
# Software Entwicklung 2

SOMMERSEMESTER 2020

Illya Geybo, Torben Ziegler, Angelo Vetrano | Restaurantverwaltung

https://gitlab.mi.hdm-stuttgart.de/tz023/se2-projekt.git

# Dokumentation

## 1. Kurzbeschreibung

Während des zweiten Semesters des Studiengangs Mobile Medien an der Hochschule der Medien, sind wir angehalten gemeinsam in einer Gruppe ein Projekt zu erarbeiten. In unserem Fall handelt es sich hierbei um eine Restaurantverwaltungs-App.

Herrscht Hochbetrieb in Restaurants, so kann es durchaus vorkommen, dass eine vom Kunden aufgegebene Bestellung aufgrund von fehlenden Zutaten nicht zubereitet werden kann. Mit der ursprünglichen Idee sich diesem Problem zu widmen und eine App zu entwickeln, die Servicekräfte beim Bestellvorgang unterstützt, indem aufgegebene Bestellungen zuvor mit dem Lagerbestand abgeglichen werden, hat sich unsere App zu einer Restaurantverwaltung entwickelt, die Kunden die Möglichkeit gibt ihre Bestellungen selbstständig aufzugeben und Managern die Möglichkeit gibt Lagerbestände einzusehen und dieses bei Bedarf zu füllen.

Die App startet mit einem Anmeldefenster, in dem man sich mit Benutzernamen und Passwort anmeldet, gefolgt von einer Küchen- und Standortauswahl. Momentan hat man die Auswahl zwischen französischer, deutscher und italienischer Küche. Zu jeder Küche gibt es jeweils drei Standorte. Mit dem Benutzernamen customer und dem Passwort password meldet man sich als Kunde an, mit dem Benutzernamen manager und dem Passwort password meldet man sich als Manager an. War die Anmeldung erfolgreich, so gelangt man, abhängig vom angemeldeten Nutzer, entweder in die Ansicht für Kunden oder Manager.

Als Kunde bekommt man nach dem Anmelden die Speisekarte des ausgewählten Restaurants angezeigt. Während der Auswahl kann der Kunde entspannt die jeweilige Landesmusik hören. Hat man als Kunde die Bestellung aufgegeben, so findet ein Abgleich mit den Lagerbeständen statt. Abhängig davon, ob die Lagerbestände für die aufgegebene Bestellung ausreichen, wird ein Hinweis für eine erfolgreich aufgegebene Bestellung oder für nicht in ausreichender Menge verfügbaren Lagerbestände angezeigt.

Als Manager bekommt man nach dem Anmelden einen Überblick über die aktuellen Lagerbestände, bei eingetroffenen Lieferungen können diese aufgefüllt werden.

## 2. Startklasse

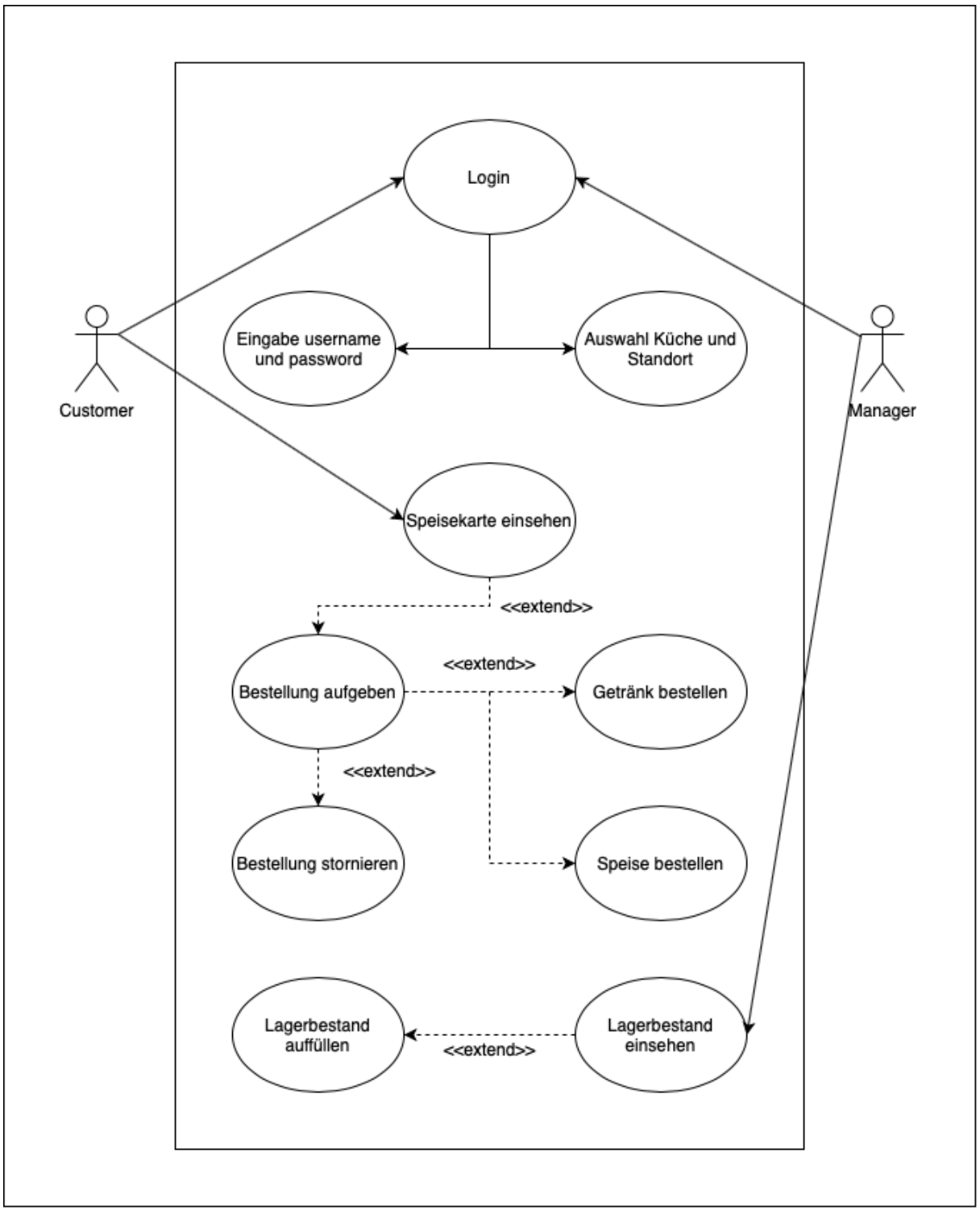
Unsere main Methode befindet sich im package GUI in der Klasse Main.

## 3. Besonderheiten

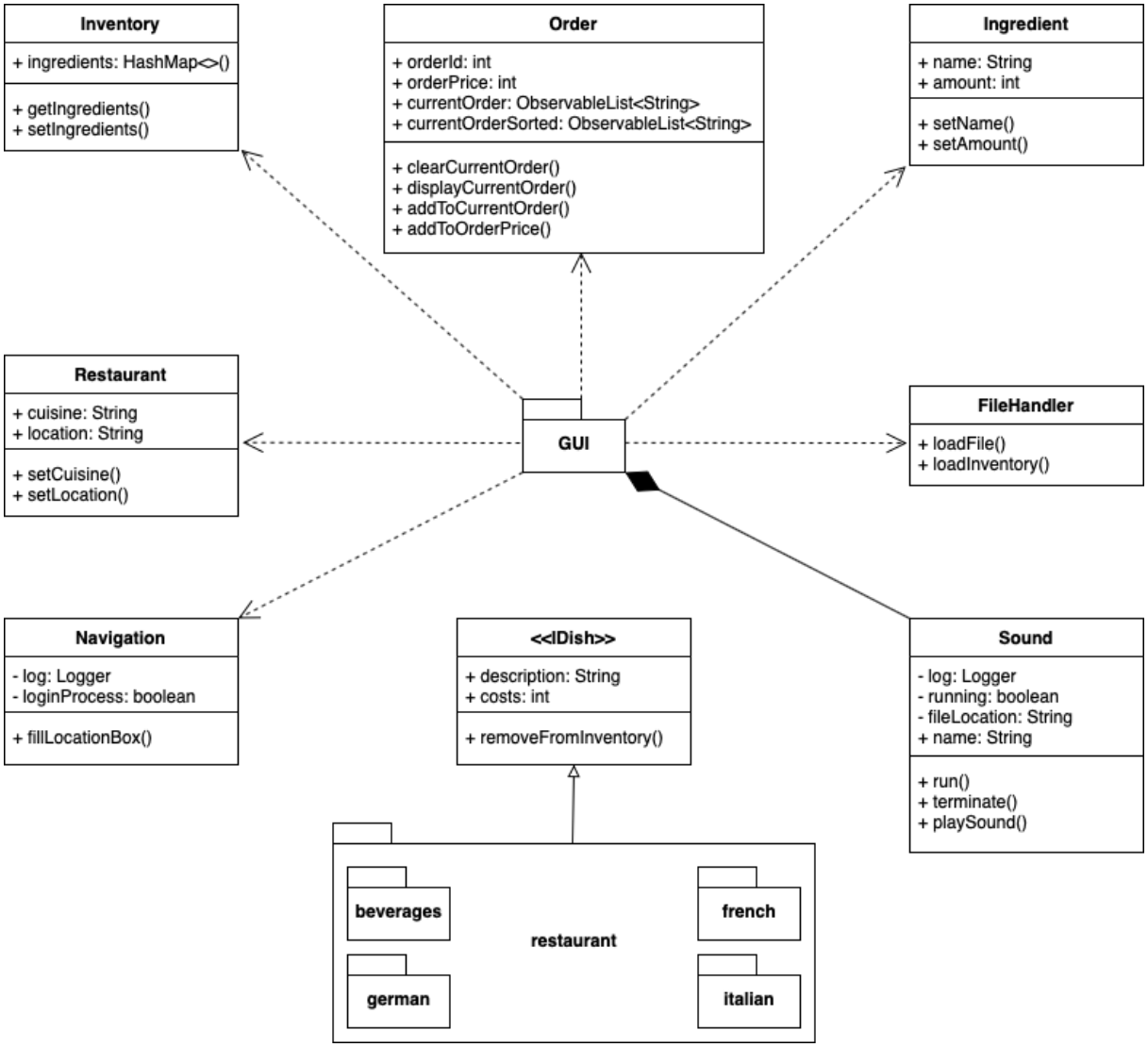
Das Anmelden in der App und das Einsehen und Auffüllen von Lagerbeständen haben wir ohne Datenbank realisiert. Beim Anmelden findet ein if else Abgleich zwischen den zwei verfügbaren Benutzern customer und manager ab, Lagerbestände der jeweiligen Küchen und Standorte sind in einer .txt Datei unter dem Pfad Inventory/ zu finden. Logininformationen:

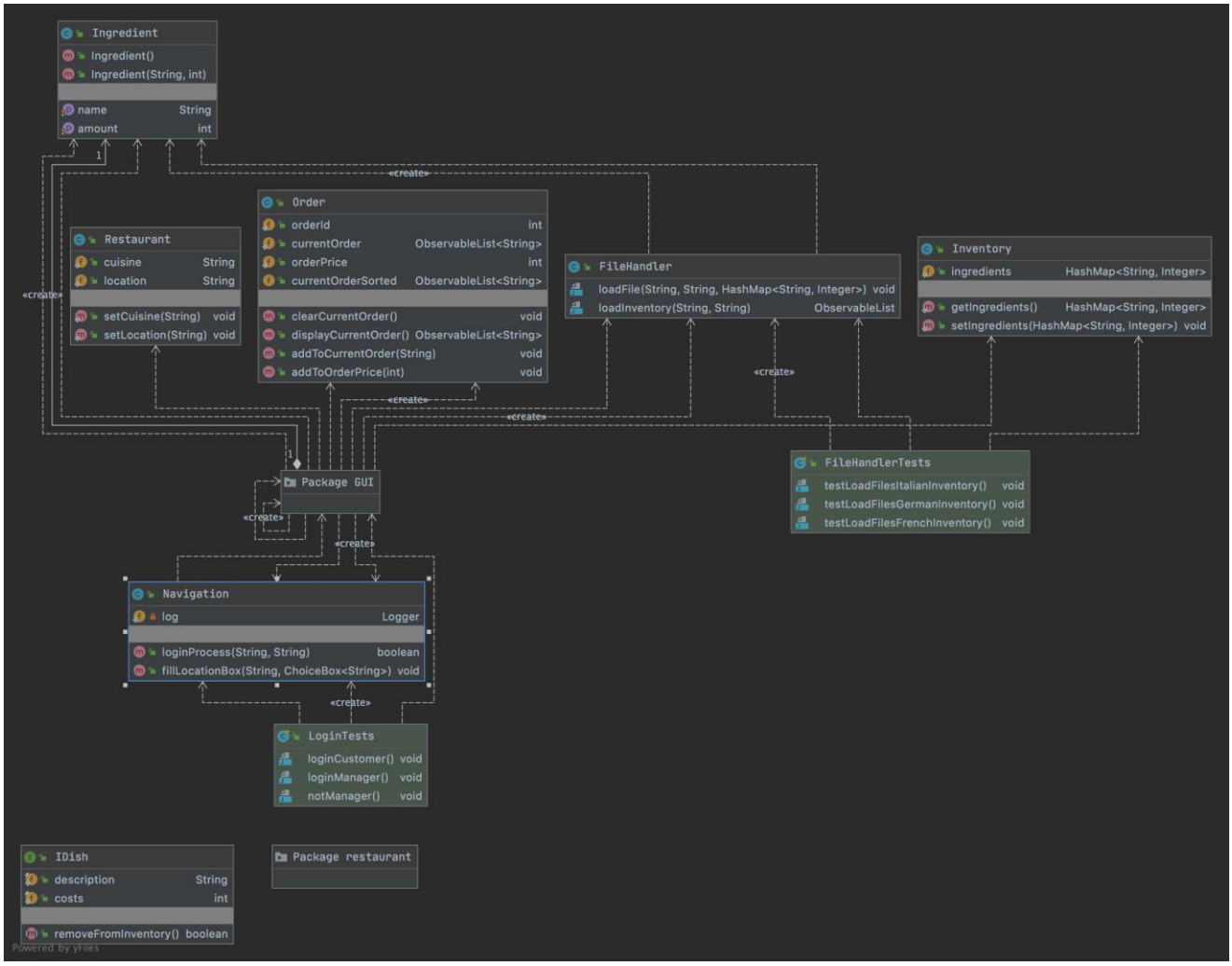
| **User** | **Username** | **Passwort** |
|---|---|---|
| **Customer** | customer | password |
| **Manager** | manager | password |

## 4. Use-Case Diagramm

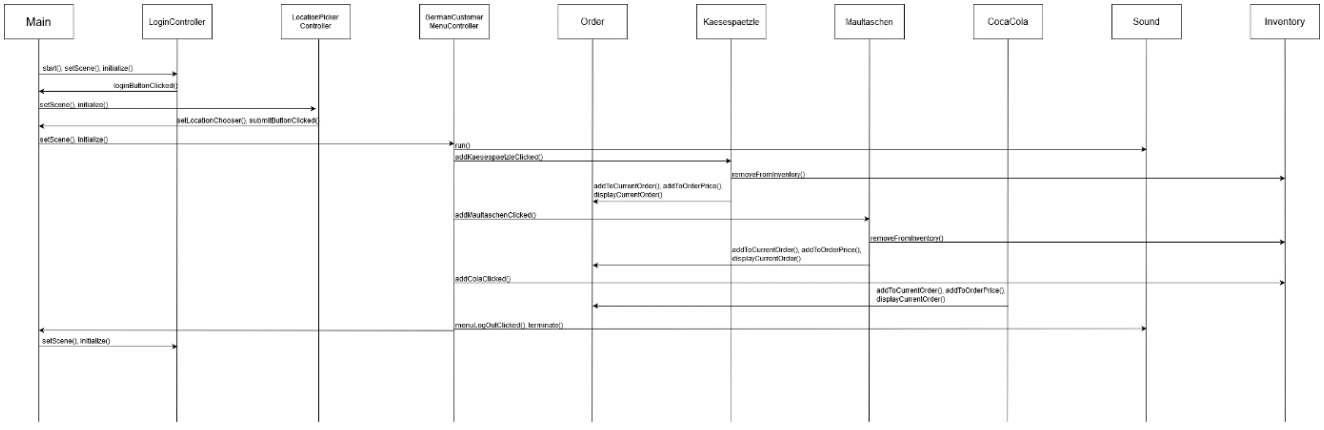


## 5. Klassendiagramm






## 6. Sequenzdiagramm




Die Diagramme findet man zusätzlich im Repository im Ordner „UML-Diagramme“.

## 7. Stellungnahme

Architektur

Das Interface IDish (Im Package de.hdm_stuttgart.mi.se2) wird für die Erstellung jeglicher Gerichte verwendet. Dementsprechend kann man beliebig viele Gerichte hinzufügen und das Programm um Restaurants und Gerichte erweitern. Das Interface speichert Informationen über den Namen des Gerichts und Preis. Es ist in der Lage mithilfe der Zutatenliste des Gerichts den Abgleich mit den Lagerbeständen zu erledigen. Beispiele für das implementierte Interface findet man im package „restaurant“.

Clean Code

An allen möglichen Stellen haben wir private oder protected access modifier verwendet. Einige Felder tragen jedoch auch public access modifier, da es für die Schnittschnelle zwischen Core Klassen und GUI notwendig war. Die Felder und Methoden in den Controller Klassen tragen überwiegend public access modifier, da sie Zugang zu den fxml-Dateien im package resources brauchen.

Static Felder werden in der Klasse Order benötigt, da auf diese von verschiedenen Klassen während des Bestellvorganges zugegriffen wird, um die Bestellung zu bilden.

Die Klasse AlertBox im GUI package verwendet eine statische Methode, eine Hilfsmethode, die es uns ermöglicht flexibel und schnell im GUI Meldungen an den User mittels einer AlertBox zurückzugeben.

Tests

Wir haben mithilfe von Junit Tests im Ordner „test“ vier verschiedene Testklassen implementiert, welche die Klassen FileHandler, Navigation, Order und Restaurant im package de.hdm_stuttgart.mi.se2 testen. Jede Klasse umfasst umfangreiche Tests, inklusive Negativ-Tests.

- FileHandlerTests: Das Laden der Informationen aus den Textdateien („Datenbank“) wird getestet

- NavigationTests: Testet ob Einloggen des Users oder Managers möglich ist

- OrderTests: Order-Prozess wird getestet (Sortierung, Hinzufügen von Gerichten, Preisbildung)

- RestaurantTests: Testet ob Küche und Location korrekt gewählt werden können

GUI (JavaFX)

Die verschiedenen Stages wurden mit JavaFX, SceneBuilder und fxml-Dateien gebaut. Außerdem verwenden wir CSS für das moderne Design. Die fxml und CSS-Dateien findet man im Ordner resources. Im Package GUI findet man die Controller für die Scenes. Wir haben ein verschachteltes Layout mit mehreren Screens. Auch kann man sich frei zwischen den Stages hin und herbewegen, ohne dass ein Neustart des Programms notwendig ist.

Logging/Exceptions

Wir haben die verschiedenen Logging-Stufen berücksichtigt und demnach Logging implementiert. Das Logfile findet man im Root Verzeichnis des Projekts als Datei A1.log. Darin werden Fehlermeldungen, Programmabläufe, Threads, Bestellungen, etc. geloggt und dokumentiert.

Exceptions werden mithilfe von try and catch Blöcken abgefangen und mit der Fehlermeldung geloggt. Beispiele findet man hierfür in der Klasse FrechCustomerMenuController im GUI package.

UML

In der Dokumentation findet man die Klassendiagramme, Use-Case-Diagramme und Sequenzdiagramme. Die Diagramme wurden gemeinsam auf Grundlage der Projektstruktur entwickelt.

Threads

Unsere Threads spielen Hintergrundmusik in den Kundenmenüs. Die Threads werden in der Klasse Sound gebaut und in den Controllern der Customer Menus implementiert.

Streams und Lambda-Funktionen

In der Klasse Order findet man in der Methode displayCurrentOrder() einen umfangreichen Stream, der die Einträge in einer Map zählt, nach Menge sortiert und anschließend in eine observableArrayList konvertiert um eine Übersicht der Bestellung zurückzugeben.

Factories

Die Klasse Ingredient (Im Package de.hdm_stuttgart.mi.se2) dient als unsere Factory um die TableView des Managers in der entsprechenden GUI Stage zu füllen. Dabei wird jeder Zutat ein Name und eine Menge zugeordnet. Die Zutaten werden mithilfe der Inventarlisten (.txt Dateien) in der Klasse FileHandler importiert und als Objekte generiert.