|  |
| --- |
|  |
| Projektbericht |
|  |

|  |
| --- |
| Ricardo Furtado de Gois, Fabian Küpper, Pascal Lentz, Martin Fijalkowski, Julius Wartenberg  7.6.2017 |

Inhaltsverzeichnis

[**1. Einleitung** 3](#_Toc484528239)

[**1.1 Projektbeschreibung** 3](#_Toc484528241)

[**1.2 Projektziel** 3](#_Toc484528244)

[**1.3 Projektumfeld** 3](#_Toc484528246)

[**1.4 Projektbegründung** 3](#_Toc484528248)

[**1.5 Projektabgrenzung** 4](#_Toc484528253)

[**2. Projektplanung** 4](#_Toc484528255)

[**2.1 Projektphasen** 4](#_Toc484528256)

[**2.2 Ressourcenplanung** 5](#_Toc484528258)

[**2.3 Entwicklungsprozess** 5](#_Toc484528259)

[**2.4 Ist-Analyse** 5](#_Toc484528260)

[**2.5 Soll-Konzept** 6](#_Toc484528261)

[**2.6 Wirtschaftlichkeitsanalyse** 6](#_Toc484528262)

[**2.6.1 Projektkosten** 6](#_Toc484528263)

[**2.6.2 Amortisationsdauer** 7](#_Toc484528265)

[**2.7 Entwurf der Benutzeroberfläche** 8](#_Toc484528267)

[**2.8 Datenbank** 8](#_Toc484528268)

[**2.8.1 Ermittlung der Entitäten** 8](#_Toc484528269)

[**2.8.2 Datenmodell** 9](#_Toc484528270)

[**2.9 Geschäftslogik** 9](#_Toc484528271)

[**2.10 Pflichtenheft** 9](#_Toc484528272)

[**3 Projektdurchführung** 9](#_Toc484528273)

[**3.1 Wahl der Art des Programms** 9](#_Toc484528274)

[**3.2 Entwicklungsumgebung** 10](#_Toc484528275)

[**3.3 Implementierung der Datenstrukturen** 10](#_Toc484528276)

[**3.4 Implementierung der Geschäftslogik** 10](#_Toc484528277)

[**3.5 Implementierung der Benutzeroberfläche** 11](#_Toc484528278)

[**4 Qualitätsmanagement** 11](#_Toc484528279)

[**4.1 Manuelle Test** 11](#_Toc484528280)

[**5 Projektabschluss** 11](#_Toc484528281)

[**5.1 Soll- Ist-Vergleich** 11](#_Toc484528282)

[**Glossar** 13](#_Toc484528284)

[**A.1 Verwendete Ressourcen** 14](#_Toc484528285)

[**A.2 Pflichtenheft** 15](#_Toc484528286)

[**A.3 Manuelle Testliste** 16](#_Toc484528287)

[**A.4 Entity-Relationship-Model** 17](#_Toc484528288)

[**A.5 Entwurf der Benutzeroberfläche** 18](#_Toc484528289)

[**A.6 Klassendiagramm** 20](#_Toc484528290)

[**A.7 SQL-Skript zur Erstellung der Datenbank** 21](#_Toc484528291)

[**A.8 Projektmappenverzeichnis** 24](#_Toc484528292)

# **1. Einleitung**

# Die folgende Projektdokumentation schildert den Ablauf des Mittelstufenprojektes, welches die Autoren im Rahmen ihrer Ausbildung zum Fachinformatiker Fachrichtung Anwendungsentwicklung durchgeführt haben.

# **1.1 Projektbeschreibung**

# Der Tante-Emma Laden führt im Moment alle Vorgänge noch manuell mit Papier und Stift durch. Dies macht sich negativ auf die Produktivität der Mitarbeiter aus

# In diesem Projekt sollen die gerade beschriebenen Prozesse mithilfe einer Kassen- und Verwaltungssoftware vereinfacht werden.

# **1.2 Projektziel**

# Ziel des Projektes ist die Vereinfachung der alltäglichen Vorgänge. Dazu sollen die manuellen Vorgänge durch eine Kassen- und Verwaltungssoftware ersetzt werden. Diese Software wird in Form einer Desktopanwendung mit graphischer Benutzeroberfläche und eigener Datenbank realisiert. Die Anwendung soll die Verwaltung von Kundendaten sowie die Verwaltung von Bestellungen vornehmen. Gleichzeitig sollen auch neue Daten eingepflegt werden können. Beispielsweise sollen neue Artikel in eine Lagerverwaltung eingefügt werden. Außerdem sollen alle Programmbereiche, abhängig von den Berechtigungen des Benutzers, angezeigt werden, oder bestimmte Aktionen nicht zulassen. Alle eingepflegten Daten sollen in der Datenbank automatisch abgelegt werden. Außerdem soll mit der Anwendung auch der alltägliche Kassenbetrieb vereinfacht und beschleunigt werden.

# **1.3 Projektumfeld**

# Auftraggeber des Projektes ist die Inhaberin eines Tante-Emma Ladens.

# **1.4 Projektbegründung**

# Die Hauptschwachstelle der momentanen Arbeitsprozesse, in dem Tante-Emma Laden, ist das hohe Maß an manueller Arbeit.

# Ein Beispiel hierfür wären das Erfassen und Verwalten von Kundendaten oder Bestellungen. Hierbei kann es schnell zu Flüchtigkeitsfehlern kommen, die dann wiederum zu Folgefehlern führen. Außerdem könnte es sogar sein, dass essentielle Daten gar nicht erfasst werden, weil sie einfach vergessen wurden. Des Weiteren muss in regelmäßigen Abständen ein Mitarbeiter alle vorhandenen Bestellungen durchgehen und die Bestellungen der nächsten Tage raussuchen.

# Ein weiteres Beispiel hierfür wäre der Kassenbetrieb. Momentan müssen alle Rechnungen noch manuell durchgeführt werden. Dies ist auch sehr fehleranfällig.

# Aufgrund dieser Probleme und der dadurch steigenden Fehleranfälligkeit hat sich die Inhaberin des Tante-Emma Ladens dazu entschieden, die Entwicklung einer Anwendung in Auftrag zu geben, durch die alle Arbeitsprozesse vereinfacht und beschleunigt werden sollen.

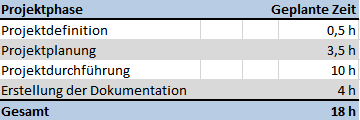
# **1.5 Projektabgrenzung**

# Da der Projektumfang auf 18 Stunden beschränkt ist, soll die Anwendung nicht mit zusätzlicher Hardware kommunizieren können, wie beispielsweise einem Bondrucker, oder einem Scanner.

# **2. Projektplanung**

# **2.1 Projektphasen**

Für die Umsetzung des Projektes standen den Autoren 18 Stunden (24 Schulstunden) zur Verfügung. Diese wurden vor Projektbeginn auf mehrere Phasen verteilt. Eine Grobe Zeitplanung sowie die einzelnen Phasen lassen sich der Tabelle 1: Grobe Zeitplanung entnehmen.



# Tabelle 1: Grobe Zeitplanung

# **2.2 Ressourcenplanung**

In der Übersicht, welche sich im Anhang A.1 Verwendete Ressourcen befindet, sind alle Ressourcen aufgelistet, die für das Projekt benötigt und eingesetzt wurden. Mit den Verwendeten Ressourcen sind sowohl Software- und Hardware- als auch Personalressourcen gemeint. Bei der Auswahl der verwendeten Software wurde darauf geachtet, dass diese kostenfrei zur Verfügung steht, oder bereits Lizenzen für die in A.1 Verwendete Ressourcen genannten Anwendungen schon vorhanden sind. Dadurch sollen anfallenden Projektkosten möglichst minimal ausfallen.

# **2.3 Entwicklungsprozess**

Bevor mit der eigentlichen Realisierung des Projektes begonnen werden konnte, mussten sich die Autoren für einen Vorgehensmodell der Softwareentwicklung entscheiden. Dieses Vorgehensmodell definiert den Ablauf der Umsetzung. Für dieses Projekt wurde von den Autoren das Wasserfall-Modell gewählt. Das Hauptmerkmal dieses Vorgehensmodells ist der lineare Projektablauf. Die nächste Phase im Projekt wird erst begonnen, sobald die vorangehende Phase abgeschlossen ist.

Durch den linearen Ablauf des Projektes und die klare Abgrenzung der einzelnen Phasen ist eine einfache und kontrollierte Planung des Projektes möglich.

In der Projektplanung wurde viel Zeit für die Implementierungsphase einkalkuliert, damit noch ausreichend Zeit für Tests vorhanden war.

# **2.4 Ist-Analyse**

Wie bereits in Abschnitt 1.1 Projektbeschreibung erwähnt müssen momentan alle Arbeitsabläufe noch manuell von dem Personal durchgeführt werden.

Alle Kunden- und Bestellungsdaten müssen momentan noch auf Papier erfasst werden und in einen extra Ordner einsortiert werden. Des Weiteren muss das Personal noch auf einem separatem Dokument notieren, welche Bestellungen in den nächsten Tagen ausgeliefert werden müssen. In regelmäßigen Abständen muss dieses Dokument auch um neue Bestellungen erweitert werden. Außerdem müssen die Mitarbeiter, um zu schauen welche Artikel noch vorrätig sind, durch den ganzen Laden laufen und die Artikel aus den Regalen raussuchen.

Letztlich werden die Rechnungen auch noch manuell von den Mitarbeitern erstellt.

Diese Arbeitsabläufe sind sehr zeitaufwändig und auch sehr fehleranfällig.

# **2.5 Soll-Konzept**

Es soll eine einfach zu bedienende und übersichtliche Anwendung erstellt werden. Die Anwendung soll eine Startseite haben, in der alltägliche Kunden schnell und einfach abkassiert werden können. Des Weiteren sollen nicht nur Käufe möglich sein, sondern auch Bestellungen erfasst werden. Die Bestellungen sollen auch auf einer eigenen Übersichtsseite dargestellt und eventuell überarbeitet werden können. Das gleiche gilt auch für Kunden. Diese müssen auch in der Anwendung eingepflegt und verwaltet werden können. Jeder Artikel in dem Laden muss auch in das Programm eingepflegt werden können. Außerdem muss auch ersichtlich sein, ob und wie viele Artikel vorrätig sind.

Es sollen auch Benutzer eingerichtet werden, die unterschiedliche Berechtigungen haben. Die Benutzer und deren Berechtigungen, lassen sich auf einer eigenen Übersicht bearbeiten.

Alle unter 2.4 Ist-Analyse beschrieben Vorgänge sollen optimiert und beschleunigt werden.

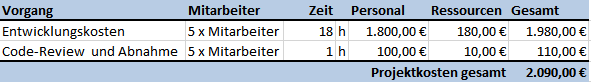
# **2.6 Wirtschaftlichkeitsanalyse**

Aufgrund der Verzögerungen durch die, in den Abschnitten 1.4 Projektbegründung und 2.4 Ist-Analyse, geschilderten Arbeitsabläufe ist die Umsetzung des Projektes erforderlich. Ob aus wirtschaftlicher Sicht die Realisierung auch gerechtfertigt ist, soll in den folgenden Abschnitten geklärt werden.

# **2.6.1 Projektkosten**

Im Folgenden werden die Projektkosten, die während der Entwicklung des Projektes anfallen, kalkuliert. Dafür werden neben den Personalkosten, die durch die Realisierung des Projektes verursacht werden, auch noch die Ressourcen berücksichtigt. Die Kalkulation wird anhand von Stundensätzen durchgeführt. Der Stundensatz eines Entwicklers beträgt 20 €. Für die Ressourcennutzung wurde ein pauschaler Stundensatz von 10 € festgelegt.

Die jeweiligen Kosten für die einzelnen Vorgänge, sowie die gesamten Projektkosten lassen sich der Tabelle 2: Kostenaufstellung entnehmen.

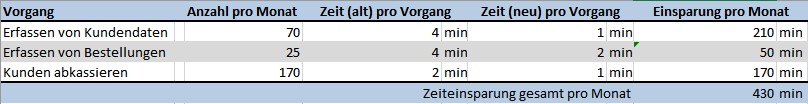


# Tabelle 2: Kostenaufstellung

# **2.6.2 Amortisationsdauer**

In diesem Abschnitt soll ermittelt werden, ab welchem Zeitpunkt sich die Projektkosten wieder amortisiert haben. Anhand des Amortisationswertes kann beurteilt werden, ob die Umsetzung des Projektes aus wirtschaftlicher Sicht sinnvoll ist. Die Amortisationsdauer wird berechnet, indem man die Anschaffungskosten durch die laufende Kostenersparnis teilt.

Durch die Vereinfachung der einzelnen Arbeitsschritte ließe sich Arbeitszeit einsparen. Durch die Einsparung der Arbeitszeit, werden auch gleichzeitig die Personalkosten reduziert. Die Zeitersparnis der einzelnen Vorgänge wird in der Übersicht Tabelle 3: Zeiteinsparung dargestellt. Die angegebenen Zeiten wurden durch die Autoren geschätzt.



# Tabelle 3: Zeiteinsparung

**Berechnung der Amortisationsdauer:**

430 x 12 = 5160 = 86

86 x (20€ + 10€) = 2580

= 0,81 Jahre = 9 Monate 3 Wochen

Anhand der Amortisationsrechnung ergibt sich für das Projekt eine Amortisationsdauer von 9 Monaten und 3 Wochen. Über diesen Zeitraum muss die Anwendung mindestens eingesetzt werden, damit sich die Anschaffungskosten und die Kosteneinsparung ausgleichen.

Da die Inhaberin des Tante-Emma Ladens diese Anwendung langfristig einsetzen möchte, kann das Projekt auch aus wirtschaftlicher Sicht als sinnvoll erachtet werden.

# **2.7 Entwurf der Benutzeroberfläche**

Um die neue Anwendung möglichst benutzerfreundlich und einfach bedienen zu können, soll eine klar strukturierte Benutzeroberfläche entwickelt werden. Aus der Hauptansicht sollten außerdem auch alle weiteren Programmteile möglichst einfach und schnell erreicht werden können. In der Hauptansicht selber, soll es möglich sein schnell neue Kunden abzukassieren.

Anhand dieser Vorgaben haben wir eine Benutzeroberfläche entwickelt. Der Entwürfe der Benutzeroberflächen befinden sich im Anhang unter A.5 Entwurf der Benutzeroberfläche.

# **2.8 Datenbank**

# **2.8.1 Ermittlung der Entitäten**

Anhand der Informationen die wir beim Kick-Off-Meeting erhalten haben, lassen sich einige Entitäten bilden. Eine Zentrale Stellen nehmen der Mitarbeiter und der Kunde ein. Daher haben sich der *Mitarbeiter* und der *Kunde* schon als Entität herausgestellt. Da jeder Mitarbeiter auch Berechtigungen besitzt, gibt es auch die Entität *Rechte.* Eine weitere zentrale Funktion soll die Bestellung von Artikeln sein. Auch hieraus lässt sich wieder schließen, dass es eine Entität *Bestellung* und eine Entität *Artikel* geben muss. Da ein Artikel direkt einer Bestellung zugeordnet werden kann, kann hier auch von einer direkten Beziehung der beiden Entitäten ausgegangen werden. Da eine Bestellung natürlich einen Kunden braucht und einen Mitarbeiter der die Bestellung bearbeitet, lässt sich der Bestellung auch direkt die Beziehung zu den Entitäten *Kunde* und *Mitarbeiter* zuweisen.

Des Weiteren umfasst eine Bestellung nicht nur eine Stückzahl eines Artikels, sondern es können auch mehrere öfter drin vorkommen. Dadurch kann ein weiteres Entity ermittelt werden, welches den Artikel, die Bestellung und die jeweilige Menge des Artikels enthält. Dieses Entity hat den Namen *Posten*.

Zuletzt wird auch noch ein Entity benötigt, das die vorhandene Menge des Artikels und die Position im Lager enthält. Somit lässt sich dem Entity *Artikel* ein weiteres Entity zuordnen. Dieses Entity haben die Autoren *Bestand* genannt.

# **2.8.2 Datenmodell**

Auf Grundlage der herausgearbeiteten Entitäten in 2.8.1 Ermittlung der Entitäten wurde ein ERD erstellt, welches die Beziehungen der einzelnen Entitäten noch einmal graphisch veranschaulichen soll. Das ERD befindet sich im Anhang A.4 Entity-Relationship-Model.

# **2.9 Geschäftslogik**

Die Klassen der Anwendung wurden aufgrund der in 2.5 Soll-Konzept festgelegten Anforderungen und der in 2.8.1 Ermittlung der Entitäten ermittelten Werte bestimmt. Das vollständige Klassendiagramm der Anwendung befindet sich im Anhang unter A.6 Klassendiagramm. Das Klassendiagramm veranschaulicht die einzelnen Klassen der Anwendung und ihre Beziehung zueinander. Dieses Klassendiagramm dient als Grundlage für die Implementierung der Geschäftslogik.

# **2.10 Pflichtenheft**

Am Ende der Planungsphase wurde mithilfe des Lastenhefts und der Vorarbeit aus der aus der Planungsphase das Pflichtenheft verfasst. Das Pflichtenheft diente als Leitfaden für die Projektdurchführung.

Das Pflichtenheft befindet sich im Anhang A.2 Pflichtenheft.

# **3 Projektdurchführung**

# **3.1 Wahl der Art des Programms**

Da für die Nutzer eine Anwendung geschaffen werden soll, welche die Informationen auf einfache Weise darstellen soll, fiel die Wahl auf eine Windowsforms-Anwendung. Als Alternative für die Windowsforms-Anwendung gab es noch eine Webanwendung sowie eine Konsolenanwendung.

Der Vorteil einer Windowsforms-Anwendung ist, dass die Entwicklungszeit für eine graphische Benutzeroberfläche wesentlich kürzer ist und die Erstellung auch einfacher ist, als bei einer Webanwendung. Des Weiteren ist eine Windowsforms-Anwendung auch schneller als eine Webanwendung.

Außerdem lässt sich eine Windowsforms-Anwendung, bei entsprechender Gestaltung, intuitiver bedienen als eine Konsolenanwendung.

# **3.2 Entwicklungsumgebung**

Das Projekt wurde auf einem Rechner mit Windows 7 als Betriebssystem entwickelt. Neben dem aktiviertem Betriebssystem, wird außerdem noch eine MySQL-Datenbank genutzt.

Die eigentlich Anwendung wurde mit Microsoft Visual Studio 2013 in C# entwickelt.

# **3.3 Implementierung der Datenstrukturen**

Auf Basis der Strukturen, die bereits in Abschnitt 2.8 Datenbank für die Anwendung definiert wurden, wurde die Datenbank implementiert. Als erstes wurde eine Neue Datenbank erstellt. Diese Datenbank wurde von den Autoren *Kassensystem* genannt. Anschließend wurden die benötigten Tabellen manuell über ein SQL-Skript erstellt. Dieses Skript wurde im Vorfeld von den Autoren vorbereitet. Das verwendete SQL-Skript befindet sich im Anhang unter A.7 SQL-Skript zur Erstellung der Datenbank.

Die Datenbank entspricht somit der in der Projektplanung definierten Struktur und erfüllt die nötigen Anforderungen.

# **3.4 Implementierung der Geschäftslogik**

Da die Implementierung der Geschäftslogik der Hauptbestandteil des ganzen Projektes ist, soll diese in dem Abschnitt genauer erläutert werden. Um die Implementierung der Geschäftslogik möglichst komfortabel zu gestalten ist die Entwicklungsumgebung ein wichtiger Faktor. Dies lässt sich dem Abschnitt 3.2 Entwicklungsumgebung entnehmen.

Zu Beginn der Projektdurchführung wurde zunächst das Grundgerüst des Projektes erstellt. Dazu wurde zunächst ein neues Projekt in der Entwicklungsumgebung erstellt. Die Struktur innerhalb des Projektes wurde anhand der GUI-Elemente in dem Projekt sortiert. Beispielsweise wurden alle für die Bestellung benötigten Klassen zu der Bestellübersicht sortiert, oder alle relevanten Klassen für Benutzer und den entsprechenden Rechten wurden zur Benutzerverwaltung sortiert. Ein Beispiel hierfür ist im Anhang unter A.8 Projektmappenverzeichnis.

Analog zu den einzelnen Klassen der Geschäftslogik wurde auch noch ein Interface definiert. Das Interface *IDBMethoden* definiert die Kommunikation zur Datenbank und erleichtert somit die Entwicklung der für die Geschäftslogik relevanten Klassen.Alle Datenbank-Tabellen sind gleichzeitig auch Klassen in der Anwendung, die dieses Interface implementiert haben.

# **3.5 Implementierung der Benutzeroberfläche**

Die Implementierung der Benutzeroberfläche wurde mit Hilfe des .NET Frameworks durchgeführt. Die Benutzeroberflächen wurden anhand der Entwürfe im Anhang A.5 Entwurf der Benutzeroberfläche gestaltet. Die Funktionalitäten die sich hinter den einzelnen GUI-Elementen befinden, wurden mit Hilfe von Events realisiert. Dies bedeutet, dass nur bei bestimmten Anwender-Aktionen der Code ausgeführt wird. Da die Methoden der einzelnen Klassen bereits vor der Implementierung der Benutzeroberfläche getestet wurden, mussten nur noch die Abläufe innerhalb der GUI-Elemente selber getestet werden.

# **4 Qualitätsmanagement**

# **4.1 Manuelle Test**

Vor der eigentlichen Erstellung der Anwendung wurde eine Testliste geschrieben, mit der die einzelnen vorgegebenen Funktionen auf ihre Richtigkeit überprüft werden konnten.

Während der Erstellung der Anwendung wurde darauf geachtet, dass die Funktionen, die in der Testliste aufgeführt sind, auch funktionieren. Dies wurde durch die regelmäßige Überprüfung der einzelnen Funktionen gewährleistet.

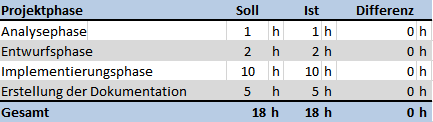
Die Testliste mit den einzelnen Testpunkten befindet sich im Anhang unter A.3 Manuelle Testliste

Bei dem abschließenden Testlauf wurden laut Testliste wie erwartet keine Mängel mehr festgestellt.

# **5 Projektabschluss**

# **5.1 Soll- Ist-Vergleich**

Bei einer rückblickenden Betrachtung des Projektes, kann festgehalten werden, dass alle zuvor festgelegten Anforderungen gemäß dem Pflichtenheft (A.2 Pflichtenheft) erfüllt wurden. Die Zeitplanung, die in dem Abschnitt 2.1 Projektphasen festgelegt wurde, konnte auch eingehalten werden. In der Tabelle 4: Soll-/Ist-Vergleich wird die tatsächliche Zeit, die für das Projekt benötigt wurde, der zuvor eingeplanten Zeit gegenübergestellt. Es ist zu erkennen, dass es keine Abweichungen von der Zeitplanung gab.



# Tabelle 4: Soll-/Ist-Vergleich

# **Glossar**

MySQL Relationales Datenbankmanagementsystem

GUI Graphical User Interface

ERD Entity-Relationship-Diagramm

# **A.1 Verwendete Ressourcen**

**Hardware**

* Arbeitsplätze mit Rechnern
* Tastatur
* Maus

**Software**

* Windows 7 – Betriebssystem
* Visual Studio 2013 – Entwicklungsumgebung C#
* MySQL – Datenbanksystem
* MySQL Workbench – Verwaltungswerkzeug für MySQL-Datenbanken
* UMLET – Tool zur Erstellung von Diagrammen (Klassendiagramm, Sequenzdiagramm, usw.)

**Personal**

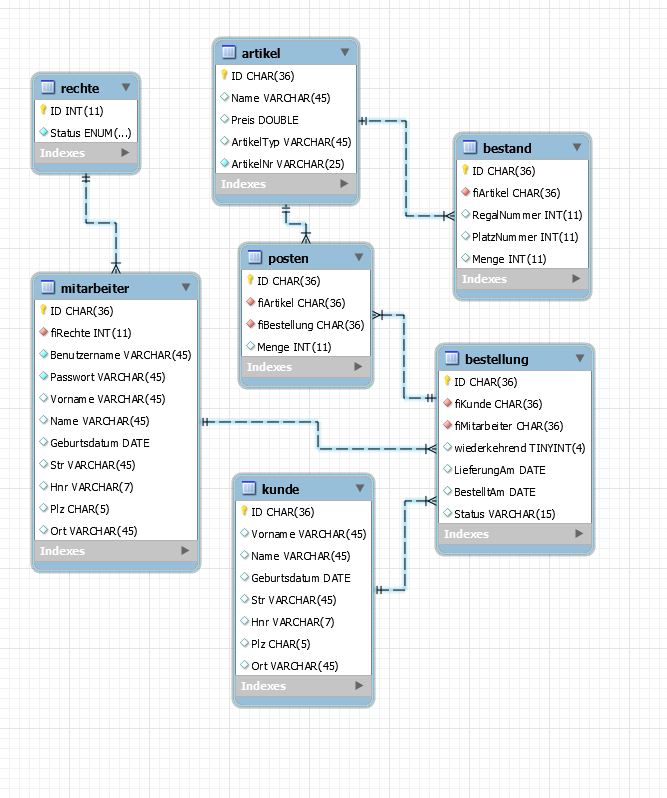
* 5 Entwickler – Umsetzung des Projektes

# **A.2 Pflichtenheft**

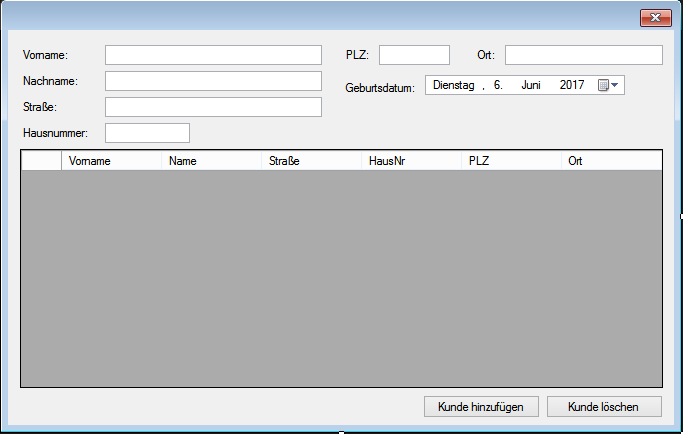
# **A.3 Manuelle Testliste**

|  |  |  |  |
| --- | --- | --- | --- |
| Testschritt | Soll-Spezifikation | Ergebnis | Status |
| 1 | Das Programm sollte in der Lage sein, einen Kunden zu löschen, neu anzulegen, oder auch zu bearbeiten. |  |  |
| 2 | Es muss eine Übersicht geben, in der Alle Kunden aufgelistet sind. |  |  |
| 3 | Es müssen Bestellungen erfasst werden können. |  |  |
| 4 | Die Bestellungen müssen in einer Übersichtsliste aufgelistet werden. |  |  |
| 5 | Die Bestellungen müssen auch bearbeitet, oder gelöscht/storniert werden können. |  |  |
| 6 | Eine Bestellung muss einem Kunden zugeordnet werden können. |  |  |
| 7 | Es muss eine Übersicht geben, in der alle Artikel und alle vorrätigen Artikel aufgelistet werden. |  |  |
| 8 | Es muss eine Kassen-Übersicht geben, in der normale Kunden abkassiert werden können. |  |  |
| 9 | Es muss eine Übersicht geben, in der alle Benutzer/Mitarbeiter aufgelistet sind. |  |  |
| 10 | Jeder Mitarbeiter/Benutzer muss Rechte besitzen. |  |  |
| 11 | Die einzelnen Funktionen der Anwendung müssen von der Berechtigung des Benutzers/Mitarbeiters abhängig sein. |  |  |
| 12 | Beim Verkauf, oder bestellen, von Artikeln muss automatisch ein Lagerabgang verbucht werden. |  |  |
| 13 | Die Anwendung muss eine Lager-Übersicht haben, in der ein Lagerzugang verbucht werden kann. |  |  |
| 14 | Es sollen nur vorhandene Artikel gekauft oder bestellt werden können. |  |  |

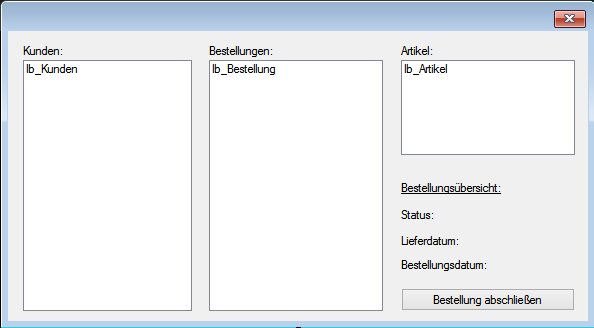
# **A.4 Entity-Relationship-Model**



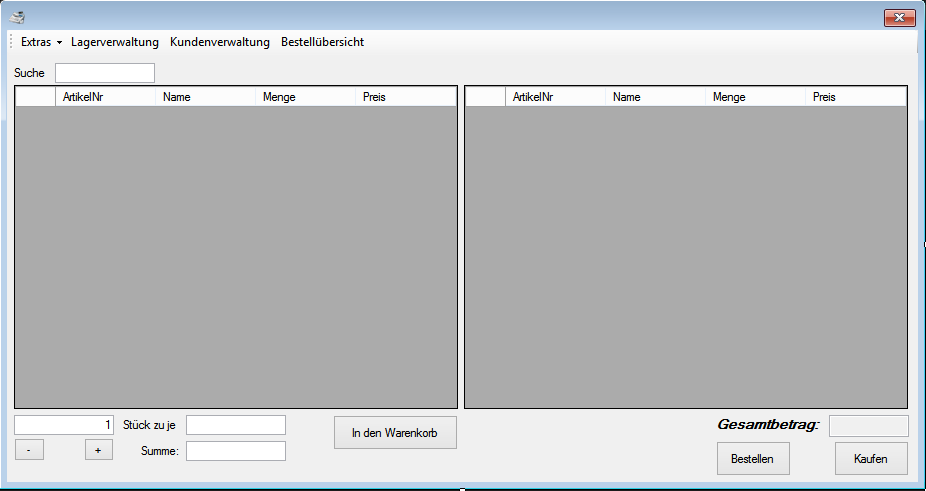
# **A.5 Entwurf der Benutzeroberfläche**



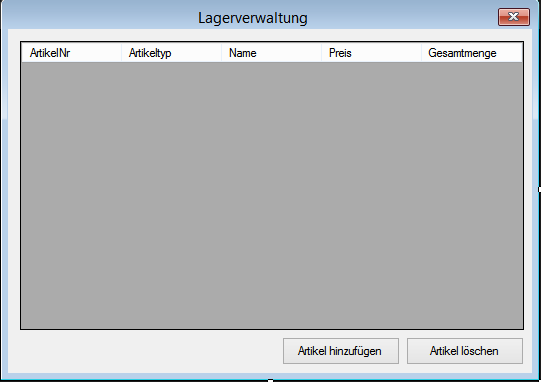
Entwurf der Kundenverwaltung



Entwurf der Bestellungsübersicht

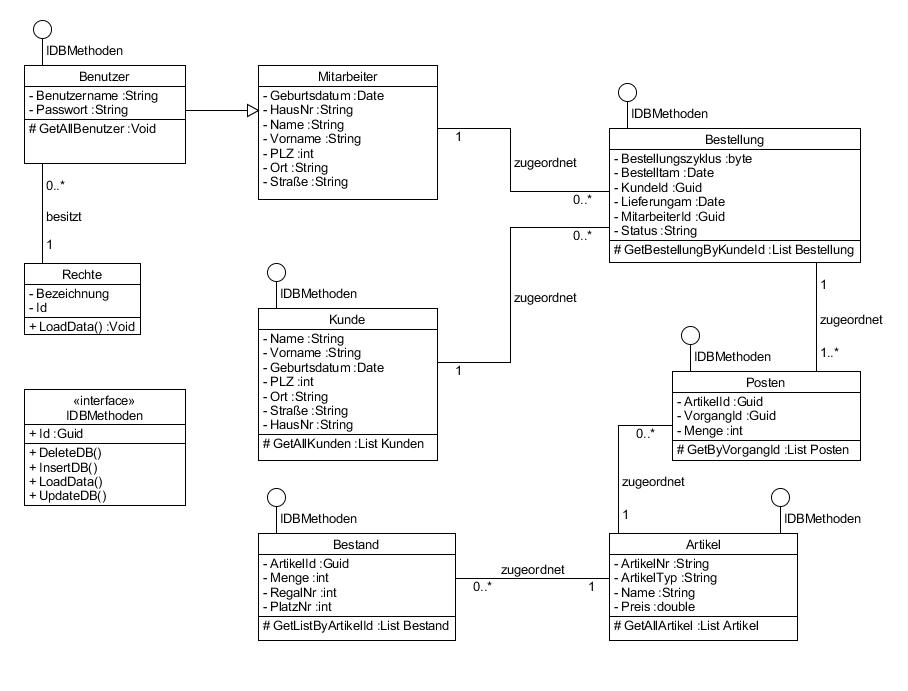


Entwurf der Hauptseite



Entwurf der Lagerverwaltung

# **A.6 Klassendiagramm**



# **A.7 SQL-Skript zur Erstellung der Datenbank**

DROP DATABASE IF EXISTS EMMA;

CREATE DATABASE EMMA;

USE EMMA;

DROP TABLE IF EXISTS Rechte;

DROP TABLE IF EXISTS Artikel;

DROP TABLE IF EXISTS Bestand;

DROP TABLE IF EXISTS Bestellung;

DROP TABLE IF EXISTS Posten;

DROP TABLE IF EXISTS Person;

DROP TABLE IF EXISTS Mitarbeiter;

CREATE TABLE `Rechte` (

`ID` int(11) NOT NULL,

`Status` enum('Hilfskraft','Personal','Chef') NOT NULL,

PRIMARY KEY (`ID`)

) ENGINE=InnoDB;

CREATE TABLE `Kunde` (

`ID` char(36) NOT NULL,

`Vorname` varchar(45) DEFAULT NULL,

`Name` varchar(45) DEFAULT NULL,

`Geburtsdatum` DATE ,

`Str` varchar(45) DEFAULT NULL,

`Hnr` varchar(7) DEFAULT NULL,

`Plz` char(5) DEFAULT NULL,

`Ort` varchar(45) DEFAULT NULL,

PRIMARY KEY (`ID`)

) ENGINE=InnoDB ;

CREATE TABLE `Mitarbeiter` (

`ID` char(36) NOT NULL,

`fiRechte` int(11) NOT NULL,

`Benutzername` varchar(45) NOT NULL,

`Passwort` varchar(45) NOT NULL,

`Vorname` varchar(45) DEFAULT NULL,

`Name` varchar(45) DEFAULT NULL,

`Geburtsdatum` DATE ,

`Str` varchar(45) DEFAULT NULL,

`Hnr` varchar(7) DEFAULT NULL,

`Plz` char(5) DEFAULT NULL,

`Ort` varchar(45) DEFAULT NULL,

PRIMARY KEY (`ID`),

KEY `fiRechte` (`fiRechte`),

FOREIGN KEY (`fiRechte`) REFERENCES `rechte` (`ID`) ON DELETE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `Artikel` (

`ID` char(36) NOT NULL,

`Name` varchar(45) NULL,

`Preis` double DEFAULT NULL,

`ArtikelTyp` varchar(45) DEFAULT NULL,

`ArtikelNr` varchar(25) NOT NULL,

PRIMARY KEY (`ID`)

) ENGINE=InnoDB;

CREATE TABLE `Bestand` (

`ID` char(36) NOT NULL,

`fiArtikel` char(36) NOT NULL,

`RegalNummer` int(11) DEFAULT NULL,

`PlatzNummer` int(11) DEFAULT NULL,

`Menge` int(11) DEFAULT NULL,

PRIMARY KEY (`ID`),

CONSTRAINT `bestand\_ibfk\_1` FOREIGN KEY (`fiArtikel`) REFERENCES `artikel` (`ID`) ON DELETE CASCADE

) ENGINE=InnoDB ;

CREATE TABLE `Bestellung` (

`ID` char(36) NOT NULL,

`fiKunde` char(36) NOT NULL,

`fiMitarbeiter` char(36) NOT NULL,

`wiederkehrend` tinyint DEFAULT NULL,

`LieferungAm` date DEFAULT NULL,

`BestelltAm` date DEFAULT NULL,

`Status` varchar(15) DEFAULT NULL,

PRIMARY KEY (`ID`),

KEY `fiKunde` (`fiKunde`),

KEY `fiMitarbeiter` (`fiMitarbeiter`),

CONSTRAINT `bestellung\_ibfk\_1` FOREIGN KEY (`fiKunde`) REFERENCES `kunde` (`ID`) ON DELETE CASCADE,

CONSTRAINT `bestellung\_ibfk\_2` FOREIGN KEY (`fiMitarbeiter`) REFERENCES `mitarbeiter` (`ID`) ON DELETE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `Posten` (

`ID` char(36) NOT NULL,

`fiArtikel` char(36) NOT NULL,

`fiBestellung` char(36) NOT NULL,

`Menge` int(11) DEFAULT NULL,

PRIMARY KEY (`ID`),

KEY `fiArtikel` (`fiArtikel`),

KEY `fiBestellung` (`fiBestellung`),

CONSTRAINT `posten\_ibfk\_1` FOREIGN KEY (`fiArtikel`) REFERENCES `artikel` (`ID`) ON DELETE CASCADE,

CONSTRAINT `posten\_ibfk\_2` FOREIGN KEY (`fiBestellung`) REFERENCES `bestellung` (`ID`) ON DELETE CASCADE

) ENGINE=InnoDB;

# **A.8 Projektmappenverzeichnis**

