

**Abschlussprüfung Sommer 2023**

**Fachinformatiker für Anwendungsentwicklung  
Dokumentation zur betrieblichen Projektarbeit**



# **Datenbanken Management- und Modifikations-System mit GUI in C# und ASP.NET**

Abgabetermin: 26.08.2023

**Prüfungsbewerber:**

Benjamin Born  
Dreikreuzweg 67  
69151 Neckargemünd

**Ausbildungsbetrieb:**

SRH BBW Neckargemünd  
Im Spitzerfeld 25  
69151 Neckargemünd

# Inhaltsverzeichnis

<b>1. Einführung .....</b>	<b>3</b>
1.1 Projektumfeld .....	3
1.2 Auftraggeber.....	3
1.3 Zum Ausbildungsbetrieb und dem Auszubildenden .....	4
1.4 Kundengespräch über Anforderungen .....	4
1.5 Projektziel.....	5
1.6 Projektbegründung .....	5
1.7 Projektschnittstellen .....	5
<b>2. Projektplanung.....</b>	<b>5</b>
2.1 Zeitplan des Projekts .....	5
2.2 Abweichungen vom Projektantrag.....	7
2.3 Ressourcenplanung .....	7
2.4 Entwicklungsprozess.....	7
<b>3. Analysephase .....</b>	<b>8</b>
3.1 Ist-Analyse.....	8
3.2 Soll-Zustand .....	9
3.3 Wirtschaftlichkeitsanalyse .....	9
3.3.1 Make or Buy-Entscheidung .....	10
3.3.2 Projektkosten .....	10
3.3.3 Amortisationsdauer .....	10
3.4 Nutzwertanalyse.....	11
3.5 Anwendungsfälle .....	11
3.6 Qualitätsanforderungen .....	13
3.7 Lastenheft/Fachkonzept .....	14
3.7.1 Lastenheft.....	14
3.7.2 Pflichtenheft .....	14
<b>4. Entwurfsphase.....</b>	<b>15</b>
4.1 Zielplattform.....	15
4.2 Architekturdesign.....	15
4.3 Entwurf der Benutzeroberfläche .....	16
4.4 Geschäftslogik.....	17
4.5 Maßnahmen zur Qualitätssicherung.....	18
<b>5. Implementierungsphase .....</b>	<b>19</b>

5.1 Implementierung der Datenstrukturen .....	19
5.2 Implementierung der Benutzeroberfläche.....	20
5.3 Implementierung der Geschäftslogik .....	25
<b>6. Qualitätskontrolle .....</b>	<b>35</b>
6.1 Whitebox-Test .....	36
6.2 Blackbox-Test.....	36
<b>7. Abnahmephase .....</b>	<b>36</b>
<b>8. Einführungsphase .....</b>	<b>37</b>
<b>9. Fazit .....</b>	<b>37</b>
9.1 Soll-/Ist-Vergleich .....	37
9.2 Lessons Learned .....	38
9.3 Ausblick.....	38
<b>10. Kundendokumentation.....</b>	<b>38</b>
<b>11. Glossar .....</b>	<b>46</b>
11.1 Glossarquellen .....	49
<b>12. Literaturverzeichnis.....</b>	<b>50</b>
<b>13. Quellcode .....</b>	<b>51</b>
13.1 Index.cshtml.cs.....	51
13.2 Index.cshtml .....	76
13.3 Database.cs .....	91
13.4 Table.cs.....	92
13.5 _Layout.cshtml .....	93
13.6 site.css .....	95
13.7 Impressum.cshtml .....	96
13.8 Datenschutz.cshtml .....	97
13.9 IHK Abschlussprüfung Abnahmeprotokoll .....	106

## 1. Einführung

### 1.1 Projektumfeld

Das Unternehmen Schwenge ist im Metallbau und Gartenbau tätig und baut Überdachungen sowie Standard- und Spezialkonstruktionen, Ganzglasanlagen und Carports für Kunden. Sie haben ein komplexes Lagersystem, dass nicht mehr auf dem neusten Stand ist und ihnen Probleme verursacht.

Das Unternehmen Schwenge möchte deswegen ein eigenes Management/Modifikation System mit UI für ihre Geschäftsoperationen. Dies wird benötigt zur Optimierung und Anpassung und die Bedürfnisse des Unternehmens, sie möchten auch gerne, ihre Daten auf einem auf das Unternehmen angepasstes Programm selbst verwalten.

Das Programm wird möglicherweise in der Zukunft von dem Unternehmen ausgebaut, aber dies ist jetzt noch nicht sicher, jedoch ist es möglich, dass wir ein Folgeantrag von der Firma Schwenge bekommen zu der Erweiterung des Programmes.

Deswegen ist eine ausführliche Codekontrolle und Kommentierung des Codes nötig, dass später eventuell der Ausbau und die Wartung des Programmes keine längere Wiedereinarbeitungszeit benötigt.

Für die Entwicklung des Programmes soll Razor-Pages in C# als Framework benutzt werden. Die Darstellung der GUI ist in HTML geplant.

### 1.2 Auftraggeber



Die Firma Schwenge ist seit 1965 im Gartenbau tätig. Sie fing an mit allgemeinem Garten- und Landschaftsbau.

Das Geschäft war gefragt und beliebt, deswegen hat Gartenbau Schwenge dann ab 1980 noch den Bereich Metallbau in ihr Unternehmen mit eingegliedert.

Seitdem bieten sie sowohl Gartenbau und Gartenpflege, als auch Metallbau und die Konstruktion von Gartenhäusern und anderen Sachen wie zum Beispiel Zäunen und Gartentüren an. Das Unternehmen ist relativ bekannt in dem Umkreis und hat eine treue Kundschaft, die sich vor allem aus Gartenliebhabern und Hobbygärtnern zusammensetzt.

### 1.3 Zum Ausbildungsbetrieb und dem Auszubildenden

Das Softwareprojekt wird von einem Auszubildenden in der SRH GmbH Neckargemünd bearbeitet. Zum Ausbildungsbetrieb gibt es auf deren Webseite eine kurze Vorstellung:

„Das Berufsbildungswerk Neckargemünd GmbH (BBWN) bei Heidelberg bildet junge Menschen mit Handicap außerbetrieblich aus. Neben unterschiedlichen berufsvorbereitenden Maßnahmen werden mehr als 40 Berufe für einen erfolgreichen Start in die Arbeitswelt angeboten. Die Bildungsangebote und Methoden werden laufend den Entwicklungen der Wirtschaft angepasst, sodass eine hohe Ausbildungsqualität auf aktuellem Stand gewährleistet ist.

Im BBWN stehen Auszubildenden im Rahmen der beruflichen Rehabilitation und Jugendhilfe umfangreiche medizinische, therapeutische sowie pflegerische Fachdienste zur Verfügung. Das BBWN kooperiert mit zahlreichen Unternehmen der Metropolregion Rhein-Neckar sowie nationalen und internationalen Unternehmen und bereiten sie systematisch auf ihren Berufseinstieg vor.“

Der Auszubildende wurde gewählt wegen seiner guten Noten und motivierten Arbeitsweise, die er zeigte. Er hat bereits Erfahrung in Asp.Net Razor-Pages und ist damit der perfekte Kandidat um das Projekt umzusetzen. Auch hat er in der letzten Zeit in einem Praktikum viel mit Datenbankmanipulation gearbeitet und ist deswegen allgemein von den Vorkenntnissen für dieses Projekt gut geeignet.

### 1.4 Kundengespräch über Anforderungen

In einem Gespräch mit dem Kunden wurden die gewünschten Features und andere Spezifikationen festgelegt. Unter anderem möchte der Kunde eine Datenbank selber erstellen mit dem Programm. Auch soll der Import von bestehenden MySQL Datenbanken möglich sein. Die Erstellung von Tabellen in der Datenbank sowie die Änderung von bestehenden Tabellen ist gewünscht.

Falls diese Basis-Funktionen fertig sind und es wie nach Zeitplan geplant läuft, möchte der Kunde auch noch folgende Funktionen. Das Filtern in der Datenbank nach Tabellen oder Schlüsselworten sowie das Suchen von Tabellen in mehreren Datenbanken. Auch ist das Löschen von Datenbanken oder einzelnen Tabellen in der Datenbank erwünscht.

## 1.5 Projektziel

Das Ziel des Projektes ist es, die Anforderung des Kunden in Umfang des Zeitplans umzusetzen und eine Dokumentation dafür zu erstellen. Eine Softwaredokumentation als auch eine Kundendokumentation.

Das Unternehmen Schwenge möchte ein eigenes Management/Modifikation System mit UI für ihre Geschäftsoperationen. Dies soll zur Optimierung des Arbeitsprozesses dienen und Arbeitszeit in diesem Bereich einsparen, die dann sinnvoll anderweitig verwendet werden kann.

Auch soll das Programm von dem Kunden selber erweitert werden können oder potenziell wird es einen Folgeauftrag zur Erweiterung geben.

## 1.6 Projektbegründung

Das Programm spart in der Theorie sehr viel Zeit, da das alte Programm Probleme verursacht und der Arbeitsverlauf nicht mehr effizient ist. Es wird sehr viel Zeit damit verbracht, die Fehler des alten Programms manuell in der Datenbank zu beheben, da die Daten sonst unbrauchbar sind.

Mit einem neuen Programm, das nicht mehr manuell korrigiert werden muss und effizient funktioniert, wird eine deutliche Arbeitslasterleichterung erreicht. Über eine längere Zeit werden sich die Kosten des Anschaffens des neuen Programms auch amortisieren.

## 1.7 Projektschnittstellen

Die Anbindung der MySQL Datenbank, die bisher noch nicht erstellt wurde. Sie wird später zum Test selber generiert und ultimativ gegebenenfalls vom Programm erstellt. Die Datenbank wird zur Laufzeit in ein virtuelles DataSet geladen und dort werden dann die Änderungen gesammelt und beim Speichern übermittelt.

Auch gibt es die Verbindung zu der Web GUI, die Datenbankendaten anzeigt und Werkzeuge für den User bereitstellt, um diese zu verändern. Diese wird in einer Kombination aus HTML und CSS dargestellt.

# 2. Projektplanung

## 2.1 Zeitplan des Projekts

Das Projekt findet im Zeitraum von 80 Arbeitsstunden statt. Der Zeitraum ist von 10.07.23 – 21.07.23. Diese werden in einem Betriebsraum zu Tageszeit abgewickelt. Die Arbeitszeiten sind von 8 Uhr bis 17 Uhr, mit täglichen Pausenzeiten von einer Stunde.

## Detaillierte Zeitplanung

## Benötigte Zeit in Stunden

### Projektplanung:

Kundengespräche	5
Ist- und Soll-Analyse	3
Anforderungsanalyse	4
SQL-Abfrage Forschung	6
OOP Planung und Struktur	8
GUI Struktur Planung	2
<b>Zwischensumme:</b>	<b><u>28</u></b>

### Kernfunktionen Implementierung:

Datenbank-Schnittstelle (WPF, Razor-Pages)	2
Erstellung neuer Datenbanken	2
Import bestehender Datenbanken	1
Erstellung von neuen Tabellen	3
Änderung von Tabellen (Attribute, Beziehungen)	3
Schnittstellenverbindung zum Backend	3
GUI-Design	3
Testen und Polieren der Funktionen:	7
<b>Zwischensumme:</b>	<b><u>24</u></b>

### Dokumentation:

Einführung	3
Planungsphase	5
Analysephase	8
Entwurfsphase	6
Implementationsphase	6
<b>Zwischensumme:</b>	<b><u>28</u></b>
	<b><u>Insgesamt: 80 Stunden</u></b>

## 2.2 Abweichungen vom Projektantrag

Derzeit sind keine Abweichungen bekannt.

## 2.3 Ressourcenplanung

Es werden folgende Ressourcen für die Bearbeitung des Auftrages benötigt:

### Hardware

- Schreibtisch
- Monitor
- Computer
- Keyboard
- Maus

### Software

- Windows 10
- Word
- Umllet
- Microsoft Visual Studio Lizenz
- MySQL Datenbank
- XAMPP
- Snipping Tool

### Personal

- Softwareentwickler

## 2.4 Entwicklungsprozess

Der Entwicklungsprozess, der bei der Bearbeitung des Projektes befolgt wird, ist die Wasserfallmethode. Die Organisation ist in aufeinanderfolgende Projektphasen organisiert und wie bei einem Wasserfall wird das Projekt von einer Phase in die nächste bearbeitet, nachdem die vorherige Phase fertiggestellt wurde.

Diese Methode ist für dieses Projekt optimal, da nur eine Person dieses bearbeitet. Die simple Struktur macht den Aufwand diese Methode zu benutzen geringer und somit wird nicht so viel Zeit in diesen Teil gesteckt.

Auch ist das Modell gut für eine hohe Planungssicherheit. Durch die geordnete Struktur können auch komplexere Projekte gezielt geplant und zuverlässig durchgeführt werden.



## 3. Analysephase

### 3.1 Ist-Analyse

Zu Beginn wird eine Ist-Soll-Analyse durchgeführt, um die genauen Rahmenbedingungen des Projekts festzulegen und im Nachhinein einen Plan für die noch benötigte Struktur der Software zu erstellen.

Bei einer Ist-Analyse werden die Geschäftsprozesse ausführlich analysiert, mit dem Ziel diese eventuell zu optimieren und bereits bestehende Prozesse, die später unter anderem nicht mehr erstellt werden müssen. Zu einer Ist-Analyse gehört unter anderem die Befragung des Unternehmens und Ihrer Mitarbeiter sowie auch ein Interview mit dem Geschäftsführer über die bestehenden Geschäftsprozesse.

Durch eine Befragung der Mitarbeiter, sowie einem Gespräch mit dem internen IT-Angestellten, wurde folgender Ist-Zustand ermittelt.

Die Features, die der Kunde gewünscht hat, wurden bereits im Kundengespräch in [1.4 Kundengespräch über Anforderungen](#) angesprochen.

Folgende Punkte des Projekts sind vom Auftraggeber bereits erfüllt:

#### Hardware

- Schreibtisch
- Monitor
- Computer
- Keyboard
- Maus

#### Software

- Windows 10
- Word
- XAMPP
- Snipping Tool

(Dies ist ein Ausschnitt aus der Ressourcenplanung)

## 3.2 Soll-Zustand

Folgende Features sollen unterstützt werden:

- **Erstellung einer Datenbank**  
Die Datenbank wird als Datei erstellt und ist nun bereit, Daten aufzunehmen.
- **Import von bestehenden Datenbanken**  
Es wurde hier das Datenbankmanagementsystem MySQL ausgewählt. Es gibt keine Pläne, andere Formate zu unterstützen.
- **Die Erstellung von Tabellen in Datenbanken**  
Eine Tabelle ist eine Sammlung von Daten, die mehrere Spalte und Reihen umfasst. Die Spalten zeigen die Art von Informationen, die in den folgenden Reihen unter dieser Spalte angezeigt werden. Zum Beispiel, falls die Spalte Wohnort heißt, werden dort in den folgenden Reihen Wohnorte festgelegt. Alle Spalten einer Reihe zusammen ergeben einen Datensatz. Ein Datensatz ist eine Gruppierung von zusammengehörenden Daten.
- **Änderung von bestehenden Tabellen (Attribute, Beziehungen)**  
Die Datensätze der Tabelle oder die Anzahl der Spalten kann geändert werden. Hier können auch die Beziehungen der Tabelle zu einer anderen Tabelle festgelegt werden. Zum Beispiel die Tabelle Auto hat einen Fremdschlüssel in Besitzer. Das heißt, ein Auto hat einen Besitzer.
- **Filtern von Datenbanken und Suchen von Tabellen in Datenbanken**  
Das Filtern auf bestimmte Schlüsselwerte. Zum Beispiel sucht man das Auto Porsche in der Tabelle. Falls nun Datensätze der Tabelle diesen beinhaltet, wird diese Reihe der Tabelle als Suchergebnis ausgegeben. Das Filtern ist in drei verschiedenen Versionen möglich:
  - Filtern nach Tabellennamen
  - Filtern nach Spaltennamen
  - Filtern nach allen Zeilen und Spalten in der Datenbank
- **Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken**  
Das Löschen der Datenbank oder einzelnen Tabellen der Datenbank aus dem Programm.
- **Schutz von systemrelevanten Datenbanken**  
Dem Benutzer soll es nicht möglich sein, im Programm die Systemrelevanten MySQL Datenbanken zu modifizieren.

## 3.3 Wirtschaftlichkeitsanalyse

Das Programm spart in der Theorie sehr viel Arbeitszeit ein, die dann woanders benutzt werden kann. Durch das neue Programm, das nicht mehr manuell korrigiert werden muss und effizienter funktioniert, wird eine deutliche Arbeitslasterleichterung erreicht. Über eine längere Zeit werden sich die Kosten des Anschaffens des neuen Programms amortisieren, dazu kommen wir aber später nochmal genauer.

Dadurch werden jeden Tag Arbeitsstunden gespart und wenn man einen bestimmten Satz pro Arbeitsstunde und die gesparte Zeit miteinander multipliziert, kann man sich ausrechnen, wie viel Geld man zum Beispiel in einem Monat spart.

### 3.3.1 Make or Buy-Entscheidung

Es gibt Alternativen eines Datenbanken Management- und Modifikation-Systeme wie zum Beispiel PhpMyAdmin, aber dieses kann nicht nach spezifischen Anforderungen beliebig geändert werden, um die Wünsche des Kunden zufriedenzustellen, sondern ist ein Produkt, das nicht für einen speziellen Zweck entwickelt wurde.

Deswegen lohnt es sich für das Unternehmen Schwenge ein Datenbanken Management- und Modifikationssystem speziell für ihren Betrieb und ihre Arbeitsabläufe entwickeln zu lassen, denn dann bekommt das Unternehmen ein maßgeschneidertes Programm, das perfekt zu den Wünschen von diesem passt.

### 3.3.2 Projektkosten

Vorgang	Zeit	Kosten / Stunde	Berechnung	Kosten
Entwicklung	76 h	80,00 €	$80€ \cdot 76h$	6.080,00 €
Fachgespräch	3 h	80,00 €	$80€ \cdot 3h$	240,00 €
Abnahme	1 h	80,00 €	$80€ \cdot 1h$	80,00 €
Visual Studio Lizenz	1	20,00 €	$1 \times 20,00€$	20,00 €
			Gesamt	<u>6.420,00 €</u>
			Gesamt mit Mehrwertsteuer (19%)	<u>7.639,80 €</u>

### 3.3.3 Amortisationsdauer

Wir nehmen an, dass das Unternehmen Schwenge pro Monat etwa 20 Stunden damit verbringt, die alte Software zu bedienen und dessen Fehler manuell in der Datenbank zu korrigieren. Das sind umgerechnet etwa 1 Stunde pro Tag.

Zur Übersicht zur Berechnung ist hier die Formel:

$$\frac{\text{Personalaufwand in Stunden}}{\text{Wochen / Werktage}} = \text{Stunden pro Tag Personalaufwand}$$

Anwendung der Formel:

$$\frac{20 \text{ Stunden}}{4 \text{ Wochen} \cdot 5 \text{ Werktage}} = 1 \text{ Stunde}$$

Die Kosten des Mitarbeiters, der dafür zuständig ist, sind 40 Euro die Stunde.

Nun können wir die Amortisationsdauer ausrechnen mit der folgenden Formel:

$$\frac{\text{Produktpreise}}{\text{Kosten der Firma pro Tag}} = \text{Amortisationsdauer in Tagen}$$

Anwendung der Formel:

$$\frac{7639,80\text{€}}{40\text{€}} \approx 191 \text{ Werktage}$$

Umrechnung in Wochen:

$$\frac{191 \text{ Werktage}}{5 \text{ Werktage}} \approx 39 \text{ Werkwochen}$$

Man kann daraus herleiten, dass die Software etwa 39 Werkwochen, jeweils mit 5 Werktagen braucht, um sich zu amortisieren.

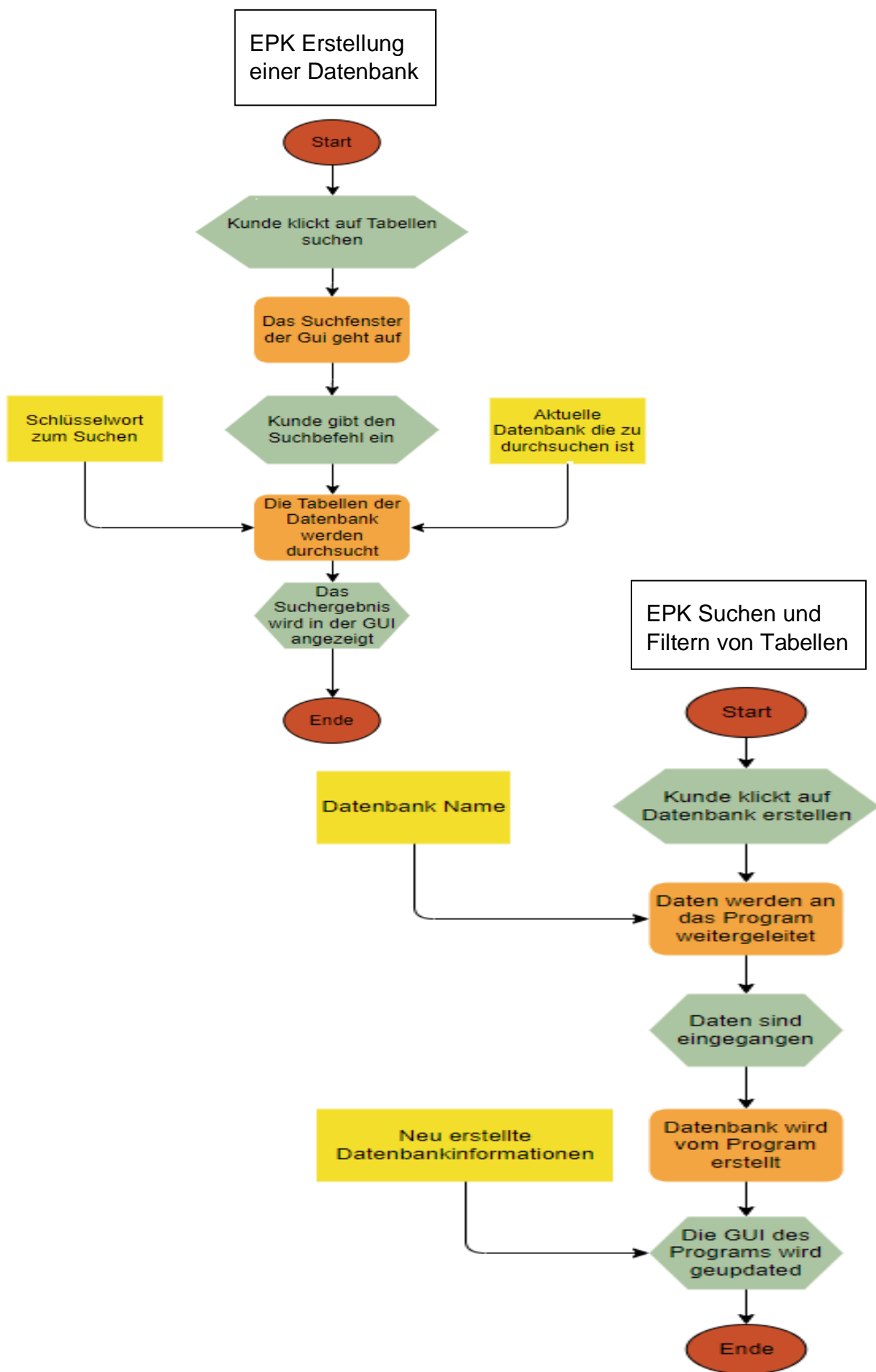
### 3.4 Nutzwertanalyse

Das geplante Programm wird die Anwendungszeit deutlich verringern. Die Fehler des alten Programms müssen nicht mehr manuell in der Datenbank behoben werden, sondern die Datenbank ist auf Anhieb richtig und kann direkt weiterverwendet werden. Deswegen nimmt man, um das gleiche Ergebnis zu erreichen, weniger Zeit in Anspruch. Die Zeit, die man in diesem Geschäftsprozess einspart, kann sinnvoll an einem anderen Platz eingesetzt werden. Somit erreicht man ein im Durchschnitt effektiveres Unternehmen. Durch ein effektiveres Unternehmen können die Kosten reduziert werden und somit steigen die Gewinne auch indirekt an.

### 3.5 Anwendungsfälle

Das Programm soll folgende Anwendungsfälle abdecken:

- Erstellung einer Datenbank
- Import von bestehenden Datenbanken
- Die Erstellung von Tabellen in Datenbanken
- Änderung von bestehenden Tabellen (Attribute, Beziehungen)
- Filtern von Datenbanken
- Suchen von Tabellen in mehreren Datenbanken
- Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken
- Schutz von systemrelevanten Datenbanken



## 3.6 Qualitätsanforderungen

Das Softwareprodukt wird sich an den Richtlinien der ISO 25-010 orientieren und sich an diese halten.

Es gibt 8 Qualitätsmerkmale und diese lauten wie folgt:

1. **Funktionalität:**  
Die gewünschte Funktionalität ist gegeben und die Softwarefunktionen sind vollständig.
2. **Effizienz/Performance:**  
Ist das System so programmiert, dass es effizient läuft und unter anderem zum Beispiel keinen Memory Leak hat.
3. **Kompatibilität:**  
Wie kompatibel ist das System auf anderen Systemen auf anderer oder gleicher Hardware.
4. **Benutzbarkeit:**  
Wie gut erreichen die Anwender ihre Ziele mit der Software und wie einfach ist das System von den Nutzern zu bedienen.
5. **Zuverlässigkeit:**  
Die Zuverlässigkeit der Leistung des Programmes und ob die Software ausgereift ist.
6. **Sicherheit:**  
Wie sicher sind die Daten der Nutzer und ob das Programm vor Manipulation geschützt ist.
7. **Wartbarkeit:**  
Wie anpassungsfähig das System ist im Blick auf Fehlerkorrektur oder Veränderbarkeit der Software.
8. **Portabilität:**  
Kann man die Software leicht installieren und ob man die Software von einer Umgebung in das andere übertragen kann.

Dazu werden bei der Qualitätskontrolle mehrere Schritte eingeleitet, um diese Richtlinien zu garantieren.

Es wird ein Blackbox-Test sowie auch ein Whitebox-Test durchgeführt. Dadurch werden die Qualität, Effizienz und die Fehlerfreiheit des Programmes garantiert. Diese Tests werden später in [6. Qualitätskontrolle](#) nochmal genau erläutert und durchgeführt.

## 3.7 Lastenheft/Fachkonzept

### 3.7.1 Lastenheft

Es ist ein individuell angepasstes Datenbank Management/Modifikation System mit Web UI für die Nutzung in Geschäftsoperationen zu erstellen. Dies soll zur Optimierung des Arbeitsprozesses dienen und Arbeitszeit in diesem Bereich einsparen, die dann sinnvoll anderweitig verwendet werden kann.

Die Grundfunktionen des Programmes sind die Erstellung einer Datenbank, der Import von bestehenden Datenbanken, die Erstellung von Tabellen in Datenbanken, die Änderung von bestehenden Tabellen (Attribute, Beziehungen). MySQL soll als Datenbankmanagementsystem für das Programm verwendet werden.

In dem Webfenster sollen die Anwender durch GUI Elemente das Programm bedienen können.

Eine ausführliche Erklärung und Auflistung der einzelnen Funktionen finden Sie in [3.2 Soll-Zustand](#).

### 3.7.2 Pflichtenheft

Es wird eine Anwendung für Windows 10 erstellt, diese sollte auch kompatibel mit Windows 7, 8 und 11 sein. Eine Unterstützung für andere Betriebssystem ist im Moment nicht geplant.

Folgende Funktionen werden in der Anwendung enthalten sein:

- Erstellung einer Datenbank
- Import von bestehenden Datenbanken
- Die Erstellung von Tabellen in Datenbanken
- Änderung von bestehenden Tabellen (Attribute, Beziehungen)
- Filtern von Datenbanken
- Suchen von Tabellen in mehreren Datenbanken
- Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken
- Schutz von systemrelevanten Datenbanken

Für die Entwicklung des Programmes soll Razor-Pages in C# als Framework benutzt werden. Die Darstellung der GUI ist in HTML und CSS geplant. Für die Entwicklung wird Microsoft Visual Studio 2022 verwendet.

Als Datenbankmanagementsystem wird MySQL benutzt, um die Daten in diesem zu speichern, nachdem sie geändert wurden. Eine Unterstützung von anderen Datenbankmanagementsystemen ist derzeit nicht geplant.

## 4. Entwurfsphase

### 4.1 Zielplattform

Die Zielplattform des Softwareproduktes ist Windows 10. Hier wird mit Microsoft Visual Studio 2022 entwickelt. Also Programmiersprache wurde C# gewählt. Zum Design der Webseite wird eine Kombination aus HTML, CSS und Javascript verwendet. Die Schnittpunkte der Datenbank und der Web-GUI werden dann in Asp.Net Razorpages zusammengeführt und mit der Funktionalität des Softwareproduktes verbunden. Das Programm ist lokal und hat keine Netzwerkanbindung. Es kann nur von einem Nutzer auf einmal benutzt werden. Als Hardware werden nur ein Computer, ein Bildschirm, eine Maus und eine Tastatur benötigt.

### 4.2 Architekturdesign

Das Programm soll objektorientiert strukturiert sein. Es ist geplant, dies in dem Razor-Pages Framework zu entwickeln. Für die grafische Darstellung soll HTML in Kombination mit CSS benutzt werden. Die MySQL Datenbanken durch das Framework eingebunden.

Kurze Einführung in das verwendete Framework „Asp.Net Razorpages“:

Asp.Net ist eine leichte und schnelle Technologie, die zum Erstellen und Designen von dynamischen Webseiten und deren Inhalt verwendet wird. Hierbei wird eine Kombination von Frontend-Technologien (z.B. HTML 5, CSS 3 & JavaScript) und Backend-Technologien (ASP.NET Web Pages, Razor Engine) verwendet. Auch ist es möglich, dass Internetseiten direkt auf Datenbanken zugreifen können und man den meisten funktionellen Code von Javascript in das Backend, das mit C# geschrieben wird, verlagern kann.

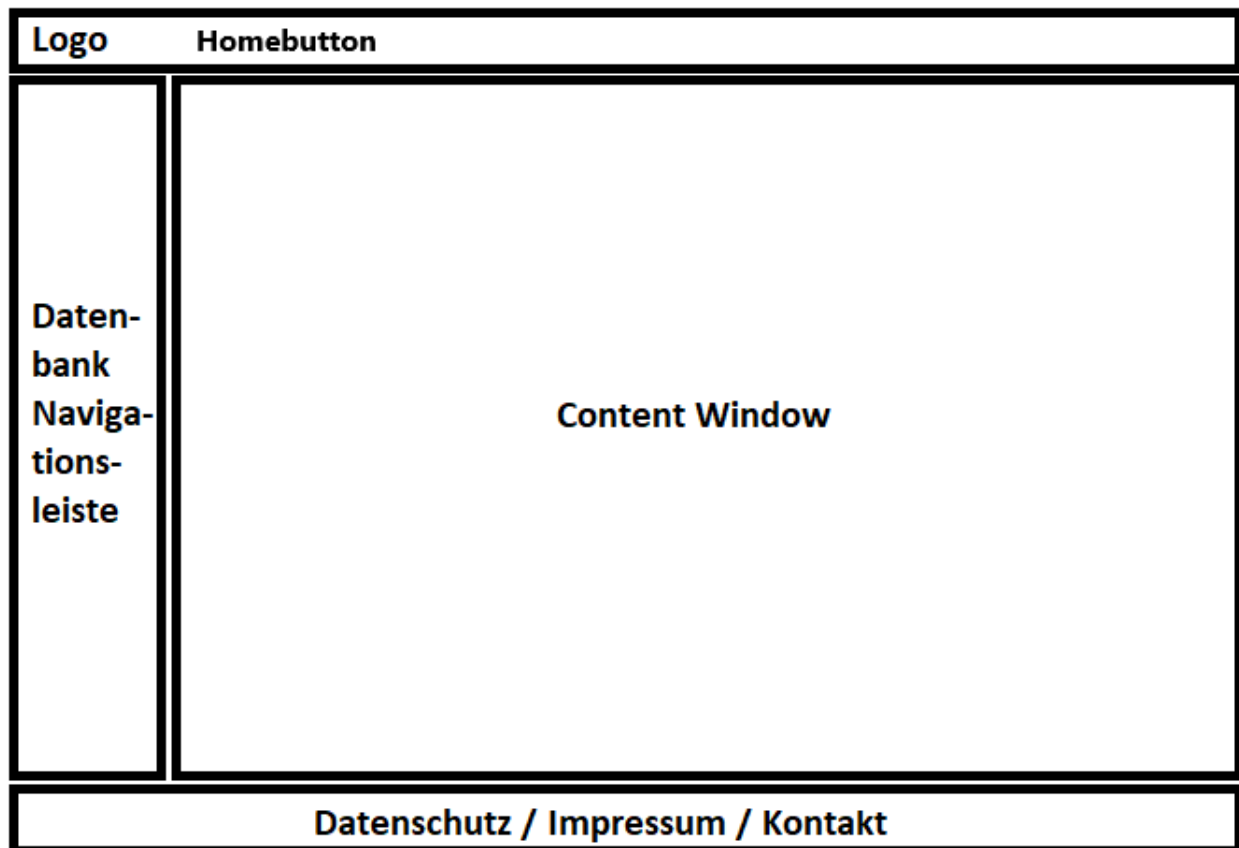
Nun kommen wir eben noch zu dem Razor Syntax. Mithilfe von diesem können serverseitige Code-Snippets in HTML Code eingebettet werden. Diese Fragmente können in C# oder in Visual Basic vorhanden sein. Im Softwareprodukt wird jedoch nur C# verwendet. Jetzt parst die Razor Engine den Code und führt den Inhalt aus.



### 4.3 Entwurf der Benutzeroberfläche

Als Benutzeroberfläche wurde eine Web GUI gewählt, die man in HTML, CSS und Javascript zusammensetzt. Es soll eine Navigationsleiste ganz oben geben, mit dem Knopf der zurück auf die Homepage führt.

Nach dem Gespräch mit dem Kunden, in dem unter anderem auch das Design der GUI besprochen wurde. Nach einigen Versuchen wurde folgender Skizze entwickelt.



#### Web-GUI Skizze:

In der oberen Einstellungsleiste ist ganz links Platz für das Logo der Firma. Rechts werden dann verschiedene Einstellungen stehen, wie zum Beispiel das Öffnen der Datenbank oder das Abspeichern der Änderungen.

Mittig bis Mittig-Rechts ist das Content-Window. Es wird benutzt, um viele verschiedene Seiten darzustellen und die Tabellen sowie die Datenbankdaten darzustellen.

Mittig-Links ist die Datenbanknavigationsleiste, diese wird verwendet um den Datenbankenbaum und die Tabellen darzustellen.

Unten in der Fußleiste sollen die Datenschutzrichtlinien, das Impressum und die Kontaktdaten platziert werden.

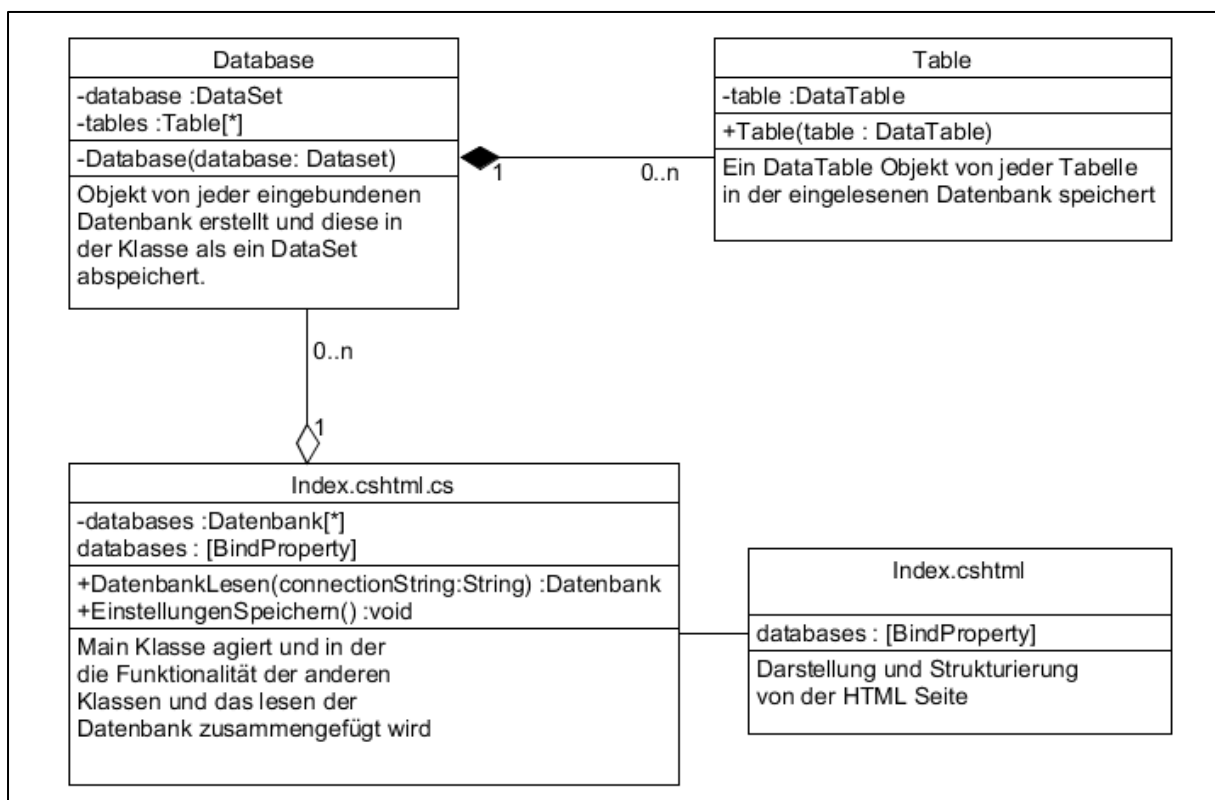
## 4.4 Geschäftslogik

Es ist geplant, die Logik der Software in mehrere Klassen aufzuteilen.

- Eine ‚Index.cshtml‘, die hauptsächlich zur Darstellung und Strukturierung von der HTML-Seite zuständig ist und von bestimmten Variablen eine [Bindproperty] hat, um diese dynamisch anzuzeigen.
- Die ‚Index.cshtml.cs‘, die als Main Klasse agiert und in der die Funktionalität der anderen Klassen und das Lesen der Datenbank zusammengefügt wird.
- Zusätzlich gibt es eine ‚Database‘ Klasse, die ein Objekt von jeder eingebundenen Datenbank erstellt und diese in der Klasse als ein DataSet abspeichert. Auch hat diese eine Liste von Tabellen Objekten je nachdem wie viele Tabellen die Datenbank hat.
- Die ‚Table‘ Klasse, die jeweils ein DataTable Objekt von jeder Tabelle in der eingelesenen Datenbank speichert.

Also hat ein Datenbankobjekt, mehrere Tabellen Objekte.

Für eine genauere und finale Erklärung der Geschäftslogik bitte schauen Sie in [5.3 Implementierung der Geschäftslogik](#).



Entwurfs-UML der benutzten Klassen

## 4.5 Maßnahmen zur Qualitätssicherung

Es werden eine Reihe von Tests durchgeführt, um die Qualität des Softwareproduktes zu garantieren und dem Anwender eine angenehme Bedienung zu verschaffen.

In den folgenden Punkten werden die verschiedenen Maßnahmen, die zur Qualitätssicherung unternommen werden, nochmal detailliert erklärt.

Sie werden dann später unter dem Punkt [6. Qualitätskontrolle](#) durchgeführt.

- **Blackbox-Test**

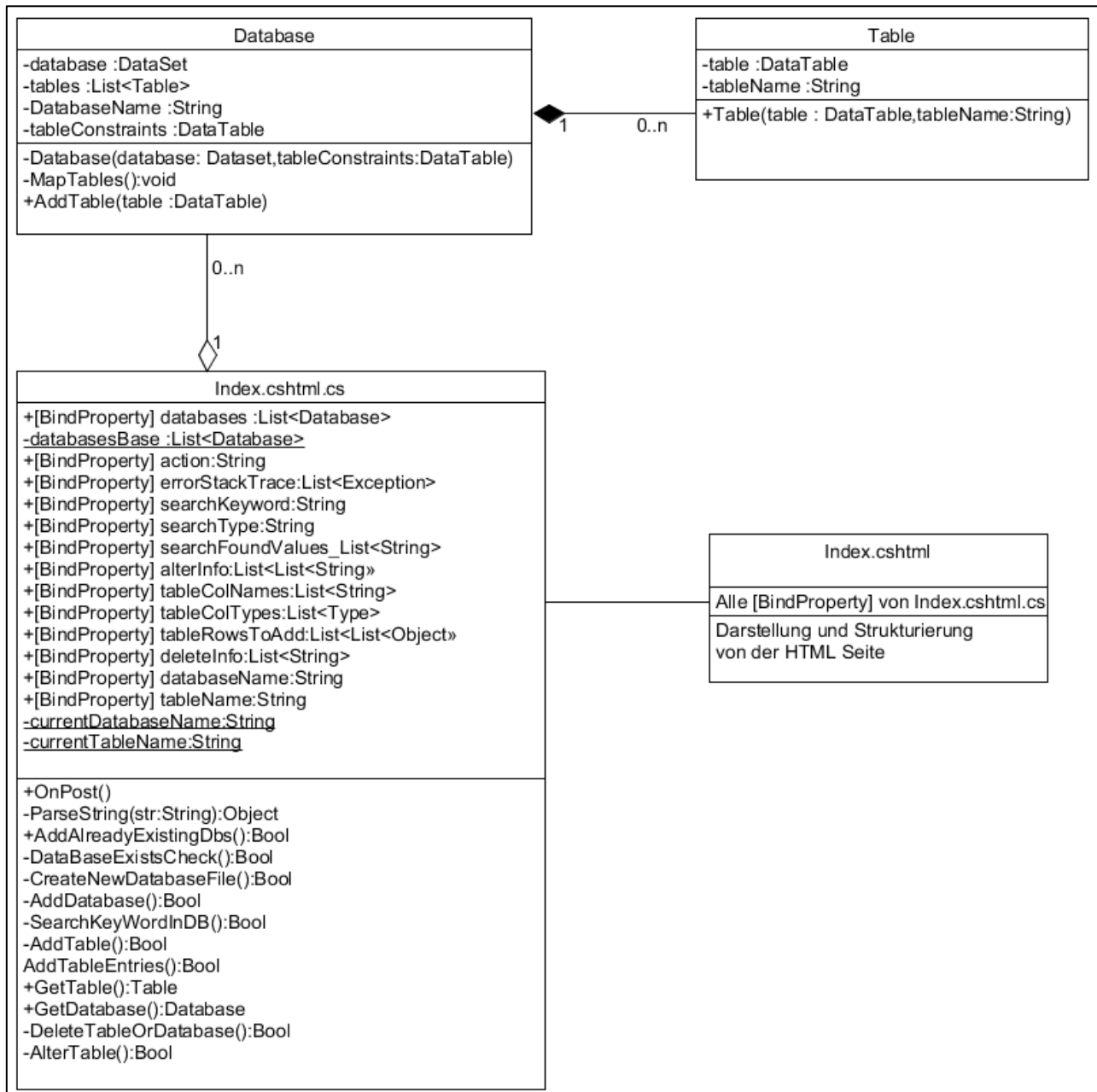
Hier wird eine Person, die nicht mit der Software vertraut ist zum Testen benutzt, um mögliche Fehler bei der Laufzeit zu finden.

- **Whitebox-Test**

Hier testen Personen, die mit der Software vertraut sind, das Programm und versuchen mögliche Fehler in der Funktion bei Laufzeit zu finden

## 5. Implementierungsphase

### 5.1 Implementierung der Datenstrukturen



#### Erklärung des UML, dass oben abgebildet ist:

Die Datenbanken werden erst mit einem „Select \* Tabellename“ für jede Tabelle mit dem SqlDataAdapter in DataTable gefüllt. Nun werden diese zu einem DataSet zusammengefügt und als ein Datenbank-Objekt im Programm gespeichert. Das Datenbank-Objekt erstellt sich für jede Tabelle nochmal ein Table-Objekt in dem die Daten für die Tabellen der Datenbank gespeichert werden.

Zur Anzeige der Datenbanken im Programm werden alle Table-Objekte mit einer Schleife durchgelaufen und deren ‚TableName‘ Attribut in einer einspaltigen Tabelle angezeigt.

Zur Anzeige der Tabellen wird die DataTable des Tabellen-Objekts mit einer Schleife für jede Reihe und eine Schleife in dieser Schleife für jede Spalte jeweils Zelle für Zelle in einer Tabelle angezeigt.

#### **Index.cshtml:**

Hier werden die Informationen, die das Programm aus der Datenbank zieht und bearbeitet, mit Hilfe von [Bindproperty] angezeigt. Für Tabellen und Spalten, sowie deren Informationen werden Tabellen zur Anzeige gewählt.

#### **Index.cshtml.cs:**

Ist die Hauptklasse des Programms und ist für die meiste Funktionalität zuständig.


Hier werden alle Funktionen des Programms in Methoden ausgeführt und es werden die [Bindproperty] Attribute festgelegt. Die in dieser Klasse verarbeiteten Informationen werden in Index.cshtml dann zur Darstellung für den Benutzer verwendet.

In der [5.3 Geschäftslogik](#) werden die Funktionen nochmal in Detail erklärt.

## **5.2 Implementierung der Benutzeroberfläche**



Die Startseite hat Links, eine Seitenleiste, wo man auf die bereits bestehenden Datenbanken zugreifen kann. Falls noch keine bestehen wird nur der ‚Neue Datenbank‘ erstellen Knopf sowie eine Meldung, dass es noch keine bestehenden Datenbanken gibt, angezeigt.



Startseite

## Importierte Datenbanken

testdbwithvalues1

ansprechpartner

lieferanten

testdbwithvalues2

artikel

filiale

verkauf

testdbwithvalues3

auftraege

hanswurst

kunden

modelle


Neue Datenbank erstellen

## Neue Datenbank erstellen

Datenbankname:

Submit

Hier sieht man die Oberfläche zum Erstellen einer neuen Datenbank. Es gibt ein Textfeld, um den Namen der neuen Datenbank einzugeben und nach einem Klick auf ‚Submit‘ wird die Datenbank erstellt.



Startseite

## Importierte Datenbanken

testdbwithvalues1

ansprechpartner

lieferanten

testdbwithvalues2

artikel

filiale

verkauf

testdbwithvalues3

auftraege

hanswurst

kunden

modelle

Neue Datenbank erstellen

## Datenbank: testdbwithvalues1

### Tabellen:

ansprechpartner

lieferanten

Neue Tabelle erstellen

Datenbank Filtern: Suchart: Table

Suchtext:

Datenbank filtern


Datenbank löschen

Relationen der Tabellen ändern

Oben sieht man die Anzeige, die man bekommt, nachdem man in der Seitenleiste auf eine Datenbank klickt. Oben stehen der Datenbankname und darunter die Auflistung der Tabellen. Nun kommen wir zu den Funktionen, die auf dieser Seite sind. Es ist möglich, eine neue Tabelle zu erstellen. Es gibt jetzt eine Weiterleitung zu einer Seite, bei der man die Daten für die neue Tabelle eingeben soll. Die Benutzeroberfläche dazu erkläre ich weiter unten. Man kann die Datenbank nach Schlüsselwörtern durchsuchen.

Hier kann man nun nach Tabellennamen, nach Spaltennamen oder jede einzelne Zelle von allen Tabellen durchsuchen. Mit dem Klick auf Datenbank filtern wird man auf eine neue Seite mit den Suchergebnissen weitergeleitet.

Es ist möglich, die Datenbank zu löschen. Nach dem Klick auf Datenbank löschen wird man nochmal gefragt, ob man sich sicher ist, ob man diese löschen will.


Startseite

## Importierte Datenbanken

**testdbwithvalues1**  

ansprechpartner

lieferanten

**testdbwithvalues2**  

artikel

filiale

verkauf

**testdbwithvalues3**  

auftraege

hanswurst

kunden


modelle

Neue Datenbank erstellen

Folgende Ergebnisse wurden bei der Suche nach "a" gefunden:

Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner		
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	AID	
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	Name	
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	Vorname	
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	E-Mail	
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	Vorname	Reihe:1
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	Vorname	Reihe:2
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	E-Mail	Reihe:2
Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner	Vorname	Reihe:3

Hier sieht man die Benutzeroberfläche, die zur Anzeige der Suchergebnisse, die man auf der Datenbankseite gesucht hat, dient. Diese werden wie im Tabellenkopf dargestellt angezeigt, falls sie diese Attribute haben. Ein Tabellenname Suchergebnis hat zum Beispiel kein Spaltenname oder eine Reihenposition.


Startseite

## Importierte Datenbanken

**testdbwithvalues1**  

ansprechpartner

lieferanten

**testdbwithvalues2**  

artikel

filiale

verkauf

**testdbwithvalues3**  

auftraege

hanswurst

kunden

modelle

Neue Datenbank erstellen

## Neue Tabelle erstellen

Tabellenname:

Spalte 1:

Spaltenname:  Spaltentyp:

Spalte 2:

Spaltenname:  Spaltentyp:

Spalte 3:

Spaltenname:  Spaltentyp:

Spalte 4:

Spaltenname:  Spaltentyp:

Spalte 5:

Spaltenname:  Spaltentyp:

Submit

Hier sieht man das Fenster zum Erstellen von einer neuen Tabelle, das nach einem Klick auf ‚Neue Tabelle erstellen‘ auf dem Datenbankfenster angezeigt wird.

Importierte  
Datenbanken

## testdbwithvalues1

[ansprechpartner](#)  
[lieferanten](#)

## testdbwithvalues2

[artikel](#)  
[filiale](#)  
[verkauf](#)

## testdbwithvalues3

[auftraege](#)  
[hanswurst](#)  
[kunden](#)  
[modelle](#)[Neue Datenbank erstellen](#)

## Tabelle: artikel


[PRK]ArtikelID	Artikelname	Verkaufspreis	Einkaufspreis
100	Salat de Luxe	1.50	0.70
101	Hamburger TS Royal	4.80	2.10
102	Steakburger	5.10	2.25
103	Cheeseburger	1.80	1.15
104	McSundea Schoko	1.00	0.01
105	Cheeseburger	2.00	1.00
106	Hamburger Royal TS	3.00	2.00
107	McFlurry	2.00	1.00
108	Signature Burger Beff and Egg	7.00	4.00
109	Hamburger Original	1.50	0.80
110	Chicken Nuggets	2.00	1.00
111	BicMac	5.00	3.00
113	MCBurger	1.00	0.50
115	Tasty Chicken	3.00	1.00
116	Chicken MC-Kai	4.00	3.00
117	Big Rösti Chicken	3.00	9.00

[Neue Reihe hinzufügen](#)[Tabellenattribute ändern](#)[Tabelle löschen](#)

Oben sieht man die Tabellenseite, auf die man verlinkt wird, nachdem man entweder auf der Seitenleiste oder im Datenbankfenster auf eine Tabelle klickt. Auf der Seite werden alle Spalten und die Daten dieser in einer Tabelle angezeigt.

Auch gibt es unter der Anzeige folgende Funktionen: Das Hinzufügen einer neuen Reihe von Daten in die Tabelle. Mehr dazu weiter unten. Das Ändern der Tabellenattribute: Hier kann man die Attribute der Spalten in der Tabelle ändern oder eine neue Spalte in der Tabelle hinzufügen. Auch kann man die Relationen der Tabelle ändern.




Startseite

### Importierte Datenbanken

testdbwithvalues1

ansprechpartner

lieferanten

testdbwithvalues2

artikel

filiale

verkauf

testdbwithvalues3

auftraege

hanswurst

kunden

modelle

Neue Datenbank erstellen

## Neue Tabellenreihe hinzufügen

[PRK]ArtikelID	Artikelname	Verkaufspreis	Einkaufspreis
Int32	String	Decimal	Decimal


Tabellenreihe erstellen

Auf dieser Seite kann man eine neue Reihe von Daten in die Tabelle einfügen.

Die Tabellenheader zeigen die Namen der Spalten an. [PRK] bedeutet die Reihe ist ein Primärschlüssel und [FRK] bedeutet die Reihe ist ein Fremdschlüssel.

Die zweite Reihe zeigt den Datentyp der Spalte nochmal zu Information mit an, damit man das Format der Daten weiß, die man eingibt.

Die letzte Reihe besteht aus Textfeldern, in die man die Daten der neuen Tabellenreihe eingibt.


Startseite

### Importierte Datenbanken

testdbwithvalues1

ansprechpartner

lieferanten

testdbwithvalues2

artikel

filiale

verkauf

testdbwithvalues3

auftraege

hanswurst

kunden

modelle

Neue Datenbank erstellen

## Spalten der Tabelle filiale ändern

[PRK]FilialeID	sadasd
Int32	String
Neue Attribute:	
Name	Name
Datentyp	Datentyp

Submit

## Neue Spalte in der Tabelle filiale erstellen

Name: 
Datentyp:

Submit

(Sie Können nur eine der beiden Aktionen auf einmal aufführen)

In diesem Fenster kann man die Attribute der Spalten der Tabelle ändern. Es ist möglich, den Namen und den Datentyp zu ändern. Über neue Attribute stehen nochmal die alten Attribute zur Übersicht. Darunter kann man mit einem Textfeld den neuen Namen eingeben und mit der Auswahlliste kann man den neuen Datentyp auswählen. Außerdem kann man darunter noch eine neue Spalte in die Tabelle

einfügen, dazu hat man ein Textfeld für den Namen und eine Auswahlliste für den Datentyp.  
Auch ist vermerkt, dass man nur eine der beiden Aktionen auf einmal ausführen kann.

Zu guter Letzt kommen wir noch zu dem Ändern der Relationen der Tabellen in der Datenbank.

Zur Übersicht sind nochmal alle Tabellen und deren Spalten aufgelistet.

Danach kann man mit zwei Auswahllisten auswählen, welche Spalten man als Primär- und Fremdschlüssel wählen möchte. Ein Klick auf ‚Primär- und Fremdschlüssel Verbindung hinzufügen‘ führt nun die Aktion aus.

### 5.3 Implementierung der Geschäftslogik

Es werden folgende Punkte der Geschäftslogik nochmal genauer erklärt mithilfe von Bildern des Programmcodes:

- [Erstellung eines Datenbank-Objekts](#)
- [Erstellung einer neuen Datenbank](#)
- [Import von bestehenden Datenbanken und Schutz von systemrelevanten Datenbanken](#)
- [Die Erstellung von Tabellen in Datenbanken](#)
- [Änderung von bestehenden Tabellen \(Attribute, Beziehungen\)](#)
- [Filtern von Datenbanken und Suchen von Tabellen in mehreren Datenbanken](#)
- [Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken](#)
- [Einbindung der Datasets in Datenbank Objekte und der Tabellen in Table Objekten](#)
- [Bindproperty und Form Benutzung und Aufbau der Webseite](#)

## Erstellung eines Datenbank-Objekts:

```
using(MySqlConnection connectionDB = new MySqlConnection(connectionString)) {
    connectionDB.Open();
    //Datenbankname
    database.DataSetName = connectionDB.Database;
    //Tabellennamen mit Sql-Abfrage abrufen
    MySqlCommand command = new MySqlCommand("Show tables;", connectionDB);
    MySqlDataAdapter adapter = new MySqlDataAdapter(command);
    adapter.Fill(tableNames);
    if(tableNames == null) {
        throw new Exception("There was an issue with the tablenames when importing the " + connectionDB.Database + " Database");
    }
    //Abrufen von Beziehungen aus Tabellen in der Datenbank mit Sql-Befehl (Quelle ist im Dokument)
    command = new MySqlCommand("SELECT\n\r" +
        "    'TABLE_NAME',                                -- Foreign key table\n\r" +
        "    'COLUMN_NAME',                                -- Foreign key column\n\r" +
        "    'REFERENCED_TABLE_NAME',                        -- Origin key table\n\r" +
        "    'REFERENCED_COLUMN_NAME'                        -- Origin key column\n\r" +
        "FROM\n\r" +
        "    'INFORMATION_SCHEMA'. 'KEY_COLUMN_USAGE' -- Will fail if user don't have privilege\n\r" +
        "WHERE\n\r" +
        "$" 'CONSTRAINT_SCHEMA' = '{connectionDB.Database}'\n\r"
        , connectionDB);
    adapter = new MySqlDataAdapter(command);
    adapter.Fill(tableConstraints);
    // Holt jede Tabelle mit Select * aus der Datenbank und speichern Sie sie alle in einem Dataset.
    foreach(DataRow rowZero in tableNames.Rows) {
        String tableName = rowZero.ItemArray[0].ToString();
        string query = "SELECT * FROM " + tableName.ToString();
        command = new MySqlCommand(query, connectionDB);
        adapter.SelectCommand = command;
        DataTable dataTable = new DataTable(tableName.ToString());
        adapter.Fill(dataTable);
        database.Tables.Add(dataTable);
    }
    // Datenbankobjekt wird mit dem Datensatz erstellt und zur Liste der Datenbanken hinzugefügt
    Database db = new Database(database, tableConstraints);
    databases.Add(db);
}
//Bei Erfolg wird true zurückgegeben
return true;
```

Hier werden Daten mit einem MySqlAdapter aus der Datenbank geholt und in einer DataTable gespeichert.

Die Relationen der Tabellen in der Datenbank als DataTable gespeichert.

Erst werden alle Tabellennamen mit einem ‚Show Table‘ Befehl aus der Datenbank geholt. Dann wird durch die Tabellennamen mit einer Schleife gegangen und es werden die einzelnen Tabellen mit ‚Select \* from tableName‘ aus der Datenbank geholt und in ein DataSet gespeichert. Mit diesem DataSet wird dann das Datenbank-Objekt erstellt.

### Erstellung einer neuen Datenbank:

```
private Boolean CreateNewDatabaseFile() {
    try {
        string connectionString = $"server=localhost;uid=root;pwd=";
        string databaseName = this.databaseName;
        using(MySqlConnection connection = new MySqlConnection(connectionString)) {
            connection.Open();
            MySqlCommand command = new MySqlCommand($"CREATE DATABASE {databaseName};", connection);
            command.ExecuteNonQuery();
            this.connectionString = connectionString + "database=" + databaseName + ";";
        }
        return true;
    }
}
```

Hier wird eine neue Datenbank mit dem vom Benutzer eingegebenen Namen erstellt. Es wird eine SQL Verbindung gemacht und mit dem 'Create Database Befehl' eine Datenbank erstellt.

### Import von bestehenden Datenbanken und Schutz von systemrelevanten Datenbanken:

```
using(MySqlConnection connection = new MySqlConnection(connectionString)) {
    MySqlDataAdapter adapter = new MySqlDataAdapter("SHOW DATABASES;", connection);
    DataTable table = new DataTable();
    adapter.Fill(table);
    String[] systemDbNames = { "information_schema", "mysql", "performance_schema", "phpmyadmin" };
    foreach(DataRow row in table.Rows) {
        if(!systemDbNames.Contains(row[table.Columns[0]].ToString())) {
            this.connectionString = $"server = localhost; user = root; password = ; database = {row[table.Columns[0]]}; ";
            databaseName = row[table.Columns[0]].ToString();
            Boolean dbExists = false;
            foreach(Database db in databases) {
                if(databaseName == db.databaseName) {
                    dbExists = true;
                }
            }
            if(dbExists) {
                continue;
            }
            AddDatabase();
        }
    }
}
return true;
```

Hier werden mit dem 'Show Databases' Befehl alle Datenbanken, die zurzeit in MySQL bestehen geholt. Dann werden die systemrelevanten Datenbanken zum Schutz nicht als Database-Objekt importiert und alle anderen Datenbanken werden als Objekt in das Programm geladen.

## Die Erstellung von Tabellen in Datenbanken:

```
foreach(Database database in databases) {
    if(database.databaseName == databaseName) {
        //Sobald die passende Db gefunden wurde, fügen wir eine Tabelle hinzu
        DataTable table = new DataTable(tableName);
        for(int i = 0; i < tableColNames.Count; i++) {
            table.Columns.Add(tableColNames[i], tableColTypes[i]);
        }
        database.AddTable(table);
        database.dataSet.AcceptChanges();
        erg = true;
        using(MySqlConnection connectionDB = new MySqlConnection(connectionString)) {
            connectionDB.Open();
            String query = $"CREATE TABLE {tableName} (";
            for(int i = 0; i < tableColNames.Count; i++) {
                query += $"`{tableColNames[i]}` {tableColTypes[i].Name},";
            }
            query = query.TrimEnd(',');
            query += ")";
            query = query.Replace("String", "VarChar(25)").Replace("Int32", "Int").Replace("Double", "Decimal");
            MySqlCommand command = new MySqlCommand(query, connectionDB);
            command.ExecuteNonQuery();
        }
    }
}
return erg;
```

Erst wird mit einer Schleife durch alle Datenbanken gegangen, um das Datenbank-Objekt zu finden, in dem man die Tabelle hinzufügen soll. Nun wird mit einer Schleife durch alle Tabellenspalten, die der Benutzer eingegeben hat, gegangen, um diese als Spalten im Objekt hinzuzufügen. Jetzt wird eine SQL Verbindung aufgemacht, um die Änderungen in die Datenbank zu schreiben.

## Änderung von bestehenden Tabellen (Attribute, Beziehungen) :

```
using(MySqlConnection connection = new MySqlConnection(connectionString)) {
    connection.Open();
    //Columnname ändern
    foreach(List<String> list in alterInfo) {
        String query = $"ALTER TABLE {tableName} ";
        if(list[0] == "rename") {
            query += $"CHANGE `{list[1]}` `{list[2]}` {list[3]}";
            query = query.Replace("String", "VarChar(25)").Replace("Int32", "Int").Replace("Double", "Decimal");
        }
        else if(list[0] == "relations") {
            query = $"ALTER TABLE {list[1]} ";
            query += $"ADD CONSTRAINT 'PK_{list[1]}' PRIMARY KEY({list[2]}); ";
            query += $"ALTER TABLE {list[3]} ";
            query += $"ADD CONSTRAINT 'FK_{list[3]}' FOREIGN KEY({list[4]}) REFERENCES `{list[1]}`({list[2]}); ";
        }
        //Columntyp ändern
        else if(list[0] == "modify") {
            query += $"CHANGE COLUMN `{list[3]}` `{list[3]}` {list[2]}";
            query = query.Replace("String", "VarChar(25)").Replace("Int32", "Int").Replace("Double", "Decimal");
        }
        //Column hinzufügen
        else if(list[0] == "add") {
            query += $"ADD `{list[1]}` {list[2]}";
            query = query.Replace("String", "VarChar(25)").Replace("Int32", "Int").Replace("Double", "Decimal");
        }
        MySqlCommand command = new MySqlCommand(query, connection);
        command.ExecuteNonQuery();
        Thread.Sleep(5);
    }
}
databases.Clear();
databasesBase.Clear();
AddAlreadyExistingDbs();
return true;
```

Hier werden die Spaltenattribute der Tabellen geändert, je nachdem welche Funktion der Benutzer auswählt. Es ist möglich, den Tabellennamen zu ändern, die Beziehung der Tabellen in der Datenbank zu ändern, die Datentypen der Spalten zu ändern und eine Spalte in der Tabelle hinzuzufügen.

## Filtern von Datenbanken und Suchen von Tabellen in mehreren Datenbanken:

```
foreach(Database database in databases) {
    if(database.databaseName == databaseName) {
        foreach(Table table in database.tables) {
            if(searchType == "table" || searchType == "data") {
                if(table.table.TableName.ToLower().Contains(searchKeyword.ToLower())) {
                    erg = true;
                    searchFoundValues.Add(database.databaseName + " " + table.table.TableName);
                }
            }
            if(searchType == "columns" || searchType == "data") {
                foreach(DataColumn column in table.table.Columns) {
                    if(column.ColumnName.ToLower().Contains(searchKeyword.ToLower())) {
                        erg = true;
                        searchFoundValues.Add(database.databaseName + " " + table.table.TableName + " " + column.ColumnName);
                    }
                }
            }
            if(searchType == "data") {
                foreach(DataRow row in table.table.Rows) {
                    foreach(DataColumn column in table.table.Columns) {
                        if(row[column].ToString().ToLower().Contains(searchKeyword.ToLower())) {
                            erg = true;
                            searchFoundValues.Add(database.databaseName + " " + table.table.TableName + " " + column.ColumnName + " " + "Reihe:" + rowPosition);
                        }
                    }
                    rowPosition++;
                }
            }
        }
    }
}
return erg;
```

Erstmal werden alle Datenbank-Objekte durchgegangen, um zu finden, mit welcher Datenbank man arbeiten soll. Nach dem Finden der richtigen Datenbank wird durch jede Tabelle der Datenbank mit einer Schleife gegangen. Hier wird nun geschaut, welcher Typ von Suche denn ausgewählt wurde. Bei einer Daten-Suche werden Tabellennamen, Spaltennamen und Zellen nach dem Suchbegriff durchsucht.

Bei Spaltennamen und Tabellennamen suchen werden jeweils nur die Spaltennamen oder die Tabellennamen durchsucht.

Die Ergebnisse werden in einem Array gespeichert, der später zur Anzeige verwendet wird.

## Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken:

```
using(MySqlConnection connection = new MySqlConnection(connectionString)) {
    connection.Open();
    if(deleteInfo[0].ToLower() == "table") {
        Database database = GetDatabase();
        Table table = GetTable();
        if(table != null) {
            database.dataSet.Tables.Remove(table.table);
            database.tables.Remove(table);
            database.dataSet.AcceptChanges();
            MySqlCommand command = new MySqlCommand($"DROP TABLE {tableName};", connection);
            command.ExecuteNonQuery();
        }
        else {
            throw new Exception($"Die Tabelle {tableName} von der Datenbank {databaseName} konnte nicht gelöscht werden, da sie nicht gefunden wurde.");
        }
    }
    else if(deleteInfo[0].ToLower() == "database") {
        Database database = GetDatabase();
        if(database != null) {
            databases?.Remove(database);
            database.dataSet.Clear();
            database.dataSet = null;
            database = null;
            MySqlCommand command = new MySqlCommand($"DROP DATABASE {databaseName};", connection);
            command.ExecuteNonQuery();
        }
        else {
            throw new Exception($"Die Datenbank {databaseName} konnte nicht gelöscht werden, da sie nicht gefunden wurde.");
        }
    }
}
return true;
```

Erst wird geschaut, ob eine Tabelle oder Datenbank gelöscht werden soll. Nun prüft das Programm, ob die Tabelle oder Datenbank existiert und falls ja, wird diese erst aus dem Datenbank- oder Tabellen-Objekt gelöscht. Danach wird mit einem ‚Drop Table/Database‘ Befehl die Tabelle aus der SQL Datenbank gelöscht.

## Einbindung der Datasets in Datenbank Objekte und der Tabellen in Table Objekten:

```
// Datenbankobjekt wird mit dem Datensatz erstellt und zur Liste der Datenbanken hinzugefügt
Database db = new Database(database, tableConstraints);
databases.Add(db);
```

*Ausschnitt aus der AddDatabase Methode*

Hier sieht man die Einbindung des Datensets und der Relationen-DataTable als Database Objekt im Programm.

Diese wird dann zu der Liste der Datenbanken, die im Programm geladen sind hinzugefügt.

## Database Klasse:

```
16 references
public class Database {
    //Original DataSet
    8 references
    public DataSet dataSet { get; set; }
    //List of Table objects
    15 references
    public List<Table> tables { get; set; }
    //Name of the database
    22 references
    public string databaseName{get;set;}
    //Table of the relationships in the Database between the different tables
    5 references
    public DataTable tableConstraints { get; set; }
    1 reference
    public Database(DataSet database, DataTable tableConstraints) {
        this.tableConstraints = tableConstraints;
        this.dataSet = database;
        this.databaseName = database.DataSetName;
        if(tables == null) {
            tables = new List<Table>();
        }
        MapTables();
    }
    /// <summary>
    /// Adding new table objects from the dataset after reading the database
    /// </summary>
    1 reference
    private void MapTables() {
        try {
            foreach(DataTable table in dataSet.Tables) {
                tables.Add(new Table(table,table.TableName));
            }
        } catch(Exception) {
            throw;
        }
    }
    /// <summary>
    /// Add table as object and the table to the original dataset
    /// </summary>
    /// <param name="table"></param>
    /// <returns></returns>
    1 reference
    public Boolean AddTable(DataTable table) {
        tables.Add(new Table(table,table.TableName));
        dataSet.Tables.Add(table);
        return true;
    }
}
```

In der Database Klasse wird das DataSet, die TableConstraints und der Datenbankname abgespeichert und nun wird die MapTables Methode aufgerufen, die für jede DataTable in dem DataSet ein Table-Objekt erstellt. Dies wird dann in die Tabellen von Typ List<Table> von dem Database-Objekt hinzugefügt.



## Table Klasse:

```
15 references
public class Table {
    39 references
    public DataTable table { get; set; }
    21 references
    public string tableName { get; set; }
    2 references
    public Table(DataTable table, string tableName) {
        this.table = table;
        this.tableName = tableName;
    }
}
```

In der Table-Klasse werden nur die Tabelle als DataTable und der Tabellename als String abgespeichert.

## Bindproperty und Form Benutzung und Aufbau der Webseite:

```
//List of Database, die alle Datenbanken enthält, die zur Laufzeit in das Programm importiert wurden
21 references
[BindProperty] public List<Database>? databases { get; set; }
//Die Statische Liste der Datenbanken
private static List<Database>? databasesBase = new List<Database>();
//Die Art der Aktion, die das Programm ausführt, wenn der Benutzer eine Aktion ausführt
39 references
[BindProperty] public String? action { get; set; }
//Der ConnectionString zur aktuellen Datenbank
public String connectionString = new String("");
//Die Fehlerliste, die verwendet wird, um Fehler für den Benutzer auszugeben, die er später sehen kann
18 references
[BindProperty] public List<Exception>? errorStackTrace { get; set; }
//Das Schlüsselwort, das der Benutzer zum Suchen oder Filtern je nach Aktion eingibt
6 references
[BindProperty] public String? searchKeyword { get; set; }
//Die Art der Suche, die der Benutzer durchführen möchte (Tabellennamen, Spaltennamen oder alle Daten)
6 references
[BindProperty] public String? searchType { get; set; }
//Die Werte dessen, was bei der Suche gefunden wurde und wo es gefunden wurde
5 references
[BindProperty] public List<String>? searchFoundValues { get; set; }
//Informationen darüber, was in einer Tabelle geändert werden soll
7 references
[BindProperty] public List<List<String>>? alterInfo { get; set; }
// Die Spaltennamen der neuen Tabelle
6 references
[BindProperty] public List<String>? tableColNames { get; set; }
//Der Wertetyp der neuen Tabellenspalten
6 references
[BindProperty] public List<Type>? tableColTypes { get; set; }
//Zeilen, die der aktuellen Tabelle hinzugefügt werden
3 references
[BindProperty] public List<List<Object>>? tableRowsToAdd { get; set; }
//Informationen darüber, welche Art von Löschung durchgeführt werden muss
4 references
[BindProperty] public List<String>? deleteInfo { get; set; }
//Diese 2 Eigenschaften unten ändern sich, je nachdem, welche Tabelle oder Datenbank angezeigt wird, und werden in vielen der Aktionen verwendet
//Name der aktuell ausgewählten Datenbank
35 references
[BindProperty] public String? databaseName { get; set; }
//Name der aktuell ausgewählten Tabelle
23 references
[BindProperty] public String? tableName { get; set; }
//Statische Speicherung des aktuell ausgewählten Datenbanknamens
private static String? currentDatabaseName;
//Statische Speicherung des aktuell ausgewählten Tabellennamens
private static String? currentTableName;
```

Hier sieht man alle Attribute, die in der ‚Index.cshtml.cs‘ benutzt werden und eine Beschreibung von ihnen.

```
//Hier wird geschaut welches Form Daten enthält und somit vom user benutzt wird und dann werden je nachdem andere Aktionen ausgeführt
//Datenbank anzeigen
if(Request.Form["navdatabase"] != "") {
    action = Request.Form["navdatabase"].ToString().Split(',')[0];
    databaseName = Request.Form["navdatabase"].ToString().Split(',')[1];
    currentDatabaseName = databaseName;
}
//Tabelle anzeigen
if(Request.Form["navdatatable"] != "") {
    action = Request.Form["navdatatable"].ToString().Split(',')[0];
    databaseName = Request.Form["navdatatable"].ToString().Split(',')[1];
    if(Request.Form["viewTableNames"] != "") {
        tableName = Request.Form["viewTableNames"].ToString();
    }
    else {
        tableName = Request.Form["navdatatable"].ToString().Split(',')[2];
    }
    //tableName = Request.Form["navdatatable"].ToString().Split(',')[2];
    currentDatabaseName = databaseName;
    currentTableName = tableName;
}
//Tabellen Relationen erstellen Schritt 1 (Seite zum angeben der nötigen Daten anzeigen)
if(Request.Form["changeRelationsBtn"].ToString() == "Relationen der Tabellen ändern") {

    if(GetDatabase().tables.Count < 2) {
        throw new Exception("Es gibt keine Tabellen in der Datenbank, es werden mindestens 2 benötigt um relationen zu erstellen");
    }

    action = Request.Form["changeTableRelations"].ToString();
    return;
}
```

Beispiel zur Auswertung des Forms

Hier wird geschaut, ob die bestimmte Form Daten enthält, die für diesen Fall passend sind und wenn ja, werden diese ausgewertet und eine Aktion gesetzt, die den passenden Inhalt auf der Webseite anzeigt.

```
//Datenbanken anzeigen
else if(@Model.action.ToLower() == "viewdatabase") {
    @:<br><h1>Datenbank: @Model.GetDatabase().databaseName<br>Tabellen:<br></h1><br>
    @:<table class="viewTables">
        foreach(Table table in @Model.GetDatabase().tables) {
            <tr>
                <td>
                    <input type="submit" name="viewTableNames" value="@table.tableName" />
                    <input type="hidden" name="navdatatable" value="viewDatatable,@Model.GetDatabase().databaseName,@table.tableName">
                </td>
            </tr>
        }
    @:</table>
    <text>
    <br>
    <input type="submit" class="boldSubmit" name="createBtn" value="Neue Tabelle erstellen" />
    <input type="hidden" name="createnewTable" value="createnewtable,@Model.GetDatabase().databaseName" />
    <br><hr>
    Datenbank Filtern:
    Suchart: <select name="searchType">
        <option value="table">Table</option>
        <option value="data">Data</option>
        <option value="column">Column</option>
    </select>
    <br /><br>Suchtext: <input type="text" name="searchkeyword">
    <input type="hidden" name="searchDatabases" value="searchkeyword" />
    <br><br><input type="submit" class="boldSubmit" name="searchBtn" value="Datenbank filtern" />
    <br><hr>
    <input type="submit" class="boldSubmit" name="deleteBtn" value="Datenbank löschen" onclick="clicked(event)"/>
    <input type="hidden" name="deleteDatabase" value="delete,database,@Model.GetDatabase().databaseName" />
    <br><hr>
    <input type="submit" class="boldSubmit" name="changeRelationsBtn" value="Relationen der Tabellen ändern" />
    <input type="hidden" name="changeTableRelations" value="changeRelations" />
    <br><hr>
    </text>
}
```

Auswertung der Verarbeiteten Daten und anzeigen der Informationen (Bestimmte Datenbank anzeigen)

```

//Tabelle anzeigen
else if(@Model.action.ToLower() == "viewdatatable") {
    @:<br><h1>Tabelle: @Model.tableName</h1><br>
    @:<table>
    @:<tr>
        //Alle Kopfzeilen der Tabelle anzeigen
        foreach(DataColumn col in @Model.GetTable().table.Columns) {
            @:<th>
            Boolean added = false;
            foreach(DataRow row in @Model.GetDatabase().tableConstraints.Rows){
                if (row[1].ToString()==@col.ColumnName && row[0].ToString()==@Model.GetTable().tableName){
                    if(row[2].ToString()==" " && row[3].ToString()==""){
                        @:[PRK]@col.ColumnName.ToString()
                        added = true;
                        break;
                    }
                    else{
                        @:[FRK]@col.ColumnName.ToString()
                        added = true;
                        break;
                    }
                }
            }
            if(!added){
                @col.ColumnName.ToString()
            }
            @:</th>
        }
    @:</tr>
    foreach(DataRow row in @Model.GetTable().table.Rows) {
        @:<tr>
        //Alle Zellen der Tabelle anzeigen
        foreach(DataColumn col in @Model.GetTable().table.Columns) {
            <td>
                @row[col].ToString()
            </td>
        }
        @:</tr>
    }
    @:</table>
    //Funktionen nach der Anzeige
    <text>
        <br><br>
        <input type="submit" class="boldSubmit" name="createBtn" value="Neue Reihe hinzufügen" />
        <input type="hidden" name="createnewtablerow" value="addnewtablerow,@Model.GetDatabase().databaseName,@Model.GetTable().tableName" />
        <br><br>
        <input type="submit" class="boldSubmit" name="alterBtn" value="Tabellenattribute ändern" />
        <input type="hidden" name="alterTable" value="altertable,@Model.GetDatabase().databaseName,@Model.GetTable().tableName" />
        <input type="submit" class="boldSubmit" name="deleteBtn" value="Tabelle löschen" onclick="clicked(event)"/>
        <input type="hidden" name="deleteDatabase" value="delete,table,@Model.GetDatabase().databaseName,@Model.GetTable().tableName" />
        <br><br>
    </text>
}

```

Auswertung der Verarbeiteten Daten und anzeigen der Informationen (Bestimmte Tabelle anzeigen)

Nach dem Auswerten der Daten werden diese auf der Seite angezeigt. Hier wird die Aktion ‚viewdatabase‘ oder die Aktion ‚viewdatatable‘ angezeigt, falls die Aktion, die nach der Auswertung gesetzt wurde, dazu passt.

```

<div>
<header>
<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
  <div class="container">
    <a class="navbar-brand" asp-area="" asp-page="/Index"></a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
      aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
      <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
          <a class="nav-link text-dark" asp-area="" asp-page="/Index">Startseite</a>
        </li>
      </ul>
    </div>
  </div>
  <div class="logo">
    <i class="bx bx-menu menu-icon"></i>
    <span class="logo-name"></span>
  </div>
</nav>
</header>

<div class="container">
  <main role="main" class="pb-3">
    @RenderBody()
  </main>
</div>

<footer class="border-top footer text-muted">
  <div class="container">
    &copy; 2023 - DatenbankManagementSystem Author: Benjamin Born - <a asp-area="" asp-page="/Datenschutz">Datenschutz</a> - <a asp-area="" asp-page="/Impressum">Impressum</a>
  </div>
</footer>

```

Ausschnitt aus Layout.cshtml

Der Code von Layout.cshtml ist für das Darstellen von der oberen Navigationsleiste und dem Logo zuständig und dem Anzeigen des Content-Windows mit '@RenderBody()', mit dem im Content-Window, verschiedene Seiten dynamisch mit der Razor-Page Logik angezeigt werden können.

## 6. Qualitätskontrolle

Es wurden jeweils ein Blackbox-Test und ein Whitebox-Test zur Sicherung der Qualität des Programmes durchgeführt. Nun wird nochmal kurz erklärt, um was es sich bei diesen Tests genau handelt.

Beim Whitebox Test testen und analysieren generell die Programmierer selber den Code und die Funktion des Programmes und testen die Funktionen alle. Auch wird die Logik des Programmcodes nochmal genau unter die Lupe genommen.

Beim Blackbox-Test testen außenstehende Personen, die nicht an dem Programm mitarbeitet haben und kein Wissen vom Programmcode haben, die Funktionen des Programms mithilfe der Benutzeroberfläche.

Folgende Funktionen wurden bei den Tests geprüft:

- Erstellung einer Datenbank
- Import von bestehenden Datenbanken
- Die Erstellung von Tabellen in Datenbanken
- Änderung von bestehenden Tabellen (Attribute, Beziehungen)
- Filtern von Datenbanken
- Suchen von Tabellen in mehreren Datenbanken
- Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken
- Schutz von systemrelevanten Datenbanken

## 6.1 Whitebox-Test

Es wurde ein funktionaler Whitebox Test vom Entwickler selber durchgeführt.

Dabei wurden folgende Fehler gefunden:

Die Verlinkung der Tabellen auf der Datenbankansicht Seite war nicht komplett funktionell. Nur die erste Tabelle wurde richtig verlinkt. Dies wurde mit einer einzigartigen Benennung der ‚Submit‘-Knöpfe, die diese verlinken gelöst, die dann im Backend des Programmes ausgewertet werden.

Bei der Veränderung der Tabellenattribute wurde ein Sql Fehler ausgegeben, obwohl der Syntax richtig war. Dies wurde mit dem Ändern des Sql Syntax von ‚Alter [Tabellenname] RENAME [alter Name] [neuer Name]‘ in ‚Alter [Tabellenname] CHANGE COLUMN [alter Name] [neuer Name] [neuer Tabellen Datentyp]‘ gelöst. Diese neue Syntax wurde auch für das Ändern des Datentyps benutzt.

## 6.2 Blackbox-Test

Ein Kollege Roy de Blank führte in meinem Programm einen funktionalen Blackbox-Test durch.

Dabei wurden folgende Fehler gefunden:

Es ist möglich Sonderzeichen bei der Erstellung der Tabelle oder Datenbank einzugeben, welche unter anderem zu einem Verbindungs-String Error auslösen kann, falls ein ‚?‘ in dem Namen ist.

Dies wurde gelöst durch die Implementierung einer Regex Funktion ‚[^\a-zA-Z0-9]+‘, um mit ‚.isMatch()‘ zu kontrollieren, ob in dem Tabellennamen Sonderzeichen enthalten sind.

Auch war es möglich eine neue Tabellenreihe hinzuzufügen ohne mindestens eine Spalte mit Namen und Datentyp zu definieren. Dies wurde durch das Hinzufügen des ‚required‘ Attributes in dem HTML Form gelöst.

## 7. Abnahmephase

Der Kunde wird das Programm vorbereitet als Installationsdatei geliefert bekommen. Er kann dann einen Pfad auswählen, an dem er das Programm installieren will. Nach der Installation des Programmes muss der Kunde, falls er dies noch nicht hat, XAMPP installieren, damit ein MySQL Server gegeben ist, auf dem das Programm die Datenbanken verwalten und modifizieren kann.

Nachdem die Installation durchgeführt wurde, kann der Kunde nun mit der .exe des Programms dieses ausführen. Falls bereits Datenbanken auf dem MySQL Server vorhanden sind, werden diese beim Start des Programms auch direkt geladen und können modifiziert werden. Falls noch keine bestehen, kann der Kunde mit dem Programm jederzeit eine neue Datenbank erstellen und Daten zu dieser hinzufügen.

Das Verfahren der Installation des Programmes ist nochmal detailliert in [10. Kundendokumentation](#) erklärt.

## 8. Einführungsphase

Nach der Erklärung des Programms konnte der Kunde direkt damit anfangen seinen Mitarbeiter mit diesem Programm arbeiten zu lassen, da die Bedienung sehr selbsterklärend ist, nachdem er die [Kundendokumentation](#) gelesen hatte.

Er konnte dann auch direkt das alte Programm, das Fehler hatte und nicht mehr effizient gelaufen ist, durch das neue Programm ersetzen und mit diesem seine Arbeit ausführen.

Nach einer Woche wurde eine positive Rückmeldung vom Kunden über das Programm gegeben und es wurde ein Folgeauftrag angeboten um das Programm, um einige noch gewünschte Funktionen zu erweitern. Mehr Details dazu in [9.3 Ausblick](#).

## 9. Fazit

### 9.1 Soll-/Ist-Vergleich

Der [Sollzustand](#) wird hier eben mit dem Ist-Zustand des Programmes verglichen, um sicherzustellen, dass alle Funktionen, die vom Kunden gewünscht waren integriert sind und das Programm wie gewünscht funktioniert.

- **Erstellung einer Datenbank**  
Die Erstellung einer neuen Datenbank ist in der Benutzeroberfläche möglich.
- **Import von bestehenden Datenbanken**  
Der Import von bestehenden Datenbanken in MySQL wird bei Programmstart automatisch ausgeführt.
- **Die Erstellung von Tabellen in Datenbanken**  
In der Benutzeroberfläche kann eine Tabelle in einer beliebigen Datenbank erstellt werden.
- **Änderung von bestehenden Tabellen (Attribute, Beziehungen)**  
In der Datenbankansicht ist es möglich, die Relationen der Tabellen zu ändern.  
In der Tabellenansicht ist es möglich, die Spaltenattribute zu ändern oder eine neue Spalte hinzuzufügen.
- **Filtern von Datenbanken und Suchen von Tabellen in mehreren Datenbanken**  
Das Filtern von Datenbanken ist in drei verschiedenen Wegen möglich.
  - Filtern nach Tabellennamen
  - Filtern nach Spaltennamen
  - Filtern nach allen Zeilen und Spalten in der Datenbank
- **Löschen von Datenbanken oder einzelnen Tabellen in Datenbanken**  
Das Löschen von Datenbanken oder einzelnen Tabellen ist implementiert.
- **Schutz von systemrelevanten Datenbanken**  
Die systemrelevanten Datenbanken werden nicht in das Programm geladen und sind nicht für den Benutzer sichtbar.

## 9.2 Lessons Learned

Die Implementierung des Programmes ist gut gelaufen. Alle Funktionen wurden wie geplant implementiert. Es gab einige Probleme mit dem Verbinden des Backend und der Benutzeroberfläche. Dies kann entweder daran liegen, dass der Programmierer wenig Erfahrung mit dem Framework Asp.Net Razorpages hat oder das generell für eine lokale Anwendung eher eine Benutzeroberfläche in WinForm oder in WPF besser geeignet ist. Dennoch funktioniert die Benutzeroberfläche wie gewünscht und das Programm wurde im Rahmen des [Zeitplans](#) fertiggestellt.

## 9.3 Ausblick

Es könnten noch Funktionen wie z.B. der Import und das Erstellen von .sql Backups wie in PhpMyAdmin oder das Bauen einer Netzwerkversion implementiert werden.

Bei der Netzwerkversion könnten mehrere Mitarbeiter auf einmal auf das Programm zugreifen und die Datenbank verwalten.

Es könnten noch Benutzerkonten von MySQL integriert werden, damit man mehrere Konten mit verschiedenen Rechten vergeben kann.

Auch wäre es möglich, Trigger einzubauen. Was zum Beispiel passiert falls, eine Datenbank gelöscht wird.

Das Programm ist dafür optimiert, dass es einfach ist noch Funktionen einzubauen ohne viel vom bestehenden Programmcode umschreiben zu müssen.

## 10. Kundendokumentation

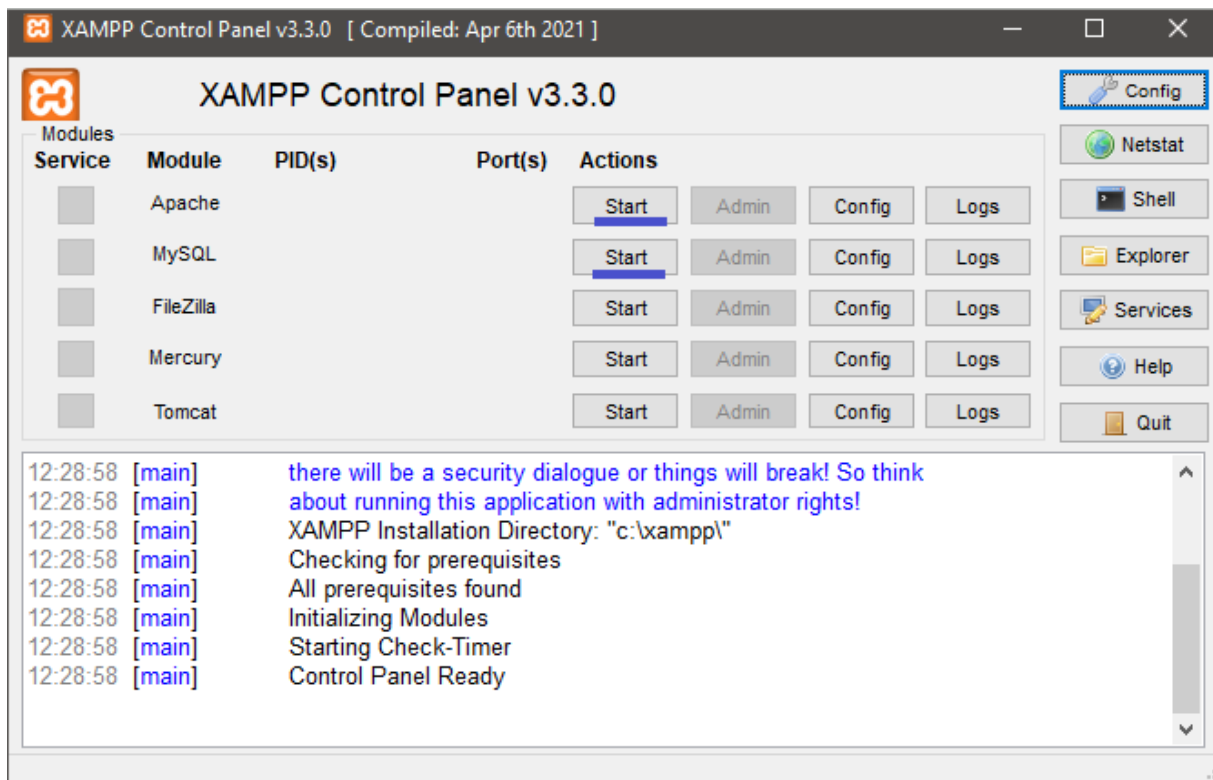
Bedienungsanleitung zu dem Datenbankmanagement System Programm:

Bitte führen Sie erst die Setup-Datei aus und installieren Sie das Programm in dem gewünschten Pfad auf Ihrem Computer. Nach der Installation von diesem bitte überprüfen Sie, ob sie XAMPP auf Ihrem PC installiert haben. Falls dies nicht der Fall ist, können Sie das nötige Programm auf dieser Seite herunterladen:

<https://www.apachefriends.org/download.html>

Nachdem die Installation erfolgreich ist, bitte starten Sie XAMPP. Sie sollten nun folgendes Fenster sehen:





Starten Sie in XAMPP jeweils den Apache- und den MySQL-Service. Oben im Programm sind die Knöpfe die sie drücken müssen blau unterstrichen.

Nun können Sie das Programm starten und Sie werden folgende Seite sehen:

## Startseite



Hier können Sie nun links auf der Seitenleiste alle Datenbanken sehen, die bei Ihnen schon bestehen. Falls noch keine bestehen, können Sie dort eine [neue Datenbank erstellen](#). Dies wird im nächsten Schritt auch nochmal erklärt. Sie können jetzt auf der Seitenleiste entweder auf eine Datenbank klicken, um die [Datenbankansicht](#) anzuzeigen oder auf eine der Tabellen, die unter dem Datenbank-Knopf angezeigt werden. Dann werden Sie auf die [Tabellenansicht](#) weitergeleitet.



## Neue Datenbank erstellen

### Importierte Datenbanken

#### testdbwithvalues1

ansprechpartner  
lieferanten

#### testdbwithvalues2

artikel  
filiale  
verkauf

#### testdbwithvalues3

auftraege  
hanswurst  
kunden  
modelle

Neue Datenbank erstellen

## Neue Datenbank erstellen

Datenbankname:

Submit

Hier können Sie einen Namen für die Datenbank eingeben und dann auf ‚Submit‘ drücken. Danach wird die Seite aktualisiert und die Datenbank wird auf der Seitenleiste auftauchen.

## Datenbankansicht

### Importierte Datenbanken

#### testdbwithvalues1

ansprechpartner  
lieferanten

#### testdbwithvalues2

artikel  
filiale  
verkauf

#### testdbwithvalues3

auftraege  
hanswurst  
kunden  
modelle

Neue Datenbank erstellen

## Datenbank: testdbwithvalues1 Tabellen:

ansprechpartner  
lieferanten

Neue Tabelle erstellen

Datenbank Filtern: Suchart:

Suchtext:

Datenbank filtern

Datenbank löschen

Relationen der Tabellen ändern

Hier haben Sie eine Übersicht von der Datenbank, die Sie in der Seitenleiste angeklickt haben. Hier können Sie nun auf eine der Tabellen in der Datenbank klicken, um die [Tabellenansicht](#) dieser anzuzeigen. Oder Sie können eine [neue Tabelle erstellen](#) mit dem Knopf ‚Neue Tabelle erstellen‘. Auch ist es von hier aus möglich, die Daten dieser Datenbank zu filtern.

Es gibt drei verschiedene Möglichkeiten diese zu filtern:

- Durchsuchen nach Tabellennamen (Table)
- Durchsuchen nach Spaltennamen (Column)
- Durchsuchen nach allen Zeilen und Spalten in jeder Tabelle dieser Datenbank (Data)


Wählen Sie Ihren Suchtypen aus und danach müssen einfach das Wort, nachdem Sie suchen wollen in das Suchtext-Feld eingeben und auf ‚Datenbank filtern‘ klicken.

Das [Ergebnis der Suche](#) wird danach in einem neuen Fenster für Sie angezeigt.

Es gibt auch noch eine Möglichkeit die Datenbank mit dem drücken auf ‚Datenbank löschen‘ zu löschen.

Falls Sie die [Relationen der Tabelle in der Datenbank hinzufügen](#) wollen, können Sie dies mit einem Klick auf ‚Relationen der Tabelle ändern‘

## Datenbank Filtern Ergebnis



**SCHWENGE**  
garden construction

Startseite

### Importierte Datenbanken

**testdbwithvalues1**  

ansprechpartner

lieferanten

**testdbwithvalues2**  

artikel

filiale

verkauf

**testdbwithvalues3**  

auftraege

hanswurst

kunden

modelle


Neue Datenbank erstellen

Folgende Ergebnisse wurden bei der Suche nach "a" gefunden:

Datenbankname	Tabellenname	Spaltenname	Reihenposition
testdbwithvalues1	ansprechpartner		
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	AID	
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	Name	
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	Vorname	
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	E-Mail	
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	Vorname	Reihe:1
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	Vorname	Reihe:2
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	E-Mail	Reihe:2
<b>Datenbankname</b>	<b>Tabellenname</b>	<b>Spaltenname</b>	<b>Reihenposition</b>
testdbwithvalues1	ansprechpartner	Vorname	Reihe:3

Hier werden die Ergebnisse der Suche, die in der [Datenbankansicht](#) ausgeführt wurde. Je mehr Ergebnisse die Suche hat, desto mehr Reihen hat die Tabelle, die zum Anzeigen der Suchergebnisse benutzt wird.

## Neue Tabelle erstellen

 **SCHWENGE**  
garden construction

Startseite

### Importierte Datenbanken

**testdbwithvalues1**

ansprechpartner  
lieferanten

**testdbwithvalues2**

artikel  
filiale  
verkauf

**testdbwithvalues3**

auftraege  
hanswurst  
kunden  
modelle

Neue Datenbank erstellen

## Neue Tabelle erstellen

Tabellenname:

Spalte 1:  
Spaltenname:  Spaltentyp:

Spalte 2:  
Spaltenname:  Spaltentyp:

Spalte 3:  
Spaltenname:  Spaltentyp:

Spalte 4:  
Spaltenname:  Spaltentyp:

Spalte 5:  
Spaltenname:  Spaltentyp:

Submit

Hier können Sie eine neue Tabelle erstellen. Dazu müssen Sie einen Tabellennamen und mindestens eine Spalte mit Namen und Datentyp ausfüllen.

Mit dem drücken auf ‚Submit‘ wird die Tabelle dann erstellt.

## Tabellenansicht

 Startseite

### Importierte Datenbanken

**testdbwithvalues1**  
ansprechpartner  
lieferanten

**testdbwithvalues2**  
artikel  
filiale  
verkauf

**testdbwithvalues3**  
auftraege  
hanswurst  
kunden  
modelle

Neue Datenbank erstellen

### Tabelle: artikel

[PRK]ArtikelID	Artikelname	Verkaufspreis	Einkaufspreis
100	Salat de Luxe	1.50	0.70
101	Hamburger TS Royal	4.80	2.10
102	Steakburger	5.10	2.25
103	Cheeseburger	1.80	1.15
104	McSundea Schoko	1.00	0.01
105	Cheeseburger	2.00	1.00
106	Hamburger Royal TS	3.00	2.00
107	McFlurry	2.00	1.00
108	Signature Burger Beff and Egg	7.00	4.00
109	Hamburger Original	1.50	0.80
110	Chicken Nuggets	2.00	1.00
111	BicMac	5.00	3.00
113	MCBurger	1.00	0.50
115	Tasty Chicken	3.00	1.00
116	Chicken MC-Kai	4.00	3.00
117	Big Rösti Chicken	3.00	9.00

Neue Reihe hinzufügen

Tabellenattribute ändern Tabelle löschen

Hier werden die Spalten und Zeilen der Tabellendaten für Sie angezeigt. Falls ein [PRK] oder [FRK] vor einem Spaltennamen angezeigt wird, ist diese Spalte ein Primärschlüssel oder ein Fremdschlüssel.


Es gibt auch noch folgende Funktionen auf dieser Seite:

-[Neue Tabellenreihe hinzufügen](#)

-[Tabellenattribute ändern](#)

-Tabelle löschen

## Neue Tabellenreihe hinzufügen


Startseite

### Importierte Datenbanken

**testdbwithvalues1**  

ansprechpartner

lieferanten

**testdbwithvalues2**  

artikel

filiale

verkauf

**testdbwithvalues3**  

auftraege

hanswurst

kunden

modelle

Neue Datenbank erstellen


## Neue Tabellenreihe hinzufügen

[PRK]ArtikelID	Artikelname	Verkaufspreis	Einkaufspreis
Int32	String	Decimal	Decimal

Tabellenreihe erstellen

Hier kann man eine neue Reihe zu der Tabelle hinzufügen. Es stehen jeweils der Name der Spalte und der Datentyp dieser über den Werten, den Sie in den Textboxen darunter eingeben können.

## Relationen in der Datenbank hinzufügen


Startseite

### Importierte Datenbanken

**testdb1**  

hans

schwein

**testdbwithvalues1**  

ansprechpartner

lieferanten

**testdbwithvalues2**  

artikel

filiale

verkauf

**testdbwithvalues3**  

auftraege

hanswurst

kunden

modelle

Neue Datenbank erstellen

## Relationen in der Datenbank testdbwithvalues1 hinzufügen

ansprechpartner	[PRK]AID	Name	Vorname	Telefon	E-Mail	Bereich	LID
lieferanten	[PRK]LID	Name	Straße	PLZ	ORT		

Primary Key

[ansprechpartner]AID


Foreign Key

[ansprechpartner]AID

Primär- und Fremdschlüssel Verbindung hinzufügen

Hier können Sie einen Primär- und einen Fremdschlüssel definieren, der 2 Tabellen miteinander verbindet. Danach können Sie mit ‚Primär- und Fremdschlüssel Verbindung hinzufügen‘ diese hinzufügen.

## Spalten der Tabelle ändern

 Startseite

### Importierte Datenbanken

**testdbwithvalues1**  
ansprechpartner  
lieferanten

**testdbwithvalues2**  
artikel  
filiale  
verkauf

**testdbwithvalues3**  
auftraege  
hanswurst  
kunden  
modelle

Neue Datenbank erstellen

## Spalten der Tabelle filiale ändern

[PRK]FilialeID	sadasd
Int32	String
Neue Attribute:	
Name	Name
Datentyp	Datentyp
<input type="text"/>	<input type="text"/>

Submit

## Neue Spalte in der Tabelle filiale erstellen

Name:  Datentyp:

Submit

(Sie Können nur eine der beiden Aktionen auf einmal aufführen)

Auf dieser Seite ist es möglich jeweils entweder die Namen oder Datentypen der Spalten der Tabellen zu ändern. Dafür können Sie bei Namen einen neuen Namen für die Spalte eingeben oder bei Datentyp einen neuen Datentyp einstellen. Mit dem Klick auf Submit wird diese Aktion dann ausgeführt.

Es ist auch möglich, eine Spalte in der Tabelle hinzuzufügen. Dazu müssen Sie jeweils den Namen und den Datentyp der Tabelle eingeben und dann auf ‚Submit‘ klicken.

## 11. Glossar

Begriff	Erklärung
BBWN	Berufsbildungswerk Neckargemünd
Razor Pages	Ein neueres, vereinfachtes Modell zur Programmierung von Webanwendungen
Asp.Net	Ein kostenloses Web-Framework für die Erstellung von Webanwendungen mit HTML, CSS und JavaScript
Datenbankmanipulation	Prozess, bei dem Daten manipuliert oder neu formatiert werden, damit sie leichter gelesen oder besser organisiert werden können
MySQL	MySQL ist ein quelloffenes relationales Datenbankmanagementsystem
Import	Von einem anderen Programm oder einer anderen Speicherform Informationen zu kopieren
Basis-Funktionen	Grundfunktionen
DataSet	Ein DataSet besteht aus einer Auflistung von Tabellen, Beziehungen und Einschränkungen.
DataTable	Eine DataTable stellt eine Tabelle mit relationalen Daten im Speicher dar. Sie kann Teil eines DataSet sein
HTML	Hyper Text Markup Language
CSS	Cascade Style Sheets
Javascript	Eine Programmiersprache, die eine der Kerntechnologien des Internets ist
Systemrelevante	Bedeutung oder Wichtigkeit von diesen Programmteilen für das System
Datenbank Management System	Ein Datenbankmanagementsystem ist verantwortlich für die Verwaltung der Daten, den Benutzern den Zugriff auf die Daten zu ermöglichen, und die Organisationsstruktur einer Datenbank.
Microsoft Visual Studio	Visual Studio ist eine integrierte Entwicklungsumgebung von Microsoft. Sie wird für die Entwicklung von Computerprogrammen verwendet, einschließlich Websites, Webanwendungen, Webdienste und mobile Anwendungen.
Microsoft Visual Studio Lizenz	Die Produktlizenz des Programmes
Entwicklungsprozess	Prozess, in dem sich eine Entwicklung vollzieht
Geschäftsprozesse	Ein Geschäftsprozess ist eine Abfolge von Aktivitäten zur Erfüllung einer betrieblichen Aufgabe.

Systemintegrator	Ein Systemintegrator ist eine Einzelperson oder ein Unternehmen, das Computersysteme für Kunden zusammenstellt
Schlüsselwerte	Ein Schlüssel, der ein eindeutiger Bezeichner (Identifizier) für ein Datenelement ist
phpMyAdmin	phpMyAdmin ist eine kostenlose Webanwendung für die Verwaltung von MySQL- und MariaDB-Datenbanken.
Management-System	Ein Managementsystem bezeichnet die Art und Weise, wie sich ein Softwareprogramm in der Struktur und dem Ablauf organisiert, um systematisch zu handeln, reibungslose Abläufe sicherzustellen und geplante Ergebnisse zu erreichen.
Modifikations-System	Die Veränderung von Daten in einer Datenbank in diesem Falle
Software	Software ist ein Programm oder eine Menge von Programmen, die dazu dienen, einen Computer zu betreiben.
Softwareprodukt	Ein Produkt oder eine Software, die hergestellt wird, um an Benutzer verkauft zu werden.
Personalaufwand	Der Personalaufwand ist die Summe der an die Mitarbeiter ausgezahlten Löhne und Gehälter zuzüglich der Sozialabgaben sowie der Altersvorsorge und Unterstützungsleistungen eines Unternehmens für seine Arbeitnehmerinnen und Arbeitnehmer.
Framework	Ein Framework ist ein Rahmenwerk für die Softwareentwicklung und Programmierung, dass die Grundstruktur und das Programmiergerüst für die zu erstellende Software vorgibt.
Programmiersprache	Eine Programmiersprache ist ein Notationssystem zum Schreiben von Computerprogrammen.
Frontend	Das Frontend ist der für Nutzer sichtbare Teil einer Webseite oder Applikation.
Backend	Das Backend bezieht sich auf den funktionalen Teil von Softwarelösungen, also die Datenverarbeitung, welche Nutzer nicht sehen.
Razor Engine	ASP.NET Razor View Engine (Razor) ist die in ASP.NET verwendete Vorlagensprache für Webseiten, bei der man C# oder Visual Basic .NET-Fragmente in HTML-Seiten einbetten kann.
parst	Ein Parser ist ein Programm, das Teil des Compilers ist, und das Parsing ist Teil des Compiler-Prozesses. Das Parsen erfolgt während der Analysephase der Kompilierung. Beim Parsen wird der Code aus dem



	Präprozessor entnommen, in kleinere Teile zerlegt und analysiert, damit andere Software ihn verstehen kann.
GUI	GUI ist das Akronym für ‚Graphical User Interface‘ und beschreibt die grafische Benutzeroberfläche von Computersystemen, um die Bedienung zu erleichtern.
Content-Window	Das Fenster, das benutzt wird, um den Großteil der Informationen der einzelnen Seiten anzuzeigen
[Bindproperty]	Das Attribut [Bindproperty] kann bei einer PageModel-Klasse verwendet werden, um die Modellbindung auf alle öffentlichen Eigenschaften der Klasse anzuwenden.
Objekt	In der Informatik kann es sich bei einem Objekt um eine Variable, eine Datenstruktur, eine Funktion oder eine Methode handeln.
Root Ordner	Damit ist das Verzeichnis eines Dateisystems gemeint, das sich in der Hierarchie an erster Stelle befindet. In diesem Fall der Ordner aus dem das Programm ausgeführt wird.
Datenbank	Eine Datenbank ist eine organisierte Sammlung von strukturierten Informationen oder Daten, die typischerweise elektronisch in einem Computersystem gespeichert sind.
[PRK] und [FRK]	Primär und Fremdschlüsselabkürzung die bei der Darstellung des Programmes verwendet wird.
DataTable	Eine virtuelle Tabelle mit der zur Laufzeit des Programmes gearbeitet wird
DataSet	Ein DataSet ist das zentrale Element der Datenbankzugriffsschnittstelle ADO.NET und repräsentiert eine Menge von Tabellen (Relationen) im Hauptspeicher.
SqlDataAdapter	Stellt einen Satz von Datenbefehlen und eine Datenbankverbindung dar, die verwendet werden, um das DataSet aufzufüllen und eine SQL Server-Datenbank zu aktualisieren.
Compiler	Ein Compiler ist ein spezielles Programm, das den Quellcode einer Programmiersprache in Maschinencode, Bytecode oder eine andere Programmiersprache übersetzt.
Regex	Ein Regulärer Ausdruck stellt ein verallgemeinertes Suchmuster in der Informatik dar.
XAMPP	XAMPP ist eine Kombination des Apache Webserver, der Datenbank MySQL und der Skriptsprachen Perl und PHP. Jedoch wurde im Projekt nur die MySQL Datenbank und phpMyAdmin benötigt.

Methoden/Funktionen	Funktionen sind unabhängig. Einer Funktion kann beliebiges übergeben werden, mit dem dann weitergearbeitet wird. Methoden sind festgelegt – sprich jedes Objekt verfügt über bestimmte Möglichkeiten.
Link	Verlinkung zu einer Stelle oder Seite
Seitenleiste	Ein Menü das Links oder Rechts an der Seite einer Webseite ist und zur Navigation der Webseite dient.
Benutzeroberfläche	Alles was der Benutzer sehen und bedienen kann.

## 11.1 Glossarquellen

Links zu den Definitionen, bei denen diese 1 zu 1 aus dem Internet genommen wurde.

Begriff	Link
Microsoft Visual Studio	<a href="https://g.co/kgs/ZgHFqi">https://g.co/kgs/ZgHFqi</a>
phpMyAdmin	<a href="https://g.co/kgs/isyhPJ">https://g.co/kgs/isyhPJ</a>
Management-System	<a href="https://g.co/kgs/xBzEjL">https://g.co/kgs/xBzEjL</a>
Software	<a href="https://www.techtarget.com/searchapparchitecture/definition/software">https://www.techtarget.com/searchapparchitecture/definition/software</a>
Personalaufwand	<a href="https://www.billomat.com/lexikon/p/personalaufwand/">https://www.billomat.com/lexikon/p/personalaufwand/</a>
Framework	<a href="https://www.cloudcomputing-insider.de/was-ist-ein-framework-a-1104630/#:~:text=Ein%20Framework%20ist%20ein%20Rahmenwerk,unterst%C3%BCtzt%20objekt%2D%20und%20komponentenorientierte%20Entwicklungsans%C3%A4tze.">https://www.cloudcomputing-insider.de/was-ist-ein-framework-a-1104630/#:~:text=Ein%20Framework%20ist%20ein%20Rahmenwerk,unterst%C3%BCtzt%20objekt%2D%20und%20komponentenorientierte%20Entwicklungsans%C3%A4tze.</a>
Razor Engine	<a href="https://www.it-visions.de/glossar/alle/7299/ASPNET_Razor_View_Engine.aspx">https://www.it-visions.de/glossar/alle/7299/ASPNET_Razor_View_Engine.aspx</a>
parst	<a href="https://www.computerweekly.com/de/definition/Parser#:~:text=Ein%20Parser%20ist%20ein%20Programm,andere%20Software%20ihn%20verstehen%20kann.">https://www.computerweekly.com/de/definition/Parser#:~:text=Ein%20Parser%20ist%20ein%20Programm,andere%20Software%20ihn%20verstehen%20kann.</a>
GUI	<a href="http://www.softselect.de/business-software-glossar/gui#:~:text=GUI%20ist%20das%20Akronym%20f%C3%BCr,von%20Eingabeger%C3%A4ten%20wie%20einer%20Maus.">http://www.softselect.de/business-software-glossar/gui#:~:text=GUI%20ist%20das%20Akronym%20f%C3%BCr,von%20Eingabeger%C3%A4ten%20wie%20einer%20Maus.</a>

[Bindproperty]	<a href="https://endjin.com/blog/2022/01/model-binding-in-asp-net-core-using-razor-pages#:~:text=The%20%5BBindProperties%5D%20attribute%20can%20be,public%20properties%20of%20the%20class.">https://endjin.com/blog/2022/01/model-binding-in-asp-net-core-using-razor-pages#:~:text=The%20%5BBindProperties%5D%20attribute%20can%20be,public%20properties%20of%20the%20class.</a>
Datenbank	<a href="https://www.oracle.com/de/database/what-is-database/">https://www.oracle.com/de/database/what-is-database/</a>
DataSet	<a href="https://www.it-visions.de/glossar/alle/494/Das_Dataset_als_zentrales_Element_der_Datenbankzugriffsschnittstelle_ADONET.aspx#:~:text=Ein%20DataSet%20ist%20das%20zentrale,%20DMemory%20Database%20bezeichnet%20werden.">https://www.it-visions.de/glossar/alle/494/Das_Dataset_als_zentrales_Element_der_Datenbankzugriffsschnittstelle_ADONET.aspx#:~:text=Ein%20DataSet%20ist%20das%20zentrale,%20DMemory%20Database%20bezeichnet%20werden.</a>
SqlDataAdapter	<a href="https://learn.microsoft.com/de-de/dotnet/api/system.data.sqlclient.sqldataadapter?view=dotnet-plat-ext-7.0">https://learn.microsoft.com/de-de/dotnet/api/system.data.sqlclient.sqldataadapter?view=dotnet-plat-ext-7.0</a>
Compiler	<a href="https://www.computerweekly.com/de/definition/Compiler-Kompiler">https://www.computerweekly.com/de/definition/Compiler-Kompiler</a>

## 12. Literaturverzeichnis

### Internetseiten:

<https://lumiformapp.com/de/ressourcen-checklisten/iso-25010-zertifizierung>

<https://www.adito.de/knowhow/blog/ist-analyse>

[http://www.softselect.de/wissenspool/erlaeuterung\\_lastenheft\\_vs\\_pflichtenheft#:~:text=Das%20Lastenheft%20beschreibt%20die%20gesamte,Lastenheft%20gew%C3%BCnschten%20Funktionen%20umgesetzt%20werden.](http://www.softselect.de/wissenspool/erlaeuterung_lastenheft_vs_pflichtenheft#:~:text=Das%20Lastenheft%20beschreibt%20die%20gesamte,Lastenheft%20gew%C3%BCnschten%20Funktionen%20umgesetzt%20werden.)

<https://www.centron.de/2013/06/18/einfuehrung-in-asp-net-web-pages-und-razor-syntax/>

<https://www.atlassian.com/de/continuous-delivery/software-testing/types-of-software-testing>

<https://fachinformatiker-azubi.de/Schreibtischtest>

<https://www.cool-lab.net/faq/was-ist-ein-schreibtischtest/>

<https://www.srh.de/de>

### Bilder:

[https://cdn.pixabay.com/photo/2020/08/07/05/45/cloud-5469737\\_1280.jpg](https://cdn.pixabay.com/photo/2020/08/07/05/45/cloud-5469737_1280.jpg)

<https://app.logo.com/> (Auftraggeber Logo)

### Codequellen:

<https://stackoverflow.com/questions/35408854/how-to-know-relations-between-tables>

MySQL Befehl zum Herausfinden von Beziehungen zwischen den Tabellen in einer DB  
(Datum 10.07.2023)

### Impressum Generator:

<https://www.impressum-generator.de/>

### Datenschutz Generator:

<https://datenschutz-generator.de/>

EPK gemacht mit:

<https://online.visual-paradigm.com/>

## 13. Quellcode

### 13.1 Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using MySQLConnector;
using System.Data;
using System.Text.RegularExpressions;
namespace DatenbankManagementSystem {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        public IndexModel(ILogger<IndexModel> logger) {
            _logger = logger;
        }

        //List of Database, die alle Datenbanken enthält, die zur Laufzeit in das
        //Programm importiert wurden
        [BindProperty] public List<Database>? databases { get; set; }
        //Die Statische Liste der Datenbanken
        private static List<Database>? databasesBase = new List<Database>();
        //Die Art der Aktion, die das Programm ausführt, wenn der Benutzer eine Aktion
        //ausführt
        [BindProperty] public String? action { get; set; }
        //Der ConnectionString zur aktuellen Datenbank
        public String connectionString = new String("");
        //Die Fehlerliste, die verwendet wird, um Fehler für den Benutzer auszugeben,
        //die er später sehen kann
        [BindProperty] public List<Exception>? errorStackTrace { get; set; }
        //Das Schlüsselwort, das der Benutzer zum Suchen oder Filtern je nach Aktion
        //eingibt
        [BindProperty] public String? searchKeyWord { get; set; }
        //Die Art der Suche, die der Benutzer durchführen möchte (Tabellennamen,
        //Spaltennamen oder alle Daten)
        [BindProperty] public String? searchType { get; set; }
```

//Die Werte dessen, was bei der Suche gefunden wurde und wo es gefunden wurde

```
[BindProperty] public List<String>? searchFoundValues { get; set; }
```

//Informationen darüber, was in einer Tabelle geändert werden soll

```
[BindProperty] public List<List<String>>? alterInfo { get; set; }
```

// Die Spaltennamen der neuen Tabelle

```
[BindProperty] public List<String>? tableColNames { get; set; }
```

//Der Wertetyp der neuen Tabellenspalten

```
[BindProperty] public List<Type>? tableColTypes { get; set; }
```

//Zeilen, die der aktuellen Tabelle hinzugefügt werden

```
[BindProperty] public List<List<Object>>? tableRowsToAdd { get; set; }
```

//Informationen darüber, welche Art von Löschung durchgeführt werden muss

```
[BindProperty] public List<String>? deleteInfo { get; set; }
```

//Diese 2 Eigenschaften unten ändern sich, je nachdem, welche Tabelle oder Datenbank angezeigt wird, und werden in vielen der Aktionen verwendet

//Name der aktuell ausgewählten Datenbank

```
[BindProperty] public String? databaseName { get; set; }
```

//Name der aktuell ausgewählten Tabelle

```
[BindProperty] public String? tableName { get; set; }
```

//Statische Speicherung des aktuell ausgewählten Datenbanknamens

```
private static String? currentDatabaseName;
```

//Statische Speicherung des aktuell ausgewählten Tabellennamens

```
private static String? currentTableName;
```

```
public void OnPost() {
```

```
    try {
```

```
        // Regex zum Rausfiltern von Sonderzeichen
```

```
        String regex = "[^a-zA-Z0-9]+";
```

```
        Regex rgex = new Regex(regex);
```

```
        //Neuerstellung der errorStackTrace falls sie null ist
```

```
        if(errorStackTrace == null) {
```

```
            errorStackTrace = new List<Exception>();
```

```
        }
```

```
        //Datenbanken werden durch statische Datenbanken überschrieben
```

```
        databases = databasesBase;
```

```
        //Bereits bestehende DBs in das Program laden
```

```

if(action == "addalreadyexistingdbs") {
    AddAlreadyExistingDbs();
    databasesBase = databases;
    return;
}

//Falls wir einen aktuellen Db oder Tabellennamen haben übernehmen wir
diesen hier
if(currentDatabaseName != null) {
    databaseName = currentDatabaseName;
}
if(currentTableName != null) {
    tableName = currentTableName;
}

//Hier wird geschaut welches Form Daten enthält und somit vom user
benutzt wird und dann werden je nachdem andere Aktionen ausgeführt
//Datenbank anzeigen
if(Request.Form["navdatabase"].ToString() != "") {
    action = Request.Form["navdatabase"].ToString().Split(',')[0];
    databaseName = Request.Form["navdatabase"].ToString().Split(',')[1];
    currentDatabaseName = databaseName;
}

//Tabelle anzeigen
if(Request.Form["navdatatable"].ToString() != "") {
    action = Request.Form["navdatatable"].ToString().Split(',')[0];
    databaseName = Request.Form["navdatatable"].ToString().Split(',')[1];
    if(Request.Form["viewTableNames"].ToString() != "") {
        tableName = Request.Form["viewTableNames"].ToString();
    }
    else {
        tableName = Request.Form["navdatatable"].ToString().Split(',')[2];
    }
    //tableName = Request.Form["navdatatable"].ToString().Split(',')[2];
    currentDatabaseName = databaseName;
    currentTableName = tableName;
}

```

```

//Tabellen Relationen erstellen Schritt 1 (Seite zum angeben der nötigen
Daten anzeigen)
if(Request.Form["changeRelationsBtn"].ToString() == "Relationen der
Tabellen ändern") {
    if(GetDatabase().tables.Count < 2) {
        throw new Exception("Es gibt keine Tabellen in der Datenbank, es
werden mindestens 2 benötigt um relationen zu erstellen");
    }
    action = Request.Form["changeTableRelations"].ToString();
    return;
}

//Tabellen Relationen erstellen Schritt 2 (Seite zum angeben der nötigen
Daten anzeigen)
if(Request.Form["changeRelations"].ToString() == "Primär- und
Fremdschlüssel Verbindung hinzufügen") {
    alterInfo = new List<List<string>>();
    //Liste von Infos die zu alterInfo hinzugefügt werden und später
verarbeitet werden
    List<String> relationTargets = new List<string>();
    if(Request.Form["primaryKey"].ToString().Split(',')[1].ToLower() !=
Request.Form["foreignKey"].ToString().Split(',')[1].ToLower()) {
        throw new Exception("Der Primary und Foreign Key müssen den
selben Namen haben");
    }
    if(Request.Form["primaryKey"].ToString() ==
Request.Form["foreignKey"].ToString()) {
        throw new Exception("Der Primary und Foreign Key können nicht
derselbe sein");
    }
    action = Request.Form["changeTableRelations"].ToString().Split(',')[0];
    databaseName =
Request.Form["changeTableRelations"].ToString().Split(',')[1];
    tableName =
Request.Form["changeTableRelations"].ToString().Split(',')[2];
    relationTargets.Add("relations");
    relationTargets.Add(Request.Form["primaryKey"].ToString().Split(',')[0]);
    relationTargets.Add(Request.Form["primaryKey"].ToString().Split(',')[1]);

```

```

        relationTargets.Add(Request.Form["foreignKey"].ToString().Split(',')[0]);
        relationTargets.Add(Request.Form["foreignKey"].ToString().Split(',')[1]);
        alterInfo.Add(relationTargets);
    }

    //Datenbank erstellen Schritt 1 (Seite zum angeben der nötigen Daten
anzeigen)
    if(Request.Form["createDatabase"].ToString() != "") {
        action = Request.Form["createDatabase"].ToString();
        return;
    }

    //Datenbank erstellen Schritt 2
    if(Request.Form["txtNewDataBase"].ToString() != "") {
        if(regex.IsMatch(Request.Form["txtNewDataBase"].ToString())) {
            throw new Exception("Der Datenbankname darf keine Sonderzeichen
enthalten :"+ Request.Form["txtNewDataBase"].ToString());
        }
        action = "createDb";
        databaseName = Request.Form["txtNewDataBase"].ToString();
    }

    //Neue Tabelle erstellen Schritt 1 (Seite zum angeben der nötigen Daten
anzeigen)
    if(Request.Form["createBtn"].ToString() == "Neue Tabelle erstellen") {
        action = Request.Form["createnewTable"].ToString().Split(',')[0];
        databaseName =
Request.Form["createnewTable"].ToString().Split(',')[1];
        currentDatabaseName = databaseName;
        return;
    }

    //Neue Tabelle erstellen Schritt 2
    if(Request.Form["txtNewDataTable"].ToString() != "") {
        if(regex.IsMatch(Request.Form["txtNewDataTable"].ToString())) {
            throw new Exception("Der Tabellennamen darf keine Sonderzeichen
enthalten: "+Request.Form["txtNewDataTable"].ToString());
        }
        tableColNames = new List<string>();
    }

```



```

tableColTypes = new List<Type>();
action = "addtable";
tableName = Request.Form["txtNewDataTable"].ToString();
//Prüfen welche Spalten ausgefüllt sind. Mindestens 1 ist erforderlich.
for(int i = 1; i <= 5; i++) {
    if(Request.Form["columnName" + i].ToString() != "") {
        if(Regex.IsMatch(Request.Form["columnName" + i].ToString())) {
            throw new Exception("Der Spaltenname darf keine
Sonderzeichen enthalten: " + Request.Form["columnName" + i].ToString());
        }
        tableColNames.Add(Request.Form["columnName" + i]);
        if(Request.Form["columnType" + i] == "varchar") {
            tableColTypes.Add("").GetType();
        }
        else if(Request.Form["columnType" + i] == "integer") {
            tableColTypes.Add(0.GetType());
        }
        else if(Request.Form["columnType" + i] == "decimal") {
            tableColTypes.Add(0.0.GetType());
        }
    }
}

//Tabelle ändern Schritt 1 (Seite zum angeben der nötigen Daten anzeigen)
if(Request.Form["alterBtn"].ToString() != "") {
    action = Request.Form["alterTable"].ToString().Split(',')[0];
    databaseName = Request.Form["alterTable"].ToString().Split(',')[1];
    tableName = Request.Form["alterTable"].ToString().Split(',')[2];
    currentDatabaseName = databaseName;
    currentTableName = tableName;
    return;
}

//Tabelle ändern Schritt 2
if(Request.Form["AlterTable"].ToString() != "") {

```

```

alterInfo = new List<List<string>>();
databaseName = Request.Form["alterTable"].ToString().Split(',')[1];
action = Request.Form["alterTable"].ToString().Split(',')[0];
tableName = Request.Form["alterTable"].ToString().Split(',')[2];
if(Request.Form["altertableaddcol"].ToString().ToLower() == "submit") {
    if(Request.Form["newcolname"].ToString() != "") {
        if(Request.Form["newcoltype"].ToString() != "") {
            List<String> tempList = new List<string>();
            tempList.Add("add");
            tempList.Add(Request.Form["newcolname"].ToString());
            tempList.Add(Request.Form["newcoltype"].ToString());
            alterInfo.Add(tempList);
        }
        else {
            throw new Exception("Sie müssen einen Tabellendatentyp
eingeben wenn sie eine neue Spalte erstellen wollen");
        }
    }
    else {
        throw new Exception("Sie müssen einen Tabellennamen eingeben
wenn sie eine neue Spalte erstellen wollen");
    }
}
foreach(DataColumn col in GetTable().table.Columns) {
    //Falls Datentyp geändert werden soll
    if(Request.Form[col.ColumnName + ",Datentyp"].ToString() != "") {
        List<String> tempList = new List<string>();
        tempList.Add("modify");
        tempList.Add(col.DataType.Name.ToString());
        tempList.Add(Request.Form[col.ColumnName +
",Datentyp"].ToString());
        tempList.Add(col.ColumnName.ToString());
        alterInfo.Add(tempList);
    }
    //Falls name geändert werden soll

```

```

        if(Request.Form[col.ColumnName + ",Name"].ToString() != "") {
            List<String> tempList = new List<string>();
            tempList.Add("rename");
            tempList.Add(col.ColumnName.ToString());
            tempList.Add(Request.Form[col.ColumnName +
",Name"].ToString());
            tempList.Add(Request.Form[col.ColumnName +
",Datentyp"].ToString());
            alterInfo.Add(tempList);
        }
    }
}

//Delete Table or Database
if(Request.Form["deleteBtn"].ToString() != "") {
    deleteInfo = new List<string>();
    action = Request.Form["deleteDatabase"].ToString().Split(',')[0];
    deleteInfo.Add(Request.Form["deleteDatabase"].ToString().Split(',')[1]);
    if(currentDatabaseName != null) {
        databaseName = currentDatabaseName;
    }
    if(currentTableName != null) {
        tableName = currentTableName;
    }
    connectionString = $"server = localhost; user = root; password = ;
database ={databaseName}; ";
}

//Filter Databases
if(Request.Form["searchBtn"].ToString() != "") {
    if(Request.Form["searchkeyword"].ToString() == "") {
        throw new Exception("Bitte etwas beim Suchtext eingeben");
    }
    action = Request.Form["searchDatabases"].ToString();
    searchKeyWord = Request.Form["searchkeyword"].ToString();
    searchType = Request.Form["searchType"].ToString();
}

```

```

//Add table entry step 1
if(Request.Form["createBtn"].ToString() == "Neue Reihe hinzufügen") {
    action = Request.Form["createnewtablerow"].ToString().Split(',')[0];
    databaseName =
Request.Form["createnewtablerow"].ToString().Split(',')[1];
    tableName = Request.Form["createnewtablerow"].ToString().Split(',')[2];
    currentTableName = tableName;
    currentDatabaseName = databaseName;
}
//Add table entry step 2
if(Request.Form["createBtn"].ToString() == "Tabellenreihe erstellen") {
    int i = 1;
    if(currentTableName != null) {
        tableName = currentTableName;
    }
    if(currentDatabaseName != null) {
        databaseName = currentDatabaseName;
    }
    List<object> rowAttributes = new List<object>();
    tableRowsToAdd = new List<List<object>>();
    action = "addtableentries";
    foreach(DataColumn col in GetTable().table.Columns) {
        if(ParseString(Request.Form["columnname" + i]).GetType() ==
col.DataType || col.DataType=="").GetType()) {
            rowAttributes.Add(Request.Form["columnname" + i]);
        }
        else {
            throw new Exception("Einer der Datentypen beim Tabellen
hinzufügen hat nicht gepasst: " + col.ColumnName);
        }
        i++;
    }
    tableRowsToAdd.Add(rowAttributes);
}

```

```

        // Erstellen Sie eine neue Datenbank und fügen Sie sie als Objekt zum
        Programm hinzu
        if(action.ToLower() == "createdb") {
            if(DatabaseExistsCheck()) {
                connectionString = $"server = localhost; user = root; password =;
database ={databaseName}; ";
                CreateNewDatabaseFile();
                AddDatabase();
                action = "viewdatabase";
            }
            else {
                return;
            }
        }
        // Hinzufügen einer Tabelle zur bestehenden Datenbank
        else if(action.ToLower() == "addtable" && databases.Count >= 1) {
            if(currentDatabaseName != null) {
                databaseName = currentDatabaseName;
            }
            connectionString = $"server = localhost; user = root; password =;
database ={databaseName}; ";
            AddTable();
            action = "viewdatatable";
        }
        // Hinzufügen eines Tabelleneintrages zu einer bestehenden Tabelle
        else if(action.ToLower() == "addtableentries" && databases.Count >= 1) {
            AddTableEntrys();
            action = "viewdatatable";
        }
        // Durchsuchen einer vorhandenen Datenbank nach Schlüsselwörtern
        else if(action.ToLower() == "searchkeyword" && databases.Count >= 1 &&
searchKeyWord != null) {
            SearchKeyWordInDB();
            action = "showFilterErg";
        }

```

```

//Löschen einer Tabelle oder Datenbank auf der Grundlage von deleteInfo
else if(action.ToLower() == "delete" && databases.Count >= 1) {
    DeleteTableOrDatabase();
    action = "startsite";
}
//Tabellenattribute ändern
else if(action.ToLower() == "altertable" && databases.Count > 0 &&
GetDatabase().tables.Count > 0) {
    AlterTable();
    action = "viewdatatable";
    if(currentDatabaseName != null) {
        databaseName = currentDatabaseName;
    }
    if(currentTableName != null) {
        tableName = currentTableName;
    }
}
databasesBase = databases;
}
catch(Exception ex) {
    if(!errorStackTrace.Contains(ex)) {
        errorStackTrace.Add(ex);
    }
    return;
}
}
//
// Methoden Start
//
//Überprüft, welcher Datentyp eine Zeichenkette ist und gibt die Zeichenkette als
diesen Typ zurück
private object ParseString(string str) {
    Int32 intValue;
    Int64 bigintValue;
    double doubleValue;

```

```

        if(Int32.TryParse(str, out intValue))
            return intValue;
        else if(Int64.TryParse(str, out bigintValue))
            return bigintValue;
        else if(double.TryParse(str, out doubleValue))
            return doubleValue;
        else return str;
    }
    /// <summary>
    /// Schaut ob es bereits Datenbanken in Mysql gibt und l d diese in das Program
    /// </summary>
    /// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
    public Boolean AddAlreadyExistingDbs() {
        try {
            databases = databasesBase;
            this.connectionString = "SERVER=localhost;UID='root';" +
"PASSWORD=";
            using(MySqlConnection connection = new
MySqlConnection(connectionString)) {
                MySqlDataAdapter adapter = new MySqlDataAdapter("SHOW
DATABASES;", connection);
                DataTable table = new DataTable();
                adapter.Fill(table);
                String[] systemDbNames = { "information_schema", "mysql",
"performance_schema", "phpmyadmin" };
                foreach(DataRow row in table.Rows) {
                    if(!systemDbNames.Contains(row[table.Columns[0]].ToString())) {
                        this.connectionString = $"server = localhost; user = root; password
=; database ={row[table.Columns[0]]}; ";
                        databaseName = row[table.Columns[0]].ToString();
                        Boolean dbExists = false;
                        foreach(Database db in databases) {
                            if(databaseName == db.databaseName) {
                                dbExists = true;
                            }
                        }
                    }
                }
            }
        }
    }

```

```

        }
        if(dbExists) {
            continue;
        }
        AddDatabase();
    }
}
return true;
}
catch(Exception ex) {
    errorStackTrace.Add(ex);
    throw;
}
}
/// <summary>
/// Schaut ob die Datenbank bereits existiert mit einer Schleife und einem
Vergleich
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean DatabaseExistsCheck() {
    try {
        foreach(Database database in databases) {
            if(database.databaseName == databaseName) {
                throw new Exception("Database with the same name is already
loaded in the Program");
            }
        }
        return true;
    }
    catch(Exception ex) {
        errorStackTrace.Add(ex);
        return false;
    }
}
}

```



```

/// <summary>
/// Erstellung einer neuen, leeren MySQL-Datenbank, wenn dies erforderlich ist
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean CreateNewDatabaseFile() {
    try {
        string connectionString = $"server=localhost;uid=root;pwd=";
        string databaseName = this.databaseName;
        using(MySqlConnection connection = new
MySQLConnection(connectionString)) {
            connection.Open();

            MySqlCommand command = new MySqlCommand($"CREATE
DATABASE {databaseName};", connection);
            command.ExecuteNonQuery();

            this.connectionString = connectionString + "database=" +
databaseName + ";";
        }
        return true;
    }
    catch(Exception ex) {
        errorStackTrace.Add(ex);
        return false;
    }
}
/// <summary>
/// Füllt alle Tabellen einer Datenbank in ein Dataset, um später damit zu
arbeiten
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean AddDatabase() {
    try {
        DataSet database = new DataSet();
        DataTable tableConstraints = new DataTable();
        DataTable tableNames = new DataTable();

```

```

        using(MySqlConnection connectionDB = new
        MySqlConnection(connectionString)) {
            connectionDB.Open();
            //Datenbankname
            database.DataSetName = connectionDB.Database;
            //Tabellennamen mit Sql-Abfrage abrufen
            MySqlCommand command = new MySqlCommand("Show tables;",
            connectionDB);
            MySqlDataAdapter adapter = new MySqlDataAdapter(command);
            adapter.Fill(tableNames);
            if(tableNames == null) {
                throw new Exception("There was an issue with the tablename when
            importing the " + connectionDB.Database + " Database");
            }
            //Abrufen von Beziehungen aus Tabellen in der Datenbank mit Sql-
            Befehl (Quelle ist im Dokument)
            command = new MySqlCommand("SELECT\n\r" +
                " `TABLE_NAME`,                -- Foreign key
            table\n\r" +
                " `COLUMN_NAME`,                -- Foreign key
            column\n\r" +
                " `REFERENCED_TABLE_NAME`,      -- Origin
            key table\n\r" +
                " `REFERENCED_COLUMN_NAME`      --
            Origin key column\n\r" +
                "FROM\n\r" +
                "
            `INFORMATION_SCHEMA`.`KEY_COLUMN_USAGE` -- Will fail if user don't have
            privilege\n\r" +
                "WHERE\n\r" +
                "$" `CONSTRAINT_SCHEMA`
            ='{connectionDB.Database}'\n\r"
                , connectionDB);
            adapter = new MySqlDataAdapter(command);
            adapter.Fill(tableConstraints);
            // Holt jede Tabelle mit Select * aus der Datenbank und speichern Sie sie
            alle in einem Dataset.

```

```

        foreach(DataRow rowZero in tableNames.Rows) {
            String tableName = rowZero.ItemArray[0].ToString();
            string query = "SELECT * FROM " + tableName.ToString();
            command = new MySqlCommand(query, connectionDB);
            adapter.SelectCommand = command;
            DataTable dataTable = new DataTable(tableName.ToString());
            adapter.Fill(dataTable);
            database.Tables.Add(dataTable);
        }

        // Datenbankobjekt wird mit dem Datensatz erstellt und zur Liste der
        // Datenbanken hinzugefügt
        Database db = new Database(database, tableConstraints);
        databases.Add(db);
    }

    //Bei Erfolg wird true zurückgegeben
    return true;
}

catch(Exception ex) {
    //Bei Fehler wird false zurückgegeben und die Ausnahme zum
    //ErrorStackTrace hinzugefügt
    errorStackTrace.Add(ex);
    return false;
}
}

/// <summary>
/// Sucht nach Schlüsselwörtern in verschiedenen Stilen, je nachdem, was der
/// Benutzer suchen möchte:
/// Tabellennamen, Spaltennamen oder Datenzellen
/// Wenn der Suchtyp Daten ist, werden auch die Tabellen- und Spaltennamen
/// durchsucht.
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean SearchKeywordInDB() {
    try {
        Boolean erg = false;

```

```

int rowPosition = 1;
foreach(Database database in databases) {
    if(database.databaseName == databaseName) {
        foreach(Table table in database.tables) {
            if(searchType == "table" || searchType == "data") {

if(table.table.TableName.ToLower().Contains(searchKeyWord.ToLower())) {
                erg = true;
                searchFoundValues.Add(database.databaseName + " " +
table.table.TableName);
            }
        }
        if(searchType == "column" || searchType == "data") {
            foreach(DataColumn column in table.table.Columns) {

if(column.ColumnName.ToLower().Contains(searchKeyWord.ToLower())) {
                erg = true;
                searchFoundValues.Add(database.databaseName + " " +
table.table.TableName + " " + column.ColumnName);
            }
        }
    }
    if(searchType == "data") {
        foreach(DataRow row in table.table.Rows) {
            foreach(DataColumn column in table.table.Columns) {

if(row[column].ToString().ToLower().Contains(searchKeyWord.ToLower())) {
                erg = true;
                searchFoundValues.Add(database.databaseName + " " +
table.table.TableName + " " + column.ColumnName + " " + "Reihe:" + rowPosition);
            }
        }
        rowPosition++;
    }
}
}
}

```

```

        }
    }
    return erg;
}

catch(Exception ex) {
    errorStackTrace.Add(ex);
    return false;
}
}

/// <summary>
/// Fügt eine Tabelle zu einer bestehenden Datenbank hinzu
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean AddTable() {
    Boolean erg = false;
    try {
        foreach(Database database in databases) {
            if(database.databaseName == databaseName) {
                //Sobald die passende Db gefunden wurde, fügen wir eine Tabelle
                hinzu
                DataTable table = new DataTable(tableName);
                for(int i = 0; i < tableColNames.Count; i++) {
                    table.Columns.Add(tableColNames[i], tableColTypes[i]);
                }
                database.AddTable(table);
                database.dataSet.AcceptChanges();
                erg = true;
                using(MySqlConnection connectionDB = new
                MySqlConnection(connectionString)) {
                    connectionDB.Open();
                    String query = $"CREATE TABLE {tableName} (";
                    for(int i = 0; i < tableColNames.Count; i++) {
                        query += $"`{tableColNames[i]}` {tableColTypes[i].Name}, ";
                    }
                    query = query.TrimEnd(', ');

```

```

        query += ");";
        query = query.Replace("String", "VarChar(25)").Replace("Int32",
"Int").Replace("Double", "Decimal");
        MySqlCommand command = new MySqlCommand(query,
connectionDB);
        command.ExecuteNonQuery();
    }
}
}
return erg;
}
catch(Exception ex) {
    errorStackTrace.Add(ex);
    return false;
}
}

/// <summary>
/// Tabelleneinträge, die sich in der tableRowsToAdd befinden, werden der
aktuell verwendeten Tabelle hinzugefügt
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean AddTableEntrys() {
    int counter = 0;
    try {
        this.connectionString = $"server = localhost; user = root; password =;
database ={databaseName}; ";
        Table table = GetTable();
        foreach(List<Object> tableRow in tableRowsToAdd) {
            DataRow row = table.table.NewRow();
            foreach(DataColumn col in table.table.Columns) {
                row[col] = ParseString(tableRow[counter].ToString());
                counter++;
            }
            table.table.Rows.Add(row);
            counter = 0;
        }
    }
}

```

```

        using(MySqlConnection connectionDB = new
        MySqlConnection(connectionString)) {
            connectionDB.Open();
            String query = $"INSERT INTO {tableName} (";
            for(int i = 0; i < table.table.Columns.Count; i++) {
                query += $"`{table.table.Columns[i].ColumnName}`,";
            }
            query = query.TrimEnd(',');
            query += ")";
            query += " VALUES (";
            for(int i = 0; i < table.table.Columns.Count; i++) {
                if(table.table.Rows[table.table.Rows.Count -
1].ItemArray[i]?.GetType() == typeof(String)) {
                    query += $"`{table.table.Rows[table.table.Rows.Count -
1].ItemArray[i]}`,";
                }
                else {
                    query += $"{table.table.Rows[table.table.Rows.Count -
1].ItemArray[i]},";
                }
            }
            query = query.TrimEnd(',');
            query += ")";
            MySqlCommand command = new MySqlCommand(query,
connectionDB);
            command.ExecuteNonQuery();
        }
    }
    return true;
}
catch(Exception ex) {
    errorStackTrace.Add(ex);
    return false;
}
}

```

```

    /// <summary>
    /// Ruft das Tabellenobjekt basierend auf der aktuell verwendeten Datenbank
    und der aktuell verwendeten Tabelle ab
    /// </summary>
    /// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
    public Table GetTable() {
        try {
            foreach(Database database in databases) {
                if(database.databaseName == databaseName) {
                    //Sobald die passende Db gefunden wurde, beginnen wir mit der
    Suche in der Tabelle
                    foreach(Table table in database.tables) {
                        if(table.tableName == tableName) {
                            //Sobald die passende Tabelle gefunden wurde, wird sie
    zurückgegeben
                            return table;
                        }
                    }
                }
            }
            throw new Exception("Die Tabelle wurde nicht gefunden in GetTable()");
        }
        catch(Exception ex) {
            errorStackTrace.Add(ex);
            throw;
        }
    }
    /// <summary>
    /// Ruft das Datenbankobjekt basierend auf der aktuell verwendeten Datenbank
    ab
    /// </summary>
    /// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
    public Database GetDatabase() {
        try {
            foreach(Database database in databases) {

```



```

        if(database.databaseName == databaseName) {
            //Wenn die passende Db gefunden wurde, fügen wir eine Tabelle
hinzu
            return database;
        }
    }
    throw new Exception("Die Datenbank wurde nicht gefunden in
GetDatabase()");
}
catch(Exception ex) {
    errorStackTrace.Add(ex);
    throw;
}
}

/// <summary>
/// Die ausgewählte Tabelle oder Datenbank, die der Benutzer zum Löschen
ausgewählt hat, wird im Datenbank- und Tabellenobjekt gelöscht.
/// Sie wird auch im Dataset-Objekt gelöscht, oder im Fall von Datenbanken wird
der gesamte Dataset gelöscht.
/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean DeleteTableOrDatabase() {
    try {
        using(MySqlConnection connection = new
MySQLConnection(connectionString)) {
            connection.Open();
            if(deleteInfo[0].ToLower() == "table") {
                Database database = GetDatabase();
                Table table = GetTable();
                if(table != null) {
                    database.dataSet.Tables.Remove(table.table);
                    database.tables.Remove(table);
                    database.dataSet.AcceptChanges();
                    MySqlCommand command = new MySqlCommand($"DROP
TABLE {tableName};", connection);
                    command.ExecuteNonQuery();

```

```

        }
        else {
            throw new Exception($"Die Tabelle {tableName} von der
Datenbank {databaseName} konnte nicht gelöscht werden, da sie nicht gefunden
wurde.");
        }
    }
    else if(deleteInfo[0].ToLower() == "database") {
        Database database = GetDatabase();
        if(database != null) {
            databases?.Remove(database);
            database.dataSet.Clear();
            database.dataSet = null;
            database = null;

            MySqlCommand command = new MySqlCommand($"DROP
DATABASE {databaseName};", connection);
            command.ExecuteNonQuery();
        }
        else {
            throw new Exception($"Die Datenbank {databaseName} konnte
nicht gelöscht werden, da sie nicht gefunden wurde.");
        }
    }
}
return true;
}
catch(Exception ex) {
    errorStackTrace.Add(ex);
    return false;
}
}

/// <summary>
/// Die Tabelle wird entsprechend der Auswahl des Benutzers geändert.
/// Merkmale: Ändern von Spaltennamen und Attributen und Ändern der
Beziehungen zwischen Tabellen

```

```

/// </summary>
/// <returns>True, wenn erfolgreich, false, wenn erfolglos</returns>
private Boolean AlterTable() {
    try {
        connectionString = $"server = localhost; user = root; password =; database
={databaseName}; ";
        using(MySqlConnection connection = new
        MySqlConnection(connectionString)) {
            connection.Open();
            //Columnname ändern
            foreach(List<String> list in alterInfo) {
                String query = $"ALTER TABLE {tableName} ";
                if(list[0] == "rename") {
                    query += $"CHANGE `{list[1]}` `{list[2]}` {list[3]}";
                    query = query.Replace("String", "varChar(25)").Replace("Int32",
                    "Int").Replace("Double", "Decimal");
                }
                else if(list[0] == "relations") {
                    query = $"ALTER TABLE {list[1]} ";
                    query += $"ADD CONSTRAINT `PK_{list[1]}` PRIMARY KEY({
list[2]}); ";
                    query += $"ALTER TABLE {list[3]} ";
                    query += $"ADD CONSTRAINT `FK_{list[3]}` FOREIGN
KEY({list[4]}) REFERENCES `{list[1]}` ({list[2]}); ";
                }
                //Columntyp ändern
                else if(list[0] == "modify") {
                    query += $"CHANGE COLUMN `{list[3]}` `{list[3]}` {list[2]}";
                    query = query.Replace("String", "varChar(25)").Replace("Int32",
                    "Int").Replace("Double", "Decimal");
                }
                //Column hinzufügen
                else if(list[0] == "add") {
                    query += $"ADD `{list[1]}` {list[2]}";
                    query = query.Replace("String", "varChar(25)").Replace("Int32",
                    "Int").Replace("Double", "Decimal");
                }
            }
        }
    }
}

```

```

        }
        MySqlCommand command = new MySqlCommand(query,
connection);
        command.ExecuteNonQuery();
        Thread.Sleep(5);
    }
}
databases.Clear();
databasesBase.Clear();
AddAlreadyExistingDbs();
return true;
}
catch(Exception ex) {
    errorStackTrace.Add(ex);
    throw;
}
}
//
// Methods End
//
}
}

```

## 13.2 Index.cshtml

```
@page
@model IndexModel
@using System.Data;
@{
    ViewData["Title"] = "Datenbank Management System";
}
<script>
    document.getElementById("submitter").disabled = false;
    function clicked(e)
    {
        if(!confirm('Sind sie sicher, dass sie den Löschvorgang durchführen wollen?')) {
            e.preventDefault();
        }
    }
</script>
<div class="content">
    <div class="sidebar">
        <div class="logo">
            <i class="bx bx-menu menu-icon"></i>
            <span class="logo-name"><h3>Importierte Datenbanken</h3><br /><br
/></span>
        </div>
        <div class="sidebar-content">
            @{
                //Die Datenbanken, die bereits in mysql vorhanden sind, werden hier
                geladen, falls es welche gibt
                if(Model.databases == null) {
                    Model.action = "addalreadyexistingdbs";
                    Model.OnPost();
                    Model.action = "startsite";
                }
                //Wenn vorhandene DB gefunden wurden, werden sie hier in der
                Seitenleiste angezeigt
            }
        </div>
    </div>
</div>
```

```

if(Model.databases?.Count > 0) {
    foreach(Database database in @Model.databases) {
        <text>
            <form method="post">
                <i class="bx bx-home-alt icon"></i>
                <h6><input type="submit" class="boldSubmit"
value="@database.databaseName"></h6>
                <input type="hidden" name="navdatabase"
value="viewDatabase,@database.databaseName">
            </form>
        </text>
        foreach(Table datatable in database.tables) {
            <text>
                <form method="post">
                    <i class="bx bx-home-alt icon"></i>
                    &emsp;<input type="submit" value="@datatable.tableName"
/>
                    <input type="hidden" name="navdatatable"
value="viewDatatable,@database.databaseName,@datatable.tableName">
                </form>
            </text>
        }
        if(database.tables.Count > 0) {
            <text><br /></text>
        }
    }
    <text>
        <form method="post">
            <input type="submit" class="boldSubmit" name="createBtn"
value="Neue Datenbank erstellen" />
            <input type="hidden" name="createDatabase" value="create"
/><br>
        </form>
    </text>
}

```

```

        //Wenn keine DBs vorhanden sind, wird dies hier angezeigt und es ist
        möglich eine DB erstellen
        else {
            <form method="post">
                Im Moment sind noch keine Datenbanken in mySQL<br /> <br
/><br />
                <input type="submit" name="createBtn" class="boldSubmit"
value="Neue Datenbank erstellen" />
                <input type="hidden" name="createDatabase" value="create" />
            </form>
        }
    }
</div>
</div>
<div class="text-center">
    <form method="post">
        @{
            //Hier wird das Inhaltsfenster aufgebaut, je nachdem, was der Benutzer
sieht

            //Wenn ein Fehler auftritt, wird dieses Inhaltsfenster angezeigt
            if(Model.errorStackTrace != null) {
                if(Model.errorStackTrace.Count > 0) {
                    <text>
                        <br><br><br>
                        <h1>Folgende Fehler sind passiert:</h1> <br>
                    </text>
                    foreach(Exception ex in Model.errorStackTrace) {
                        <text>
                            @ex.Message <br>
                        </text>
                    }
                    Model.action = "startseite";
                }
            }
        }
        //Startseite

```

```

if(@Model.action.ToLower() == "startsite") {
    <text>
        <br><br><br><br>
        <h1>Startseite</h1>
        <br><br><br><br>
        <h3>
            Willkommen zu meinem Datenmodifikationsprogramm<br />
            Author: Benjamin Born
        </h3>
    </text>
}
//Neue Datenbank erstellen
else if(@Model.action.ToLower() == "create") {
    <text>
        <h1>Neue Datenbank erstellen</h1><br>
        Datenbankname: <input type="text" name="txtNewDataBase"
required>

        <br><br>
        <input type="submit" class="boldSubmit" value="Submit">
    </text>
}
//Neue Tabelle erstellen
else if(@Model.action.ToLower() == "createnewtable") {
    <text>
        <h1>Neue Tabelle erstellen</h1><br>
        Tabellename: <input type="text" name="txtNewDataTable"
required>

        <br><br>
        Spalte 1:<br>
        Spaltenname: <input type="text" name="columnName1" required>
        Spaltentyp:
        <select name="columnType1" required>
            <option value="varchar">VarChar</option>
            <option value="integer">Integer</option>
            <option value="decimal">Decimal</option>

```



```

        </select>
        <br><br>
    </text>
    for(int i = 2; i <= 5; i++) {
        <text>
        Spalte @i:<br>
        Spaltenname: <input type="text" name="columnName@(i)">
Spaltentyp:
        <select name="columnType@(i)">
            <option value="varchar">VarChar</option>
            <option value="integer">Integer</option>
            <option value="decimal">Decimal</option>
        </select>
        <br><br>
        </text>
    }
    <text><br /><input type="submit" class="boldSubmit"
value="Submit"></text>
}
//Tabelle ändern
else if(@Model.action.ToLower() == "altertable") {
    <h1>Spalten der Tabelle @Model.GetTable().tableName
ändern</h1><br />
    @:<table>
    @:<tr>
        foreach(DataColumn col in @Model.GetTable().table.Columns) {
            @:<th>
            Boolean added = false;
            foreach(DataRow row in
@Model.GetDatabase().tableConstraints.Rows){
                if (row[1].ToString()==@col.ColumnName &&
row[0].ToString()== @Model.GetTable().tableName){
                    if(row[2].ToString()==" && row[3].ToString()==""){
                        @:[PRK]@col.ColumnName.ToString()
                        added = true;

```

```

        break;
    }
    else{
        @:[FRK]@col.ColumnName.ToString()
        added = true;
        break;
    }
}
}
if(!added){
    @col.ColumnName.ToString()
}
@:</th>
}
@:</tr>
@:<tr>
foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <td>
        @col.DataType.Name
    </td>
}
@:</tr>
@:<tr>
@:<th colspan="@Model.GetTable().table.Columns.Count">Neue
Attribute:</th>
@:</tr>
@:<tr>
foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <th>
        Name
    </th>
}
@:</tr>
@:<tr>

```

```

foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <th>
    <input type="text" name="@col.ColumnName.ToString(),Name">
    </th>
}
@:</tr>
@:<tr>
foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <th>
    Datentyp
    </th>
}
@:</tr>
@:<tr>
foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <td>
    <select name="@col.ColumnName.ToString(),Datentyp">
        <option value=""></option>
        <option value="String">String</option>
        <option value="Int32">Int32</option>
        <option value="Double">Double</option>
    </select>
    </td>
}
@:</tr>
@:</table>
<text>

<br /><input type="submit" class="boldSubmit" value="Submit">
<input type="hidden" name="altertable"
value="altertable,@Model.GetDatabase().databaseName,@Model.GetTable().tableName">

<br><hr>
<h1>Neue Spalte in der Tabelle @Model.GetTable().tableName
erstellen</h1>

```

```

        <br>
        Name: <input type="text" name="newColName">
        Datentyp: <select name="newColType">
            <option value=""></option>
            <option value="String">String</option>
            <option value="Int32">Int32</option>
            <option value="Double">Double</option>
        </select>
        <br><br>
        <input type="submit" class="boldSubmit" name="altertableaddcol"
value="Submit" />
        <input type="hidden" name="createnewcol" value="createnewcol"
/>

        <hr>
        (Sie können nur eine der beiden Aktionen auf einmal aufführen)
        </text>
    }
    // Neue Tabellenreihe hinzufügen
    else if(@Model.action.ToLower() == "addnewtablerow") {
        <h1>Neue Tabellenreihe hinzufügen</h1>
        @:<table>
        @:<tr>
            foreach(DataColumn col in @Model.GetTable().table.Columns) {
                @:<th>
                Boolean added = false;
                foreach(DataRow row in
@Model.GetDatabase().tableConstraints.Rows){
                    if (row[1].ToString()==@col.ColumnName &&
row[0].ToString()== @Model.GetTable().tableName){
                        if(row[2].ToString()==" " && row[3].ToString()==""){
                            @:[PRK]@col.ColumnName.ToString()
                            added = true;
                            break;
                        }
                    }
                }
            }
        }
    }
    else{

```

```

        @:[FRK]@col.ColumnName.ToString()
        added = true;
        break;
    }
}
}
if(!added){
    @col.ColumnName.ToString()
}
@:</th>
}
@:</tr>
@:<tr>
foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <td>
        @col.DataType.Name
    </td>
}
@:</tr>
@:<tr>
int i = 1;
foreach(DataColumn col in @Model.GetTable().table.Columns) {
    <td><input type="text" width="20" name="columnName@(i)"
required></td>
    i++;
}
@:</tr>
@:</table>
<text>
    <br /><input type="submit" class="boldSubmit" name="createBtn"
value="Tabellenreihe erstellen" />
</text>
}
//Datenbanken anzeigen
else if(@Model.action.ToLower() == "viewdatabase") {

```

```

@:<br><h1>Datenbank:
@Model.GetDatabase().databaseName<br>Tabellen:<br></h1><br>
@:<table class="viewTables">
    foreach(Table table in @Model.GetDatabase().tables) {
        <tr>
            <td>
                <input type="submit" name="viewTableNames"
value="@table.tableName" />
                <input type="hidden" name="navdatatable"
value="viewDatatable,@Model.GetDatabase().databaseName,@table.tableName">
            </td>
        </tr>
    }
@:</table>
<text>
<br>
    <input type="submit" class="boldSubmit" name="createBtn"
value="Neue Tabelle erstellen" />
    <input type="hidden" name="createnewTable"
value="createnewtable,@Model.GetDatabase().databaseName" />
    <br><hr>
Datenbank Filtern:
Suchart: <select name="searchType">
    <option value="table">Table</option>
    <option value="data">Data</option>
    <option value="column">Column</option>
</select>
<br /><br>Suchtext: <input type="text" name="searchkeyword">
    <input type="hidden" name="searchDatabases"
value="searchkeyword" />
    <br><br /><input type="submit" class="boldSubmit"
name="searchBtn" value="Datenbank filtern" />
    <br><hr>
    <input type="submit" class="boldSubmit" name="deleteBtn"
value="Datenbank löschen" onclick="clicked(event)" />
    <input type="hidden" name="deleteDatabase"
value="delete,database,@Model.GetDatabase().databaseName" />

```

```

        <br><hr>
        <input type="submit" class="boldSubmit" name="changeRelationsBtn"
value="Relationen der Tabellen ändern" />
        <input type="hidden" name="changeTableRelations"
value="changeRelations" />
        <br><hr>
    </text>
}
//Tabelle anzeigen
else if (@Model.action.ToLower() == "viewdatatable") {
    @:<br><h1>Tabelle: @Model.tableName</h1><br>
    @:<table>
    @:<tr>
        //Alle Kopfzeilen der Tabelle anzeigen
        foreach(DataColumn col in @Model.GetTable().table.Columns) {
            @:<th>
            Boolean added = false;
            foreach(DataRow row in
@Model.GetDatabase().tableConstraints.Rows){
                if (row[1].ToString()==@col.ColumnName &&
row[0].ToString()==@Model.GetTable().tableName){
                    if(row[2].ToString()==" && row[3].ToString()==""){
                        @:[PRK]@col.ColumnName.ToString()
                        added = true;
                        break;
                    }
                }
            }
            else{
                @:[FRK]@col.ColumnName.ToString()
                added = true;
                break;
            }
        }
    }
    if(!added){
        @col.ColumnName.ToString()
    }
}

```

```

    }
    @:</th>
}
@:</tr>
foreach(DataRow row in @Model.GetTable().table.Rows) {
    @:<tr>
        //Alle Zellen der Tabelle anzeigen
        foreach(DataColumn col in @Model.GetTable().table.Columns) {
            <td>
                @row[col].ToString()
            </td>
        }
    @:</tr>
}
@:</table>
//Funktionen nach der Anzeige
<text>
    <br><hr>
    <input type="submit" class="boldSubmit" name="createBtn"
value="Neue Reihe hinzufügen" />
    <input type="hidden" name="createnewtablerow"
value="addnewtablerow,@Model.GetDatabase().databaseName,@Model.GetTable()
.tableName" />
    <br><hr>
    <input type="submit" class="boldSubmit" name="alterBtn"
value="Tabellenattribute ändern" />
    <input type="hidden" name="alterTable"
value="altertable,@Model.GetDatabase().databaseName,@Model.GetTable().tabl
eName" />
    <input type="submit" class="boldSubmit" name="deleteBtn"
value="Tabelle löschen" onclick="clicked(event)"/>
    <input type="hidden" name="deleteDatabase"
value="delete,table,@Model.GetDatabase().databaseName,@Model.GetTable().tabl
eName" />
    <br><hr>
</text>

```



```

    }
    //Filter Ergebnisse anzeigen
    else if (@Model.action.ToLower() == "showfiltererg") {
        if (@Model.searchFoundValues.Count==0){
            <text>
            <br />Es wurden keine Ergebnisse bei der Suche gefunden
            <br> <a class="link" asp-area="" asp-page="/Index">Zur
Startseite</a>
            </text>
        }
        else{
            @:<table >
            @:Folgende Ergebnisse wurden bei der Suche nach
"@Model.searchKeyWord" gefunden:
            foreach(String str in @Model.searchFoundValues) {
                @:<tr>
                <th>Datenbankname</th>
                <th>Tabellenname</th>
                <th>Spaltenname</th>
                <th>Reihenposition</th>
                @:</tr>
                @:<tr>
                foreach(String strInner in str.Split(' ')) {
                    <td>
                    @strInner
                    </td>
                }
                @:</tr>
            }
            @:</table >
        }
    }
    else if (@Model.action.ToLower() == "changerelations") {
        <h1>Relationen in der Datenbank
@Model.GetDatabase().databaseName hinzufügen</h1>

```

```

@:<table>
foreach(Table table in @Model.GetDatabase().tables) {
    <text>
    <tr>
    <th>
        @table.tableName
    </th>
    </text>
    foreach(DataColumn col in table.table.Columns) {
        @:<td>
        Boolean added = false;
        foreach(DataRow row in
@Model.GetDatabase().tableConstraints.Rows){
            if (row[1].ToString()==@col.ColumnName &&
row[0].ToString()== table.tableName){
                if(row[2].ToString()==" && row[3].ToString()==""){
                    @:[PRK]@col.ColumnName.ToString()
                    added = true;
                    break;
                }
            }
            else{
                @:[FRK]@col.ColumnName.ToString()
                added = true;
                break;
            }
        }
        if(!added){
            @col.ColumnName.ToString()
        }
        @:</td>
    }
    @:</tr>
}
@:</table>

```

```

@:<table>
  @:<tr><th>Primary Key</th><th>Foreign Key</th></tr>
  @:<br /><br />
  @:<tr><td><select name="PrimaryKey">
foreach(Table table in @Model.GetDatabase().tables) {
  foreach(DataColumn col in table.table.Columns) {
    <option
value="@table.tableName, @col.ColumnName">[@table.tableName]@col.ColumnName</option>
  }
}
  @:</select></td>
  @:<td><select name="ForeignKey">
foreach(Table table in @Model.GetDatabase().tables) {
  foreach(DataColumn col in table.table.Columns) {
    <option
value="@table.tableName, @col.ColumnName">[@table.tableName]@col.ColumnName</option>
  }
}
  @:</select></td></tr>
  @:</table>
  <br><input type="submit" class="boldSubmit"
name="changeRelations" value="Primär- und Fremdschlüssel Verbindung
hinzufügen" />
  <input type="hidden" name="changeTableRelations"
value="altertable, @Model.databaseName, @Model.tableName" />
  }
}
</form>
</div>
</div>

```

### 13.3 Database.cs

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.Data;
namespace DatenbankManagementSystem {
    /// <summary>
    /// Database Object that works with the DataSet it gets when it is created.
    /// It also creates Table Objects with the DataTables from the DataSet
    /// </summary>
    public class Database {
        //Original DataSet
        public DataSet dataSet { get; set; }
        //List of Table objects
        public List<Table> tables { get; set; }
        //Name of the database
        public string databaseName{get;set;}
        //Table of the relationships in the Database between the different tables
        public DataTable tableConstraints { get; set; }
        public Database(DataSet database, DataTable tableConstraints) {
            this.tableConstraints = tableConstraints;
            this.dataSet = database;
            this.databaseName = database.DataSetName;
            if(tables == null) {
                tables = new List<Table>();
            }
            MapTables();
        }
        /// <summary>
        /// Adding new table objects from the dataset after reading the database
        /// </summary>
        private void MapTables() {
            try {
                foreach(DataTable table in dataSet.Tables) {
                    tables.Add(new Table(table,table.TableName));
                }
            }
        }
    }
}
```

```

        }
    }
    catch(Exception) {
        throw;
    }
}
/// <summary>
/// Add table as object and the table to the original dataset
/// </summary>
/// <param name="table"></param>
/// <returns></returns>
public Boolean AddTable(DataTable table) {
    tables.Add(new Table(table,table.TableName));
    dataSet.Tables.Add(table);
    return true;
}
}
}

```

## 13.4 Table.cs

```

using System.Data;
namespace DatenbankManagementSystem {
    public class Table {
        public DataTable table { get; set; }
        public string tableName { get; set; }
        public Table(DataTable table, string tableName) {
            this.table = table;
            this.tableName = tableName;
        }
    }
}

```

## 13.5 \_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - DatenbankManagementSystem</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="stylesheet" href="~/DatenbankManagementSystem.styles.css" asp-
append-version="true" />
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-
white border-bottom box-shadow mb-3">
      <div class="container">
        <a class="navbar-brand" asp-area="" asp-page="/Index"></a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="navbar-collapse collapse d-sm-inline-flex justify-content-
between">
          <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">
              <a class="nav-link text-dark" asp-area="" asp-
page="/Index">Startseite</a>
            </li>
          </ul>
        </div>
      </div>
      <div class="logo">
```

```

        <i class="bx bx-menu menu-icon"></i>
        <span class="logo-name"></span>
    </div>

</nav>
</header>

<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>

<footer class="border-top footer text-muted">
    <div class="container">
        &copy; 2023 - DatenbankManagementSystem Author: Benjamin Born - <a
asp-area="" asp-page="/Datenschutz">Datenschutz</a> - <a asp-area="" asp-
page="/Impressum">Impressum</a>
    </div>
</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

## 13.6 site.css

```
html {  
  font-size: 14px;  
}  
@media (min-width: 768px) {  
  html {  
    font-size: 16px;  
  }  
}  
html {  
  position: relative;  
  min-height: 100%;  
}  
body {  
  margin-bottom: 60px;  
}  
.content {  
  display: grid;  
  grid-template-columns: 18% 85%;  
  grid-template-rows: 100%;  
}  
.sidebar {  
  grid-column-start: 1;  
  border: none;  
  padding: 10px;  
  font-size: 12px;  
}  
.text-center {  
  grid-column-start: 2;  
}  
table, th, td {  
  border: 1px solid black;  
}
```



```

table {
    margin-left: auto;
    margin-right: auto;
}
.link{
    color:aqua;
}
.boldSubmit{
    font-weight:bold;
}
.viewTables {
    margin: auto;
    border: 1px solid black;
    border-collapse: collapse;
    background-color:silver;
}

```

## 13.7 Impressum.cshtml

```

@page
@model ImpressumModel
@{
    ViewData["Title"] = "Impressum";
}
<h1>@ViewData["Title"]</h1>
<div class='impressum'>
    <p>Angaben gemäß § 5 TMG</p><p>
        Hans Gerber <br>
        Musterstr 12<br>
        85746 Hausen <br>
    </p><p>
        <strong>Vertreten durch: </strong><br>
        Hans Gerber<br>
    </p><p>

```

**Haftungsausschluss:**  
**Datenschutz**

Die Nutzung unserer Webseite ist in der Regel ohne Angabe personenbezogener Daten möglich. Soweit auf unseren Seiten personenbezogene Daten (beispielsweise Name, Anschrift oder eMail-Adressen) erhoben werden, erfolgt dies, soweit möglich, stets auf freiwilliger Basis. Diese Daten werden ohne Ihre ausdrückliche Zustimmung nicht an Dritte weitergegeben.

Wir weisen darauf hin, dass die Datenübertragung im Internet (z.B. bei der Kommunikation per E-Mail) Sicherheitslücken aufweisen kann. Ein lückenloser Schutz der Daten vor dem Zugriff durch Dritte ist nicht möglich.

Der Nutzung von im Rahmen der Impressumspflicht veröffentlichten Kontaktdaten durch Dritte zur Übersendung von nicht ausdrücklich angeforderter Werbung und Informationsmaterialien wird hiermit ausdrücklich widersprochen. Die Betreiber der Seiten behalten sich ausdrücklich rechtliche Schritte im Falle der unverlangten Zusendung von Werbeinformationen, etwa durch Spam-Mails, vor.

Website Impressum von [impressum-generator.de](https://www.impressum-generator.de)

## 13.8 Datenschutz.cshtml

```
@page
@model DatenschutzModel
@{
    ViewData["Title"] = "Datenschutzerklärung";
}
<h1>@ViewData["Title"]</h1>
<h2 id="m716">Präambel</h2>
<p>Mit der folgenden Datenschutzerklärung möchten wir Sie darüber aufklären, welche Arten Ihrer personenbezogenen Daten (nachfolgend auch kurz als "Daten" bezeichnet) wir zu welchen Zwecken und in welchem Umfang verarbeiten. Die Datenschutzerklärung gilt für alle von uns durchgeführten Verarbeitungen personenbezogener Daten, sowohl im Rahmen der Erbringung unserer Leistungen als auch insbesondere auf unseren Webseiten, in mobilen Applikationen sowie innerhalb externer Onlinepräsenzen, wie z.B. unserer Social-Media-Profile (nachfolgend zusammenfassend bezeichnet als "Onlineangebot").</p>
<p>Die verwendeten Begriffe sind nicht geschlechtsspezifisch.</p>
<p>Stand: 12. Juli 2023</p>
<h2>Inhaltsübersicht</h2>
```

- [Präambel](#m716)
- [Verantwortlicher](#m3)
- [Übersicht der Verarbeitungen](#mOverview)
- [Maßgebliche Rechtsgrundlagen](#m2427)
- [Sicherheitsmaßnahmen](#m27)
- [Übermittlung von personenbezogenen Daten](#m25)
- [Internationale Datentransfers](#m24)
- [Löschung von Daten](#m12)
- [Rechte der betroffenen Personen](#m10)
- [Einsatz von Cookies](#m134)
- [Bereitstellung des Onlineangebotes und Webhosting](#m225)
- [Kontakt- und Anfragenverwaltung](#m182)
- [Änderung und Aktualisierung der Datenschutzerklärung](#m15)
- [Begriffsdefinitionen](#m42)

## Verantwortlicher

## Maßgebliche Rechtsgrundlagen

**Maßgebliche Rechtsgrundlagen nach der DSGVO:** Im Folgenden erhalten Sie eine Übersicht der Rechtsgrundlagen der DSGVO, auf deren Basis wir personenbezogene Daten verarbeiten. Bitte nehmen Sie zur Kenntnis, dass neben den Regelungen der DSGVO nationale Datenschutzvorgaben in Ihrem bzw. unserem Wohn- oder Sitzland gelten können. Sollten ferner im Einzelfall speziellere Rechtsgrundlagen maßgeblich sein, teilen wir Ihnen diese in der Datenschutzerklärung mit.

- Einwilligung (Art. 6 Abs. 1 S. 1 lit. a) DSGVO)** - Die betroffene Person hat ihre Einwilligung in die Verarbeitung der sie betreffenden personenbezogenen Daten für einen spezifischen Zweck oder mehrere bestimmte Zwecke gegeben.
- Vertragserfüllung und vorvertragliche Anfragen (Art. 6 Abs. 1 S. 1 lit. b) DSGVO)** - Die Verarbeitung ist für die Erfüllung eines Vertrags, dessen Vertragspartei die betroffene Person ist, oder zur Durchführung vorvertraglicher Maßnahmen erforderlich, die auf Anfrage der betroffenen Person erfolgen.
- Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO)** - Die Verarbeitung ist zur Wahrung der berechtigten Interessen des Verantwortlichen oder eines Dritten erforderlich, sofern nicht die Interessen oder Grundrechte und Grundfreiheiten der betroffenen Person, die den Schutz personenbezogener Daten erfordern, überwiegen.

**Nationale Datenschutzregelungen in Deutschland:** Zusätzlich zu den Datenschutzregelungen der DSGVO gelten nationale Regelungen zum Datenschutz in Deutschland. Hierzu gehört insbesondere das Gesetz zum Schutz vor Missbrauch personenbezogener Daten bei der Datenverarbeitung (Bundesdatenschutzgesetz – BDSG). Das BDSG enthält insbesondere Spezialregelungen zum Recht auf Auskunft, zum Recht auf Löschung, zum Widerspruchsrecht, zur Verarbeitung besonderer Kategorien personenbezogener Daten, zur Verarbeitung für andere Zwecke und zur Übermittlung sowie automatisierten Entscheidungsfindung im Einzelfall einschließlich Profiling. Ferner können Landesdatenschutzgesetze der einzelnen Bundesländer zur Anwendung gelangen.

<h2 id="mOverview">Übersicht der Verarbeitungen</h2><p>Die nachfolgende Übersicht fasst die Arten der verarbeiteten Daten und die Zwecke ihrer Verarbeitung zusammen und verweist auf die betroffenen Personen.</p>

<h3>Arten der verarbeiteten Daten</h3>

<ul><li>Kontaktdaten.</li><li>Inhaltsdaten.</li><li>Nutzungsdaten.</li><li>Meta-, Kommunikations- und Verfahrensdaten.</li></ul>

<h3>Kategorien betroffener Personen</h3>

<ul><li>Kommunikationspartner.</li><li>Nutzer.</li></ul>

<h3>Zwecke der Verarbeitung</h3>

<ul><li>Kontaktanfragen und Kommunikation.</li><li>Sicherheitsmaßnahmen.</li><li>Verwaltung und Beantwortung von Anfragen.</li><li>Feedback.</li><li>Bereitstellung unseres Onlineangebotes und Nutzerfreundlichkeit.</li><li>Informationstechnische Infrastruktur.</li></ul>

<h2 id="m27">Sicherheitsmaßnahmen</h2><p>Wir treffen nach Maßgabe der gesetzlichen Vorgaben unter Berücksichtigung des Stands der Technik, der Implementierungskosten und der Art, des Umfangs, der Umstände und der Zwecke der Verarbeitung sowie der unterschiedlichen Eintrittswahrscheinlichkeiten und des Ausmaßes der Bedrohung der Rechte und Freiheiten natürlicher Personen geeignete technische und organisatorische Maßnahmen, um ein dem Risiko angemessenes Schutzniveau zu gewährleisten.</p>

<p>Zu den Maßnahmen gehören insbesondere die Sicherung der Vertraulichkeit, Integrität und Verfügbarkeit von Daten durch Kontrolle des physischen und elektronischen Zugangs zu den Daten als auch des sie betreffenden Zugriffs, der Eingabe, der Weitergabe, der Sicherung der Verfügbarkeit und ihrer Trennung. Des Weiteren haben wir Verfahren eingerichtet, die eine Wahrnehmung von Betroffenenrechten, die Löschung von Daten und Reaktionen auf die Gefährdung der Daten gewährleisten. Ferner berücksichtigen wir den Schutz personenbezogener Daten bereits bei der Entwicklung bzw. Auswahl von Hardware, Software sowie Verfahren entsprechend dem Prinzip des Datenschutzes, durch Technikgestaltung und durch datenschutzfreundliche Voreinstellungen.</p>

<p>TLS-Verschlüsselung (https): Um Ihre via unserem Online-Angebot übermittelten Daten zu schützen, nutzen wir eine TLS-Verschlüsselung. Sie erkennen derart verschlüsselte Verbindungen an dem Präfix https:// in der Adresszeile Ihres Browsers.</p>

<h2 id="m25">Übermittlung von personenbezogenen Daten</h2><p>Im Rahmen unserer Verarbeitung von personenbezogenen Daten kommt es vor, dass die Daten an andere Stellen, Unternehmen, rechtlich selbstständige Organisationseinheiten oder Personen übermittelt oder sie ihnen gegenüber offengelegt werden. Zu den Empfängern dieser Daten können z.B. mit IT-Aufgaben beauftragte Dienstleister oder Anbieter von Diensten und Inhalten, die in eine Webseite eingebunden werden, gehören. In solchen Fällen beachten wir die gesetzlichen Vorgaben und schließen insbesondere entsprechende Verträge bzw. Vereinbarungen, die dem Schutz Ihrer Daten dienen, mit den Empfängern Ihrer Daten ab.</p>

## Internationale Datentransfers

Datenverarbeitung in Drittländern: Sofern wir Daten in einem Drittland (d.h., außerhalb der Europäischen Union (EU), des Europäischen Wirtschaftsraums (EWR)) verarbeiten oder die Verarbeitung im Rahmen der Inanspruchnahme von Diensten Dritter oder der Offenlegung bzw. Übermittlung von Daten an andere Personen, Stellen oder Unternehmen stattfindet, erfolgt dies nur im Einklang mit den gesetzlichen Vorgaben.

Vorbehaltlich ausdrücklicher Einwilligung oder vertraglich oder gesetzlich erforderlicher Übermittlung (s. Art. 49 DSGVO) verarbeiten oder lassen wir die Daten nur in Drittländern mit einem anerkannten Datenschutzniveau (Art. 45 DSGVO), beim Vorliegen und Einhaltung vertraglichen Verpflichtung durch sogenannte Standardschutzklauseln der EU-Kommission (Art. 46 DSGVO) oder beim Vorliegen von Zertifizierungen oder verbindlicher internen Datenschutzvorschriften (s. Art. 44 bis 49 DSGVO, Informationsseite der EU-Kommission: [https://ec.europa.eu/info/law/law-topic/data-protection/international-dimension-data-protection\\_de](https://ec.europa.eu/info/law/law-topic/data-protection/international-dimension-data-protection_de)).

Trans-Atlantic Data Privacy Framework (TADPF): Im Rahmen des sogenannten „Trans-Atlantic Data Privacy Framework“ (TADPF) hat die EU-Kommission das Datenschutzniveau ebenfalls für bestimmte Unternehmen aus den USA anerkannt. Die Liste der zertifizierten Unternehmen als auch weitere Informationen zu dem TADPF können Sie der Webseite des Handelsministeriums der USA unter <https://www.dataprivacyframework.gov/> (in Englisch) entnehmen. Informationen in deutscher und in anderen Sprachen finden Sie auf der Webseite der EU-Kommission: [https://commission.europa.eu/law/law-topic/data-protection/international-dimension-data-protection/eu-us-data-transfers\\_de](https://commission.europa.eu/law/law-topic/data-protection/international-dimension-data-protection/eu-us-data-transfers_de) Wir informieren Sie ferner über die von uns eingesetzten Unternehmen, die unter dem Trans-Atlantic Data Privacy Framework zertifiziert sind.

## Löschung von Daten

Die von uns verarbeiteten Daten werden nach Maßgabe der gesetzlichen Vorgaben gelöscht, sobald deren zur Verarbeitung erlaubten Einwilligungen widerrufen werden oder sonstige Erlaubnisse entfallen (z.B. wenn der Zweck der Verarbeitung dieser Daten entfallen ist oder sie für den Zweck nicht erforderlich sind). Sofern die Daten nicht gelöscht werden, weil sie für andere und gesetzlich zulässige Zwecke erforderlich sind, wird deren Verarbeitung auf diese Zwecke beschränkt. D.h., die Daten werden gesperrt und nicht für andere Zwecke verarbeitet. Das gilt z.B. für Daten, die aus handels- oder steuerrechtlichen Gründen aufbewahrt werden müssen oder deren Speicherung zur Geltendmachung, Ausübung oder Verteidigung von Rechtsansprüchen oder zum Schutz der Rechte einer anderen natürlichen oder juristischen Person erforderlich ist.

Unsere Datenschutzhinweise können ferner weitere Angaben zu der Aufbewahrung und Löschung von Daten beinhalten, die für die jeweiligen Verarbeitungen vorrangig gelten.

**Rechte der betroffenen Personen**  
Rechte der betroffenen Personen aus der DSGVO: Ihnen stehen als Betroffene nach der DSGVO verschiedene Rechte zu, die sich insbesondere aus Art. 15 bis 21 DSGVO ergeben:

- Widerspruchsrecht:** Sie haben das Recht, aus Gründen, die sich aus Ihrer besonderen Situation ergeben, jederzeit gegen die Verarbeitung der Sie betreffenden personenbezogenen Daten, die aufgrund von Art. 6 Abs. 1 lit. e oder f DSGVO erfolgt, Widerspruch einzulegen; dies gilt auch für ein auf diese Bestimmungen gestütztes Profiling. Werden die Sie betreffenden personenbezogenen Daten verarbeitet, um Direktwerbung zu betreiben, haben Sie das Recht, jederzeit Widerspruch gegen die Verarbeitung der Sie betreffenden personenbezogenen Daten zum Zwecke derartiger Werbung einzulegen; dies gilt auch für das Profiling, soweit es mit solcher Direktwerbung in Verbindung steht.
- Widerrufsrecht bei Einwilligungen:** Sie haben das Recht, erteilte Einwilligungen jederzeit zu widerrufen.
- Auskunftsrecht:** Sie haben das Recht, eine Bestätigung darüber zu verlangen, ob betreffende Daten verarbeitet werden und auf Auskunft über diese Daten sowie auf weitere Informationen und Kopie der Daten entsprechend den gesetzlichen Vorgaben.
- Recht auf Berichtigung:** Sie haben entsprechend den gesetzlichen Vorgaben das Recht, die Vervollständigung der Sie betreffenden Daten oder die Berichtigung der Sie betreffenden unrichtigen Daten zu verlangen.
- Recht auf Löschung und Einschränkung der Verarbeitung:** Sie haben nach Maßgabe der gesetzlichen Vorgaben das Recht, zu verlangen, dass Sie betreffende Daten unverzüglich gelöscht werden, bzw. alternativ nach Maßgabe der gesetzlichen Vorgaben eine Einschränkung der Verarbeitung der Daten zu verlangen.
- Recht auf Datenübertragbarkeit:** Sie haben das Recht, Sie betreffende Daten, die Sie uns bereitgestellt haben, nach Maßgabe der gesetzlichen Vorgaben in einem strukturierten, gängigen und maschinenlesbaren Format zu erhalten oder deren Übermittlung an einen anderen Verantwortlichen zu fordern.
- Beschwerde bei Aufsichtsbehörde:** Sie haben unbeschadet eines anderweitigen verwaltungsrechtlichen oder gerichtlichen Rechtsbehelfs das Recht auf Beschwerde bei einer Aufsichtsbehörde, insbesondere in dem Mitgliedstaat ihres gewöhnlichen Aufenthaltsorts, ihres Arbeitsplatzes oder des Orts des mutmaßlichen Verstoßes, wenn Sie der Ansicht sind, dass die Verarbeitung der Sie betreffenden personenbezogenen Daten gegen die Vorgaben der DSGVO verstößt.

**Einsatz von Cookies**  
Cookies sind kleine Textdateien, bzw. sonstige Speichervermerke, die Informationen auf Endgeräten speichern und Informationen aus den Endgeräten auslesen. Z.B. um den Login-Status in einem Nutzerkonto, einen Warenkorbinhalt in einem E-Shop, die aufgerufenen Inhalte oder verwendete Funktionen eines Onlineangebotes speichern. Cookies können ferner zu unterschiedlichen Zwecken eingesetzt werden, z.B. zu Zwecken der Funktionsfähigkeit, Sicherheit und Komfort von Onlineangeboten sowie der Erstellung von Analysen der Besucherströme.

**Hinweise zur Einwilligung:** Wir setzen Cookies im Einklang mit den gesetzlichen Vorschriften ein. Daher holen wir von den Nutzern eine vorhergehende Einwilligung ein, außer wenn diese gesetzlich nicht gefordert ist. Eine Einwilligung ist insbesondere nicht notwendig, wenn das Speichern und das Auslesen der Informationen, also auch von Cookies, unbedingt erforderlich sind, um dem den Nutzern einen von ihnen ausdrücklich gewünschten Telemediendienst (also unser Onlineangebot) zur Verfügung zu stellen. Zu den unbedingt erforderlichen Cookies gehören in der Regel Cookies mit Funktionen, die der Anzeige und Lauffähigkeit des Onlineangebotes, dem Lastausgleich, der Sicherheit, der Speicherung der Präferenzen und Auswahlmöglichkeiten der Nutzer oder ähnlichen mit der Bereitstellung der Haupt- und Nebenfunktionen des von den Nutzern angeforderten Onlineangebotes zusammenhängenden Zwecken dienen. Die widerrufliche Einwilligung wird gegenüber den Nutzern deutlich kommuniziert und enthält die Informationen zu der jeweiligen Cookie-Nutzung.

**Hinweise zu datenschutzrechtlichen Rechtsgrundlagen:** Auf welcher datenschutzrechtlichen Rechtsgrundlage wir die personenbezogenen Daten der Nutzer mit Hilfe von Cookies verarbeiten, hängt davon ab, ob wir Nutzer um eine Einwilligung bitten. Falls die Nutzer einwilligen, ist die Rechtsgrundlage der Verarbeitung Ihrer Daten die erklärte Einwilligung. Andernfalls werden die mithilfe von Cookies verarbeiteten Daten auf Grundlage unserer berechtigten Interessen (z.B. an einem betriebswirtschaftlichen Betrieb unseres Onlineangebotes und Verbesserung seiner Nutzbarkeit) verarbeitet oder, wenn dies im Rahmen der Erfüllung unserer vertraglichen Pflichten erfolgt, wenn der Einsatz von Cookies erforderlich ist, um unsere vertraglichen Verpflichtungen zu erfüllen. Zu welchen Zwecken die Cookies von uns verarbeitet werden, darüber klären wir im Laufe dieser Datenschutzerklärung oder im Rahmen von unseren Einwilligungs- und Verarbeitungsprozessen auf.

**Speicherdauer:** Im Hinblick auf die Speicherdauer werden die folgenden Arten von Cookies unterschieden:

- Temporäre Cookies (auch: Session- oder Sitzungs-Cookies):** Temporäre Cookies werden spätestens gelöscht, nachdem ein Nutzer ein Online-Angebot verlassen und sein Endgerät (z.B. Browser oder mobile Applikation) geschlossen hat.
- Permanente Cookies:** Permanente Cookies bleiben auch nach dem Schließen des Endgerätes gespeichert. So können beispielsweise der Login-Status gespeichert oder bevorzugte Inhalte direkt angezeigt werden, wenn der Nutzer eine Website erneut besucht. Ebenso können die mit Hilfe von Cookies erhobenen Daten der Nutzer zur Reichweitenmessung verwendet werden. Sofern wir Nutzern keine expliziten Angaben zur Art und Speicherdauer von Cookies mitteilen (z. B. im Rahmen der Einholung der Einwilligung), sollten Nutzer davon ausgehen, dass Cookies permanent sind und die Speicherdauer bis zu zwei Jahre betragen kann.

**Allgemeine Hinweise zum Widerruf und Widerspruch (sog. "Opt-Out"):** Nutzer können die von ihnen abgegebenen Einwilligungen jederzeit widerrufen und der Verarbeitung entsprechend den gesetzlichen Vorgaben widersprechen. Hierzu können Nutzer unter anderem die Verwendung von Cookies in den Einstellungen ihres Browsers einschränken (wobei dadurch auch die

Funktionalität unseres Onlineangebotes eingeschränkt sein kann). Ein Widerspruch gegen die Verwendung von Cookies zu Online-Marketing-Zwecken kann auch über die Websites <https://optout.aboutads.info/> und <https://www.youronlinechoices.com/> erklärt werden.

- Rechtsgrundlagen:** Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO); Einwilligung (Art. 6 Abs. 1 S. 1 lit. a) DSGVO).

**Weitere Hinweise zu Verarbeitungsprozessen, Verfahren und Diensten:**

- Verarbeitung von Cookie-Daten auf Grundlage einer Einwilligung:** Wir setzen ein Verfahren zum Cookie-Einwilligungs-Management ein, in dessen Rahmen die Einwilligungen der Nutzer in den Einsatz von Cookies, bzw. der im Rahmen des Cookie-Einwilligungs-Management-Verfahrens genannten Verarbeitungen und Anbieter eingeholt sowie von den Nutzern verwaltet und widerrufen werden können. Hierbei wird die Einwilligungserklärung gespeichert, um deren Abfrage nicht erneut wiederholen zu müssen und die Einwilligung entsprechend der gesetzlichen Verpflichtung nachweisen zu können. Die Speicherung kann serverseitig und/oder in einem Cookie (sogenanntes Opt-In-Cookie, bzw. mithilfe vergleichbarer Technologien) erfolgen, um die Einwilligung einem Nutzer, bzw. dessen Gerät zuordnen zu können. Vorbehaltlich individueller Angaben zu den Anbietern von Cookie-Management-Diensten, gelten die folgenden Hinweise: Die Dauer der Speicherung der Einwilligung kann bis zu zwei Jahren betragen. Hierbei wird ein pseudonymer Nutzer-Identifikator gebildet und mit dem Zeitpunkt der Einwilligung, Angaben zur Reichweite der Einwilligung (z. B. welche Kategorien von Cookies und/oder Diensteanbieter) sowie dem Browser, System und verwendeten Endgerät gespeichert; **Rechtsgrundlagen:** Einwilligung (Art. 6 Abs. 1 S. 1 lit. a) DSGVO).

## Bereitstellung des Onlineangebotes und Webhosting

Wir verarbeiten die Daten der Nutzer, um ihnen unsere Online-Dienste zur Verfügung stellen zu können. Zu diesem Zweck verarbeiten wir die IP-Adresse des Nutzers, die notwendig ist, um die Inhalte und Funktionen unserer Online-Dienste an den Browser oder das Endgerät der Nutzer zu übermitteln.

- Verarbeitete Datenarten:** Nutzungsdaten (z.B. besuchte Webseiten, Interesse an Inhalten, Zugriffszeiten); Meta-, Kommunikations- und Verfahrensdaten (z. B. IP-Adressen, Zeitangaben, Identifikationsnummern, Einwilligungsstatus).
- Betroffene Personen:** Nutzer (z.B. Webseitenbesucher, Nutzer von Onlinediensten).
- Zwecke der Verarbeitung:** Bereitstellung unseres Onlineangebotes und Nutzerfreundlichkeit; Informationstechnische Infrastruktur (Betrieb und Bereitstellung von Informationssystemen und technischen Geräten (Computer, Server etc.).); Sicherheitsmaßnahmen.
- Rechtsgrundlagen:** Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO).

**Weitere Hinweise zu Verarbeitungsprozessen, Verfahren und Diensten:**



**Bereitstellung Onlineangebot auf gemietetem Speicherplatz:**  
Für die Bereitstellung unseres Onlineangebotes nutzen wir Speicherplatz, Rechenkapazität und Software, die wir von einem entsprechenden Serveranbieter (auch "Webhoster" genannt) mieten oder anderweitig beziehen;  
**Rechtsgrundlagen:** Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO).

**Erhebung von Zugriffsdaten und Logfiles:** Der Zugriff auf unser Onlineangebot wird in Form von so genannten "Server-Logfiles" protokolliert. Zu den Serverlogfiles können die Adresse und Name der abgerufenen Webseiten und Dateien, Datum und Uhrzeit des Abrufs, übertragene Datenmengen, Meldung über erfolgreichen Abruf, Browsertyp nebst Version, das Betriebssystem des Nutzers, Referrer URL (die zuvor besuchte Seite) und im Regelfall IP-Adressen und der anfragende Provider gehören.

Die Serverlogfiles können zum einen zu Zwecken der Sicherheit eingesetzt werden, z.B., um eine Überlastung der Server zu vermeiden (insbesondere im Fall von missbräuchlichen Angriffen, sogenannten DDoS-Attacken) und zum anderen, um die Auslastung der Server und ihre Stabilität sicherzustellen;  
**Rechtsgrundlagen:** Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO); **Löschung von Daten:** Logfile-Informationen werden für die Dauer von maximal 30 Tagen gespeichert und danach gelöscht oder anonymisiert. Daten, deren weitere Aufbewahrung zu Beweiszwecken erforderlich ist, sind bis zur endgültigen Klärung des jeweiligen Vorfalls von der Löschung ausgenommen.

**Kontakt- und Anfragenverwaltung**  
Bei der Kontaktaufnahme mit uns (z.B. per Post, Kontaktformular, E-Mail, Telefon oder via soziale Medien) sowie im Rahmen bestehender Nutzer- und Geschäftsbeziehungen werden die Angaben der anfragenden Personen verarbeitet soweit dies zur Beantwortung der Kontaktanfragen und etwaiger angefragter Maßnahmen erforderlich ist.

**Verarbeitete Datenarten:** Kontaktdaten (z.B. E-Mail, Telefonnummern); Inhaltsdaten (z.B. Eingaben in Onlineformularen); Nutzungsdaten (z.B. besuchte Webseiten, Interesse an Inhalten, Zugriffszeiten); Meta-, Kommunikations- und Verfahrensdaten (z. B. IP-Adressen, Zeitangaben, Identifikationsnummern, Einwilligungsstatus).  
**Betroffene Personen:** Kommunikationspartner.  
**Zwecke der Verarbeitung:** Kontaktanfragen und Kommunikation; Verwaltung und Beantwortung von Anfragen; Feedback (z.B. Sammeln von Feedback via Online-Formular); Bereitstellung unseres Onlineangebotes und Nutzerfreundlichkeit.  
**Rechtsgrundlagen:** Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO); Vertragserfüllung und vorvertragliche

Anfragen (Art. 6 Abs. 1 S. 1 lit. b) DSGVO).

- Weitere Hinweise zu Verarbeitungsprozessen, Verfahren und Diensten:**

- Kontaktformular:** Wenn Nutzer über unser Kontaktformular, E-Mail oder andere Kommunikationswege mit uns in Kontakt treten, verarbeiten wir die uns in diesem Zusammenhang mitgeteilten Daten zur Bearbeitung des mitgeteilten Anliegens;
- Rechtsgrundlagen:** Vertragserfüllung und vorvertragliche Anfragen (Art. 6 Abs. 1 S. 1 lit. b) DSGVO), Berechtigte Interessen (Art. 6 Abs. 1 S. 1 lit. f) DSGVO).

## Änderung und Aktualisierung der Datenschutzerklärung

Wir bitten Sie, sich regelmäßig über den Inhalt unserer Datenschutzerklärung zu informieren. Wir passen die Datenschutzerklärung an, sobald die Änderungen der von uns durchgeführten Datenverarbeitungen dies erforderlich machen. Wir informieren Sie, sobald durch die Änderungen eine Mitwirkungshandlung Ihrerseits (z.B. Einwilligung) oder eine sonstige individuelle Benachrichtigung erforderlich wird.

Sofern wir in dieser Datenschutzerklärung Adressen und Kontaktinformationen von Unternehmen und Organisationen angeben, bitten wir zu beachten, dass die Adressen sich über die Zeit ändern können und bitten die Angaben vor Kontaktaufnahme zu prüfen.

## Begriffsdefinitionen

In diesem Abschnitt erhalten Sie eine Übersicht über die in dieser Datenschutzerklärung verwendeten Begrifflichkeiten. Soweit die Begrifflichkeiten gesetzlich definiert sind, gelten deren gesetzliche Definitionen. Die nachfolgenden Erläuterungen sollen dagegen vor allem dem Verständnis dienen.

- Personenbezogene Daten:** "Personenbezogene Daten" sind alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person (im Folgenden "betroffene Person") beziehen; als identifizierbar wird eine natürliche Person angesehen, die direkt oder indirekt, insbesondere mittels Zuordnung zu einer Kennung wie einem Namen, zu einer Kennnummer, zu Standortdaten, zu einer Online-Kennung (z.B. Cookie) oder zu einem oder mehreren besonderen Merkmalen identifiziert werden kann, die Ausdruck der physischen, physiologischen, genetischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität dieser natürlichen Person sind.

- Verantwortlicher:** Als "Verantwortlicher" wird die natürliche oder juristische Person, Behörde, Einrichtung oder andere Stelle, die allein oder gemeinsam mit anderen über die Zwecke und Mittel der Verarbeitung von personenbezogenen Daten entscheidet, bezeichnet.

- Verarbeitung:** "Verarbeitung" ist jeder mit oder ohne Hilfe automatisierter Verfahren ausgeführte Vorgang oder jede solche Vorgangsreihe im Zusammenhang mit personenbezogenen Daten. Der Begriff reicht weit und umfasst praktisch jeden Umgang mit Daten, sei es das Erheben, das Auswerten, das Speichern, das Übermitteln oder das Löschen.

[Erstellt mit kostenlosem Datenschutz-Generator.de von Dr. Thomas Schwenke](https://datenschutz-generator.de/ "Rechtstext von Dr. Schwenke - für weitere Informationen bitte anklicken.")

## 13.9 IHK Abschlussprüfung Abnahmeprotokoll

- |             |  |                                     |
|-------------|--|-------------------------------------|
| 1. Feature  | Das Laden der bereits bestehenden MySQL Datenbanken ins Programm               | <input checked="" type="checkbox"/> |
| 2. Feature  | Erstellung einer neuen Datenbank   | <input checked="" type="checkbox"/> |
| 3. Feature  | Erstellung von Tabellen in einer Datenbank                                     | <input checked="" type="checkbox"/> |
| 4. Feature  | Änderung vom Spaltennamen der schon bestehenden Tabelle                        | <input checked="" type="checkbox"/> |
| 5. Feature  | Änderung vom Datentyp der schon bestehenden Tabelle                            | <input checked="" type="checkbox"/> |
| 6. Feature  | Hinzufügen einer Spalte in einer schon bestehenden Tabelle                     | <input checked="" type="checkbox"/> |
| 7. Feature  | Filtern/Suchen mit Schlüsselwort nach Tabellennamen in einer DB                | <input checked="" type="checkbox"/> |
| 8. Feature  | Filtern/Suchen mit Schlüsselwort nach Spaltennamen in einer DB                 | <input checked="" type="checkbox"/> |
| 9. Feature  | Filtern/Suchen mit Schlüsselwort in allen Zellen(Daten) einer DB               | <input checked="" type="checkbox"/> |
| 10. Feature | Löschen von Datenbanken  | <input checked="" type="checkbox"/> |
| 11. Feature | Löschen von Tabellen in einer DB   | <input checked="" type="checkbox"/> |
| 12. Feature | Schutz von System DBs durch nicht anzeigen im Programm                         | <input checked="" type="checkbox"/> |
| 13. Feature | Anzeige und Verlinkung von Tabellen auf der Datenbankansicht                   | <input checked="" type="checkbox"/> |
| 14. Feature | Anzeige und Verlinkung von Tabellen und Datenbanken in der Seitenleiste        | <input checked="" type="checkbox"/> |
| 15. Feature | Anzeige von den Daten in der Tabellenansicht                                   | <input checked="" type="checkbox"/> |
| 16. Feature | Markierung von Primär und Fremdschlüsselspalten durch [PRK] und [FRK]          | <input checked="" type="checkbox"/> |
| 17. Feature | Ändern von Relationen von Tabellen in DBs                                      | <input checked="" type="checkbox"/> |
| 18. Feature | Verlinkung auf die Startseite mit dem Logo oben links und dem Startseite Knopf | <input checked="" type="checkbox"/> |
| 19. Feature | Hinzufügen einer neuen Reihe von Daten in eine Tabelle                         | <input checked="" type="checkbox"/> |

23.08.23

Datum

B. Born

Auszubildender

T. Müller

Ausbilder

Tim Klaus Müller

13 → Weiterleitung auf die Datenbank  
14 → Weiterleitung auf die Tabelle