**M326**

Objektorientiert entwerfen und implementieren

Flavio Schaffer, Marius Burk, Yannick Folda

02.06.2022

Inhalt

[1 Einleitung 2](#_Toc105759681)

[2 Anforderungen 2](#_Toc105759682)

[2.1 Anforderungen: Funktionalität 2](#_Toc105759683)

[2.2 Anforderungen: Datenmodell 2](#_Toc105759684)

[2.2.1 Katalog: 2](#_Toc105759685)

[2.2.2 Kundenverwaltung: 2](#_Toc105759686)

[2.2.3 Ausleihe: 2](#_Toc105759687)

[3 Teil 1: OOA 3](#_Toc105759688)

[3.1 Überlegungen 3](#_Toc105759689)

[3.1.1 Architektur 3](#_Toc105759690)

[3.2 Überlegungen Use Case-Diagramm 4](#_Toc105759691)

[3.3 Domain Modell 5](#_Toc105759692)

[3.4 ERM 6](#_Toc105759693)

[4 Teil 2: OOD 6](#_Toc105759694)

[4.1 Designpatterns 6](#_Toc105759695)

[4.1.1 Fassadenpattern 7](#_Toc105759696)

[4.2 Klassendiagramm 7](#_Toc105759697)

[4.2.1 Gemeinsame Klassen 8](#_Toc105759698)

[4.2.2 UI-Schicht 8](#_Toc105759699)

[4.2.3 Anwendungsschicht 9](#_Toc105759700)

[4.2.4 Datenbankschicht 10](#_Toc105759701)

[4.3 Sequenzdiagramm 11](#_Toc105759702)

[4.3.1 Ablauf 1 (Kunde suchen) 11](#_Toc105759703)

[4.3.2 Ablauf 2 (Kunde Hinzufügen) 12](#_Toc105759704)

[4.3.3 Ablauf 3 (Adresse ändern) 13](#_Toc105759705)

[5 Abbildungsverzeichnis 14](#_Toc105759706)

[6 Quellenverzeichnis 14](#_Toc105759707)

# Einleitung

In dieser Dokumentation wird die Aufgabe aus dem ÜK-Modul 326 gelöst. Die Lösungen und Überlegungen werden festgehalten und notiert. In diesem Kurs geht es um die Objektorientierte Analyse und Design. Hierbei werden zuerst die Diagramme für die Objektorientierte Analyse und dann die Diagramme für das Objektorientierte Design erstellt. Danach werden die verschiedenen Diagramme umgesetzt. Dies Aufgabe wurde in Gruppenaufgabe gelöst. Die Gruppe besteht aus Burk Marius. Folda Yannick und Schaffer Flavio.

# Anforderungen

In der ersten Aufgabe muss ein Programm für eine Bibliothek erstellt werden. Dieses muss zuerst Analysiert und Designt werden. Hierfür gibt es verschiedene Vorgaben, die aus der Aufgabe ausgelesen werden können. Dies sind die Vorgaben:

## Anforderungen: Funktionalität

* Der Bibliothek-Mitarbeiter kann Kunden nach Namen suchen, neue zufügen und die Adresse des Kunden ändern, als auch den Katalog mit den Medien pflegen, d.h. nach Titel suchen und neue Medien zufügen.
* Zudem kann der Bibliothek-Mitarbeiter durch Eingabe der Nummer der Kundenkarte als auch der Inventarnummer des Mediums eine Ausleihe oder eine Rückgabe manuell eingeben.
* Der Kunde kann nur Ausleihen tätigen und ausgeliehene Titel und Medien zurückbringen.

## Anforderungen: Datenmodell

* Die Daten sind in einer MySQL Datenbank zu persistieren, das Datenmodell muss dabei mindestens die im Folgenden aufgelisteten Entitäten umfassen.

### Katalog:

* Vier unterschiedliche Medienarten (Buch, Audio-CD, DVD und Computer-Games)
* Folgende Angaben müssen für jedes Medium mindestens vorliegen: eindeutige ID (Inventarnummer), Titel, Genre, Altersfreigabe (ab wieviel Jahren), Autor/Interpret, ISBN/EAN, Standort-Code (Regalbezeichnung).

### Kundenverwaltung:

* Vollständige Adressverwaltung (Name, Vorname, Geburtsjahr, Strasse, Nummer, PLZ, Ort, Kundennummer)
* E-Mail für Mahnungen und Rechnungen (Rückgabe-Erinnerungen)

### Ausleihe:

* Einstellung von Konditionen (maximale Leihdauer, maximale Anzahl ausleihbare Medien)
* Automatisiertes Versenden von Rückgabe-Erinnerungen mittels Hintergrund-Prozess
* Schnittstelle für RFID (Kundenkarte, Aufkleber auf Medium)
* Rückgabekontrolle durch Bibliothek-Mitarbeiter (Verarbeitung von Medienverlusten und Beschädigungen)

# Teil 1: OOA

Dieser Teil des Dokuments befasst sich mit der Objekt-Orientierten Analyse.

Die Objekt-Orientierte Analyse sind objektorientierte Varianten des Analyseprozesses. Es wird die Architektur, das Usecase-Diagramm, Domainmodell und ERM erstellt.

## Überlegungen

In den nachfolgenen Unterkapitel sind die Überlegungen, welche in den Arbeitsprozess eingeflossen sind, festgehalten.

### Architektur

Wie in der Aufgabe vorgegeben, muss die 3-Schichten-Architektur angewendet und umgesetzt werden. Der Grund dafür ist, da in dieser die einzelnen Schichten leicht verändert werden können, ohne das ganze Programm anpassen zu müssen. Zudem bietet es mehr Strukturmöglichkeiten.

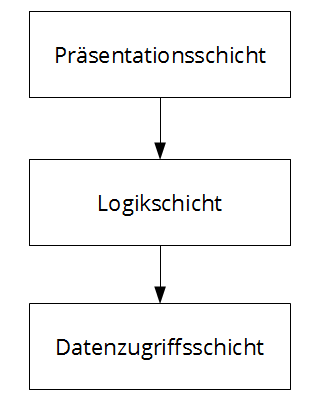


Abbildung 1 Architektur

## Überlegungen Use Case-Diagramm

In dem Use Case Diagramm werden die Rollen und Funktionen eines Programmes aufgezeichnet. In diesem Falle gibt es die Rollen Bibliothek-Mitarbeiter, Kunde, Hintergrundprozess und Datenbankadministrator. Die Funktionen wurden auf die verschiedenen Rollen aufgeteilt. Wir werden in der Umsetzung den Usecase Kunde verwalten auswählen. Dies sieht wie folgt aus:

**Bibliothek-Mitarbeiter:**

**Kunden verwalten:** Name suchen, Kunde hinzufügen, Kunde Adresse ändern,

**Kataloge pflegen:** Medien hinzufügen, Titel suchen

**Status ändern:** Name suchen, Titel suchen, Medium ausleihen, Medium zurückbringen

**Kunde:** Medium ausleihen, Medium zurückbringen

**Hintergrundprozess:** Rückgabenerinnerung versenden

**Datenbank-Administrator:**

**Datenbank verwalten:** Tabelle erstellen, lösche, bearbeiten

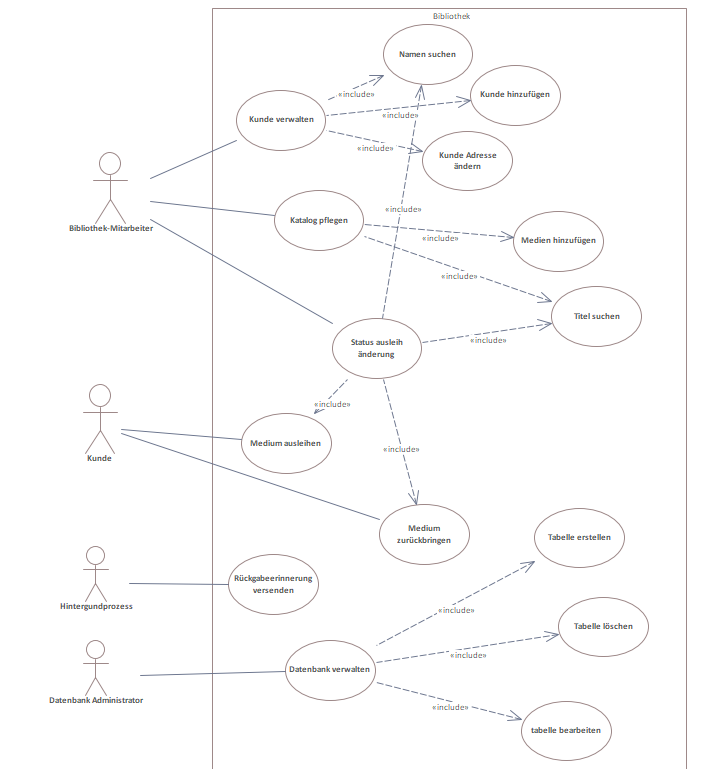


Abbildung 2: Use Case Diagramm

## Domain Modell

Im UML-Klassendiagramm werden die Klassen, Instanzen, Variablen und Methoden aufgezeigt. Des Weiteren zeigt es die Verbindungen zwischen den verschiedenen Klassen. Für die Klassen wurde

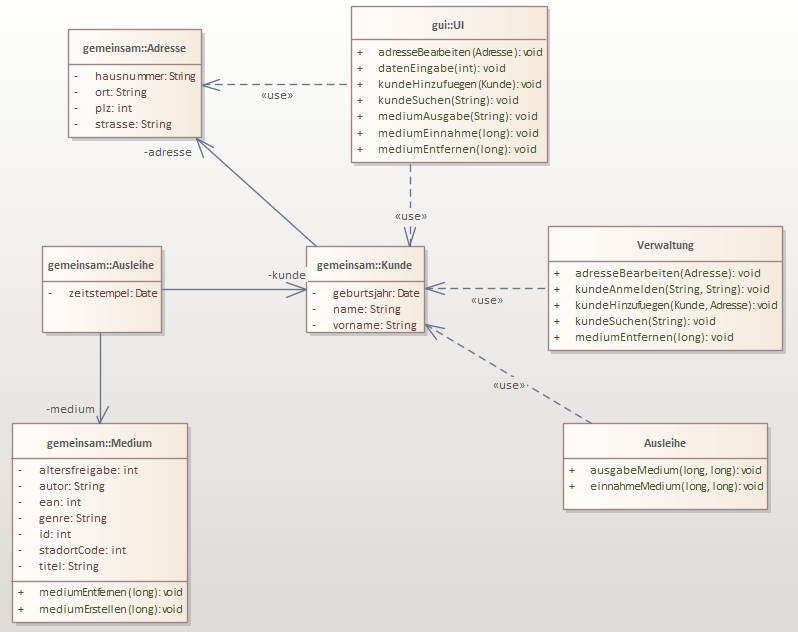


Abbildung 3 Domainmodell

## ERM

Das ERM dient dazu, einen Entwurf einer Datenbank visuell darzustellen. Dadurch kann der Architekt die Datenbank als Ganzes, mit allen Beziehungen und Constraints sehen. Hierbei kann aus dem ERM dann die Datenbank erzeugt werden. Dabei wurde zuerst überlegt welche Objekte in der Datenbank abgespeichert werden müssen anschliessend konnte aus dem ERM der SQL code generiert werden. Zudem gibt es in der Tabelle Kunde ein Fremdschlüsselverbindung von «idAdresse» in der Tabelle Adresse. Dieser Fremdschlüssel wird verwendet, dass die «idAdresse» übereinstimmt,

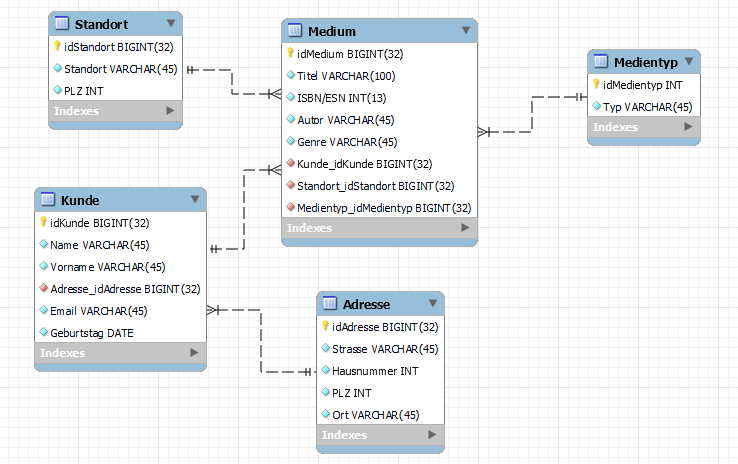


Abbildung 4 ERM

# Teil 2: OOD

Der zweite Teil des Dokuments befasst sich mit dem Objekt Orientierten Design. Im Objektorientierten Design werde die Designpatteren, das fertige Klassendiagramm und das Sequenzdiagramm erarbeitet und festgelegt.

## Designpatterns

Designpattern dienen als sogenannte Lösungsschablone für wiederkehrende Entwurfsprobleme. Es regelt einen bestimmten Aufbau eines Programmes. Es stellt eine Wiederverwendbare vorlage dar.

### Fassadenpattern

Das Fassadenpatteren ist ein Strukturmuster. Es bietet eine einheitliche und vereinfachte Schnittstelle zu verschiedenen Sub-Systemen. Es dient der Trennung der Benutzerrelevanten und der Betriebsrelevanten Vorgänge. Das Fassadenpattern wurde dem Anwendungs- und DB-Package verwendet.

## Paketübersicht

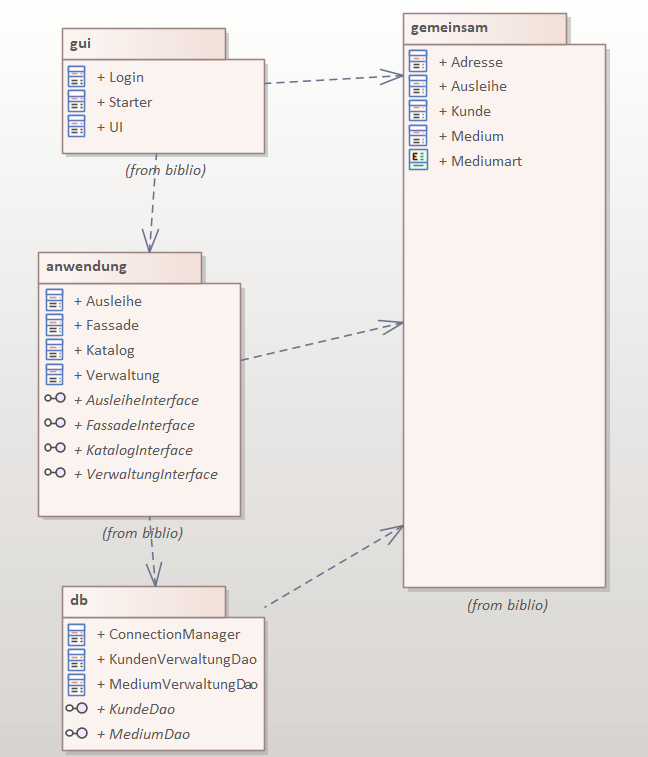


Abbildung 5 Paketübersicht

## Klassendiagramm

Das Klassendiagramm zeigt die verschiedenen Klassen und in welcher Beziehung sie Miteinander stehen. Die Klassen sind ebenfalls in den verschiedenen Schichten der Architektur dargestellt.

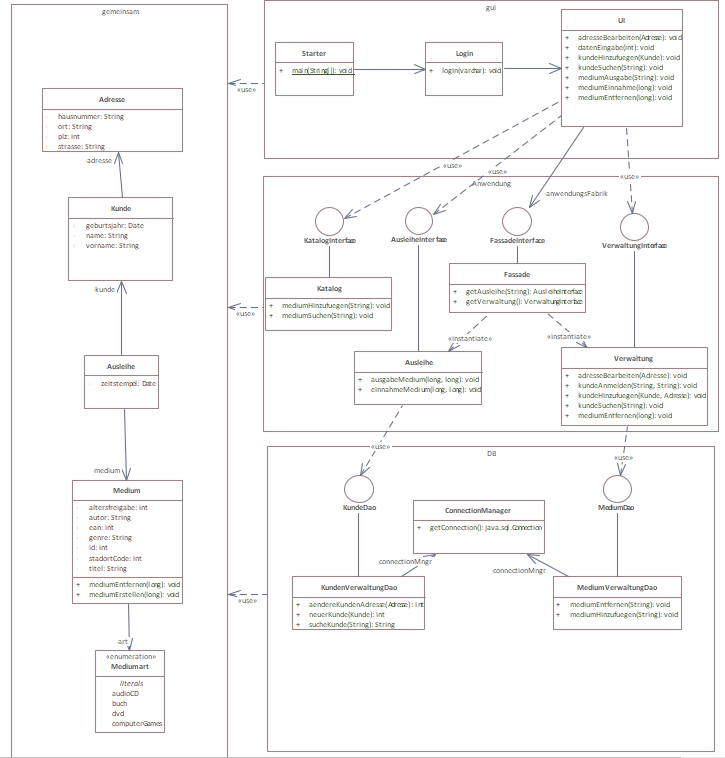


Abbildung 6 UML-Klassendiagramm

### Gemeinsame Klassen

In den gemeinsamen Klassen werden, die diejenigen Klassen verwaltet, die von jeder Schicht gebraucht werden. Dazu zählt z.B. die Klasse Kunde den Zugriff auf UI haben soll, damit sie Methoden und variablen weitergeben kann. Zudem wurde eine weiter Kommunikationsklasse zwischen Kunde und medium erstellt das Festhalten soll, wann welcher Artikel ausgeliehen wurde. Für die Inplementation war diese Klasse jedoch nicht wichtig, da andere Use Cases verwendet wurden.

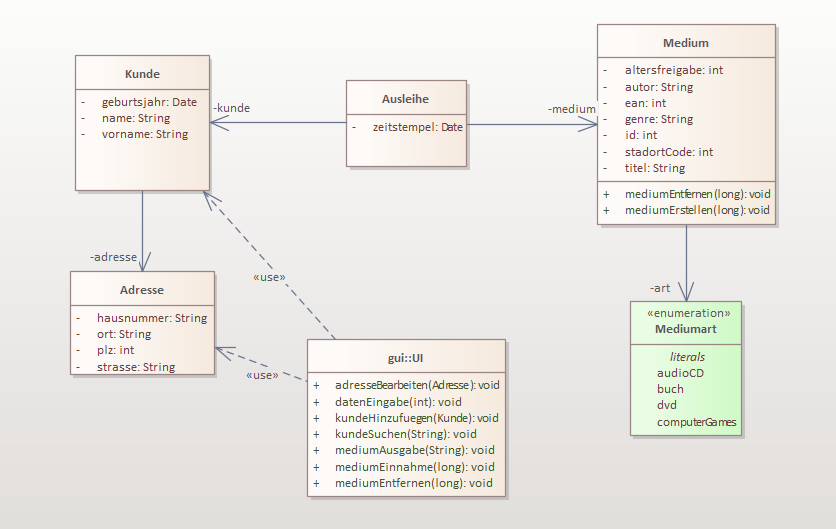


Abbildung 7 Gemeinsame Klassen

### UI-Schicht

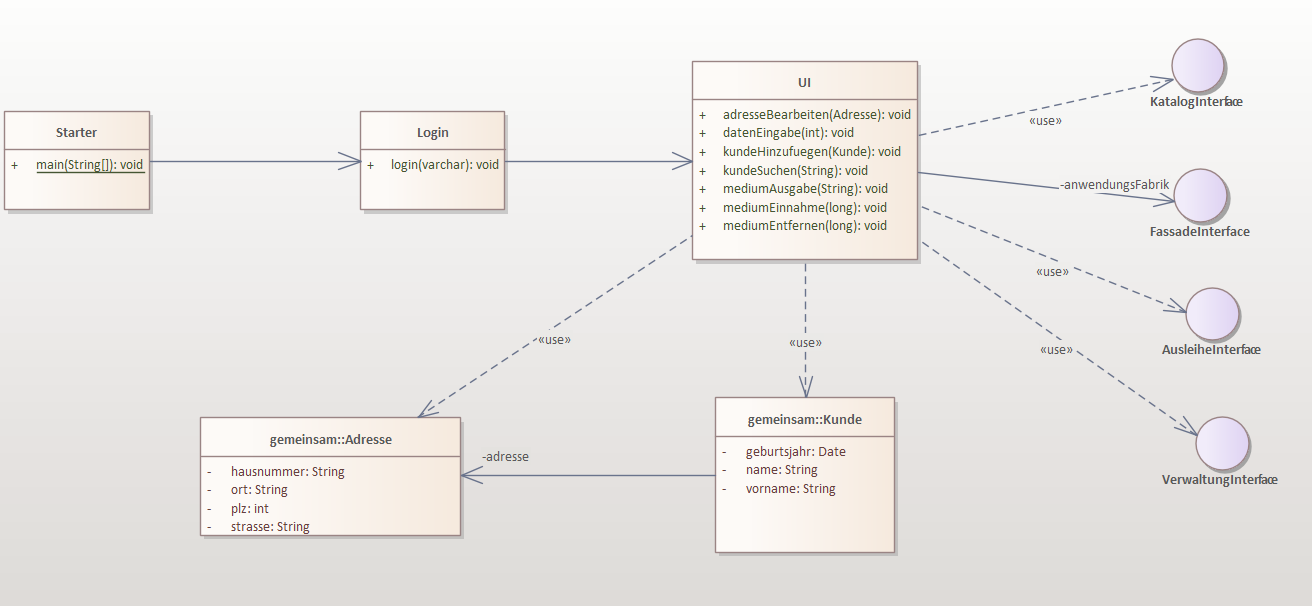


Abbildung 8 UI-Schicht

Die UI-Schicht ist die Schicht, mit welcher der Endbenutzer interagiert. Der Nutzer sollte möglichst wenig mit den internen Vorgängen konfrontiert werden.

Die UI-Schicht sorgt dafür, dass dies nicht passiert.

### Anwendungsschicht

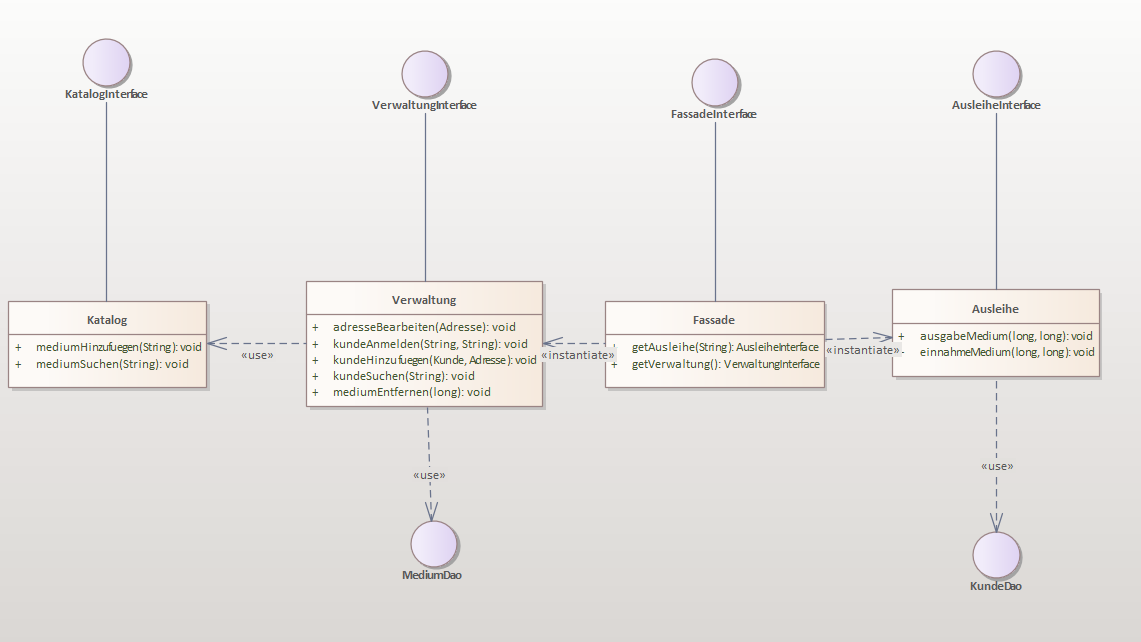


Abbildung 9 Anwendungsschicht

In der Anwendungsschicht sind jene Vorgänge, welche das Programm zum Laufen bringen.

Dazu zählen Funktionen wie Kunden hinzufügen oder nach Kunden zu suchen.

### Datenbankschicht

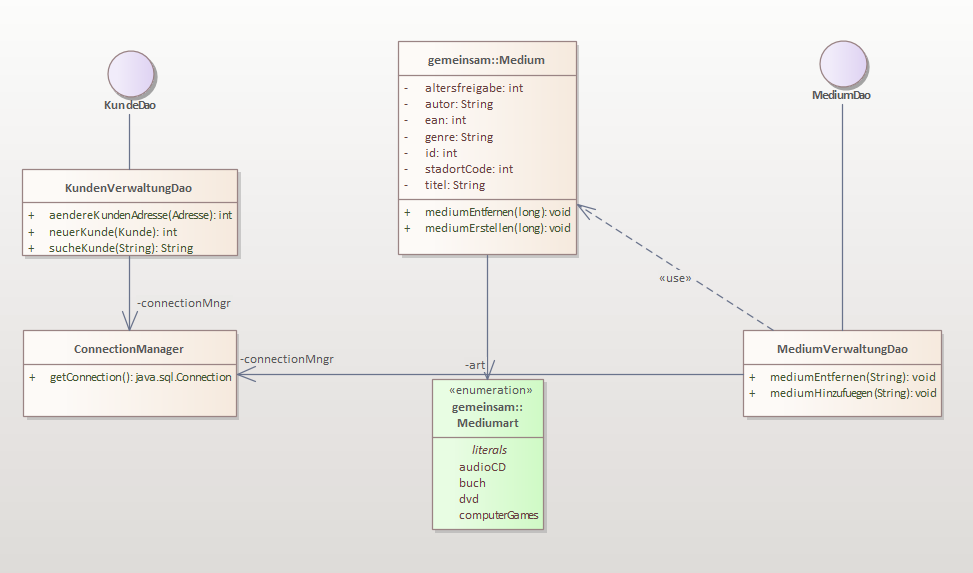


Abbildung 10 Datenbankschicht

In der Datenbankschicht werden die Daten in der Datenbank verwaltet, hier werden die eingegebenen Daten von neuen Benutzern in die DB übertragen, oder die persönlichen Daten von Nutzern ausgelesen.

## 

## Sequenzdiagramm

### Ablauf 1 (Kunde suchen)

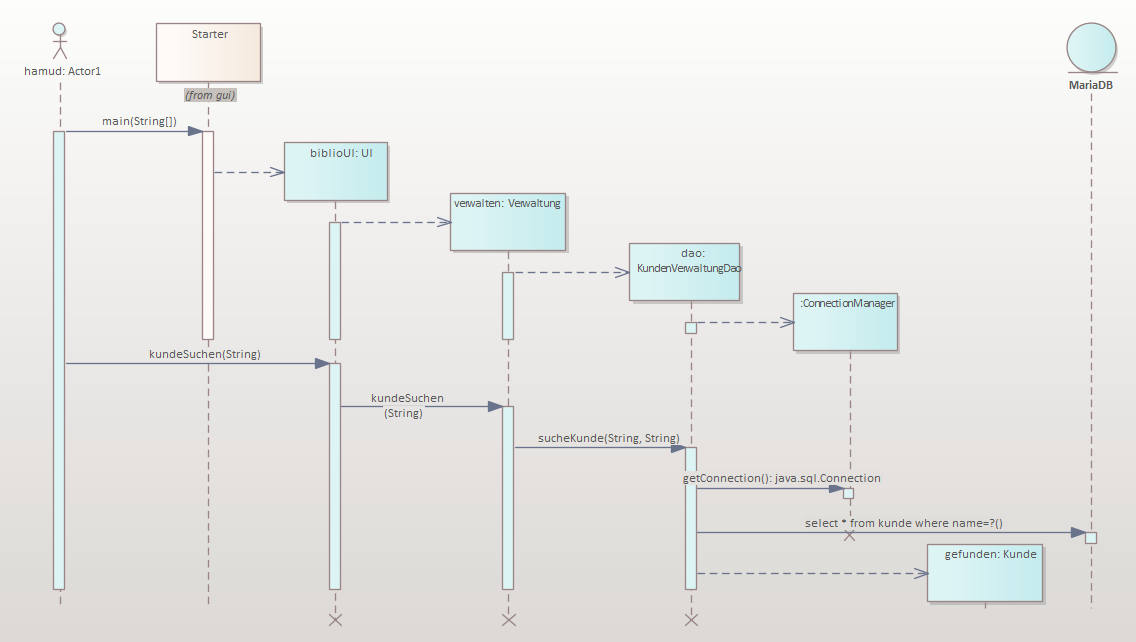


Abbildung 11 Use Case 1

In diesem ersten Sequenzdiagramm wird beschrieben, wie und in welcher Reihenfolge dies intern bewerkstelligt wird.

Einen Kunden zu suchen ist von den Vorgängen her einer der komplexeren Funktionen.

### Ablauf 2 (Kunde Hinzufügen)

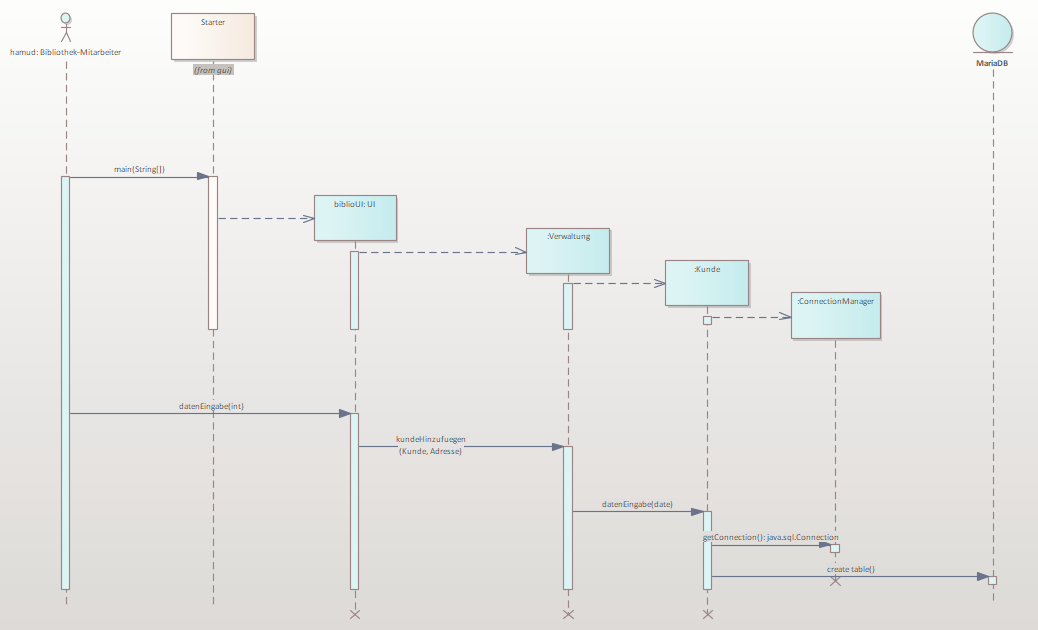


Abbildung 12 Use Case 2

Etwas simpler wird es, wenn ein Kunde hinzugefügt wird. Dazu müssen nämlich nur die persönlichen Daten erfasst werden, welche wiederum in der Datenbank abgespeichert werden.

### Ablauf 3 (Adresse ändern)

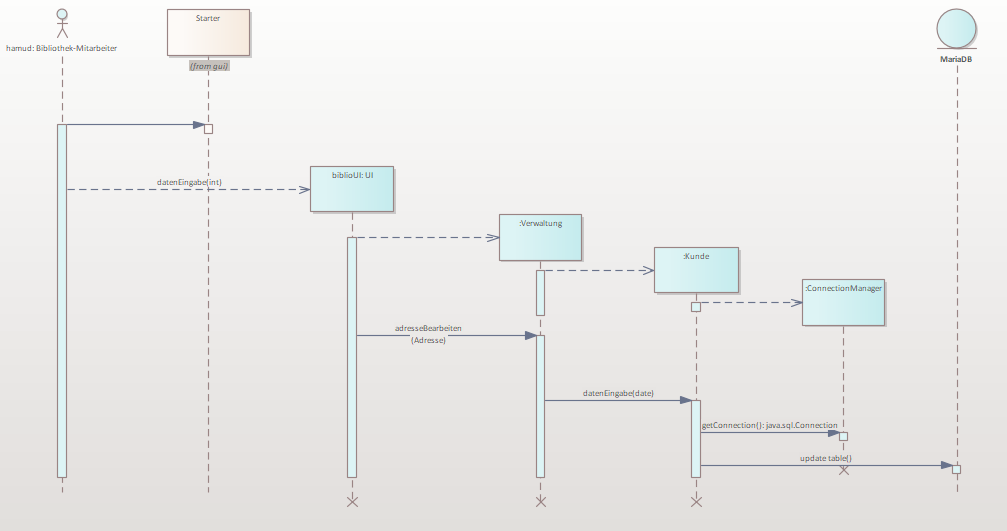


Abbildung 13 Use Case 3

Auf einem ähnlichen Komplexitätsniveau wie der Kundenerstellung ist auch das Ändern der Adresse. Es müssen nur die jeweiligen Daten erfasst werden, und die Datenbank zu „update“.

# Abbildungsverzeichnis

[Abbildung 1 Architektur 3](https://aprentas-my.sharepoint.com/personal/yannick_folda_stud_aprentas_com/Documents/M326/M326-Doku.docx#_Toc105761967)

[Abbildung 2: Use Case Diagramm 4](#_Toc105761968)

[Abbildung 3 ERM 6](#_Toc105761969)

[Abbildung 4 UML-Klassendiagramm 7](#_Toc105761970)

[Abbildung 5 Gemeinsame Klassen 8](#_Toc105761971)

[Abbildung 6 UI-Schicht 8](#_Toc105761972)

[Abbildung 7 Anwendungsschicht 9](#_Toc105761973)

[Abbildung 8 Datenbankschicht 10](#_Toc105761974)

[Abbildung 9 Use Case 1 11](#_Toc105761975)

[Abbildung 10 Use Case 2 12](#_Toc105761976)

[Abbildung 11 Use Case 3 13](#_Toc105761977)

# Quellenverzeichnis

Abbildung 1 Architektur:

[attachment-13053.png (1077×589) (mycsharp.de)](https://cdn.mycsharp.de/forumpostattachments/attachment-13053.png)