Abschlussprüfung Winter 2022/2023

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur Projektarbeit

**Patienten- und Terminverwaltung für eine Gemeinschaftspraxis**

**Webanwendung zur Erfassung und Verwaltung von**

**Patientendaten mit Terminvergabe**

Abgabetermin 15.12.2022

**Prüfungsbewerber:**

Frank Bartl  
Ortsstraße 15  
63928 Eichenbühl-Guggenberg

**Ausbildungsbetrieb:**

Berufsförderungswerk Schömberg gGmbH

Bühlhof 6

75328 Schömberg

**Projektbetreuer:**

Holger Bube  
Reha Ausbilder

# Einleitung

Heutzutage ist es kaum vorstellbar, ohne eine digitalisierte und effiziente Patienten- und Terminverwaltung zu arbeiten, da diese den Organisations-Aufwand einer Praxis erheblich verbessert.

Diese Dokumentation dient dazu einen Überblick zu verschaffen, wie eine Patienten- und Terminverwaltungs-Software für eine Arztpraxis erdacht und implementiert werden kann. Es handelt sich um einen fiktiven Kunden, eine Gemeinschaftspraxis mit 4 Ärzten und rund 300 Patienten.

Die Einführung der Software wird dazu beitragen, die Arbeitsauslastung koordinierter zu gestalten und damit die Effizienz zu steigern.

# Hinweise zur Lesbarkeit

Um den Lesefluss zu gewährleisten, wird an einigen Stellen bei Personenbezeichnungen und personenbezogenen Hauptwörtern ausschließlich die männlichen Form verwendet. Im Sinne der Gleichbehandlung gelten entsprechende Begriffe grundsätzlich für alle Geschlechter.

Für eine bessere Übersicht sind die Überschriften der einzelnen Kapitel farblich hinterlegt. Auch können die Überschriften im Inhaltsverzeichnis bei der Desktopvariante angeklickt werden, um zum entsprechenden Kapitel zu springen.   
Wichtige Sätze oder Schlagwörter sind fett und/oder kursive geschrieben. Für Abkürzungen und Tabellen ist ein separates Verzeichnis vorhanden.

# Rechtliche Hinweise

Der hier Aufgeführte Inhalt unterliegt dem deutschen Urheberrecht. Die Vervielfältigung, Bearbeitung, Verbreitung und jede Art der Verwertung außerhalb der Grenzen des Urheberrechts bedürfen der schriftlichen Zustimmung des Autors bzw. Erstellers.

# Inhaltsverzeichnis

[Einleitung II](#_Toc120601889)

[Hinweise zur Lesbarkeit II](#_Toc120601890)

[Rechtliche Hinweise II](#_Toc120601891)

[Inhaltsverzeichnis III](#_Toc120601892)

[Abbildungsverzeichnis V](#_Toc120601893)

[Abkürzungsverzeichnis V](#_Toc120601894)

[Tabellenverzeichnis V](#_Toc120601895)

[1 Projektbeschreibung 1](#_Toc120601896)

[1.1 Ist-Zustand 1](#_Toc120601897)

[1.2 Problematik 1](#_Toc120601898)

[1.3 Soll-Zustand 1](#_Toc120601899)

[1.4 Projektbegründung 2](#_Toc120601900)

[1.5 Abweichungen gegenüber der Projektantrag 2](#_Toc120601901)

[1.6 Projektschnittstellen 2](#_Toc120601902)

[2 Projektplanung 2](#_Toc120601903)

[2.1 Projektphasen 2](#_Toc120601904)

[2.2 Personalplanung 3](#_Toc120601905)

[2.3 Gesamtkosten 3](#_Toc120601906)

[2.4 Sachmittelplanung 4](#_Toc120601907)

[2.5 Wirtschaftliche Betrachtung 4](#_Toc120601908)

[2.6 Ablaufplanung 5](#_Toc120601909)

[3 Entwurfsphase 7](#_Toc120601910)

[3.1 Zielplattform 7](#_Toc120601911)

[3.2 Layoutdesign 7](#_Toc120601912)

[3.3 Anwendungsfälle 8](#_Toc120601913)

[3.4 Relationale Datenbank 9](#_Toc120601914)

[3.5 Relationales Datenmodell 10](#_Toc120601915)

[4 Implementierung 10](#_Toc120601916)

[4.1 Erstellen der Datenbank 11](#_Toc120601917)

[4.2 Verbindung der Datenbank mit Visual Studio 11](#_Toc120601918)

[4.3 Implementierung der Benutzeroberfläche 12](#_Toc120601919)

[4.4 Implementierung der Features Alle Patienten und Suchen 13](#_Toc120601920)

[5 Fazit 15](#_Toc120601921)

[5.1 Soll-/Ist-Vergleich 16](#_Toc120601922)

[5.2 Lessons Learned 17](#_Toc120601923)

[5.3 Ausblick 17](#_Toc120601924)

[Anhang 19](#_Toc120601925)

[A1 EPK-Modell 19](#_Toc120601926)

[A2 ER-Modell 20](#_Toc120601927)

[A3 Relationales Datenmodell 21](#_Toc120601928)

[A4 Datenbank Code 22](#_Toc120601929)

# Abbildungsverzeichnis

[Abbildung 1 Projektstrukturplan 6](#_Toc120709026)

[Abbildung 2 GANTT-Diagramm 6](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120709027)

[Abbildung 3 Skizze Layout 8](#_Toc120709028)

[Abbildung 4 Anwendungsfalldiagramm 9](#_Toc120709029)

[Abbildung 5 Teilausschnitt ER-Modell 10](#_Toc120709030)

[Abbildung 6 Verbindungs Tools 11](#_Toc120709031)

[Abbildung 7 Startseite 12](#_Toc120709032)

[Abbildung 8 Patientenseite 12](#_Toc120709033)

[Abbildung 9 Sonstiges 13](#_Toc120709034)

[Abbildung 10 PatientenKomplett 14](#_Toc120709035)

[Abbildung 11 Suchfunktion von Patienten 15](#_Toc120709036)

# Abkürzungsverzeichnis

E

EPK *Ereignisgesteuerte Prozesskette*

ER-Models *Entity-Relationship-Modell*

M

MariaDB Datenbank

O

OR-Mapper 7

# Tabellenverzeichnis

[Tabelle 1 Grobe Zeitplanung 3](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120786473)

[Tabelle 2 Gesamtkosten 4](#_Toc120786474)

[Tabelle 3 Materialkosten vor Einführung 5](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120786475)

[Tabelle 4 Miterbeiterlohnkosten vor Einführung 5](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120786476)

[Tabelle 5 Materialkosten nach Einführung 5](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120786477)

[Tabelle 6 Mitarbeiterlohnkosten nach Einführung 6](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120786478)

[Tabelle 7 Ersparnis der Gesamtkosten im Jahr 6](https://itbfw-my.sharepoint.com/personal/frank_bartl_it-bfw_de/Documents/Dokumente/GitHub/Application.java/Dokumente/Dokumentation_Patientenverwaltung_Abschlussprüfung.docx#_Toc120786479)

[Tabelle 8 Soll-/Ist-Vergleich Kosten 17](#_Toc120786480)

[Tabelle 9 Soll-/Ist-Vergleich Zeit 18](#_Toc120786481)

# Projektbeschreibung

Die folgende Dokumentation schildert den Ablauf des Projekts für die Abschlussprüfung der IHK, welches der Autor im Rahmen der Umschulung zum Fachinformatiker Fachrichtung Anwendungsentwicklung durchgeführt hat. Der Ausbildungsbetrieb ist das Berufsförderungswerk Schömberg gGmbH welcher bundesweit vertreten ist. Es unterstützt Menschen dabei nach langer Krankheit oder anderen Defiziten, die ihre alte Arbeit nicht mehr ausüben können, wieder zurück in die Arbeitswelt zu kommen.

Es handelt sich bei diesem Projekt um ein fiktives Szenario. Der Kunde, "Dr. Klein & Kollegen" haben eine Gemeinschaftspraxis für Allgemeinmedizin. Sie praktizieren seit rund 20 Jahren im Stadtzentrum von Miltenberg. Das Praxisteam ist besteht aus 2 Ärzten und 8 Arzthelferinnen. Der Patientenstamm umfasst ca. 300 Patienten und wächst weiter an.

## Ist-Zustand

Die Patienten- und Terminverwaltung erfolgt aktuell nur in handschriftlicher Form. Externe Befunde die per E-Mail gesendet werden, werden ausgedruckt und in die Karteikarte einsortiert. Des Weiteren werden die Patientenkarten vom Praxispersonal in den Behandlungsraum gebracht. Nach der Behandlung werden die Kateikarten in den Aktenschrank einsortiert.

## Problematik

Die aktuelle Situation in der Arztpraxis kostet wertvolle Arbeitszeit und Ressourcen, wie z.B. Karteikarten und Toner. Die Arbeitsorganisation ist ineffizient und die handschriftlichen Einträge in die Karteikarte schwer nachvollziehbar, da stellenweise sehr unleserlich geschrieben wird.

## Soll-Zustand

Der Kunde wünscht sich eine Webanwendung, die dabei hilft, die Termin- und Patientenverwaltung zu erleichtern.

Mit Hilfe der Anwendung wird die gesamte Verwaltung digitalisiert, Termine können bequem einem Arzt zugeordnet werden, eine versehentliche Doppelbelegung ist nicht mehr möglich. Befunde und Behandlungsnotizen werden der digitalen Patientenkarteikarte zugeordnet.

Bei der Einführung wird es eine Schulung für die Mitarbeiter geben, damit diese sicher mit dem Programm umgehen können. Für weitere Unterstützung wird ein Support vertrag erstellt.

## Projektbegründung

Durch die aktuelle Situation ist das Praxispersonal, die meiste Zeit mit organisatorischen Tätigkeiten beschäftigt, Karteikarten für den Termin herauszusuchen und dem Arzt ins Behandlungszimmer zulegen. Nach dem Termin bringen die Ärzte die Karten zurück an den Empfang, damit diese wieder einsortiert werden kann. Dadurch haben die Arzthelferinnen weniger Zeit für Behandlungen wie Blutdruck messen, Blutabnehmen EKG und dergleichen, was für die Patienten deutlich längere Wartezeiten bedeutet.

Mit Einführung der Verwaltungssoftware, wird der organisatorische Aufwand erheblich reduziert. Die Arzthelferinnen brauchen lediglich neue Patienten über die Software anlegen. Der Arzt kann sich selbstständig über die Suchfunktion bequem den Patienten des jeweiligen Termins anzeigen lassen und weitere Behandlungen oder neue Diagnosen zu dem Patienten hinzufügen.   
Allgemein wird ein effizienteres Arbeiten ermöglicht, was eine höhere Anzahl von Patienten bedeutet, die an einem Tag behandelt werden können. Weiter können massiv Ressourcen eigespart werden, was nicht nur die Kosten reduziert, es ist ein großer Beitrag zum Umweltschutz.

## Abweichungen gegenüber dem Projektantrag

Im Vergleich zum Projektantrag ist der Punkt Code Review hinzugekommen, dieser dient zur ersten Überprüfung des Quellcodes durch einen weiteren Mitarbeiter.

## Projektschnittstellen

Das Projekt wird von dem Autor zum größten Teil allein umgesetzt. Um die Wünsche des Kunden zu erfüllen, kommt es in regelmäßigen Abständen zu Austausch mit dem Arzt Dr Klein. Des Weiteren gibt es einen Kollegen, der für das Code Review eingesetzt wird.

# Projektplanung

Im Kapitel Projektplanung wird im Wesentlichen erklärt, wie die Planung im Allgemeinen stattgefunden hat. Es werden die Phasen der Planung und die Kosten Berechnung dargestellt.

## Projektphasen

Die Projektphasen sind in die folgenden Punkte unterteilt:

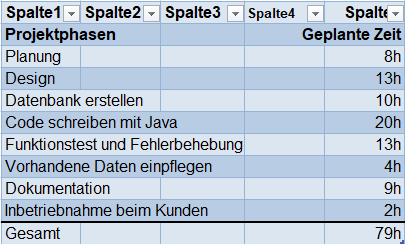
* Kundenwünsche aufnehmen
* Oberflächen Design
* Datenbank erstellen
* Code schreiben Java
* Dokumentation
* Vorhandene Daten einpflegen
* Tests und Fehlerbehebung
* Inbetriebnahme beim Kunden inklusive Mitarbeiterschulung

Tabelle 1 Grobe Zeitplanung

Die *Tabelle 1 zeigt eine* Grobe Zeitplanung. Da es sich um ein fiktives Projekt handelt sind die Zeiten ohne Gewähr. Was von dieser groben Zeiteiteilung wirklich eintrifft bzw. wie sich die Zeiten im Verlaufe des Projekts verschieben wird im Kapitel *Fazit* erläutert.

## Personalplanung

Für die Umsetzung des Projekts wird eine Person eingesetzt. Lediglich für die Qualitätssicherung wird eine zusätzliche Person eingesetzt, diese wird ein Code Review durchführen.   
Der Stundensatz liegt bei 95€. Bei der Planung wird von einer Arbeitszeit von 79 Std ausgegangen. Daraus ergibt sich Personaleinsatz ohne MwSt. in € von 7.505,00€. Die Gesamtkosten können im Kapitel eingesehen *Gesamtkosten* werden.

## Gesamtkosten

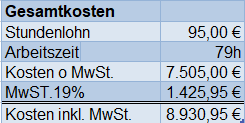
Wie man der *Tabelle 2 Gesamtkosten* entnehmen kann, stellen sich diese aus den Personalkosten plus der Mehrwertsteuer zusammen.

Tabelle 2 Gesamtkosten

## Sachmittelplanung

Die benötigten Sachmittel für dieses Projekt sind:

Hardware und Software

* Desktop PC
* Microsoft Visio → zum Erstellen des ER-Models und EPK
* Microsoft Projekt → zum Erstellen des GANTT Diagramms
* IntelliJ IDEA Community Edition → Erarbeiten des Codes und Oberflächen Gestaltung
* HeidiSQL → Bereitstellung der Datenbank MariaDB
* GitHub → Visionsverwaltung

## Wirtschaftliche Betrachtung

Durch die Einführung der Patientenverwaltungssoftware werden die organisatorischen Strukturen der Praxis verändert. Die Arzthelferinnen können sich mehr auf die Behandlung und Betreuung der Patienten konzentrieren. Im Behandlungsraum gibt es zudem keine Wartezeiten mehr durch evtl. falsche Patientenkarten. Der gesamte Ablauf wird reibungsloser und hat weniger Fehlerquellen.

## Amortisationsrechnung

Die nachfolgenden Tabellen zeigen die Material- und Mitarbeiterlohnkosten vor und nach der Einführung der Verwaltungssoftware.

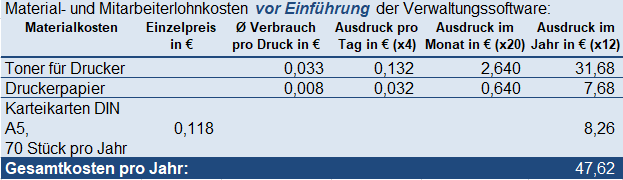
Material- und Mitarbeiterlohnkosten ***vor Einführung*** der Verwaltungssoftware

Tabelle 3 Materialkosten vor Einführung

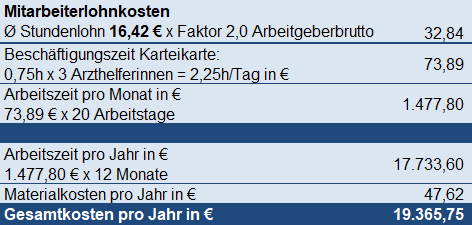


Tabelle 4 Miterbeiterlohnkosten vor Einführung

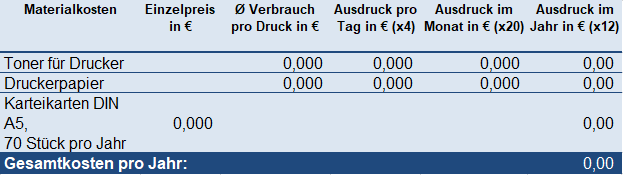
Material- und Mitarbeiterlohnkosten ***nach Einführung*** der Verwaltungssoftware

Tabelle 5 Materialkosten nach Einführung

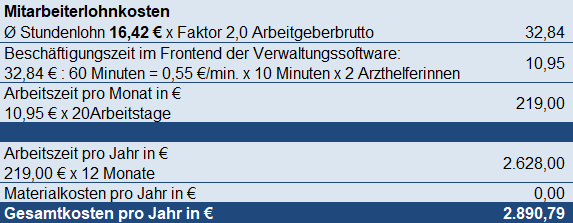


Tabelle 6 Mitarbeiterlohnkosten nach Einführung

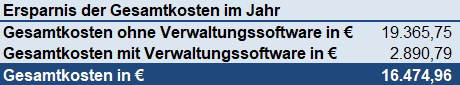


Tabelle 7 Ersparnis der Gesamtkosten im Jahr

Wie zu erkennen ist, ist die Investition in eine Verwaltungssoftware durchaus sinnvoll, da die bisherigen Materialkosten für Druckerpapier, Toner und Karteikarten wegfallen und zusätzlich die Mitarbeiterlohnkosten um 85,04% gesenkt werden.

Die Amortisationszeit beträgt (Berechnungsbetragt geteilt durch Ersparnis pro Jahr):

6 Monate 16 Tage

Die Amortisierung ist nach ca. 28 erreicht. Das ist der Zeitraum, in der die Verwaltungssoftware mindestens zum Einsatz kommen muss, damit sich das Projekt wirtschaftlich rechnet.

## Ablaufplanung

Bevor mit der Umsetzung des Projekts begonnen werden konnte, mussten sich der Autor für einen geeigneten Entwicklungsprozess entscheiden. Dieser definiert die Vorgehensweise, nach der die Umsetzung erfolgen soll. Da der Autor das Projekt bis auf das Code-Review alleine betreut, wird das Wasserfallmodel zur Projektsteuerung herangezogen.

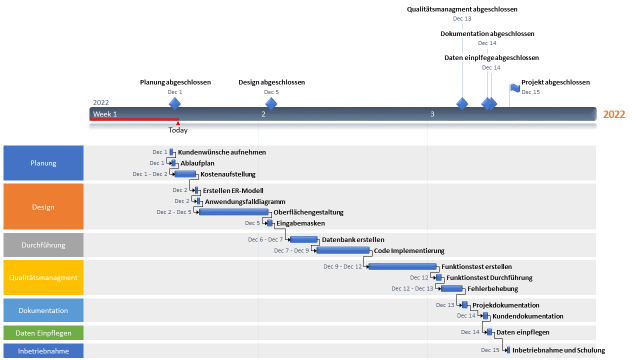
Die dabei verwendeten Modelle sind:

* Projektstrukturplan
* GANTT-Diagramm
* EPK

**Projektstrukturplan**



**GANTT-Diagramm**

**EPK-Modell**

Das EPK-Modell ist dem Anhang *A1 EPK-Modell* zu entnehmen

# Entwurfsphase

Das Kapitel *Entwurfsphase* beschäftigt sich mit der Planung, von der Oberfläche bis hin zur Businesslogik und der Datenbank.

## Zielplattform

Die Auswahl der Programmiersprache für das Projekt wurde zu Beginn der Planungsphase festgelegt. Da der Autor hauptsächlich in der Programmiersprache C# arbeitet und sich dort am sichersten fühlt, wurde nach diesen Kriterien ausgewählt.

Der Wunsch eine Desktop Applikation zu erhalten hat die Möglichkeiten in C# auf zwei Varianten beschränkt. Zum einen gäbe es die Windows Presentation Foundation (WPF), diese ist ein Framework zur Entwicklung von grafischen Benutzerschnittstellen für Desktop Applikationen. Da der Autor mit dieser keine Erfahrung hat, viel die Auswahl auf eine Windows Forms Anwendung. Bei der Auswahl des im Hintergrund laufenden Datenbanksystems gab es keine Vorgaben. Aus vorherigen Projekten hat sich der Einsatz einer lokalen Datenbank bewährt. Durch diese positiven Erfahrungen ist die Auswahl auf MariaDB gefallen. Mit dem Clint HeidiSQL können Verwaltungsaufgaben einfach durchgeführt werden. Um den Zugriff auf die Datenbank zu erleichtern, wird der OR-Mapper NPoco eingesetzt.

## Layoutdesign

Der Wunsch des Kunden war es ein einfach zu bedienendes Programm zu haben, das die wesentlichen Aspekte der Patientenverwaltung abdeckt. Hierfür wird eine Windows Desktop Oberfläche gestaltet die drei Forms aufweist und die notwendigen Buttons beinhaltet. Die Startseite mit Hintergrundbild der Praxis, von der aus alle Bereiche bequem erreicht werden können. Es gibt neben der Startseite zwei weitere Oberflächen, die dem Benutzer zur Verfügung stehen, Patient und Arzt. Lediglich der Bereich Sonstiges bleibt dem Admin vorbehalten. Dieser ist eine Arzthelferin, die eine separate Schulung erhalten wird. Das erste Layout wurde skizziert und mit dem Kunden besprochen.

Das erste Layout ist in der *Abbildung 3 Skizze Layout* zu sehen.

Ein Bild, das Text, Whiteboard enthält.

Automatisch generierte Beschreibung

Abbildung 3 Skizze Layout

## Anwendungsfälle

Um eine Übersicht aller Anwendungsfälle zu erhalten, welche in der Anwendung möglich sein sollen, wurde in der Entwurfsphase ein Anwendungsfalldiagramm (Use-Case-Diagramm) erstellt.

* Der Benutzer hat die Möglichkeit Patienten und Ärzte anzulegen, zu ändern, zu löschen und zu suchen.
* Der Admin hat zusätzlich die Möglichkeit neue Anreden, Diagnosen und Versicherungen anlegen.



Abbildung 4 Anwendungsfalldiagramm

## Relationale Datenbank

Der Datenbankentwurf wurde in Visio in einem ER-Modell logisch aufgebaut und dargestellt, zur bessern Ansicht und um im Anschluss die SQL-Datenbank einfacher zu implementieren.

Die Datenbank besteht im Wesentlichen aus einer Haupttabelle "Patient" und 6 Nebentabellen, um Redundanz zu vermeiden. Die Tabellen sind mit Beziehungen verbunden und bestehen aus Hauptschlüsseln, den Fremdschlüsseln und den Attributen.

Das unten dargestellteER-Modell zeigt lediglich einen Teilausschnitt. Das gesamte ER-Modell ist in Querformat dem Anhang *A2 ER-Modell* zu entnehmen.

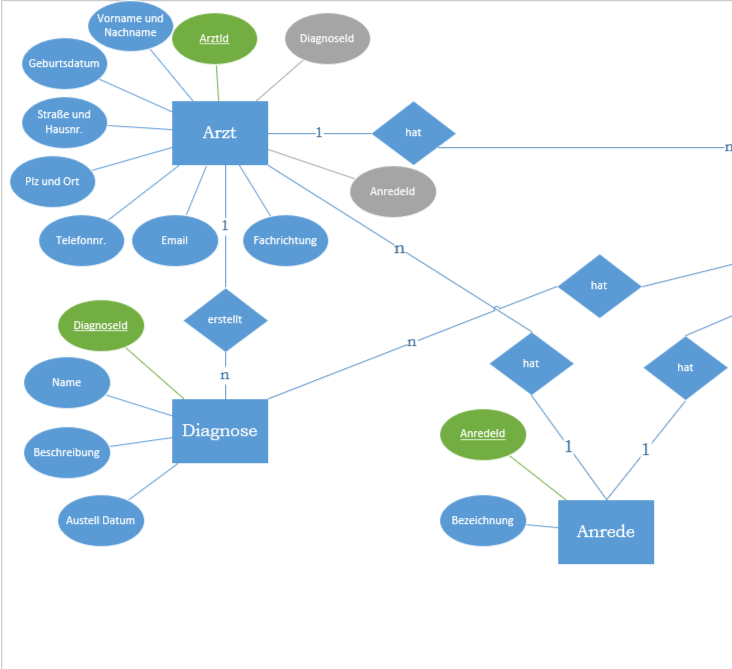


Abbildung 5 Teilausschnitt ER-Modell

## Relationales Datenmodell

Das Klassendiagramm wandelt das konzeptionelle ER-Modell in ein Tabellenmodell um und wurde ebenfalls in Visio aufgebaut. Es gibt einen Tabellarischen Überblick der einzelnen Typen und deren Attribute. Das Diagramm zeigt wie die einzelnen Tabellen in Beziehung miteinander stehen. Welcher Fremdschlüssel zu welchem Primärschlüssel gehört.

Das entsprechende Diagramm kann dem Anhang *A3 Relationales Datenmodell* entnommen werden*.*

# Implementierung

Die Eingabe wird über die Entwicklungsumgebung von Microsoft Visual Studio erarbeitet und mit der Programmiersprache C# eine Windows-Forms Oberfläche programmiert. Um die Daten abzulegen und zu speichern, wurde mithilfe von HeidiSQl und auf Basis von MariaDB eine Datenbank erstellt, die mit Visual Studio verbunden ist. Um die Datenbank für die Anwendung zu erstellen, wurde ein Skript mit dem Create Befehlen erstellt. Dieses wurde in HeidiSQL ausgeführt und somit die DB auf dem lokalen MAriaDB Server erstellt.

## Erstellen der Datenbank

In HeidiSQL wurde eine Datenbank angelegt, auf die über den Port 3306 mit dem dazugehörigen Benutzernamen und Passwort zugegriffen werden kann.

Hier wurden die Tabellen mit ihren Attributen, Primär. und Fremdschlüssel angelegt. Die Datentypen, die verwendet wurden, sind Int, Varchar und DateTime. Diese wurden bis auf DateTime in ihrer Länge, der entsprechend zu erwartenden Eingabelänge, begrenzt, um Speicherplatz zu sparen.

Der entsprechende Datenbank Code ist dem Anhang *A4 Datenbank Code* zu entnehmen.

## Verbindung der Datenbank mit Visual Studio

Damit die Daten in Zukunft von der Eingabe Oberfläche in der Datenbank landen, war es nötig diese miteinander zu verbinden. Hierfür wurden in C# MySQL.Data und NPoco installiert.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 6 Verbindungs Tools

Für das gesamte Projekt wird eine Connection verwendet. Diese wird mit dem ConnectionString initialisiert. Diese Verbindung wird immer auf- und abgebaut. Für die Verbindung wird folgendes Skript verwendet, der in einer Konfigurationsdatei angelegt ist.

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="mariadb"

connectionString="Server=localhost;Uid=root;Pwd=Start123;Database=patientenverwaltung;Port=3306"

providerName="MySql.Data.MySqlClient" />

</connectionStrings>

</configuration>

## Implementierung der Benutzeroberfläche

Die Implementierung der Benutzeroberfläche anhand des Designs aus Kapitel *3.1 Layoutdesign* wurde komplett in C# mit WindowsForms umgesetzt.

Die Startseite wurde mit einem Bild der Praxis gefüllt und am unteren Bildrand die Menüleiste eingefügt. Die *Abbildung 7* Startseite zeigt die Startseite in der finalen Version. Die weiteren Bereiche sind über die Buttons erreichbar.

Der Bereich Arzt wird nicht näher gezeigt, da sich dieser nur in wenigen Details von der Patientenseite unterscheidet.

Ein Bild, das Text, Gebäude, Himmel, draußen enthält.

Automatisch generierte Beschreibung

Abbildung 7 Startseite

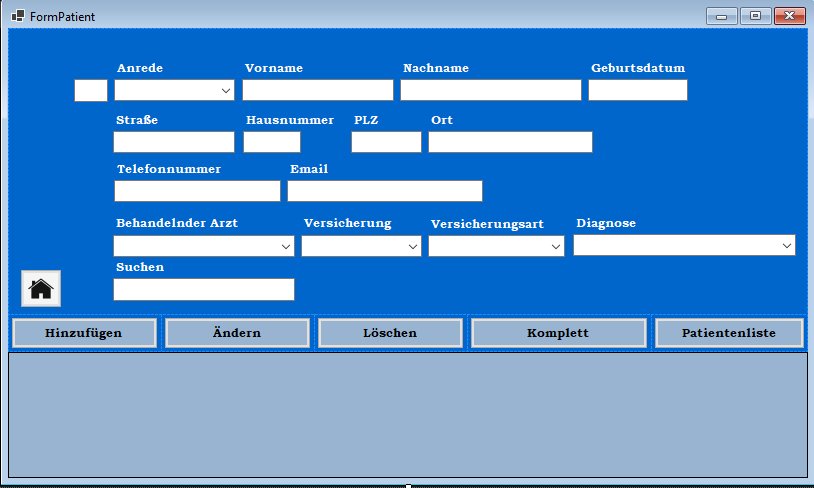


Abbildung 8 Patientenseite

Alle Seiten haben einen leicht zu findenden Homebutton spendiert bekommen, wie in der *Abbildung 8 Patientenseite* zu sehen ist, um darüber bequem zurück zur Startseite zu gelangen.

Der Bereich Sonstiges beinhaltet drei Kategorien, Versicherung Diagnose und Anrede, die hier mit einem tapControll zu bedienen sind.

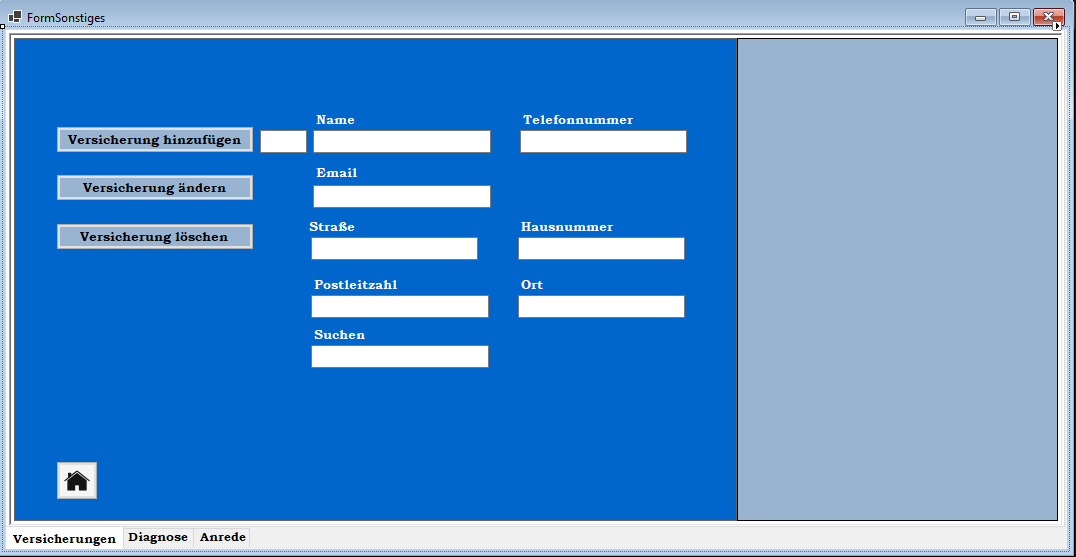


Abbildung 9 Sonstiges

## Implementierung der Features Alle Patienten und Suchen

Ein wesentlicher Bestandteil der Software soll es sein die Patienten mit ihrem behandelnden Arzt, deren Versicherung und Diagnose anzuzeigen. Hierfür wurde ein DataGridView genutzt. Da sich nicht alle Daten in der Patienten Klasse befinden, wurde eine PateientenKomplett Klasse erstellt, in der alle benötigten Tabellen eingefügt wurden.

Der nachfolgende Codeblock zeigt die Klasse PatientenKomplett

internal class PatientenKomplett

{

public Arzt? Arzt { get; set; }

public Anrede? Anrede { get; set; }

public Patient? Patient { get; set; }

public Versicherung? Versicherung { get; set; }

public Versicherungsart? Versicherungsart { get; set; }

public Diagnose? Diagnose { get; set; }

public override string ToString()

{

return $" {Nachname} {Vorname}";

}

}

Damit diese dann sich die benötigten Daten aus den einzelnen Tabellen holen und anzeigen kann, wurde noch ein SQL- Statement geschrieben und bei Bedarf dann aufgerufen.

string sql = "SELECT ar.\*, a.\*, p.\*, ver.\*, v.\*, di.\* FROM patient p " +

"JOIN anrede a ON a.AnredeId = p.AnredeId " +

"JOIN versicherung ver ON ver.VersicherungId = p.VersicherungId " +

"JOIN versicherungsart v ON v.VersicherungsartId = p.VersicherungsartId " +

"JOIN arzt ar ON ar.ArztId = p.ArztId " +

"JOIN diagnose di ON di.DiagnoseId = p.DiagnoseId " +

"ORDER BY p.Nachname, p.Vorname";

Das Ergebnis zeigt, dass das Statement funktioniert und alle gewünschten Daten auf einem DataGridView zusammenlaufen.

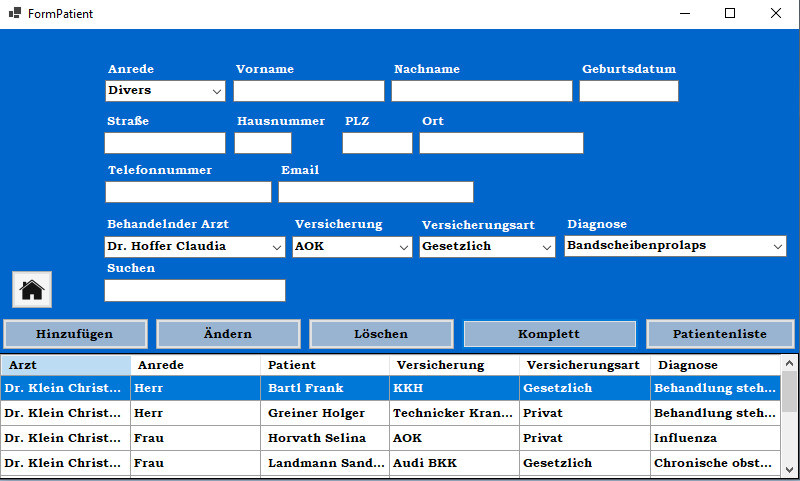


Abbildung 10 PatientenKomplett

Ein weiteres Feature ist es sich über die Suchfunktion einen bestimmten Datensatz anzeigen zu lassen. Hierfür wurde ein weiteres SQL-Statement geschrieben.

string input = txtSuchenPatient.Text + "%";

string sql = "SELECT ar.\*, a.\*, p.\*, ver.\*, v.\*, di.\* FROM patient p " +

"JOIN anrede a ON a.AnredeId = p.AnredeId " +

"JOIN versicherung ver ON ver.VersicherungId = p.VersicherungId " +

"JOIN versicherungsart v ON v.VersicherungsartId = p.VersicherungsartId " +

"JOIN arzt ar ON ar.ArztId = p.ArztId " +

"JOIN diagnose di ON di.DiagnoseId = p.DiagnoseId where p.Nachname like @0";

db.Connection.Open();

lstPatientenKomplett = db.Fetch<PatientenKomplett>(sql, input);

Dieses ermöglicht nach dem Anfangsbuchstaben des Nachnamens des gewünschten Patienten, Arzt, Diagnose oder Versicherung zu suchen. Bei Diagnose und Versicherung wurde nach dem Namen gesucht.

Das Ergebnis zeigt, das die Suchfunktion ihre Arbeit korrekt ausführt.

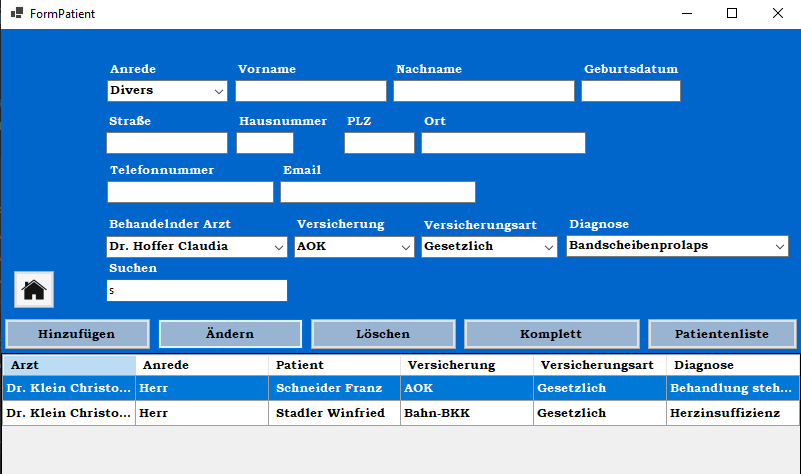


Abbildung 11 Suchfunktion von Patienten

# Fazit

Zu Beginn des Projektes stand ich vor der großen Herausforderung, wie ich das ganze umsetzten soll. Es waren sehr viel Informationen zu bearbeiten, zu sortieren und zu verstehen. Wie kann ich die Anforderung des Kunden umsetzten? Wie gehe ich am besten vor? Welche Features sind für eine reibungslose Funktion wichtig? Welche Features können später noch hinzugefügt werden.

Dank der guten Vorbereitung ist es mir gelungen ein funktionierendes Programm zu entwickeln und umzusetzen. Die einzelnen Phasen des Projekts waren allesamt sehr interessant und haben mein Wissen in jedem Bereich erweitert. Auch mit wenig IT-Kenntnissen, ist es mir dank einer strukturierten Arbeitsweise, gelungen mein Projekt erfolgreich abzuschließen. Natürlich ist nicht immer alles ohne Probleme gelaufen.

Ein Problem, das auch bis zum Schluss nicht gelöst wurde, ist, dass die UTF8 Kodierung, den Datennamen Straße nicht angenommen hat. Letzten Endes hat nur eine Umbenennung des Datennamen das Problem gelöst.

Zusammenfassend kann man sagen, dass das Projekt eine große Herausforderung war und mir meine Defiziente deutlich aufgezeigt hat. Dennoch habe ich einen Weg gefunden, alles zu einem erfolgreichen Ende zu bringen.

Dieses Projekt hat mich noch einmal darin bestärkt, dass die richtige Entscheidung war die Umschulung zum Fachinformatiker zu beginnen.

## Soll-/Ist-Vergleich

Die vom Kunden gewünschte Anforderungen wurde allesamt erfüllt. Die in den Funktionstests aufgefallenen Mängel, wurden allesamt beseitigt. Die Mitarbeiter des Kunden wurden geschult und das Programm wurde erfolgreich eingeführt. Der Kunde ist mit der Leistungserbringung vollends zufrieden und erwartet freudig neue Updates, die bereits besprochen und in Auftrag gegeben wurden. In den folgenden Tabellen 4 und 5 Soll-/Ist-Vergleich Zeiten und Kosten, werden die geplanten Kosten und die geplante Zeit, mit den tatsächlichen Daten verglichen und ausgewertet.

In der *Tabelle 4 Soll-/Ist-Vergleich Kosten* wird ersichtlich, dass die geplante Zeit deutlich unterschritten wurde und sich somit die Kosten reduziert haben, was den Kunden sehr gefreut hat.

|  |  |  |  |
| --- | --- | --- | --- |
| **Kosten** |  | geplant | tatsächlich |
| Stundenlohn |  | 95 € | 95 € |
| Arbeitszeit |  | 80 Std | 63,58 Std |
| Kosten o.MwSt. | | 7.600,00 € | 6.040,10 € |
| MwSt. 19% |  | 1.444,00 € | 1147,62 |
|  |  |  |  |
| Kosten mit MwSt. | | 9.044,00 € | 7.187,72 € |

Tabelle 8 Soll-/Ist-Vergleich Kosten

In der *Tabelle 5 Soll-/Ist-Vergleich Zeit* wird nun ersichtlich, in welchen Bereichen sich die tatsächliche Zeit von der geplanten abweicht.

Besonders in den Bereichen Code schreiben und Design konnte die Ziel Vorgabe deutlich unterschritten werden. Der größte Zeitfresser war das Schreiben der Dokumentation, die mit 16,5 Stunden zu Buche schlägt.

Tabelle 9 Soll-/Ist-Vergleich Zeit

## Lessons Learned

Im Laufe des Projekts wurde in allen Bereichen etwas dazu gelernt. Eine der wichtigsten Erkenntnisse war es, dass es beim Programmieren nicht die eine Lösung gibt. Auch der Umgang mit Visual Studio und HeidiSQL wurde gefestigt. Bei der Dokumentation der Zeiten ist aufgefallen, dass wenn man die Trennung der einzelnen Arbeitsschritte nicht sorgfältig trennt, kommt es bei der Zeitauswertung zu Problemen. Abschließend wird festgestellt, dass es noch viel zu lernen gibt.

## Ausblick

Obwohl schon einige Features umgesetzt worden sind, gibt es noch ein großes Entwicklungspotenzial. Zum einen, dass eine Auflistung der Diagnosen, die ein Patient erhalten hat, noch mit auf dessen Datensatz angezeigt wird. Das Ganze dann mit Datum der Ausstellung. Auch soll eine Kalenderfunktion Einzug erhalten, damit die Arzthelferin bei der Terminvergabe leicht einsehen kann, wann der zuständige Arzt am jeweiligen Tag Zeit hat.

# Anhang

## A1 EPK-Modell



## A2 ER-Modell



## A3 Relationales Datenmodell



## A4 Datenbank Code

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung