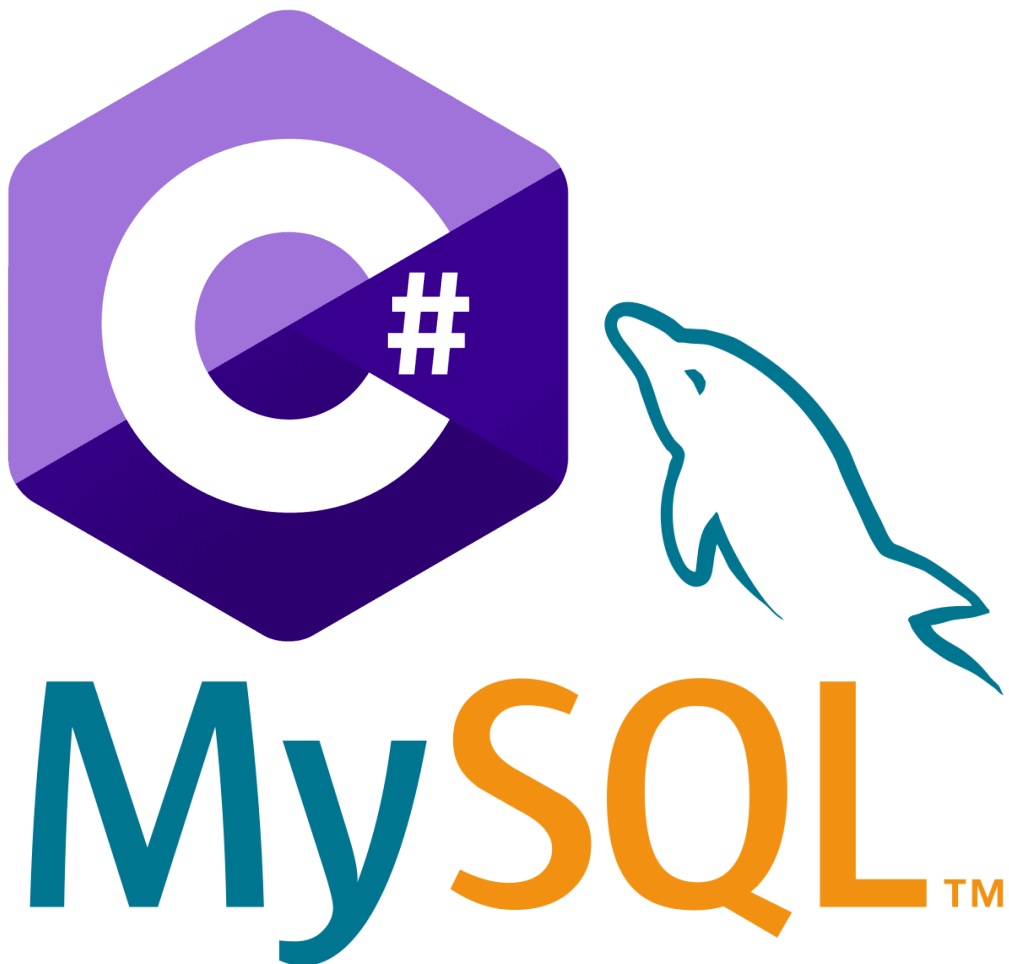


Schulprojekt - Telefonbuch

Gruppe:

Marvin Kaiser
Mercan Aldemir
Ioana Craioveanu



1 Inhaltsverzeichnis

1 Inhaltsverzeichnis	2
2 Ausgangssituation	3
2.1 Ziel des Projekts	3
2.2 Anforderungen	3
2.3 Gewünschtes Ergebnis	3
3 Komponenten	4
3.1 Die Datenbank	4
3.2 Die Oberfläche	4
4 Die grafische Benutzeroberfläche - GUI	4
4.1 Navigation	5
4.1.1 Routing	5
4.2 Das erstellen eines Benutzers	7
4.2.1 Grundüberlegung	7
4.2.2 Prozessablauf	7
4.3 Auffinden eines Benutzer	8
4.3.1 Grundüberlegung	8
4.3.2 Prozessablauf - Daten eines Eintrags ansehen	8
4.3.3 Prozessablauf - Suchen eines Eintrags	8
4.4 Das Löschen eines Benutzers	10
4.4.1 Prozessablauf	10
4.5 Das Verändern eines bestehenden Benutzers	10
4.5.1 Grundüberlegung	10
4.5.2 Prozessablauf	10
5 Die Datenbank	11
5.1 Schnittstelle	11
5.2 Design	11
5.2.1 ER-Modell	11
5.4 Datenbankmanager	12
5.4.1 Datensatz erstellen	12
5.4.2 Datensätze auffinden	13
5.4.3 Datensätze löschen	13
5.4.4 Datensätze aktualisieren	14
6 Soll-Ist-Vergleich	14
7 Ausblick	15
8 Fazit	15
9 Anhang	16
10 Quellen	16

2 Ausgangssituation

In der Schule, im Fach BTS soll in 2- oder 3er Gruppen ein Projekt erstellt werden. Dieses Projekt dient zu 100% der Projektkompetenz-Note und zu 50% der BTS-Note bei. Am Ende des Projekts soll ein vollständiges Programm zur Verwaltung da sein. Eine Datenbank auf Basis von Excel, mit einer MySQL Datenbank in XAMPP oder in einer Access-Datenbank ist erforderlich.

2.1 Ziel des Projekts

Als Projekt haben wir uns für ein Telefonbuch entschieden, da es eine recht simple Anwendung ist, die aber einen enormen Nutzen hat. Es gibt Millionen von Telefonnummern, die man sich kaum bis gar nicht merken kann. Daher wollen wir eine Anwendung bereitstellen, die es Benutzern ermöglicht, all ihre benötigten Kontakte abzuspeichern. Selbstverständlich sollen nicht nur der Name und die Telefonnummer gespeichert werden. Ebenfalls sollen die E-Mail Adresse, sowie die Anschrift einer jeden Person gespeichert werden.

2.2 Anforderungen

Das Projekt soll, je nach gewähltem Schwierigkeitsgrad, bestimmte Anforderungen erfüllen. So können die Daten entweder in einem Array, einer Excel-Datei oder auch in einer Datenbank gespeichert werden. Je nach Könner-Level sollen Typenerkennung und eine Suchfunktion eingebaut werden. Zusätzlich sollen ein ER-Modell und ein Relationenschema für die Datenbank erstellt werden, sowie ein Struktogramm. Die grafische Benutzeroberfläche wird mit Hilfe einer simplen "Windows Forms" Anwendung erstellt. Zusätzlich wird ein Speichermedium benötigt, welches uns erlaubt, Einträge zu speichern und abzurufen. Dementsprechend haben wir uns für eine Datenbank entschieden, die viele Daten effizient speichern kann. Vorab haben wir uns bereits erste Gedanken zur Funktionalität unseres Produktes gemacht, daher sollte es Grundfunktionen wie das Hinzufügen eines neuen Eintrags, das Ändern sowie Löschen vorhandener Einträge besitzen. Zusätzlich kamen wir zu dem Schluss, dass eine Suchfunktion zum Durchforschen der Einträge eine sinnvolle Ergänzung sei. Sollte ein Benutzer sehr viele Einträge in seiner Datenbank besitzen, kann er diese bequem nach seinem gewünschten Eintrag durchsuchen. Hierbei ist es wichtig zu erwähnen, dass unser Projekt nur einen Prototyp darstellt und noch an vielen Aspekten weiterentwickelt wird. Zusätzlich ist zu beachten, dass wir für unser Projekt ausschließlich Freeware¹ benutzen, da es sich, wie bereits erwähnt, um kein fertiges Produkt handelt.

2.3 Gewünschtes Ergebnis

Das Resultat soll eine schöne, einheitliche und leicht zu benutzende Oberfläche darstellen. Die Anwendung soll Funktionen wie das Erstellen, Verändern und Löschen eines Eintrags bereitstellen, sowie eine Suchfunktion, um zwischen Einträgen schnell und flexibel zu navigieren.

¹ kostenlose Software

3 Komponenten

Unser Projekt besteht aus einer simpel gehaltenen GUI² und einer Anbindung an eine Datenbank. Die Datenbank soll alle vom Benutzer gespeicherten Einträge empfangen und abspeichern. Die GUI ermöglicht es dem Benutzer, alle benötigten Funktionen auszuführen, die er für das Verwalten seiner Daten benötigt.

3.1 Die Datenbank

Die Datenbank stellt die Kernfunktion unseres Programms zur Verfügung, Daten für einen späteren Gebrauch abzuspeichern. Daher ist es essentiell, dass wir eine Datenbank verwenden, die alle benötigten Funktionen bereitstellt. Unsere Wahl fiel auf eine MySQL³ Datenbank. Eine MySQL Datenbank erfüllt alle Ansprüche, die wir an eine Datenbank haben. Sie lässt sich leicht bedienen, leicht konfigurieren und ist zudem kostenlos nutzbar. Datenbanksysteme wie MongoDB kamen natürlich auch in Frage, jedoch hatten wir am meisten Erfahrungen mit MySQL Datenbanken gemacht und somit war es eine Frage der persönlichen Präferenz.

3.2 Die Oberfläche

Bei unserer Oberfläche haben wir uns für “Windows Forms” entschieden. Mit Hilfe von Visual Studio⁴ lassen sich einfache GUI Elemente flexibel und schnell erstellen. Windows Forms Anwendungen sind meistens recht einfach aufgebaut und entsprechen daher genau unserem Ziel. Windows Forms fängt bereits die Events⁵ ab, die ein Benutzer auslösen kann, somit muss nur die Funktionalität hinterlegt werden, die ausgeführt werden soll. So können wir zum Beispiel einfach die Funktionalität hinterlegen, ohne einen Event Listener⁶ zu registrieren, Visual Studio macht das vollautomatisch und hilft uns dabei, sich auf das Wesentliche zu konzentrieren.

4 Die grafische Benutzeroberfläche - GUI

Um die Anwendung funktionstüchtig zu machen, werden Funktionen wie das Erstellen, das Ändern sowie auch das Löschen der Datensätze benötigt. Es muss somit eine Verbindung zur Datenbank hergestellt werden, damit diese Funktionen möglich sind. Außerdem werden Funktionalitäten wie das Erkennen von gefüllten oder auch nicht befüllten Input-Feldern benötigt, da wir keine leeren Einträge haben wollen. Das bedeutet, dass Einträge nur erstellt werden können, wenn sie auch befüllt sind. Zudem benötigen wir auch das Erkennen von E-Mail-Adressen, zum Beispiel max.mustermann@gmail.com. Es muss ein “@” vorhanden sein, um die E-Mail zu erkennen.

² grafische Benutzeroberfläche

³ Datenbanksystem von Oracle Corporation

⁴ Programmierumgebung

⁵ vom Benutzer ausgelöste Ereignisse

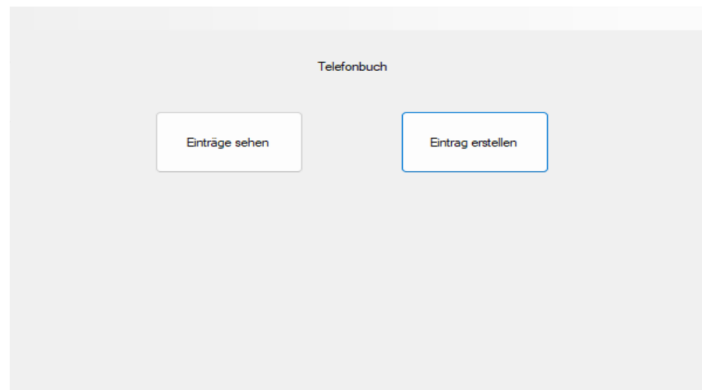
⁶ überwacht ausgelöste Ereignisse

4.1 Navigation

4.1.1 Routing

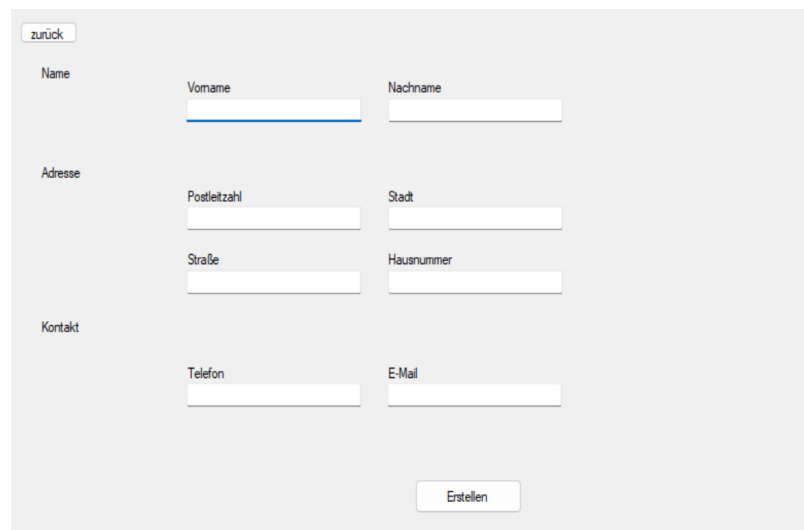
Für eine übersichtliche und strukturierte Darstellung haben wir mehrere Formen genutzt. Damit die User von der einen Form zur anderen Form springen können, durch beispielsweise Buttons, gibt es das Routing. Durch jeden Wechsel wird die aktuelle Form geschlossen und die darauf führende Form geöffnet, damit die Anwendung ohne Probleme geschlossen werden kann.

Der Start ist klassischer Weise auf der Startseite. Auf dieser Seite gibt es zwei Buttons, die zu unterschiedlichen Seiten führen. Wird der Klick auf den Button “Eintrag erstellen” ausgeführt, so kommt folgendes Muster:



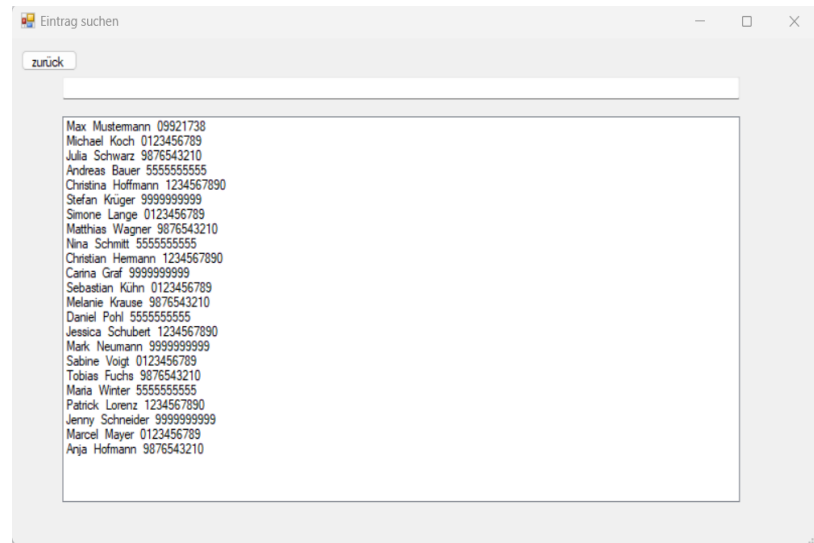
The screenshot shows a light gray background with the title 'Telefonbuch' at the top center. Below the title, there are two rectangular buttons. The left button is labeled 'Einträge sehen' and the right button is labeled 'Eintrag erstellen'.

Oben links ist der Button “zurück” zu sehen, der die Benutzer zurück auf die Startseite führt. Durch den “Einträge sehen” - Button kommt man auf die Seite, auf der alle Einträge zu sehen sind.

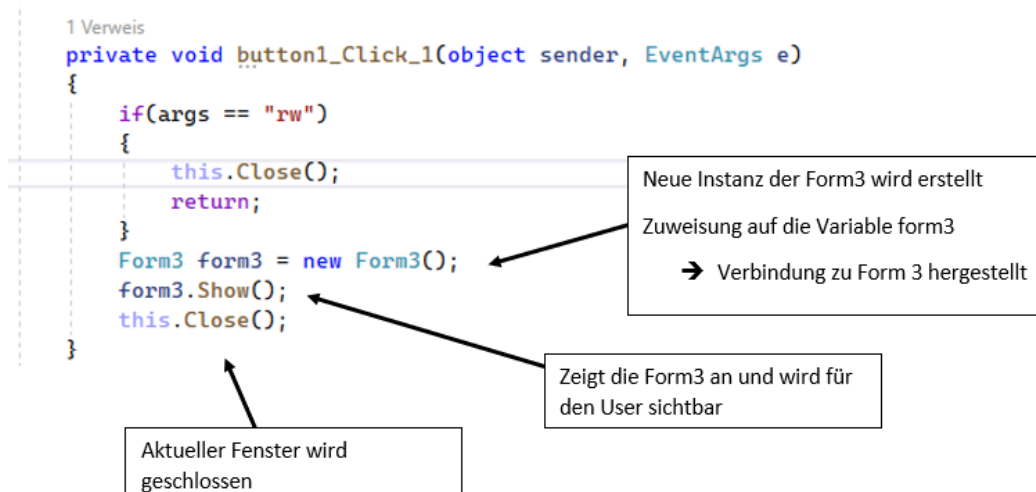
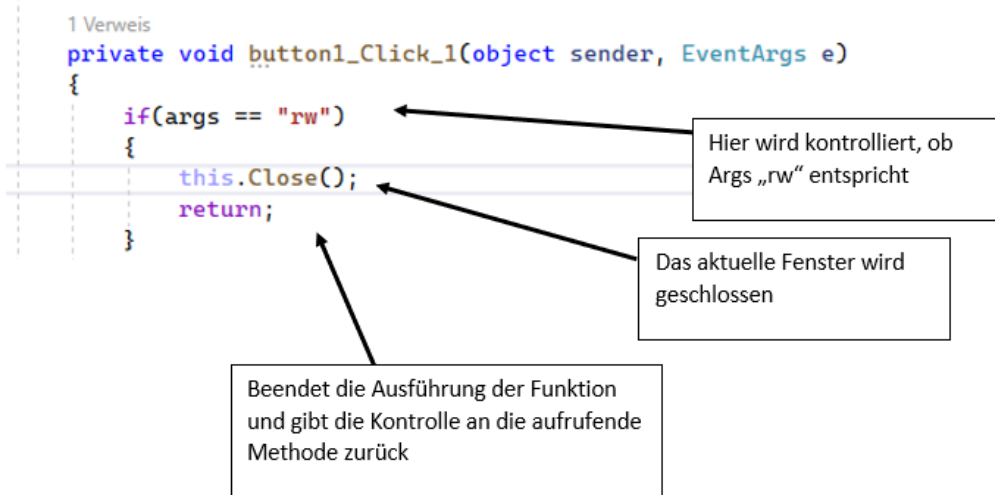


The screenshot shows a form titled 'Telefonbuch' with a 'zurück' button in the top left corner. The form is organized into three sections: 'Name', 'Adresse', and 'Kontakt'. The 'Name' section has two input fields for 'Vorname' and 'Nachname'. The 'Adresse' section has four input fields: 'Postleitzahl', 'Stadt', 'Straße', and 'Hausnummer'. The 'Kontakt' section has two input fields for 'Telefon' and 'E-Mail'. At the bottom right of the form is a button labeled 'Erstellen'.

Klickt man einen dieser Einträge an, so erscheint dieselbe Form wie bei dem Button "Eintrag erstellen", mit dem Unterschied, dass es den "Erstellen" Button nicht gibt. Es existieren nur die Buttons "Löschen" und "Änderung speichern".



Genauere Erklärung des Codes anhand eines Beispiels bezüglich des Routings:



4.2 Das erstellen eines Benutzers

4.2.1 Grundüberlegung

Bevor wir loslegen konnten, mussten wir uns zuerst die Frage stellen, welche Daten in den Einträgen enthalten sein sollten. Benötigte Daten waren auf jeden Fall:

Vorname, Nachname, Telefonnummer

Zusätzlich entschieden wir uns die Attribute:

Straße, Hausnummer, Stadt, Postleitzahl und E-Mail zu unserem Datensatz aufzunehmen.

Hierbei kam es zu dem Entschluss, die Daten Postleitzahl und Stadt optional zu machen.

Nachdem der Inhalt der Daten geklärt war, galt es nun eine entsprechende Oberfläche zu gestalten, die die nötigen Eingabelemente bereitstellt.

4.2.2 Prozessablauf

Der Benutzer soll auf ein Formular weitergeleitet werden, welches alle Textfelder für die Eingabe der Daten enthält. Zusätzlich benötigt es einen Button⁷ zum Absenden des Formulars. Der Button stößt den Prozess der Datenübertragung zur Datenbank an. Hierbei werden die eingegebenen Daten aus den jeweiligen Textfeldern ausgelesen und vorab durch eine Inhaltskontrolle geschickt. Hier wird überprüft, ob die Eingabe valide ist. Zum einen wird überprüft, ob in das Textfeld der korrekte Datentyp eingetragen wird, man soll ja nicht als Hausnummer einen String⁸ eintragen können. Zudem muss auch geprüft werden, ob das Textfeld nicht leer ist. Diese Datenvalidierung ist essentiell, um Fehlerquellen zu vermeiden.

```
if (
    string.IsNullOrEmpty(data.Vorname) &&
    string.IsNullOrEmpty(data.Nachname) &&
    string.IsNullOrEmpty(data.Strasse) &&
    string.IsNullOrEmpty(data.Hausnummer) &&
    string.IsNullOrEmpty(data.Telefon) &&
    string.IsNullOrEmpty(data.Email) &&
    (string.IsNullOrEmpty(data.Plz) && string.IsNullOrEmpty(data.OrtName)
    || string.Empty == data.Plz && string.Empty == data.OrtName ) )
```

Im oberen Skript kann man deutlich sehen, wie nacheinander alle Felder abgefragt werden, ob diese leer sind. Zusätzlich wurde die Funktion eingebaut, dass der Ortsname und die Postleitzahl optional sind. Sollte es zu einer falschen Eingabe kommen, wird ein Fehler angezeigt. Nachdem die Daten validiert wurden, wird das fertige Datenpaket an den Datenbankmanager weitergeleitet. Der Datenbankmanager stellt unsere Schnittstelle zwischen der GUI und der Datenbank dar und übernimmt das Übermitteln an die Datenbank.

⁷ Knopf

⁸ Text Datentyp - Zeichenkette

4.3 Auffinden eines Benutzer

4.3.1 Grundüberlegung

Um die Daten eines bestehenden Eintrags abzurufen, muss der Benutzer in der Lage sein, diesen spezifischen Eintrag abzurufen. Wir haben uns dafür entschieden eine ListBox⁹ zu benutzen und dort alle Einträge der Datenbank hinzuzufügen. Jedoch sollte dies nicht die einzige Möglichkeit sein, einen Eintrag zu einer Person zu finden. Deshalb haben wir uns dafür entschieden, eine Suchfunktion einzufügen. Bei sehr vielen Einträgen kann die Liste an Elementen sehr lang und unübersichtlich werden, daher wäre eine Suchfunktion eine passende Ergänzung zur Listbox. Zuerst wurde überlegt, alle Daten der Datenbank auszulesen und diese dann anschließend zu sortieren. Jedoch liegt dann die gesamte Last auf unserer Anwendung. Deshalb haben wir uns dazu entschieden, die Daten in der Datenbank zu sortieren und somit die Last auf die Datenbank zu verlagern.

4.3.2 Prozessablauf - Daten eines Eintrags ansehen

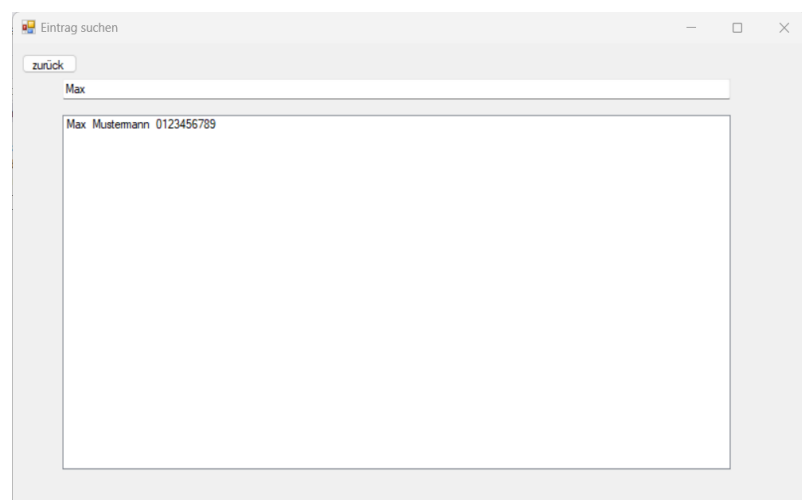
In einer Listbox werden alle gefundenen Benutzer hinzugefügt. Die Benutzer werden vom Datenbankmanager bereitgestellt. Diese Einträge können nun mit einem Klick geöffnet werden. Das Klicken öffnet das gleiche Formular wie zum Erstellen eines Eintrags, weil dieses Formular bereits alle nötigen

Textfelder besitzt. Anhand eines Parameters teilen wir dem Formular mit, dass wir keinen Eintrag erstellen wollen, sondern ihn anschauen wollen. Dadurch wird der "Erstellen" Button automatisch ausgeblendet. Beim Öffnen des Formulars werden zudem alle wichtigen Daten an die Form übergeben und in die richtigen vorhandenen Textfelder eingefügt.

```
this.input_city.Text = entry.OrtName;  
this.input_email.Text = entry.Email;  
this.input_vorname.Text = entry.Vorname;  
this.input_nachname.Text = entry.Nachname;  
this.input_postleitzahl.Text = entry.Plz;  
this.input_telefon.Text = entry.Telefon;  
this.input_HR.Text = entry.Hausnummer;  
this.input_straße.Text = entry.Strasse;
```

4.3.3 Prozessablauf - Suchen eines Eintrags

Um einen Eintrag in der Datenbank durch die GUI zu suchen, muss die Datenbank bei jeder Änderung der Sucheingabe neu abgefragt werden. Dies wurde durch das Event "textchanged" gelöst, welches von Windows Form nativ bereitgestellt wird. Immer wenn sich der Text im Suchfeld verändert, wird die Datenbank neu abgefragt.



⁹ Komponente von Windows Forms

Hierbei macht es in unseren Augen keinen Sinn, falls man eine Hausnummer als Suchbegriff eingibt, deshalb haben wir diese in der Suchfunktion auch nicht inbegriffen.

```
string[] arguments = textBox1.Text.ToUpper().Split(' ');
string inputText = arguments[0];
listBox1.Items.Clear();
string sql = "SELECT * FROM users " +
    "WHERE vorname LIKE '" + inputText + "%' " +
    "OR nachname LIKE '" + inputText + "%' " +
    "OR strasse LIKE '" + inputText + "%' " +
    "OR telefon LIKE '" + inputText + "%' " +
    "OR email LIKE '" + inputText + "%' ";
List<UserEntry> reader = DatabaseManager getUsersFromDatabase(sql);
```

Anhand dieser Funktion kann man erkennen, dass nur das erste "Argument" der Wortkette in dem SQL-Befehl in Betracht gezogen wird. Sollte man jedoch nicht nur einen Vornamen, sondern zusätzlich auch einen Nachnamen eingeben wollen, so funktioniert dies noch nicht, deshalb mussten wir unsere Suchfunktion etwas modifizieren. Wir haben unsere Funktion um eine Schleife erweitert, die alle zusätzlichen Argumente, die in der Suchleiste eingegeben wurde, überprüft. Wenn alle Argumente in einem Ergebnis vorkommen, wird es zur ListBox hinzugefügt.

```
for(int c = 0; c < reader.Count; c++)
{
    bool add = true ;
    if(arguments.Length > 1)
    {
        for(int i = 1; i < arguments.Length; i++)
        {
            if (!reader[c].Vorname.ToUpper().StartsWith(arguments[i]) &&
                !reader[c].Nachname.ToUpper().StartsWith(arguments[i]) &&
                !reader[c].Strasse.ToUpper().StartsWith(arguments[i]) &&
                !reader[c].Telefon.ToUpper().StartsWith(arguments[i]) &&
                !reader[c].Email.ToUpper().StartsWith(arguments[i]))
            {
                if(i == arguments.Length - 1)
                {
                    add = false;
                }
            }
        }
    }
    if (add)
    {
        listBox1.Items.Add(reader[c]);
    }
}
```

4.4 Das Löschen eines Benutzers

4.4.1 Prozessablauf

Um einen Eintrag zu löschen, benutzen wir das gleiche Vorgehen wie bei der Betrachtung eines Eintrags. Man wählt sich den gewünschten Eintrag aus und wird auf das selbe Formular weitergeleitet wie bei der Betrachtung und der Erstellung der Einträge. Jedoch benötigen wir nun einen zusätzlichen Button zum Löschen dieses Eintrags. Bevor der Eintrag jedoch gelöscht wird, soll nochmals abgefragt werden, ob man sich sicher sei, bevor der Datensatz vollständig aus der Datenbank entfernt wird.

```
private void button2_Click(object sender, EventArgs e)
{
    var result = MessageBox.Show("Soll dieser Benutzer entfernt werden?",
        "Benutzer löschen", MessageBoxButtons.YesNo);
    if(result == DialogResult.Yes)
    {
        DatabaseManager.deleteUser(this.userId);
        MessageBox.Show("Eintrag wurde gelöscht");
        this.Close();
    }
}
```

4.5 Das Verändern eines bestehenden Benutzers

4.5.1 Grundüberlegung

Das Verändern eines bestehenden Eintrags funktioniert nach dem gleichen Prinzip wie bei der Erstellung eines neuen Eintrags. Man kann sich durch das Suchfenster den gewünschten Eintrag auswählen, nun öffnet sich wieder die Form zum Einsehen der Daten, jedoch benötigen wir einen zusätzlichen Button, der die Daten aktualisiert. Somit existieren nun drei Buttons auf diesem Formular. Der "Erstellen" Button (ausgeblendet), der "Update" Button und der "Löschen" Button.

4.5.2 Prozessablauf

Der Ablauf ist nahezu identisch mit der Erstellung eines Eintrags. Nachdem alle gewünschten Daten verändert worden sind, werden diese auf Fehler und Korrektheit überprüft. Sollte dabei ein Fehler entstehen, wird eine entsprechende Fehlermeldung ausgegeben. Falls nicht,

wird der Eintrag zum Datenbankmanager geschickt, dieser erstellt nun keinen neuen Eintrag, sondern aktualisiert den vorhandenen Eintrag. Er baut aus den überreichten Daten einen SQL-Befehl und leitet ihn an die Datenbank weiter.

```
sql = "UPDATE users SET vorname='"
+ user.Vorname + "' , nachname='"
+ user.Nachname + "' , strasse='"
+ user.Strasse + "' , telefon='"
+ user.Telefon + "' , email='"
+ user.Email + "' , hausnummer='"
+ user.Hausnummer + "' , ortId='"
+ user.OrtID + "' WHERE userId=" + user.UserID;
```

5 Die Datenbank

Die Datenbank ist das Fundament unserer Anwendung. In ihr werden alle Datensätze gespeichert, die von dem Benutzer jemals eingetragen worden sind. Daher ist es auch wichtig, dass effizienter Umgang mit ihr gepflegt wird

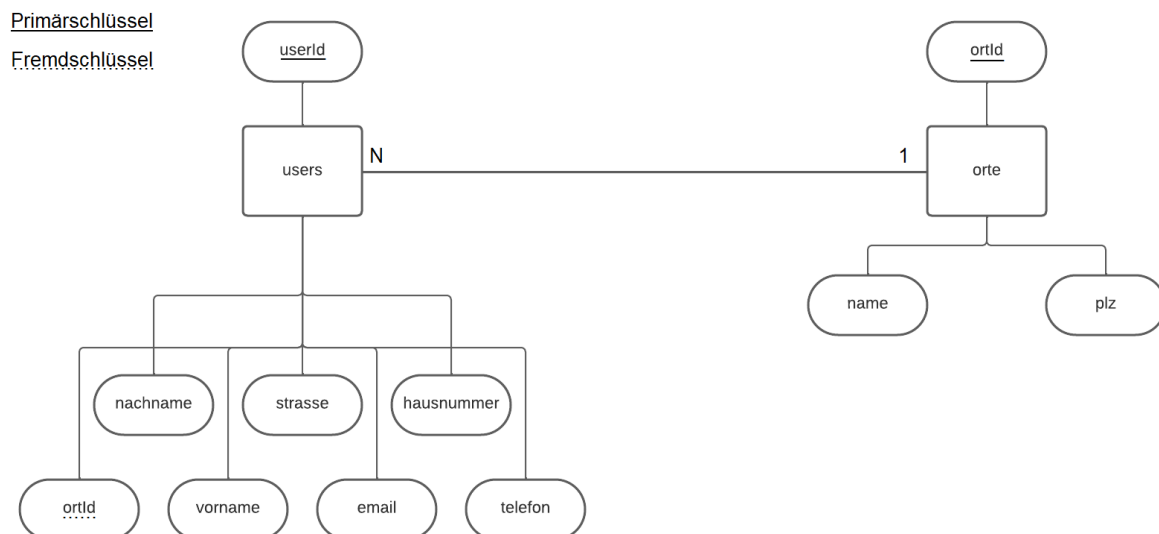
5.1 Schnittstelle

Der Datenbankmanager ist das Bindeglied zwischen Datenbank und GUI. Er führt SQL-Anweisungen aus und gibt das Ergebnis an die Oberfläche zurück. Der Datenbankmanager wurde als einzelne Klasse ausgelagert und liefert alle wichtigen Methoden zum kommunizieren mit der Datenbank

5.2 Design

Um eine Datenbank effizient nutzen zu können, muss diese auch auf einer klaren Struktur aufgebaut sein. Sie sollte allen drei Normalformen entsprechen und muss daher gut geplant sein. Sollte die Datenbank nicht normalisiert sein, kann es zu doppelten Einträgen, höheren Speicherkosten und Leistungseinbußen kommen.

5.2.1 ER-Modell



Relationenschreibweise:

Primärschlüssel

Fremdschlüssel

Users(userId, vorname, nachname, strasse, hausnummer, email, telefon, ortld)

Orte(ortld, name, plz)

5.4 Datenbankmanager

Der Datenbankmanager ist unsere zentrale Komponente, die für jegliche Kommunikation mit der Datenbank verantwortlich ist. Er stellt alle nötigen Funktionen bereit, um Datensätze zu kreieren, zu modifizieren oder zu entfernen. Er baut sich aus den übergebenen Parametern einen SQL-Befehl und schickt diesen an die Datenbank. Er ist das zentrale Stellglied für die Datenbank.

5.4.1 Datensatz erstellen

Um einen neuen Eintrag in der Datenbank erstellen zu können, braucht er die benötigten Attribute wie Vorname, Nachname, Telefonnummer etc. Diese werden von einem Formular an die Funktion übergeben.

Da die Eingabe eines Stadtnamens und einer Postleitzahl optional ist, muss überprüft werden, ob in dem Formular eine Postleitzahl / ein Stadtname eingetragen wurde. Falls ein Stadtname eingetragen wurde, wird überprüft, ob diese Stadt bereits in der Datenbank vorhanden ist, wenn ein neuer Ort in der Datenbank erstellt wird und der Eintrag bekommt eine Referenz in Form einer Identifikationsnummer auf den Ort. Sollte der Ort bereits existieren, wird die Identifikationsnummer dieses Ortes ausfindig gemacht und in den Eintrag geschrieben. Sollte der Benutzer keinen Ortsnamen oder Postleitzahl in ein Textfeld hineingeschrieben haben, so wird keine Referenz erstellt und als Identifikationsnummer wird NULL eingetragen.

```
if(newUser.OrtName != null && newUser.OrtName != string.Empty && newUser.OrtName != "-")
{
    int ortId;
    try
    {
        Ort ort = getOrtByAttributes(newUser.OrtName, newUser.Plz);
        if(ort == null)
        {
            createOrt(newUser.OrtName, newUser.Plz);
            ortId = int.Parse(getOrtByAttributes(newUser.OrtName, newUser.Plz).ortid);
        }
        else
        {
            ortId = int.Parse(ort.ortid);
        }
    }
    catch (Exception ex)
    {
        // City not found --> Create New City entry
        createOrt(newUser.OrtName, newUser.Plz);
        ortId = int.Parse(getOrtByAttributes(newUser.OrtName, newUser.Plz).ortid);
    }
    newUser.OrtID = ortId.ToString();
}
```

Ist jedoch ein Ort sowie eine Postleitzahl eingetragen, so kann diese einfach in den SQL-Befehl eingefügt werden. Anschließend wird anhand der existierenden Informationen ein SQL-Befehl gebaut, der von der Datenbank ausgeführt werden kann.

```
string insertSQL =
    "INSERT INTO users " +
    "(vorname,nachname,strasse,hausnummer,telefon,email,ortId) " +
    "VALUES " +
    "(" +
    + newUser.Vorname + "','" +
    + newUser.Nachname + "','" +
    + newUser.Strasse + "','" +
    + newUser.Hausnummer + "','" +
    + newUser.Telefon + "','" +
    + newUser.Email + "','" +
    + newUser.OrtID + "'" +
    + ")";
runCommand(insertSQL);
```

5.4.2 Datensätze auffinden

Einen Datensatz in der Datenbank aufzufinden ist relativ einfach. Der benötigte SQL-Befehl wird von einem Formular geschickt, das die Suchkomponente beherbergt. Daher wird die Funktion oft aufgerufen. Die Funktion führt den überreichten SQL-Befehl aus und transferiert die Ergebnisse in eine Liste von "User Entries". UserEntry ist eine eigene Klasse die einen Datenbankeintrag als Objekt darstellt

```
public static List<UserEntry> getUsersFromDatabase(string sql)
{
    MySqlCommand cmd = connection.CreateCommand();
    cmd.CommandText = sql;
    MySqlDataReader reader = cmd.ExecuteReader();
    List<UserEntry> list = new List<UserEntry>();
    while (reader.Read())
    {
        string userId = reader.GetString("userId");
        string vorname = reader.GetString("vorname");
        string nachname = reader.GetString("nachname");
        string strasse = reader.GetString("strasse");
        string hausnummer = reader.GetInt32("hausnummer").ToString();
        string telefon = reader.GetString("telefon");
        string email = reader.GetString("email");
        string ortid = "-";
        try
        {
            ortid = reader.GetString("ortId");
        } catch (Exception e) { }
        string ortname = "";
        string plz = "";

        list.Add(new UserEntry(userId, vorname, nachname,
            strasse, hausnummer, telefon, email, ortid, ortname, plz));
    }
    reader.Close();
    return list;
}
```

5.4.3 Datensätze löschen

Datensätze in einer Datenbank lassen sich unkompliziert löschen. Da jedem Eintrag eine eindeutige Identifikationsnummer zugewiesen ist, kann der gesamte Eintrag anhand der ID lokalisiert und gelöscht werden. Mit der übergebenen "userId"

```
public static void deleteUser(string userId)
{
    string sql = "DELETE FROM users WHERE userId= '"
        + userId + "'";
    MySqlCommand cmd = connection.CreateCommand();
    cmd.CommandText = sql;
    cmd.ExecuteReader().Close();
}
```

wird ein SQL-Befehl gebaut und zur Datenbank versendet. Diese löscht daraufhin den zu der ID korrespondierenden Eintrag. Anschließend wird der Befehl durch die Funktion "Close" geschlossen, um die benutzten Ressourcen wieder freizugeben.

5.4.4 Datensätze aktualisieren

Einen Datensatz zu aktualisieren ist um einiges komplizierter als die einfache Erstellung eines Datensatzes. Der "User Entry"

wird an die "updateUser" Funktion übergeben und enthält alle wichtigen Daten, die von den Textfeldern gesammelt wurden. Da die Daten "Postleitzahl" und "Ortsname" optional sind, müssen daher verschiedene Prozesse, je nach bereitgestellten

Daten, angestoßen werden. Wird eine Postleitzahl und ein Ortsname übergeben, so muss zuerst überprüft werden, ob dieser Ort bereits besteht und gegebenenfalls neu erstellt werden.

Sobald dies erledigt wurde, wird die "ortId" des neu erstellten Ortes herausgefunden und dem neuen Eintrag zugewiesen.

Sollten jedoch keine Angaben zu Ort und Postleitzahl gemacht werden, so wird der Eintrag zu dieser Person erstellt und erhält keine Referenz auf einen Ort, es wird "NULL" als "ortId" reingeschrieben.

```
sql = "UPDATE users SET vorname='"
+ user.Vorname + "' , nachname='"
+ user.Nachname + "' , strasse='"
+ user.Strasse + "' , telefon='"
+ user.Telefon + "' , email='"
+ user.Email + "' , hausnummer='"
+ user.Hausnummer + "' , ortId='"
+ user.OrtID + "' WHERE userId=" + user.UserID;
```

```
sql = "UPDATE users SET vorname='"
+ user.Vorname + "' , nachname='"
+ user.Nachname + "' , strasse='"
+ user.Strasse + "' , telefon='"
+ user.Telefon + "' , email='"
+ user.Email + "' , hausnummer='"
+ user.Hausnummer + "' , ortId=NULL WHERE userId="
+ user.UserID;
```

6 Soll-Ist-Vergleich

Zielbeschreibung	Status
Erstellen von Einträgen	erfüllt
Verändern von Einträgen	erfüllt
Löschen von Einträgen	erfüllt
Suchen von Einträgen	erfüllt
schöne, einheitliche und leicht zu benutzende Oberfläche	teilweise erfüllt

7 Ausblick

Das Projekt besitzt die notwendigen Grundfunktionalitäten, kann allerdings noch erweitert und ergänzt werden. Beispielsweise kann eine Filterfunktion eingefügt werden, sodass nur die gewünschten Attribute wie Vorname, E-Mail oder Telefon angezeigt werden.

Auf die Gestaltung der einzelnen Forms kann in Zukunft ebenfalls mehr Wert gelegt werden, da wir diese aufgrund der beschränkten Zeit sehr einfach gehalten haben. Es können beispielsweise Farben oder Bilder, bzw. Icons hinzugefügt werden. Zusätzlich kann das Design der Liste, in welcher die einzelnen Datensätze angezeigt werden, benutzerfreundlicher und ansprechender modelliert werden. Die Anwendung ist zwar einheitlich, leicht zu bedienen und zu verstehen, allerdings fehlt der Aspekt "Schönheit". Dieser könnte in Zukunft noch ausgiebig optimiert werden.

8 Fazit

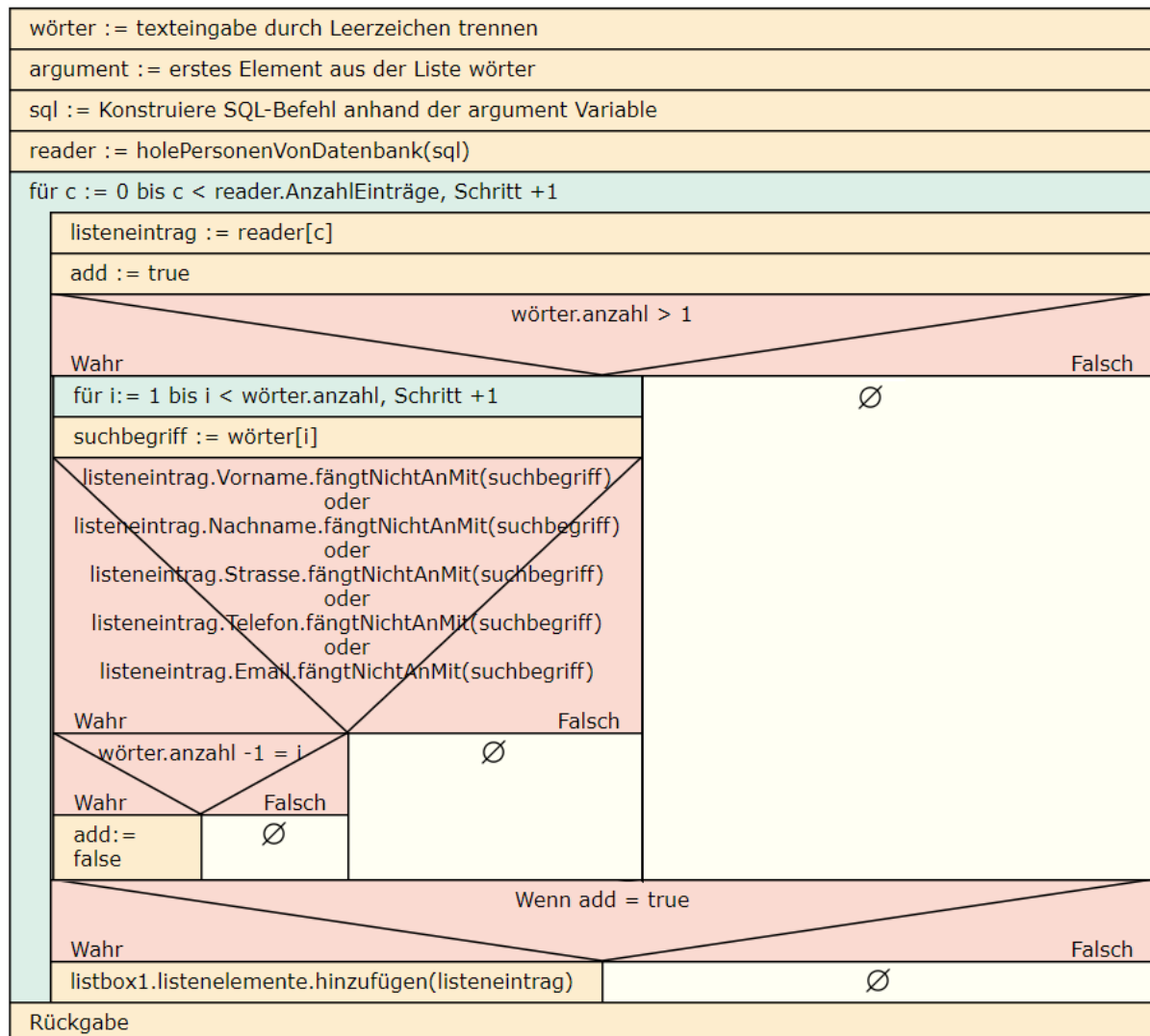
Das Projekt wurde im Großen und Ganzen gut gemeistert. Die Anwendung ist funktionstüchtig und User können sie benutzen. Aufgrund eines kurzen Zeitraumes lag der Fokus darauf, dass die Anwendung funktioniert und fertig wird. Daher wurde weniger darauf geachtet, ob sie schön gestaltet ist oder nicht. Das Ziel des Projekts war ein Verwaltungsprogramm, weshalb der Fokus auf die Funktionalität gesetzt wurde. Zu Beginn des Projekts lief alles gut und die Planung war super, jedoch wurden im Laufe der Zeit Themen thematisiert, die einen festen Standpunkt hätten haben sollen. Zum Ende des Projekts wurde es daher etwas knapp mit der Zeit. Daher Tipps für uns um es in Zukunft besser zu machen:

Mehr und besser kommunizieren und auch alle im Team über ein Thema einholen, damit jeder auf demselben Stand ist. Eventuell auch von Anfang an ein Ziel setzen bzw. planen, wie etwas aussehen soll.

9 Anhang

textBox1.textChanged(sender: object, e: EventArgs)

lokale Variablen:
 wörter:string[], argument:string, sql:string,
 reader:Liste, listeneintrag:object,
 add:Boolean, i:GZ, c:GZ, suchbegriff:string



10 Quellen

Cover

<https://www.codeguru.com/wp-content/uploads/2021/08/C-Sharp-Tutorials.png>

https://upload.wikimedia.org/wikipedia/de/thumb/d/dd/MySQL_logo.svg/1200px-MySQL_logo.svg.png