MainWindow Code XAML :

```
<Window x:Class="TP3_WPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:TP3_WPF"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Window.Resources>
        <ToolTip x:Key="PreviewScreen" x:Shared="True" Background="Transparent"
                        Placement="Right" Name="previewToolTip">
            <Border BorderBrush="RoyalBlue" BorderThickness="2" CornerRadius="2">
                <Image Source="{Binding Mode=OneWay}" Opacity="0.5" />
            </Border>
        </ToolTip>
        <Style TargetType="{x:Type ListBoxItem}">
            <Setter Property="Background" Value="Transparent" />
            <!--<Setter Property="MaxHeight" Value="100" />-->
            <Setter Property="Margin" Value="5" />
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type ListBoxItem}" >
                        <Border SnapsToDevicePixels="True"
HorizontalAlignment="Stretch"
                                    VerticalAlignment="Stretch"
Background="{TemplateBinding Background}">
                            <Image Source="{Binding Mode=OneWay}" Opacity="1"
Cursor="Hand" x:Name="image"/>
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsSelected" Value="True">
                                <Setter Property="Background" Value="#445B6249"
/>
                            </Trigger>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Trigger.Setters>
                                    <Setter Property="ToolTip"
Value="{StaticResource PreviewScreen}"/>
                                </Trigger.Setters>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
        <Style TargetType="{x:Type ListBox}" x:Key="PhotoListBoxStyle">
            <Setter Property="Foreground" Value="White" />
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type ListBox}" >
                        <WrapPanel Margin="5" IsItemsHost="True"
Orientation="Horizontal"
                                            VerticalAlignment="Top"
HorizontalAlignment="Stretch"
                                            ItemHeight="{Binding
ElementName=ZoomSlider, Path='Value'}"
                                            ItemWidth="{Binding
ElementName=ZoomSlider, Path='Value'}"
                                   />
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
        <Style x:Key="LabelHeader" TargetType="{x:Type Label}">
            <Setter Property="Background">
                <Setter.Value>
                    <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,0.5" >
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset="0.5" Color="{x:Static
SystemColors.AppWorkspaceColor}" />
                            <GradientStop Offset="2" Color="Transparent" />
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Setter.Value>
            </Setter>
            <Setter Property="Foreground" Value="White" />
            <Setter Property="FontWeight" Value="Bold" />
        </Style>
        <Style x:Key="LabelPhoto" TargetType="{x:Type Label}" >
            <Setter Property="Background" Value="GhostWhite" />
            <Setter Property="Foreground" Value="DarkRed" />
            <Setter Property="Height" Value="25"/>
            <Setter Property="FontWeight"  Value="Bold" />
        </Style>
    </Window.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="250" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" Grid.Column="0">
            <Button x:Name="bt1" Content="Sélectionner" DockPanel.Dock="Right"
Margin="0,0,0,0" VerticalAlignment="Top" Click="bt1_Click"/>
            <TextBox x:Name="tb1"   DockPanel.Dock="Left" Margin="0,0,0,0" />
        </DockPanel>
        <GroupBox Header="Sélection" Grid.Row="1">
            <ScrollViewer VerticalScrollBarVisibility="Auto">
                <ListBox x:Name="lb1" Style="{StaticResource PhotoListBoxStyle}"
SelectionChanged="lb1_SelectionChanged" />
            </ScrollViewer>
        </GroupBox>
        <GroupBox Grid.Column="1" Header="Propriétés" Grid.Row="1">
            <ScrollViewer >
                <StackPanel>
                    <Label Content="Source" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_1" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Date prise de vue" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_2" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Titre" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_3" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Appareil" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
```

```
                    <Label x:Name="lb_4" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Software" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_5" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Iso" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_6" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Ouverture" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_7" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                    <Label Content="Focale" Margin="0,0,0,0"
Style="{DynamicResource LabelHeader}"/>
                    <Label x:Name="lb_8" Content="" Margin="0,0,0,0"
Style="{DynamicResource LabelPhoto }"/>
                </StackPanel>
            </ScrollViewer>

        </GroupBox>
        <DockPanel Grid.Column="1" Grid.Row="0">
            <Button Content="Diaporama" Width="70" Margin="0,0,0,0"
DockPanel.Dock="Left" Click="Button_Click_1"/>
            <Label Content="Zoom" DockPanel.Dock="Left"/>
            <Slider x:Name="ZoomSlider" Minimum="80" Maximum="320" Value="160"
TickFrequency="80"
                    TickPlacement="BottomRight" SmallChange="5"
LargeChange="20"/>

        </DockPanel>

        <!--<Label Grid.Column="1" Content="Source2" Margin="0,0,0,0"
Grid.Row="1"  Style="{DynamicResource LabelHeader}"/>-->



    </Grid>
</Window>
```

MainWindow Code C# :

```
using System;
using System.Collections.Generic;
using System.IO;
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.Forms;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace TP3_WPF
{
    /// <summary>
    /// Logique d'interaction pour MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        Photo ph;

        public MainWindow()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {

        }

        private void bt1_Click(object sender, RoutedEventArgs e)
        {
            FolderBrowserDialog fbd = new FolderBrowserDialog();
            DialogResult dr = fbd.ShowDialog();
            if (dr == System.Windows.Forms.DialogResult.OK)
            {
                DirectoryInfo di = new
DirectoryInfo(fbd.SelectedPath.ToString()); // fait pointer di sur le chemin
sélectionné
                tb1.Text = di.FullName;
                //foreach (FileInfo fi in di.GetFiles("*.jpg")  // Pour les .jpg
                foreach (FileInfo fi in
di.GetFiles("*.jpg").Concat(di.GetFiles("*.png"))) // .jpg et .png
                {
                    lb1.Items.Add(fi.FullName);
                }
            }
        }
        private void lb1_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (lb1.SelectedItems.Count > 0)  // vérifie si il y a bien un item
de sémectionner
            {
                //lb_1.Content = lb1.SelectedItem.ToString();

                ph = new Photo(lb1.SelectedItem.ToString());
                lb_1.Content = ph.Source;
                lb_2.Content = ph.Metadata.DateTaken.ToString();
                lb_3.Content = ph.Metadata.Title;
                lb_4.Content = ph.Metadata.Camera;
                lb_5.Content = ph.Metadata.Application;
                lb_6.Content = ph.Metadata.IsoSpeed;
                lb_7.Content = ph.Metadata.Ouverture;
                lb_8.Content = ph.Metadata.Focale;
            }
        }

        private void Button_Click_1(object sender, RoutedEventArgs e) // Bouton
Diaporama
        {
            Diapo dp = new Diapo();
            dp.sDiapo.Clear();
            foreach (String sPath in lb1.Items)
            {
                dp.sDiapo.Add(sPath);
            }
            dp.Show();
        }
    }
}
```

Classe Photo :

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

namespace TP3_WPF
{
    internal class Photo
    {
        private String _path;
        private Uri _source;
        private PhotoMetadata _phmetadata; // Object de la sousclasse
PhotoMetadata

        public Photo(string path) // Constructeur
        {
            _path = path;
            _source = null;
            if (path != null)
            {
                _source = new Uri(path);
                _phmetadata = new PhotoMetadata(_source);
            }
        }
        public override string ToString() // Surcharge de la méthode ToString
        {
            return _source.ToString();
        }
        public string Source { get { return _path; } }
        public PhotoMetadata Metadata { get { return _phmetadata; } } // méthode
retournént la variable privée _phmetadata
        public class PhotoMetadata  // Sous classe PhotoMetadata
        {
            public BitmapMetadata _metadata;
            public PhotoMetadata(Uri imageUri) // Constructeur de la sous classe
            {
                BitmapFrame frame = BitmapFrame.Create(imageUri,
                        BitmapCreateOptions.DelayCreation,
BitmapCacheOption.None);
                _metadata = (BitmapMetadata)frame.Metadata;
            }
            public DateTime? DateTaken
            {
                get
                {
                    Object val = _metadata.DateTaken;
                    if (val != null)
                    {
                        return
Convert.ToDateTime(_metadata.DateTaken.ToString());
                    }
                    else
                    {
                        return null;
                    }
                }
            }
            public String Title
            {
                get
                {
                    try
                    {
                        Object val = _metadata.Title;
                        return (val != null ? (string)val : String.Empty);
                    }
                    catch { return String.Empty; }
                }
            }
            public String Camera
            {
                get
                {
                    try
                    {
                        Object val = _metadata.CameraManufacturer + " " +
_metadata.CameraModel;
                        return (val != null ? (string)val : String.Empty);
                    }
                    catch { return String.Empty; }
                }
            }
            public String Application
            {
                get
                {
                    try
                    {
                        Object val = _metadata.ApplicationName;
                        return (val != null ? (string)val : String.Empty);
                    }
                    catch { return String.Empty; }
                }
            }

            public String IsoSpeed
            {
                get
                {
                    Object val =
QueryMetadata("/app1/ifd/exif/subifd:{uint=34855}");
                    if (val != null)
                    {
                        return val.ToString();
                    }
                    else
                    {
                        return String.Empty;
                    }
                }
            }
            public String Ouverture
            {
                get
                {
                    Object val =
QueryMetadata("/app1/ifd/exif/subifd:{uint=33437}");
                    if (val != null)
                    {
                        val = ParseUnsignedRational((ulong)val);
                        return val.ToString();
                    }
                    else
                    {
                        return String.Empty;
                    }
                }
            }
            public String Focale
            {
                get
                {
                    Object val =
QueryMetadata("/app1/ifd/exif/subifd:{uint=37386}");
                    if (val != null)
                    {
                        val = ParseUnsignedRational((ulong)val) + " mm";
                        return val.ToString();
                    }
                    else
                    {
                        return String.Empty;
                    }
                }
            }
            private decimal ParseUnsignedRational(ulong exifValue)
            {
                return (decimal)(exifValue & 0xFFFFFFFFL) / (decimal)((exifValue
& 0xFFFFFFFF00000000L) >> 32);
            }
            private decimal ParseSignedRational(long exifValue)
            {
                return (decimal)(exifValue & 0xFFFFFFFFL) / (decimal)((exifValue
& 0x7FFFFFFF00000000L) >> 32);
            }
            private object QueryMetadata(string query)
            {
                if (_metadata.ContainsQuery(query))
                    return _metadata.GetQuery(query);
                else
                    return null;
            }
        }

    }
}
```

Diapo code Xaml :

```
<Window x:Class="TP3_WPF.Diapo"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:TP3_WPF"
        mc:Ignorable="d"
        Title="Diapo" Height="450" Width="800" Activated="Window_Activated">
```

```xml
    <Window.Resources>
        <Storyboard x:Key="VisibleToInvisible" 
Completed="VisibleToInvisible_Completed" >
            <DoubleAnimation Storyboard.TargetName="TransparentStop" 
Storyboard.TargetProperty="Offset" To="0" Duration="0:0:2" />
            <DoubleAnimation Storyboard.TargetName="BlackStop" 
Storyboard.TargetProperty="Offset" To="0" Duration="0:0:2" />
        </Storyboard>
        <Storyboard x:Key="InvisibleToVisible" 
Completed="InvisibleToVisible_Completed" >
            <DoubleAnimation Storyboard.TargetName="TransparentStop" 
Storyboard.TargetProperty="Offset" To="1" Duration="0:0:2" />
            <DoubleAnimation Storyboard.TargetName="BlackStop" 
Storyboard.TargetProperty="Offset" To="1" Duration="0:0:2" />
        </Storyboard>
    </Window.Resources>
    <Window.Triggers>
        <EventTrigger RoutedEvent="Window.Loaded">
            <EventTrigger.Actions>
                <BeginStoryboard Storyboard="{StaticResource 
VisibleToInvisible}"/>
            </EventTrigger.Actions>
        </EventTrigger>
    </Window.Triggers>
    <Grid>
        <Image x:Name="Image2" ></Image>
        <Image x:Name="Image1" >
            <Image.OpacityMask>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,0">
                    <GradientStop Offset="1" Color="Black" x:Name="BlackStop"/>
                    <GradientStop Offset="1" Color="Transparent" 
x:Name="TransparentStop"/>
                </LinearGradientBrush>
            </Image.OpacityMask>
        </Image>
    </Grid>
</Window>
```

Diapo code C# :

```csharp
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.Animation;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace TP3_WPF
{
    /// <summary>
    /// Logique d'interaction pour Diapo.xaml
    /// </summary>
    public partial class Diapo : Window
    {
        public List<String> sDiapo = new List<String>();
        int indice;
        public Diapo()
        {
            InitializeComponent();
        }

        private void Window_Activated(object sender, EventArgs e)
        {
            indice = 0;
            ImageSourceConverter s = new ImageSourceConverter();
            Image1.Source = (ImageSource)s.ConvertFromString(sDiapo[0]);
            ImageSourceConverter s2 = new ImageSourceConverter();
            Image2.Source = (ImageSource)s2.ConvertFromString(sDiapo[1]);
        }

        private void VisibleToInvisible_Completed(object sender, EventArgs e)
        {
            ImageSourceConverter s = new ImageSourceConverter();
            Image1.Source = (ImageSource)s.ConvertFromString(sDiapo[indice]);
            indice++;
            if (indice == sDiapo.Count) { indice = 0; }

            Storyboard sb = (Storyboard)this.FindResource("InvisibleToVisible");
            sb.Begin();

        }
        //
        private void InvisibleToVisible_Completed(object sender, EventArgs e)
        {
            ImageSourceConverter s = new ImageSourceConverter();
            Image2.Source = (ImageSource)s.ConvertFromString(sDiapo[indice]);
            indice++;
            if (indice == sDiapo.Count) { indice = 0; }


            Storyboard sb2 = (Storyboard)this.FindResource("VisibleToInvisible");
            sb2.Begin();

        }
    }
}
```