C#

WCF: Server der Daten verarbeitet
WPF: Erstellt man Fenster mit, Buttons, Labels,…
LINQ: Daten aus Datenstruktur suchen
Databinding: Automatische Aktualisierung von Daten im WPF zB.:Variable fix auf Textfeld gebinded -> bei änderung von Variable wird textfeld automatisch aktualisiert

1. Neues WCF Projekt
2. Im WCF Projekt ein neues WPF Projekt erstellen
3. XML erstellen -> Base Directory von WCF
4. IService.cs Klasse -> für jede Datenklasse einen DataContract erstellen (siehe Bild)

```
[DataContract]
[XmlRootAttribute("Persons")]
[Serializable]
5 Verweise
public class PersonList{
    [DataMember]
    [XmlElement("Person")]
    1-Verweis
    public Person[] persons { get; set; }
}

[DataContract]
[Serializable]
3 Verweise
public class Person {
    [DataMember]
    [XmlElement("Name")]
    1-Verweis
    public string name { get; set; }

    [DataMember]
    [XmlElement("Surname")]
    1-Verweis
    public string surname { get; set; }

    [DataMember]
    [XmlElement("City")]
    1-Verweis
    public string city { get; set; }
}
```

5. IService1.cs Klasse -> Methoden, die gebraucht werden, mit OperationContract-Tags definieren.

```
public interface IService1 {

    [OperationContract]
    1-Verweis
    PersonList GetPersons();

    [OperationContract]
    1-Verweis
    List<Person> GetPerson(string search);

    // TODO: Hier Dienstvorgänge hinzufügen
}
```

6. Service1.cs
    a. Alles aus Klasse löschen
    b. danach alle Methoden aus Interface erstellen lassen
    c. Xml lesen mit Code (siehe Bild)

```
public Service1() {
    using (TextReader reader = new StreamReader(AppDomain.CurrentDomain.BaseDirectory + "/persons.xml")) {
        XmlSerializer serializer = new XmlSerializer(typeof(PersonList));
        persons = (PersonList)serializer.Deserialize(reader);
    }
}
```

    d. Methoden implementieren (LINQ verwenden)
    e. WCF kann man ausführen -> mit WCFTestClient starten und Methoden testen
7. WPF -> Dienstverweis hinzufügen -> Discover (Ermitteln) dann wird service1 automatisch erkannt.
8. MainWindow.xaml ->
    a. Datagrid und Textbox und weiteres schön anordnen. (Namen an die Elemente vergeben)
    b. MainWindow.xaml.cs -> in Datagrid Eigenschaft Intemsource="{Binding}" und autogenerateColumns=false
    c. Datagrid.Columns hinzufügen mit allen Columns die gebraucht werden

```
<DataGrid x:Name="grid" HorizontalAlignment="Left" Height="271" VerticalAlignment="Top" Width="532" Margin="53,94,0,0" ItemsSource="{Binding}" AutoGenerateColumns="False">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Name" Binding="{Binding Path=name}" />
        <DataGridTextColumn Header="Surname" Binding="{Binding Path=surname}"/>
        <DataGridTextColumn Header="City" Binding="{Binding Path=city}"/>
    </DataGrid.Columns>
</DataGrid>
```

9. Mainwindow.xaml.cs ->
    a. Neue Instanz von SErviceRefernce1.IService1 und Observablecollection erstellen
    b. Unter Initializecomponent beides instanzieren
    c. Datacontext von grid auf Collection verweisen
    d. mit service.Methode kann man auf die Methoden des WCF zugreifen (siehe Bild)

```
public partial class MainWindow : Window {

    ServiceReference1.IService1 service;
    ObservableCollection<ServiceReference1.Person> p;

    0 Verweise
    public MainWindow() {
        InitializeComponent();

        service = new ServiceReference1.Service1Client();
        p = new ObservableCollection<ServiceReference1.Person>(service.GetPersons().persons);
        grid.DataContext = p;
    }

    1-Verweis
    private void search_TextChanged(object sender, TextChangedEventArgs e) {
        p.Clear();
        service.GetPerson(search.Text).ToList().ForEach(p.Add);
    }
}
```

```
public class Service1 : IService1 {
    PersonList persons = null;
    0 Verweise
    public Service1() {
        using (TextReader reader = new StreamReader(AppDomain.CurrentDomain.BaseDirectory + "/persons.xml")) {
            XmlSerializer serializer = new XmlSerializer(typeof(PersonList));
            persons = (PersonList)serializer.Deserialize(reader);
        }
    }

    1-Verweis
    public PersonList GetPersons() {
        return persons;
    }

    1-Verweis
    public List<Person> GetPersonFilter(string search) {
        return (from p in persons.personList
                where p.city.IndexOf(search, StringComparison.OrdinalIgnoreCase) >= 0 ||
                p.name.IndexOf(search, StringComparison.OrdinalIgnoreCase) >= 0 ||
                p.surname.IndexOf(search, StringComparison.OrdinalIgnoreCase) >= 0
                select p).ToList();
    }
}
```

Sami is a huso