BerufsAkademie Rhein-Main
University of Cooperative Education - staatlich anerkannt -

Entwickelung eines objektorientierten Programms in Java, das die Artikel- und Bestellungs-Verwaltung eines Online-Shops modelliert

Seminararbeit
erstellt im Rahmen der Veranstaltung:
Programmieren, Design und Implementierung von Algorithmen

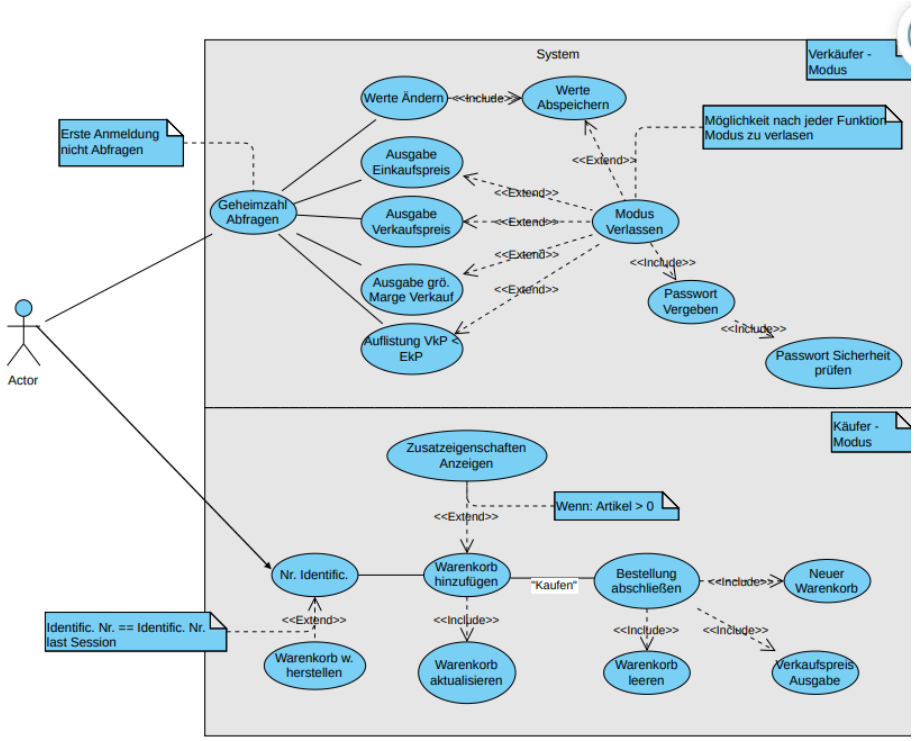
Studiengruppe:	WS-24-III
Name Studierender:	Pawel Tadeusz Sas, Simon Ben-Ralf Braun, Felix Pascal Hempel, Maximilian Mohr
Anzahl der Wörter: (inkl. wörtliche Zitate / Fußnoten)	2891 Wörter
Anzahl der Wörter: (exkl. wörtliche Zitate / Fußnoten)	2568 Wörter
Akademischer Gutachter:	Dipl.-Ing. Jürgen Rolf
Abgabedatum:	17.12.2024

Inhaltsverzeichnis

1 Einführung in das Konzept von dem Online-Shop

Die Anforderung an das Projekt aus dem Programmieren-Modul ist, einen Online-Shop zu entwickeln, in dem man zwischen einem Käufer und Verkäufer Ansicht wechseln kann. Die einzelnen Abschnitte der Software sind mit ihren eigenartigen Funktionen und Anforderungen voneinander verschieden und erfordern ein breites Spektrum an Werkzeugen, welche dem Team in der Programmiersprache „Java“ zur Verfügung gestellt wurden. Um die Anforderungen aus dem Lastenheft zu dem Projekt vollständig und korrekt umsetzen zu können, benötigt es ein konkretes Softwarekonzept zu entwickeln.

Im ersten Schritt der Entwicklung des Konzeptes war es, aus dem gegebenen Lastenheft, ein sogenanntes UC-Diagramm, abgekürzt aus dem englischen „Use-Case-Diagramm“ zu entwickeln, um die Anforderungen und Funktionen, sowie die Bereiche und die Zugriffsrechte von Benutzern in der Software für alle Teammitglieder verständlich und einheitlich zu erklären.

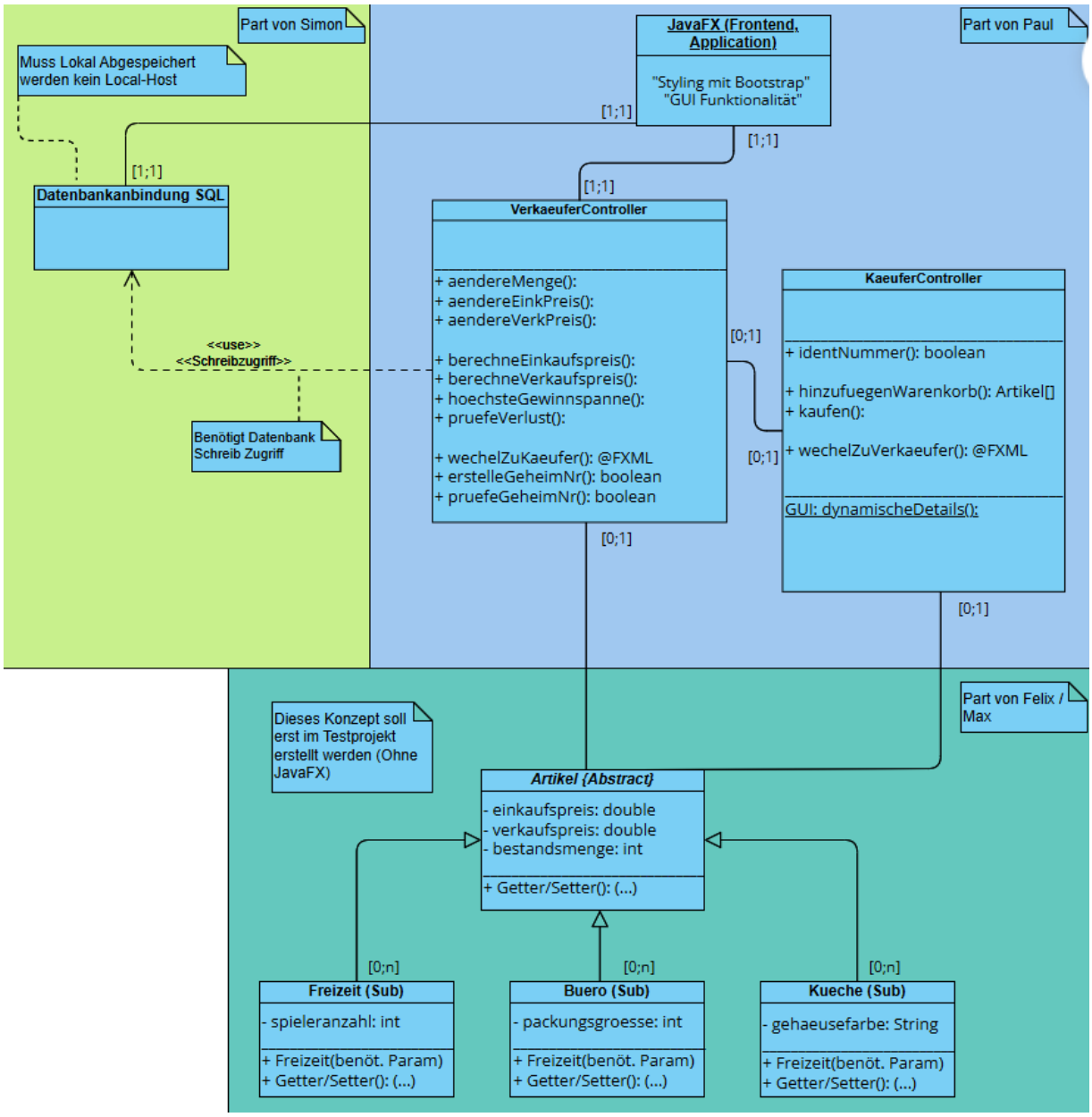


(Anforderungsfalldiagramm, auch Use-Case-Diagramm genannt)

Im Nachfolgenden galt es, die Projektstruktur in einem Meeting zu entwickeln. Dabei haben alle aus dem Team für eine Erweiterung der Java-Software um eine GUI, also eine graphische Benutzeroberfläche gestimmt.

Eine Möglichkeit, für die Java-Software eine graphische Oberfläche zu entwickeln, ist es, das sogenannte „JavaFX“ Framework zu benutzen, welches bestimmte Strukturen implementieren muss. Hier wird es bei der kurzen Erwähnung bleiben, die weitere Vorgehensweise sowie die Funktionen und Aufbau der Szenen in unserem Projekt haben in dieser Dokumentation ein eigenes Kapitel.

Nachdem bekannt wurde, welche Gebiete unser Projekt umfasst, diese in einem vorläufigen Klassendiagramm dargestellt wurden, konnten die Aufgaben innerhalb des Projektes nach Möglichkeiten und Präferenzen aufgeteilt werden, sodass jedes Teammitglied ein Teilbereich der Entwicklung übernehmen konnte. Die Schwierigkeit dabei besteht. Im konkreten entstand eine Aufteilung auf die Bereiche: Datenbank, Backend, Frontend. Im folgenden Klassendiagramm, welches in der Anfangsphase der Entwicklung verwendet wurde, sieht man anhand der farblich abgegrenzten Gebiete der einzelnen Bereiche, sowie die Verbindungen zwischen den einzelnen Klassen und unter Umständen auch dessen Vererbungsstrukturen.



(Klassendiagramm des Online-Shops)

Jeder Bereich hat seine eigenen Herausforderungen, wie eine Anbindung und Verwaltung von Produktobjekten in der Datenbank, oder der Aufbau von den Produktklassen selbst, in denen sich die Konzepte der Generalisierung und Spezialisierung anbieten. Es musste sichergestellt werden, dass die Attribute von Produkten auch passend in die Datenbank hineingefügt werden können, weshalb auch in so einer Aufgabe Zusammenarbeit zwischen dem Backend-Team und dem Datenbanken-Team gefragt wurde.

Es stellt sich jedoch die Frage, wie wurden die Projektdateien verwaltet?

Die Entscheidung im Team fiel auf die Lösung über eine Repository von GitHub entschieden, was uns als Möglichkeit eröffnet hat, die Entwicklung der Datenbank und dem Backend in ein einzelnes Projekt in einer sogenannten „Branch“ aufzuteilen, während das Frontend in einer eigenen Version entwickelt wurde. Die Vorteile dieses Konzeptes sind, dass die einfacheren Aufgaben, wie Implementierungen von Produktklassen, in einem übersichtlicheren Rahmen gestaltet werden konnte. Die Übersicht schaffte eine schnellere Fertigung des Backends, sowie eine allgemeine Übersicht über die Aufteilung, da erst am Ende der Teil „Datenbank und Backend“ zu der graphischen Oberfläche hinzugefügt wurde.

Es entstanden jedoch Herausforderungen, zu einem mussten die Methoden zu Weitergabe von Variablen an und Methodenaufrufe von der graphischen Oberfläche recht spät in der Entwicklung in einem kurzen Zeitraum entwickelt werden. Es musste eine Schnittstelle zwischen den beiden Subprojekten geschaffen werden, was möglicherweise nicht eine optimale Vorgehensweise ist, beziehungsweise zu größeren Verzögerungen in der Implementierung führt, welche möglicherweise in dem Zeitplan nicht bedacht werden.

1.1 Was ist Objektorientierung und wie kommt es im Projekt zum Einsatz?

Objektorientierung ist eine Mittel, um Programme zu schreiben, so dass sie eine gute **Übersichtlichkeit** und **strukturiert** bekommen. Man stellt sich Dinge als **Objekte** vor so wie im echten Leben. Jedes Objekt hat **Eigenschaften** und kann **etwas tun**.

Was sind Klassen und Objekte?

- Eine **Klasse** ist wie ein **Bauplan** oder eine Vorlage.
- Ein **Objekt** ist ein Produkt, das nach diesem Bauplan gebaut wird.

Beispiel:
Stell dir eine Klasse **Auto** vor. Sie ist der Bauplan für Autos.
Jedes **Objekt** ist ein einzelnes Auto, das aus diesem Bauplan gebaut wurde.

- **Eigenschaften**: Farbe, Marke, Preis (z. B. “Grün”, “Audi”, “35.000€”)
- **Aktionen**: Fahren, Hupen

Wie sieht das in unserem Programm aus?
Es gibt eine **Superklasse** (Produkt). Diese Klasse ist wie ein allgemeiner Bauplan für alle möglichen Produkte. Alle Produkte haben gemeinsame Eigenschaften:

- **Einkaufspreis**: Was das Produkt kostet, um es zu kaufen.
- **Verkaufspreis**: Für wie viel das Produkt angeboten wird.
- **Bestandsmenge**: Wie viele Produkte noch da sind.

Beispiel:
Stell dir ein Online-Shop vor, das verschiedene Produkte verkauft:

- Spielzeug (Freizeit)
- Büroartikel (Büro)
- Küchenartikel (Küche)

Wie funktioniert Vererbung?
Anstatt für jedes Produkt alles neu zu schreiben, findet das Konzept der **Vererbung** ihren Einsatz . Die Superklasse (**Produkt)** gibt ihre Eigenschaften an andere Klassen weiter.

- **Freizeit** erbt von **Produkt** und fügt etwas Neues (den **Spieler**) hinzu.
- **Büro** erbt von **Produkt** und fügt die **Packungsgröße** hinzu.
- **Küche** erbt von **Produkt** und fügt die **Gehäusefarbe** hinzu.

Erklärung:

- Die Superklasse **Produkt** ist wie eine Grundausstattung (z. B. alle Autos haben Reifen und Motoren).
- Die anderen Klassen erweitern das Ganze (z. B. Sportwagen haben zusätzlich einen Spoiler, Lieferwagen haben mehr Stauraum).

Beispiel aus dem Alltag

Stell dir vor, du hast einen Bauplan für Möbel:

1. **Superklasse** Möbel:

- Eigenschaften: Material, Preis, Gewicht

2. **Unterklassen (Erweiterungen)**:

- **Tisch**: Fügt “Anzahl der Beine” hinzu.
- **Stuhl**: Fügt “Sitzpolster” hinzu.
- **Schrank**: Fügt “Anzahl der Türen” hinzu.

Warum ist das praktisch?

1. **Ordnung und Übersichtlichkeit des Codes**:

- Man schreibt nicht alles doppelt, sondern benutzt die Grundausstattung (Superklasse).
- Änderungen an der Grundausstattung wirken überall.

2. **Erweiterbarkeit**:

- Wenn ihr neue Produktarten braucht, fügt ihr einfach neue Klassen hinzu, die von Produkt erben. Der alte Code bleibt so, wie er ist.

3. **Verständlichkeit**:

- Auch wenn man nicht alles versteht, kann man leicht erkennen:

Freizeit, **Büro** und **Küche** sind **Produkte**, aber mit besonderen Eigenschaften.

Codeauszug aus unserem Projekt:

```
// Superklasse: Allgemeiner Bauplan für alle Produkte
public class Produkt {
private double einkaufspreis;  //Was das Produkt kostet
private double verkaufspreis;  //Für wie viel man es verkauft
private int bestandsmenge;   //Wie viele Produkte es noch gibt
}

// Subklasse Freizeit: Erweiterung für Freizeit-Produkte
public class Freizeitartikel extends Produkt {
//Besonderheit: Welcher Spieler oder Spielzeug-Typ
private String spieler;
}

// Subklasse Büro: Erweiterung für Büro-Produkte
public class Bueroartikel extends Produkt {
//Besonderheit: Größe der Packung
private String packungsgroesse;
}

// Subklasse Küche: Erweiterung für Küchen-Produkte
public class Kuechenartikel extends Produkt {
//Besonderheit: Farbe des Gehäuses
private String gehaeusefarbe;
}
```

Zusammenfassung:

- **Klasse**: Ein Bauplan (z. B. “Produkt”).
- **Objekt**: Ein echtes Ding, das nach diesem Plan erstellt wird (z. B. “Bleistift”, “Pfanne”).
- **Vererbung**: Ein Bauplan (Superklasse) gibt seine Eigenschaften an andere Baupläne weiter, die noch eigene Besonderheiten hinzufügen.
- **Vorteil**: Ihr spart Zeit, vermeidet doppelten Code und euer Programm bleibt übersichtlich.

So könnten die Produktarten ganz einfach erweitert werden, ohne alles neu schreiben zu müssen!

1.2 Konzept der Datenkapselung und Generalisierung / Spezialisierung

Vererbungsstrukturen kennt man ursprünglich aus der Biologie. Eltern bekommen Kinder und diese haben einen Genpool mit dem Gen-Satz beider Elternpaare. Folgendes Prinzip kann auch auf das Programmieren bezogen werden. Dabei handelt es sich nicht um Gen-Satz Vererbung, sondern um die Klassen und dessen Aufbau. Elternklassen haben Grundinformationen und Grundregeln, die an die Kinderklassen weitergegeben werden. Jedoch wird dies beim Programmieren „Superklasse“ und „Subklassen“ genannt. Eine Klasse besteht aus Methoden und Eigenschaften, auch Attributen genannt. Diese können innerhalb der Klasse verwendet werden, möchte man jedoch diese Eigenschaften in einer anderen Klasse verwenden müsste man den kompletten dafür benötigten Code kopieren und in die andere Klasse implementieren. Dies zu machen wäre redundant und um redundanten Code zu verhindern, gibt es die Vererbung.

Superklassen geben Methoden und Eigenschaften vor, als Beispiel dafür kann man ein Tier nehmen.

Superklasse „Tier“ dieser Klasse werden allgemeine Eigenschaften hinzugefügt, die jedes Tier hat.

Eine dieser Eigenschaften wäre zum Beispiel der Integer Gewicht.

Eine Subklasse würde jetzt die Klasse Tier spezialisieren. Als Beispiel für eine spezifizierte Klasse gilt die Klasse „Haustier“ oder eine gewisse Tier Gruppe wie „Vogel“. Diese Subklassen haben auch die Eigenschaft „Gewicht“ durch die Vererbung als Attribut bekommen. Subklassen können Superklassen erweitern jedoch die Grundstruktur nicht verändern dies liegt am Open-Closed-Prinzip von Bertrand Mayer beschrieben in seinem Buch „Object-Oriented Software Construct“ von 1988.

Bei der Vererbung bekommen die Subklassen ihr Konstrukt durch die Superklasse vorgegeben. Hierfür gibt es die super() Methode, welche in den Konstruktor der Subklasse hineingeschrieben wird und es erlaubt direkt Methoden oder Eigenschaften aus Superklassen zu verwenden, indem man mit „super.[Methode/Variable]“ auf die darüberliegende Superklasse verweist.

Wenn man von der Vererbung hört, denkt man erstmal an die Vorteile und dass man dadurch eigentlich Vererbung nach Vererbung machen kann.

Jedoch hat diese sogenannte Mehrfachvererbung Probleme. Ein Beispiel solcher Probleme wäre das Diamond-Problem, hierbei erbt eine Klasse von mind. 2 Klassen, die beide von einer gleichen Superklasse kommen. Denn bei diesem Fall kommt es dazu das Methoden oder Eigenschaften mehrfach durchgeführt werden.

Das Problem der Mehrfachvererbung von Klassen tritt nicht in jeder Programmiersprache auf, da Sprachen wie Java oder C# dies nicht zulassen.

1.3 Einbau und Architektur von SQL-Datenspeicherung in dem Online-Shop

Um die Daten speichern zu können, wurde sich früh auf eine relationale Datenbank festgelegt, weil diese am besten mit der logisch notwendigen Datenstruktur vereinbar ist. Für die relationale Datenbank wird SQL, die “Structured Query Language”, verwendet. SQL ist eine sehr weit verbreitete Datenbanksprache und dank der hohen Nutzerzahl liegen viele Lehrmittel und Hilfestellungen vor.

Für die Nutzung einer relationalen Datenbank mit SQL ist die Wahl auf SQLite gefallen, eine Programmbibliothek, die es erlaubt, auf eine Datenbank zuzugreifen und SQL-Statements auf ihr auszuführen, ohne diese separat hosten zu müssen, was eine wünschenswerte praktische Simplizität bietet.

Um SQLite nutzen zu können, wird ein sogenannter JDBC-Treiber benötigt, welcher die Datenbankverbindungsfähigkeit für Java bereitstellt.

Im Programm selbst wird mit dem Starten des Programms über einen statischen Codeblock automatisch eine Verbindung mit der Datenbank aufgebaut, um weitere Interaktionen zu ermöglichen. Die Interaktionen erfolgen dann über größtenteils vorher verfasste SQL-Statements, die lediglich um die notwendigen Parameter ergänzt werden.

Die Datenstruktur ist im Code und in der Datenbank möglichst gleich, so werden die Produkte in die drei Bereiche “Freizeitartikel”, “Büroartikel” und “Küchenartikel” unterteilt. Die Datenstruktur im Code stellt dies mit einer Vererbungsstruktur dar, in der z.B. ein Küchenartikel neben dem eigenen Attribut “Gehäusefarbe” auch noch die geerbten Attribute eines Produkts wie z.B. “ID” oder “Name” hat.
In der Datenbank funktioniert die Vererbung über einen geteilten Primärschlüssel in den Tabellen, so würde der Küchenartikel aus dem vorherigen Beispiel in der Tabelle für Produkte mit den Generischen Attributen stehen, wobei die ID der Primärschlüssel ist, während in einer Separaten Tabelle unter dem gleichen Primärschlüssel noch das Attribut “Gehäusefarbe” hinterlegt ist.
Bei der Erstellung der passenden Objekte im Code wird dann abgefragt, in welchen Tabellen der Primärschlüssel aus der Produkttabelle noch vorkommt und entsprechend den Ergebnissen wird ein passendes typisiertes Produkt in Form von z.B. eines Küchenartikels erstellt.

1.4 Graphische Darstellung der Software mit Hilfe des JavaFX Frameworks

Während der Planungsphase der Software fiel die Entscheidung, wie bereits erwähnt, für die graphische Darstellung des „Online-Shops“ mithilfe des JavaFX Frameworks. Was genau ist jedoch JavaFX, was bedeutet der Begriff Rich-Client in dem Projektzusammenhang und wieso ist die Verwendung von JavaFX in einer Realen Anwendung für ein Online-Shop ein nicht vielversprechender Ansazt?

JavaFX wurde ursprünglich entwickelt, um ein Konkurrenzprodukt zu den Damaligen RIA (aus dem englischen: Rich Internet Applications) darzustellen, dabei sollte diese eine Alternative zu Adobe Flash und Microsoft Silverlight bieten.[1]

JavaFX als RIA-Lösung im Browser ist sowas wie ein Vorläufer der heutzutage verbreiteten RIA-Lösungen wie dem NextJS Framework. Heutzutage werden genau solche Frameworks wie NextJS für WebApps verwendet, was zu Folge hat, dass Lösungen wie JavaFX immer weniger verbreitet sind. Moderne Lösungsansätze wie das erwähnte NextJS werden jedoch nicht in der Programmiersprache Java entwickelt, sondern basieren auf JavaScript und HTML5.

Für die geforderte Aufgabe in dem Programmieren Projekt ist es jedoch wichtig, die grundlegenden Konzepte von Java in dem Backend des Online-Shops abzubilden, daher ist der Ansazt, eine Software, die Clientseitig läuft zwar eine keine gute Idee für ein Produkt, welches so im Markt eingesetzt werden sollte, da ein Nutzer für einen Online-Shop nicht extra eine Software installieren möchte, jedoch eignet sich gut, um die Abläufe sowie die Struktur der umgesetzten Software graphisch und angenehm für die Zuhörer in der Präsentation abzubilden.

[1] Vgl. Kofler, 2022, S. 359.

In dem Falle wurde ein Online-Shop als ein sogenannter Rich-Client Software entwickelt, dabei impliziert der Name, dass die Software, welche Lokal aus der Benutzerhardware installiert werden muss, über viele dynamische Funktionen verfügt.

Weiter ermöglicht dieser Ansatz die Verwendung von den Vorteilen, welche die Szenenverwaltung von JavaFX mit sich bringt.

So können Funktionen wie ein Szenenwechsel zwischen der Verkäuferansicht und der Startseite für die Benutzer übersichtlich in von uns benannten „Dialogfenstern“ umgesetzt werden, bei denen die Controller aus dem „Hauptfenster“, in dem sich die Szene befindet, an ein neues Fenster übergeben werden, um dann von diesem ferngesteuert zu werden. Somit wird eine Benutzerfreundlichkeit der Software gewährleistet.

Eine Erleichterung der Arbeit war jedoch das Tool „Scenebuilder“, in dem die auf XML basierten FXML-Dateien, welche als die genannten Szenen aufgerufen werden, designt werden können. Die Software „Scenebuilder“ bietet eine Lösung, direkt graphisch per Drag-and-drop Interfaceobjekte von JavaFX in die Szene zu platzieren. Eine weitere Erleichterung ist, die Möglichkeit des Stylings über die Einbindung einer Datei für die Styling-Definition. Dafür wird das CSS, also „Cascading-Style-Sheets“ verwendet, in welcher die einzelnen Interfaceobjekte ihr Aussehen nach Gruppierungen zugewiesen bekommen. Um ein Beispiel aus dem Projekt zu nennen, sind alle Interfaceobjekte des Typs „Pane“ in einer einheitlichen Designsprache gehalten. Dies wird eben durch die CSS-Datei ermöglicht.

Es muss jedoch erwähnt werden, dass mit einer Implementierung des JavaFX Frameworks neue Herausforderungen an das Projekt gesetzt werden. Da Seit der Java Version 11 die Bibliothek für JavaFX nicht mehr, in der Standard JDK vorhanden ist, müssen die dazugehörigen Bibliotheken erst heruntergeladen werden und mit einem Projekt „Buildsystem“, im unserem Falle Maven eingestellt werden. Nur so kann gewährleistet werden, dass die kompilierte. Jar Datei des Projektes auch auf jeder Hardware ausgeführt werden kann.

2 Fazit und weitere Möglichkeiten für die Weiterentwicklung

Zusammenfassend wurde das anfangs angestrebte Konzept des Online-Shops während der Entwicklung erfolgreich umgesetzt. Herausforderungen wie die Anbindung einer Datenbank an die Software wurden durch sorgfältige Recherche seitens des Datenbank-Teams gelöst, in dem passende JDBC Treiber gefunden wurden, welche den benötigten Anforderungen, einer lokalen Speicherung der Daten mithilfe von SQLite entsprach, ebenso wie der strukturelle Aufbau und Umgang mit den Daten der Produktobjekte im Hintergrund, welche von dem Backend-Team und dessen aufwendiger Recherche aus dem Java Buch „Grundkurs“, verfasst von Michael Kofler, einen großen Beitrag geleistet hat. Dabei muss auch die Umsetzung und Implementierung der graphischen Oberfläche erwähnt werden, welche mit dem Gedanken entwickelt wurde, die dahinterstehenden Konzepte in einer einfachen Darstellung den Zuschauern herüberzubringen.

Darüber hinaus wurden auch einige modulübergreifende Erfahrungen gemacht, wie zum Beispiel der Umgang mit dem Dual-Problem der effizienten Arbeitsaufteilung und dem damit verbundenem Mehraufwand bei der Zusammenbringung von den einzelnen Elementen der Software. So verlief das Projekt wie in der Einführung in das Konzept geplant war, wobei jeder der Teammitglieder Erfahrungen gemacht hat, was genau das Dual-Problem ausmacht und wie man damit umgeht.

In den finalen Projekttreffen konnte am Ende jeder Beteiligte mit einer Überzeugung sagen, dass viele neue Einblicke in die Softwareentwicklung und Projektmanagement gewonnen worden sind, was das primäre Ziel dieser Projektarbeit wurde. Zwar ist das Konzept des Online-Shops keines, dass wirklich so einen Erfolg auf dem Markt haben würde, aber das war auch nicht die Absicht, somit gibt es auch viele Gebiete an denen Entscheidungen getroffen worden sind, die so bei einem echten Online-Shop anders getroffen werden müssten.

Da wäre zum Beispiel die Wahl des Frontends, welches in der Praxis in unserem Team das auf der JavaScript-Programmiersprache basierte ReactJS, oder NextJS wäre, mit einer Lösung, die nicht lokal auf der Endnutzer Hardware liegt, sondern eine SAAS, also Software as a Service Web-Applikation wäre, auf die man über den Browser zugreift und sowohl von dem Styling auf mobile Geräte wie auf Desktops angepasst wäre. Doch auch in diesem Projekt gab es mögliche Entscheidungen, wie die Wahl der Datenbankstruktur, da Teammitglieder über Erfahrung mit dem Datenbankmanagementsystem „MySQL“ verfügten, dieses jedoch zumindest lokal gehostet werden muss, was nicht dem Konzept der Software entsprach.

Die Aufgabenteilung ließ niemanden Zurück und sowohl die Erfahrenen wie Einsteiger haben einen Mehrwert und vor allem Freude und Verwirklichung an dieser Projektarbeit gefunden.

3 Plagiatserklärung

Wir versichern, dass die Dokumentation von uns selbstständig erarbeitet wurde und wir keine anderen als die angegebenen Hilfsmittel benutzt haben. Diejenigen Teile der Dokumentation, die anderen Werken im Wortlaut oder dem Sinn nach entnommen wurden, sind als solche kenntlich gemacht worden.

4 Quellenverzeichnis

https://lerneprogrammieren.com/was-bedeutet-vererbung-in-der-programmierung/
(Robert Mertens, 24.12.2023)
https://simpleclub.com/lessons/fachinformatikerin-vererbung-in-java
(Simpleclub, unbekannt)
https://www.w3schools.com/java/ref_keyword_super.asp
(W3Schools, unbekannt)
https://www.biteno.com/was-ist-sqlite/
(Daniel Faust, 20.01.2020)

Alle Quellenangaben wurden zuletzt am 17. Dezember 2024 aufgerufen und auf deren Inhalt geprüft.

5 Literaturverzeichnis

1. Kofler, Michael – „Java: Der Grundkurs im Taschenbuchformat mit Aufgaben und Lösungen“, 2022, Rheinwerk-Computing