline --rerun-tasks --max-workers=4 assembleDebug
```

- *-profile* ermöglicht das Profilieren
- *-offline* verweigert das benützen von Abhängigkeiten aus dem Internet
- *-rerun-tasks* wird verwendet um *Gradle* um alle Prozesse zu wiederholen
- *-max-workers=4* beschreibt die Anzahl der parallelen Prozesse [?]
- *assembleDebug* erstellt eine *debug.apk* von der App

3.25.4 Entwurf

Jeder Entwurf sollte schön und benutzerfreundlich ausschauen. Somit kann man mit *Theming* ein solches erstellen. Dazu verwendet man meistens *Material Theming*. Natürlich kann man aber auch eigene Entwürfe erstellen und verwenden. Um Daten anzeigen lassen zu können, sollte man sich mit dem Thema *Data Binding* auseinander gesetzt haben. Dazu kommen noch die die Möglichkeiten Animationen, Grafiken und Gästen in die App einzubauen.

3.25.5 Data Binding

Data Binding ist eine sehr wichtige Funktion im Programmieren mit einer Benutzeroberfläche. Es wird verwendet, um das Backend zu informieren, falls sich ein Wert ändert. Durch das Data Binding kann man also Aktivitäten und Aktionen kontrollieren und ausnutzen. Um eine Variable binden zu können, muss man *mutableStateOf* benutzen.

Man unterscheidet in *One-Way Binding* und *Two-Way Binding*

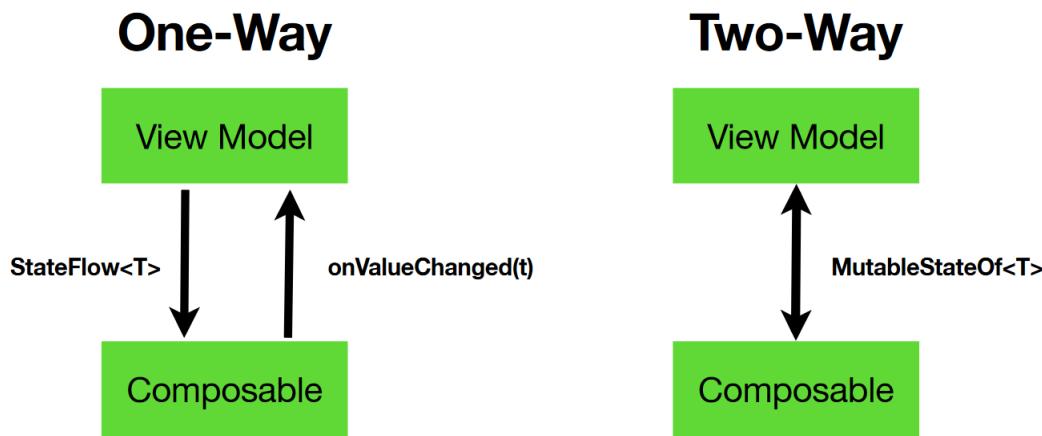


Abbildung 16: Data Binding

Es wird heutzutage mehr auf das Two-Way Binding zugegriffen, da es einfacher zu benutzen ist und man sich viel Code erspart.

3.26 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 kompatibel mit Java ist, können beim Testen die Frameworks und Bibliotheken von Java sehr weiterhelfen. Ursprünglich hat man den Kotlin Code in Bytecode übersetzt und diesen dann in der JVM laufen gelassen. Aber den Code kann man auch in Javascript umschreiben lassen und somit für Web Anwendungen verwenden.

Einer der Hauptvorteile von Kotlin sind die Unterstützungen für Multiplatform-Programmierung. Die Flexibilität sowie die Vorteile der nativen Programmierung bleiben erhalten, es reduziert sich aber der Umfang des Schreibens sowie des Programmiercodes.

3.26.1 Die Unterteilung der Multiplatform von Kotlin?

Man unterteilt in drei Abschnitte, mit *Common Kotlin* als Kern.

- JVM Code mit Kotlin/JVM
- JS Code mit Kotlin/JS
- Native Code mit Kotlin/Native

Der Kern *Common Kotlin* beinhaltet das Nötigste für das Programmieren. Wie zum Beispiel wichtige Bibliotheken und basis Tools.

3.27 XML

[?] [?] XML steht für eXtensible Markup Language und ist ein textformbasiertes Datenformat. Das Wort eXtensible heißt übersetzt erweiterbar und soll die Erweiterungsmöglichkeiten der Sprache beschreiben.

Sie wurde im Jahre 1996 entwickelt und wurde zwei Jahre später zum W3C-Standard. 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 heißt das XML auch plattformunabhängig ist. Wie in HTML, gibt es auch hier sogenannte Tags, die aber selbst zu konfigurieren sind. Der große Vorteil von XML ist, das es sehr leicht lesbar ist, sowohl von Mensch als auch von Maschine. Es wird in Android verwendet, um das Layout jeder Aktivität des Android zu entwerfen und erleichtert die Datenübertragung zwischen Anwendungen.

3.27.1 Wie funktioniert XML

Der Autor hat die zu benutzenden Tags selber zu definieren. Es werden keine vordefinierten Tags wie `<p1>` oder `<h1>` wie in HTML verwendet.

```
<note>
  <date>2015-09-01</date>
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```

Abbildung 17: XML Tags

3.27.2 Vergleich

Es gibt einerseits das XML-Schema, welches die neuere Generation benutzt. Aber auch das DTD-Schema. DTD steht für Documenttypedefinition und wurde früher benutzt, um den Typ eines Dokumentes zu bestimmen.

Angefangen wird solch ein Dokument wie folgt:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE note [
  <!ENTITY nbsp "&#xA0;">
  <!ENTITY writer "Writer: Donald Duck.">
  <!ENTITY copyright "Copyright: W3Schools.">
]>

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  <footer>&writer;&nbsp;&copyright;</footer>
</note>
```

Abbildung 18: DTD Schema

[?] So eine Dokumentendefinition ist in folgende Punkte gegliedert:

- Notationen
- Elementattributen
- Entitäten
- Elementtypen

DTD-Dokumente können auch *Paramtere-Entities* benutzen, die aber nur in *CDA-TA* (*Character Data*) oder *PCDATA* (*Parsed Character Data*) vorhanden sind. Mit Parameter-Entities wird es ermöglicht, fremde Files in das DTD einzubinden.

3.28 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.

3.28.1 Docker Compose

[?] Compose ist ein Tool zum Definieren und Ausführen von Docker-Anwendungen mit mehreren Containern. Um Container ausführen zu können, werden compose-files geschrieben um Docker-Anwendungen zu starten. Die Compose-File ist eine *.YAML* File, in der die konfiguration statt findet. Anschließend erstellen und starten Sie mit einem einzigen Befehl alle Dienste aus Ihrer Konfiguration.

3.28.2 Verwendung

Man setzt die Umgebung der App mit einer Dockerfile. Danach wird die *docker-compose.yaml* file erstellt, welche die einzelnen Dienste startet. Starten mit *docker-compose up*. Killen/Beenden mit *docker-compose down*.

3.28.3 Docker Container

[?] Einen Docker Container kann man sich als Schiffe, Lkws oder auch als Zug vorstellen. Der Container ist eine virtuelle Maschine, die aber von einzelnen Anwendungen bereitgestellt werden. Mithilfe von Docker Images werden Container erstellt. Man kann diese aber auch in neue Images speichern und intern die Prozesse steuern. Der Unterschied zu normalen virtuellen Maschinen ist das es kein eigenes Betriebssystem hat. Docker Container greifen auf die Linux-Hosts zu, falls vorhanden. Liegt aber kein Linux vor, so hilft der Hypervisor mit einem kleinen Linux-Kernel nach.

In einem Docker Container werden mehrere Anwendungen installiert. Da es mit Linux-Hosts arbeitet, werden die Anwendungen mittels *apt*, *apk* oder mittels *pip*, *npm*, *cargo*. Außerhalb von den externen Methoden gibt es auch die Möglichkeit mittels *make*, *mvn* Pakete und Programme zu installieren.

3.28.4 Docker Image

Ein Docker Image ist ein mehrschichtiges Modell, was zu einer Datei entworfen wird. Diese führt speziellen Code in einem Docker Container aus. Docker Images haben kein Verfalldatum. Die sind mehrfach und gleichzeitig auf mehreren Containern benutzbar.

eazy-menue-backend

Published on Jan 17 by Musikfreunde in eazy-menue-backend

eazy-menue-frontend

Published on Jan 17 by Musikfreunde in eazy-menue-frontend

Abbildung 19: Docker Images

3.28.5 Docker Images von EazyMenu

Das *eazy-menue-backend*-Image beinhaltet den Quarkus Server und die dazugehörige Keycloakanbindung. Das *eazy-menue-frontend*-Image enthält die Kantinenwebapp.

3.29 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 zur Vermeidung von wiederholten Anfragen. OkHttp hilft bei Verbindungsproblemen, indem man alternative Adressen benutzt und unterstützt weiters noch TLS-Funktionen. Der Client wird sich lautlos von Verbindungsproblemen erholen, sprich er bleibt bei Verbindungsproblemen noch vorhanden. Falls ein Dienst mehrere IP-Adressen verfügt, wird sich der Client automatisch alternative Adressen aussuchen und mit denen die Aufrufe noch einmal versuchen. Das braucht man häufig für IPv4 und IPv6, weil die in redundanten Rechenzentren behütet werden.

Es kann so konfiguriert werden, dass es auf eine breite Konnektivität zurückgreift. Der Client ist sehr gut für die Verwendung von blockierten synchronen und nicht blockierten asynchronen Aufrufen.

3.29.1 Was braucht man wenn man OKHttp benutzen will?

OkHttp funktioniert ab der Android 5.0 aufwärts und Java 8. Es hängt von Okio und der Kotlin Standardbibliothek ab.

Kotlin Standardbibliothek

[?] Damit man überhaupt mit Kotlin programmieren kann, bietet die Standardbibliothek *kotlin-stdlib* viele Packages an. Standardgemäß bietet diese die einzelnen Datentypen wie zum Beispiel:

- String
- Char
- Boolean
- Int
- Any
- etc.

Mit den mitgelieferten Packages wie zum Beispiel `kotlin.io` kann man ganz übliche Input/Output Funktionen verwenden. Mit dem Package `kotlin.collection` werden Schleifen zu Verfügung gestellt.

Wenn man die neueste Version von OkHttp sucht, ist man auf Maven Central genau richtig. Diese fügt man dann im build.gradle file ein und nach einem neu laden des Scriptes kann man es auch schon verwenden.

3.30 Retrofit

[?]

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

3.30.1 Verwendung

Wenn man Retrofit benützen möchte, muss man darauf achten immer eine *HTTP*-Annotation über die jeweilige Methode zu schreiben. Die verfügbaren Annotationen lauten:

- HTTP
- GET
- POST
- PUT
- DELETE
- OPTIONS
- HEAD

Zu den Annotationen kann man natürlich einen zugehörigen Pfad der Methode angeben.

```
@POST("persons/person")
```

Man kann auch jeweilige *Query*-Parameter mitgeben.

```
@POST("persons/person?id=1")
```

Um einen Body übergeben zu können, schreibt man vor dem Methodenparameter *@Body*.

```
@POST("person/new") Call<Person> addPerson(@Body Person person);
```

3.31 Gradle

[?] [?] Gradle ist ein 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 Bedingungsmöglichkeit von Maven. Der Unterschied zu den Maven-Projektdefinitionen ist, dass man Gradle-Scripts gleich ausführen kann.

Gradle unterstützt neben Java noch die Programmiersprachen Groovy und Kotlin. DAG wird verwendet, um die Tasks in richtiger Reihenfolge zu dirigieren. Heutzutage hat Gradle eine eigene Maschine 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.

Man kann das Script in verschiedenen Punkten aufteilen und bearbeiten:

- Plugins
- Android
- Dependencies

3.32 AsciiDoc

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

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

Der größte Vorteil an AsciiDoc ist, dass es gut mit Github verwaltbar ist. Noch dazu braucht man keinen besonderen Editor für AsciiDoc.

3.33 Asciidoctor

Asciidoctor ist eine open source Textverarbeitung, welche AsciiDoc Text in HTML 5, PDF oder in anderen Formaten umwandelt.

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

3.34 PlantUML

PlantUML ist ein open-source tool, welches hauptsächlich zum Modelieren von Diagrammen verwendet wird. Mithilfe von PlantUML hat man die möglichkeit verschiedenste Diagramme zu erstellen wie zum Beispiel:

- UML Class Diagram
- Archimate
- Block diagram
- BPMN
- C4
- Computer network diagram
- ERD
- Gantt chart
- Mind map

Für unserer Projekt haben wir PlantUML hauptsächlich nur für das Klassen Diagramm verwendet. [?]

3.34.1 UML Class diagram

PlantUML ist für den Menschen einfach leesbar und nicht schwierig zum Anwenden.
Ein Beispiel für ein Klassen Diagramm sieht folgendermaßen aus:

Listing 14: Simple PlantUML example

```

1      @startuml
2      class Menue {
3          private Long id;
4          private String canteenDesc;
5          private String serviceDesc;
6          ...
7      }
8
9      class Kantine {
10         private Long id;
11         private String canteenDesc;
12         private String serviceDesc;
13         ...
14     }
15
16     class Bestellung {
17         private Long id;
18         private String orderedBy;
19         private String orderedFor;
20         ...
21     }
22
23
24     class Oeffnungszeiten {
25         private Long id;
26         private char status;
27         private String timeWindowFrom;
28         ...
29     }
30
31     Bestellung    "1"-l-* Menue
32
33     Bestellung    *--"1" Oeffnungszeiten
34
35
36     Oeffnungszeiten *-l-"1" Kantine
37
38     Menue *--"1" Kantine
39
@enduml

```

Das Klassen diagramm sieht dann in etwa so aus:

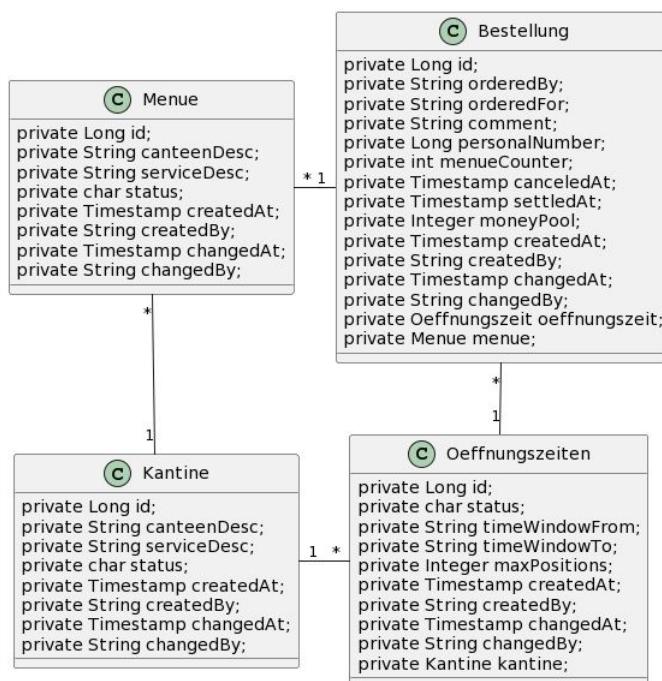


Abbildung 20: PlantUML

3.35 Swagger

Swagger ist ein Open-Source Werkzeug welches verwendet wird um HTTP-Webservices zu entwerfen, dokumentieren 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 verwenden kann, muss man in den Application.Properties folgende Zeile eintragen:

quarkus.swagger-ui.always-include=true

Die Swagger Page wird dann unter folgender URL abgerufen:

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

3.35 Swagger

Bei der Swagger Page hat man die Möglichkeit, alle Requests zu sehen die verwendet werden.

The screenshot shows the Swagger UI interface for the **eazy-menue-backend API**. At the top, it displays the API name, version (1.0.0-SNAPSHOT), and an OAS3 badge. Below the header, there are four main service sections: **Bestellung Service**, **Kantine Service**, **Menue Service**, and **Oeffnungszeit Service**. Each section lists various API endpoints with their methods (e.g., GET, POST, PUT) and URLs. The endpoints are organized into collapsible rows, indicated by a caret icon at the end of each row.

- Bestellung Service**
 - GET /menue/bestellung
 - PUT /menue/bestellung
 - POST /menue/bestellung
 - GET /menue/bestellung/categories
 - GET /menue/bestellung/categories/{name}
 - GET /menue/bestellung/stats/categories/{name}
 - GET /menue/bestellung/stats/{name}
 - GET /menue/bestellung/{name}
- Kantine Service**
 - GET /menue/kantine
- Menue Service**
 - GET /menue/menus
 - PUT /menue/menus
 - POST /menue/menus
 - GET /menue/menus/recommendation
 - GET /menue/menus/{date}
- Oeffnungszeit Service**
 - GET /menue/oeffnungszeiten
 - GET /menue/oeffnungszeiten/{id}

Abbildung 21: SwaggerUI

Mit Swagger hat man auch die Möglichkeit einzelne Data Transfer Objects veranschaulichen, welche in den Requests verwendet wird.

The screenshot shows the Swagger UI interface with a sidebar on the left containing navigation links like Home, API, Definitions, and Help. The main content area is titled "Schemas". It displays two separate schema definitions:

```
BestellungDTO ↴ {  
    id          integer($int64)  
    orderedBy   string  
    amount      integer($int32)  
    comment     string  
    menuId      integer($int64)  
    orderedFor  string  
    timeId      integer($int64)  
    personalNummer integer($int64)  
}  
  
MenueDTO ↴ {  
    id          integer($int64)  
    date        string  
    code        string($byte)  
    appetizer   string  
    mainDish    string  
    dessert     string  
    categories  string  
}
```

Abbildung 22: SwaggerUI

4 Implementierung

4.1 Systemarchitektur

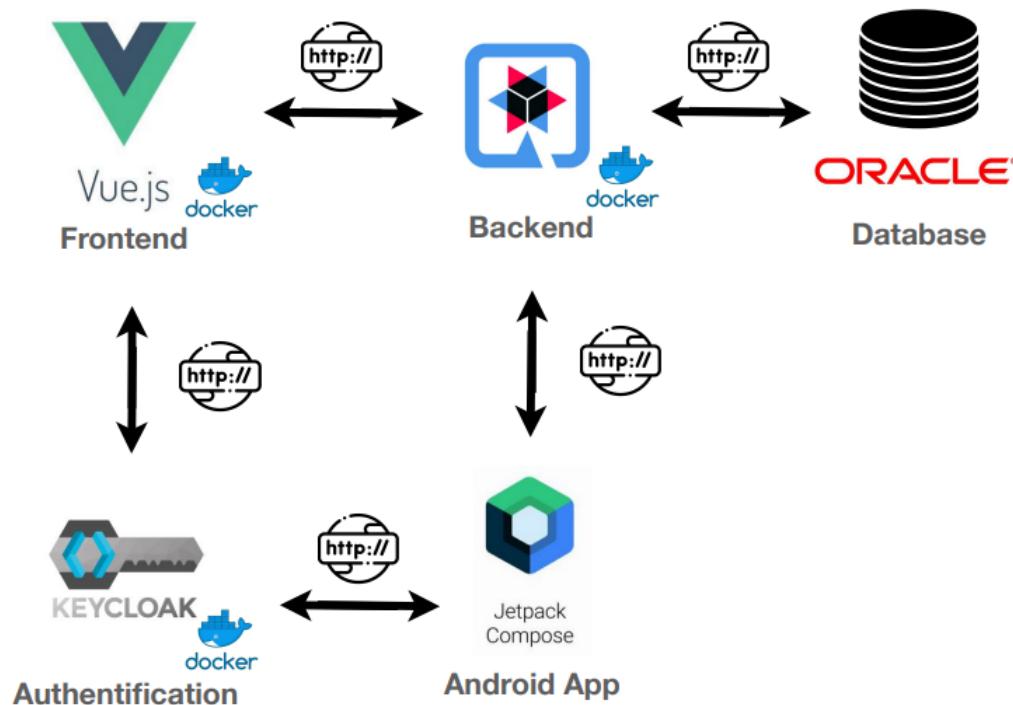


Abbildung 23: Systemarchitektur des Programms

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.

4.2.1 Use Cases

Kantinenarbeiter

- Neue Menüs anlegen
 - Kantinenmitarbeitende können 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 Kantinenmitarbeitenden sollen eine Übersicht, die 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.

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 obigen 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.

4.2.2 Planung

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

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.

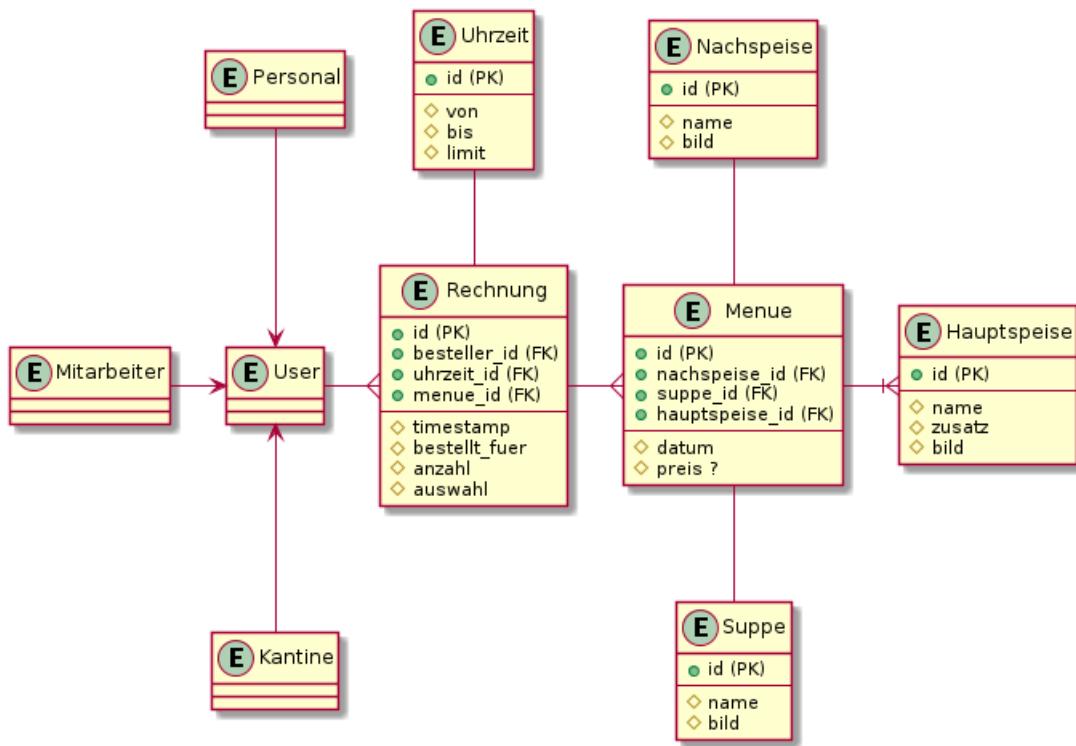


Abbildung 24: Erste Version des Datenmodells

Die zweite 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. Die Benutzertabellen sind noch leer, da zu dem Zeitpunkt noch nicht feststand ob diese Tabellen wirklich benutzt werden sollen, da es bereits eine Datenbank für Mitarbeiter gab in der Firma.

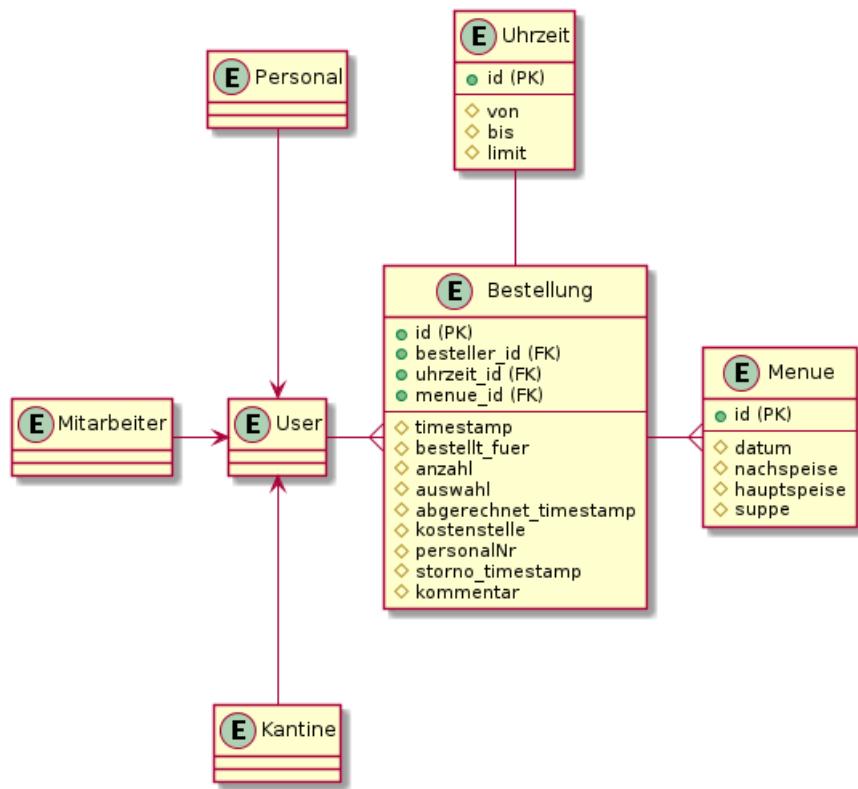


Abbildung 25: Zweite Version des Datenmodells

In der finalen Version wurde entschieden auf die Benutzertabellen zu verzichten, da die Mitarbeiter mittels Keycloak ihre Anmelddaten übermitteln. Daher reicht auch nur die Personalnummer in der Bestellung-Tabelle. Der Menue-Tabelle wurde ein Kategorien-Attribut hinzufügt. Dies dient dazu, um jedes Menü in bestimmte Kategorien einzuteilen. Ebenfalls zu erkennen sind neue Felder wie “GEANDERTUM” oder “ANGELEGTVON”. Diese wurden ergänzt, um Logdaten direkt in der Tabelle einsehen zu können. Die Felder werden automatisch ausgefüllt, nachdem eine Transaktion auf die Tabellen erfolgt.

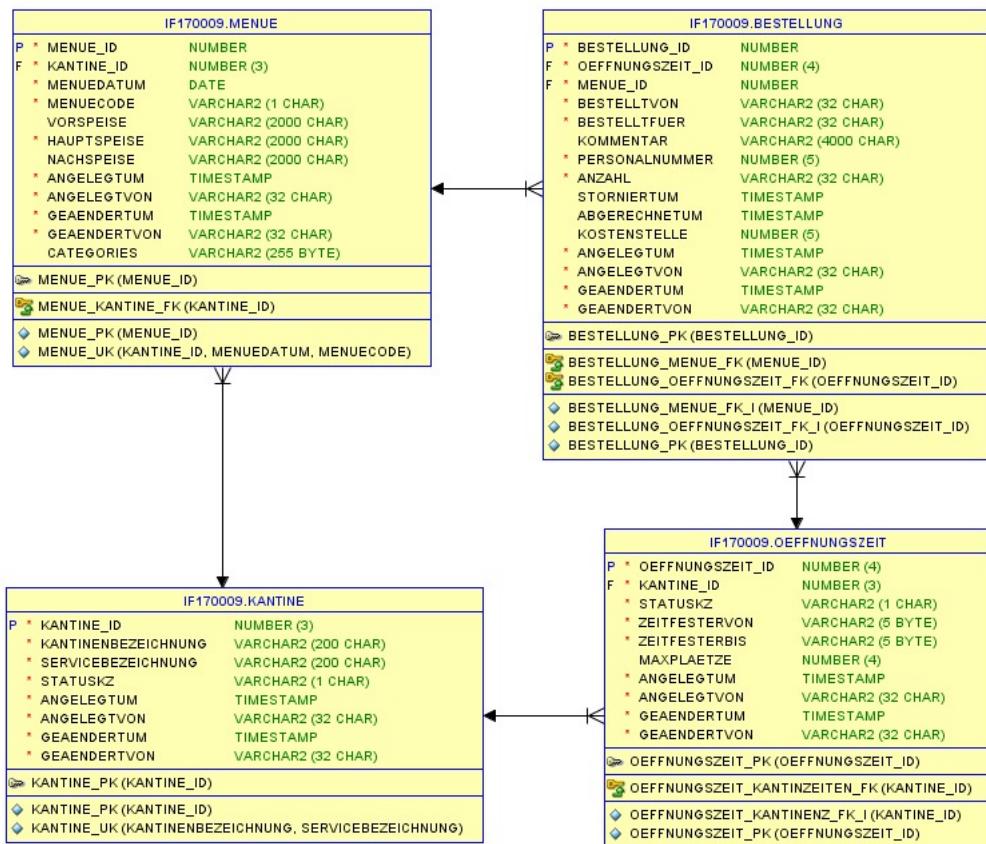


Abbildung 26: Finale Version des Datenmodells

4.2.3 Entitäten

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

- Bestellung
- Categories
- Kantine
- Menue
- Oeffnungszeit

4.2.4 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.

4.2.5 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 Mahlzeit. Somit kann man ganz einfach zwischen einzelnen Mahlzeiten unterscheiden und sie auch gruppieren.

4.2.6 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.

4.2.7 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 weitere 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.

4.2.8 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.

4.3 REST-Schnittstellen

4.3.1 Allgemein

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

Wir verwenden 13 Endpoints:

Bestellung

Typ	Path	Info
GET	/bestellung/username	Bestellungen von einem Mitarbeiter
GET	/bestellung/stats/username	Anzahl der Bestellungen pro Wochentag eines Mitarbeitern
GET	/bestellung/categories	Alle Kategorien
GET	/bestellung/categories/username	Alle Kategorien von einem bestimmten Mitarbeiter
GET	/bestellung?date=datum	Alle Bestellungen an einem bestimmten Tag
POST	/menue/bestellung	Bestellung erstellen
POST	/menue/bestellung	Bestellung stornieren

Menue

Typ	Path	Info
GET	/menues/menus	Alle Menüs
POST	/menues/menus	Menü erstellen
POST	/menues/menus	Menü verändern
GET	/menu/menus/recommendation	Menü-Vorschlag (Recommendation)
GET	/menues/menus/<date>	Alle Menüs an einem bestimmten Tag

Öffnungszeiten

Typ	Path	Info
GET	/menues/oeffnungszeiten	Alle aktiven Essenszeiten

4.3.2 Bestellung**Bestellungen von einem Mitarbeiter**

Um zu sehen welche Bestellungen ein Mitarbeiter 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 Resultat für den Mitarbeiter **spabo** sieht folgendermassen aus:

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

Output:

```

1 {
2   "createdAt": "2021-11-26T15:33:33.62898Z[UTC]",
3   "id": 1315,
4   "menuDate": "2021-11-29",
5   "menuName": "Spaghetti",
6   "orderedFor": "spabo",
7   "timeWindow": "11:15 - 11:45"
8 }
```

Die Bestellungen werden nach dem Datum, an dem eine Bestellung angelegt wurde, sortiert. Somit werden immer die neusten Bestellungen gleich am Anfang 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 Resultat für den Mitarbeiter **besbe** sieht folgendermaßen aus:

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

Output:

```
1 [  
2 {  
3   "amount": 2,  
4   "weekday": "Montag"  
5 },  
6 {  
7   "amount": 1,  
8   "weekday": "Mittwoch"  
9 },  
10 {  
11   "amount": 1,  
12   "weekday": "Donnerstag"  
13 },  
14 {  
15   "amount": 0,  
16   "weekday": "Dienstag"  
17 }  
18 ]
```

Alle Kategorien

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

Anhand der Kategorien, wird mit Hilfe eines Algorithmus entschieden, welche Mahlzeit am besten zu einem Mitarbeiter passt. Es wird darauf geschaut, welche Kategorie am häufigsten vorkommt.

Die verwendbaren Kategorien sehen folgendermassen aus:

URL: GET <http://localhost:8080/menue/bestellung/categories>

Output:

```
1 [  
2   "Vegetarisch",  
3   "Vegan",  
4   "Schwein",  
5   "Rind",  
6   "Huhn",  
7   "Pute",  
8   "Salat",  
9   "Nudel"  
10  "Fisch",  
11  "Sonstiges"  
12 ]
```

Das wird hauptsächlich verwendet, um bei dem Frontend alle Kategorien anzuzeigen die für eine bestimmte Mahlzeit 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 hat, werden auch keine Kategorien zurückgegeben. In dem folgenden Beispiel kann man aber sehen, wie das Resultat für den Mitarbeiter **spabo** aussieht:

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

Output:

```
1 [  
2   "Vegetarisch",  
3   "Nudel",  
4   "Vegan",  
5   "Schwein",  
6   "Salat",  
7   "Pute"  
8 ]
```

Alle Bestellungen an einem bestimmten Tag

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

Es kann aber auch sein, dass 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 den **16.03.2022**.

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

Output:

```

1  [
2    {
3      "code": "A",
4      "date": "2022-03-16",
5      "menue": "Schnitzel",
6      "menueCounter": 1,
7      "orderedFor": "spabo",
8      "personalNumber": 1023,
9      "timewindow": "11:15 - 11:45"
10 },
11 [
12   {
13     "code": "C",
14     "date": "2022-03-16",
15     "menue": "Eisbergsalat",
16     "menueCounter": 1,
17     "orderedFor": "spabo",
18     "personalNumber": 1023,
19     "timewindow": "12:00 - 12:30"
20   },
21 ...
22 ]

```

Bestellung erstellen

Wenn ein User eine Bestellung aufgeben 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:

```

1  {
2    "orderedBy": "spabo",
3    "amount": "1",
4    "comment": "",
5    "menueId": 707,
6    "orderedFor": "spabo",
7    "timeId": 6,
8    "personalNummer": 1023
9  }

```

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 überhaupt schon gab, muss der Pfad `/menus/-menues` aufgerufen werden.

Ein Beispiel für ein Menue sieht folgendermassen aus:

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

```
1 [  
2 {  
3     "appetizer": "Suppe",  
4     "categories": "Vegan;Vegetarisch;Salat",  
5     "code": "C",  
6     "date": "2022-03-22",  
7     "dessert": "Milka",  
8     "id": 706,  
9     "mainDish": "Kartofell Salat"  
10 },  
11 {  
12     "appetizer": "Suppe",  
13     "categories": "Vegan;Vegetarisch;Salat",  
14     "code": "C",  
15     "date": "2022-03-22",  
16     "dessert": "Milka",  
17     "id": 705,  
18     "mainDish": "Kartofell Salat"  
19 },  
20 ...  
21 ]
```

Menü erstellen

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

Da eine Bestellung aus genau 3 Menus 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/menus/`

JSON Objekt für Menue A:

```
1 {
2   "id":null,
3   "date":"2022-03-22",
4   "code":"A",
5   "appetizer":"Suppe",
6   "mainDish":"Pasta",
7   "dessert":"Milka",
8   "categories":"Rind;Nudel"
9 }
```

JSON Objekt für Menue B:

```
1 {
2   "id":null,
3   "date":"2022-03-22",
4   "code":"B",
5   "appetizer":"Suppe",
6   "mainDish":"Lachs",
7   "dessert":"Milka",
8   "categories":"Fisch"
9 }
```

JSON Objekt für Menue C:

```
1 {
2   "id":null,
3   "date":"2022-03-22",
4   "code":"C",
5   "appetizer":"Suppe",
6   "mainDish":"Kartoffel Salat",
7   "dessert":"Milka",
8   "categories":"Vegan;Vegetarisch;Salat"
9 }
```

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

Menü verändern

Wie auch bei dem Erstellen eines Menüs, wird der Pfad `/menue/menus` 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/menus/`

JSON Objekt für Menue A:

```
1 {
2   "id":null,
3   "date":"2022-03-22",
4   "code":"A",
5   "appetizer":"Suppe",
6   "mainDish":"Pasta",
7   "dessert":"Milka",
8   "categories":"Rind;Nudel"
9 }
```

JSON Objekt für Menue B:

```
1 {
2   "id":null,
3   "date":"2022-03-22",
4   "code":"B",
5   "appetizer":"Suppe",
6   "mainDish":"Lachs",
7   "dessert":"Milka",
8   "categories":"Fisch"
9 }
```

JSON Objekt für Menue C:

```
1 {
2   "id":null,
3   "date":"2022-03-22",
4   "code":"C",
5   "appetizer":"Suppe",
6   "mainDish":"Kartofell Salat",
7   "dessert":"Milka",
8   "categories":"Vegan;Vegetarisch;Salat"
9 }
```

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.

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

Der Algorithmus sieht folgendermassen aus:

URL: PUT <http://localhost:8080/menue/menus/recommendation>

```
public Menue getRecommendation(String name, String date){
    List<String> categories =
        bestellungRepository.getAllCategoriesByUsername(name);
    List<Menue> menus = getMenusByDate(date);
    Menue recommendedMenue = null;

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

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

Um herausfinden zu können welche Mahlzeit optimal wäre, werden mit Hilfe des Algorithmus alle Kategorien des Users aus der Datenbank geholt. Jeder User der eine Bestellung getätigt hat, hatte die Möglichkeit zu sehen, welche Kategorien die Mahlzeit auch besitzt. Somit kann man leicht herauszufinden welche Kategorien auch am häufigsten in der Bestellhistorie vorkommen.

Unser Algorithmus würde nicht funktionieren, gäbe es keine Bestellungen oder Mahlzeiten würden keine Kategorien besitzen. 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/menus/<date>** aufgerufen.

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

URL: PUT <http://localhost:8080/menue/menus/2021-12-08>

```

1  {
2    [
3      {
4        "appetizer": "Nudelsuppe",
5        "categories": "Schwein",
6        "changedAt": "2021-12-07T19:29:08.840936Z[UTC]",
7        "changedBy": "IF170009:IF170009:beni",
8        "code": "A",
9        "createdAt": "2021-12-07T18:37:30.322514Z[UTC]",
10       "createdBy": "IF170009:beni",
11       "date": "2021-12-08",
12       "desert": "Obst",
13       "id": 499,
14       "kantine": {
15         "canteenDesc": "Betriebskueche",
16         "changedAt": "2021-11-25T21:29:47.471379Z[UTC]",
17         "changedBy": "IF170009:beni",
18         "createdAt": "2021-11-25T21:29:47.471196Z[UTC]",
19         "createdBy": "IF170009:beni",
20         "id": 3,
21         "serviceDesc": "Mittagstisch - CORONA LIGHT",
22         "status": "A"
23       },
24       "mainDish": "Schweineschnitzel"
25     },
26     {
27       "appetizer": "Nudelsuppe",
28       "categories": "Rind;Nudel",
29       "changedAt": "2021-12-07T19:29:08.995487Z[UTC]",
30       "changedBy": "IF170009:IF170009:beni",
31       "code": "B",
32       "createdAt": "2021-12-07T18:37:30.490055Z[UTC]",
33       "createdBy": "IF170009:beni",
34       "date": "2021-12-08",
35       "desert": "Obst",
36       "id": 500,
37       "kantine": {
38         "canteenDesc": "Betriebskueche",
39         "changedAt": "2021-11-25T21:29:47.471379Z[UTC]",
40         "changedBy": "IF170009:beni",
41         "createdAt": "2021-11-25T21:29:47.471196Z[UTC]",
42         "createdBy": "IF170009:beni",
43         "id": 3,
44         "serviceDesc": "Mittagstisch - CORONA LIGHT",
45         "status": "A"
46       },
47       "mainDish": "Lasagne"
48     },
49     ...
50   ]
51 }
```

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.

4.3.4 Öffnungszeiten

Alle aktiven Essenszeiten

Um zu sehen ob noch freie Sitzplätze zur Verfügung stehen, wird der Pfad `/menue/öffnungszeiten` aufgerufen.

In unserer Kantine gibt es 4 Zeiten an denen man essen gehen kann. Diese dauern immer eine halbe Stunde und haben eine Zwischenpause von 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/menus/2021-12-08`

```
1  [
2      [
3          {
4              "chosen": false,
5              "freeSeats": -1,
6              "id": 6,
7              "maxSeats": 20,
8              "time": "11:15 - 11:45"
9          },
10         {
11             "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

Eine Authentifizierung ist eine Benutzeranfrage an den Server. Nach einem versuchtem Login, sucht man in der Benutzer-Datenbank nach den Werten die man eingetragen hat. Die häufigsten Protokolle die für das Internet sind:

- Http
- Https

Früher war der Benutzer verpflichtend sich immer wieder neu anzumelden. Heute gibt es sogenannte "Access-Tokens", die ebenso für die Benutzer Existenz Abfrage verwendet wird.

In unserer Diplomarbeit wird Keycloak als Server verwendet. Mittels dem erstellten Keycloak Realm für die Diplomarbeit, stehen in diesem alle User mit den zugeteilten Rollen drinnen. Nach einem Anmeldeversuch wird nachgefragt ob der User vorhanden ist. Somit authentifiziert sich der User zu dem Server. Danach erfolgt die Authorisierung.

Bei so einer Authentifizierung spricht man über *Single Core Authentifizierung*. Dabei hat man nur den Benutzer und das jeweilige Kennwort zu Verfügung.

[?] Möchte man aber ein stärkere Authentifizierung haben, so benötigt man mehrere Authentifizierungsfaktoren.

- Wissensfaktor
- Besitzfaktor
- Inhärenzfaktor

Beim **Wissensfaktor** beschreibt alle Arten von Authentifizierung, die ein Benutzer kennt. Ein Benutzer kennt zum Beispiel seinen Username und seinen Pin-Code.

Anders ist es beim **Besitzfaktor**, welcher mehr auf "Etwas, das man hat" sich fokussiert. Es geht um Gegenstände wie zum Beispiel eine Smartwatch, Handy oder jegliche Chips dessen Funktion ist einen OTP zu erzeugen und immer wieder verwenden zu können.

Ein OTP ist ein *One Time Password*. Es ist in der Regel sicherer als ein, vom Server oder Benutzer erstelltem, statischem Password.

Fast jedes moderne Handy hat heutzutage einen **Inhärenzfaktor**. Das ist der Faktor, der auf Fingerabdrücke sowie Gesichtserkennung Wert liegt. Bei einer Gesichtserkennung werden virtuell mit Infrarotlicht tausende Punkte auf das Gesicht konstruiert. Daraus entwickelt sich ein 3D Modell, welches die Höhen, Tiefen und viele weitere Eigenschaften des Gesichtes aufnimmt und analysiert.

4.5 Interface Webapp

4.5.1 Planung

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

Zeit	freie Plätze
11:30-12:00	51
12:00-12:30	51
12:30-13:00	51
13:00-13:30	51

Abbildung 27: UI-Prototypen für den Bestellvorgang

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 gesehen werden

können.

Um den Rest des Platzes auszunutzen wurde eine Menüauswahl geplant, die so viel Informationen wie möglich darstellen kann, ohne den Benutzer zu überfordern. Nach langem 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.

The image displays two wireframe prototypes. The top prototype is a login screen featuring a user icon, two input fields labeled 'USERNAME' and 'PASSWORD', and a blue 'LOGIN' button. The bottom prototype is a dashboard titled 'Menübestellung' with a sidebar showing 'Übersicht' and 'Verlauf'. The main area contains a search bar, a filter dropdown, and a table with columns 'Tag', 'Menü', and 'bestellt am, um'. One row in the table shows 'Di. 06.07.2021', 'Schnitzel RisiBisi', and '06.07.2021 08:53:42'. At the bottom right is a red 'STORNO' button with a circular arrow icon.

Abbildung 28: UI-Prototypen für 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 ausschaut, 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.

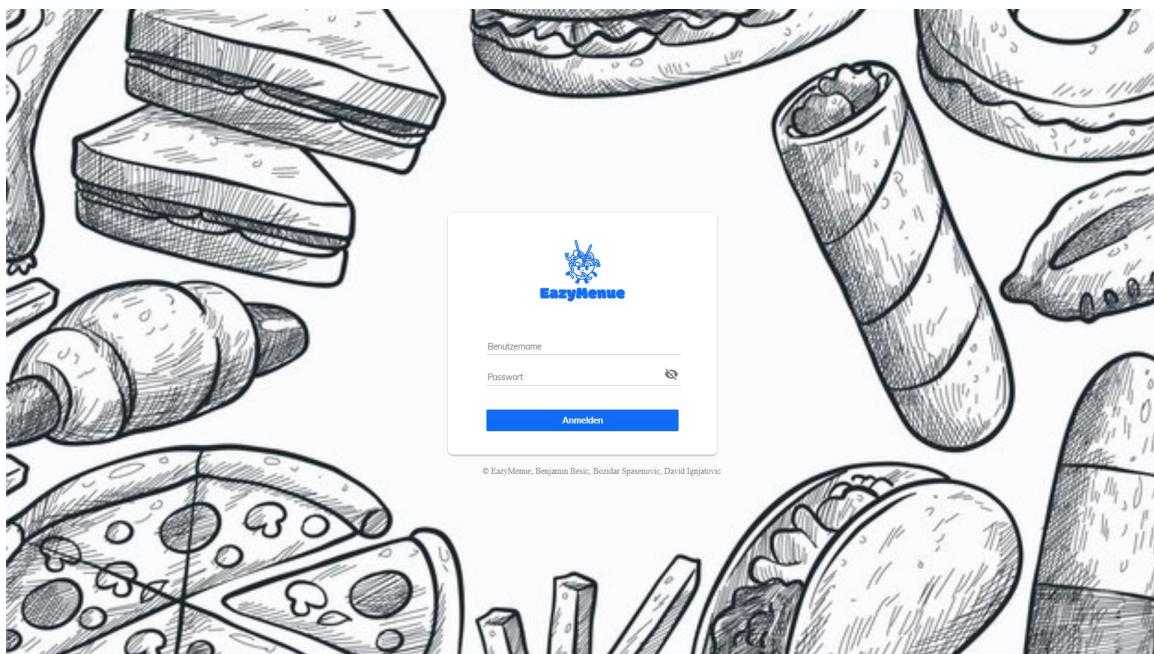


Abbildung 29: Login Screen

4.5.3 Mitarbeiter-Ansicht

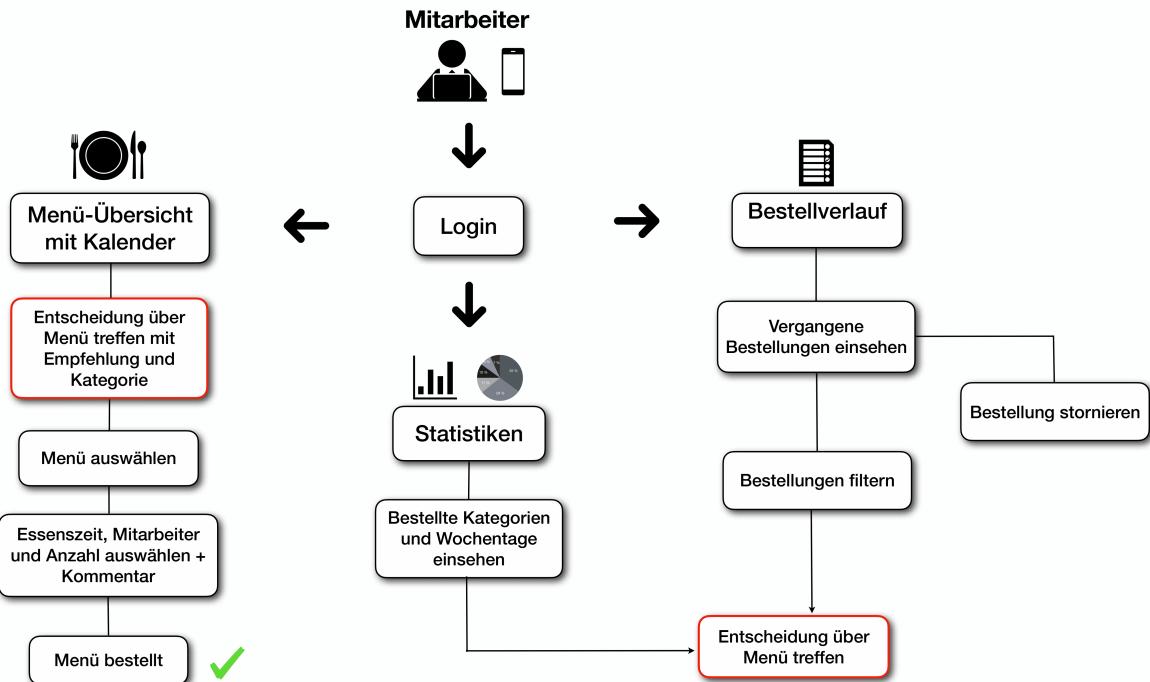


Abbildung 30: Die wichtigsten Aktivitäten der Mitarbeitenden

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, dabei steht auch 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.

4.5 Interface Webapp

The screenshot shows the home view for an employee. At the top, there is a date picker showing "Mittwoch, 23. Februar 2022". Below it is a calendar for February 2022. The 23rd is highlighted with a blue circle. The days of the week are labeled Mo through So. The menu items are displayed in three columns:

- A**: Pizza (green icon), Bestellen, ?
- B**: Spaghetti (black icon), Bestellen, ?
- C**: Eisbergsalat (black icon), Bestellen, ?

At the bottom, there is a note: "Für Fragen nach den Speisen enthaltenen Allergenen steht Ihnen unser geschultes Küchenfachpersonal zu Verfügung." followed by a question mark icon.

Abbildung 31: Home Ansicht eines Mitarbeiters

This screenshot shows a dropdown menu titled "Kategorien" (Categories) with the following options:

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

Abbildung 32: 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 einen 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.

Ersteller	Zeit	Auswahl	Freie Plätze
spabo	11:15 - 11:45	<input type="checkbox"/>	20
Menü am:	12:00 - 12:30	<input type="checkbox"/>	20
Mo., 7. Feb. 2022	12:45 - 13:15	<input type="checkbox"/>	20
Menü:	13:30 - 14:00	<input type="checkbox"/>	20
Puten Cordon Bleu			
Anzahl:			
1			
Anzahl:			
-			
+			
Für wen?			
spabo			
Kommentar:			
Dein Kommentar ist uns wichtig!			
Abschließen		Abbrechen	

Abbildung 33: 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 gelegenen 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.

Suche: Name	Filter: Name		
Menüdatum	Menüname	Zeit	Erstellt am, um
Mo., 7. Feb. 2022	Puten Cordon Bleu	12:00 - 12:30	2022-02-03T22:57:18.590748Z[UTC]
Mo., 31. Jan. 2022	Gnocchi	13:30 - 14:00	2022-01-27T16:18:44.542041Z[UTC]
Mo., 31. Jan. 2022	Gnocchi	13:30 - 14:00	2022-01-27T16:18:47.760049Z[UTC]
Mo., 31. Jan. 2022	Gnocchi	13:30 - 14:00	2022-01-27T15:51:48.556739Z[UTC]
Mo., 31. Jan. 2022	Gnocchi	11:15 - 11:45	2022-01-27T15:51:13.454731Z[UTC]
Mo., 31. Jan. 2022	Gnocchi	13:30 - 14:00	2022-01-27T22:56:26.055348Z[UTC]
Mo., 6. Dez. 2021	Griechischer Salat	11:15 - 11:45	2021-11-26T16:43:34.561392Z[UTC]
Mi., 1. Dez. 2021	Schnitzel	11:15 - 11:45	2021-11-26T16:42:59.794713Z[UTC]

« 1 2 »

[Storno](#)

Abbildung 34: 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.

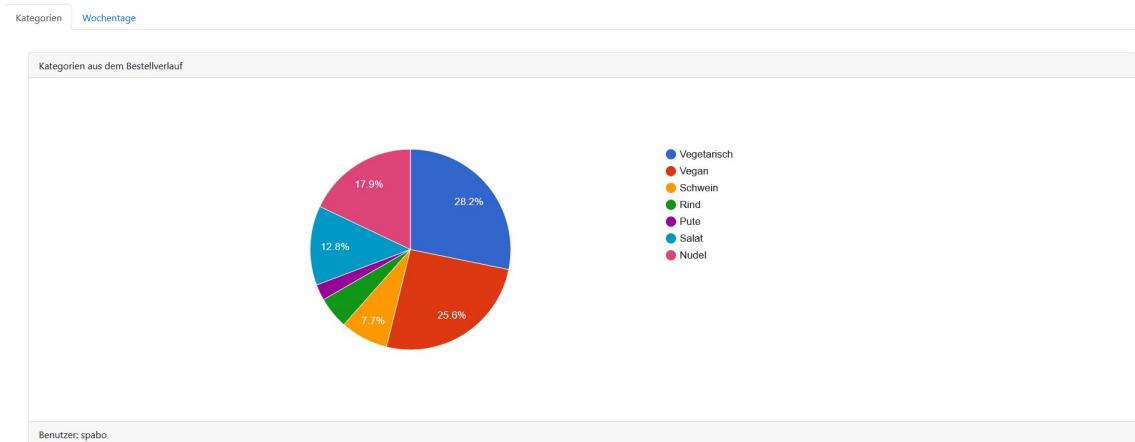


Abbildung 35: Kategorie Statistiken

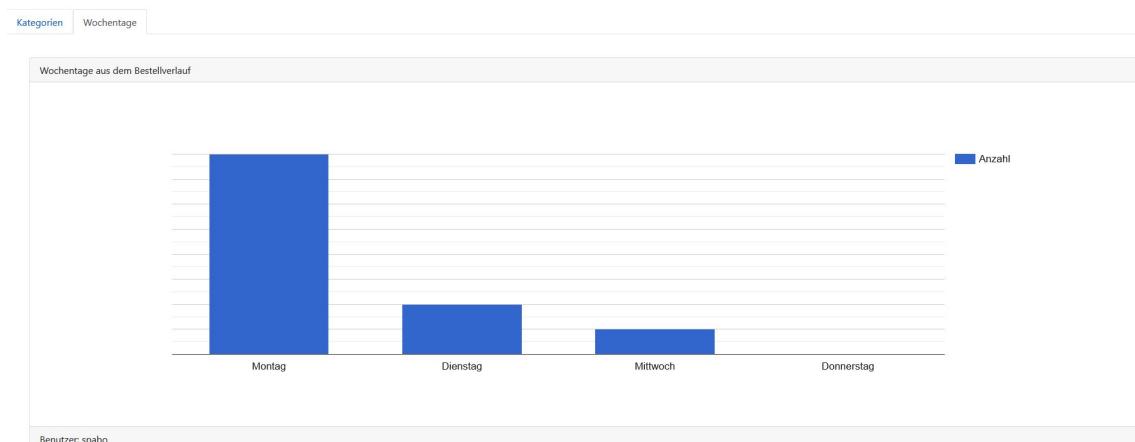


Abbildung 36: Wochentage Statistiken

4.5.4 Kantinen-Ansicht

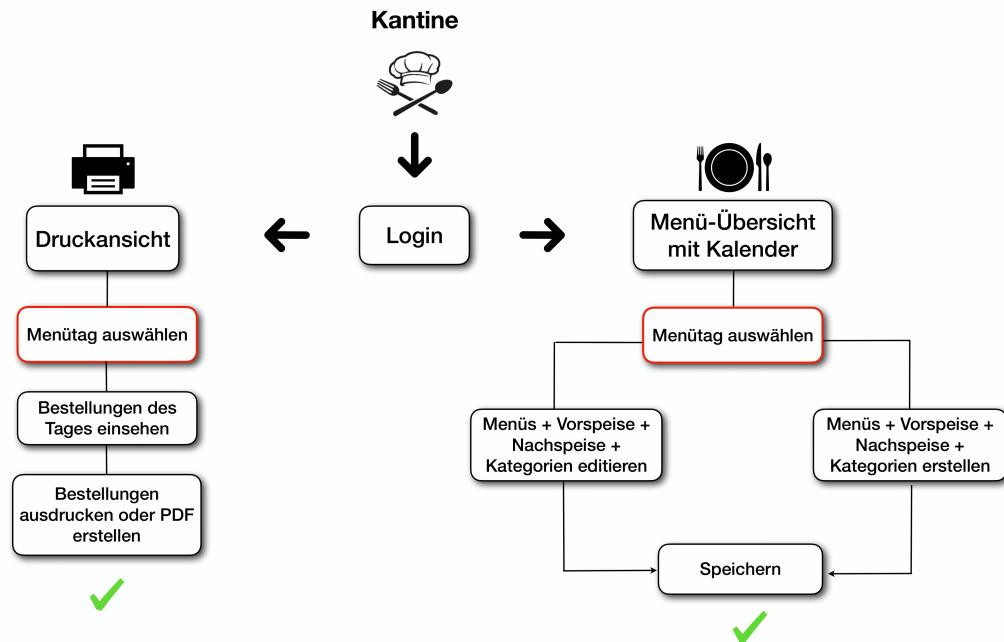


Abbildung 37: Die wichtigsten Aktivitäten der Kantine

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.

4.5 Interface Webapp

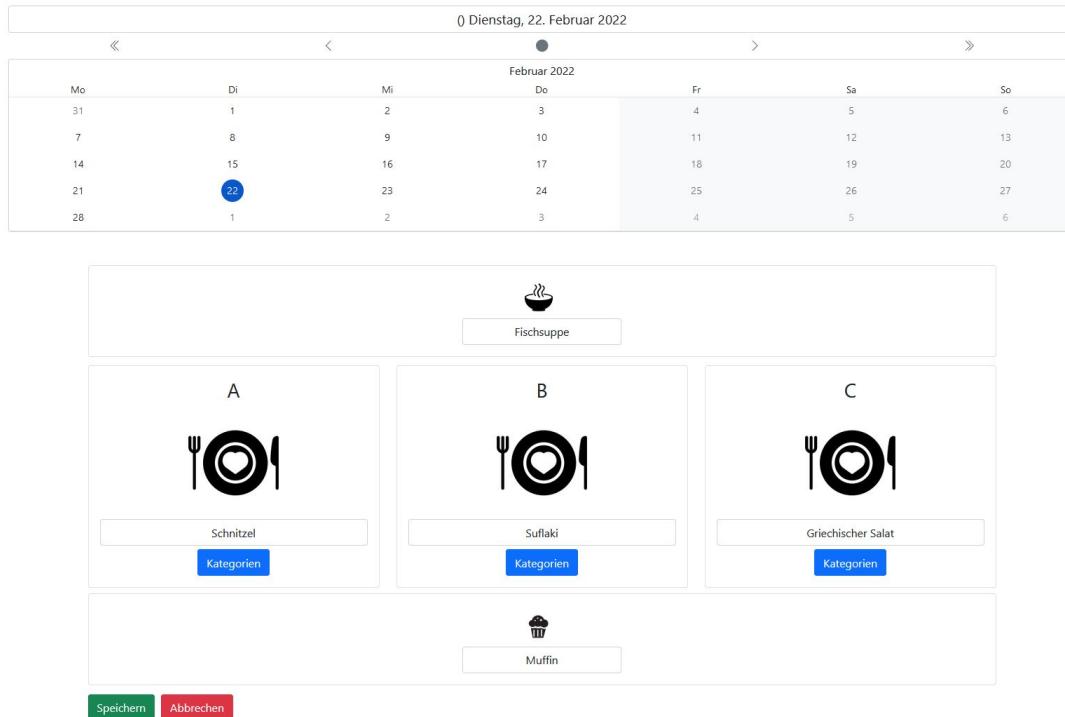


Abbildung 38: Kantine Homeansicht

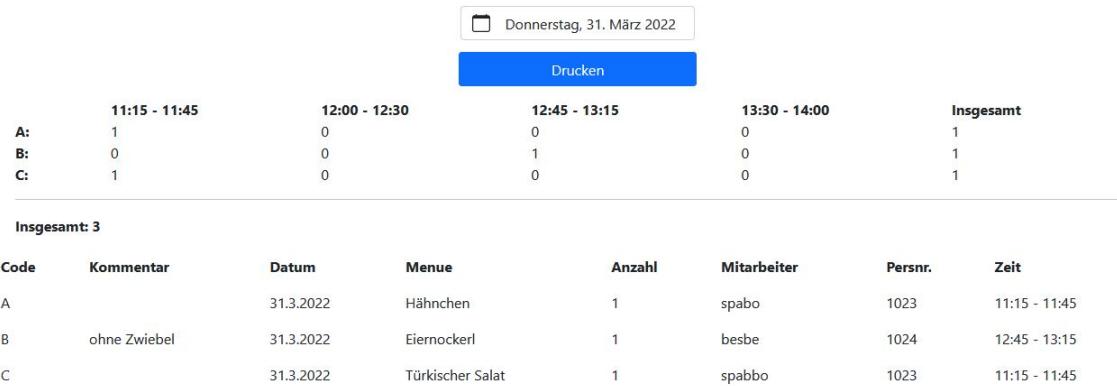


Abbildung 39: Kategorieauswahl

Drucken

Die Druckansicht dient den Kantinemitarbeitern dazu, um die Bestellungen des jeweiligen Tages 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.



The screenshot shows a web-based printing interface for a canteen. At the top, there is a date field showing "Donnerstag, 31. März 2022" and a blue "Drucken" button. Below this is a summary table:

	11:15 - 11:45	12:00 - 12:30	12:45 - 13:15	13:30 - 14:00	Insgesamt
A:	1	0	0	0	1
B:	0	0	1	0	1
C:	1	0	0	0	1

Below the summary table is a section titled "Insgesamt: 3" which contains a detailed list of orders:

Code	Kommentar	Datum	Menue	Anzahl	Mitarbeiter	Persnr.	Zeit
A		31.3.2022	Hähnchen	1	spabo	1023	11:15 - 11:45
B	ohne Zwiebel	31.3.2022	Eiernockerl	1	besbe	1024	12:45 - 13:15
C		31.3.2022	Türkischer Salat	1	spabbo	1023	11:15 - 11:45

Abbildung 40: Druckansicht der Kantine



The screenshot shows a PDF document titled "menuebestellung-webui" with the URL "http://localhost:8083/drucker". The PDF contains the same data as the screenshot above, including the summary table and the detailed order list.

Abbildung 41: PDF der Ansicht

4.6 Android-App

4.6.1 Android Manifest

Alle Android Projekte haben eine *AndroidManifest.xml* Datei im Root-Verzeichnis. Diese wird benötigt, da sie Auskunft über die App für folgende Systeme bieten muss:

- Buuild-Tools
- Geräte mit Android Betriebssystemen
- Google Play

In der Datei werden jene Komponenten angegeben, welche alle Dienste und Aktivitäten beinhaltet. Zu jeder Komponente gibt es Eigenschaften, wie zum Beispiel die Attribute einer Klasse oder den Klassennamen, die man von der jeweiligen Kotlin Klasse bekommt.

Komponenten von *AndroidManifest.xml*:

- <activity>
- <service>
- <receiver>
- <provider>

Dazu kommen noch Spezifikationen gegenüber der Hardware, was soviel bedeutet wie die Auswahl von Smartphones oder Tablets auf der die App installiert werden kann. Neben den Komponenten, gibt es auch die Rechtevergabeung. Es wird bestimmt auf was die App alles Zugriff hat, sowie was eine andere App benötigt um mit der kommunizieren zu können. Es ist sehr wichtig bei einer App die mit Internet in Verbindung gebracht wird folgende Zeile hinzuzufügen.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Konventionen

In der *AndroidManifest.xml* gibt es sogenannte Konventionen die man einhalten muss. Von den Elementen her gibt es zwei die enthalten sein müssen. Das wären <*manifest*> und <*application*>. Diese Zwei dürfen auch nur einmal vorkommen.

Die Konvention für Attribute ist, dass vor jedem Attribut *android:* davor zu stehen hat. Attribute sind aber generell nicht notwendig und können weggelassen werden.

Es gibt auch die Ressourcen Werte. Diese Werte sind jene Werte die dem Benutzer gezeigt werden. Um Hardcoding zu vermeiden, werden solche Variablen in Ressource-Dateien gespeichert. Diese Datei ist leicht modifizierbar und wird auch sofort in der App übernommen. Ein Beispiel wäre:

```
<activity android:icon="@drawable/logo" ... >
```

4.6.2 Testing

Beim Erstellen von einem Android Projekt, werden noch zwei Packages automatisch dazu erstellt, welche vorgefertigte Tests beinhalten. Diese sollen dazu dienen, eigene Tests schreiben zu können und somit Die App auf Fehler und Aktivität zu prüfen.

4.6.3 Properties

Gradle hat eigene *gradle.properties* und *gradle-wrapper.properties*.

[?] Gradle-Properties bieten eine simple Alternative um Builds zu adaptieren, die in manchen Lagen anders ausgeführt werden müssen.

Auf welche Arten kann man die Properties bearbeiten?

- gradle.properties File

name=name

- Kommandozeile

-PpropertyName=propertyValue

- Properties von Java

-Dorg.gradle.project.name=name

- Umgebungsvariablen

4.6.4 Aufbau

Die Android App ist in verschiedene 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/menus/"  
  
    @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. 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

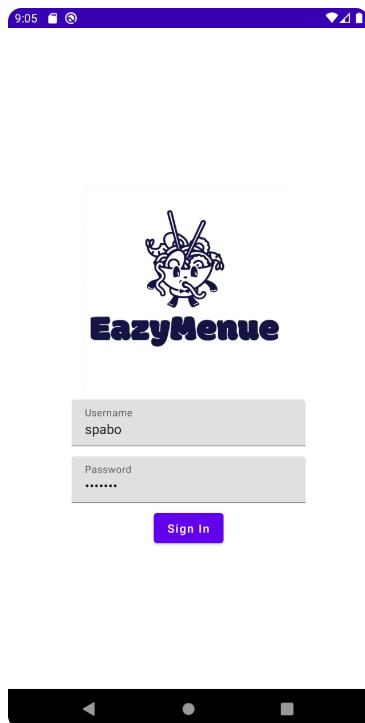


Abbildung 42: Loginscreen Android

Für die Login und somit Anfangsansicht, wurde ein simples Design ausgewählt. Wie in der Abbildung zu sehen ist, wird das Logo von EazyMenu gezeigt. Das Bild wurde mittels generierter Kotlin Klasse *R* in der Ansicht angezeigt.

```
Row() {
    Image(
        painterResource(R.drawable.logo_eazy_menue),
        contentDescription = "",
        contentScale = ContentScale.Inside,
        modifier = Modifier.padding(end = 10.dp)
    )
}
```

Painter Resource

[?] Dabei wird über die *painterResource*-Methode das jeweilige Bild ausgesucht mittels der Id von dem Bild. Die Ressourcen mit der angegebenen Id dürfen entweder PNG-, JPG-Dateien oder VectorDrawable-XML-Assets sein.

Neben Bildern hat man noch die Möglichkeit auch auf andere Ressourcen zuzugreifen wie zum Beispiel:

- Strings

- Icons
- Dimensionen
- Farben
- animierte Bilder und Icons

Unter dem Logo sind zwei Inputfelder zu sehen, welche für den Benutzernamen und Password zuständig sind. Damit man sich einloggen kann, wird der Knopf *Sign in* bereitgestellt. Nach dem klicken auf den Button folgt die Authentifizierung.

4.7.1 Authentication

Die Authentifizierung erfolgt mit der Methode `checkAccessToken`, welche nichts anderes macht wie einen Request zum Keycloak-Server zu schicken. Im Request wird nach einem Access-Token gefragt. Ein User ist vorhanden wenn ein AccessToken im Response vorhanden ist.

```
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. Wird im Response ein Token zurückgegeben, so wird er gespeichert. In diesem Schritt werden gleichzeitig mehrere Threads verwendet und deswegen einen countDownLatch eingebaut. Er ist eine Hilfe für das Warten zwischen einzelnen Threads. Er wartet also vor der Rückgabe der Methode darauf, dass der Token gespeichert wird.

```
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

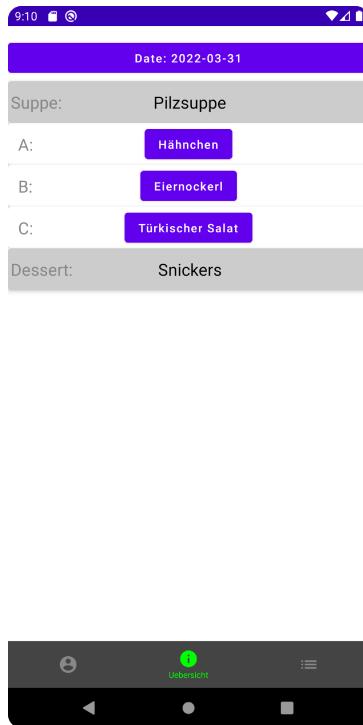


Abbildung 43: Übersicht Android

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 getMenusForDate(date: String) übergeben. Anschließend wird der Wert in der Übersicht angezeigt und man wird wieder zurück navigiert.

```
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}"
        }
    }
getMenusForDate(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 vor 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 die Darstellung zu ändern.

- modifier
- shape
- backgroundColor
- contentColor
- border
- elevation
- content

Dargestellt wird es mit einer einfachen Funktion.

```
Card {
    Text(
        text = "Card"
    )
}
```

Um die einzelnen Modifizierungen einzubauen, gibt man der Funktion die jeweiligen Parameter mit.

```
Card(  
    modifier: Modifier = Modifier,  
    shape: Shape = RectangleShape(5.dp),  
    backgroundColor: Color = Color.Transparent,  
    contentColor: Color = Color.Red,  
    elevation: Dp = 10.dp  
) {  
    Text(  
        text = "Card"  
    )  
}
```

4.8.3 Log out

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

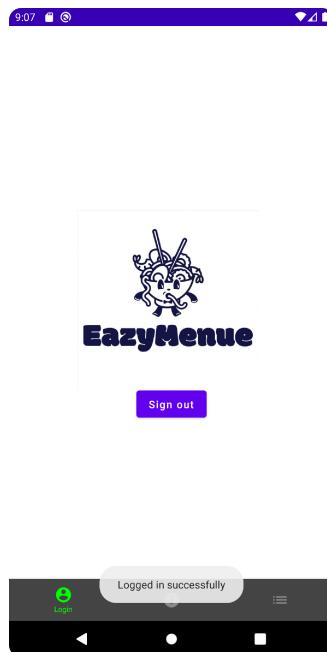


Abbildung 44: Signout Android

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 =
                    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 verwenden 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. Abschließend 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. Durch das Draufklicken auf die Mülltonne, wird die Bestellung storniert.

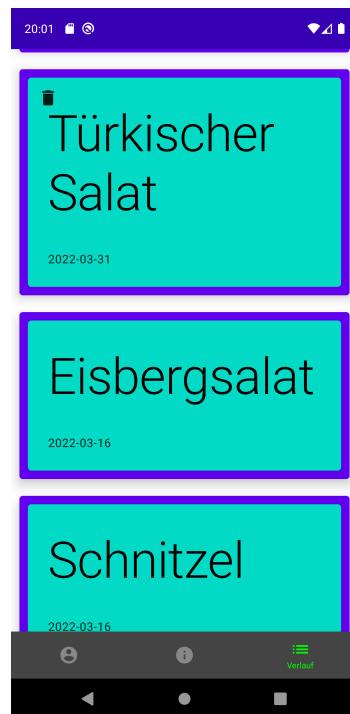


Abbildung 45: VerlaufAndroid Android

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.

Ist der Löschkvorgang erfolgreich, wird eine animierte Löschfunktion dem User gezeigt. Dazu wird *AnimatedVisibility* benutzt.

4.9.1 Animation

Jetpack Compose bietet programmieren viele Möglichkeiten der App ein benutzerfreundliches Erscheinungsbild zu verleihen. Am besten wirken Animationen auf einen Benutzer.

Wird der Inhalt geändert, benutzt man `AnimationVisibility` oder `AnimatedContent`. Dabei wird erstmals geschaut, ob das zu löschen Item noch enthalten ist. Wichtig ist es

```

AnimatedVisibility(
    visible = !deletedItem.contains(bestellung),
    enter = expandVertically(),
    exit = shrinkVertically(
        animationSpec = tween(
            durationMillis = 1000
        )
    )
) { this: AnimatedVisibilityScope
    Card(
        modifier = Modifier
        shape = MaterialTheme.shapes.small,
        backgroundColor = MaterialTheme.colors.secondary
    ) {...}
}

```

Abbildung 46: Bestellansicht Android

eine *mutableListOf<>* zu verwenden um bei jeder Änderung die Liste aktualisieren zu können.

```

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.

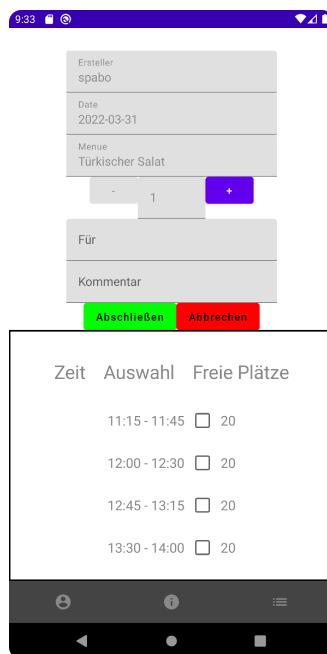


Abbildung 47: Bestellansicht Android

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)
            }
        }
    }
})

```

Interface Callback

[?] Da Java keine *Pointer* Konzepte verwendet, gibt es die Callbackfunktion nicht wie in C oder C++. Da hingegen gibt es das sogenannte Callback Interface. Jede Schnittstelle hat einen Speicherort. Diese wird dann, anstatt der Funktion selbst, übergeben und verwendet.

onResponse()

Die onResponse Methode ist eine abstrakte, vom Interface Callback vorgegebene Hilfsmethode. Diese wird aufgerufen wenn ein Response erfolgreich war. Solange der *ResponseBody* noch besteht, so ist der Response noch vorhanden. Um ihn zu löschen, hat man den Body vorerst zu schließen.

onFailure()

Die onFailure Methode ist ebenso eine abstrakte, vom Interface Callback vorgegebene Hilfsmethode. Diese wird bei einem nicht erfolgreichem Response aufgerufen. Ursachen wären Internetprobleme von der Server als auch Client Seite.

5 Zusammenfassung

5.1 Zusammenfassung

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.

Das Ergebnis ist eine Webapp mit zwei Arten von Usern. Die User bekommen durch ihren Bestellverlauf vorgeschlagene Menüs angezeigt. Der User kann dadurch auch gleichzeitig sehen, welche Arten von Gerichten er bevorzugt.

5.2 Ausblick

Die weiteren Schritte für das Projekt ist die Freigabe für mehrere Firmen mit einer Kantine. Um das zu schaffen benötigt man einen Server über den man einen Docker-Container mit dem Backend laufen lässt um danach die Webapp im Browser aufrufen zu können. Noch ein weiterer Schritt wär es die Webapp auf mehrere Sprachen zu erweitern um auch auf internationaler Ebene eine Verwendung zu finden.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

.1 Protokolle

Datum	25-10-2021
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	Das Programm ist lauffähig und erfüllt alle usecases.
Bis zum nächsten mal	Schriftliche Diplomarbeit

Tabelle 1: Protokoll 25-10-2021

Datum	12-11-2021
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	Diagramme wurden gezeigt mit dummy werten.
Bis zum nächsten mal	Food Recommender, Android, Schriftlicher Teil

Tabelle 2: Protokoll 12-11-2021

Datum	26-11-2021
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	erste Version Android, Recommender, schriftlicher Teil
Bis zum nächsten mal	Android: restlichen Fragmente, logik frontend mit backend verbinden grafiken mit neuem backend statistiken versuchen den schriftlichen Teil für einen ersten Überblick fertigstellen

Tabelle 3: Protokoll 26-11-2021

Datum	10-12-2021
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	Jetpack compose Es wurde das neue Projekt gezeigt, welches von XML auf Jetpack compose umgeschrieben. Die App ist zurzeit mit statischen werten befüllt. Problem ist das Jetpack compose relativ neu ist, und wenig informationsquellen online zur Verfügung stehen. 1.2. Backend Backend ist im groben und ganzen fertig. Logic für die Statistiken wird noch erledigt. 1.3. Recommender Frontend wurde fertiggestellt für den Recommender.
Bis zum nächsten mal	Statistik (Backend/Frontend) wird fertiggestellt Android App wird erweitert

Tabelle 4: Protokoll 10-12-2021

Datum	06-01-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	<p>Es wurde das neue Projekt gezeigt, welches von XML auf Jetpack compose umgeschrieben. Die App ist zurzeit mit statischen werten befüllt. Problem ist das Jetpack compose relativ neu ist, und wenig informationsquellen online zur verfügung stehen.</p> <p>Backend ist im groben und ganzen fertig. Logic für die Statistiken wird noch erledigt.</p> <p>Frontend wurde fertiggestellt für den Recommender.</p>
Bis zum nächsten mal	<p>Android app muss noch fertig gemacht werden. (Spasenovic)</p> <p>KeyCloak mit JetPackCompose</p> <p>Statistiken anzeigen</p> <p>Recommender und Statistiken evt. erweitern (Ignjatovic und Besic)</p>

Tabelle 5: Protokoll 06-01-2022

Datum	28-01-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	<p>KeyCloak funktioniert jetzt</p> <p>Bestellungen werden angezeigt</p> <p>Probleme</p> <p>Beim Bestellen kommt ein fehler (unsupported mediatype).</p> <p>Login button muss 2 mal gedrückt werden.</p> <p>möglicher Fehler: api läuft asynchron</p> <p>hartkodierte urls vermeiden</p>
Bis zum nächsten mal	<p>Schriftlich weiterschreiben / Beisc, Ignjatovic (evt. Spasenovic)</p> <p>verwendete Technologien</p> <p>implementierung</p> <p>Android Projekt verschönern (Spasenovic)</p> <p>Automatisierte Test</p>

Tabelle 6: Protokoll 28-01-2022

Datum	04-02-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	<p>Was funktioniert:</p> <p>Login noch nicht ganz fertig - KeyCloak fehlt</p> <p>Logik hinter der App funktioniert</p> <p>Was geht leider noch nicht</p> <p>Post funktioniert nicht ganz</p> <p>Statistiken in der Android App gehen nicht</p>
Bis zum nächsten mal	<p>Kotlin Login verbessern / Spasenovic</p> <p>Backend Bug fixen / Ignjatovic</p> <p>schriftlicher Teil (Implementierung) / evt (alle)</p>

Tabelle 7: Protokoll 04-02-2022

Datum	23-02-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	<p>Was geht nicht</p> <p>Probleme mit dem formbody in postBestellung</p> <p>403 MediaType not supported</p> <p>Was fehlt noch</p> <p>Bestellung</p> <p>eventuelle verschönerungen</p> <p>Was wurde gemacht</p> <p>Login gefixt mit CountDownLatch()</p> <p>Verlauf</p>
Bis zum nächsten mal	<p>Schriftlicher Teil / alle</p> <p>Implementierung / Verwendete Technologien</p> <p>Bestellung fixen / Spasenovic</p> <p>Android</p> <p>Unit Test / Ignjatovic</p> <p>Backend</p>

Tabelle 8: Protokoll 23-02-2022

Datum	17-03-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	bestellung funktioniert jetzt fehler war falscher name des Atributes personalnummer und orderFor war null
Bis zum nächsten mal	Android bessere package namen verwenden Ip Addressen nicht hardcoden Schriftlich Planung liegt verloren sagt wenig aus Datenmodel Diagramme Datenmodel Diagramme zur Implementierung verschieden. Eventuell Planung zur Implementierung. Mockups erweitern mit Text. UseCases zur Implementierung Backend Swagger wäre nicht schlecht und dann im schriftlichen Teil einbauen Im schriftlichen Teil Weniger Code bei attribute

Tabelle 9: Protokoll 17-03-2022

Datum	25-03-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	Request in eigen File geschrieben Component im schriftlichen Teil beschrieben
Bis zum nächsten mal	abstract zusammenfassung hinten ausführliche zusammenfassung könnte mehrere Seiten haben

Tabelle 10: Protokoll 25-03-2022

Datum	01-04-2022
Anwesende	Benjamin Besic, Bozidar Spasenovic, David Ignjatovic
Zwischenstand	Github Actions/Packages geschrieben Implementierung geschrieben
Bis zum nächsten mal	

Tabelle 11: Protokoll 01-04-2022