CrowdDJ – Ausbaustufe 2

Inhaltsverzeichnis

Ziele der 2. Ausbaustufe

Es wurde gefordert, dass eine WPF-Anwendung gebaut wird, mit der man sich mit der Datenbank verbinden kann und die Daten dementsprechend für eine Oberfläche aufbereitet kann. Da wir nach dem MVVM Pattern arbeiten sollen, war es nötig, die View vom Model getrennt zu halten. Deshalb wurde eine ViewModel eingeführt, welche Daten aus dem Model holt und diese aufbereitet, sodass die Elemente in der View nur noch mit diesen Befüllt werden müssen. Um z.B.: ein DataGrid mit Daten zu füllen, muss in der entsprechenden ViewModel eine Collection vorhanden sein, aus welcher das DataGrid die Daten holt. Als Collection eignen sich hierfür entweder List<T> oder ObservableCollection<T>.

Was sich im Vergleich zur ersten Ausbaustufe verändert hat

Die BusinessLogic wurde ausgebaut und gegen ein Interface programmiert. Man erhält nun nur noch Daten über die Businesslogic, da die MemberDAOs private gesetzt wurden, welche aber anschließend von der Businesslogic aufgerufen werden und schlussendlich Daten aus der Datenbank holen. Aus diesem Grund wird in jedem ViewModel ein Businesslogic Objekt angelegt, über das man sich die Daten für die View holen kann.

Die einzelnen DAOs wurde dahingehen überarbeitet, dass sie nun Datenbankyp unabhängiger sind. Es wurde eine DataBase eingeführt, welche die Datenbankbefehle, speziell für die ausgewählte Datenbank, zusammenbaut und auch feuert.

Die Domainklasse Party wurde überarbeitet. PartyId ist nun ein String, aus der man den QRCode erstellen kann. Aus diesen Grund wurden auch die Tabellen, welche abhängig von Party waren, überarbeitet und die PartyId immer auf String gesetzt. Selbiges gilt für alle Methodenaufrufe etc. in den Interfaces, DAOs etc.

Alles innerhalb der Solution wird nun über Projekte verwaltet und nicht mehr über Ordner.

2. Ausbaustufe

Für diese Ausbaustufe wurden 3 Projekte hinzugefügt: CrowdDJ.Playstation, CrowdDJ.QRCode und CrowdDJ.SimpleVote.

Um dies zu Lösen wurden die in den Übungen aufbereiteten Beispiele als Referenzen hergenommen und für meinen Nutzen entsprechend überarbeitet.

CrowdDJ.Playstation

Herzstück dieser Ausbaustufe ist die Playstation. Sie enthält alle Ansichten für den Admin. Eingelogt wird sich ganz simpel mit der E-Mail Adresse und dem dazugehörigen Passwort. (MainWindow.xaml, MainWindow.xaml.cs, LogInViewModel.cs). Die ViewModels wurden von der ViewModelBase abgeleitet. Diese selbst ist von INotifyPropertyChanged abgeleitet.

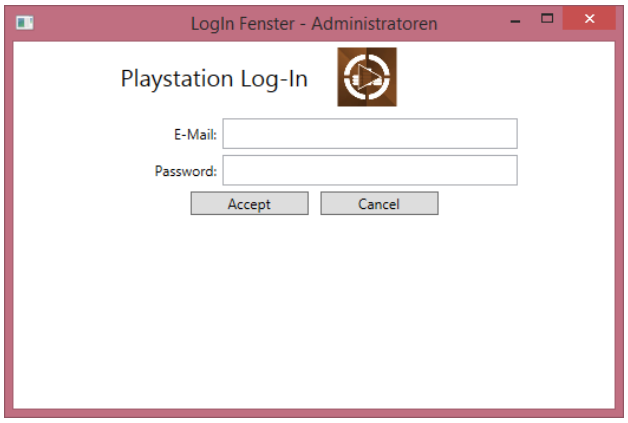
ViewModelBase.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;

namespace CrowdDJ.Playstation.ViewModels
{
    public class ViewModelBase : INotifyPropertyChanged
    {
        public ViewModelBase()
        {
        }
        public void OnPropertyChanged(string name)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(name));
            }
        }
        public event PropertyChangedEventHandler PropertyChanged;
    }
}
```

RelayCommand.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    public class RelayCommand : ICommand
    {
        private readonly Action<object> executeAction;
        private readonly Predicate<object> canExecutePredicate;

        public RelayCommand(Action<object> execute)
            : this(execute, null)
        {
        }

        public RelayCommand(Action<object> execute, Predicate<object> canExecute)
        {
            if (execute == null)
                throw new ArgumentNullException("execute");

            executeAction = execute;
            canExecutePredicate = canExecute;
        }
```

```
        /// <summary>
        /// Defines the method to be called when the command is invoked.
        /// </summary>
        /// <param name="parameter">Data used by the command.  If the command does not
        ///       require data to be passed, this object can be set to null.</param>
        public void Execute(object parameter)
        {
            executeAction(parameter);
        }

        /// <summary>
        /// Defines the method that determines whether the command can execute in its
        ///      current state.
        /// </summary>
        /// <param name="parameter">Data used by the command.  If the command does not
        ///      require data to be passed, this object can be set to null.</param>
        /// <returns>
        /// true if this command can be executed; otherwise, false.
        /// </returns>
        public bool CanExecute(object parameter)
        {
            return canExecutePredicate == null ? true :
                                    canExecutePredicate(parameter);
        }

        public event EventHandler CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }
    }
}
```



MainWindow.xaml:

```
<Window x:Class="CrowdDJ.Playstation.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="LogIn Fenster - Administratoren" Height="350" Width="525">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>

        <Grid Grid.Row="0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*" />
                <ColumnDefinition Width="1*" />
            </Grid.ColumnDefinitions>
            <Label Grid.Column="0"  HorizontalAlignment="Right"
                    VerticalAlignment="Center" FontSize="20">
                Playstation Log-In
            </Label>
            <Image Grid.Column="1" HorizontalAlignment="Left"
                    Source="Pictures/CrowdDJ-Logo.jpg" Height="50"
                    Margin="20,5,5,5"/>
        </Grid>

        <Grid Grid.Row="1" Margin="0,5,0,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.8*" />
                <ColumnDefinition Width="1.5*" />
            </Grid.ColumnDefinitions>
            <Label Grid.Column="0" HorizontalAlignment="Right">E-Mail: </Label>
            <TextBox Grid.Column="1" HorizontalAlignment="Left" Width="250"
                                            Text="{Binding Email}"/>
        </Grid>

        <Grid Grid.Row="2" Margin="0,5,0,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.8*" />
                <ColumnDefinition Width="1.5*" />
            </Grid.ColumnDefinitions>
            <Label Grid.Column="0" HorizontalAlignment="Right">Password: </Label>
            <PasswordBox Grid.Column="1" HorizontalAlignment="Left" Width="250"
                                            Name="txtPassword" />
```

```
        </Grid>

        <Grid Grid.Row="3" Margin="0,5,0,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*" />
                <ColumnDefinition Width="1*" />
            </Grid.ColumnDefinitions>
            <Button Grid.Column="0" Content="Accept" HorizontalAlignment="Right"
                                                Width="100" Margin="0,0,5,0"
                    Command="{Binding LoginCommand}" CommandParameter="{Binding
                                                ElementName=txtPassword}"/>
            <Button Grid.Column="1" Content="Cancel" HorizontalAlignment="Left"
                                                Width="100" Margin="5,0,0,0" />
        </Grid>
    </Grid>
</Window>
```

MainWindow.xaml.cs

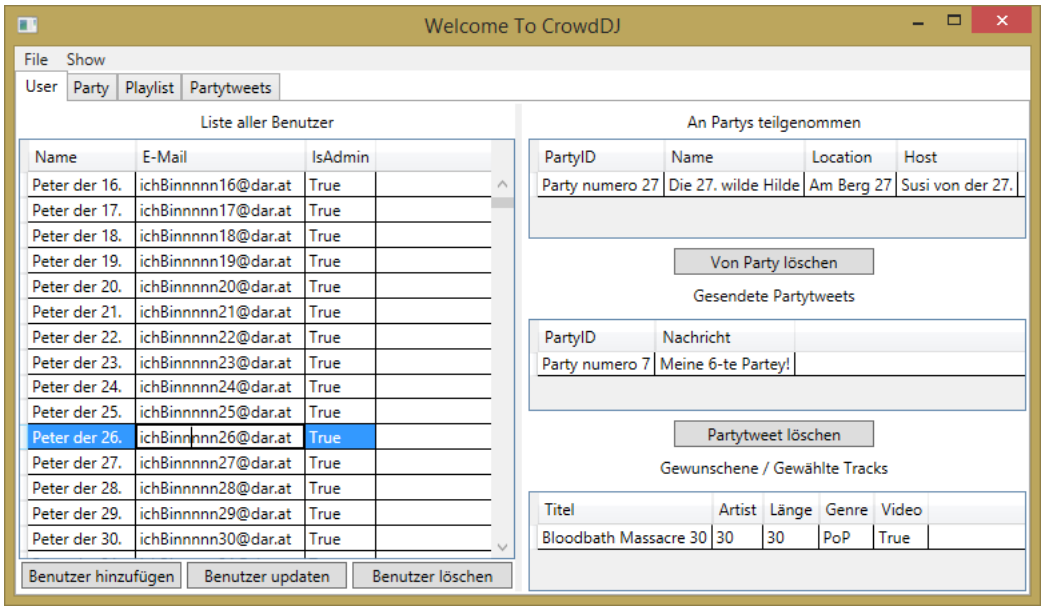
```
using CrowdDJ.Playstation.ViewModels;
using CrowdDJ.Playstation.Views;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CrowdDJ.Playstation
{
    /// <summary>
    /// Interaktionslogik für MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            this.DataContext = new LogInViewModel();
        }
    }
}
```

LogInViewModel.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using CrowdDJ.Playstation.Views;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
```

```
using System.Windows.Controls;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    class LogInViewModel
    {
        private string email;
        public string Email
        {
            get { return email; }
            set { email = value; }
        }

        private string password;



       public string Password
        {
            get { return password; }
            set { password = value; }
        }


        public ICommand LoginCommand { get; set; }
        public ICommand CancelLoginCommand { get; set; }

        private ICrowdDJBL bl = new CrowdDJBL();

        public LogInViewModel()
        {
            this.LoginCommand = new RelayCommand(this.Login);
            this.CancelLoginCommand = new RelayCommand(this.CancelLogin);
        }

        private void CancelLogin(object obj)
        {
            Application.Current.Windows[0].Close();
        }

        private void Login(object obj)
        {
            Password = ((PasswordBox)obj).Password;
            User dummyUser = bl.FindUserByEmail(Email);
            if (dummyUser.Password.Equals(Password))
            {
                MessageBoxResult result = MessageBox.Show("Anmeldung erfolgreich",
                                                          "Gratuliere",
                                                MessageBoxButton.OK,
                                                MessageBoxImage.Exclamation);
                Window window = new MainWindowLayout();
                window.Show();
                Application.Current.Windows[0].Close();
            }
        }
    }
}
```

MainWindowLayout



Sobald man sich richtig eingeloggt hat, öffnet sich das Hauptfenster, in dem 4 Tabs sind: Ein Benutzer-, ein Party-, ein Playlist- und ein Partytweetstab.

UserTab

Der Benutzertab dienst zur Administration der Benutzer! Die View enthält 4 Datagrids zum Anzeigen wichtiger Daten, die den Benutzer betreffen. Ein DataGrid zum Anzeigen aller Benutzer, einen zum Anzeigen an den Teilgenommenen Partys des angeklickten Benutzers, dessen Partytweets und gewunschenen Tracks!

MainWindowLayout.xaml

```
<Window x:Class="CrowdDJ.Playstation.Views.MainWindowLayout"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:userView="clr-namespace:CrowdDJ.Playstation.UserControls"
        Title="Welcome To CrowdDJ" Height="450" Width="600">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Menu IsMainMenu="True" Grid.Row="0">
            <MenuItem Header="File">
                <MenuItem Header="Close" Command="Close"/>
            </MenuItem>
            <MenuItem Header="Show">
                <MenuItem Header="User"/>
                <MenuItem Header="Party"/>
                <MenuItem Header="Playlist"/>
            </MenuItem>
        </Menu>
```

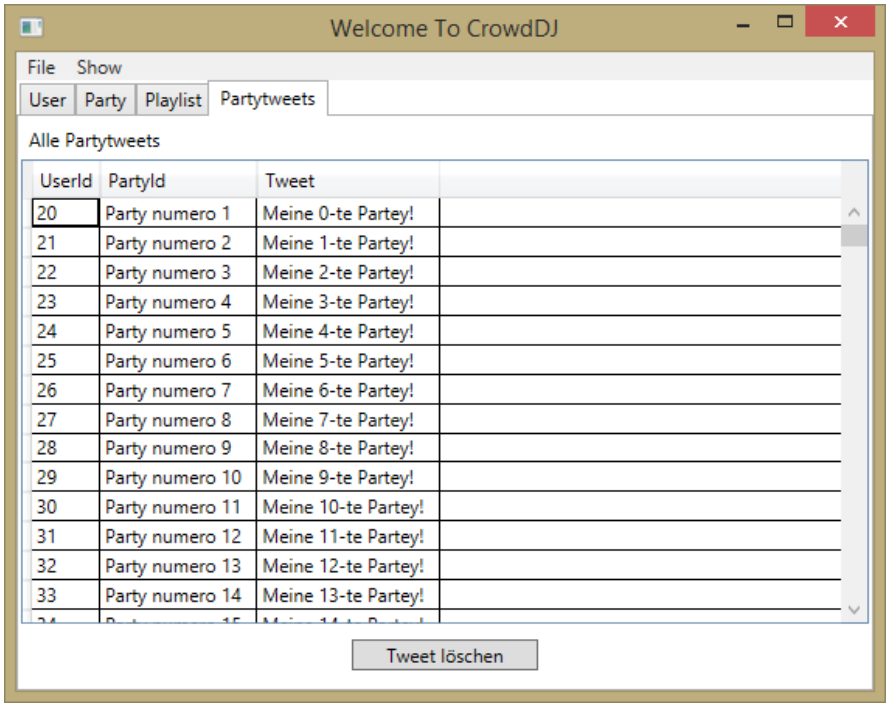
```
<TabControl Grid.Row="1">
            <TabItem Header="User">
                <userView:UserWindowControl />
            </TabItem>
            <TabItem Header="Party">
                <userView:PartyWindowControl />
            </TabItem>
            <TabItem Header="Playlist">
                <userView:PlaylistWindowControl />
            </TabItem>
            <TabItem Header="Partytweets">
                <userView:PartytweetWindowControl />
            </TabItem>
        </TabControl>
    </Grid>
</Window>
```

MainWindowLayout.xaml.cs

```
using CrowdDJ.Playstation.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace CrowdDJ.Playstation.Views
{
    /// <summary>
    /// Interaktionslogik für MainWindowLayout.xaml
    /// </summary>
    public partial class MainWindowLayout : Window
    {
        public MainWindowLayout()
        {
            InitializeComponent();
        }
    }
}
```

UserControls und VMs

Da im MainWindowLayout kein DataContext gesetzt wird, wird dieses über die UserControls gemacht, welche innerhalb der einzelnen Tabs sind.



PartytweetWindowControl.xaml

```
<UserControl x:Class="CrowdDJ.Playstation.UserControls.PartytweetWindowControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:i="clr-
      namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>
        <Label Grid.Row="0" Content="Alle Partytweets" />
        <DataGrid Name="PartytweetDataGrid"
                          Grid.Row="1" AutoGenerateColumns="False"
                          ItemsSource="{Binding Path=AllPartytweets}"
                          SelectedItem="{Binding Path=IsSelectedPartyTweet}"
                          EnableRowVirtualization="False">
            <i:Interaction.Triggers>
                <i:EventTrigger EventName="MouseDoubleClick">
                    <i:InvokeCommandAction Command="{Binding DoubleClickCommand}"
                                           CommandParameter="{Binding
                                              ElementName=PartytweetDataGrid,
                                              Path=SelectedItem}"/>
                </i:EventTrigger>
            </i:Interaction.Triggers>
```

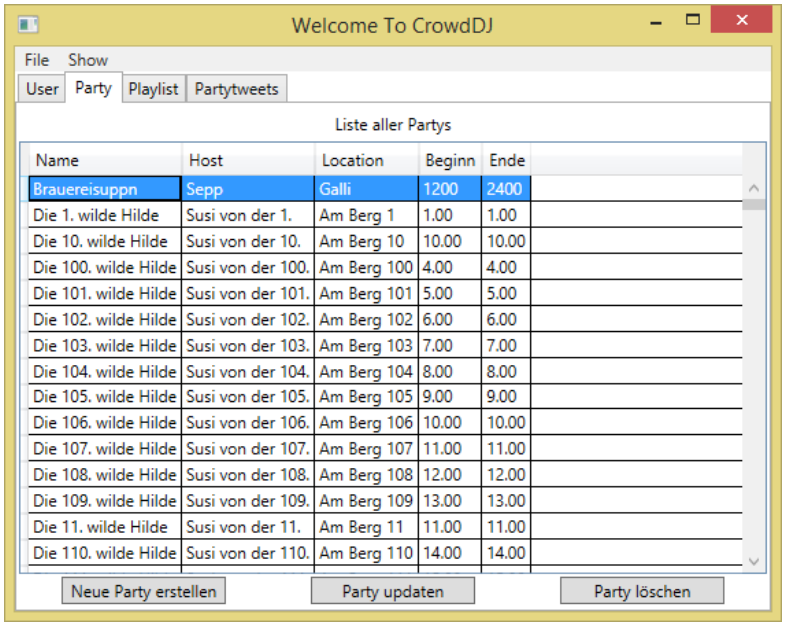
```
            <DataGrid.Columns>
                <DataGridTextColumn Header="UserId" Binding="{Binding Path=UserId}"
                                                                    Width="auto" />
                <DataGridTextColumn Header="PartyId" Binding="{Binding Path=PartyId}"
                                                                    Width="auto" />
                <DataGridTextColumn Header="Tweet" Binding="{Binding Path=Message}"
                                                                    Width="auto" />
            </DataGrid.Columns>
        </DataGrid>
        <Grid Grid.Row="2" Margin="0,10,0,10">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="120" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Button Grid.Column="1" Name="bttnDeleteTweet" Content="Tweet löschen"
                     Command="{Binding DeletePartytweetCommand}"
                    CommandParameter="{Binding ElementName=PartytweetDataGrid,
                                       Path=SelectedItem}"/>
        </Grid>
    </Grid>
</UserControl>
```

PartytweetWindowControl.xaml.ca

```
using CrowdDJ.Playstation.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CrowdDJ.Playstation.UserControls
{
    /// <summary>
    /// Interaktionslogik für PartytweetWindowControl.xaml
    /// </summary>
    public partial class PartytweetWindowControl : UserControl
    {
        public PartytweetWindowControl()
        {
            InitializeComponent();
            this.DataContext = new PartytweetVM();
        }
    }
}
```

PartytweetVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    class PartytweetVM : ViewModelBase
    {
        private ObservableCollection<Partytweet> allPartytweets;
        public ObservableCollection<Partytweet> AllPartytweets
        {
            get { return allPartytweets; }
            set { allPartytweets = value; }
        }
        public Partytweet IsSelectedPartytweet { get; set; }

        public ICommand DoubleClickCommand { get; set; }
        public ICommand DeletePartytweetCommand { get; set; }

        private ICrowdDJBL bl = new CrowdDJBL();

        public PartytweetVM()
        {
            this.DoubleClickCommand = new RelayCommand(this.SetSelectedPartytweet);
            this.DeletePartytweetCommand = new RelayCommand(this.DeletePartytweet);
            AllPartytweets = bl.GetAllTweets();
        }

        private void DeletePartytweet(object obj)
        {
            if ((IsSelectedPartytweet != null) && (IsSelectedPartytweet == obj as
                                        Partytweet))
            {
                bl.DeletePartytweet(IsSelectedPartytweet);
                AllPartytweets.Remove(IsSelectedPartytweet);
            }
            else
            {
                MessageBoxResult result = MessageBox.Show("Keine Zeile ausgewählt!
                                Bitte einen Doppelklick auf eine Zeile!",
                                "Fehler...", MessageBoxButton.OK,
                                MessageBoxImage.Error);
            }
        }

        private void SetSelectedPartytweet(object obj)
        {
            IsSelectedPartytweet = obj as Partytweet;
        }
    }
}
```



PartyWindowControl.xaml

```
<UserControl x:Class="CrowdDJ.Playstation.UserControls.PartyWindowControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:i="clr-
namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>
        <Label Grid.Row="0" Content="Liste aller Partys" HorizontalAlignment="Center"
                                                                              />
        <DataGrid Name="PartyDataGrid"
                          Grid.Row="1" AutoGenerateColumns="False"
                          ItemsSource="{Binding Path=AllParties}"
                          SelectedItem="{Binding Path=IsSelectedParty}"
                          EnableRowVirtualization="False">
            <i:Interaction.Triggers>
                <i:EventTrigger EventName="MouseDoubleClick">
                    <i:InvokeCommandAction Command="{Binding DoubleClickCommand}"
                                           CommandParameter="{Binding
                                              ElementName=PartyDataGrid,
                                              Path=SelectedItem}"/>
                </i:EventTrigger>
            </i:Interaction.Triggers>
            <DataGrid.Columns>
                <DataGridTextColumn Header="Name" Binding="{Binding Path=Name}"
                            Width="auto" />
                <DataGridTextColumn Header="Host" Binding="{Binding Path=Host}"
                            Width="auto" />
                <DataGridTextColumn Header="Location" Binding="{Binding
                            Path=Location}" Width="auto" />
                <DataGridTextColumn Header="Beginn" Binding="{Binding
                            Path=PartyBegin}" Width="auto" />
                <DataGridTextColumn Header="Ende" Binding="{Binding Path=PartyEnd}"
```

```
                            Width="auto" />
                </DataGrid.Columns>
            </DataGrid>
            <Grid Grid.Row="2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Button Grid.Column="0" Name="btnAddParty" Content="Neue Party erstellen"
                        HorizontalAlignment="Center" Width="120" Height="auto" Margin="2"
                        Command="{Binding Path=AddNewPartyCommand}"/>
                <Button Grid.Column="1" Name="btnUpdateParty" Content="Party updaten"
                        HorizontalAlignment="Center" Width="120" Height="auto" Margin="2"
                        Command="{Binding Path=UpdatePartyCommand}"
                        CommandParameter="{Binding ElementName=PartyDataGrid,
                                                   Path=SelectedItem}"/>
                <Button Grid.Column="2" Name="btnDeleteParty" Content="Party löschen"
                        HorizontalAlignment="Center" Width="120" Height="auto" Margin="2"
                        Command="{Binding Path=DeletePartyCommand}"
                        CommandParameter="{Binding ElementName=PartyDataGrid,
                                                   Path=SelectedItem}"/>
            </Grid>
        </Grid>
</UserControl>
```

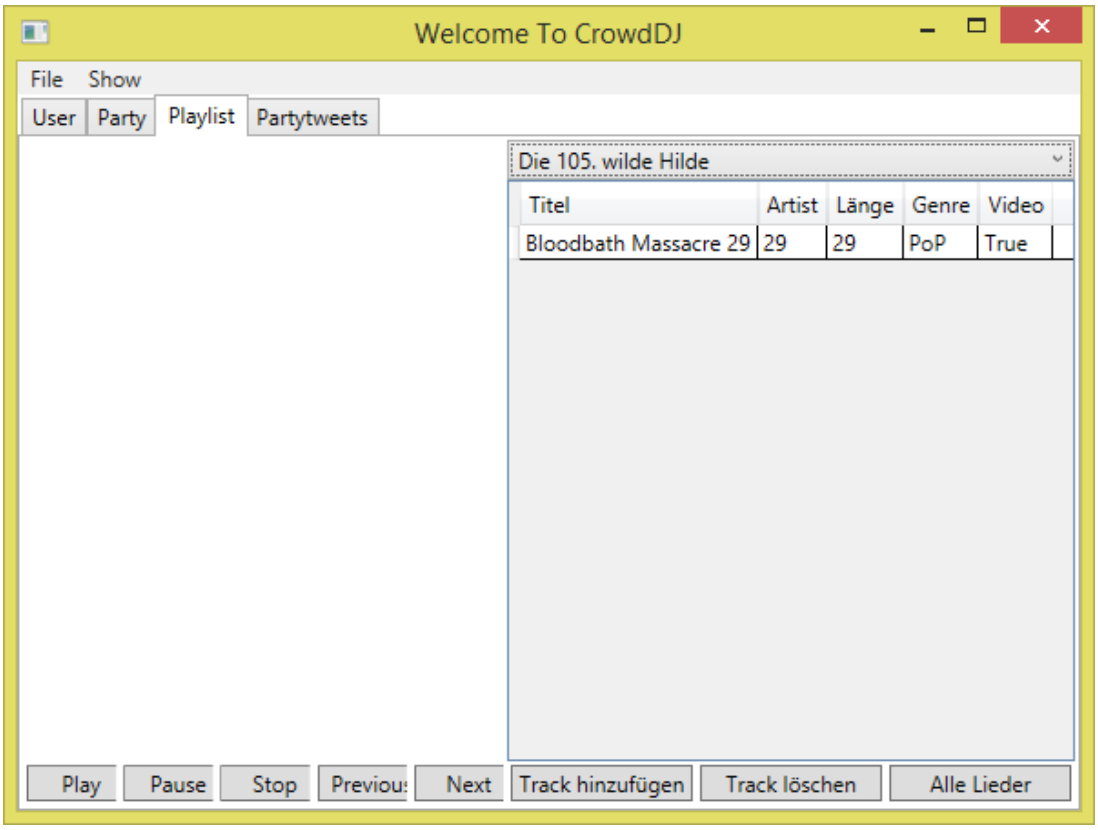
PartyWindowControl.xaml.cs

```
using CrowdDJ.Playstation.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CrowdDJ.Playstation.UserControls
{
    /// <summary>
    /// Interaktionslogik für PartyWindowControl.xaml
    /// </summary>
    public partial class PartyWindowControl : UserControl
    {
        public PartyWindowControl()
        {
            InitializeComponent();
            this.DataContext = new PartyVM();
        }
    }
}
```

PartyVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using CrowdDJ.Playstation.Views;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    class PartyVM : ViewModelBase
    {
        private ObservableCollection<Party> allParties;
        public ObservableCollection<Party> AllParties
        {
            get { return allParties; }
            set { allParties = value; OnPropertyChanged("AllParties"); }
        }
        public Party IsSelectedParty { get; set; }
        public ICommand DeletePartyCommand { get; private set; }
        public ICommand UpdatePartyCommand { get; private set; }
        public ICommand AddNewPartyCommand { get; private set; }
        public ICommand DoubleClickCommand { get; private set; }

        private ICrowdDJBL bl = new CrowdDJBL();

        public PartyVM()
        {
            this.DeletePartyCommand = new RelayCommand(this.DeleteParty);
            this.UpdatePartyCommand = new RelayCommand(this.UpdateParty);
            this.AddNewPartyCommand = new RelayCommand(this.AddParty);
            this.DoubleClickCommand = new RelayCommand(this.SetSelectedParty);
            AllParties = bl.GetAllParties();
            IsSelectedParty = AllParties.First();
        }

        private void SetSelectedParty(object obj)
        {
            IsSelectedParty = obj as Party;
        }

        private void AddParty(object obj)
        {
            Window window = new PartyAddNewPartyWindow();
            window.Show();
        }

        private void UpdateParty(object obj)
        {
            SetSelectedParty(obj);
            bl.UpdateParty(IsSelectedParty, IsSelectedParty.PartyId);
        }
```

```
        private void DeleteParty(object obj)
        {
            if ((IsSelectedParty == null) && (obj as Party != IsSelectedParty))
            {
                bl.RemovePartyWithId(IsSelectedParty.PartyId);
                AllParties.Remove(IsSelectedParty);
            }
            else
            {
                MessageBoxResult result = MessageBox.Show("Keine Zeile ausgewählt!
                            Bitte einen Doppelklick auf eine Zeile!", "Fehler...",
                                                          MessageBoxButton.OK,
                                                           MessageBoxImage.Error);
            }
        }

    }
}
```



PlaylistWindowControl.xaml

```
<UserControl x:Class="CrowdDJ.Playstation.UserControls.PlaylistWindowControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="600">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="auto" />
        </Grid.ColumnDefinitions>
```

```
<Grid Grid.Column="0" >
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="auto" />
    </Grid.RowDefinitions>
    <MediaElement Grid.Row="0" Name="mediaPlay"
                  Source="TT.mp4" LoadedBehavior="Manual" />
    <Grid Grid.Row="1">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Button Grid.Column="0" Name="btnPlay" Content="Play"
                                          Width="60"
                Margin="2" Click="bttnPlay_Click"/>
        <Button Grid.Column="1" Name="btnPause" Content="Pause"
                                          Width="60"
                Margin="2" Command="{Binding PauseMediaCommand}"/>
        <Button Grid.Column="2" Name="btnStop" Content="Stop"
                                           Width="60"
                Margin="2" Command="{Binding StopMediaCommand}"/>
        <Button Grid.Column="3" Name="btnPrevious" Content="Previous"
                                           Width="60"
                Margin="2" Command="{Binding PreviousMediaCommand}"/>
        <Button Grid.Column="4" Name="btnNext" Content="Next"
                                           Width="60"
                Margin="2" Command="{Binding NextMediaCommand}"/>
    </Grid>
</Grid>
<Grid Grid.Column="1">
    <Grid.RowDefinitions>
        <RowDefinition Height="auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="auto" />
    </Grid.RowDefinitions>
    <ComboBox Grid.Row="0" ItemsSource="{Binding Path=AllParties}"
              SelectedItem="{Binding Path=IsSelectedParty}">
        <ComboBox.ItemTemplate>
            <DataTemplate>
                <TextBlock Text="{Binding Path=Name}" />
            </DataTemplate>
        </ComboBox.ItemTemplate>
    </ComboBox>
    <DataGrid Grid.Row="1" AutoGenerateColumns="false"
              ItemsSource="{Binding Path=Tracks, IsAsync=True}"
              SelectedItem="{Binding Path=IsSelectedParty}"
              EnableRowVirtualization="False">
        <DataGrid.Columns>
            <DataGridTextColumn Header="Titel" Binding="{Binding Path=Title}"
                                Width="auto" />
            <DataGridTextColumn Header="Artist" Binding="{Binding
                                Path=Artist}" Width="auto" />
            <DataGridTextColumn Header="Länge" Binding="{Binding Path=Length}"
                                Width="auto" />
            <DataGridTextColumn Header="Genre" Binding="{Binding Path=Genre}"
                                Width="auto" />
            <DataGridTextColumn Header="Video" Binding="{Binding
                                Path=IsVideo}" Width="auto" />
        </DataGrid.Columns>
    </DataGrid>
```

```
            <Grid Grid.Row="2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="auto" />
                    <ColumnDefinition Width="auto" />
                    <ColumnDefinition Width="auto" />
                </Grid.ColumnDefinitions>
                <Button Grid.Column="0" Name="bttnAddTrack" Content="Track hinzufügen"
                                        Width="100"
                        Margin="2" Command="{Binding Path=AddNewTrackCommand}"/>
                <Button Grid.Column="1" Name="bttnDeleteTrack" Content="Track löschen"
                                        Width="100"
                        Margin="2"/>
                <Button Grid.Column="2" Name="btnShowAllTracks" Content="Alle Lieder"
                                        Width="100"
                        Margin="2" Command="{Binding Path=ShowAllTracksCommand}"/>
            </Grid>
        </Grid>
    </Grid>
</UserControl>
```

PlaylistWindowControl.xaml.cs

```
using CrowdDJ.Playstation.ViewModels;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CrowdDJ.Playstation.UserControls
{
    /// <summary>
    /// Interaktionslogik für PlaylistWindowControl.xaml
    /// </summary>
    public partial class PlaylistWindowControl : UserControl
    {
        ObservableCollection<Uri> youtubeLinks = new ObservableCollection<Uri>();
        public Uri SelectedItem { get; set; }
        int i = 0;
        public PlaylistWindowControl()
        {
            youtubeLinks.Add(new Uri(@"https://www.youtube.com/watch?v=uKDuIhNVLyg"));
            youtubeLinks.Add(new Uri(@"https://www.youtube.com/watch?v=TiVIFMbwxOc"));
            youtubeLinks.Add(new Uri(@"https://www.youtube.com/watch?v=Sj9A-5VmDTE"));
            SelectedItem = youtubeLinks.First();
            InitializeComponent();
            this.DataContext = new PlaylistVM();
        }
```

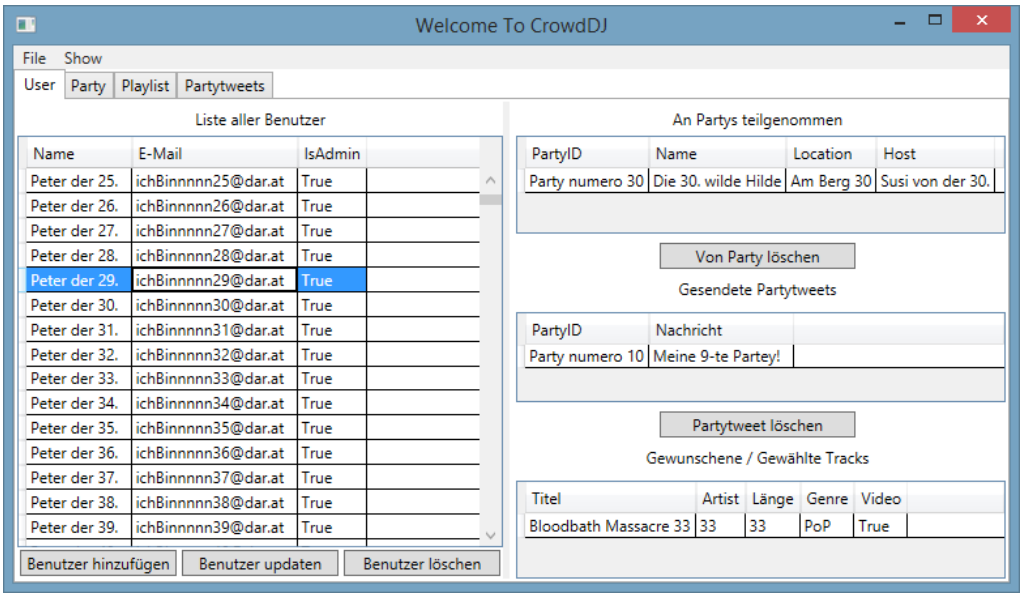
```
        private void bttnRewind_Click(object sender, RoutedEventArgs e)
        {
            if (SelectedItem == youtubeLinks.First())
            {
                mediaPlay.Play();
            }
            else
            {
                i--;
                SelectedItem = youtubeLinks[i];
                mediaPlay.Play();
            }


        }

        private void bttnNext_Click(object sender, RoutedEventArgs e)
        {
            if (SelectedItem == youtubeLinks.Last())
            {
                mediaPlay.Play();
            }
            else
            {
                i++;
                SelectedItem = youtubeLinks[i];
                mediaPlay.Play();
            }
        }

        private void bttnPlay_Click(object sender, RoutedEventArgs e)
        {
            mediaPlay.Play();
        }

        private void bttnStop_Click(object sender, RoutedEventArgs e)
        {
            mediaPlay.Stop();
        }

        private void bttnPause_Click(object sender, RoutedEventArgs e)
        {
            mediaPlay.Pause();
        }
    }
}
```

Anmerkung: Es gelang mir bis zum Schluss nicht, dass der MediaPlayer von WPF auch nur irgendein Video abspielt. Egal, ob hardcoded, aus einer Collection oder aus der VM es geschah nichts. Das wird bis zur 3. Ausbaustufe überarbeitet!

PlaylistVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using CrowdDJ.Playstation.Views;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    class PlaylistVM : ViewModelBase
    {
        private ObservableCollection<Party> allparties;
        public ObservableCollection<Party> AllParties
        {
            get { return allparties; }
            set { allparties = value; OnPropertyChanged("AllParties"); }
        }

        private ObservableCollection<Track> tracks;
        public ObservableCollection<Track> Tracks
        {
            get { return tracks; }
            set { tracks = value; OnPropertyChanged("Tracks"); }
        }

        private Party isSelectedParty;
        public Party IsSelectedParty
        {
            get { return isSelectedParty; }
            set
            {
                isSelectedParty = value;
                OnPropertyChanged("IsSelectedParty");
                UpdateTracks();
            }
        }
        private Party previousParty = null;
        private ICrowdDJBL bl = new CrowdDJBL();

        public ICommand AddNewTrackCommand { get; private set; }
        public ICommand ShowAllTracksCommand { get; private set; }

        public PlaylistVM()
        {
            this.AddNewTrackCommand = new RelayCommand(this.AddNewTrack);
            this.ShowAllTracksCommand = new RelayCommand(this.ShowAllTrack);
            AllParties = bl.GetAllParties();
            previousParty = AllParties[8];
            IsSelectedParty = previousParty;
        }

        private void ShowAllTrack(object obj)
        {
            Tracks = bl.GetAllTracks();
        }

        private void AddNewTrack(object obj)
```

```
        {
            Window window = new PlaylistAddTrackWindow();
            window.Show();
        }

        private void UpdateTracks()
        {
            Playlist playlist = bl.GetPlaylistForParty(IsSelectedParty.PartyId);
            if (playlist == null)
            {
                MessageBoxResult result = MessageBox.Show("Keine Playlist für " +
                                  IsSelectedParty.Name, "Fehler",
                                  MessageBoxButton.OK, MessageBoxImage.Error);
                IsSelectedParty = previousParty;
            }
            else
            {
                Tracks = bl.GetAllTracksInPlaylist(playlist.PlaylistId);
                previousParty = IsSelectedParty;
            }
        }
    }
}
```



UserWindowControl.xaml

```
<UserControl x:Class="CrowdDJ.Playstation.UserControls.UserWindowControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
            xmlns:i="clr-
namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="600">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="0" Margin="0,0,5,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="auto" />
                <RowDefinition Height="1*" />
```

```
        </Grid.RowDefinitions>
        <Label Grid.Row="0" Content="Liste aller Benutzer"
                              HorizontalAlignment="Center" />
        <Grid Grid.Row="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="*" />
                <RowDefinition Height="auto" />
            </Grid.RowDefinitions>
            <DataGrid Name="UserDataGrid"
                      Grid.Row="0" AutoGenerateColumns="False"
                      ItemsSource="{Binding Path=AllUser}"
                      SelectedItem="{Binding Path=IsSelectedUser}"
                      EnableRowVirtualization="False">
                <i:Interaction.Triggers>
                    <i:EventTrigger EventName="MouseDoubleClick">
                        <i:InvokeCommandAction Command="{Binding
                              DoubleClickCommand}"
                              CommandParameter="{Binding ElementName=UserDataGrid,
                              Path=SelectedItem}"/>
                    </i:EventTrigger>
                </i:Interaction.Triggers>
                <DataGrid.Columns>
                    <DataGridTextColumn Header="Name" Binding="{Binding
                                        Path=Name}" Width="auto" />
                    <DataGridTextColumn Header="E-Mail" Binding="{Binding
                                        Path=Email}" Width="auto" />
                    <DataGridTextColumn Header="IsAdmin" Binding="{Binding
                                        Path=IsAdmin}" Width="auto" />
                </DataGrid.Columns>
            </DataGrid>
            <Grid Grid.Row="1">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Button Grid.Column="0" Name="bttnAddUser" Content="Benutzer
                                                                    hinzufügen"
                        Width="120" Margin="2" Command="{Binding
                                                         AddUserWindowCommand}"/>
                <Button Grid.Column="1" Name="btnUpdateUser" Content="Benutzer
                                                                    updaten"
                        Width="120" Margin="2" Command="{Binding
                                                         UpdateUserCommand}"
                        CommandParameter="{Binding ElementName=UserDataGrid,
                                                         Path=SelectedItem}"/>
                <Button Grid.Column="2" Name="bttnDeleteUser" Content="Benutzer
                                                                    löschen"
                        Width="120" Margin="2" Command="{Binding
                                                         DeleteUserCommand}"/>
            </Grid>
        </Grid>
    </Grid>
    <GridSplitter Grid.Column="0" Width="5" Margin="5, 0, 0, 0"
                                  VerticalAlignment="Stretch" />
```

```
    <Grid Grid.Column="1">
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>
        <Label Grid.Row="0" HorizontalAlignment="Center" Content="An Partys
                                    teilgenommen"/>
        <DataGrid Grid.Row="1" AutoGenerateColumns="False"
                  ItemsSource="{Binding Path=AttemptedParties, Mode=OneWay,
                                    IsAsync=True}" Margin="5,0,0,5" >
            <DataGrid.Columns>
                <DataGridTextColumn Header="PartyID" Binding="{Binding
                                    Path=PartyId}" Width="auto" />
                <DataGridTextColumn Header="Name" Binding="{Binding Path=Name}"
                                     Width="auto" />
                <DataGridTextColumn Header="Location" Binding="{Binding
                                    Path=Location}" Width="auto" />
                <DataGridTextColumn Header="Host" Binding="{Binding Path=Host}"
                                    Width="auto" />
            </DataGrid.Columns>
        </DataGrid>
        <Button Grid.Row="2" Name="bttnDeleteFromParty" Content="Von Party
                 löschen"
                MaxWidth="150" Margin="2"/>
        <Label Grid.Row="3" HorizontalAlignment="Center" Content="Gesendete
                 Partytweets"/>
        <DataGrid Grid.Row="4" ItemsSource="{Binding Path=SentPartyTweets,
                 Mode=OneWay, IsAsync=True}" AutoGenerateColumns="False"
                 Margin="5,5,0,5">
            <DataGrid.Columns>
                <DataGridTextColumn Header="PartyID" Binding="{Binding
                       Path=PartyId}" Width="auto" />
                <DataGridTextColumn Header="Nachricht" Binding="{Binding
                       Path=Message}" Width="auto" />
            </DataGrid.Columns>
        </DataGrid>
        <Button Grid.Row="5" Name="bttnDeleteTweet" Content="Partytweet löschen"
                 MaxWidth="150" Margin="2"/>
        <Label Grid.Row="6" HorizontalAlignment="Center" Content="Gewunschene /
                                                        Gewählte Tracks" />
        <DataGrid Grid.Row="7" ItemsSource="{Binding Path=VotedTracks,
          Mode=TwoWay, IsAsync=True}" AutoGenerateColumns="False" Margin="5,5,0,0">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Titel" Binding="{Binding Path=Title}"
                       Width="auto" />
                <DataGridTextColumn Header="Artist" Binding="{Binding
                       Path=Artist}" Width="auto" />
                <DataGridTextColumn Header="Länge" Binding="{Binding Path=Length}"
                       Width="auto" />
                <DataGridTextColumn Header="Genre" Binding="{Binding Path=Genre}"
                       Width="auto" />
                <DataGridTextColumn Header="Video" Binding="{Binding
                       Path=IsVideo}" Width="auto" />
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Grid> </UserControl>
```

UserWindowControl.xaml.cs

```
using CrowdDJ.Playstation.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace CrowdDJ.Playstation.UserControls
{
    /// <summary>
    /// Interaktionslogik für UserWindowControl.xaml
    /// </summary>
    public partial class UserWindowControl : UserControl
    {
        public UserWindowControl()
        {
            InitializeComponent();
            this.DataContext = new UserVM();
        }
    }
}
```

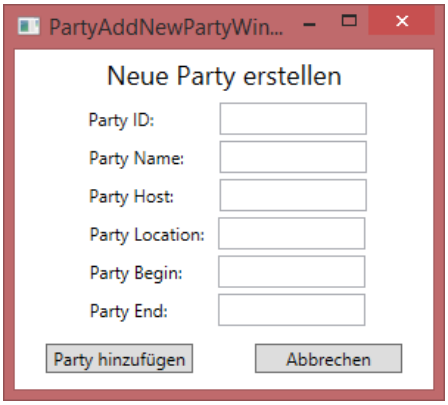
UserVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using CrowdDJ.Playstation.Views;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    public class UserVM : ViewModelBase
    {
        private ObservableCollection<User> allUser { get; set; }
        public ObservableCollection<User> AllUser
        {
            get { return allUser; }
            set { allUser = value; OnPropertyChanged("AllUser"); }
        }
```

```
private ObservableCollection<Party> attemptedParties { get; set; }
public ObservableCollection<Party> AttemptedParties
{
    get { return attemptedParties; }
    set { attemptedParties = value; OnPropertyChanged("AttemptedParties"); }
}

private ObservableCollection<Partytweet> sentPartyTweets { get; set; }
public ObservableCollection<Partytweet> SentPartyTweets
{
    get { return sentPartyTweets; }
    set { sentPartyTweets = value; OnPropertyChanged("SentPartyTweets"); }
}

private ObservableCollection<Track> votedTracks { get; set; }
public ObservableCollection<Track> VotedTracks
{
    get { return votedTracks; }
    set { votedTracks = value; OnPropertyChanged("VotedTracks"); }
}
public User IsSelectedUser { get; set; }
public ICommand DeleteUserCommand { get; private set; }
public ICommand AddUserWindowCommand { get; private set; }
public ICommand UpdateUserCommand { get; private set; }
public ICommand DoubleClickCommand { get; set; }

private ICrowdDJBL bl = new CrowdDJBL();

public UserVM()
{
    this.DeleteUserCommand = new RelayCommand(this.DeleteUser);
    this.DoubleClickCommand = new RelayCommand(this.SetIsSelectedUser);
    this.UpdateUserCommand = new RelayCommand(this.UpdateUser);
    this.AddUserWindowCommand = new RelayCommand(this.AddNewUserWindow);
    AllUser = bl.GetAllUser();
    IsSelectedUser = AllUser.First();
    UpdateDataGrids();
}

private void UpdateDataGrids()
{
    AttemptedParties = bl.GetAttemptedPartyList(IsSelectedUser.UserId);
    SentPartyTweets = bl.GetTweetsForUser(IsSelectedUser.UserId);
    VotedTracks = bl.GetTracksRecommendedByUser(IsSelectedUser.UserId);
}

private void SetIsSelectedUser(object obj)
{
    IsSelectedUser = obj as User;
    UpdateDataGrids();
}

private void AddNewUserWindow(object obj)
{
    Window window = new UserAddNewUserWindow();
    window.Show();
}
```

```
        private void DeleteUser(object obj)
        {
            if ((IsSelectedUser != null) && (obj as User == IsSelectedUser))
            {
                bl.DeleteUser(IsSelectedUser.UserId);
                AllUser.Remove(IsSelectedUser);
            }
            else
            {
                MessageBoxResult result = MessageBox.Show("Keine Zeile ausgewählt!
                          Bitte einen Doppelklick auf eine Zeile!", "Fehler...",
                                                          MessageBoxButton.OK,
                                                          MessageBoxImage.Error);
            }
        }

        private void UpdateUser(object obj)
        {
            SetIsSelectedUser(obj);
            bl.UpdateUser(IsSelectedUser, IsSelectedUser.UserId);
        }
    }
}
```

Views



PartyAddNewPartyWindow.xaml

```
<Window x:Class="CrowdDJ.Playstation.Views.PartyAddNewPartyWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="PartyAddNewPartyWindow" Height="270" Width="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Label Grid.Row="0" Content="Neue Party erstellen" FontSize="18"
                                    HorizontalAlignment="Center"/>
        <Grid Grid.Row="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
```

```
</Grid.RowDefinitions>
<Grid Grid.Row="0" HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="auto" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Party ID:" />
    <TextBox Grid.Column="1" Width="100" Margin="40,2,0,2"
             Text="{Binding Path=PartyId}"/>
</Grid>
<Grid Grid.Row="1" HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="auto" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Party Name:" />
    <TextBox Grid.Column="1" Width="100" Margin="19,2,0,2"
             Text="{Binding Path=Name}"/>
</Grid>
<Grid Grid.Row="2" HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="auto" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Party Host:" />
    <TextBox Grid.Column="1" Width="100" Margin="26,2,0,2"
             Text="{Binding Path=Host}"/>
</Grid>
<Grid Grid.Row="3" HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="auto" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Party Location:" />
    <TextBox Grid.Column="1" Width="100" Margin="5,2,0,2"
             Text="{Binding Path=Location}"/>
</Grid>
<Grid Grid.Row="4" HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="auto" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Party Begin:" />
    <TextBox Grid.Column="1" Width="100" Margin="20,2,0,2"
             Text="{Binding Path=PartyBegin}"/>
</Grid>
<Grid Grid.Row="5" HorizontalAlignment="Center">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="auto" />
        <ColumnDefinition Width="auto" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Party End:" />
    <TextBox Grid.Column="1" Width="100" Margin="30,2,0,2"
             Text="{Binding Path=PartyEnd}"/>
</Grid>
<Grid Grid.Row="6" Margin="0,10,0,0">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Button Name="bttnAddParty" Content="Party hinzufügen" Grid.Column="0"
            HorizontalAlignment="Center" Width="100"
            Command="{Binding AddPartyCommand}"/>
```

```
                <Button Name="bttnCancel" Content="Abbrechen" Grid.Column="1"
                        HorizontalAlignment="Center" Width="100"
                        Command="{Binding CancelAddPartyCommand}"/>
            </Grid>
        </Grid>
    </Grid>
</Window>
```

PartyAddNewPartyWindow.xaml.cs

```
namespace CrowdDJ.Playstation.Views
{
    /// <summary>
    /// Interaktionslogik für PartyAddNewPartyWindow.xaml
    /// </summary>
    public partial class PartyAddNewPartyWindow : Window
    {
        public PartyAddNewPartyWindow()
        {
            InitializeComponent();
            this.DataContext = new PartyAddNewPartyVM();
        }
    }
}
```

PartyAddNewPartyVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    public class PartyAddNewPartyVM
    {
        public string PartyId { get; set; }
        public string Name { get; set; }
        public string Host { get; set; }
        public string Location { get; set; }
        public string PartyBegin { get; set; }
        public string PartyEnd { get; set; }
        public bool IsActive { get; set; }
        public ICommand AddPartyCommand { get; private set; }
        public ICommand CancelAddPartyCommand { get; private set; }

        private ICrowdDJBL bl = new CrowdDJBL();

        public PartyAddNewPartyVM()
        {
            this.AddPartyCommand = new RelayCommand(this.AddParty);
            this.CancelAddPartyCommand = new RelayCommand(this.CancelAddParty);
        }
```

```
        private void CancelAddParty(object obj)
        {
            Application.Current.Windows[1].Close();
        }

        private void AddParty(object obj)
        {
            bl.AddParty(new Party(PartyId, Name, Location, Host, PartyBegin, PartyEnd,
                                  IsActive));
            Application.Current.Windows[1].Close();
            MessageBoxResult result = MessageBox.Show("Party wurde hinzugefügt!!",
                                "Gratuliere",
                                MessageBoxButton.OK, MessageBoxImage.Exclamation);
        }
    }
}
```

PlaylistAddTrackWindow.xaml

```
<Window x:Class="CrowdDJ.Playstation.Views.PlaylistAddTrackWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="PlaylistAddTrackWindow" Height="260" Width="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>
        <Grid Grid.Row="0">
            <Label Content="Neuen Track hinzufügen!" FontSize="18"
                        HorizontalAlignment="Center"/>
        </Grid>
        <Grid Grid.Row="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Label Grid.Column="0" Content="URL:" HorizontalAlignment="Right"
                               Margin="0,0,5,0"/>
            <TextBox Grid.Column="1" Text="{Binding Path=Url}"
                    HorizontalAlignment="Left" Margin="5,2,0,2"  Width="120" />
```

```
</Grid>
<Grid Grid.Row="2">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Titel:" HorizontalAlignment="Right"
           Margin="0,0,5,0"/>
    <TextBox Grid.Column="1" Text="{Binding Path=Title}"
           HorizontalAlignment="Left" Margin="5,2,0,2"  Width="120" />
</Grid>
<Grid Grid.Row="3">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Artist:" HorizontalAlignment="Right"
           Margin="0,0,5,0"/>
    <TextBox Grid.Column="1" Text="{Binding Path=Artist}"
           HorizontalAlignment="Left" Margin="5,2,0,2"  Width="120" />
</Grid>
<Grid Grid.Row="4">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Länge:" HorizontalAlignment="Right"
           Margin="0,0,5,0"/>
    <TextBox Grid.Column="1" Text="{Binding Path=Length}"
           HorizontalAlignment="Left" Margin="5,2,0,2"  Width="120" />
</Grid>
<Grid Grid.Row="5">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Genre:" HorizontalAlignment="Right"
           Margin="0,0,5,0"/>
    <TextBox Grid.Column="1" Text="{Binding Path=Genre}"
           HorizontalAlignment="Left" Margin="5,2,0,2" Width="120" />
</Grid>
<Grid Grid.Row="6">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Content="Video:" HorizontalAlignment="Right"
           Margin="0,0,5,0"/>
    <CheckBox Grid.Column="1" IsChecked="{Binding Path=IsVideo}"
           HorizontalAlignment="Left" Margin="60, 7,0,0" Width="120" />
</Grid>
<Grid Grid.Row="7">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Width="100" Content="Track hinzufügen"
           HorizontalAlignment="Center" Margin="2"
           Command="{Binding Path=AddNewTrackCommand}"/>
    <Button Grid.Column="1" Width="100" Content="Abbrechen"
           HorizontalAlignment="Center" Margin="2"
           Command="{Binding Path=CancelAddTrackCommand}"/>
</Grid>
```

```
    </Grid> </Window>
```

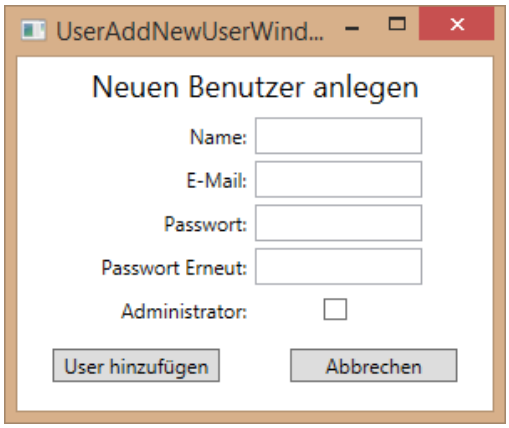
PlaylistAddTrackWindow.xaml.cs

```
namespace CrowdDJ.Playstation.Views
{
    /// <summary>
    /// Interaktionslogik für PlaylistAddTrackWindow.xaml
    /// </summary>
    public partial class PlaylistAddTrackWindow : Window
    {
        public PlaylistAddTrackWindow()
        {
            InitializeComponent();
            this.DataContext = new PlaylistAddNewTrackVM();
        }
    }
}
```

PlaylistAddNewTrackVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    class PlaylistAddNewTrackVM : ViewModelBase
    {
        public string Title { get; set; }
        public string Artist { get; set; }
        public string Url { get; set; }
        public int Length { get; set; }
        public string Genre { get; set; }
        public bool IsVideo { get; set; }

        public ICommand AddNewTrackCommand { get; private set; }
        public ICommand CancelAddTrackCommand { get; private set; }

        private ICrowdDJBL bl = new CrowdDJBL();

        public PlaylistAddNewTrackVM()
        {
            this.AddNewTrackCommand = new RelayCommand(this.AddNewTrack);
            this.CancelAddTrackCommand = new RelayCommand(this.CancelAddTrack);
        }

        private void AddNewTrack(object obj)
        {
            bl.InsertTrack(new Track(Title, Artist, Url, Length, Genre, IsVideo));
            Application.Current.Windows[1].Close();
            MessageBoxResult result = MessageBox.Show("Track wurde hinzugefügt!!",
                                                      "Gratuliere",
                                                MessageBoxButton.OK,
                                                MessageBoxImage.Exclamation);
        }
```

```
        private void CancelAddTrack(object obj)
        {
            Application.Current.Windows[1].Close();
        }
    }
}
```



UserAddNewUserWindow.xaml

```
<Window x:Class="CrowdDJ.Playstation.Views.UserAddNewUserWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:CrowdDJ.Playstation.UserControls"
        Title="UserAddNewUserWindow" Height="250" Width="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Label Grid.Row="0" Content="Neuen Benutzer anlegen" FontSize="18"
                    HorizontalAlignment="Center"/>
        <Grid Grid.Row="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
                <RowDefinition Height="auto" />
            </Grid.RowDefinitions>
            <Grid Grid.Row="0">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="1*" />
                    <ColumnDefinition Width="1*" />
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="Name:"
                       HorizontalAlignment="Right" />
                <TextBox Grid.Column="1" Width="100" Margin="0,2,0,2" Text="{Binding
                          Path=Name}"
                         HorizontalAlignment="Left"/>
            </Grid>
            <Grid Grid.Row="1">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="E-Mail:"
```

```
                    HorizontalAlignment="Right"/>
                <TextBox Grid.Column="1" Width="100" Margin="0,2,0,2" Text="{Binding
                        Path=Email}" HorizontalAlignment="Left"/>
            </Grid>
            <Grid Grid.Row="2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="Passwort:"
                    HorizontalAlignment="Right"/>
                <TextBox Grid.Column="1" Width="100" Margin="0,2,0,2" Text="{Binding
              Path=Password}" HorizontalAlignment="Left" Name="txtPassword"/>
            </Grid>
            <Grid Grid.Row="3">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="Passwort Erneut:"
                    HorizontalAlignment="Right"/>
                <TextBox Grid.Column="1" Width="100" Margin="0,2,0,2" Text="{Binding
                    Path=RePassword}" HorizontalAlignment="Left"
                                    Name="txtRePassword"/>
            </Grid>
            <Grid Grid.Row="4">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Label Grid.Column="0" Content="Administrator:"
                    HorizontalAlignment="Right"/>
                <CheckBox Grid.Column="1" Margin="40,5,0,0" IsChecked="{Binding
                    Path=IsAdmin}" HorizontalAlignment="Left" IsThreeState="False"/>
            </Grid>
            <Grid Grid.Row="5" Margin="0,10,0,0">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Button Name="bttnAddUser" Content="User hinzufügen" Grid.Column="0"
                        HorizontalAlignment="Center" Width="100" Command="{Binding
                                AddNewUserCommand}" />
                <Button Name="bttnCancel" Content="Abbrechen" Grid.Column="1"
                        HorizontalAlignment="Center" Width="100" Command="{Binding
                                NewUserCancelCommand}"/>
            </Grid>
        </Grid>
    </Grid>
</Window>
```

UserAddNewUserWindow.xaml.cs

```
namespace CrowdDJ.Playstation.Views
{
    /// <summary>
    /// Interaktionslogik für UserAddNewUserWindow.xaml
    /// </summary>
    public partial class UserAddNewUserWindow : Window
    {
        public UserAddNewUserWindow()
        {
            InitializeComponent();
            this.DataContext = new UserAddNewUserVM();
        }
    }
}
```

UserAddNewUserVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace CrowdDJ.Playstation.ViewModels
{
    class UserAddNewUserVM : ViewModelBase
    {
        private ICrowdDJBL bl;
        public string Name {get;set;}
        public string Email { get; set; }
        public string Password { get; set; }
        public string RePassword { get; set; }
        public bool IsAdmin { get; set; }
        public ICommand AddNewUserCommand { get; private set; }
        public ICommand NewUserCancelCommand { get; private set; }

        public UserAddNewUserVM()
        {
            bl = new CrowdDJBL();
            this.AddNewUserCommand = new RelayCommand(this.AddNewUser);
            this.NewUserCancelCommand = new RelayCommand(this.NewUserCancel);

        }
        private void NewUserCancel(object obj)
        {
            Application.Current.Windows[1].Close();
        }
        private void AddNewUser(object obj)
        {
            MessageBoxResult result;
            if (Password.Equals(RePassword))
            {
                bl.InsertUser(new User(Name, Password, Email, IsAdmin));
                Application.Current.Windows[1].Close();
                result = MessageBox.Show("User has successfully been added!",
                                         "Success",
```

```
                                        MessageBoxButton.OK, MessageBoxImage.Exclamation);
            }
            else
            {
                result = MessageBox.Show("Passwords are not equal!", "Error",
                                                MessageBoxButton.OK,
                                                MessageBoxImage.Warning);
            }
        }
    }
}
```

CrowdDJ.QRCode

```
//Code provided by Raffael Hermann https://github.com/codebude/QRCoder
```

CrowdDJ.SimpleVote

Die Anwendung für Gäste einer Party. Mit dem QRCode und ihrer E-Mail Adresse können sie sich zur Party anmelden. Es scheint eine Playlist der Party aus.

MainWindow.xaml

```
<Window x:Class="CrowdDJ.SimpleVote.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="CrowdDJ Für Gäste" Height="350" Width="525">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
        </Grid.RowDefinitions>
        <Label Content="Willkommen Gast!" FontSize="20" HorizontalAlignment="Center"
                                                                    />
        <Grid Grid.Row="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Grid Grid.Column="0" VerticalAlignment="Center">
                <Grid.RowDefinitions>
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
```

```
                </Grid.RowDefinitions>
                <Grid Grid.Row="0">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*" />
                        <ColumnDefinition Width="*" />
                    </Grid.ColumnDefinitions>
                    <Label Grid.Column="0" Content="Party ID hier eingeben: " />
                    <TextBox Grid.Column="1" Text="{Binding Path=PartyId}" Width="110"
                             Margin="2"/>
                </Grid>
                <Grid Grid.Row="1">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*" />
                        <ColumnDefinition Width="*" />
                    </Grid.ColumnDefinitions>
                    <Label Grid.Column="0" Content="Email hier eingeben: " />
                    <TextBox Grid.Column="1" Name="txtEmail" Text="{Binding
                             Path=Email}" Width="110" Margin="2"/>
                </Grid>
                <Grid Grid.Row="2">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*" />
                        <ColumnDefinition Width="*" />
                    </Grid.ColumnDefinitions>
                    <Button Grid.Column="0" Content="Einloggen" Margin="2"
                             Command="{Binding Path=LogInGuestCommand}"
                           CommandParameter="{Binding ElementName=txtEmail}"/>
                    <Button Grid.Column="1" Content="Beenden" Width="120" Margin="2"
                             Command="{Binding Path=ExitSimpleVoteCommand}"/>
                </Grid>
            </Grid>
            <Grid Grid.Column="1">
                <Grid.RowDefinitions>
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="*" />
                </Grid.RowDefinitions>
                <ComboBox Grid.Row="0" ItemsSource="{Binding Path=AllParties}"
                      SelectedItem="{Binding Path=SelectedParty}">
                    <ComboBox.ItemTemplate>
                        <DataTemplate>
                            <TextBlock Text="{Binding Path=Name}" />
                        </DataTemplate>
                    </ComboBox.ItemTemplate>
                </ComboBox>
                <Image Grid.Row="1" Source="{Binding Path=QrPicture}"/>
            </Grid>
        </Grid>
    </Grid>
</Window>
```

MainWindow.xaml.cs

```
namespace CrowdDJ.SimpleVote
{
    /// <summary>
    /// Interaktionslogik für MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            this.DataContext = new SimpleVoteVM();
        }
    }
}
```

SimpleVoteVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using CrowdDJ.QRCode;
using CrowdDJ.SimpleVote.SimpleVoteUser;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using System.Windows.Interop;
using System.Windows.Media.Imaging;

namespace CrowdDJ.SimpleVote
{
    public class SimpleVoteVM : ViewModelBase
    {
        public string Email { get; set; }
        public string PartyId { get; set; }

        private Party selectedParty;
        public Party SelectedParty
        {
            get { return selectedParty; }
            set
            {
                selectedParty = value;
                OnPropertyChanged("IsSelectedParty");
                UpdateQRCode();
            }
        }

        private BitmapSource qrPicture { get; set; }
        public BitmapSource QrPicture
        {
            get { return qrPicture; }
            set { qrPicture = value; OnPropertyChanged("QrPicture"); }
```

```
        }

        private ObservableCollection<Party> allParties;
        public ObservableCollection<Party> AllParties
        {
            get { return allParties; }
            set { allParties = value; OnPropertyChanged("AllParties"); }
        }

        private User user;


        public ICommand ExitSimpleVoteCommand { get; private set; }
        public ICommand LogInGuestCommand { get; private set; }

        private ICrowdDJBL bl = new CrowdDJBL();

        public SimpleVoteVM()
        {
            this.ExitSimpleVoteCommand = new RelayCommand(this.ExitSimpleVote);
            this.LogInGuestCommand = new RelayCommand(this.LogInGuest);
            AllParties = bl.GetAllParties();
            SelectedParty = AllParties.First();
        }

        private void LogInGuest(object obj)
        {
            if (bl.FindUserByEmail(Email) == null ||
                bl.FindPartyById(PartyId) == null)
            {
                MessageBoxResult result = MessageBox.Show("Benutzer nicht vorhanden",
                                                          "Error",
                                                          MessageBoxButton.OK,
                                                          MessageBoxImage.Error);

            }
            else
            {
                Window window = new SimpleVoteUserWindow(PartyId);
                window.Show();
                Application.Current.Windows[0].Close();
            }
        }

        private void ExitSimpleVote(object obj)
        {
            Application.Current.Windows[0].Close();
        }

        private void UpdateQRCode()
        {
            Bitmap cur;
            QRCodeGenerator qrGenerator = new QRCodeGenerator();
            QRCodeGenerator.QRCode qrCode = qrGenerator.CreateQrCode(
                                        SelectedParty.PartyId,
                                        QRCodeGenerator.ECCLevel.Q);
            cur = qrCode.GetGraphic(20);
            QrPicture = BitMapConverter.ToWpfBitmap(cur);

        }


    }
}
```

BitMapConverter.cs

Um den QRCode in der WPF ausgeben zu können wurde ein BitMapConverter benötigt.

```
namespace CrowdDJ.SimpleVote
{
    public static class BitMapConverter
    {
        public static BitmapSource ToWpfBitmap(this Bitmap bitmap)
        {
            using (MemoryStream stream = new MemoryStream())
            {
                bitmap.Save(stream, ImageFormat.Bmp);

                stream.Position = 0;
                BitmapImage result = new BitmapImage();
                result.BeginInit();
                result.CacheOption = BitmapCacheOption.OnLoad;
                result.StreamSource = stream;
                result.EndInit();
                result.Freeze();
                return result;
            }
        }
    }
}
```

SimpleVoteUserWindow.xaml

```
<Window x:Class="CrowdDJ.SimpleVote.SimpleVoteUser.SimpleVoteUserWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="SimpleVoteUserWindow" Height="300" Width="400">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Label Grid.Row="0" Content="{Binding Path=CurParty}" FontSize="20"
                            HorizontalAlignment="Center"/>
        <DataGrid Grid.Row="1" ItemsSource="{Binding Path=AllTracks}"
                            AutoGenerateColumns="False">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Titel" Binding="{Binding Path=Title}"
                                    Width="auto" />
                <DataGridTextColumn Header="Artist" Binding="{Binding Path=Artist}"
                                     Width="auto" />
                <DataGridTextColumn Header="Genre" Binding="{Binding Path=Genre}"
                                    Width="auto" />
                <DataGridTextColumn Header="Länge" Binding="{Binding Path=Length}"
                                    Width="auto" />
                <DataGridTextColumn Header="Ist Video" Binding="{Binding
                                    Path=IsVideo}" Width="auto" />
                <DataGridTextColumn Header="Url" Binding="{Binding Path=Url}"
                                    Width="auto" />
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Window>
```

SimpleVoteUserWindow.xaml.cs

```
namespace CrowdDJ.SimpleVote.SimpleVoteUser
{
    /// <summary>
    /// Interaktionslogik für SimpleVoteUserWindow.xaml
    /// </summary>
    public partial class SimpleVoteUserWindow : Window
    {
        public SimpleVoteUserWindow(string PartyId)
        {
            InitializeComponent();
            this.DataContext = new SimpleVoterUserVM(PartyId);
        }
    }
}
```

SimpleVoterUserVM.cs

```
using CrowdDJ.BL;
using CrowdDJ.DomainClasses;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CrowdDJ.SimpleVote.SimpleVoteUser
{
    class SimpleVoterUserVM : ViewModelBase
    {
        public string PartyId { get; set; }

        public string CurParty { get; set; }

        private ObservableCollection<Track> allTracks;
        public ObservableCollection<Track> AllTracks
        {
            get { return allTracks; }
            set { allTracks = value; }
        }
        private ICrowdDJBL bl = new CrowdDJBL();


        public SimpleVoterUserVM(string partyId)
        {
            PartyId = partyId;
            CurParty = bl.FindPartyById(PartyId).Name;
            AllTracks = bl.GetAllTracksInPlaylist(
                           bl.GetPlaylistForParty(PartyId).PlaylistId);
        }
    }
}
```

CrowdDJ.BL

Wie eingangs erwähnt wurde die Businesslogic erweitert und gegen ein Interface programmiert. Die Member sind nun privat und und nur noch über die eigenen Methoden der Businesslogic aufgerufen.

ICrowdDJBL.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CrowdDJ.DomainClasses;
using System.Collections.ObjectModel;

namespace CrowdDJ.BL
{
    public interface ICrowdDJBL
    {
        #region Guest
        bool AddGuest(Guest newGuest);
        bool RemoveGuest(Guest removeGuest);
        bool PartyIsVisitedByGuest(int searchGuestId, string partyId);
        ObservableCollection<User> GetGuestlistForParty(string partyId);
        ObservableCollection<Party> GetAttemptedPartyList(int userId);
        #endregion //Guest

        #region Party
        bool AddParty(Party newParty);
        bool RemovePartyWithId(string partyId);
        bool UpdateParty(Party party, string id);
        Party FindPartyById(string partyId);
        ObservableCollection<Party> FindPartyWithHost(string hostName);
        ObservableCollection<Party> GetAllParties();
        #endregion //Party

        #region Partytweet
        bool AddTweet(Partytweet newTweet);
        bool DeletePartytweet(Partytweet deletePartytweet);
        ObservableCollection<Partytweet> GetTweetsForParty(string partyId);
        ObservableCollection<Partytweet> GetTweetsForUser(int userId);
        ObservableCollection<Partytweet> GetAllTweets();
        #endregion //Partytweet

        #region Playlist
        bool InsertPlaylist(Playlist newPlaylist);
        bool DeletePlaylist(int id);
        bool UpdatePlaylist(int id, Playlist updatedPlaylist);
        Playlist GetPlaylistForParty(string id);
        ObservableCollection<Track> GetAllTracksInPlaylist(int playlistId);
        ObservableCollection<Playlist> GetAllPlaylists();
        #endregion //Playlist

        #region Track
        bool InsertTrack(Track newTrack);
        bool RemoveTrackWithId(int id);
        bool UpdateTrack(Track track, int id);
        Track FindTrackById(int id);
        ObservableCollection<Track> FindTrackWithTitle(string title);
        ObservableCollection<Track> FindTracksInGenre(string genre);
        ObservableCollection<Track> FindVideos();
        ObservableCollection<Track> FindSongs();
        ObservableCollection<Track> GetAllTracks();
        #endregion //Track
```

```
        #region Tracklist
        bool InsertIntoTracklist(Tracklist insertIntoTracklist);
        ObservableCollection<Track> GetTracksRecommendedByUser(int userId);
        ObservableCollection<Tracklist> GetAllTracklists();
        #endregion //Tracklist

        #region User
        bool InsertUser(User user);
        bool DeleteUser(int id);
        bool UpdateUser(User user, int id);
        ObservableCollection<User> GetAllUser();
        User FindUserById(int id);
        User FindUserByEmail(string email);
        #endregion //User

        #region Vote
        bool InsertVote(Vote newVote);
        bool AlreadyVotedForTrack(int trackId, int userId, int playlistId);
        int GetVotesForTrack(int trackId, int playlist);
        #endregion //Vote
    }
}
```

CrowdDJBL.cs

```
using CrowdDJ.DAL;
using CrowdDJ.DAO;
using CrowdDJ.Interfaces;
using System;
using System.Configuration;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.ObjectModel;

namespace CrowdDJ.BL
{
    public class CrowdDJBL : ICrowdDJBL
    {
        private IDataBase database = null;
        private IGuest guest = null;
        private IParty party = null;
        private IPartytweet partytweet = null;
        private IPlaylist playlist = null;
        private ITrack track = null;
        private ITracklist tracklist = null;
        private IUser user = null;
        private IVote vote = null;

        public CrowdDJBL()
        {
            string s =
ConfigurationManager.ConnectionStrings["CrowdDJ.Properties.Settings.CrowdDJDBConnectio
nString"].ConnectionString;
            database = new DataBase(s);
            guest = new GuestDAO(database);
            party = new PartyDAO(database);
            partytweet = new PartytweetDAO(database);
            playlist = new PlaylistDAO(database);
            track = new TrackDAO(database);
            tracklist = new TracklistDAO(database);
```

```
            user = new UserDAO(database);
            vote = new VoteDAO(database);
        }

        #region Guest
        public bool AddGuest(DomainClasses.Guest newGuest)
        {
            return guest.AddGuest(newGuest);
        }
        public bool RemoveGuest(DomainClasses.Guest removeGuest)
        {
            return guest.RemoveGuest(removeGuest);
        }
        public bool PartyIsVisitedByGuest(int searchGuestId, string partyId)
        {
            return guest.PartyIsVisitedByGuest(searchGuestId, partyId);
        }
        public ObservableCollection<DomainClasses.User> GetGuestlistForParty(string
partyId)
        {
            return guest.GetGuestlistForParty(partyId);
        }
        public ObservableCollection<DomainClasses.Party> GetAttemptedPartyList(int
userId)
        {
            return guest.GetAttemptedPartyList(userId);
        }
        #endregion

        #region Party
        public bool AddParty(DomainClasses.Party newParty)
        {
            return party.AddParty(newParty);
        }
        public bool RemovePartyWithId(string partyId)
        {
            return party.RemovePartyWithId(partyId);
        }
        public bool UpdateParty(DomainClasses.Party updateParty, string id)
        {
            return party.UpdateParty(updateParty, id);
        }
        public DomainClasses.Party FindPartyById(string partyId)
        {
            return party.FindPartyById(partyId);
        }
        public ObservableCollection<DomainClasses.Party> FindPartyWithHost(string
hostName)
        {
            return party.FindPartyWithHost(hostName);
        }
        public ObservableCollection<DomainClasses.Party> GetAllParties()
        {
            return party.GetAllParties();
        }
        #endregion

        #region Partytweet
        public bool AddTweet(DomainClasses.Partytweet newTweet)
        {
            return partytweet.AddTweet(newTweet);
        }
```

```
public bool DeletePartytweet(DomainClasses.Partytweet deletePartytweet)
{
    return partytweet.DeletePartytweet(deletePartytweet);
}

public ObservableCollection<DomainClasses.Partytweet> GetTweetsForParty(string
                                                                    partyId)
{
    return partytweet.GetTweetsForParty(partyId);
}

public ObservableCollection<DomainClasses.Partytweet> GetAllTweets()
{
    return partytweet.GetAllTweets();
}
public ObservableCollection<DomainClasses.Partytweet> GetTweetsForUser(int
                                                                    userId)
{
    return partytweet.GetTweetsForUser(userId);
}
#endregion

#region Playlist
public bool InsertPlaylist(DomainClasses.Playlist newPlaylist)
{
    return playlist.InsertPlaylist(newPlaylist);
}

public bool DeletePlaylist(int id)
{
    return playlist.DeletePlaylist(id);
}

public bool UpdatePlaylist(int id, DomainClasses.Playlist updatedPlaylist)
{
    return playlist.UpdatePlaylist(id, updatedPlaylist);
}

public DomainClasses.Playlist GetPlaylistForParty(string id)
{
    return playlist.GetPlaylistForParty(id);
}

public ObservableCollection<DomainClasses.Track> GetAllTracksInPlaylist(int
                    playlistId)
{
    return playlist.GetAllTracksInPlaylist(playlistId);
}

public ObservableCollection<DomainClasses.Playlist> GetAllPlaylists()
{
    return playlist.GetAllPlaylists();
}
#endregion

#region Track
public bool InsertTrack(DomainClasses.Track newTrack)
{
    return track.InsertTrack(newTrack);
}

public bool RemoveTrackWithId(int id)
{
```

```
        return track.RemoveTrackWithId(id);
    }

    public bool UpdateTrack(DomainClasses.Track updateTrack, int id)
    {
        return track.UpdateTrack(updateTrack, id);
    }

    public DomainClasses.Track FindTrackById(int id)
    {
        return track.FindTrackById(id);
    }

    public ObservableCollection<DomainClasses.Track> FindTrackWithTitle(string
                                                                         title)
    {
        return track.FindTrackWithTitle(title);
    }

    public ObservableCollection<DomainClasses.Track> FindTracksInGenre(string
                                                                        genre)
    {
        return track.FindTracksInGenre(genre);
    }

    public ObservableCollection<DomainClasses.Track> FindVideos()
    {
        return track.FindVideos();
    }

    public ObservableCollection<DomainClasses.Track> FindSongs()
    {
        return track.FindSongs();
    }

    public ObservableCollection<DomainClasses.Track> GetAllTracks()
    {
        return track.GetAllTracks();
    }
    #endregion

    #region Tracklist
    public bool InsertIntoTracklist(DomainClasses.Tracklist insertIntoTracklist)
    {
        return tracklist.InsertIntoTracklist(insertIntoTracklist);
    }

    public ObservableCollection<DomainClasses.Track>
                GetTracksRecommendedByUser(int userId)
    {
        return tracklist.GetTracksRecommendedByUser(userId);
    }

    public ObservableCollection<DomainClasses.Tracklist> GetAllTracklists()
    {
        return tracklist.GetAllTracklists();
    }
    #endregion

    #region User
    public bool InsertUser(DomainClasses.User newUser)
    {
        return user.InsertUser(newUser);
```

```
        }

        public bool DeleteUser(int id)
        {
            return user.DeleteUser(id);
        }

        public bool UpdateUser(DomainClasses.User updateUser, int id)
        {
            return user.UpdateUser(updateUser, id);
        }

        public ObservableCollection<DomainClasses.User> GetAllUser()
        {
            return user.GetAllUser();
        }

        public DomainClasses.User FindUserById(int id)
        {
            return user.FindUserById(id);
        }
        public DomainClasses.User FindUserByEmail(string email)
        {
            return user.FindUserByEmail(email);
        }
        #endregion

        #region Vote
        public bool InsertVote(DomainClasses.Vote newVote)
        {
            return vote.InsertVote(newVote);
        }

        public bool AlreadyVotedForTrack(int userId, int trackId, int playlistId)
        {
            return vote.AlreadyVotedForTrack(userId, trackId, playlistId);
        }

        public int GetVotesForTrack(int trackId, int playlistId)
        {
            return vote.GetVotesForTrack(trackId, playlistId);
        }
        #endregion //Vote
    }
}
```