ГУАП

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| преподаватель |  |  |  | И.А. Юрьева |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |
| ОТЧЕТЫ О ЛАБОРАТОРНЫХ РАБОТАХ | | | | |
|  | | | | |
| по дисциплине: МДК 01.01 Разработка программных модулей | | | | |
|  | | | | |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 021к |  |  |  | В.Д. Панков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

СОДЕРЖАНИЕ

[1. Лабораторная работа № 1 3](#_Toc116391770)

[1.1. Дополнительные задания на работу с файловой системой 10](#_Toc116391771)

# 1. Лабораторная работа № 1

Тема: Создание приложения «Микропроводник».

Цель работы: получение практических навыков при работе с пространством имен System.IO.

Задание 1. Разработать приложение «Микропроводник», примерный вид которого представлен на Рисунке 1.

Рисунок 1 – Приложение Микропроводник

На форме список всех дисков загружается в компонент comboBox1. Список всех каталогов для данного диска загружается в listBox1. Список файлов, находящихся в выбранном каталоге, отображается listBox2.

Задание 2. Используя дополнительные компоненты,

* для выделенного диска необходимо выводить сведения: объем диска, свободное пространство;
* для выделенного каталога: полное название каталога, время создания каталога, корневой каталог.

Задание 3. При выделении файла в списке должно запускаться соответствующее приложение.

Задание 4. Сохранить в отдельный текстовый файл имена файлов, которые открывались за последние 10 секунд работы приложения.

Примечание. При выполнении задания необходимо работать с типом DateTime.

DateTime.Now – возвращает текущее время;

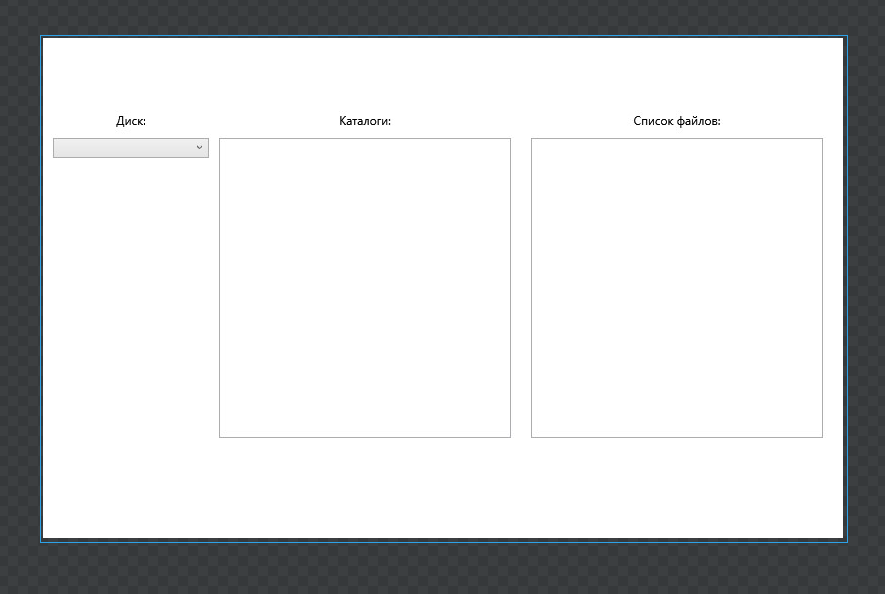
Convert.ToDateTime – преобразование строки в тип DateTime.

Самостоятельно необходимо разобраться как работать с секундами.

Решение:

Разметка:

<Window x:Class="WorkWithFiles.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"  
 mc:Ignorable="d"  
 Title="MainWindow" Height="500" Width="800"  
 MinHeight="500" MinWidth="700">  
 <Grid Margin="10, 0">  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="2\*" />  
 <ColumnDefinition Width="2\*" />  
 </Grid.ColumnDefinitions>  
 <Grid.RowDefinitions>  
 <RowDefinition Height="\*" />  
 <RowDefinition Height="3\*" />  
 <RowDefinition Height="\*" />  
 </Grid.RowDefinitions>  
 <TextBlock TextAlignment="Center" VerticalAlignment="Bottom" Margin="10">Диск:</TextBlock>  
 <StackPanel Grid.Row="1" Grid.Column="0" Orientation="Vertical">  
 <ComboBox SelectionChanged="HardDrives\_OnSelected" Name="HardDrivesComboBox" Height="20"  
 VerticalAlignment="Top" />  
 <TextBlock Name="HardDriveInformation" TextWrapping="Wrap" />  
 </StackPanel>  
 <TextBlock Grid.Row="0" Grid.Column="1" TextAlignment="Center" VerticalAlignment="Bottom" Margin="10">Каталоги:</TextBlock>  
 <TextBlock Margin="10" Grid.Row="2" Grid.Column="1" Name="FolderInformation" DockPanel.Dock="Bottom"  
 TextWrapping="Wrap" />  
 <ListBox Grid.Row="1" Grid.Column="1" VerticalAlignment="Stretch"  
 SelectionChanged="FoldersListBox\_OnSelectionChanged" Name="FoldersListBox"  
 Margin="10, 0" />  
  
 <TextBlock Grid.Row="0" Grid.Column="2" TextAlignment="Center" VerticalAlignment="Bottom" Margin="10">Список файлов:</TextBlock>  
 <ListBox Grid.Column="2" Name="FilesListBox" Grid.Row="1" SelectionMode="Single"  
 SelectionChanged="FilesListBox\_OnSelectionChanged"  
 Margin="10, 0" />  
 </Grid>  
</Window>



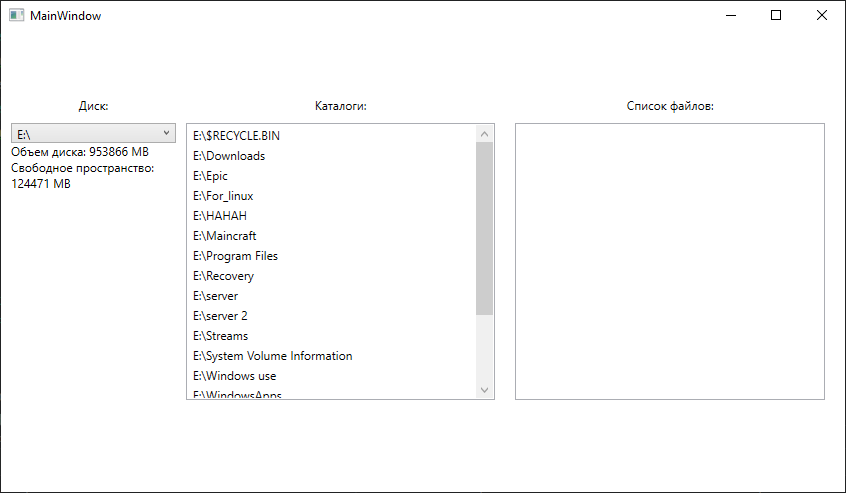
Код приложения:

using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.IO;  
using System.Linq;  
using System.Windows;  
using System.Windows.Controls;  
  
namespace WorkWithFiles;  
  
/// <summary>  
/// Interaction logic for MainWindow.xaml  
/// </summary>  
public partial class MainWindow : Window  
{  
 public MainWindow()  
 {  
 // Создаём и очищаем файл  
 var streamWriter = new StreamWriter("files.txt");  
 streamWriter.Write("");  
 streamWriter.Close();  
 InitializeComponent();  
 // Заполняем поле дисков  
 foreach (var drive in DriveInfo.GetDrives())  
 HardDrivesComboBox.Items.Add(drive.Name);  
 HardDrivesComboBox.SelectedIndex = 0;  
 }  
  
 private void HardDrives\_OnSelected(object sender, RoutedEventArgs e)  
 {  
 FoldersListBox.Items.Clear();  
 FilesListBox.Items.Clear();  
 var hardDrive = HardDrivesComboBox.SelectedItem.ToString()!;  
 foreach (var directory in Directory.GetDirectories(hardDrive))  
 FoldersListBox.Items.Add(directory);  
 // Получаем информацию о диске  
 var driveInfo = DriveInfo.GetDrives().Where(info => info.Name == hardDrive).ToList()[0];  
 HardDriveInformation.Text = @$"Объем диска: {driveInfo.TotalSize / (1024 \* 1024)} MB  
Свободное пространство: {driveInfo.TotalFreeSpace / (1024 \* 1024)} MB";  
 }  
  
  
 private void FoldersListBox\_OnSelectionChanged(object sender, SelectionChangedEventArgs e)  
 {  
 FilesListBox.Items.Clear();  
 FolderInformation.Text = "";  
 try  
 {  
 var path = FoldersListBox.SelectedItem.ToString()!;  
 var directoryInfo = new DirectoryInfo(path);  
 FolderInformation.Text = @$"Полное название каталога: {directoryInfo.Name}  
Время создания каталога: {directoryInfo.CreationTime}  
Корневой каталог: {directoryInfo.Parent?.Name ?? string.Empty}";  
 foreach (var file in Directory.GetFiles(path))  
 FilesListBox.Items.Add(file);  
 }  
 catch (NullReferenceException)  
 {  
 /\* При очистке поле Selection сбрасывается и нас выкидывает сюда, так как стоит проверка на null \*/  
 }  
 catch (UnauthorizedAccessException)  
 {  
 MessageBox.Show("Доступ запрещён к этой папке.");  
 }  
 }  
  
 /\* На открытие любого файла или при закрытии приложения  
 производим проверку все ли записанные файлы, открывались 10 секунд назад  
 \*/  
 private void fileOpened(string? path = null)  
 {  
 HashSet<(string, DateTime)> notDelete = new();  
 if (path != null)  
 notDelete.Add((path, DateTime.Now));  
 var streamReader = new StreamReader("files.txt");  
 while (!streamReader.EndOfStream)  
 {  
 var line = streamReader.ReadLine()!;  
 if (line.Trim() != "")  
 {  
 var runnedTime = DateTime.Parse(line.Split(',')[1]);  
 if (DateTime.Now - runnedTime < TimeSpan.FromSeconds(10))  
 notDelete.Add((line.Split(',')[0], runnedTime));  
 }  
 }  
  
 streamReader.Close();  
 var streamWriter = new StreamWriter("files.txt");  
 foreach (var pair in notDelete)  
 streamWriter.WriteLine($"{pair.Item1},{pair.Item2}");  
 streamWriter.Close();  
 }  
  
 protected override void OnClosed(EventArgs e)  
 {  
 fileOpened();  
 base.OnClosed(e);  
 }  
  
 private void FilesListBox\_OnSelectionChanged(object sender, SelectionChangedEventArgs e)  
 {  
 try  
 {  
 var path = FilesListBox.SelectedItem.ToString()!;  
 new Process  
 {  
 StartInfo = new ProcessStartInfo(path)  
 {  
 UseShellExecute = true  
 }  
 }.Start();  
 fileOpened(path);  
 FilesListBox.SelectedItem = null;  
 }  
 catch (NullReferenceException)  
 {  
 }  
 }  
}

Демонстрация работы приложения:



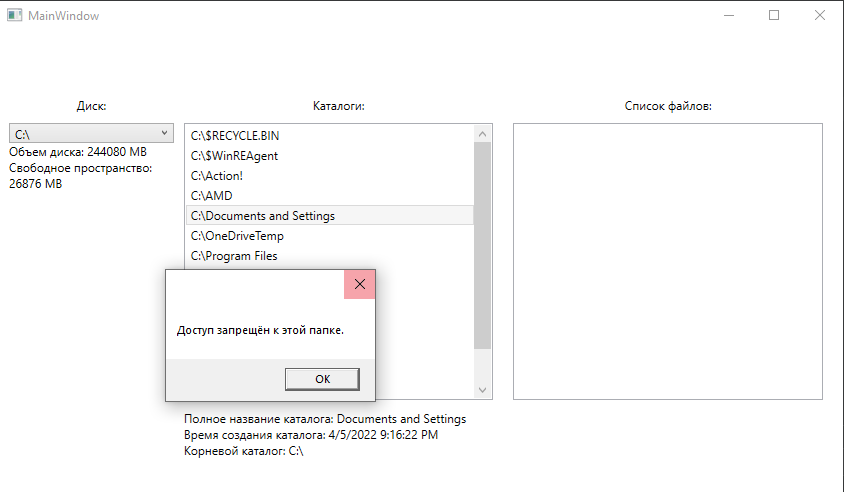
Выбран диск:



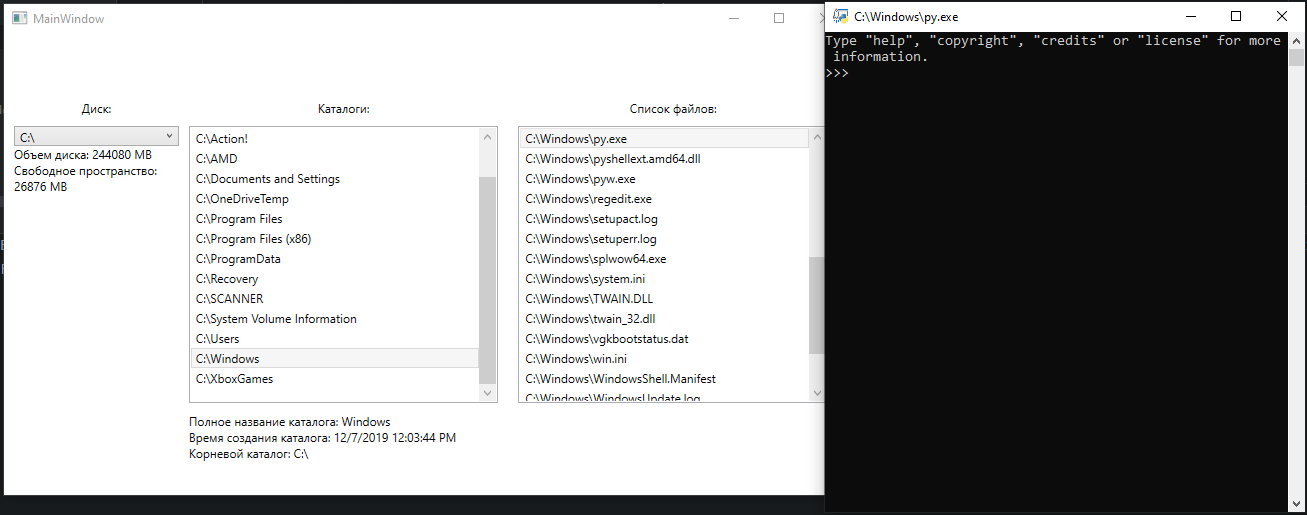
Выбрана папка:



Обработка ошибки с доступом:



Открытие файла(py.exe)



Запись открытия данного файла:

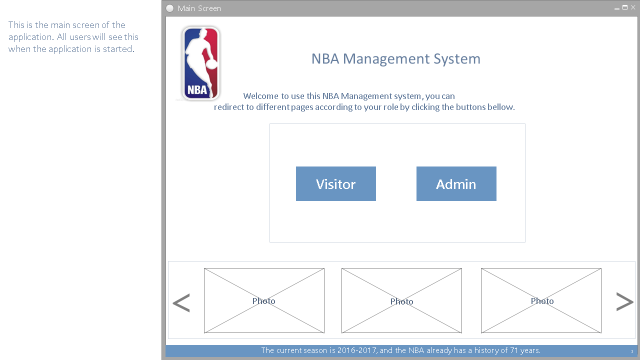
7.png

Итог работы:

Получил практические навыки при работе с пространством имен System.IO, а также создал приложение "Микропроводник".

## 1.1. Дополнительные задания на работу с файловой системой

Задание 1(фрагмент модуля задания по компетенции Программные решения для бизнеса(2018 год)).Вы являетесь разработчиком в команде, которая занимается проектированием и разработкой настольных приложений, взаимодействующих с БД. Технические сложности пока не дают возможности работать с БД в полном объеме. Поэтому список фотографий для загрузки на форму необходимо брать из конкретной папки. Содержимое папки будет изменяться, поэтому не делайте «жесткой » привязки по полному пути к фото. Есть вероятность, что данная папка со временем будет иметь структуру подпапок. В этом случае поиск файлов необходимо делать во всех подпапках и формировать список загружаемых фото по результатам этого поиска. Вы разрабатываете модуль, в котором загрузка фото из папок на форме оформлена в виде слайдера- по 3 фото за 1 раз. Смена фото происходит по нажатию соответствующих кнопок. Макет формы представлен на рисунке 1. Алгоритм изменения изображений должен работать таким образом, чтобы учесть случай, когда количество фото не кратно 3. В этом случае,в последней группе из 2-х фото добавится самое первое фото в папке(сдвиг произойдет на 1 позицию) и т.д.

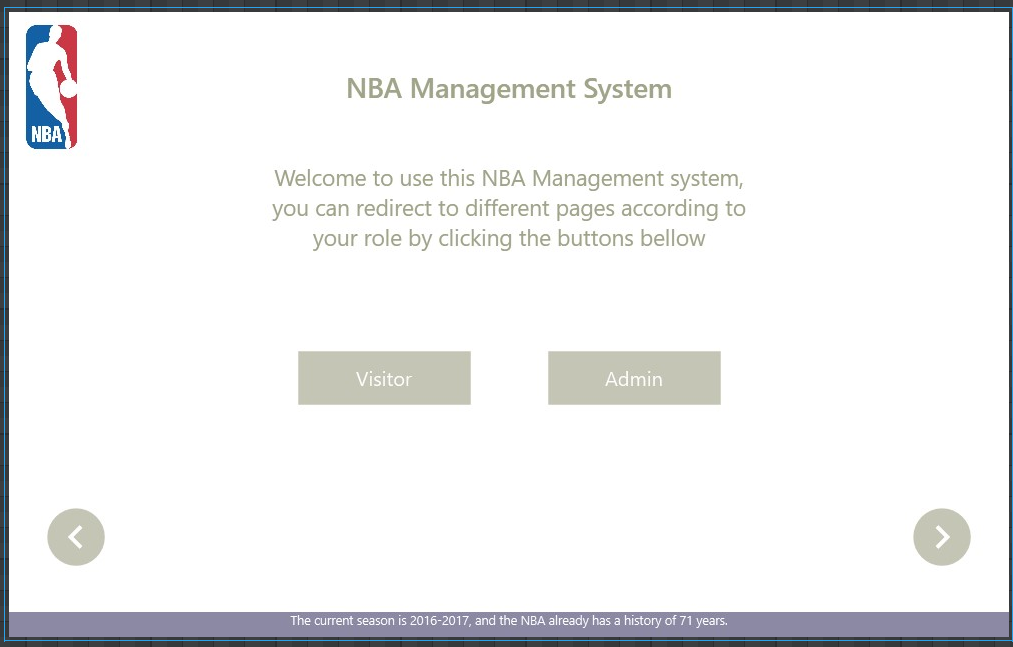


Для выполнения задания используйте папки с ресурсами – папку logo и папку Pictures.

Код:

Разметка:

<Window x:Class="NbaApp.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"  
 mc:Ignorable="d"  
 Title="MainWindow" Height="450" Width="800"  
 MinHeight="500" MinWidth="800">  
 <Grid>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition />  
 <ColumnDefinition />  
 <ColumnDefinition />  
 <ColumnDefinition />  
 </Grid.ColumnDefinitions>  
 <Grid.RowDefinitions>  
 <RowDefinition />  
 <RowDefinition />  
 <RowDefinition />  
 <RowDefinition />  
 <RowDefinition Height="20" />  
 </Grid.RowDefinitions>  
  
 <Image HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10" Source="res/logo/logo.jpg" />  
 <TextBlock FontSize="22" FontWeight="Medium" Grid.ColumnSpan="2" HorizontalAlignment="Center"  
 VerticalAlignment="Center" Grid.Column="1">  
 NBA Management System  
 </TextBlock>  
 <TextBlock Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2" FontSize="18" TextWrapping="Wrap"  
 VerticalAlignment="Top" HorizontalAlignment="Center" TextAlignment="Center">  
 Welcome to use this NBA Management system, you can redirect to different pages according to your role by clicking the buttons bellow  
 </TextBlock>  
 <Button Grid.Column="1" VerticalAlignment="Top" Margin="30" Foreground="White" Grid.Row="2">  
 <TextBlock Margin="10">  
 <Run Foreground="White">  
 Visitor  
 </Run>  
 </TextBlock>  
 </Button>  
 <Button Grid.Column="2" Margin="30" VerticalAlignment="Top" Foreground="White" Grid.Row="2">  
 <TextBlock Margin="10">  
 <Run Foreground="White">  
 Admin  
 </Run>  
 </TextBlock>  
 </Button>  
  
 <Grid Margin="10" Grid.Row="3" Grid.ColumnSpan="4">  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="7\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
 <Grid Grid.Column="1">  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition />  
 <ColumnDefinition />  
 <ColumnDefinition />  
 </Grid.ColumnDefinitions>  
 <Image Margin="10, 0" Source="{Binding Img1}" />  
 <Image Margin="10, 0" Grid.Column="1" Source="{Binding Img2}" />  
 <Image Margin="10, 0" Grid.Column="2" Source="{Binding Img3}" />  
  
 </Grid>  
  
 <Button Click="Left" Width="48" Height="48">  
 <Button.Resources>  
 <Style TargetType="{x:Type Border}">  
 <Setter Property="CornerRadius" Value="123" />  
 </Style>  
 </Button.Resources>  
 <Image Source="res/ui/left.png" Margin="3" />  
 </Button>  
 <Button Click="Right" Width="48" Height="48" Grid.Column="3">  
 <Button.Resources>  
 <Style TargetType="{x:Type Border}">  
 <Setter Property="CornerRadius" Value="123" />  
 </Style>  
 </Button.Resources>  
 <Image Source="res/ui/right.png" Margin="3" />  
 </Button>  
 </Grid>  
 <TextBlock Background="#8b89a4" Foreground="White" HorizontalAlignment="Stretch" TextAlignment="Center"  
 Grid.Row="4" Grid.ColumnSpan="4" FontSize="10">  
 The current season is 2016-2017, and the NBA already has a history of 71 years.  
 </TextBlock>  
 </Grid>  
</Window>



MainWindow.cs

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 NbaApp;  
  
/// <summary>  
/// Interaction logic for MainWindow.xaml  
/// </summary>  
public partial class MainWindow : Window  
{  
 private Images \_images = new();  
  
 public MainWindow()  
 {  
 DataContext = \_images;  
 InitializeComponent();  
 }  
  
 private void Right(object sender, RoutedEventArgs e)  
 {  
 \_images.Right();  
 }  
  
 private void Left(object sender, RoutedEventArgs e)  
 {  
 \_images.Left();  
 }  
}

Images.cs

using System;  
using System.Collections.Generic;  
using System.IO;  
using System.Linq;  
  
namespace NbaApp;  
  
public class Images : ViewModel  
{  
 // Путь до картинок  
 private const string PATH = @"C:\Users\user\Desktop\NbaApp\NbaApp\res\";  
 private string[] \_showedImages;  
  
 private int \_index = 0;  
  
 // Получаем все картинки из подпапок, а также их фильтруем по \*.jpg  
 private List<string> \_images = Directory.EnumerateFiles(PATH, "\*.jpg", SearchOption.AllDirectories).ToList();  
  
 // Параметры для binding  
 public string Img1  
 {  
 get => \_showedImages[0];  
 set => \_showedImages[0] = value;  
 }  
  
 public string Img2  
 {  
 get => \_showedImages[1];  
 set => \_showedImages[1] = value;  
 }  
  
 public string Img3  
 {  
 get => \_showedImages[2];  
 set => \_showedImages[2] = value;  
 }  
  
 public Images()  
 {  
 ShowImages();  
 }  
  
 // Функция отображения картинок по индексу  
 private void ShowImages()  
 {  
 if (\_images.Count == 0)  
 return;  
  
 // Если у нас одна картинка просто зацикливаем одну картинку и наплевать на индексы  
 if (\_images.Count == 1)  
 \_showedImages = new[] {\_images[0], \_images[0], \_images[0]};  
 else if (\_index + 3 >= \_images.Count)  
 {  
 var list\_ = \_images.GetRange(\_index, \_images.Count - \_index);  
 list\_.AddRange(\_images.GetRange(0, 3 - (\_images.Count - \_index)));  
 \_showedImages = list\_.ToArray();  
 }  
 else  
 {  
 \_showedImages = \_images.GetRange(\_index, 3).ToArray();  
 }  
  
 // Уведомление WPF о измененения в этих Properties  
 NotifyPropertyChanged("Img1");  
 NotifyPropertyChanged("Img2");  
 NotifyPropertyChanged("Img3");  
 }  
  
 // Высчитываем новый индекс при листании вправо и меняем картинки  
 public void Right()  
 {  
 if (\_images.Count == 0)  
 return;  
  
 // Если индекс превышает допустимое значение, получаем новый цикличный индекс, иначе просто листаем на 3  
 if (\_index + 3 >= \_images.Count)  
 \_index = 3 - (\_images.Count - \_index);  
 else  
 \_index += 3;  
 /\* Cуществует ошибка если индекс меньше 3, то мы не получаем новый корректно(вычитание из тройки),  
 это исправляет ошибку\*/  
 if (\_images.Count < 3 && \_index == \_images.Count)  
 \_index = 0;  
 ShowImages();  
 }  
  
 // Высчитываем новый индекс при листании влево и меняем картинки  
 public void Left()  
 {  
 if (\_images.Count == 0)  
 return;  
 if (\_index - 3 <= 0)  
 \_index = Math.Abs(\_images.Count + (\_index - 3));  
 else  
 \_index -= 3;  
 ShowImages();  
 }  
}

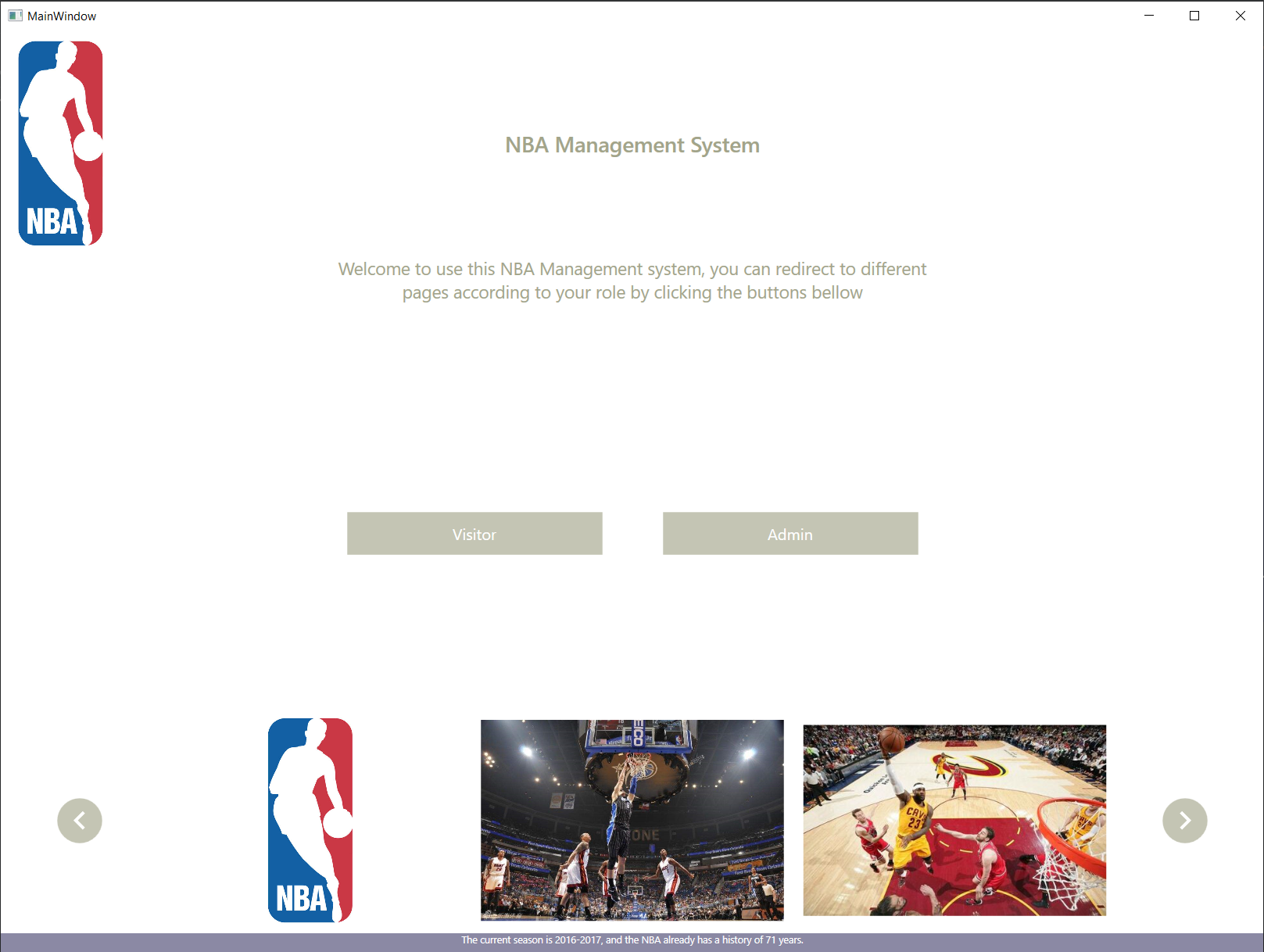
ViewModel.cs

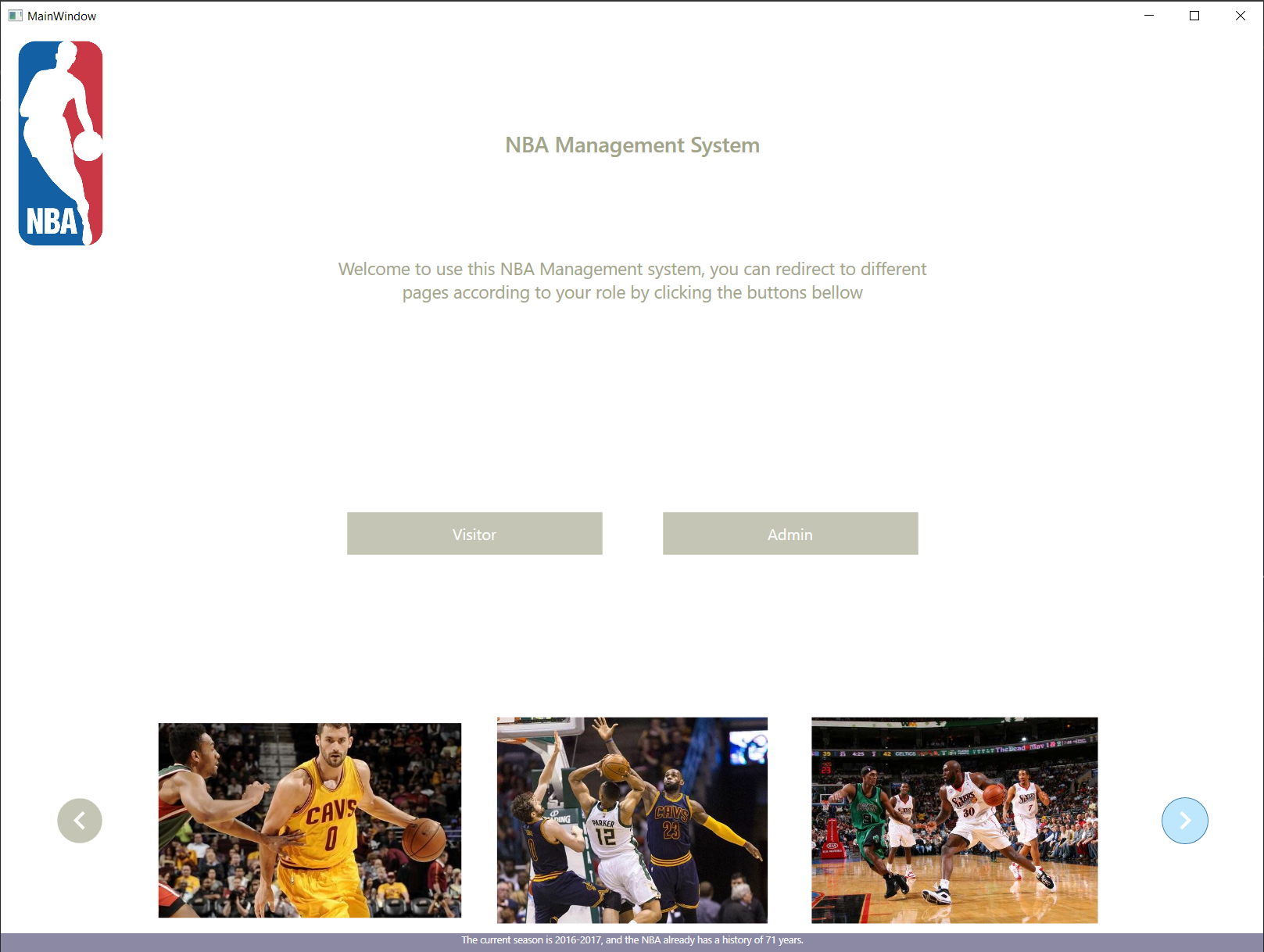
using System.ComponentModel;  
  
namespace NbaApp;  
  
// Крутой класс для уведомления WPF, что Property изменилось  
public abstract class ViewModel : INotifyPropertyChanged  
{  
 public event PropertyChangedEventHandler? PropertyChanged;  
  
 protected void NotifyPropertyChanged(string propertyName)  
 {  
 if (PropertyChanged != null)  
 PropertyChanged(this, new PropertyChangedEventArgs(propertyName));  
 }  
}

Демонстрация работы приложения:

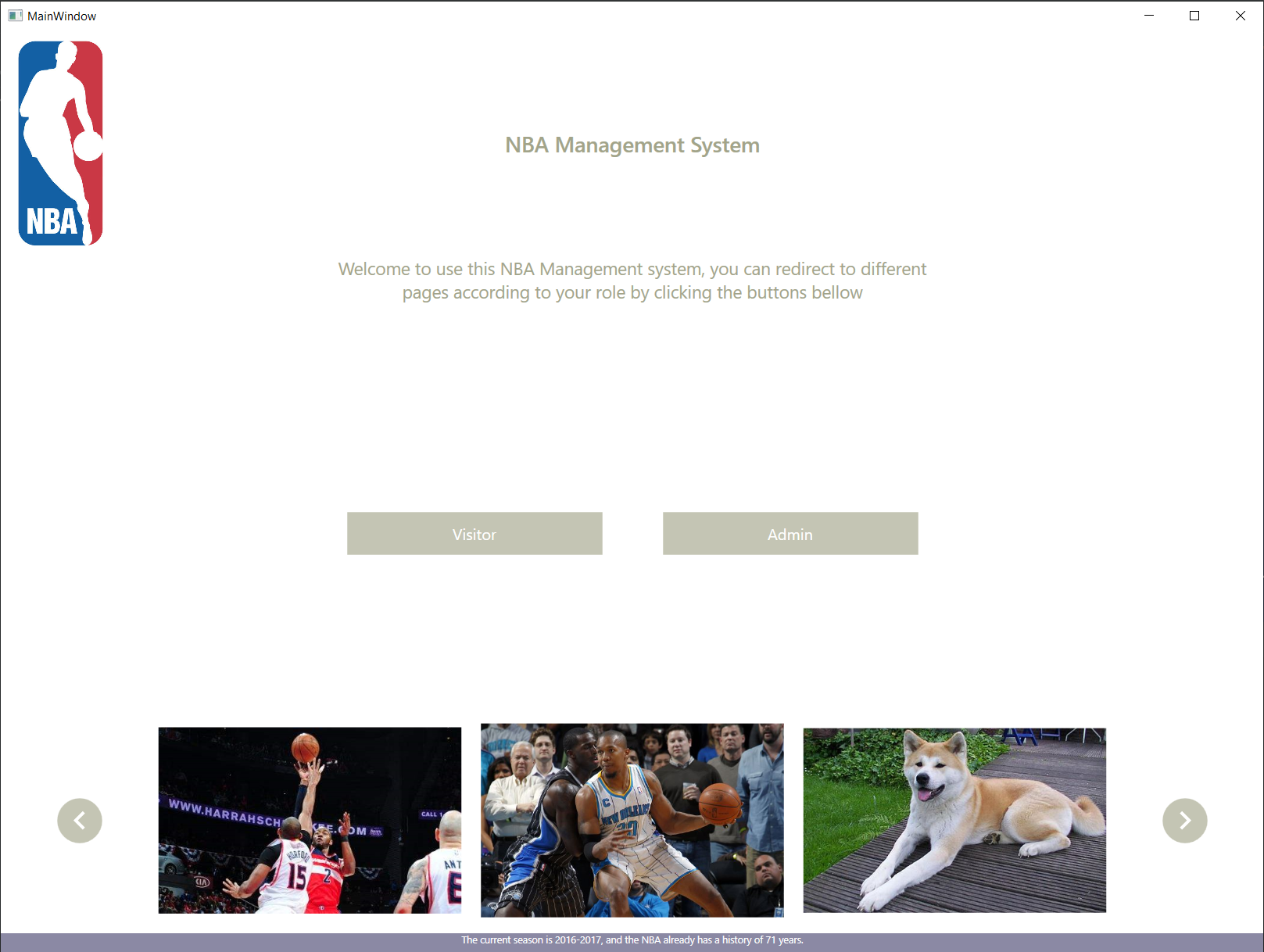
Картинки используемые в приложении:



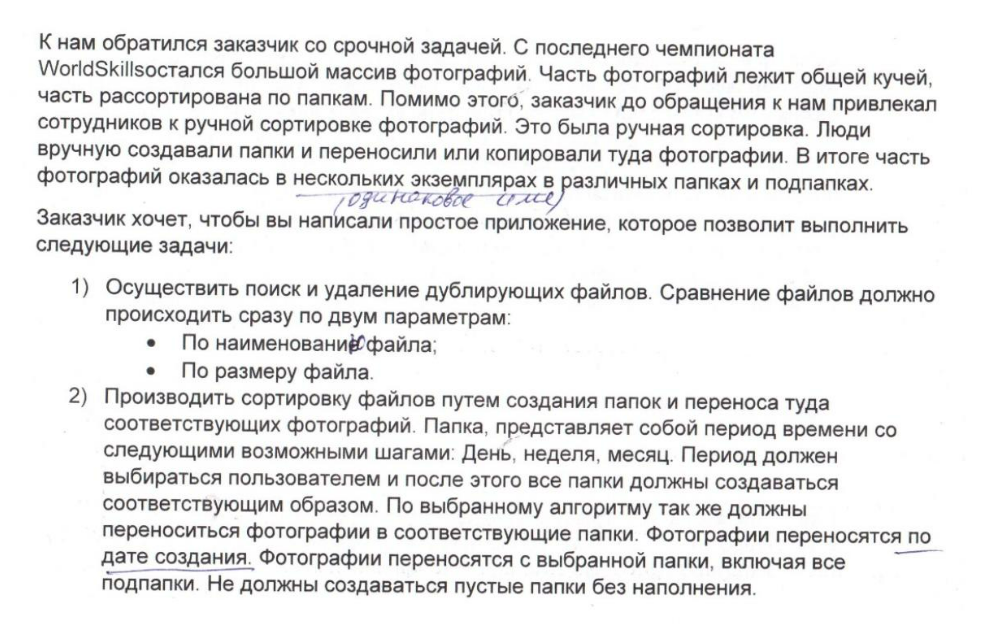








Задание 2.(фрагмент модуля задания по компетенции Программные решения для бизнеса(2017 год), overdrive на 30 минут).

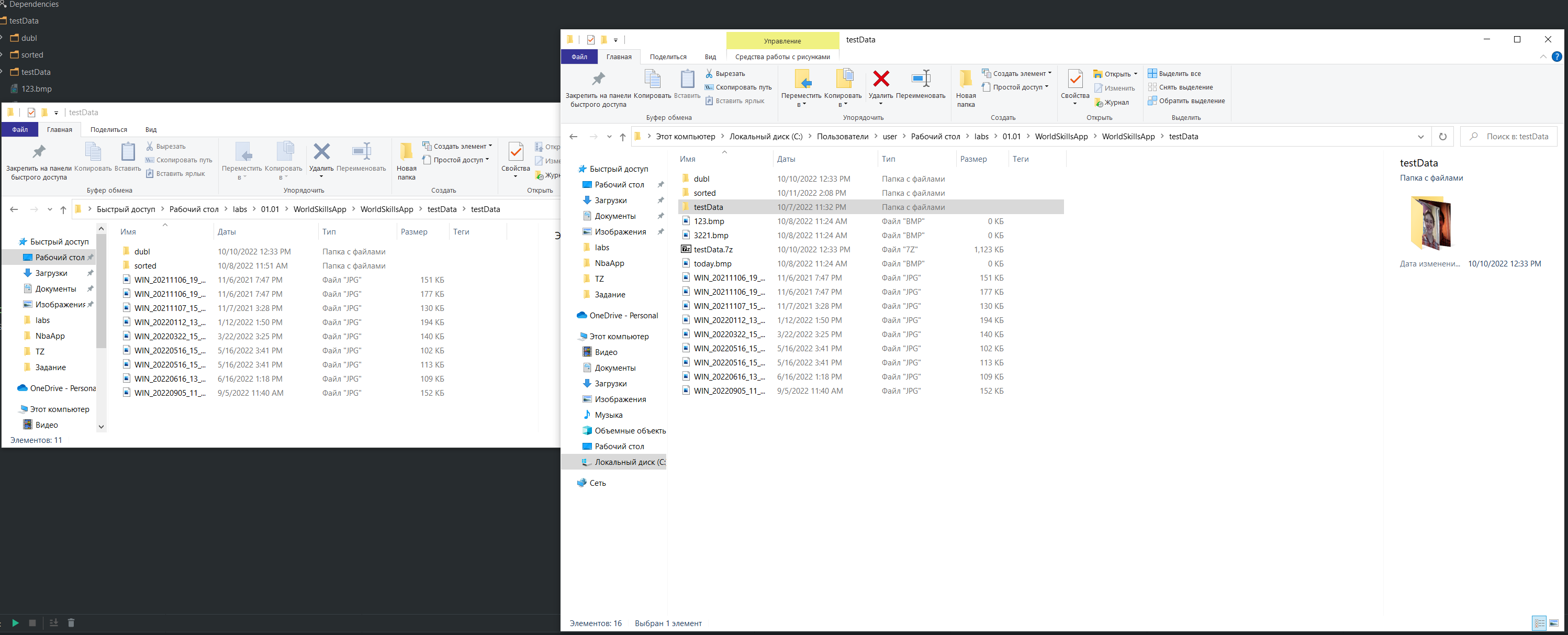


Код:

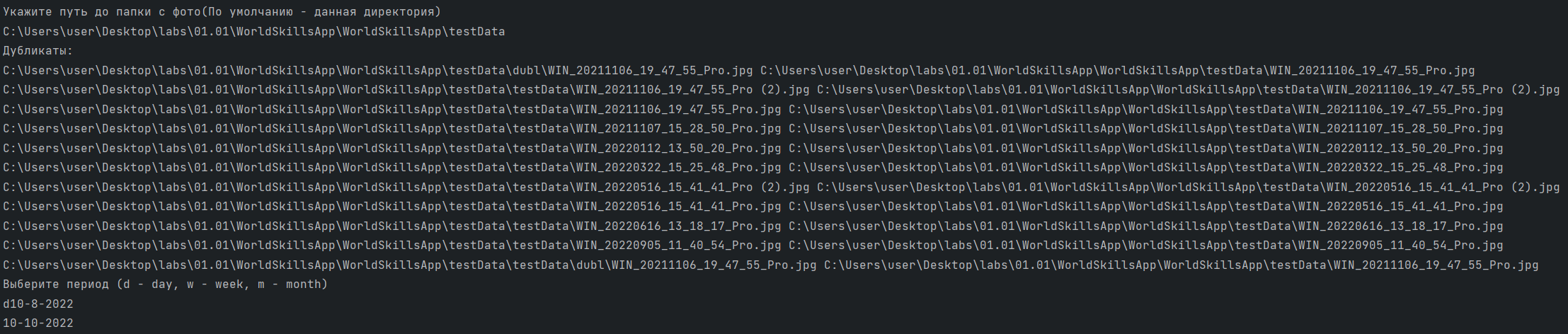
using System.Globalization;  
  
Console.WriteLine("Укажите путь до папки с фото(По умолчанию - данная директория)");  
var path = Console.ReadLine() ?? ".";  
path = path.Trim() == "" ? "." : path;  
Dictionary<(string, string), string> filesMap = new Dictionary<(string, string), string>();  
Console.WriteLine("Дубликаты: ");  
foreach (var filePath in Directory.EnumerateFiles(path, "\*.\*", SearchOption.AllDirectories))  
{  
 var fileInfo = new FileInfo(filePath);  
 if (filesMap.ContainsKey((fileInfo.Name, fileInfo.CreationTime.ToString(DateTimeFormatInfo.CurrentInfo))))  
 Console.WriteLine(  
 $"{fileInfo.FullName} {filesMap[(fileInfo.Name, fileInfo.CreationTime.ToString(CultureInfo.CurrentCulture))]}");  
 else  
 filesMap.Add((fileInfo.Name, fileInfo.CreationTime.ToString(CultureInfo.CurrentCulture)),  
 fileInfo.FullName);  
}  
  
idiot:  
Console.WriteLine("Выберите период (d - day, w - week, m - month)");  
var period = Console.ReadKey();  
Dictionary<string, List<string>> filesSorted = new Dictionary<string, List<string>>();  
  
foreach (var pair in filesMap.Keys)  
{  
 var createdTime = DateTime.Parse(pair.Item2);  
 try  
 {  
 string key = period.Key switch  
 {  
 ConsoleKey.D => $"{createdTime.DayOfYear} {createdTime.Year}",  
 ConsoleKey.W => $"{createdTime.DayOfYear / 7} {createdTime.Year}",  
 ConsoleKey.M => $"{createdTime.Month} {createdTime.Year}",  
 \_ => throw new ArgumentOutOfRangeException()  
 };  
 if (filesSorted.ContainsKey(key))  
 {  
 filesSorted[key].Add(filesMap[pair]);  
 }  
 else  
 filesSorted[key] = new List<string>() {filesMap[pair]};  
 }  
 catch (ArgumentOutOfRangeException e)  
 {  
 goto idiot;  
 }  
}  
  
Directory.SetCurrentDirectory(path);  
Directory.CreateDirectory("sorted");  
Directory.SetCurrentDirectory("sorted");  
foreach (var dateString in filesSorted.Keys)  
{  
 var ints = dateString.Split().Select(s => int.Parse(s)).ToArray();  
 (int what, int year) = (ints[0], ints[1]);  
 string folderName = period.Key switch  
 {  
 ConsoleKey.D => $"{new DateTime(year, 1, 1).AddDays(what - 1).ToShortDateString().Replace("/", "-")}",  
 ConsoleKey.W => new DateTime(year, 1, 1).AddDays(what \* 7).ToShortDateString().Replace("/", "-") + " - " +  
 new DateTime(year, 1, 1).AddDays(what \* 7 + 7).ToShortDateString().Replace("/", "-"),  
 ConsoleKey.M =>  
 $"{new DateTime(year, what, 1).ToShortDateString().Replace("/", "-")} - {new DateTime(year, what, 1).AddMonths(1).AddDays(-1).ToShortDateString().Replace("/", "-")}",  
 };  
 Console.WriteLine(folderName);  
 Directory.CreateDirectory(folderName);  
 foreach (var filePath in filesSorted[dateString])  
 {  
 int fileRepeat = 0;  
 again:  
 try  
 {  
 var newFileName = new FileInfo(filePath).Name.Split('.')[0] + (fileRepeat == 0 ? "" : fileRepeat.ToString()) + "." +  
 new FileInfo(filePath).Name.Split('.')[1];  
 File.Copy(filePath, folderName + "\\" + newFileName);  
 }  
 catch (System.IO.IOException e)  
 {  
 fileRepeat++;  
 goto again;  
 }  
 }  
}

Демонстрация работы:

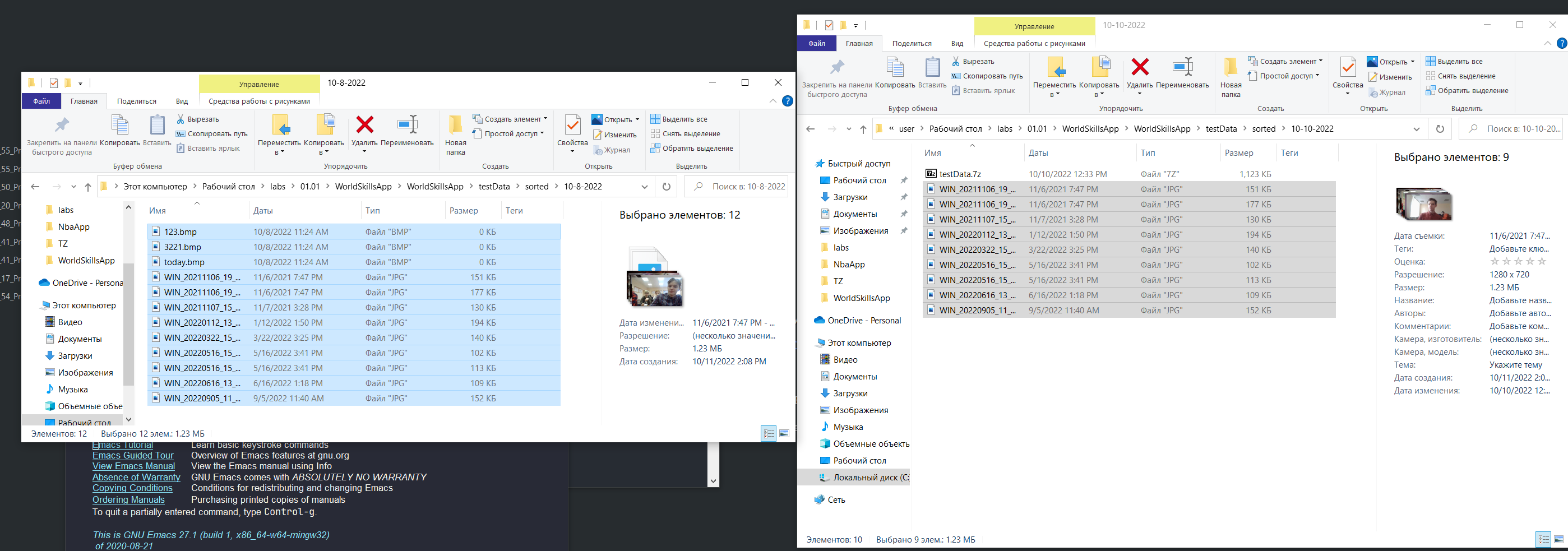
Содержимое папок:



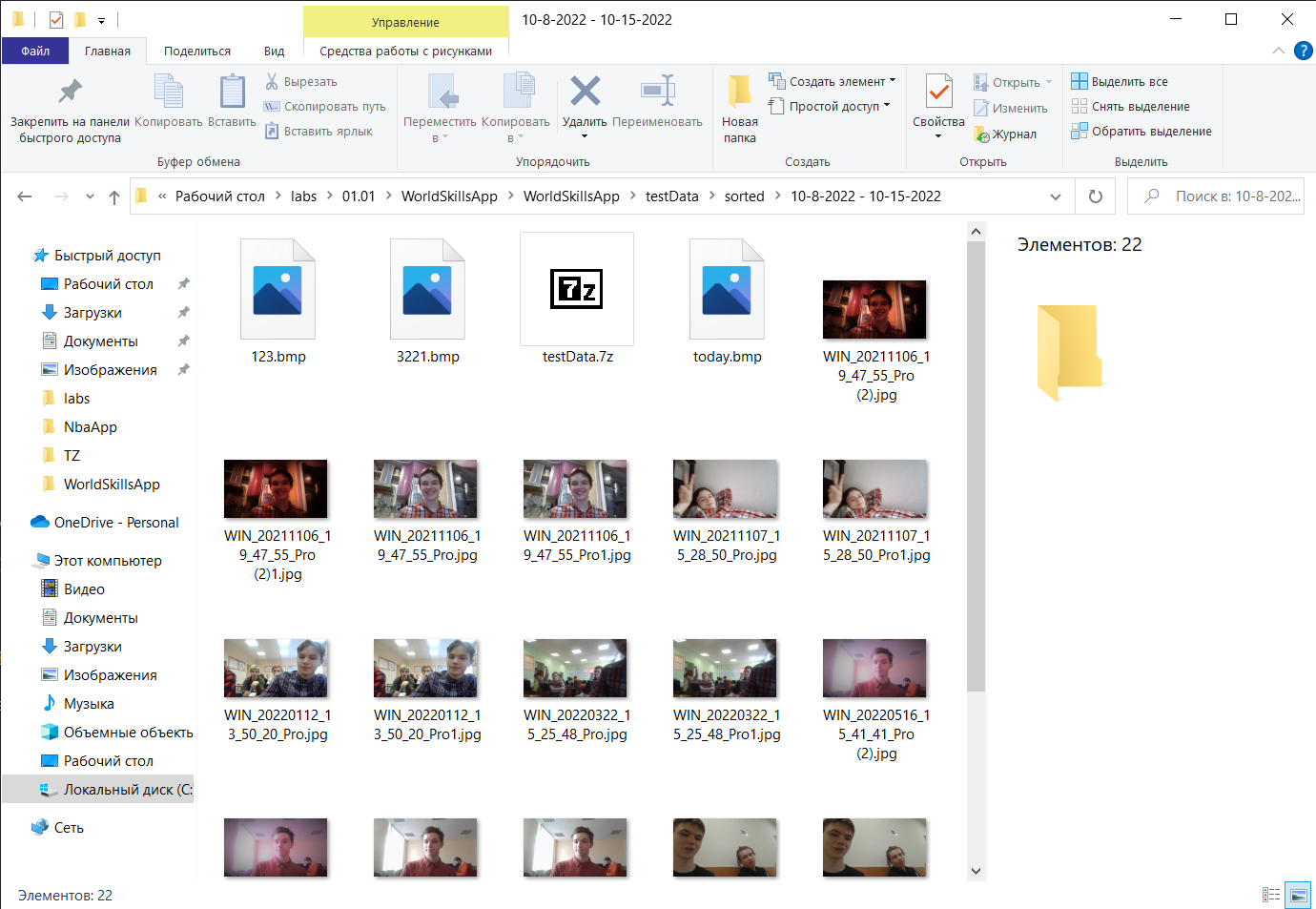
Результат работы:



По дням:

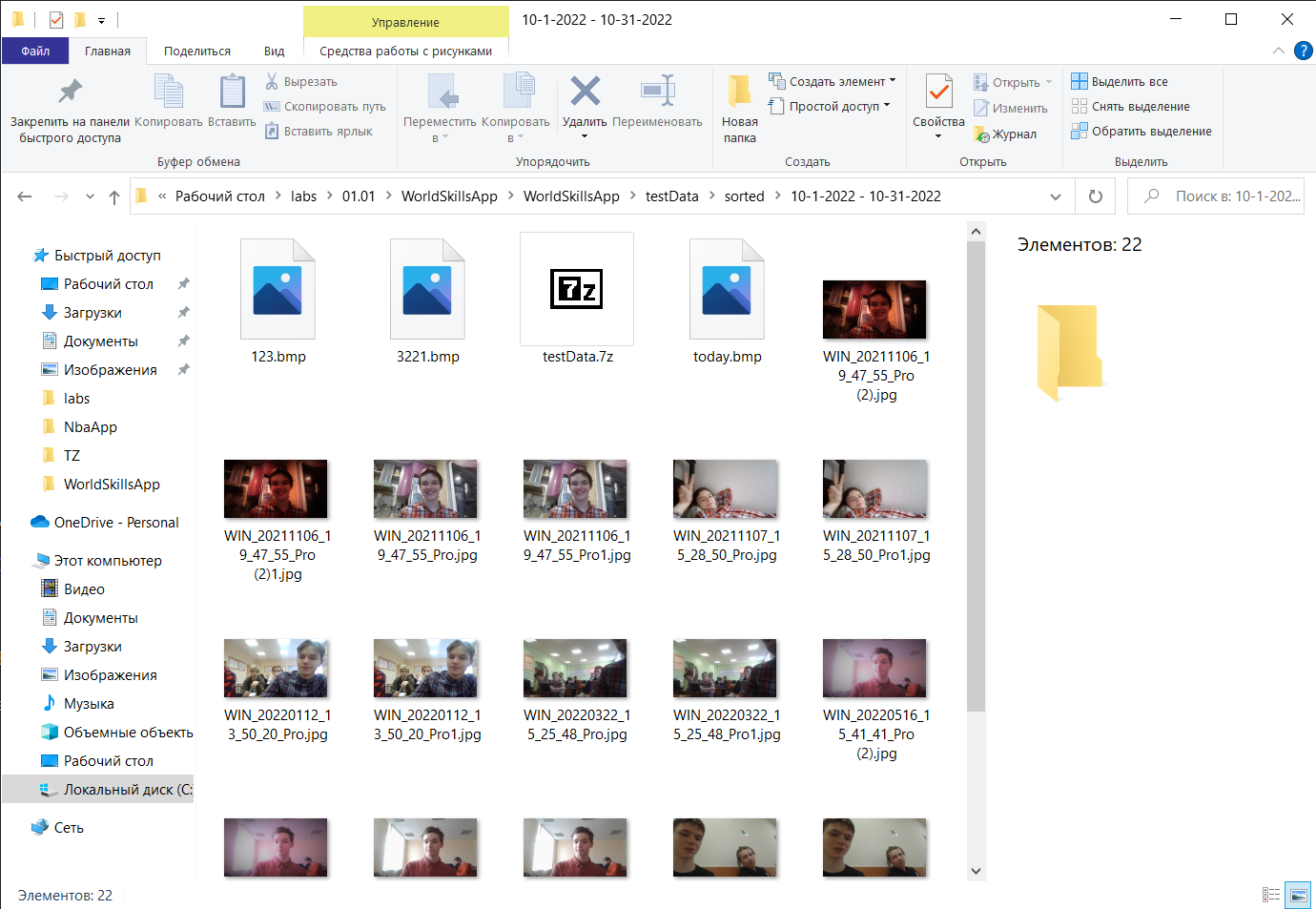


По неделям:



Все картинки, так как они были созданы в эту неделю - главное это название папки(10-8-2022 - 10-15-2022)

По месяцам:



Created: 2022-10-11 Tue 14:29

[Validate](https://validator.w3.org/check?uri=referer)