

# **Datenbank entwickeln und durch Desktop-Anwendungen darstellen in Rahmen Gesamtprojekt: «Softwarelösungen für Reisebüro».**

**IHK Nürnberg für Mittelfranken  
Sommerprüfung 2022**

**Auszubildender:**

**Andrii Dragozhynskyi**  
**Identnummer: XXXXXXXX**  
**Prüflingsnummer: XXXXX**  
**E-Mail:**  
[dragozhinskiy@yahoo.com](mailto:dragozhinskiy@yahoo.com)  
**Telefon: +49 1785499232**

**Ausbildungsbetrieb:**

**Lutz und Grub AG**  
**Frankenstraße 160**  
**90461 Nürnberg**

1	Analysephase .....	3
1.1	Kundengespräch .....	3
1.2	Ist-Analyse .....	4
1.3	Lösungsmöglichkeiten darstellen/ Alternativen .....	4
1.4	Projektziele definieren .....	4
2	Planungsphase .....	5
2.1	Projektplanung .....	5
2.2	Kostenkalkulation .....	5
3	Durchführung .....	5
3.1	Information strukturieren und Datenbank entwickeln .....	5
3.2	Anwendung entwickeln.....	7
3.3	Information in Datenbank hinzufügen .....	12
3.4	Verarbeitung die Darstellung der eingegebenen Informationen. ....	14
4	Projektergebnisse.....	14
4.1	Abschlusstest.....	14
4.2	Soll-Ist-Vergleich.....	14
4.3	Übergabe an den Kunden .....	14
5	Projektabschlussphase .....	14
5.1	Projektdokumentation .....	14
6	Glossar .....	14
7	Anhang .....	14

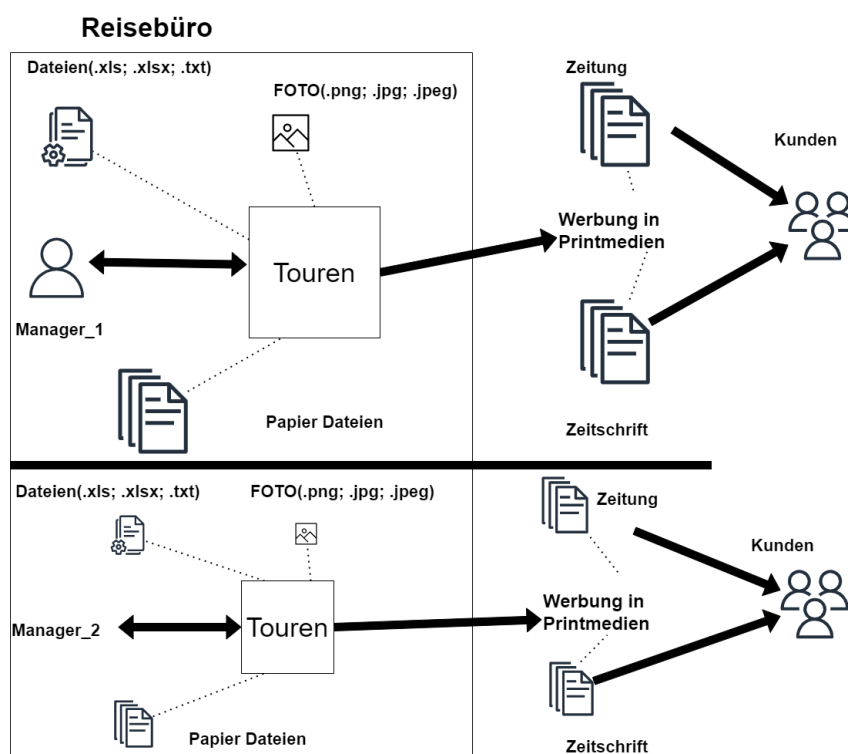
## 1 Analysephase

### 1.1 Kundengespräch

Das Unternehmen, für das ich arbeite, entwickelt und verwaltet Geschäftsanwendungen. In der Abteilung bin ich für die Entwicklung eines Frameworks für lokale Benutzeranwendungen verantwortlich, die Informationen erstellen und an lokale Datenbanken übergeben. Wir haben den Auftrag erhalten, eine Website von der „ReisenWelt“ zu erstellen, die Reisevermittlungsdienste anbietet.

Das Reisebüro ReisenWelt ist ein junges und schnell wachsendes Unternehmen mit erfahrenen Mitarbeitern. Derzeit verbreitet das Reisebüro Informationen über Tours durch Anzeigen in Printmedien. Das Unternehmen wollte seinen Kundenstamm durch die Einrichtung einer Website und Werbung in sozialen Netzwerken erweitern. Außerdem stellte sich heraus, dass jeder Mitarbeiter/in hat seine eigenen Informationen über bestimmte Touren getrennte von anderen Mitarbeiter, die in unterschiedlichen Formaten gespeichert sind. Einen Mitarbeiter/in man kann nur bestimmte Touren bieten, deswegen Kunden haben keine Möglichkeit allgemeinen Informationen über alles Touren bekommen.

Bei einem Gespräch mit einem Vertreter des Reisebüros im Rahmen eines Gesamt Projekte entstand die Idee, zuerst die Daten zu strukturieren und in eine Datenbank hinzufügen danach alle Daten in einer lokalen Desktop-Anwendung für die Mitarbeiter des Unternehmens darstellen.



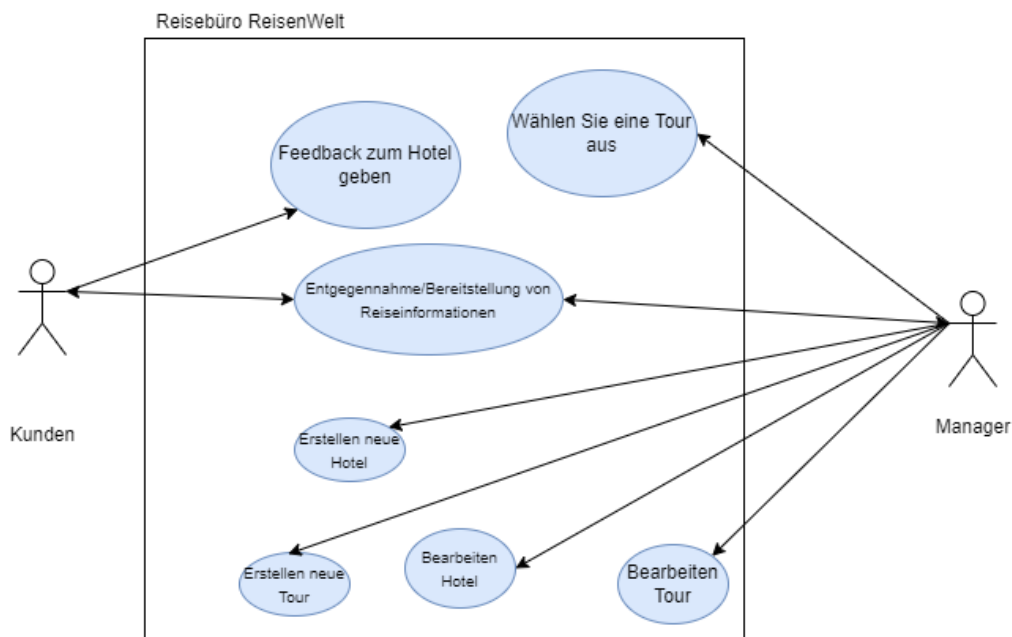


Abbildung 1: Use-Case Diagramm.

## 1.2 Ist-Analyse

## 1.3 Lösungsmöglichkeiten darstellen/ Alternativen

## 1.4 Projektziele definieren

Das Ziel dieses Projekts ist es, ein lokales Anwendungsframework mit WPF C# Technologie zu entwerfen und zu erstellen und eine strukturierte Datenbank auf Basis von Microsoft SQL Server mit MS SQL Management Studio zu erstellen, basierend auf Kundenanforderungen.

Entity Framework C# Technologie wird verwendet, um die Datenbank in der Anwendung zu behandeln.

## 2 Planungsphase

### 2.1 Projektplanung

Thema	Zeit
<b>1. Analysephase</b>	
Kundengespräch	0,5 Std
Ist-Analyse	0,5 Std
Lösungsmöglichkeiten darstellen/ Alternativen	0,5 Std.
Projektziele definieren	0,5 Std
<b>2. Planungsphase</b>	
Projektplanung	1,0 Std
Kostenkalkulation	1,0 Std
<b>3. Aus-/Durchführung</b>	
Information strukturieren und Datenbank entwickeln	3,0 Std
Anwendung entwickeln	3,0 Std
Information in Datenbank hinzufügen	3,0 Std
Verarbeitung die Darstellung der eingegebenen Informationen.	2,5 Std
<b>4. Projektergebnisse</b>	
Abschlusstest	1,0 Std
Soll-Ist-Vergleich	0,5 Std
Übergabe an den Kunden	1,0 Std
<b>5. Projektabschlussphase</b>	
Projektdokumentation	2,0 Std
<b>Gesamtzeit</b>	<b>20,0 Std</b>

Abbildung 2: Phasen Plan.

### 2.2 Kostenkalkulation

## 3 Durchführung

### 3.1 Information strukturieren und Datenbank entwickeln

Zur Durchführung des Projekts stand mir ein Arbeitsplatzrechner mit Windows 10 zur Verfügung, auf dem **Microsoft Visual Studio 2022**, **Microsoft SQL Server 2019** und **Microsoft SQL Server Management Studio 18** vorinstalliert waren. Bei der Entwicklung der Software habe ich die folgenden allgemeinen Informationsquellen verwendet, wie <https://docs.microsoft.com/>, <https://stackoverflow.com/> und <https://metanit.com/>.

Auf der Grundlage des Anwendungsfalldiagramms identifizierte ich die wichtigsten Entitäten und ihre Attribute und teilte die Entitäten in zwei Gruppen ein: Die erste Gruppe umfasst Touren und Type und die zweite Gruppe umfasst Hotel, HotelComment, HotelImage und Country. Dann setze

ich die einzelnen Einheiten in Beziehung zueinander. Da in dem entwickelten Diagramm eine "n zu m" Beziehung zwischen den Tabellen Tour und Type besteht, die nicht der dritten Normalform entspricht, musste ich eine weitere Tabelle TypeOfTour hinzufügen, ähnlich verhält es sich mit der Beziehung zwischen den Tabellen Tour und Hotel. Ich musste auch eine zusätzliche Tabelle HotelOfTour hinzufügen (Siehe unter: Abbildung 3: Database Diagramm).

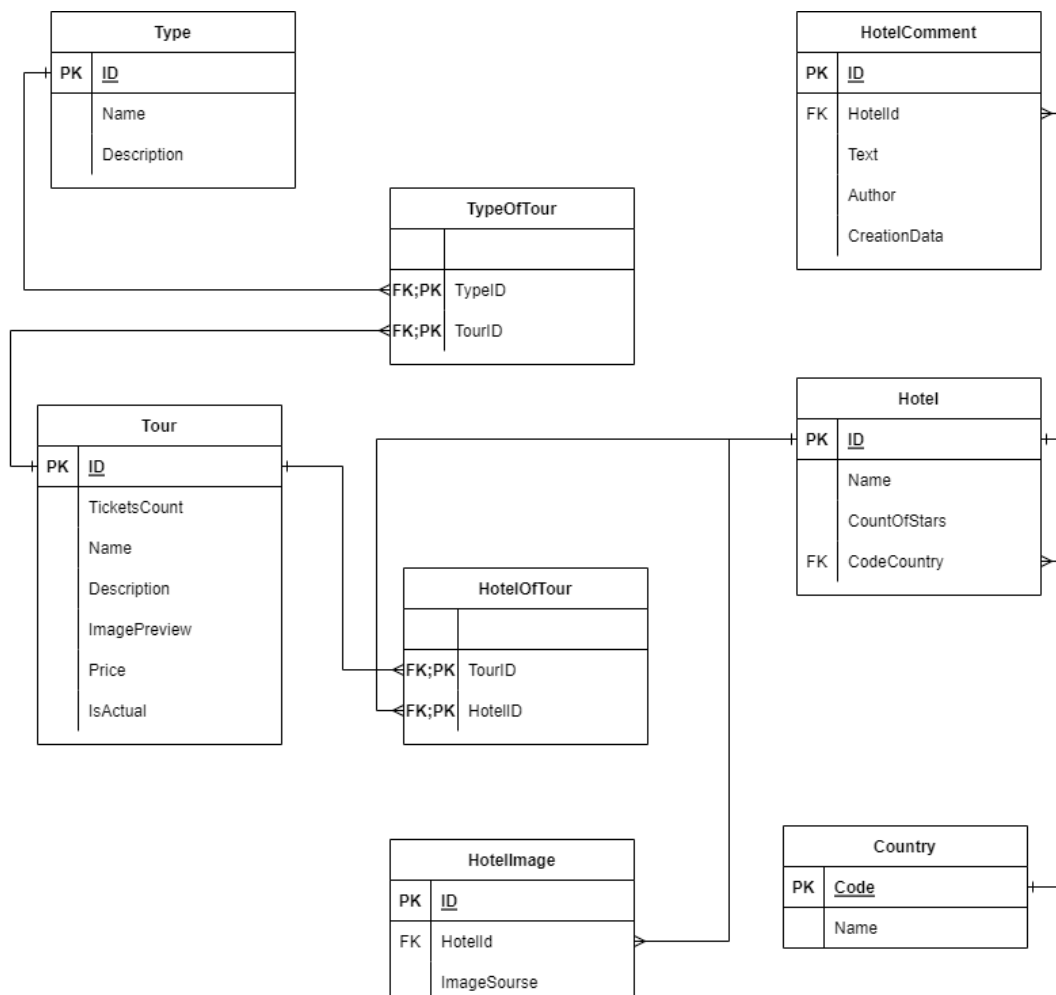


Abbildung 3: Database Diagramm.

Danach habe ich alle Tabellen mit beide Zwischentabellen (TypeOfTour, HotelOfTour) bei der Erstellung der Datenbank mit Hilfe von **Database Diagrams (Microsoft SQL Server Management Studio 18)** implementiert. Für jedes Attribut habe ich den entsprechenden Datentyp eingestellt: für die Tabelle Tour habe ich dem Attribut **ID** den Datentyp **int** zugewiesen, für Name **nvarchar(100)**, für TicketCount **int**, für Description **nvarchar(MAX)**, für ImagePreview **varbinary(MAX)**, für Price **money**, für **IsActual** bit usw. Als nächstes werde ich in der Spalte Allow Nulls für einige Attribute Nullwerte zulassen.

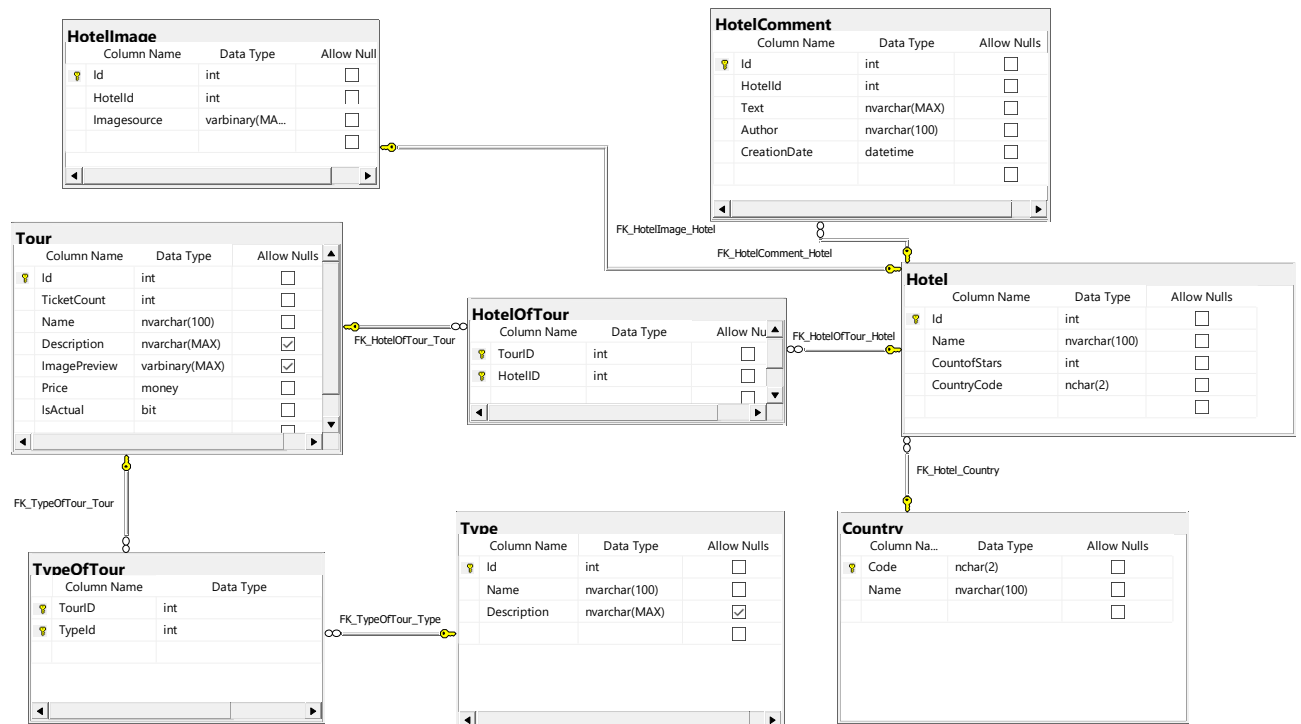


Abbildung 4:

### 3.2 Anwendung entwickeln

Ich verwende **WPF-App(.NET Framework Version 4.7.2)** als GUI für die grafische Darstellung von Informationen sowie für die Datenverarbeitung und -verwaltung.

Zunächst unterteile ich das Hauptfenster mit GRID in 3 Teile, dann wähle ich auf der Registerkarte Ressourcen eine Liste von Bildern aus und füge ein Logo ein. Ich habe das Feld Build Action auf Resource gesetzt. Ich verwende das Element Image, um das Logo anzuzeigen. Ich verwende TextBlock, um die Überschrift "ReisenWelt" zu setzen. Legen Sie den Hintergrund für den oberen und unteren Teil fest. Im Hauptanwendungsfenster platziere ich ein **Frame**-Element, in dem andere Anwendungsseiten zusammengestellt werden. Um die erstellten Elemente in der Anwendung zu verwenden, geben Sie einen Namen für den Frame "MainFrame" an.

```

9  <Grid>
10 <Grid.RowDefinitions>
11 <RowDefinition Height="75"></RowDefinition>
12 <RowDefinition Height="*" />
13 <RowDefinition Height="30" />
14 </Grid.RowDefinitions>
15 <Image Source="/Resources/Logo.png" HorizontalAlignment="Left"></Image>
16 <TextBlock Text="ReisenWelt" FontSize="30" HorizontalAlignment="Center" VerticalAlignment="Center"></TextBlock>
17 <Grid Background="#bae3e8" Panel.ZIndex="-2"></Grid>
18 <Grid Grid.Row="2" Background="#445c93"></Grid>
19 <Frame NavigationUIVisibility="Hidden" ContentRendered="MainFrame_ContentRendered" Grid.Row="1" Name="MainFrame"></Frame>
20 <Button Content="Back" Name="BtnBack" HorizontalAlignment="Right" Click="BtnBack_Click"></Button>
21 </Grid>

```

Abbildung 5:

Als nächstes erstelle ich neue Seiten namens HotelsPage und AddEditPage und setzen die Markierungen.

Die Seite HotelsPage ist in 4 Spalten unterteilt, wobei die ersten drei Spalten den Hotelnamen, die Anzahl der Sterne und das Land anzeigen. Ich gebe in der Eigenschaft „Binding“ die Daten an, die beim Laden angezeigt werden sollen. In der vierten Spalte wird eine Schaltfläche zum Ändern weiterer Informationen über das Hotel hinzugefügt. Am unteren Rand der Seite befinden sich zwei Schaltflächen für "Hotels hinzufügen" und "Hotels entfernen". Um keine neuen Spalten mit zusätzlichen Informationen zu generieren (alle verfügbaren Eigenschaften dieser Objekte), setzen wir die Eigenschaft `AutoGenerateColumns="false"` im `DataGrid`, setzen wir die Eigenschaft `ReadOnly = "true"`, um die Bearbeitung der Spalten zu verhindern.

```

...<Grid>
→ <Grid.RowDefinitions>
→ <RowDefinition Height="377*" /></RowDefinition>
→ <RowDefinition Height="50" /></RowDefinition>
→ </Grid.RowDefinitions>
→ <DataGrid x:Name="DGridHotels" AutoGenerateColumns="False" IsReadOnly="True">
→ <DataGrid.Columns>
→ <DataGridTextColumn Header="Name des Hotels" Binding="{Binding Name}" Width="*" /></DataGridTextColumn>
→ <DataGridTextColumn Header="Stern" Binding="{Binding CountOfStars}" Width="150" /></DataGridTextColumn>
→ <DataGridTextColumn Header="Staat" Binding="{Binding Country.Name}" Width="250" /></DataGridTextColumn>
→ <DataGridTemplateColumn Width="auto">
→ <DataGridTemplateColumn.CellTemplate>
→ <DataTemplate>
→ <Button Content="ändern" Name="BtnEdit" Click="BtnEdit_Click" /></Button>
→ </DataTemplate>
→ </DataGridTemplateColumn.CellTemplate>
→ </DataGridTemplateColumn>
→ </DataGrid.Columns>
→ </DataGrid>

→ <Button Content="Addieren" Grid.Row="1" HorizontalAlignment="Left" Name="BtnAdd" Click="BtnAdd_Click" /></Button>
→ <Button Content="Loeschen" Grid.Row="1" HorizontalAlignment="Right" Name="BtnLoesch" Click="BtnLoesch_Click" /></Button>
...</Grid>

```

Abbildung 6: HotelsPage

### Implementierung der Addieren Funktion

Button "Addieren" (BtnAdd\_Click)-Ich greife auf Frame zu und rufe die Methode **Navigate** auf. Ich verwende diese Methode, um zur AddEditPage zu navigieren.

### Implementierung der Bearbeitungsfunktion.

Für die Datenbearbeitungsfunktion ist es ratsam, dieselbe Seite zu verwenden, die ich für das Hinzufügen verwendet habe. Wenn der Benutzer beabsichtigt, Informationen über ein Objekt zu ändern, zeigt das System eine Hinzufügeseite mit Informationen über das zu bearbeitende Objekt an. Die geänderten Informationen werden in die Datenbank übernommen und in der Liste so angezeigt, wie sie beim Hinzufügen waren. Zunächst füge ich einen Parameter zu unserer AddEditPage hinzu. Darin übergebe ich eine Instanz des ausgewählten Hotels und weise sie, wenn sie nicht leer ist, unserem Feld `CurrentHotel` zu.

### Implementierung der Löschfunktion

Ein separates Fenster ist nicht erforderlich, um die Löschfunktion zu implementieren. Dies wird die Entwicklungszeit erheblich verkürzen, aber das Löschen erfordert besondere Aufmerksamkeit. Jede Aktion, die Daten in der Datenbank unwiderruflich ändert, muss vom Benutzer bestätigt werden. Deshalb wird eine Meldung eingefügt, in der der Benutzer gefragt wird, ob er wirklich auf die



Schaltfläche "Löschen" klicken möchte. Ich erhalte also die Liste der zu löschenden Hotels durch Zugriff auf die Hoteltabelle. Wir wählen alle markierten Punkte aus und wandeln sie in eine Hotelliste um. Und dann fragen wir den Benutzer in der Nachricht: "Sind Sie sicher, dass Sie die folgenden HotelsFuerLoeschen.Count() löschen möchten?". Hier geben wir den Titel der Nachricht an - "Achtung", dann legen wir fest, welche Schaltflächen verfügbar sind, wenn wir mit dem Benutzer sprechen: "Ja" oder "Nein", und wählen ein Bild - "Frage"

```

1 Verweis
private void BtnEdit_Click(object sender, RoutedEventArgs e)
{
    Manager.MainFrame.Navigate(new AddEditPage((sender as Button).DataContext as Hotel));
}

1 Verweis
private void BtnAdd_Click(object sender, RoutedEventArgs e)
{
    Manager.MainFrame.Navigate(new AddEditPage(null));
}

1 Verweis
private void BtnLoesch_Click(object sender, RoutedEventArgs e)
{
    var HotelsFuerLoeschen = DGridHotels.SelectedItems.Cast<Hotel>().ToList();
    if (MessageBox.Show($"Sind Sie sicher folgendes {HotelsFuerLoeschen.Count()} Artikel/n loeschen?", "Achtung!!!",
        MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)
    {
        try
        {
            ToursEntities.GetContext().Hotels.RemoveRange(HotelsFuerLoeschen);
            ToursEntities.GetContext().SaveChanges();
            MessageBox.Show("Artikel/n sind geloescht");

            DGridHotels.ItemsSource = ToursEntities.GetContext().Hotels.ToList();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }
}

```

Abbildung 7:

Die AddEditPage wird in 2 Spalten und 4 Zeilen unterteilt. In der linken Spalte steht der Name, in der rechten Spalte die Informationen, die ich eingeben oder bearbeiten möchte. Ganz unten befindet sich die Schaltfläche "Speichern". Um die ComboBox im Code zu verarbeiten, gebe ich ihr den Namen "ComboStaat". Dann gebe ich mit "Binding" an, auf welche Eigenschaft beim Laden von Daten verwiesen werden soll

```

<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="200"/>
    <ColumnDefinition Width="*"/>
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="*"/>
    <RowDefinition Height="*"/>
    <RowDefinition Height="*"/>
    <RowDefinition Height="*"/>
  </Grid.RowDefinitions>
  <TextBlock Text="Name"></TextBlock>
  <TextBlock Text="Anzahl der Stern" Grid.Row="1"></TextBlock>
  <TextBlock Text="Staat" Grid.Row="2"></TextBlock>
  <TextBox Text="{Binding Name}" MaxLength="100" Grid.Column="1"></TextBox>
  <TextBox Text="{Binding CountofStars}" Grid.Column="1" Grid.Row="1" Width="175" HorizontalAlignment="Left"></TextBox>
  <ComboBox SelectedItem="{Binding Country}" Grid.Row="2" Grid.Column="1" x:Name="ComboStaat" DisplayMemberPath="Name"></ComboBox>
  <Button Content="Speichern" Grid.ColumnSpan="2" Grid.Row="3" Name="BtnSpeichern" Click="BtnSpeichern_Click"></Button>
</Grid>

```

Abbildung 8: AddEditPage

Um Informationen in die ComboBox zu laden, rufen wir ComboStaat über die Eigenschaft ItemsSource auf und holen eine Liste von Ländern aus unserem Modell.

Um den Zusatz zu implementieren, fügen wir ein neues Feld hinzu, das eine leere Instanz „\_currentHotel“ des hinzuzufügenden Hotels enthält.

```
→ private Hotel _currentHotel = new Hotel();  
→ 2 Verweise  
→ public AddEditPage(Hotel selectedHotel)  
→ {  
→     InitializeComponent();  
→     if (selectedHotel != null)  
→     {  
→         _currentHotel = selectedHotel;  
→         DataContext = _currentHotel;  
→     }  
→     |  
→     ComboStaat.ItemsSource = ToursEntities.GetContext().Countries.ToList();  
→ }
```

Abbildung 9:

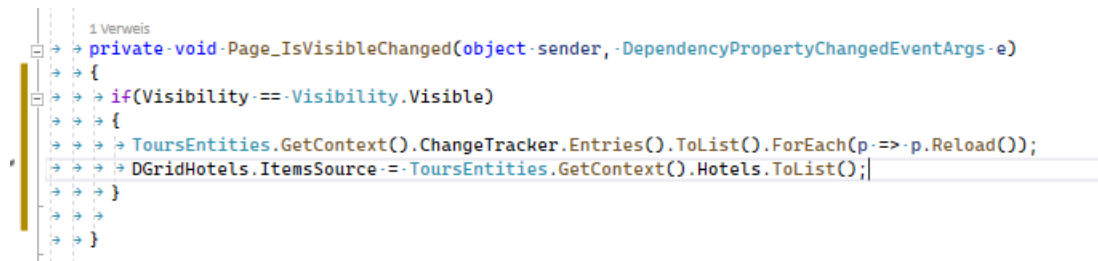
Funktionsbearbeitung für die Schaltfläche "Speichern."

Überprüfe Ich zunächst die Bedingungen für die Speicherung: Die Daten in den Feldern Hotelname und Land dürfen nicht leer sein, und die Anzahl der Hotelsterne muss zwischen 1 und 5 liegen. Ich habe dann eine Methode zum Hinzufügen eines neuen Hotels zur Hotelliste erstellt und die Speichermethode in die "try catch"-Methode eingefügt. Falls beim Speichern ein unsichtbarer Fehler auftritt, erhalte ich eine Meldung. Im Falle einer erfolgreichen Speicherung erhalte ich außerdem die Meldung "Information hat gespeichert!". Kehren Sie dann zur ursprünglichen Hotelseite zurück.

```
→ 1 Verweis  
→ private void BtnSpeichern_Click(object sender, RoutedEventArgs e)  
→ {  
→     StringBuilder errors = new StringBuilder();  
→     if (string.IsNullOrEmpty(_currentHotel.Name))  
→     {  
→         errors.AppendLine("Geben Sie bitte Name des Hotel ein.");  
→     }  
→     if (_currentHotel.CountofStars < 1 || _currentHotel.CountofStars > 5)  
→     {  
→         errors.AppendLine("Anzahl der Stern von 1 bis 5");  
→     }  
→     if (_currentHotel.Country == null)  
→     {  
→         errors.AppendLine("Wählen Sie bitte einen Staat");  
→     }  
→     if (errors.Length > 0)  
→     {  
→         {  
→             MessageBox.Show(errors.ToString());  
→             return;  
→         }  
→     }  
→     if (_currentHotel.Id == 0)  
→     {  
→         ToursEntities.GetContext().Hotels.Add(_currentHotel);  
→     }  
→     try  
→     {  
→         ToursEntities.GetContext().SaveChanges();  
→         MessageBox.Show("Information hat gespeichert!");  
→         Manager.MainFrame.GoBack();  
→     }  
→     catch (Exception ex)  
→     {  
→         {  
→             MessageBox.Show(ex.Message.ToString());  
→         }  
→     }  
→ }
```

Abbildung 10:

Nachdem Sie zur ursprünglichen Hotelseite zurückgekehrt sind, müssen Sie die aktuellen Informationen anzeigen. Um die Liste in der Tabelle zu aktualisieren, verwende ich das Ereignis "IsVisibleChanged" in die Seite HotelsPage, das jedes Mal ausgelöst wird, wenn die Seite angezeigt oder ausgeblendet wird.



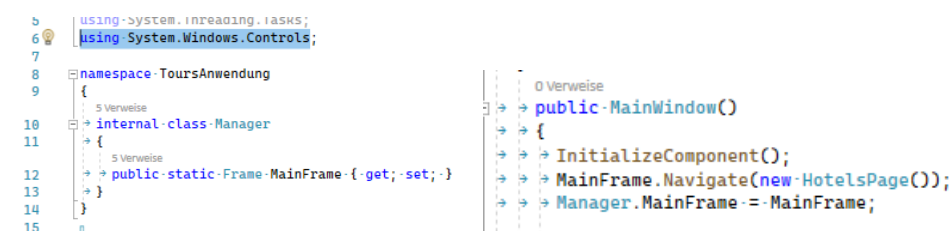
```

1 Verweis
→ private void Page_IsVisibleChanged(object sender, -DependencyPropertyChangedEventArgs e)
→ {
→     if(Visibility == -Visibility.Visible)
→     {
→         ToursEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());
→         DGridHotels.ItemsSource = -ToursEntities.GetContext().Hotels.ToList();
→     }
→ }

```

Abbildung 11:

Um die Navigation von "MainFrame" zu steuern, erstelle ich eine Klasse mit dem Namen "Manager" und erstellen eine statische Eigenschaft für MainFrame. Ich füge ein Namensfeld "using System.Windows.Controls" ein und in der Hauptfensterlogikdatei MainWindow.xaml.cs einen Wert zuweisen.



```

5 using System.Linq;
6 using System.Windows.Controls;
7
8 namespace ToursAnwendung
9 {
10     5 Verweise
11     internal class Manager
12     {
13         5 Verweise
14         public static Frame MainFrame { get; set; }
15     }
16 }

```

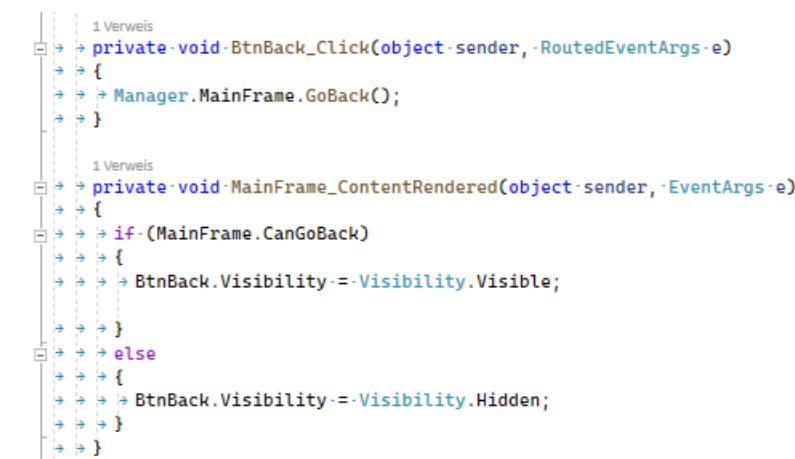
```

0 Verweise
→ public MainWindow()
→ {
→     InitializeComponent();
→     MainFrame.Navigate(new HotelsPage());
→     Manager.MainFrame = -MainFrame;
→ }

```

Abbildung 12 und 13:

Stellen Sie die Schaltfläche "Back" ein, um zur vorherigen Seite zurückzukehren. Verwendung der Methode "GoBack" bei der Ereignisbehandlung. Legen Sie auch die Sichtbarkeit dieser Schaltfläche im oberen Bedienfeld fest, falls es möglich ist, diese Schaltfläche zu verwenden.



```

1 Verweis
→ private void BtnBack_Click(object sender, -RoutedEventArgs e)
→ {
→     Manager.MainFrame.GoBack();
→ }

1 Verweis
→ private void MainFrame_ContentRendered(object sender, -EventArgs e)
→ {
→     if (MainFrame.CanGoBack)
→     {
→         BtnBack.Visibility = -Visibility.Visible;
→     }
→     else
→     {
→         BtnBack.Visibility = -Visibility.Hidden;
→     }
→ }

```

Abbildung 14:

In der Datei App.xaml erstelle ich eine Reihe von Eigenschaften für die meisten der verwendeten Elemente.

```

→ <Style-TargetType="Button">
→ <Setter-Property="Margin"-Value="5"></Setter>
→ <Setter-Property="Width"-Value="175"></Setter>
→ <Setter-Property="Height"-Value="30"></Setter>
→ <Setter-Property="Background"-Value="#f716"></Setter>
→ </Style>
→ <Style-TargetType="TextBlock">
→ <Setter-Property="VerticalAlignment"-Value="Center"></Setter>
→ <Setter-Property="HorizontalAlignment"-Value="Center"></Setter>
→ </Style>
→ <Style-TargetType="TextBox">
→ <Setter-Property="VerticalAlignment"-Value="Center"></Setter>
→ <Setter-Property="HorizontalAlignment"-Value="Stretch"></Setter>
→ <Setter-Property="Height"-Value="30"></Setter>
→ <Setter-Property="VerticalContentAlignment"-Value="Center"></Setter>
→ <Setter-Property="Margin"-Value="5"></Setter>
→ </Style>
→ <Style-TargetType="ComboBox">
→ <Setter-Property="VerticalAlignment"-Value="Center"></Setter>
→ <Setter-Property="HorizontalAlignment"-Value="Stretch"></Setter>
→ <Setter-Property="Height"-Value="30"></Setter>
→ <Setter-Property="VerticalContentAlignment"-Value="Center"></Setter>
→ <Setter-Property="Margin"-Value="5"></Setter>
→ </Style>

```

Abbildung 15:

### 3.3 Information in Datenbank hinzufügen

Installieren Sie ein **ADO.NET-Entity-Datenmodell** mit dem Namen "BaseModel" und stellen Sie eine Verbindung zu unserer Datenbank her. Um auf das "BaseModel" zuzugreifen, verwenden Sie das Singleton-Pattern. In der Datei **BaseModel.Context.cs** füge ich ein privates statisches Feld `_context` und die Methode **GetContext** hinzu.

```

...{
...private static ToursEntities _context; //Komment
1 Verweis
} ...public ToursEntities()
...: base("name=ToursEntities")
...{
...}
13 Verweise
} ...public static ToursEntities GetContext() //Komment
...{
...if (_context == null)
..._context = new ToursEntities();
...return _context;
...}

```

Abbildung 16:

Unsere Informationen sind unstrukturiert gespeichert und werden in Form von Text und großen, ungeordneten Tabellen präsentiert.

Neben dem manuellen Hinzufügen von Daten in die Datenbank ist auch eine Datenverarbeitung und ein Datenimport erforderlich. Dateien und Daten werden in verschiedenen Formaten zum Import angeboten. Im Moment haben wir zum Beispiel:

- eine Liste der Reisearten im Papierformat
- Länder im csv-Format
- eine Liste der Hotels im xls-Format

- Touren im Unicode text-Format
- Liste der Tourbilder als Bilder.

Import eine Liste von Reisearten in die Datenbank.

Übertrage ich die Liste der Reisetypen in Excel, formatieren Sie sie und fügen Sie sie über Microsoft SQL Server Management Studio 18 in unsere Datenbank ein.

Import eine Liste von Ländern in die Datenbank.

Klicke ich auf den Namen der Datenbank "Tasks — Import Data". Wähle ich eine unstrukturierte Datei als Datenquelle. Dann wähle ich Dateityp csv-Dateien aus. Entferne ich die Klausel, dass die erste Zeile die Überschrift ist. Als nächstes wähle ich aus, wohin ich die Daten importieren möchte. Dann wähle ich die Tabelle aus, in der die Daten gespeichert werden sollen. Ich drücke auf Weiter - Weiter – Beenden.

Import einer Hotelliste in die Datenbank

Möglicherweise gibt es Duplikate in den Tabellen, die ich prüfe und lösche. Dann werde ich die Tabelle formatieren, indem ich die Wertespalten in der folgenden Reihenfolge einstelle: Name, CountofStars, CountryCode, und dann den Ländercode in der Tabelle und in der Datenbank auf Konsistenz vergleichen und gegebenenfalls korrigieren und fügen Sie sie über Microsoft SQL Server Management Studio 18 in unsere Datenbank ein.

Import die Tourentabelle in die Datenbank.

```
private void ImportTours()
{
    var fileData = File.ReadAllLines(@"D:\Generell\AE_Vertiefung\02072022\Touren.txt");
    var images = Directory.GetFiles(@"D:\Generell\AE_Vertiefung\02072022\FotoTours");

    foreach (var line in fileData)
    {
        var data = line.Split('\t');

        var tempTour = new Tour
        {
            Id = ToursEntities.GetContext().Tours.Count(),
            Name = data[0].Replace("\", "\"),
            TicketCount = int.Parse(data[2]),
            Price = decimal.Parse(data[3]),
            IsActual = (data[4] == "0") ? false : true
        };

        foreach (var tourType in data[5].Split(new string[] { ",", "." }, StringSplitOptions.RemoveEmptyEntries))
        {
            var currentType = ToursEntities.GetContext().Types.ToList().FirstOrDefault(p => p.Name == tourType);
            if (currentType != null)
            {
                tempTour.Types.Add(currentType);
            }
        }

        try
        {
            tempTour.ImagePreview = File.ReadAllBytes(images.FirstOrDefault(p => p.Contains(tempTour.Name)));
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }

        if (!ToursEntities.GetContext().Tours.Equals(tempTour.Name))
        {
            ToursEntities.GetContext().Tours.Add(tempTour);
            ToursEntities.GetContext().SaveChanges();
        }
    }
}
```

Ich erstelle zwei Variablen `var fileData` und `var images`. Ich lese die Daten in der Datei Zeile für Zeile, wobei ich die Daten mit Tabulatoren trenne, und erstelle eine Instanz der Klasse, indem ich die Eigenschaften mit den entsprechenden Werten fülle, wobei ich auf die Datentypen achte. Ausfüllen der Tourtypensammlung durch Suche nach Typen anhand der in der Datei durch Komma getrennten Namen. Ich schreibe das Bild mit der Methode `ReadAllBytes()` in die Datenbank. Ich füge die Tour in die Datenbank ein und speichere sie. Rufen Sie die Methode `ImportTours()` im `MainWindow`-Konstruktor auf und überprüfen Sie das Ergebnis in der Datenbank. Tourenliste und -typen werden importiert.

### 3.4 Verarbeitung die Darstellung der eingegebenen Informationen.

Um Informationen zur Tour anzuzeigen, fügen Sie eine neue Seite mit einer `ListView` hinzu.

## 4 Projektergebnisse

### 4.1 Abschlusstest

### 4.2 Soll-Ist-Vergleich

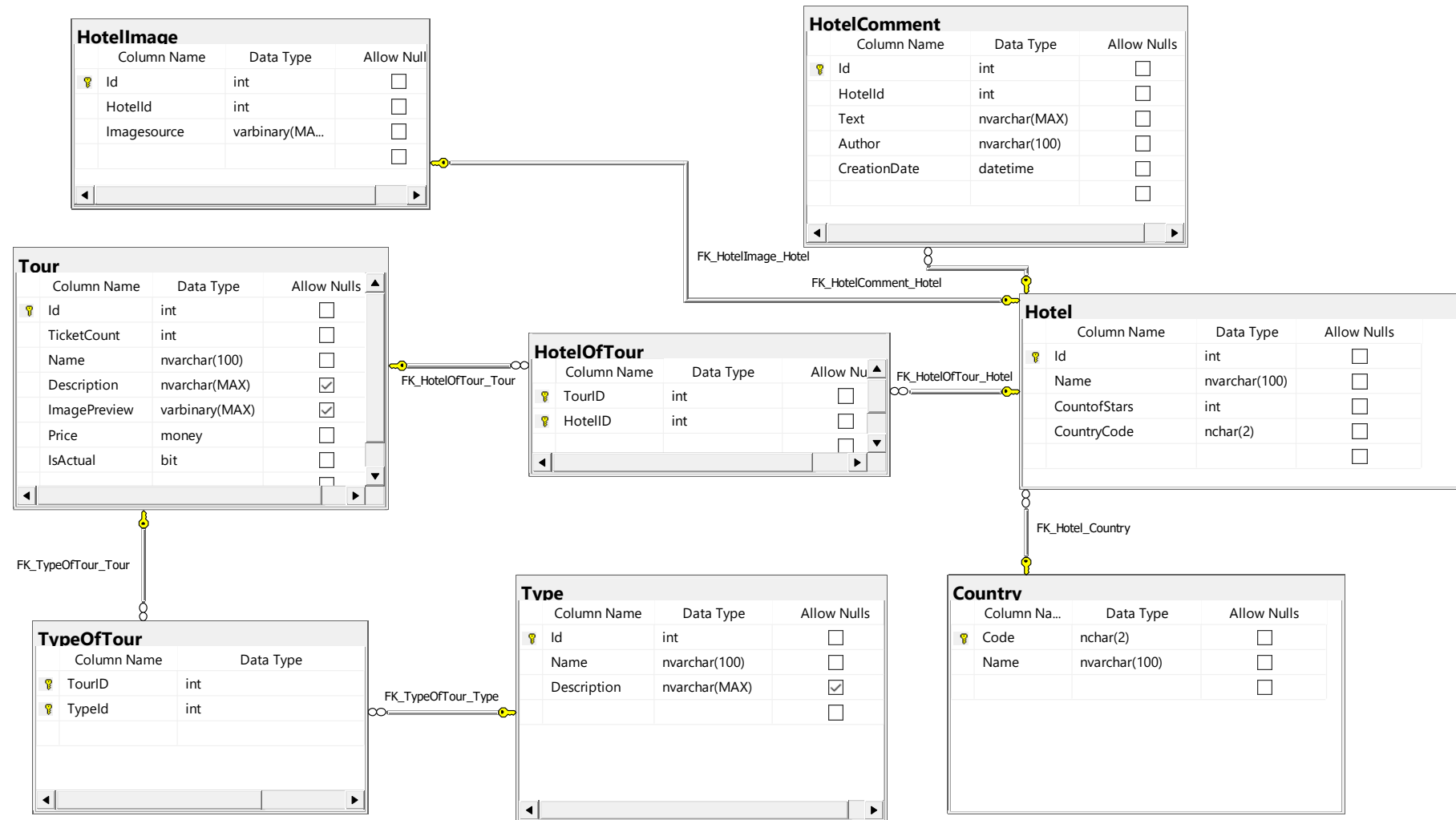
### 4.3 Übergabe an den Kunden

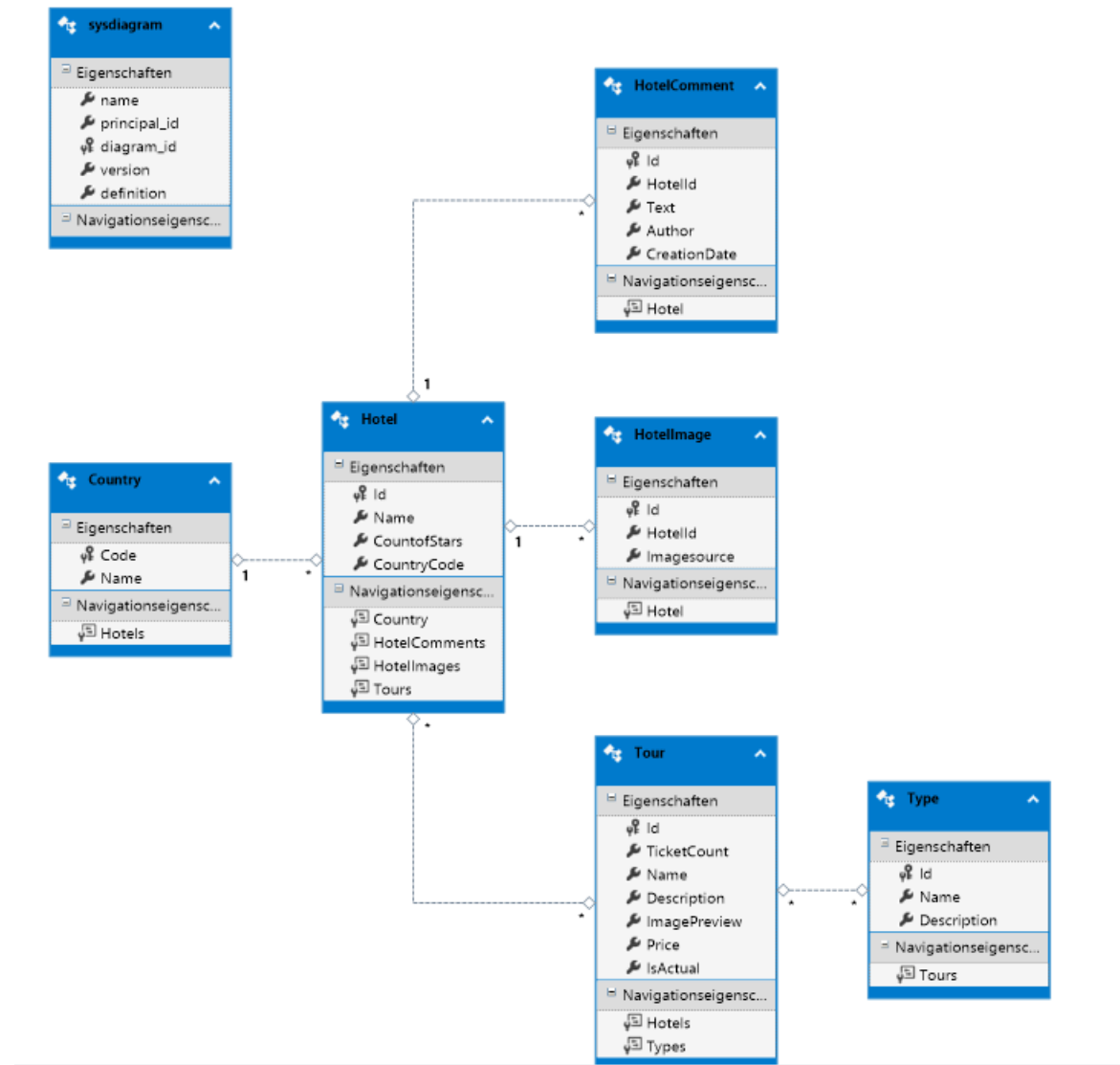
## 5 Projektabschlussphase

### 5.1 Projektdokumentation

## 6 Glossar

## 7 Anhang







```

--<Grid>
  <Grid.RowDefinitions>
    <RowDefinition.Height="auto"></RowDefinition>
    <RowDefinition.Height="*"></RowDefinition>
  </Grid.RowDefinitions>

  <WrapPanel.Orientation="Horizontal".HorizontalAlign="Center">
    <StackPanel.Orientation="Horizontal">
      <TextBlock.Text="Namen für die Suchung".Width="175".TextAlign="Right"></TextBlock>
      <TextBox.Width="225".Name="TextBoxSearch".TextChanged="TextBoxSearch_TextChanged"></TextBox>
    </StackPanel>
    <StackPanel.Orientation="Horizontal">
      <TextBlock.Text="Tours Type:".Width="175".TextAlign="Right"></TextBlock>
      <ComboBox.Width="225".Name="ComboType".SelectionChanged="ComboType_SelectionChanged".DisplayMemberPath="Name"></ComboBox>
    </StackPanel>
    <CheckBox.x.Name="CheckActual".Checked="CheckActual_Checked".Unchecked="CheckActual_Checked".Content="Zeigen nur aktuelle Touren".HorizontalAlign="Center".VerticalAlign="Center"></CheckBox>
  </WrapPanel>

  <ListView.Grid.Row="1".Name="LVViewTours".ScrollViewer.HorizontalScrollBarVisibility="Disabled".HorizontalContentAlignment="Center">
    <ListView.ItemsPanel>
      <ItemsPanelTemplate>
        <WrapPanel.Orientation="Horizontal".HorizontalAlign="Center"></WrapPanel>
      </ItemsPanelTemplate>
    </ListView.ItemsPanel>
    <ListView.ItemTemplate>
      <DataTemplate>
        <Grid.Margin="20".Width="400">
          <Grid.RowDefinitions>
            <RowDefinition.Height="70"></RowDefinition>
            <RowDefinition.Height="310"></RowDefinition>
            <RowDefinition.Height="auto"></RowDefinition>
            <RowDefinition.Height="auto"></RowDefinition>
          </Grid.RowDefinitions>
          <Image.Width="400".Grid.Row="1".Stretch="UniformToFill".HorizontalAlign="Center".Margin="5">
            <Image.Source>
              <Binding.Path="ImagePreview">
                <Binding.TargetNullValue>
                  <ImageSource.Resource/KeinFoto.png/>
                </Binding.TargetNullValue>
              </Binding>
            </Image.Source>
          </Image>
          <TextBlock.Text="{Binding.Name}".VerticalAlign="Center".TextAlign="Center".Width="390"
            <TextWrapping="Wrap".HorizontalAlign="Center".Margin="5.5".FontSize="26".Grid.Row="0"></TextBlock>
        </Grid>
      </DataTemplate>
    </ListView.ItemTemplate>
  </ListView>

```

```
<TextBlock.Text="{Binding.Name}"-VerticalAlignment="Center"-TextAlignment="Center"-Width="390"
-TextWrapping="Wrap"-HorizontalAlignment="Center"-Margin="5-5"-FontSize="26"-Grid.Row="0"></TextBlock>
<TextBlock.Text="{Binding.Price,-StringFormat={}{0:N2}-Euro}"-Grid.Row="2"-Margin="5-5-5-15"-HorizontalAlignment="Center"-FontSize="26"-FontWeight="Bold"></TextBlock>
<TextBlock.Text="{Binding.TicketCount,-StringFormat={}}Tickets-gibt-es-noch:-{0}"-Grid.Row="3"-FontSize="14"-HorizontalAlignment="Right"></TextBlock>
<TextBlock.Text="{Binding.ActualText}"-Grid.Row="3"-FontSize="14"-HorizontalAlignment="Left"></TextBlock>
</Grid>
</DataTemplate>
</ListView.ItemTemplate>
</ListView>
</Grid>
</Page>
```