Verkaufsprojekt

Software (Engineering)

Oliver Schlüter

10. Januar 2022

Inhaltsverzeichnis

Einleitung

Aufgabe

Die Aufgabe war es ein Anwendung zu realisieren, welche eine Simulation von einen Ver- oder Kaufunternehmen darstellt. Es sollte dabei ein besonderer Schwerpunkt auf die Planung und Dokumentation gelegt werden. Es ist gefordert folgende Diagramme zu erstellen:

- Programmablaufplan
- Entity-Relationship-Modell
- Klassendiagramm (UML)

Auch soll ein Accountmanagement-System vorhanden sein, d.h. Benutzer können sich registieren, anmelden und abmelden. Eine Art von Mitarbeiteransicht soll ebenfalls vorhanden sein, wo man Produkte und Benutzer verwalten kann.
Die Umsetzung soll mit der Programmiersprache C# mit Windows-Forms erfolgen.

Kapitel 1

Konzept

1.1 Idee

Meine Idee war es eine Anwendung zu erschaffen, wo Software-Produkte angeboten werden, welche von Autoren geschrieben und veröffentlicht werden. Diese Software-Produkte kann alles mögliche sein, von Spiel-Quellcode bis zu komplexen Cloud-Systemen. Voraussetzung ist nur, dass man es runter laden kann. Auch sollen Kunden die Möglichkeit bekommen Produkte und Autoren zu bewerten und ein Kommentar zu hinterlegen. Da dabei auch viel Unsinn bei auftreten könnte, brauch man Mitarbeiter, welche anstößige Kommentare blockieren und ggf. den Benutzer sperren können. Wenn es schon Mitarbeiter gibt, kann man auch eine einführen, dass sie Produkte unter die Lupe nehmen können, d.h. dass es sie den Anforderungen entsprechen und keine Schadsoftware enthalten.

1.2 Spezifikation

Um spezieller zu werden habe ich mir folgende Eigenschaften der Bestandteile erarbeitet, welche auch den Grundstein des UML-Diagramms sein soll.

1.2.1 Produkte

Jedes Produkt soll eine eindeutige ID haben, welche fünf Zeichen lang ist, außerdem einen Name und eine ausführliche Beschreibung. Fehlen dürfen nicht die Autoren, welche die Software geschrieben haben. Produkte können umsonst angeboten werden, allerdings ist in der Regel ein Festpreis festgelegt. Kunden, welche ein Produkt gekauft haben, können dieses Bewerten (1-5 Sterne, wobei 5 Sterne besonders gut ist) und kommentieren und ihre Bewertung dadurch begründen.

1.2.2 Benutzer

Es gibt drei Arten von Benutzer: Kunden, Autoren und Mitarbeiter. Sie alle besitzen eine eindeutige ID, welche fünf Zeichen lang ist, einen Vor- und Nachname, eine Email-Adresse, ihr Geburtsdatum, das Erstelldatum des Kontos und ein Passwort.
Zusätzlich werden Autoren mit einen Entwicklerstatus versehen, welches angibt, wie Erfahren und ggf. professionell sie sind. Auch können sie aus angeben, mit welchen Programmier- und Skriptsprachen sie ihre Software erzeugen.
Kunden haben eine Liste von bereits gekauften Produkten und Produkten, welche sie beobachten und sich ggf. in Zukunft kaufen wollen. Die Anwendung basiert auf das Prepaid-Prinzip, sodass jeder Kunde ein Guthaben haben, welches man jederzeit aufladen kann.
Jeder Mitarbeiter werden ein oder mehrere Aufgabenbereiche zugeteilt, wofür sie Zuständig sind und eine Liste von Produkten, welche sie zurzeit Prüfen.

1.2.3 Formulare

Nun noch die Formulare. Beim Start der Anwendung landet man auf der Startseite, wo man die Möglichkeit hat sich anzumelden, sich zu registrieren, sich abzumelden. Falls man angemeldet ist, hat man die Möglichkeit zu einen weiteren Formular zu wechseln, wo man alle Produkte sieht und mit ihnen weiter interagieren kann. Auch kann man sich seine bereits gekauften Produkte in einen weiteren Formular anzeigen lassen. Betrachten wir jedoch diese Formular genauer.

Login Formular

In diesen Formular hat man die Möglichkeit seine Benutzer ID und sein Passwort einzugeben und sich einzuloggen. Das angegebene Passwort muss mit den Passwort der Benutzer ID übereinstimmen.

Registrations Formular

Hier kann man sich einen Kunden-Account anlegen, indem man seinen Vorname, Nachname, Email, Nickname, Geburtsdatum und Passwort angibt. Die Email muss eine existierende Email-Adresse sein und das Passwort muss einigen Vorschriften befolgen (z. B. mindestens 8 Zeichen lang).

Produkte Formular

In diesen Formular erhält man eine Liste von allen Produkten. Wenn man auf ein Produkt kickt, erscheint im unteren Bereiche eine genauere Beschreibung und die Möglichkeit es zu kaufen.

Gekaufte Produkte Formular

Eine Liste von allen bereits gekauften Produkten kann man hier sehen. Man kann die einzelne Produkte wie beim Produkte Formular anklicken um eine genauere Beschreibung zu bekommen. Außerdem befindet sich auf der rechten Seite eine Liste von allen Bewertungen zu den ausgewählten Produkt. Unter diese Liste kann man ebenfalls eine Bewertung abgeben.

Einstellungsformular

In diesen Formular sieht man alle seine Daten (Benutzer ID, Vorname, Nachname, Email, Nickname, Geburtsdatum und Passwort). Man kann einige davon ändern und diese Änderungen dann über einen Knopf übernehmen oder zurücksetzen.

Kapitel 2

Planung

Ein essentieller schritt in der Softwareentwicklung ist die Planung. Auch für dieses Projekt habe ich mich ausführlich mit der Planung auseinander gesetzt. Erste habe ich mir ein generelles Konzept erarbeitet, folgend diese Ideen spezifiziert und anschließend in anschaulichen visualisiert.

2.1 Entity-Relation-Diagramm

Zu Beginn erstellte ich das Entity-Relation-Diagramm, weil dort die Beziehungen zwischen Entitätstypen genauer beschrieben werden.

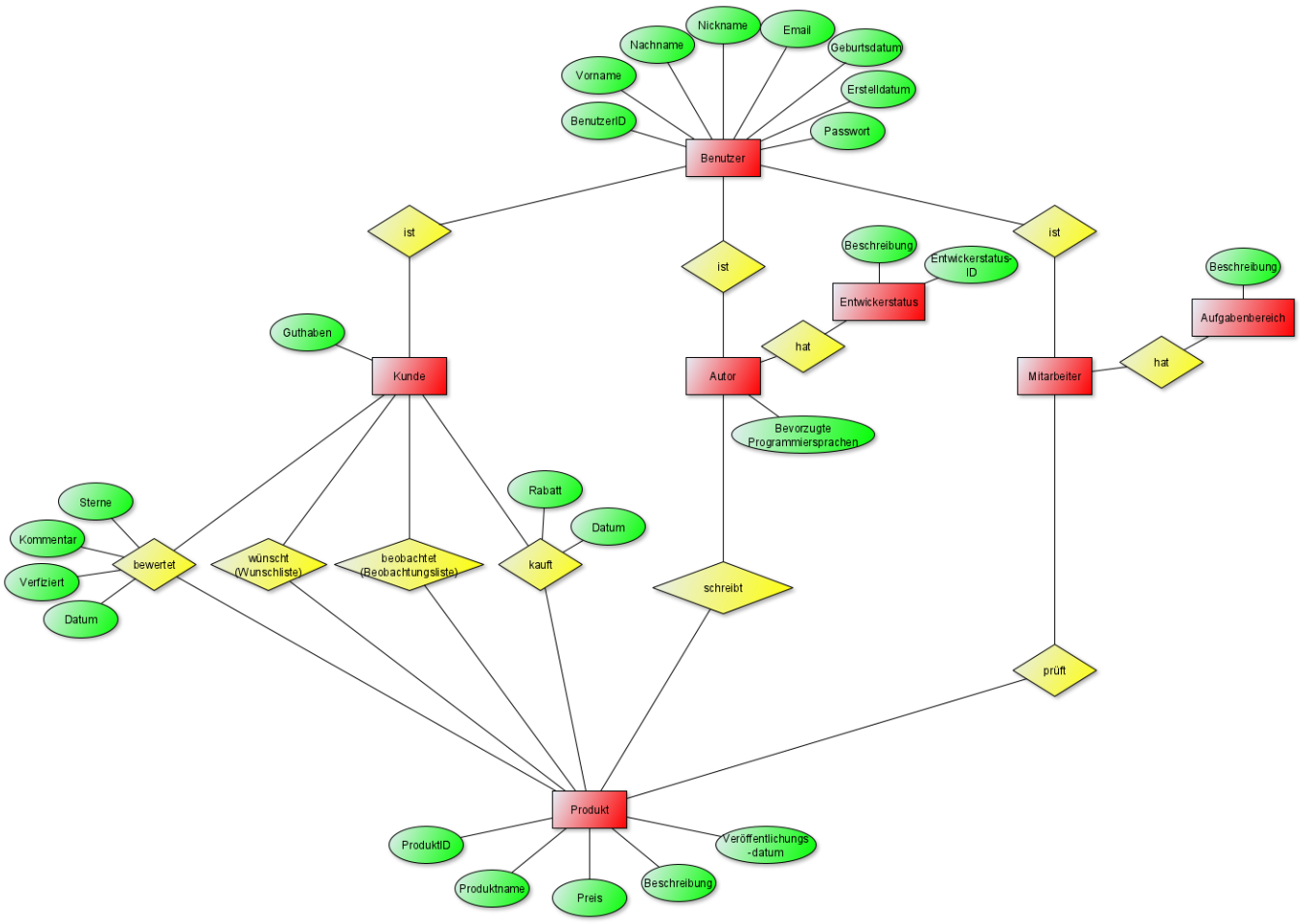

Abbildung 2.1: Entity-Relation-Diagramm

2.2 UML-Klassen-Diagramm

Nachdem ich das erste ER-Diagramm hatte, erstellte ich ein UML-Klassen-Diagramm, wo man eine Übersicht darüber bekommt, welche Klassen und Datentypen notwendig sind. Zu bemerken ist, dass die Inhalte der Enumerationen stetig geändert werden.

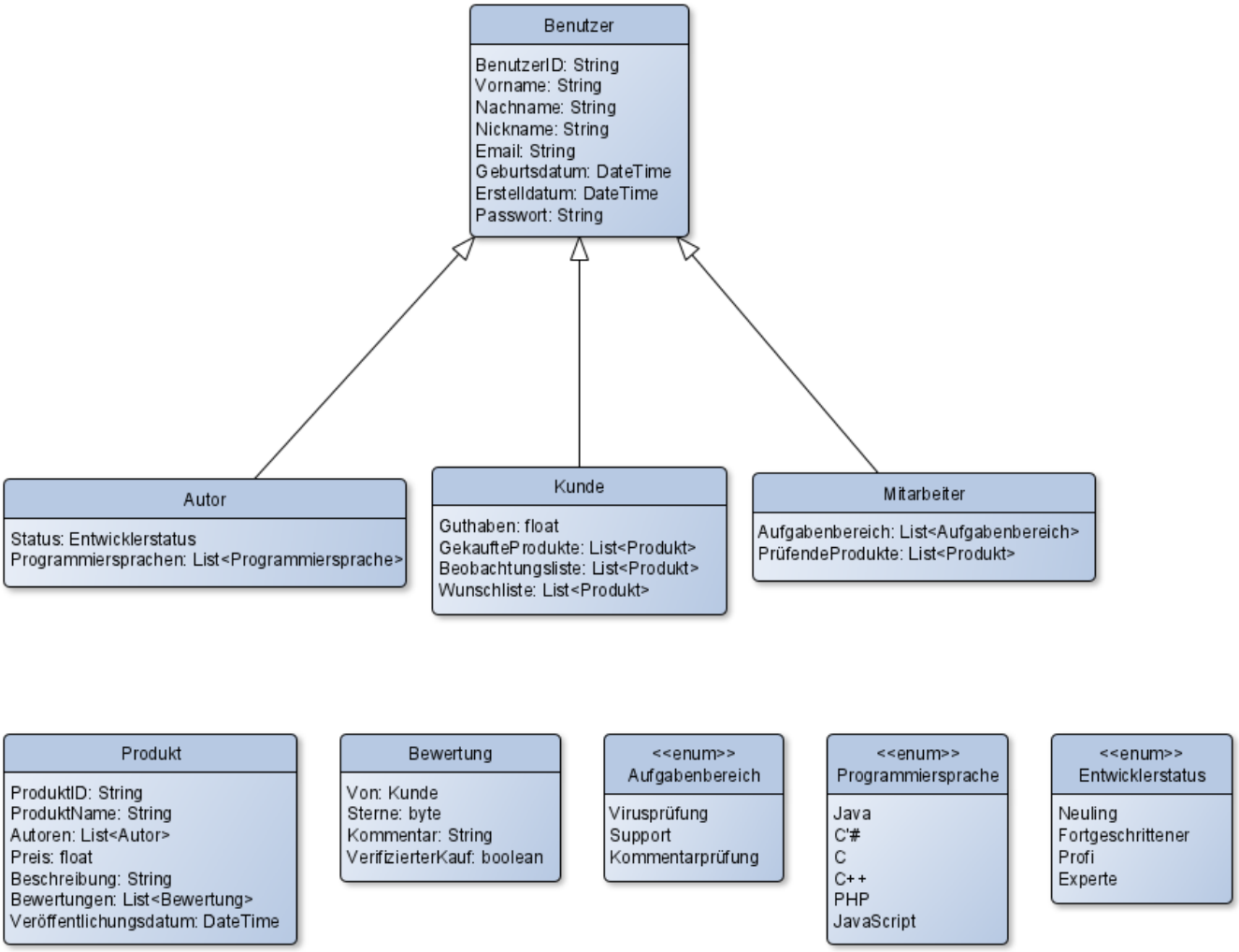

Abbildung 2.2: UML-Klassen-Diagramm

2.3 Programmablaufplan

Für einen übersichtlichen Überblick der Funktionen der Applikation ist ein Programmablaufplan sinnvoll. Vor allem zum Testen der Applikation ist er gut geeignet, da man alle möglichen Zweige einzeln testen kann.

Der Programmablaufplan befindet sich auf GitHub: https://github.com/OliverSchlueter/Verkaufsproje
blob/master/Planning/PAP%20v1.png

2.4 Datenbank Skript

Relativ schnell nach dem Erstellen der Diagramme, schrieb ich das Skript zur Erzeugung der Datenbank, wobei vorher die Überlegung der Wahl des DBMS zu treffen war. In meinen Projekt benutze ich eine MySQL-Datenbank für die Speicherung der Benutzer- sowie Produktdaten, um auf diese zuzugreifen ist diese mit einer lokalen Microsoft Access-Datenbank synchronisiert.
Das Skript finden sie hier: https://github.com/OliverSchlueter/Verkaufsprojekt/blob/master/Planning/DB-Script.sql

Kapitel 3

Umsetzung

In diesen Kapitel werden alle Klassen mit ihren Attributen von Methoden/Funktionen beschrieben.

3.1 Program.cs

In dieser Klasse wird das Programm gestartet. Außerdem enthält sie Informationen über die Version des Programms. Um immer wieder zurück zur Startseite zu gelangen befindet sich in der Program.cs Klasse das statische StartseiteForm Objekt, wobei hier das Singleton Entwurfsmuster angewandt wird, d.h. es soll nur eine Instanz der Klasse StartseiteForm existieren. Ebenfalls enthält sie Informationen über den gerade angemeldeten Benutzer (ebenfalls mit den Singleton Entwurfsmustern angewandt).

3.2 Benutzer.cs

Diese Klasse speichert alle Informationen über einen Benutzer.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public static	List¡Benutzer¿	BENUTZER	Liste alle registrierten Benutzer in der Datenbank
public	string	BenutzerID	Eindeutige ID, immer 5 Zeichen lang
public	string	Vorname	Vorname des Benutzers
public	string	Vorname	Nachname des Benutzers
public	string	Nickname	Nickname des Benutzers
public	string	Email	Email-Adresse des Benutzers
public	DateTime	Geburtsdatum	Geburtsdatum des Benutzers
public	DateTime	Erstelldatum	Erstelldatum, wann der Benutzer erstellt wurde
public	string	Passwort	Passwort des Benutzers (mindestens 8 Zeichen lang)

Methoden/Funktionen:

Modifizierer	Typ	Name	Beschreibung
public static	string	GenerateNewBenutzerID()	Generiert eine nicht schon vorhandene neue Benutzer ID
public static	Benutzer	getBenutzerFromID(string id)	Gibt das Benutzer Objekt mit der zugehörigen ID zurück. Null wenn nicht gefunden
public static	Benutzer	LoadFromDB()	Lädt alle Benutzer von der Datenbank in die BENUTZER-Liste

3.2.1 Kunde.cs

Kunde.cs ist eine Klasse, welche von Benutzer.cs erbt.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public static	List¡Kunde¿	KUNDEN	Liste alle registrierten Kunden in der Datenbank
public	float	Guthaben	Guthaben des Kunden
public	List¡Produkt¿	GekaufteProdukte	Liste der bereits gekauften Produkte
public	List¡Produkt¿	Beobachtungsliste	Liste der Produkte, welche der Kunde beobachtet
public	List¡Produkt¿	Wunschliste	Liste der Produkte, welche der Kunde sich wünscht

Methoden/Funktionen:

Modifizierer	Typ	Name	Beschreibung
public	bool	produktKaufen(Produkt produkt)	Zieht Guthaben ab und weist das Produkt den Kunde zu
public	bool	produktWuenschen(Produkt produkt)	Fügt das Produkt in die Wunschliste hinzu
public	bool	produktBeobachten(Produkt produkt)	Fügt das Produkt in der Beobachtungsliste hinzu
public	bool	produktBewerten(Produkt produkt, byte sterne, string kommentar)	Fügt eine Bewertung zum Produkt zu
public static	Benutzer	geKundeFromID(string id)	Gibt das Kunden Objekt mit der zugehörigen ID zurück. Null wenn nicht gefunden
public static	Benutzer	LoadFromDB()	Lädt alle Kunden von der Datenbank in die KUNDEN-Liste

3.2.2 Autor.cs

Autor.cs ist eine Klasse, welche von Benutzer.cs erbt.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public static	List¡Autor¿	AUTOREN	Liste alle registrierten Autoren in der Datenbank
public	Entwicklerstatus	Entwicklerstatus	Beschreibt den Entwicklerstatus des Autors
public	List¡Programmiersprache¿	Programmiersprachen	Liste der Programmiersprachen, welche der Autor beherrscht
public	List¡Produkt¿	Beobachtungsliste	Liste der Produkte, welche der Kunde beobachtet
public	List¡Produkt¿	Wunschliste	Liste der Produkte, welche der Kunde sich wünscht

Methoden/Funktionen:

Modifizierer	Typ	Name	Beschreibung
public static	Benutzer	LoadFromDB()	Lädt alle Autoren von der Datenbank in die AUTOREN-Liste

3.2.3 Mitarbeiter.cs

Mitarbeiter.cs ist eine Klasse, welche von Benutzer.cs erbt.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public static	List¡Mitarbeiter¿	MITARBEITER	Liste alle registrierten Mitarbeiter in der Datenbank
public	Aufgabenbereich	Aufgabenbereich	Gibt den Aufgabenbereich des Mitarbeiters an

3.3 Produkt.cs

In dieser Klasse werden Informationen über ein Produkt gespeichert.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public static	List¡Produkt¿	PRODUKTE	Liste alle registrierten Produkte in der Datenbank
public	string	ID	Eindeutige ID
public	string	Name	Name des Produkts
public	List¡Autor¿	Autoren	Liste aller Autoren, welche das Produkt geschrieben haben
public	float	Preis	Preis des Produkts (0 wenn es kostenfrei ist)
public	string	Beschreibung	Ausführliche Beschreibung des Produkts
public	DateTime	Veröffentlichungsdatum	Datum der Veröffentlichung des Produkts
public	List¡Bewertung¿	Bewertungen	Liste aller Bewertungen von Kunden

Methoden/Funktionen:

Modifizierer	Typ	Name	Beschreibung
public static	Benutzer	getProduktFromID(string id)	Gibt das Produkt Objekt mit der zugehörigen ID zurück. Null wenn nicht gefunden
public static	Benutzer	LoadFromDB()	Lädt alle Produkte von der Datenbank in die PRODUKTE-Liste

3.4 Bewertung.cs

In dieser Klasse werden Informationen über eine Bewertung gespeichert.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public	Kunde	Kunde	Kunde, welcher die Bewertung abgegeben hat
public	byte	Sterne	Anzahl der gegebene Sterne (1-5)
public	string	Kommentar	Kommentar zum Produkt
public	boolean	VerifizierterKauf	Ob das Produkt gekauft wurde oder nicht

Methoden/Funktionen:

Modifizierer	Typ	Name	Beschreibung
public static	Benutzer	getProduktFromID(string id)	Gibt das Produkt Objekt mit der zugehörigen ID zurück. Null wenn nicht gefunden
public static	Benutzer	LoadFromDB()	Lädt alle Produkte von der Datenbank in die PRODUKTE-Liste

3.5 DatabaseManager.cs

Dies ist eine Hilfsklasse, welche den Zugriff zur Datenbank vereinfacht.

Attribute:

Modifizierer	Typ	Name	Beschreibung
public static	DatebaseManager	Database	Globales Database Objekt
private	string	ConnectionString	Connection-String, welcher zur Verbindung benötigt wird
public	OleDbConnection	Connection	Verbindungs-Objekt zur Datenbank
public	OleDbCommand	Command	Command-Objekt zur Ausführung von Befehlen
public	OleDbDataReader	Reader	Reader-Objekt zur Auslesung von Resultaten

Methoden/Funktionen:

Modifizierer	Typ	Name	Beschreibung
public	void	execute(string sql)	Führt ein Befehl aus. Geeignet für INSERT-Befehle
public	OleDbDataReader	Read(string sql)	Führt ein Befehl aus und gibt den Reader zurück
public	void	CloseReaderAndConnection()	Schließt die Verbindung zum Reader und zur Connetion
public	List¡object[]¿	GetData(string sql)	Führt ein Befehl aus und liefert eine Liste (Reihen) mit Arrays (Spalten) von Daten
private static	void	example()	Beispiel, nicht benutzen