

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Технологии программирования»
на тему «Разработка объектно-ориентированной программы с
пользовательским интерфейсом»

Выполнил студент группы 19BB1:
Артемов К.А.

Принял:

к. т. н., доцент
Мартышкин А.И.

Отлично
Мартышкин 25.12.2021

Пенза, 2021

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М.А. Митрохин

«__» __ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Технологии программирования

Студенту Артемову Кириллу Артемовичу Группа 19ВВ1
Тема проекта Разработка объектно-ориентированной программы
с пользовательским интерфейсом

Исходные данные (технические требования) на проектирование

Разработать программу "Проводник"

Программа должна выполнять след функции:

- 1) Отображение файлов и папок
- 2) Сохранение в закладки
- 3) Создание вкладок и управление ими.
- 4) Drag and drop (способ оперирования элементами интерфейса)
- 5) возможность выделения нескольких элементов
- 6) Различное представление элементов. (табличное, плиточное)
- 7) Группировка элементов
- 8) Поиск в адресной строке.

Обязательные требования к программе:

- 1) Операционная система - ms Windows 10
- 2) Система программирования - ms visual studio 2019
- 3) Язык программирования C#

Объем работы по курсу

1. Расчетная часть

- 1) Структура данных используемых объектов
- 2) Описание архитектуры разработанного приложения и способы организации пользовательского интерфейса.
- 3) Описание программы
- 4) Описание результатов работы программы и их анализ
- 5) Руководство по пользованию.

2. Графическая часть

Диаграммы и т.д.

3. Экспериментальная часть

отладка программы

Срок выполнения проекта по разделам

- 1 В соответствии с графиком выполнения курсовой работы
- 2
- 3
- 4
- 5
- 6
- 7
- 8

Дата выдачи задания “ ”

Дата защиты проекта “ ”

Руководитель Мартышкин А.И.

Задание получил “ 25 ” сентября 2021 г.

Студент Артемов Кирилл Артемович

Содержание

Введение	6
1. Постановка задачи.....	8
2. Выбор решения.....	9
2.1. Определение структуры программы.....	9
2.2. Описание основного цикла работы.....	9
3. Описание разработки программы.....	11
4. Отладка и тестирование.	13
5. Описание программы.	14
6. Руководство пользователя.	16
Заключение	27
Список используемых источников.	28
Приложение А Листинг модуля Explorer.WPF.UI.....	29
Приложение А.1 Файл « App.xaml.cs».....	29
Приложение А.2 Файл « App.xaml».....	30
Приложение А.3 Файл « MainWindow.xaml.cs»	32
Приложение А.4 Файл « MainWindow.xaml»	32
Приложение А.5 Файл « CalcMaxWidthTabItemConverter.cs»	33
Приложение А.6 Файл « CloneParamsConverter.cs»	34
Приложение А.7 Файл « FileEntityToImageConverter.cs».....	35
Приложение А.8 Файл « IconPathToImageConverter.cs»	36
Приложение А.9 Файл « NullToVisibilityConverter.cs»	37
Приложение А.10 Файл « CommandReference.cs».....	38
Приложение А.11 Файл « WpfSynchronizationHelper.cs».....	41
Приложение А.12 Файл « XButtonsGestureHelper.cs»	41
Приложение Б Листинг модуля EXWindow	43
Приложение Б.1 Файл « Windows.cs».....	43
Приложение Б.2 Файл « WindowResizeFixerBehavior.cs»	43
Приложение Б.3 Файл « DelegateCommand.cs»	45
Приложение Б.4 Файл « GoogleChromeWindow.cs».....	46
Приложение Б.5 Файл « WindowBorderThicknessConverter.cs»	49
Приложение Б.6 Файл « WindowTitleHeightConverter.cs»	50
Приложение В Листинг модуля ExplorER.....	52
Приложение В.1 Файл « BookmarkItem.cs».....	52
Приложение В.2 Файл « BookmarksManager.cs»	52

Приложение В.3 Файл «IBookmarksManager.cs».....	56
Приложение В.4 Файл «MenuItemViewModel.cs»	56
Приложение В.5 Файл «DelegateCommand.cs».....	57
Приложение В.6 Файл «ISynchronizationHelper.cs»	58
Приложение В.7 Файл «NaturalSortComparer.cs»	58
Приложение В.8 Файл «DirectoryHistory.cs»	60
Приложение В.9 Файл «DirectoryNode.cs»	62
Приложение В.10 Файл «IDirectoryHistory.cs»	62
Приложение В.11 Файл «ExtensionToImageFileConverter.cs»	63
Приложение В.12 Файл «IconName.cs»	64
Приложение В.13 Файл «IconsManager.cs»	64
Приложение В.14 Файл « BaseViewModel.cs».....	66
Приложение В.15 Файл « ChromerTabItemViewModel.cs».....	66
Приложение В.16 Файл « FileEntityViewModel.cs».....	67
Приложение В.17 Файл « DirectoryViewModel.cs»	67
Приложение В.18 Файл « FileViewModel.cs»	68
Приложение В.19 Файл « LogicalDriveViewModel.cs».....	68
Приложение В.20 Файл « IFilesPresenter.cs»	69
Приложение В.21 Файл « OpenDirectoryEventArgs.cs»	69
Приложение В.22 Файл « BaseFilesPresenter.cs»	70
Приложение В.23 Файл « GridFilesPresenterViewModel.cs»	74
Приложение В.24 Файл « TileFilesPresenterViewModel.cs»	75
Приложение В.25 Файл « DirectoryTabItemViewModel.cs».....	75
Приложение В.26 Файл « TabsViewModel.cs».....	80
Приложение В.27 Файл « ExplorerEr.cs»	83

Введение

В современном мире невозможно представить жизнь человека без взаимодействия с компьютером. Приложение - компьютерная программа, комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления. Приложение происходит в процессе разработки его программистом. Разработка приложения - очень тонкий процесс, в нём есть свои нюансы, не зная которых сложно утверждать, что готовый продукт будет соответствовать должному качеству. С развитием информационных технологий человеческая жизнедеятельность становится легче.

Объектно-ориентированное программирование является в настоящее время наиболее популярной технологией программирования. Объектно-ориентированный подход рассматривает все ресурсы, такие как данные, модули, системы в качестве объектов. Каждый объект содержит некоторую структуру данных, обрамленную некоторыми наборами процедур (методов), предназначенных для манипулирования этими данными. Объектно-ориентированное программирование по своей сути – это создание приложений из объектов, подобно тому, как из блоков и различных деталей строятся дома. Одни объекты приходится полностью создавать самостоятельно, тогда как другие можно позаимствовать в готовом виде из разнообразных библиотек. Важное место в технологии объектно-ориентированного программирования занимает событие. В качестве событий могут рассматриваться щелчок кнопкой мыши на объекте, нажатие определенной клавиши, открытие документа и т.д. В качестве реакции на события вызывается определенная процедура, которая может изменять свойства объекта, вызывать его методы и т.д. В системах объектно-ориентированного программирования обычно используется графический интерфейс, который позволяет визуализировать процесс программирования.

Появляется возможность создавать объекты, задавать им свойства и поведение с помощью мыши.

1. Постановка задачи.

Требуется разработать приложение «Проводник». Приложение должно иметь следующий функционал: отображение папок и файлов, сохранение в закладки, создание вкладок и управление ими; drag and drop (способ оперирования элементами интерфейса), возможность выделения нескольких элементов, различное представление элементов (табличное и плиточное), группировка элементов, поиск в адресной строке.

Общие требования к программе:

1. Операционная система - MS Windows;
2. Система программирования - MS Visual Studio 2019;
3. Язык программирования - C#.

2. Выбор решения.

2.1. Определение структуры программы

Сама программа должна состоять из совмещенного окна, представляющее из себя меню навигации, работа с вкладками и основное рабочее пространство;

Это окно должно выполнять соответствующие функции: осуществлять взаимодействие с файлами и директориями, работу с закладками, открывать вкладки, представляющие из себя такое же окно.

2.2. Описание основного цикла работы

Основной цикл программы должен заключаться в следующем:

- отобразить пользователю основное окно проводника, предоставляющее пользователю все возможности программы.
- загрузить список всех файлов и директорий.
- отобразить логические диски, и директории быстрого доступа. в пользовательском интерфейсе.
- ожидать действий пользователя.
- если пользователь захотел просмотреть выбранную директорию или файл, то необходимо отобразить в интерфейсе все внутренние директории и файлы для Директории или открыть файл программой по умолчанию.
- если пользователю необходимо создать вкладку программа должна будет создать новое окно, выполняющее идентичный функционал.

Управление программой должно осуществляться по событийному варианту, так как автономно (независимо от действий пользователя)

программа не должна ничего обрабатывать. Это значит, что все действия в программе должны осуществляться по определённым событиям, которые вызывает пользователь.

Например, окно новой вкладки должно открываться только после нажатия соответствующей клавиши, точно так же должна соблюдаться логика при других событиях.

Диаграмма вариантов использования программы представлена на рисунке ниже

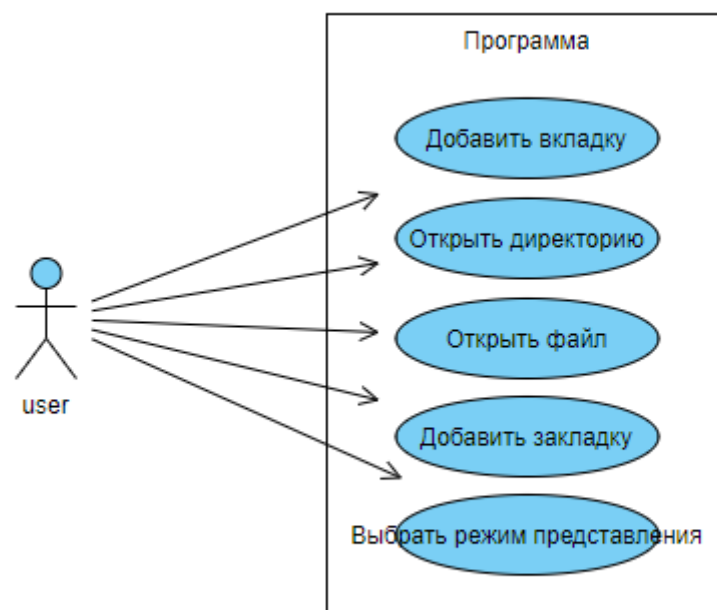


Рисунок 1 – UML – диаграмма вариантов использования

3. Описание разработки программы.

Для написания данной программы был использован язык программирования C#. Этот язык является распространённым языком программирования. Он сочетает в себе высокую эффективность низкоуровневых языков с возможностями высокоуровневых языков (быстрое создание больших сложных проектов, сводя к минимуму отладку и тестирование). В языке C# предусмотрено много операций, непосредственно исполняемых большинством микропроцессоров; в то же время он дает программисту возможность выражать свои алгоритмы и данные наиболее подходящими средствами.

В качестве среды программирования был выбран программный продукт Visual Studio 2019.

Для работы с файлами и директориями, использовалось пространство имен: System/IO классы File, FileInfo, Directory, DirectoryInfo, Path и тп

Для функции drag and drop использовалась готовая библиотека, скаченная в ms Visual studio в NuGet.

В свою очередь графическая часть редактора будет использовать ресурсы Windows WPF (открытая универсальная платформа разработки, которая поддерживается корпорацией Майкрософт и сообществом .NET Она является кроссплатформенной, поддерживает Windows, Mac OS и Linux и может использоваться на устройствах, в облаке, во внедренных системах и в сценариях IoT)

При проектировании применялся нисходящий метод разработки, так как основной смысл программы состоит в отображении файлов и директорий и работы с ними, а уже только потом графическое оформление и структурирование.

С начала реализовывались базовые функции, такие как, вывод названия директорий и файлов, возможность создания дополнительных окон. После были разработаны, основываясь на базисе, функции, взаимодействующие с

графическим интерфейсом, такие как, заполнения блоков файлов иконками в зависимости от их расширения, отображение пути до выбранного файла/директории, кнопки и стилизация окна.

4. Отладка и тестирование.

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2019. Программа предоставляет все средства, необходимые при разработке и отладке модулей и программ. В ходе отладки применялись такие возможности, как точка останова, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти и дополнительное окно, куда выводил необходимое.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было отловлено не мало багов, не все, но большинство было исправлено.

5. Описание программы.

Для реализации графической составляющей была задействована стандартная библиотека Windows WPF. Используя функции из этой библиотеки, осуществляется вывод данных на экран, получение событий клавиатуры и мыши, перемещения курсора. С помощью всех перечисленных возможностей этой библиотеки были реализованы многие функции.



Рисунок 2 – UML – диаграмма классов

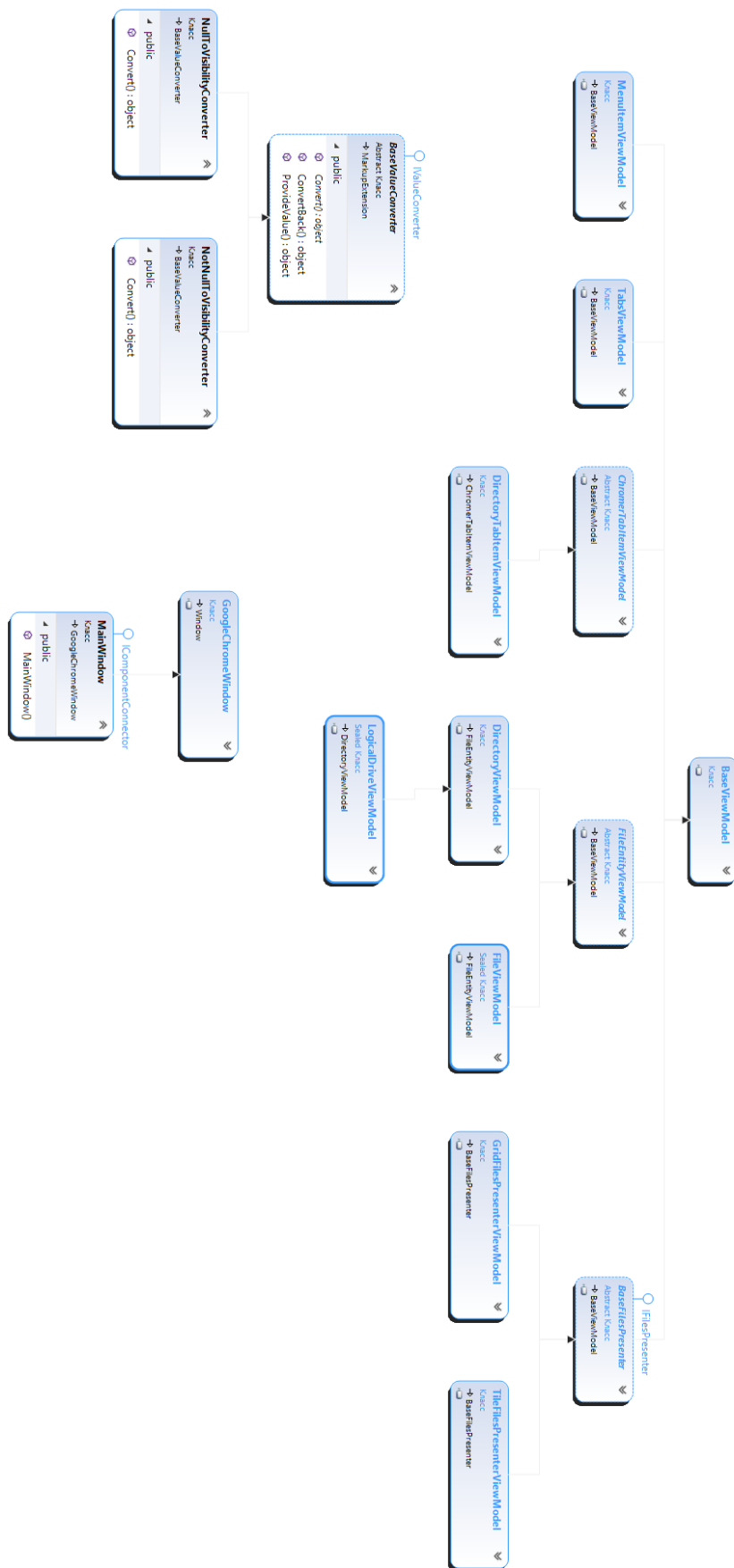


Рисунок 3 – UML – диаграмма классов

Большая часть программы обрабатывает системные события, нажатие кнопок и прочее. Поэтому невозможно выдклить главный фрагмент программы.

6. Руководство пользователя.

При входе в программу перед пользователем будет открыто основное окно рисунок 4.

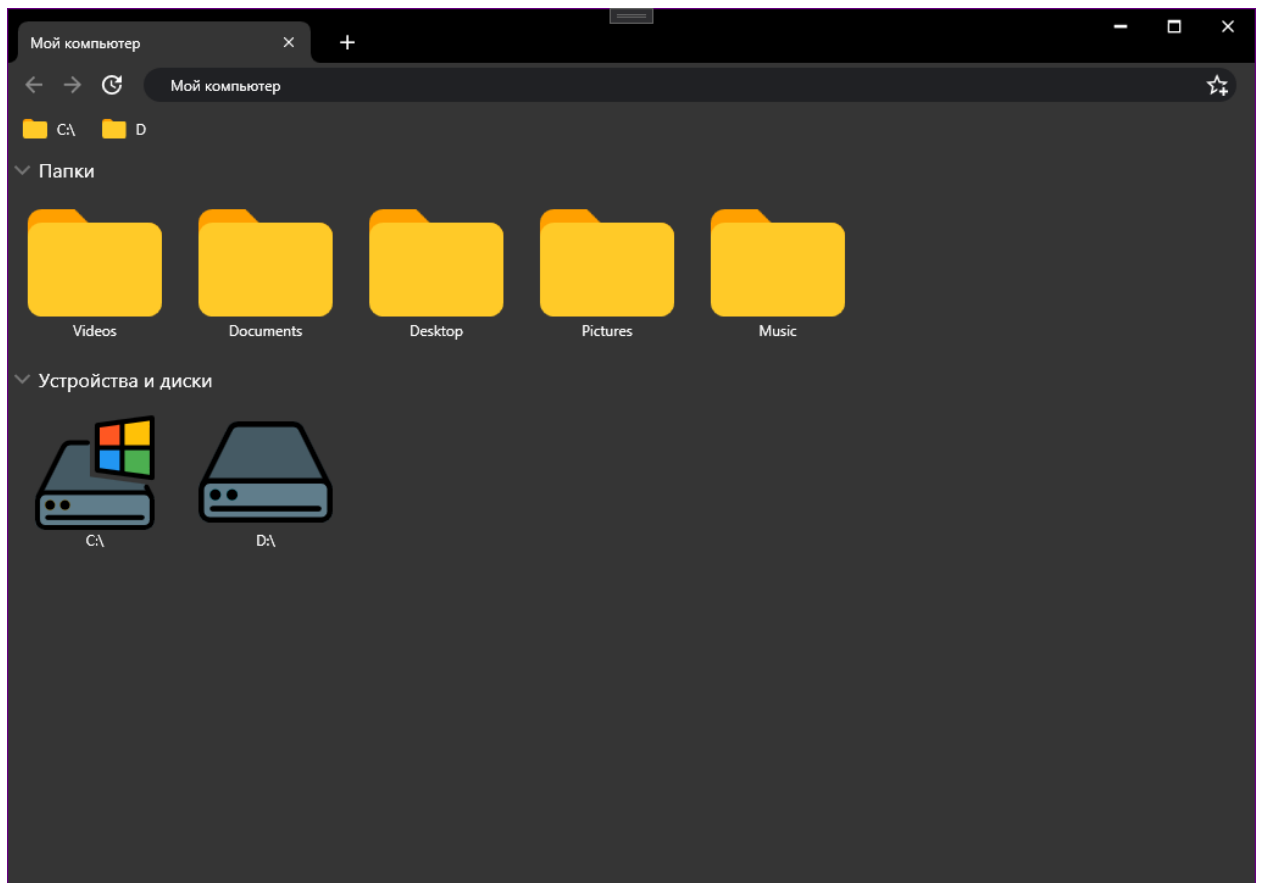


Рисунок 4– Основное окно программы

Пользователь может просматривать директории кликнув по папке дважды или через контекстное меню ПКМ на директорию рисунок 5. Есть возможность открыть в той же вкладке рисунок 6, в новой вкладке рисунок 7 или открыть в новом окне рисунок 8.

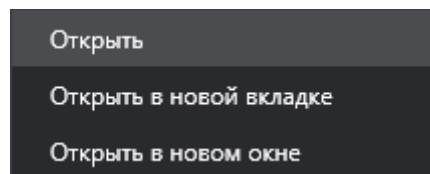


Рисунок 5 – Контекстное меню директории

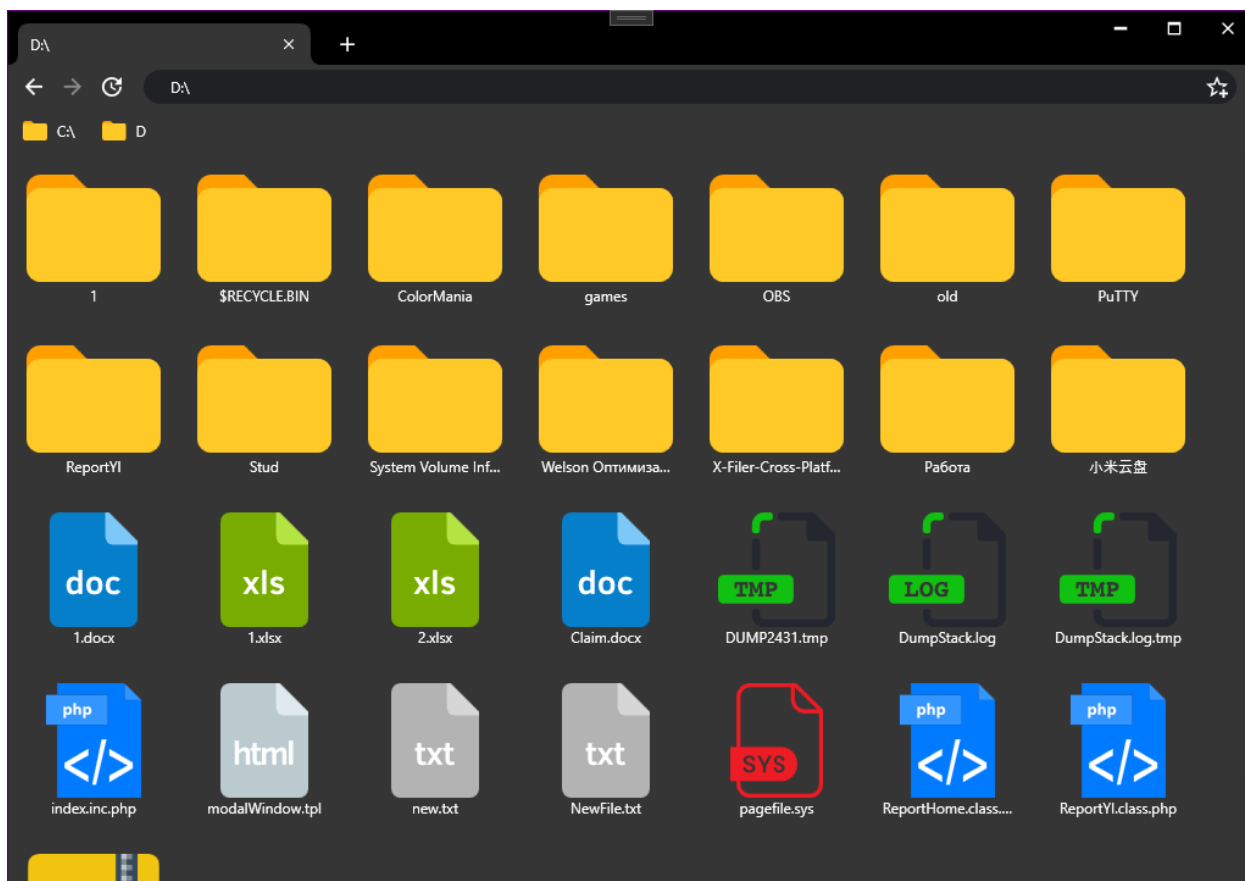


Рисунок 6 – Открытие директории в той же вкладке



Рисунок 7 – Открытие директории в новой вкладке

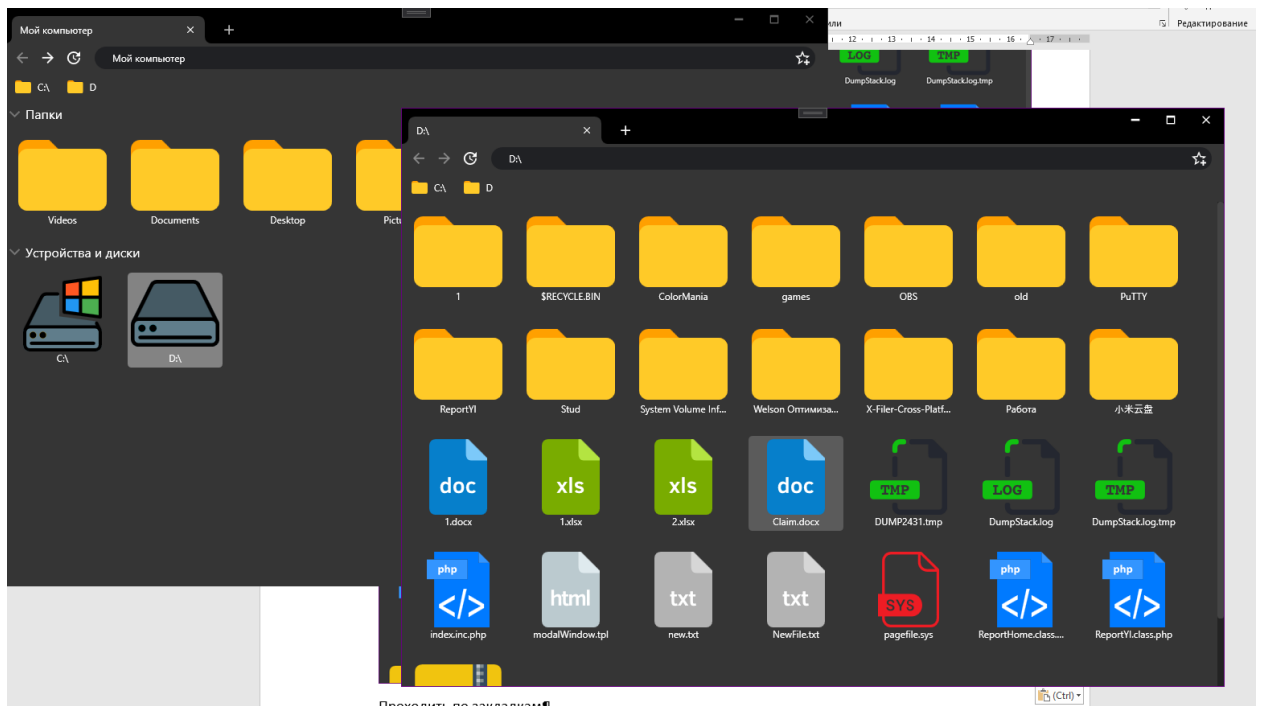


Рисунок 8 – Открытие директории в новом окне

Работа с закладками.

Пользователь может перейти на закладку в текущей вкладке рисунок 9,10.

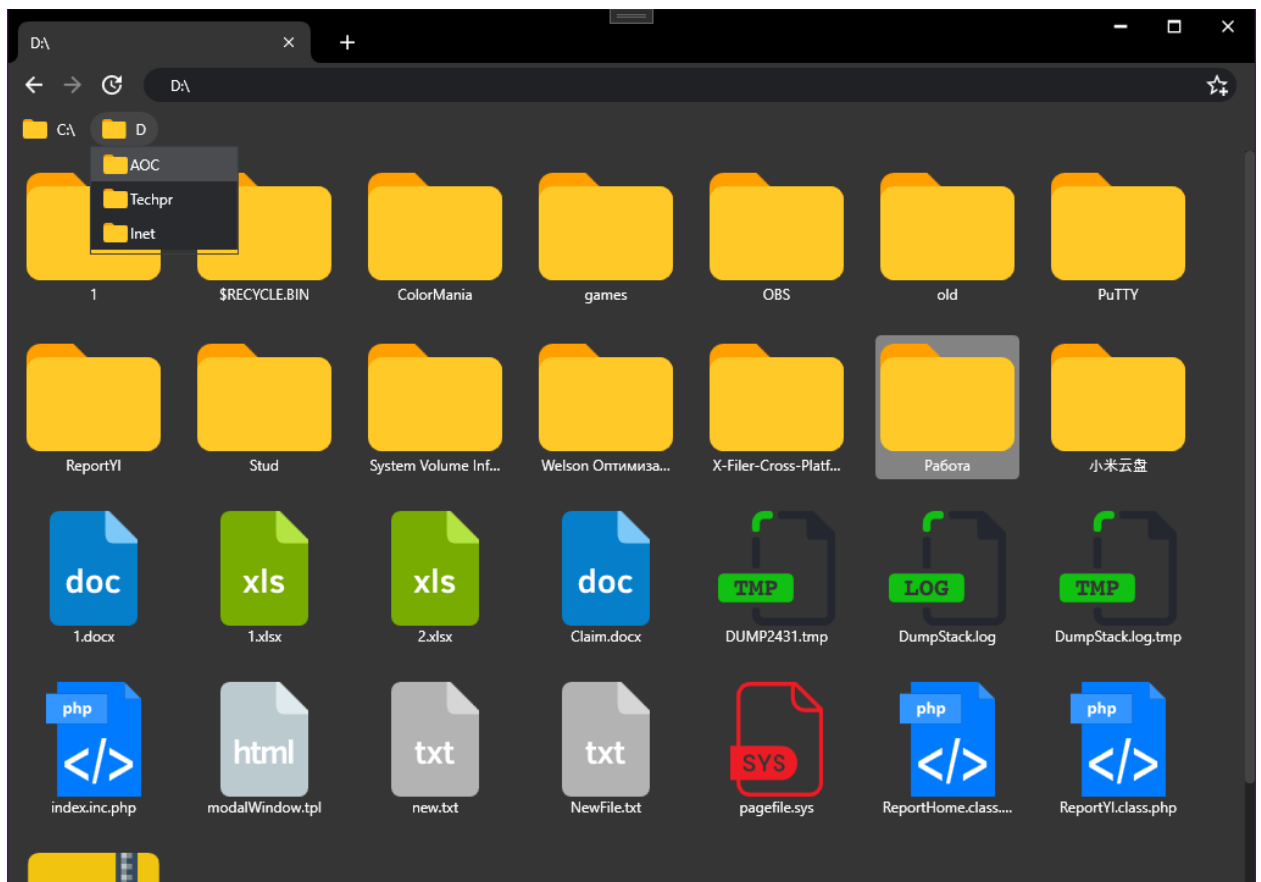


Рисунок 9 – Открытие закладки

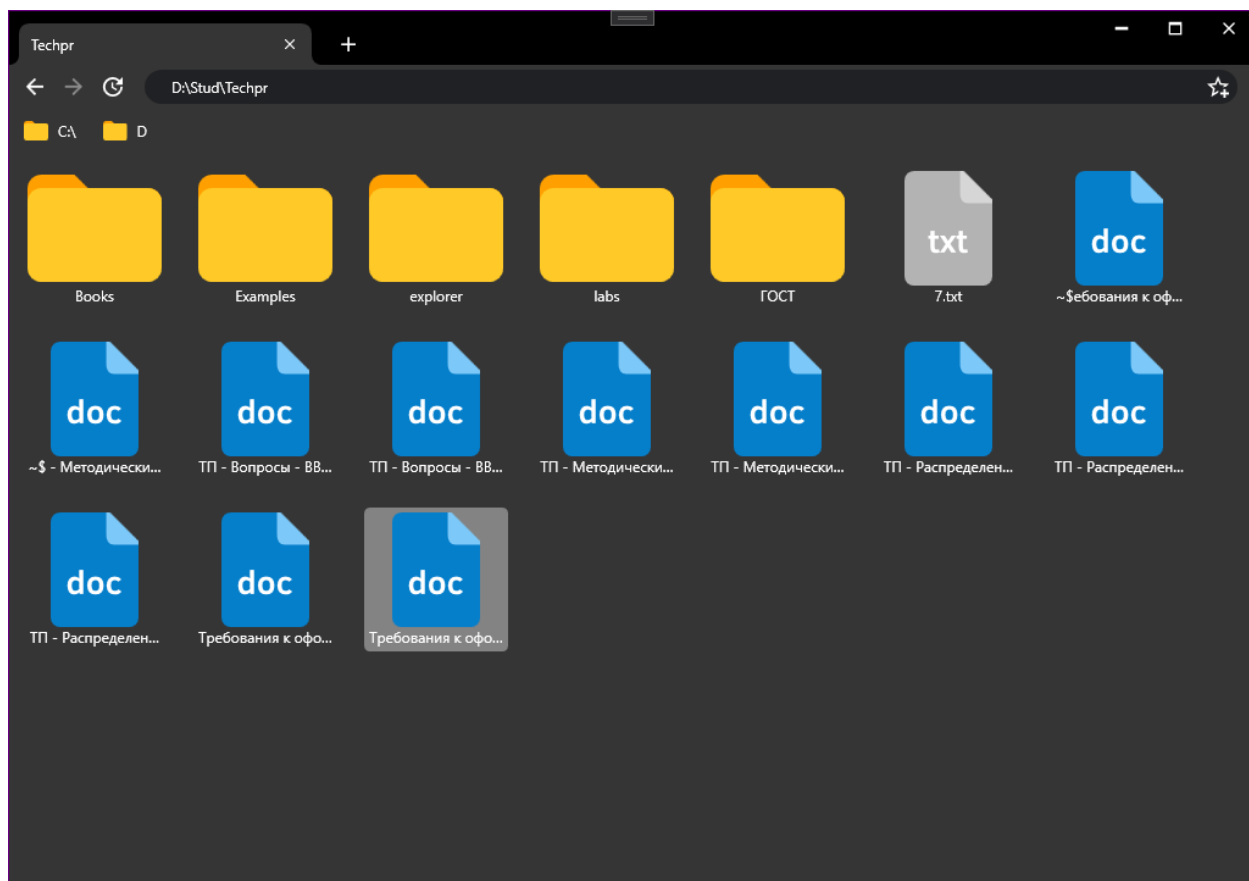


Рисунок 10 – Открытая закладка

Пользователь может добавлять закладки зайдя в директорию и нажав на кнопку звездочка рисунок 11.

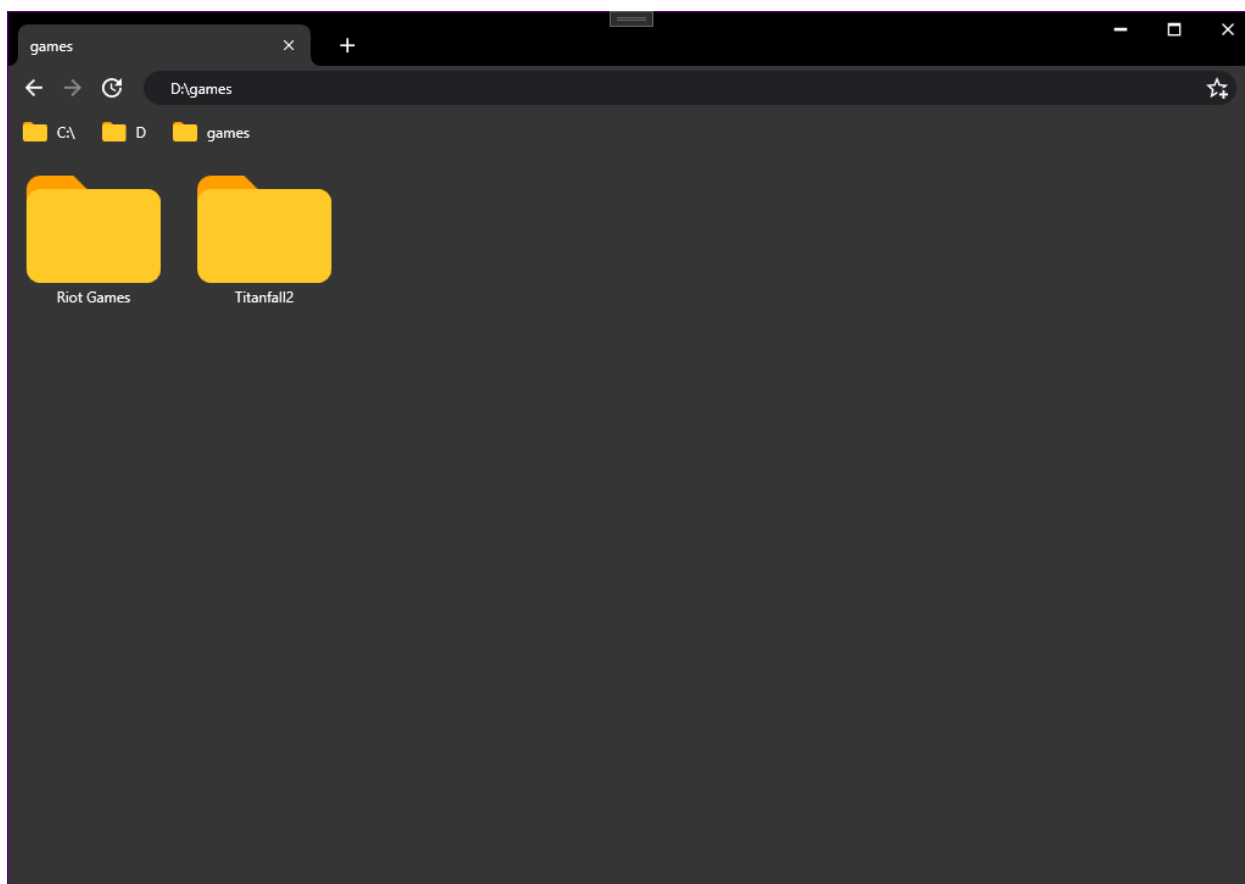


Рисунок 11 – Добавление закладки

На данный момент удаление и редактирование закладок можно проводить только в Json файле рисунок 12

```

D: > Stud > Techpr > explorer > Explorer > src > Explorer.Application > {} bookmarks.json > {} 2 > Path
1  [
2  {
3      "Path": "C:"
4  },
5  {
6      "BookmarkFolderName": "D",
7      "Children": [
8          {
9              "Path": "D:/Stud/AOC"
10         },
11         {
12             "Path": "D:/Stud/Techpr"
13         },
14         {
15             "Path": "D:/Stud/Inet"
16         }
17     ]
18 },
19 {
20     "Path": "D:/games"
21 }
22 ]

```

Рисунок 11 – Редактирование закладок в json

Работа с файлами.

Пользователь может открывать файлы двойным щелчком мыши по иконке или через контекстное меню ПКМ рисунок 12, 13. (надо доработать, убрать лишние пункты)

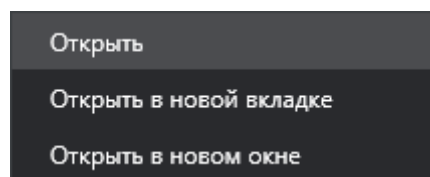


Рисунок 12 – контекстное меню файла

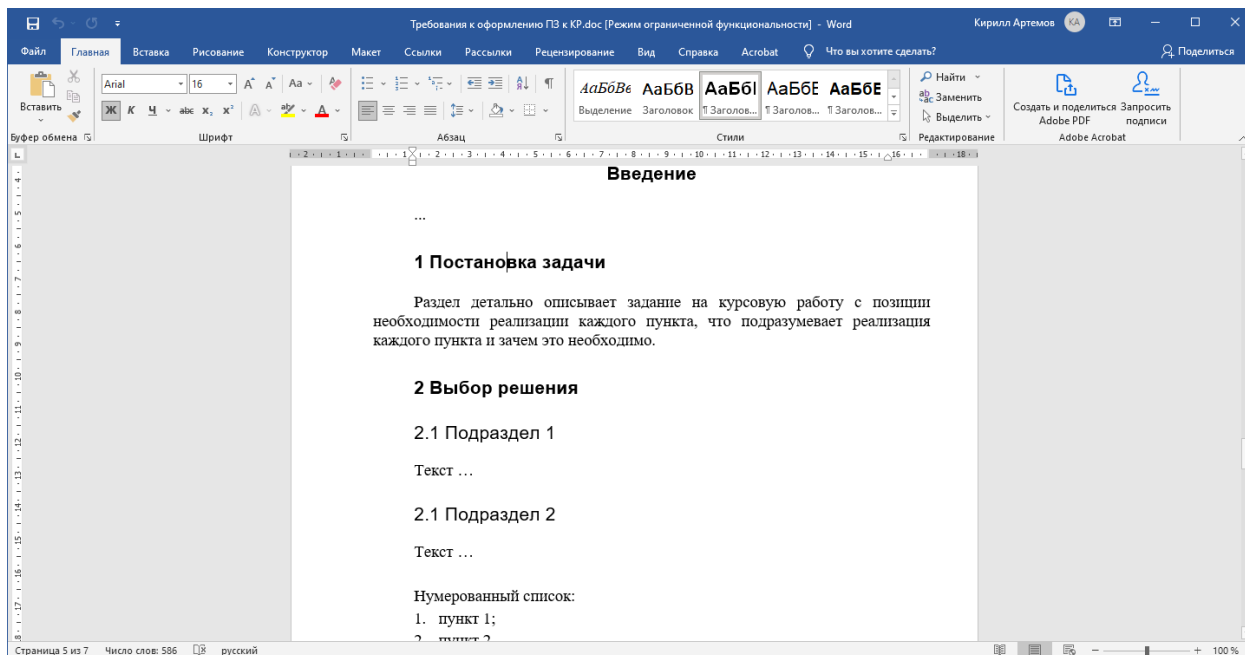


Рисунок 13 – Открытый файл

Работа с вкладками.

Пользователь может открывать новые вкладки рисунок 14, перемещать их потянув за вкладку рисунок 15.

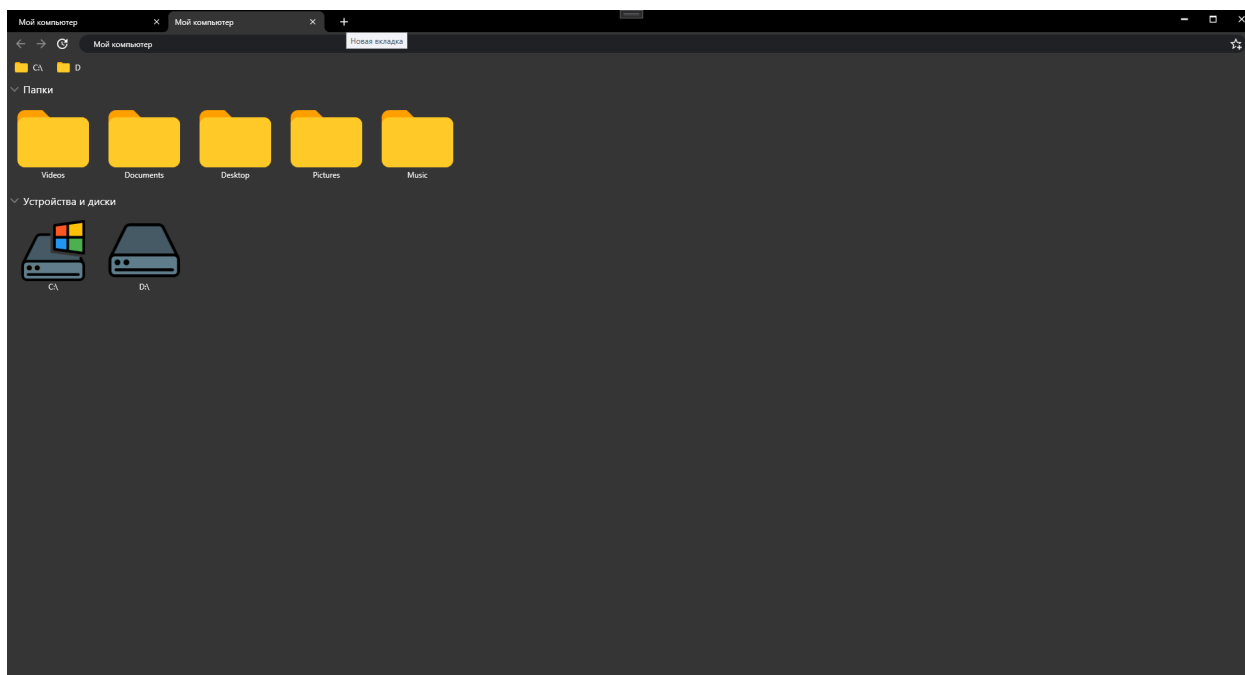


Рисунок 14 – Добавление вкладки

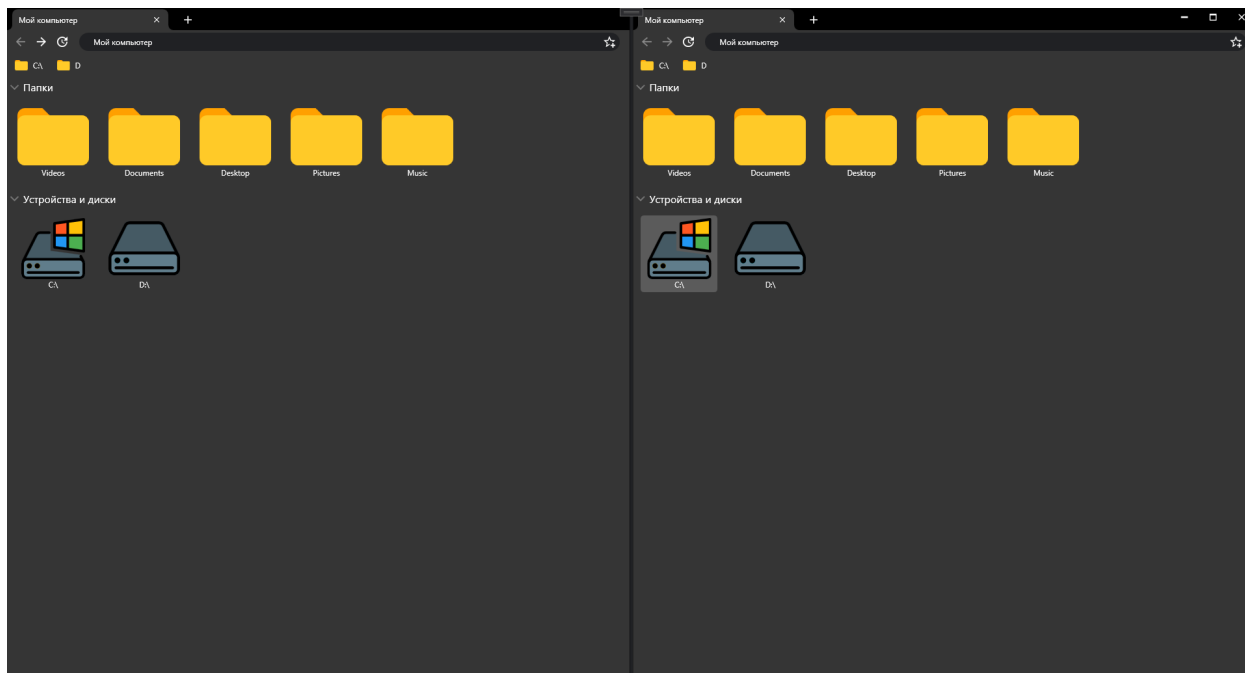


Рисунок 15 – Добавление вкладки

Перемещение по каталогам.

С помощью следующих клавиш пользователь может перемещаться по истории открываемых директорий рисунок 16,17.

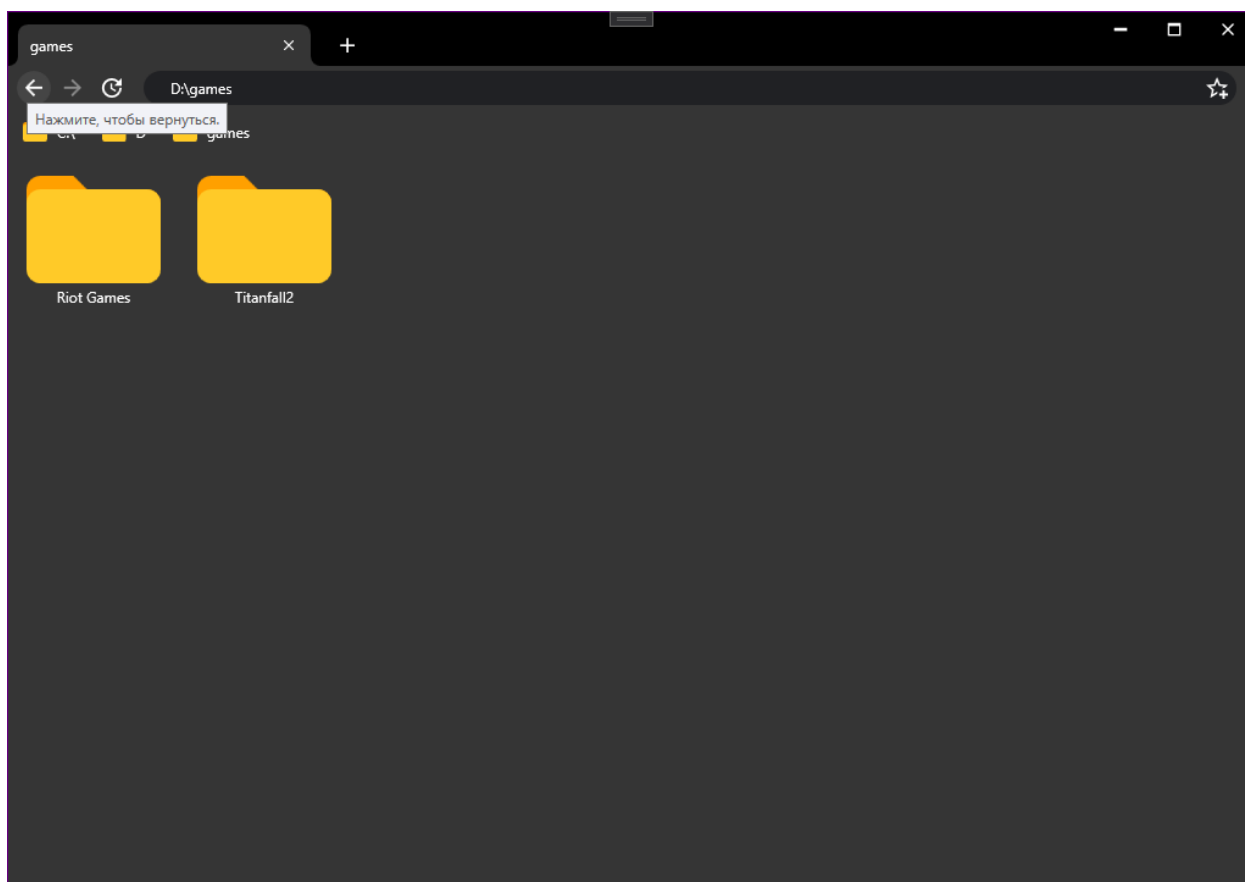


Рисунок 16 – Возврат назад

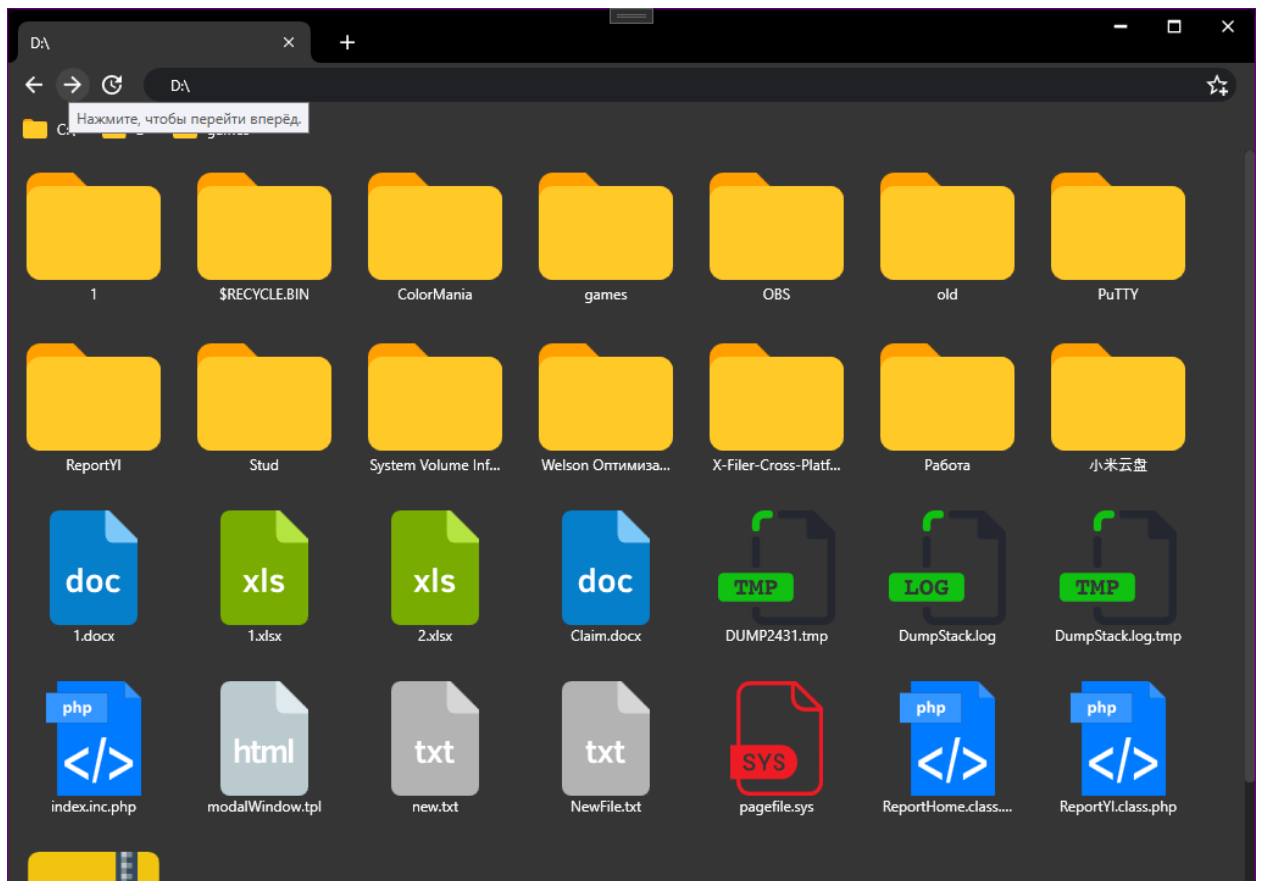


Рисунок 17 – Возврат вперед

Пользователь может выбрать вид отображения плитка или таблица рисунок 18.

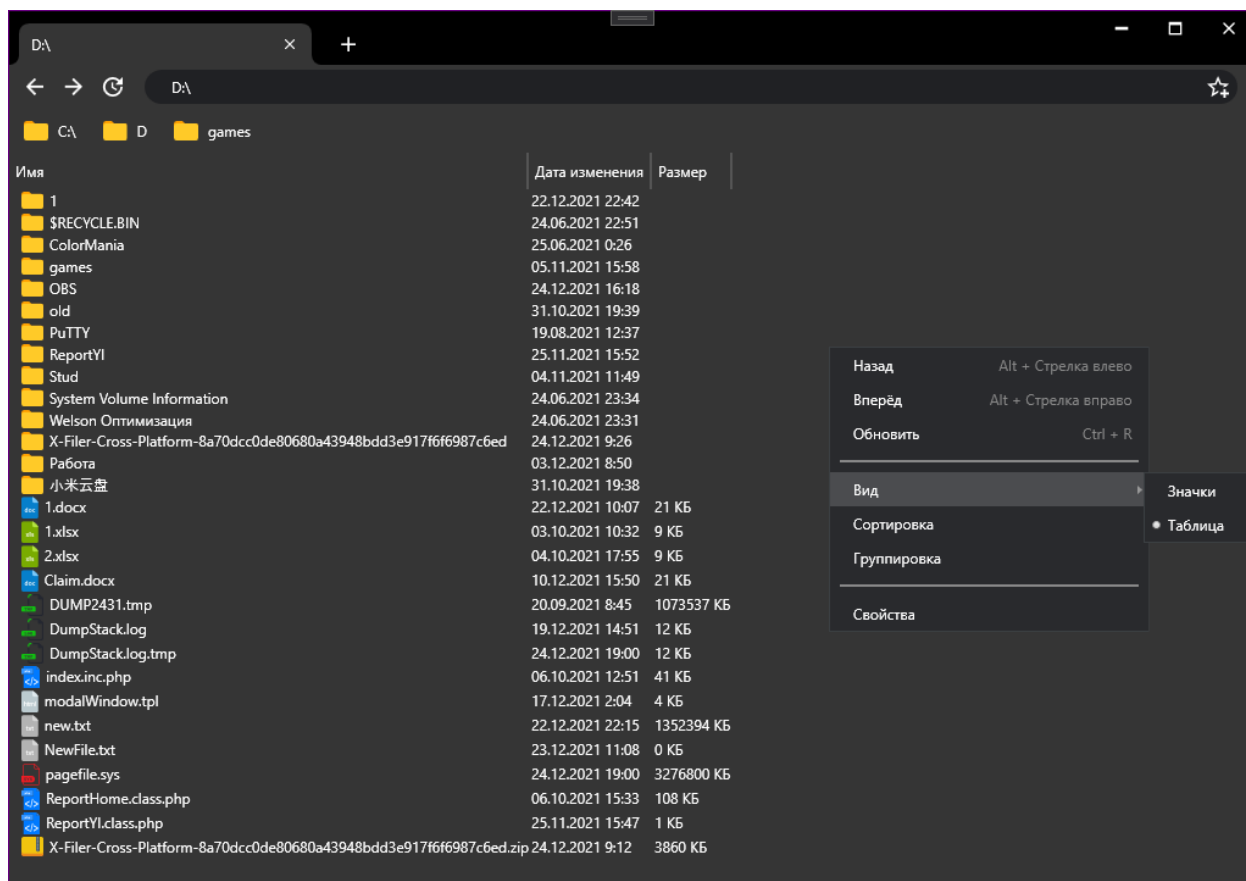


Рисунок 18 – Вид отображения

Drag and drop не доведен до ума рисунок 19.

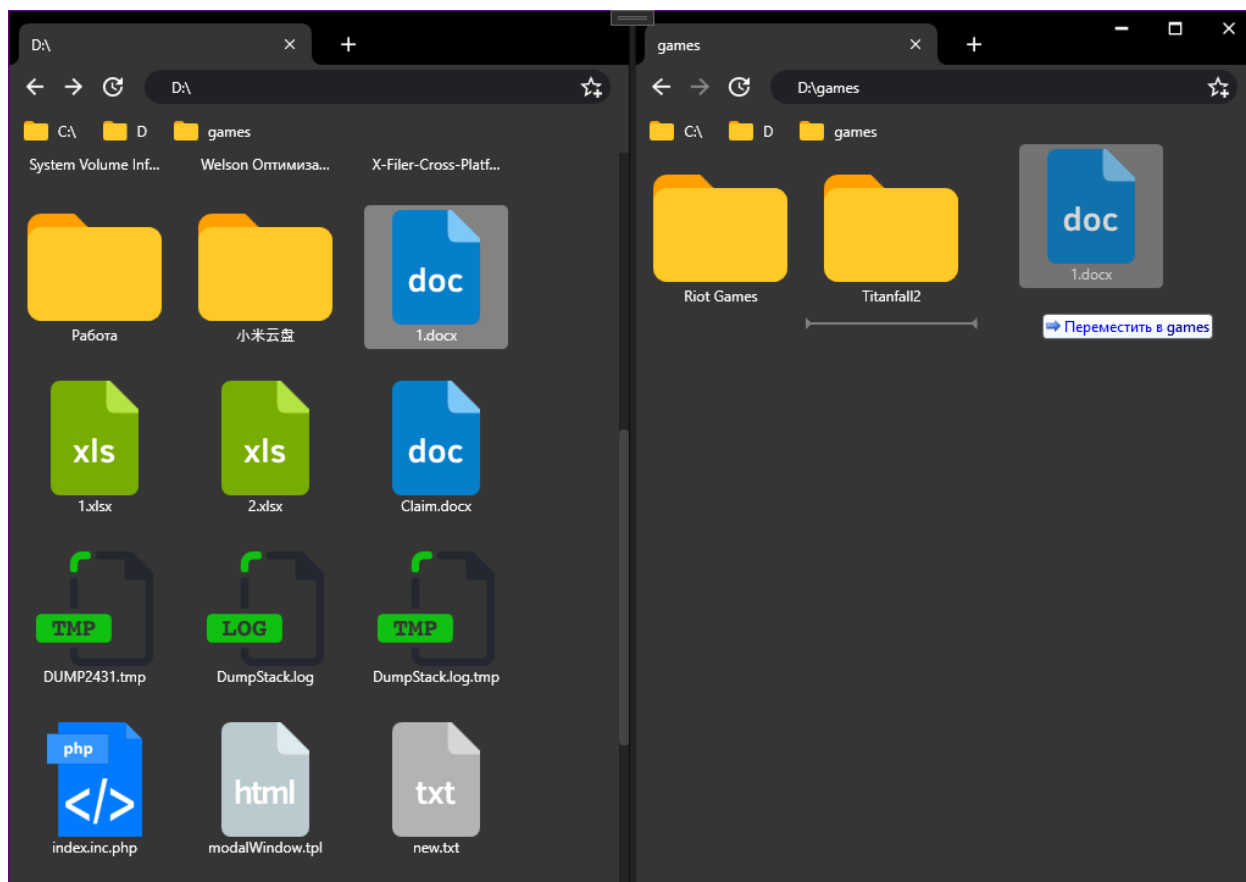


Рисунок 19– Проблема drag and drop

Перемещение не выполняется хотя все правила прописаны

Заключение

При выполнении данной курсовой работы были получены навыки разработки программ на языке высокого уровня. Были освоены приемы создания графического интерфейса. Усвоены механизмы реализации меню. Изучены функции работы с файлами. Получены базовые навыки программирования на языке C#. Изучены основные возможности среды программирования Visual Studio 2019. Получены навыки отладки и тестирования программ.

В рамках выполнения курсовой работы был написан Проводник. Не все функции удалось реализовать. Поэтому программу можно улучшить, оптимизировав код для более рационального использования графических ресурсов, доработав drag and drop, поиск через адресную строку, окно управления закладками как в браузере, изменение файлов и директорий.

Список используемых источников.

1. Основы программирования на C# 2009 г. Автор: Стенли Липпман.
2. Изучаем C#. 3-е изд. Авторы: Э. Стиллимен, Дж. Грин. 2010 г.
3. Полный учебник по WPF <https://www.wpf-tutorial.com/about-wpf/what-is-wpf/>
4. Ещё один учебник <https://metanit.com/sharp/wpf/1.php>
5. Документации WPF от Microsoft
<https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop-6.0>
<https://docs.microsoft.com/ru-ru/dotnet/api/system.io?view=net-6.0>
6. Готовая библиотека drag and drop <https://github.com/punker76/gong-wpf-dragdrop>
7. <https://www.youtube.com/>
8. <https://www.cyberforum.ru/>
9. <https://stackoverflow.com/>

Приложение А

Листинг модуля Explorer.WPF.UI

Приложение А.1

Файл « App.xaml.cs»

```
using System.Globalization;
using System.Linq;
using System.Windows;
using System.Windows.Markup;
using Explorer;
using Point = System.Drawing.Point;

namespace Explorer.WPF.UI
{
    public partial class App
    {
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);

            FrameworkElement.LanguageProperty.OverrideMetadata(typeof(FrameworkElement),
                new FrameworkPropertyMetadata(XmlLanguage.GetLanguage(CultureInfo.CurrentCulture.IetfLanguageTag)));

            WpfSynchronizationHelper synchronizationHelper =
new();

            ExplorerEr.CreateChromer(synchronizationHelper, new
BoundExampleInterTabClient(),
                ShowNewWindow);

            var mainViewModel =
ExplorerEr.Instance.CreateMainViewModel(new
DirectoryTabItemViewModel[0]);

            var myCompTabVm = new
DirectoryTabItemViewModel(synchronizationHelper,
ExplorerEr.RootName, ExplorerEr.RootName);

            mainViewModel.TabItems.Add(myCompTabVm);

            MainWindow mainWindow = new()
            {
                DataContext = mainViewModel
            };

            mainWindow.Show();
        }
    }
}
```

```

        //DebugWindow debugWindow = new();
        //debugWindow.Show();
    }

    private static void ShowNewWindow(TabsViewModel mvm,
Point location)
    {
        var activeWindow =
Current.Windows.OfType<Window>().FirstOrDefault(w => w.IsActive)
?? Current.MainWindow;

        MainWindow mainWindow = new()
        {
            DataContext = mvm,
            WindowStartupLocation =
WindowStartupLocation.Manual,
            Left = activeWindow.Left + location.X,
            Top = activeWindow.Top + location.Y
        };

        mainWindow.Show();
    }
}

```

Приложение A.2

Файл « App.xaml »

```

<Application x:Class="Explorer.WPF.UI.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:ui="clr-namespace:Explorer.WPF.UI"
xmlns:explorER="clr-
namespace:ExplorER;assembly=ExplorER"
ShutdownMode="OnLastWindowClose">

    <Application.Resources>

        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="Themes/App/Colors.xaml" />
                <ResourceDictionary
Source="Themes/App/Menu.xaml" />
                <ResourceDictionary
Source="Themes/App/Icons.xaml" />
                <ResourceDictionary
Source="Themes/App/Scroll.xaml" />
            
```

```

        <ResourceDictionary
Source="Themes/App/Expanders.xaml" />

        <ResourceDictionary
Source="pack://application:,,,/Dragablz;component/Themes/Generic
.xaml" />

        <ResourceDictionary
Source="Themes/App/Dragablze.xaml" />

        <ResourceDictionary>
            <DataTemplate DataType="{x:Type
explorEr:DirectoryTabItemViewModel}">
                <ui:DirectoryTabItem />
            </DataTemplate>

            <Style TargetType="{x:Type GroupItem}"
                x:Key="ExpanderGroupItem">
                <Setter Property="Template">
                    <Setter.Value>
                        <ControlTemplate
TargetType="{x:Type GroupItem}">
                            <Grid>
                                <Expander
IsExpanded="True"
Foreground="White"
x:Name="Expander">
                                    <Expander.Header>
                                        <TextBlock
Text="{Binding Name}"
FontSize="16" />
                                    </Expander.Header>
                                    <ItemsPresenter />
                                </Expander>
                                <ItemsPresenter
x:Name="Presenter"
Visibility="Collapsed" />
                            </Grid>
                            <ControlTemplate.Triggers>
                                <DataTrigger
Binding="{Binding Name}"
Value="{x:Null}">
                                    <Setter
TargetName="Presenter" Property="Visibility"
Value="Visible" />
                                </DataTrigger>
                            </ControlTemplate.Triggers>
                        </ControlTemplate>
                    </Setter.Value>
                </Setter>
            </Style>
        </ResourceDictionary>

```

```

                                <Setter
TargetName="Expander" Property="Visibility"
Value="Collapsed" />
                                </DataTrigger>
                                </ControlTemplate.Triggers>
                                </ControlTemplate>
                                </Setter.Value>
                                </Setter>
                                </Style>

                                </ResourceDictionary>

                                </ResourceDictionary.MergedDictionaries>
                                </ResourceDictionary>

                                </Application.Resources>

</Application>

```

Приложение A.3

Файл « MainWindow.xaml.cs »

```

namespace Explorer.WPF.UI
{
    public partial class MainWindow
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}

```

Приложение A.4

Файл « MainWindow.xaml »

```

<exWindow:GoogleChromeWindow
x:Class="Explorer.WPF.UI.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:dockablz="http://dragablz.net/winfx/xaml/dockablz"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```

```

xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                                xmlns:ui="clr-
namespace:Explorer.WPF.UI"
                                xmlns:exWindow="clr-
namespace:EXWindow;assembly=EXWindow"
                                mc:Ignorable="d"

                                Title="{Binding CurrentTabItem.Header}"

                                Height="720" Width="1024">

        <dockablz:Layout Partition="2AE89D18-F236-4D20-9605-
6C03319038E6"
                                IsFloatDropZoneEnabled="False"
                                x:Name="Layout">

                <!-- branch template lets dragablz create a new tab
control after a window is split via docking -->
                <dockablz:Layout.BranchTemplate>
                        <DataTemplate>
                                <ui:MainView />
                        </DataTemplate>
                </dockablz:Layout.BranchTemplate>

                <!-- a root, named tab control is needed, so when a tab
is torn and a new window is created, the new target tab control
can be identified -->
                <ui:MainView x:Name="InitMainView" />

        </dockablz:Layout>

</exWindow:GoogleChromeWindow>

```

Приложение A.5

Файл « CalcMaxWidthTabItemConverter.cs»

```

using System;
using System.Globalization;
using System.Windows.Data;
using System.Windows.Markup;
using EXWindow;

namespace Explorer.WPF.UI
{
    public class CalcMaxWidthTabItemConverter : MarkupExtension,
IMultiValueConverter
    {
        public override object ProvideValue(IServiceProvider
serviceProvider)
        {

```

```

        return this;
    }

    public object Convert(object[] values, Type targetType,
object parameter, CultureInfo culture)
    {
        if (values.Length == 2 &&
            values[0] is double controlWidth &&
            values[1] is int itemCount)
        {
            var newWidth = (controlWidth -
GoogleChromeWindow.SystemButtonsWidth - 10.0) / itemCount;
            if (newWidth < 256.0)
                return newWidth;
        }

        return 256.0;
    }

    public object[] ConvertBack(object value, Type[]
targetTypes, object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}

```

Приложение A.6

Файл « CloneParamsConverter.cs»

```

using System;
using System.Globalization;
using System.Windows.Data;
using System.Windows.Markup;

namespace Explorer.WPF.UI
{
    public class CloneParamsConverter : MarkupExtension,
IMultiValueConverter
    {
        public object Convert(object[] values, Type targetType,
object parameter, CultureInfo culture)
            => values.Clone();

        public object[] ConvertBack(object value, Type[]
targetTypes, object parameter, CultureInfo culture)
            => throw new NotImplementedException();

        public override object ProvideValue(IServiceProvider
serviceProvider) => this;
    }
}

```

Приложение А.7

Файл « FileEntityToImageConverter.cs»

```
using System;
using System.Globalization;
using System.Windows.Data;
using System.Windows.Markup;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using Explorer;
using Explorer.FileEntities.Base;
using SharpVectors.Converters;
using SharpVectors.Renderers.Wpf;

namespace Explorer.WPF.UI
{
    internal class FileEntityToImageConverter : MarkupExtension,
        IValueConverter
    {
        public object Convert(object value, Type targetType,
            object parameter, CultureInfo culture)
        {
            var drawingImage = new DrawingImage();

            if (!(value is FileEntityViewModel viewModel))
                return drawingImage;

            var imagePath =
                ExplorerEr.Instance.IconsManager.GetIconPath(viewModel);

            if (imagePath.Extension.ToUpper() == ".SVG")
            {
                var settings = new WpfDrawingSettings
                {
                    TextAsGeometry = false,
                    IncludeRuntime = true,
                };

                var converter = new FileSvgReader(settings);

                var drawing =
                    converter.Read(imagePath.FullName);

                if (drawing != null)
                    return new DrawingImage(drawing);
            }
            else
            {
                var bitmapSource = new BitmapImage(new
                    Uri(imagePath.FullName));
                return bitmapSource;
            }
        }
    }
}
```

```

        return dravingImage;
    }

    public object ConvertBack(object value, Type targetType,
object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }

    public override object ProvideValue(IServiceProvider
serviceProvider)
    {
        return this;
    }
}

```

Приложение A.8

Файл « IconPathToImageConverter.cs»

```

using System;
using System.Globalization;
using System.IO;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Markup;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using SharpVectors.Converters;
using SharpVectors.Renderers.Wpf;

namespace Explorer.WPF.UI
{
    internal class IconPathToImageConverter : MarkupExtension,
IValueConverter
    {
        public override object ProvideValue(IServiceProvider
serviceProvider)
        {
            return this;
        }

        public object Convert(object value, Type targetType,
object parameter, CultureInfo culture)
        {
            var image = new Image();
            image.Stretch = Stretch.Uniform;

            if (value is string imagePaths)
            {
                var imagePath = new FileInfo(imagePaths);

```

```

        if (imagePath.Extension.ToUpper() == ".SVG")
        {
            var settings = new WpfDrawingSettings
            {
                TextAsGeometry = false,
                IncludeRuntime = true,
            };

            var converter = new FileSvgReader(settings);

            var drawing =
converter.Read(imagePath.FullName);

            if (drawing != null)
            {
                var drawImage = new
DrawingImage(drawing);
                image.Source = drawImage;
            }
            else
            {
                var bitmapSource = new BitmapImage(new
Uri(imagePath.FullName));
                image.Source = bitmapSource;
            }
        }

        return image;
    }

    public object ConvertBack(object value, Type targetType,
object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}
}

```

Приложение A.9

Файл « NullToVisibilityConverter.cs»

```

using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;
using System.Windows.Markup;

namespace Explorer.WPF.UI
{

```

```

        internal class NullToVisibilityConverter :
BaseValueConverter
        {
            public override object Convert(object value, Type
targetType, object parameter, CultureInfo culture)
                => value == null ? Visibility.Visible :
Visibility.Collapsed;
        }

        internal class NotNullToVisibilityConverter :
BaseValueConverter
        {
            public override object Convert(object value, Type
targetType, object parameter, CultureInfo culture)
                => value != null ? Visibility.Visible :
Visibility.Collapsed;
        }

        internal abstract class BaseValueConverter :
MarkupExtension, IValueConverter
        {
            public override object ProvideValue(IServiceProvider
serviceProvider) => this;

            public abstract object Convert(object value, Type
targetType, object parameter, CultureInfo culture);

            public object ConvertBack(object value, Type targetType,
object parameter, CultureInfo culture)
            {
                throw new NotImplementedException();
            }
        }
    }
}

```

Приложение A.10

Файл « CommandReference.cs»

```

using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Input;

namespace Explorer.WPF.UI
{
    public class CommandReference : Freezable, ICommand
    {
        #region Dependency Properties

        public static readonly DependencyProperty
CommandProperty = DependencyProperty.Register(

```

```

        nameof(Command), typeof(ICommand),
typeof(CommandReference),
        new PropertyMetadata(OnCommandChanged));

    private static void OnCommandChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
    {
        if (d is not CommandReference commandReference)
            return;

        var oldCommand = e.OldValue as ICommand;
        var newCommand = e.NewValue as ICommand;

        commandReference.OnCommandChanged(oldCommand,
newCommand);
    }

    public static readonly DependencyProperty
CommandParameterProperty = DependencyProperty.Register(
        nameof(CommandParameter), typeof(object),
typeof(CommandReference),
        new PropertyMetadata(default(object)));

#endregion

#region Public Properties

public ICommand? Command
{
    get => (ICommand) GetValue(CommandProperty);
    set => SetValue(CommandProperty, value);
}

public object? CommandParameter
{
    get => GetValue(CommandParameterProperty);
    set => SetValue(CommandParameterProperty, value);
}

#endregion

#region ICommand Members

public bool CanExecute(object? parameter)
    => Command != null &&
Command.CanExecute(GetParameter(parameter, CommandParameter));

public void Execute(object? parameter)
    => Command?.Execute(GetParameter(parameter,
CommandParameter));

public event EventHandler? CanExecuteChanged;

```

```

#endregion

#region Private Methods

    private static object? GetParameter(object?
mainParameter, object? referenceParameter)
    {
        if (mainParameter == null)
            return referenceParameter;

        if (referenceParameter == null)
            return mainParameter;

        List<object> mergesParameters = new();

        if (referenceParameter is object[] arrayParams2)
            mergesParameters.AddRange(arrayParams2);
        else
            mergesParameters.Add(referenceParameter);

        if (mainParameter is object[] arrayParams)
            mergesParameters.AddRange(arrayParams);
        else
            mergesParameters.Add(mainParameter);

        return mergesParameters.ToArray();
    }

    private void OnCommandChanged(ICommand? oldCommand,
ICommand? newCommand)
    {
        if (oldCommand != null)
            oldCommand.CanExecuteChanged -=
OldCommandOnCanExecuteChanged;

        if (newCommand != null)
            newCommand.CanExecuteChanged +=
OldCommandOnCanExecuteChanged;
    }

    private void OldCommandOnCanExecuteChanged(object?
sender, EventArgs e) => RaiseCanExecuteChanged();

    private void RaiseCanExecuteChanged() =>
CanExecuteChanged?.Invoke(this, EventArgs.Empty);

#endregion

#region Freezable

    protected override Freezable CreateInstanceCore() =>
throw new NotImplementedException();

```

```

        #endregion
    }
}

```

Приложение A.11

Файл «WpfSynchronizationHelper.cs»

```

using System;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Threading;
using Explorer;

namespace Explorer.WPF.UI
{
    internal class WpfSynchronizationHelper :
    ISynchronizationHelper
    {
        public async Task InvokeAsync(Action action)
            => await
        Application.Current.Dispatcher.InvokeAsync(action,
        DispatcherPriority.Background);
    }
}

```

Приложение A.12

Файл «XButtonsGestureHelper.cs»

```

using System.ComponentModel;
using System.Windows.Input;
using System.Windows.Markup;

namespace Explorer.WPF.UI
{
    public class ExtendedMouseBinding : MouseBinding
    {
        [ValueSerializer(typeof(MouseGestureValueSerializer)),
        TypeConverter(typeof(ExtendedMouseGestureConverter))]
        public override InputGesture Gesture
        {
            get => base.Gesture;
            set => base.Gesture = value;
        }
    }

    public class ExtendedMouseGestureConverter :
    MouseGestureConverter
    {

```

```

        public override object
ConvertFrom(ITypeDescriptorContext context,
System.Globalization.CultureInfo culture, object source)
    {
        switch (source.ToString())
        {
            case "XButton1":
                return new
ExtendedMouseGesture(MouseButton.XButton1);
            case "XButton2":
                return new
ExtendedMouseGesture(MouseButton.XButton2);
        }
        return base.ConvertFrom(context, culture, source);
    }
}

public class ExtendedMouseGesture : MouseGesture
{
    private readonly MouseButton _mouseButton;

    public ExtendedMouseGesture(MouseButton mouseButton)
    {
        _mouseButton = mouseButton;
    }

    public override bool Matches(object targetElement,
InputEventArgs inputEventArgs)
    {
        var device = inputEventArgs.Device as MouseDevice;

        if (device != null)
        {
            switch (_mouseButton)
            {
                case MouseButton.XButton1:
                    if (device.XButton1 ==
MouseButtonState.Pressed) return true;
                    break;
                case MouseButton.XButton2:
                    if (device.XButton2 ==
MouseButtonState.Pressed) return true;
                    break;
            }
        }

        return false;
    }
}

```

Приложение Б

Листинг модуля EXWindow

Приложение Б.1

Файл «Windows.cs»

```
using System.Windows;

namespace EXWindow
{
    internal class Windows
    {
        public static readonly DependencyProperty
        IsActiveProperty = DependencyProperty.RegisterAttached(
            "IsActive", typeof(bool), typeof(Windows), new
            PropertyMetadata(default(bool)));

        public static void SetIsActive(DependencyObject element,
            bool value)
        {
            element.SetValue(IsActiveProperty, value);
        }

        public static bool GetIsActive(DependencyObject element)
        {
            return (bool) element.GetValue(IsActiveProperty);
        }

        public static readonly DependencyProperty
        WindowStateProperty = DependencyProperty.RegisterAttached(
            "WindowState", typeof(WindowState), typeof(Windows),
            new PropertyMetadata(default(WindowState)));

        public static void SetWindowState(DependencyObject
            element, WindowState value)
        {
            element.SetValue(WindowStateProperty, value);
        }

        public static WindowState
        GetWindowState(DependencyObject element)
        {
            return (WindowState)
            element.GetValue(WindowStateProperty);
        }
    }
}
```

Приложение Б.2

Файл «WindowResizeFixerBehavior.cs»

```

using System;
using System.Windows;
using Microsoft.Xaml.Behaviors;

namespace EXWindow
{
    internal class WindowResizeFixerBehavior : Behavior<Window>
    {
        private WindowResizer _fixer;

        protected override void OnAttached()
        {
            base.OnAttached();

            _fixer = new WindowResizer(AssociatedObject);

            AssociatedObject.StateChanged += (s, e) =>
UpdatePadding();

            AssociatedObject.Loaded += (s, e) =>
UpdatePadding();

            _fixer.WindowDockChanged +=
FixerOnWindowDockChanged;

            AssociatedObject.StateChanged +=
AssociatedObject_StateChanged;
        }

        private void AssociatedObject_StateChanged(object
sender, EventArgs e)
        {
            switch (AssociatedObject.WindowState)
            {
                case WindowState.Normal:
                    AssociatedObject.Margin = new Thickness(10);
                    break;
                case WindowState.Maximized:
                    AssociatedObject.Margin = new Thickness(0);
                    break;
            }
        }

        private void FixerOnWindowDockChanged(WindowDockPosition
dockPosition)
        {
            switch (dockPosition)
            {
                case WindowDockPosition.Undocked:
                    if (AssociatedObject.WindowState !=
WindowState.Maximized)
                        AssociatedObject.Margin = new
Thickness(10);
            }
        }
    }
}

```

```

        break;
    default:
        AssociatedObject.Margin = new Thickness(0);
        break;
    }
}

private void UpdatePadding()
{
    AssociatedObject.Padding =
AssociatedObject.WindowState == WindowState.Maximized
    ? _fixer.CurrentMonitorMargin
    : new Thickness(0);
}
}
}

```

Приложение Б.3

Файл «DelegateCommand.cs»

```

using System;
using System.Windows.Input;

namespace EXWindow
{
    internal class DelegateCommand : ICommand
    {
        private readonly Action<object> _execute;
        private readonly Predicate<object> _canExecute;

        public DelegateCommand(Action<object> execute,
Predicate<object> canExecute = null)
        {
            _execute = execute;
            _canExecute = canExecute;
        }

        public bool CanExecute(object parameter) => _canExecute
== null || _canExecute.Invoke(parameter);

        public void Execute(object parameter)
        {
            _execute?.Invoke(parameter);
        }

        public event EventHandler CanExecuteChanged;

        public void RaiseCanExecuteChanged()
        {
            CanExecuteChanged?.Invoke(this, EventArgs.Empty);
        }
    }
}

```

```
}
```

Приложение Б.4

Файл «GoogleChromeWindow.cs»

```
using System.Collections;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;

namespace EXWindow
{
    public class GoogleChromeWindow : Window
    {
        #region Dependency Properties

        public static readonly DependencyProperty
        TabsMaxWidthProperty = DependencyProperty.Register(
            nameof(TabsMaxWidth), typeof(double),
            typeof(GoogleChromeWindow), new
            PropertyMetadata(default(double)));

        public static readonly DependencyProperty
        CloseCommandProperty = DependencyProperty.Register(
            nameof(CloseCommand), typeof(ICommand),
            typeof(GoogleChromeWindow),
            new PropertyMetadata(default(ICommand)));

        public static readonly DependencyProperty
        CollapseCommandProperty = DependencyProperty.Register(
            nameof(CollapseCommand), typeof(ICommand),
            typeof(GoogleChromeWindow),
            new PropertyMetadata(default(ICommand)));

        public static readonly DependencyProperty
        ExpandCommandProperty = DependencyProperty.Register(
            nameof(ExpandCommand), typeof(ICommand),
            typeof(GoogleChromeWindow),
            new PropertyMetadata(default(ICommand)));

        public static readonly DependencyProperty
        ToolBarContentProperty = DependencyProperty.Register(
            nameof(ToolBarContent), typeof(FrameworkElement),
            typeof(GoogleChromeWindow),
            new PropertyMetadata(default(FrameworkElement)));

        public static readonly DependencyProperty
        ItemsSourceProperty = DependencyProperty.Register(
            nameof(ItemsSource), typeof(IEnumerable),
            typeof(GoogleChromeWindow),
            new PropertyMetadata(default(IEnumerable)));
    }
}
```

```

        public static readonly DependencyProperty
SelectedItemProperty = DependencyProperty.Register(
            nameof(SelectedItem), typeof(object),
typeof(GoogleChromeWindow), new
PropertyMetadata(default(object)));

        public static readonly DependencyProperty
TabItemTemplateProperty = DependencyProperty.Register(
            nameof(TabItemTemplate), typeof(DataTemplate),
typeof(GoogleChromeWindow),
            new PropertyMetadata(default(DataTemplate)));

        public static readonly DependencyProperty
ActiveTabBackgroundProperty = DependencyProperty.Register(
            nameof(ActiveTabBackground), typeof(Brush),
typeof(GoogleChromeWindow),
            new PropertyMetadata(default(Brush)));

#endregion

#region Public Properties

public static double SystemButtonsWidth { get; } = 44 *
3;

public double TabsMaxWidth
{
    get => (double) GetValue(TabsMaxWidthProperty);
    set => SetValue(TabsMaxWidthProperty, value);
}

public ICommand CloseCommand
{
    get => (ICommand) GetValue(CloseCommandProperty);
    set => SetValue(CloseCommandProperty, value);
}

public ICommand CollapseCommand
{
    get => (ICommand) GetValue(CollapseCommandProperty);
    set => SetValue(CollapseCommandProperty, value);
}

public ICommand ExpandCommand
{
    get => (ICommand) GetValue(ExpandCommandProperty);
    set => SetValue(ExpandCommandProperty, value);
}

public IEnumerable ItemsSource
{
    get => (IEnumerable) GetValue(ItemsSourceProperty);
    set => SetValue(ItemsSourceProperty, value);
}

```

```

    }

    public FrameworkElement ToolBarContent
    {
        get => (FrameworkElement)
GetValue(ToolBarContentProperty);
        set => SetValue(ToolBarContentProperty, value);
    }

    public object SelectedItem
    {
        get => (object) GetValue(SelectedItemProperty);
        set => SetValue(SelectedItemProperty, value);
    }

    public DataTemplate TabItemTemplate
    {
        get => (DataTemplate)
GetValue(TabItemTemplateProperty);
        set => SetValue(TabItemTemplateProperty, value);
    }

    public Brush ActiveTabBackground
    {
        get => (Brush)
GetValue(ActiveTabBackgroundProperty);
        set => SetValue(ActiveTabBackgroundProperty, value);
    }

#endregion

#region Static Constructor

static GoogleChromeWindow()
{

DefaultStyleKeyProperty.OverrideMetadata(typeof(GoogleChromeWind
ow),
        new
FrameworkPropertyMetadata(typeof(GoogleChromeWindow)));
}

#endregion

#region Constructor

public GoogleChromeWindow()
{
    CloseCommand = new DelegateCommand(OnClose);
    CollapseCommand = new DelegateCommand(OnCollapse);
    ExpandCommand = new DelegateCommand(OnExpand);
}

```

```

        WindowStartupLocation =
WindowStartupLocation.CenterScreen;

        var behavior = new WindowResizeFixerBehavior();
        behavior.Attach(this);

        //var blurBehavior = new BlurBehavior();
        //blurBehavior.Attach(this);

        CalcTabMaxWidth();

        SizeChanged += (sender, args) => CalcTabMaxWidth();
    }

#endregion

#region Private Methods

private void OnClose(object obj)
{
    Close();
}

private void OnCollapse(object obj)
{
    WindowState = WindowState.Minimized;
}

private void OnExpand(object obj)
{
    if (WindowState == WindowState.Normal)
        WindowState = WindowState.Maximized;
    else if (WindowState == WindowState.Maximized)
        WindowState = WindowState.Normal;
}

private void CalcTabMaxWidth()
{
    TabsMaxWidth = ActualWidth - 24 * 3 - 115 - 34;
}

#endregion
}
}

```

Приложение Б.5

Файл «WindowBorderThicknessConverter.cs»

```

using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

```

```

using System.Windows.Markup;

namespace EXWindow
{
    internal class WindowBorderThicknessConverter :
MarkupExtension, IValueConverter
    {
        public override object ProvideValue(IServiceProvider
serviceProvider)
        {
            return this;
        }

        public object Convert(object value, Type targetType,
object parameter, CultureInfo culture)
        {
            double length = 1;

            if (parameter is string lengthString)
                double.TryParse(lengthString, out length);

            if (value is WindowState windowState)
                return windowState == WindowState.Normal
                    ? new Thickness(length)
                    : new Thickness(0);

            return new Thickness(0);
        }

        public object ConvertBack(object value, Type targetType,
object parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}

```

Приложение Б.6

Файл «WindowTittleHeightConverter.cs»

```

using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;
using System.Windows.Markup;

namespace EXWindow
{
    internal class WindowTittleHeightConverter : MarkupExtension,
IValueConverter
    {

```

```

        public override object ProvideValue(IServiceProvider
serviceProvider)
        {
            return this;
        }

        public object Convert(object value, Type targetType,
object parameter, CultureInfo culture)
        {
            if (value is WindowState windowState)
                return windowState == WindowState.Normal ? 42 :
32;

            return 42;
        }

        public object ConvertBack(object value, Type targetType,
object parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}

```

Приложение В

Листинг модуля ExplorER

Приложение В.1

Файл «BookmarkItem.cs»

```
using System.Collections.Generic;

namespace ExplorER
{
    internal class BookmarkItem
    {
        public string Path { get; set; }

        public string BookmarkFolderName { get; set; }

        public IList<BookmarkItem> Children { get; set; }
    }
}
```

Приложение В.2

Файл «BookmarksManager.cs»

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Text.Json;
using ExplorER.Base;

namespace ExplorER
{
    internal class BookmarksManager : BaseViewModel,
        IBookmarksManager
    {
        #region Constants

        private const string BookmarksFileName =
            "bookmarks.json";

        #endregion

        #region Private Fields

        private readonly ExtensionToImageFileConverter
            _converter;

        private readonly ObservableCollection<MenuItemViewModel>
            _bookmarks;
    }
}
```

```

        private readonly List<BookmarkItem> _items;

        #endregion

        #region Public Properties

        public IReadOnlyCollection<MenuItemViewModel> Bookmarks
=> _bookmarks;
        public DelegateCommand AddBookmarkCommand { get; }

        #endregion

        #region Commands

        public DelegateCommand BookmarkClickCommand { get; }

        #endregion

        #region Constructor

        public BookmarksManager(ExtensionToImageFileConverter
converter)
        {
            _converter = converter;
            BookmarkClickCommand = new
DelegateCommand(OnBookmarkClicked);

            AddBookmarkCommand = new
DelegateCommand(OnAddBookmark);

            _items = OpenBookmarksFile();

            _bookmarks = CreateMenuItemViewModels(_items);
        }

        private ObservableCollection<MenuItemViewModel>
CreateMenuItemViewModels(IList<BookmarkItem> items)
        {
            var menuVms = new
ObservableCollection<MenuItemViewModel>();

            if (items == null || !items.Any())
                return menuVms;

            var applicationDirectory =
AppDomain.CurrentDomain.BaseDirectory;

            var iconsDirectory = new
DirectoryInfo(Path.Combine(applicationDirectory, "Resources",
"Icons"));

            foreach (var bookmarkItem in items)
            {

```

```

        var path = bookmarkItem.Path;

        var vm = new MenuItemViewModel(path)
        {
            Items =
CreateMenuItemViewModels(bookmarkItem.Children)
        };

        if (path == null)
        {
            vm.Header = bookmarkItem.BookmarkFolderName;
            vm.IconPath =
Path.Combine(iconsDirectory.FullName, IconName.BookmarkFolder +
".svg");
        }
        else
        {
            try
            {
                DefinitionPathToVm(vm, path,
iconsDirectory);
            }
            catch (Exception e)
            {
                continue;
            }
        }

        menuVms.Add(vm);
    }

    return menuVms;
}

private void DefinitionPathToVm(MenuItemViewModel vm,
string path, DirectoryInfo iconsDirectory)
{
    vm.Command = BookmarkClickCommand;

    var attr = System.IO.File.GetAttributes(path);

    if (attr.HasFlag(FileAttributes.Directory))
    {
        vm.Header = new DirectoryInfo(path).Name;
        vm.IconPath =
Path.Combine(iconsDirectory.FullName, IconName.Folder + ".svg");
    }
    else
    {
        var extension = new FileInfo(path).Extension;

        vm.Header = new FileInfo(path).Name;
        vm.IconPath =

```

```

_converter.GetImagePath(string.IsNullOrEmpty(extension) ? "" :
extension.Substring(1))
        .FullName;
    }
}

private List<BookmarkItem> OpenBookmarksFile()
{
    if (System.IO.File.Exists(BookmarksFileName))
    {
        var json =
System.IO.File.ReadAllText(BookmarksFileName);

        try
        {
            return
JsonSerializer.Deserialize<List<BookmarkItem>>(json);
        }
        catch (Exception e)
        {
        }
    }

    return new List<BookmarkItem>();
}

#endregion

#region Commands Methods

private void OnBookmarkClicked(object parameter)
{
    if (parameter is IList<object> parameters &&
        parameters.Count == 2 &&
        parameters[0] is string path &&
        parameters[1] is DirectoryTabItemViewModel
tabItemViewModel)
    {
        tabItemViewModel.OpenBookmark(path);
    }
}

private void OnAddBookmark(object obj)
{
    if (obj is string path && Directory.Exists(path))
    {
        _items.Add(new BookmarkItem
        {
            Path = path
        });
    }
}

```

```

        try
        {
            var json = JsonSerializer.Serialize(_items);

            System.IO.File.WriteAllText(BookmarksFileName, json);
        }
        catch (Exception e)
        {
        }

        _bookmarks.Clear();

        foreach (var viewModel in
CreateMenuItemViewModels(_items))
        {
            _bookmarks.Add(viewModel);
        }
    }

    #endregion
}
}

```

Приложение В.3

Файл «IBookmarksManager.cs»

```

using System.Collections.Generic;

namespace ExplorER
{
    /// <summary>
    /// Менеджер закладок
    /// </summary>
    public interface IBookmarksManager
    {
        IReadOnlyCollection<MenuItemViewModel> Bookmarks { get; }

        DelegateCommand AddBookmarkCommand { get; }
    }
}

```

Приложение В.4

Файл «MenuItemViewModel.cs»

```

using System.Collections.Generic;
using System.Windows.Input;
using ExplorER.Base;

```

```

namespace ExplorER
{
    /// <summary>
    /// View-Model MenuItem'a
    /// </summary>
    public class MenuItemViewModel : BaseViewModel
    {
        public string Path { get; set; }

        public string Header { get; set; }

        public ICommand Command { get; set; }

        public object CommandParameter { get; set; }

        public string IconPath { get; set; }

        public IList<MenuItemViewModel> Items { get; set; }

        public MenuItemViewModel(string path)
        {
            Path = path;

            CommandParameter = path;
        }
    }
}

```

Приложение B.5

Файл «DelegateCommand.cs»

```

using System;
using System.Windows.Input;

namespace ExplorER
{
    public class DelegateCommand : ICommand
    {
        private readonly Action<object?> _execute;
        private readonly Predicate<object?>? _canExecute;

        public DelegateCommand(Action<object?> execute,
            Predicate<object?>? canExecute = null)
        {
            _execute = execute;
            _canExecute = canExecute;
        }

        public bool CanExecute(object? parameter) => _canExecute
            == null || _canExecute.Invoke(parameter);

        public void Execute(object? parameter)

```

```

        {
            _execute?.Invoke(parameter);
        }

        public event EventHandler? CanExecuteChanged;

        public void RaiseCanExecuteChanged()
        {
            CanExecuteChanged?.Invoke(this, EventArgs.Empty);
        }
    }
}

```

Приложение В.6

Файл «ISynchronizationHelper.cs»

```

using System;
using System.Threading.Tasks;

namespace ExplorER
{
    public interface ISynchronizationHelper
    {
        Task InvokeAsync(Action action);
    }
}

```

Приложение В.7

Файл «NaturalSortComparer.cs»

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Text.RegularExpressions;

namespace ExplorER
{
    public class NaturalSortComparer : IComparer<string>,
        IDisposable, IComparer
    {
        #region Private Fields

        private readonly bool _isAscending;

        private Dictionary<string, string[]> _table = new();

        #endregion

        #region Constructor

```

```

public NaturalSortComparer(bool inAscendingOrder = true)
{
    this._isAscending = inAscendingOrder;
}

#endregion

#region Public Methods

public int Compare(string x, string y)
{
    if (x == y)
        return 0;

    if (!_table.TryGetValue(x, out var x1))
    {
        x1 = Regex.Split(x.Replace(" ", ""), "([0-9]+)");
        _table.Add(x, x1);
    }

    if (!_table.TryGetValue(y, out var y1))
    {
        y1 = Regex.Split(y.Replace(" ", ""), "([0-9]+)");
        _table.Add(y, y1);
    }

    int returnVal;

    for (int i = 0; i < x1.Length && i < y1.Length; i++)
    {
        if (x1[i] != y1[i])
        {
            returnVal = PartCompare(x1[i], y1[i]);
            return _isAscending ? returnVal : -
returnVal;
        }
    }

    if (y1.Length > x1.Length)
    {
        returnVal = 1;
    }
    else if (x1.Length > y1.Length)
    {
        returnVal = -1;
    }
    else
    {
        returnVal = 0;
    }
}

```

```

        return _isAscending ? returnVal : -returnVal;
    }

    public void Dispose()
    {
        _table.Clear();
        _table = null;
    }

#endregion

#region Private Methods

private static int PartCompare(string left, string
right)
{
    int x, y;
    if (!int.TryParse(left, out x))
        return left.CompareTo(right);

    if (!int.TryParse(right, out y))
        return left.CompareTo(right);

    return x.CompareTo(y);
}

#endregion

public int Compare(object? x, object? y)
    => Compare(x as string, y as string);
}
}

```

Приложение В.8

Файл «DirectoryHistory.cs»

```

using System;
using System.Collections;
using System.Collections.Generic;

namespace ExplorER
{
    internal class DirectoryHistory : IDirectoryHistory
    {
        #region Properties

        public bool CanMoveBack => Current.PreviousNode != null;
        public bool CanMoveForward => Current.NextNode != null;

        public DirectoryNode Current { get; private set; }

        #endregion
    }
}

```

```

#region Events

public event EventHandler HistoryChanged;

#endregion

#region Constructor

public DirectoryHistory(string directoryPath, string
directoryPathName)
{
    var head = new DirectoryNode(directoryPath,
directoryPathName);
    Current = head;
}

#endregion

#region Public Methods

public void MoveBack()
{
    var prev = Current.PreviousNode;

    Current = prev;

    RaiseHistoryChanged();
}

public void MoveForward()
{
    var next = Current.NextNode;

    Current = next;

    RaiseHistoryChanged();
}

public void Add(string filePath, string name)
{
    var node = new DirectoryNode(filePath, name);

    Current.NextNode = node;
    node.PreviousNode = Current;

    Current = node;

    RaiseHistoryChanged();
}

#endregion

```

```

        #region Private Methods

        private void RaiseHistoryChanged() =>
HistoryChanged?.Invoke(this, EventArgs.Empty);

        #endregion

        #region Enumerator

        public IEnumerator<DirectoryNode> GetEnumerator()
        {
            yield return Current;
        }

        IEnumerator IEnumerable.GetEnumerator() =>
GetEnumerator();

        #endregion
    }
}

```

Приложение В.9

Файл «DirectoryNode.cs»

```

namespace ExplorER
{
    internal class DirectoryNode
    {
        public DirectoryNode PreviousNode { get; set; }
        public DirectoryNode NextNode { get; set; }

        public string DirectoryPath { get; }
        public string DirectoryPathName { get; }

        public DirectoryNode(string directoryPath, string
directoryPathName)
        {
            DirectoryPath = directoryPath;
            DirectoryPathName = directoryPathName;
        }
    }
}

```

Приложение В.10

Файл «IDirectoryHistory.cs»

```

using System;
using System.Collections.Generic;

namespace ExplorER

```

```

{
    internal interface IDirectoryHistory :
IEnumerable<DirectoryNode>
    {
        bool CanMoveBack { get; }

        bool CanMoveForward { get; }

        DirectoryNode Current { get; }

        event EventHandler HistoryChanged;

        void MoveBack();

        void MoveForward();

        void Add(string filePath, string name);
    }
}

```

Приложение В.11

Файл «ExtensionToImageFileConverter.cs»

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace ExplorER
{
    internal class ExtensionToImageFileConverter
    {
        #region Private Fields

        private readonly Dictionary<string, FileInfo> _icons;

        #endregion

        #region Constructor

        public ExtensionToImageFileConverter()
        {
            var applicationDirectory =
AppDomain.CurrentDomain.BaseDirectory;

            var iconsDirectory = new
DirectoryInfo(Path.Combine(applicationDirectory, "Resources",
"Icons"));

            _icons = iconsDirectory
.GetFiles()

```

```

        .ToDictionary(fi =>
Path.GetFileNameWithoutExtension(fi.Name).ToUpper());
    }

#endregion

#region Public Methods

    /// <summary>
    /// Получение полного пути до иконки для заданного
расширения
    /// </summary>
    /// <param name="extension">Расширение в формате без
точки сначала</param>
    /// <returns></returns>
    public FileInfo GetImagePath(string extension)
    {
        if (_icons.ContainsKey(extension.ToUpper()))
            return _icons[extension.ToUpper()];

        return _icons[IconName.Blank.ToUpper()];
    }

#endregion
}
}

```

Приложение В.12

Файл «IconName.cs»

```

namespace ExplorER
{
    internal static class IconName
    {
        public const string Blank = "_blank";

        public const string Folder = "_folder";

        public const string BookmarkFolder = "_bookmark_folder";

        public const string LogicalDrive = "_logicalDrive";

        public const string SystemDrive = "_systemDrive";
    }
}

```

Приложение В.13

Файл «IconsManager.cs»

```

using System;

```

```

using System.IO;
using System.Runtime.InteropServices;
using ExplorER.FileEntities;
using ExplorER.FileEntities.Base;

namespace ExplorER
{
    internal class IconsManager : IIconsManager
    {
        private readonly ExtensionToImageFileConverter
_converter;

        public IconsManager(ExtensionToImageFileConverter
converter)
        {
            _converter = converter;
        }

        public FileInfo GetIconPath(FileEntityViewModel
viewModel)
        {
            if (viewModel is FileViewModel fileViewModel)
            {
                var extension = new
FileInfo(fileViewModel.FullName).Extension;

                var imagePath =
_converter.GetImagePath(string.IsNullOrEmpty(extension) ? "" :
extension.Substring(1));

                return imagePath;
            }

            if (viewModel is LogicalDriveViewModel
logicalDriveViewModel)
            {
                if
(RuntimeInformation.IsOSPlatform(OSPlatform.Windows))
                {
                    if (logicalDriveViewModel.FullName ==
"C:\\")
                        return
_converter.GetImagePath(IconName.SystemDrive);
                }

                return
_converter.GetImagePath(IconName.LogicalDrive);
            }

            if (viewModel is DirectoryViewModel)
            {
                return _converter.GetImagePath(IconName.Folder);
            }
        }
    }
}

```

```

        throw new NotImplementedException();
    }
}

public interface IIconsManager
{
    FileInfo GetIconPath(FileEntityViewModel viewModel);
}
}

```

Приложение B.14

Файл « BaseViewModel.cs»

```

using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace ExplorER.Base
{
    public class BaseViewModel : INotifyPropertyChanged
    {
        #region Events

        public event PropertyChangedEventHandler
        PropertyChanged;

        #endregion

        #region Protected Methods

        protected virtual void
        OnPropertyChanged([CallerMemberName] string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new
            PropertyChangedEventArgs(propertyName));
        }

        #endregion
    }
}

```

Приложение B.15

Файл « ChromerTabItemViewModel.cs»

```

namespace ExplorER.Base
{
    public abstract class ChromerTabItemViewModel :
    BaseViewModel
    {
        #region Public Properties

```

```

        public string Header { get; set; }

        public bool IsSelected { get; set; }

        public bool LogicalIndex { get; set; }

        public object Content { get; set; }

        #endregion
    }
}

```

Приложение В.16

Файл « FileEntityViewModel.cs»

```

using System;
using ExplorER.Base;

namespace ExplorER.FileEntities.Base
{
    public abstract class FileEntityViewModel : BaseViewModel
    {
        public string Name { get; set; }

        public string FullName { get; set; }

        public string Group { get; set; }

        public abstract DateTime ChangeDateTime { get; }

        protected FileEntityViewModel(string name)
        {
            Name = name;
        }

        public abstract string GetRootName();
    }
}

```

Приложение В.17

Файл « DirectoryViewModel.cs»

```

using System;
using System.IO;
using ExplorER.FileEntities.Base;

namespace ExplorER.FileEntities
{
    public class DirectoryViewModel : FileEntityViewModel
    {

```

```

{
    private readonly DirectoryInfo _directoryName;

    public DirectoryViewModel(DirectoryInfo directoryName) :
base(directoryName.Name)
    {
        _directoryName = directoryName;
        FullName = directoryName.FullName;
    }

    public override DateTime ChangeDateTime =>
_directoryName.LastWriteTime;

    public override string GetRootName()
        => new DirectoryInfo(FullName).Root.Name;
}
}

```

Приложение В.18

Файл « FileViewModel.cs»

```

using System;
using System.IO;
using ExplorER.FileEntities.Base;

namespace ExplorER.FileEntities
{
    public sealed class FileViewModel : FileEntityViewModel
    {
        private readonly FileInfo _fileInfo;

        public double Size => _fileInfo.Length / 1024.0;

        public FileViewModel(FileInfo fileInfo) :
base(fileInfo.Name)
        {
            _fileInfo = fileInfo;
            FullName = fileInfo.FullName;
        }

        public override DateTime ChangeDateTime =>
_fileInfo.LastWriteTime;

        public override string GetRootName()
            => new FileInfo(FullName).Directory?.Root.Name;
    }
}

```

Приложение В.19

Файл « LogicalDriveViewModel.cs»

```

using System.IO;

namespace ExplorER.FileEntities
{
    public sealed class LogicalDriveViewModel :
DirectoryViewModel
    {
        public LogicalDriveViewModel(string directoryName) :
base(new DirectoryInfo(directoryName))
        {
            FullName = directoryName;
        }

        public override string GetRootName()
            => FullName;
    }
}

```

Приложение B.20

Файл « IFilesPresenter.cs»

```

using System;

namespace ExplorER.FilePresenters.Base.Core
{
    public interface IFilesPresenter : IDisposable
    {
        string CurrentDirectoryPathName { get; }

        event EventHandler<OpenDirectoryEventArgs>
DirectoryOrFileOpened;
    }
}

```

Приложение B.21

Файл « OpenDirectoryEventArgs.cs»

```

using System;
using ExplorER.FileEntities.Base;

namespace ExplorER.FilePresenters.Base.Core
{
    public class OpenDirectoryEventArgs : EventArgs
    {
        public FileEntityViewModel FileEntityViewModel { get; }

        public OpenDirectoryEventArgs(FileEntityViewModel
fileEntityViewModel)
        {
            FileEntityViewModel = fileEntityViewModel;
        }
    }
}

```

```

    }
}
}

```

Приложение В.22

Файл « BaseFilesPresenter.cs»

```

using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using ExplorER.Base;
using ExplorER.FileEntities;
using ExplorER.FileEntities.Base;
using ExplorER.FilePresenters.Base.Core;

namespace ExplorER.FilePresenters.Base
{
    public abstract class BaseFilesPresenter : BaseViewModel,
        IFilesPresenter
    {
        #region Private Fields

        private readonly ISynchronizationHelper
        _synchronizationHelper;
        private readonly BackgroundWorker _backgroundWorker;

        #endregion

        #region Public Properties

        public ObservableCollection<FileEntityViewModel>
        DirectoriesAndFiles { get; set; } = new();

        public FileEntityViewModel? SelectedFileEntity { get;
        set; }

        public string CurrentDirectoryPathName { get; }

        #endregion

        #region Events

        public event EventHandler<OpenDirectoryEventArgs>?
        DirectoryOrFileOpened;

        #endregion

        #region Commands

        public DelegateCommand OpenCommand { get; }

```

```

        public DelegateCommand OpenNewTabCommand { get; }
        public DelegateCommand OpenNewWindowCommand =>
ExplorerEr.Instance.OpenNewWindowCommand;

    #endregion

    #region Constructor

        protected BaseFilesPresenter(DirectoryTabItemViewModel
directoryTabItemView, string directoryPathName)
        {
            _synchronizationHelper =
directoryTabItemView.SynchronizationHelper;

            CurrentDirectoryPathName = directoryPathName;

            _backgroundWorker = new BackgroundWorker
            {
                WorkerSupportsCancellation = true,
                WorkerReportsProgress = true
            };

            OpenCommand = new DelegateCommand(Open);
            OpenNewTabCommand = new DelegateCommand(OpenNewTab);

            _backgroundWorker.RunWorkerCompleted +=
BackgroundWorker_RunWorkerCompleted;
            _backgroundWorker.DoWork += BackgroundWorker_DoWork;
            _backgroundWorker.ProgressChanged +=
BackgroundWorker_ProgressChanged;

            _backgroundWorker.RunWorkerAsync();
        }

    #endregion

    #region Public Methods

        public void Dispose()
        {
            if (_backgroundWorker.IsBusy)
                _backgroundWorker.CancelAsync();

            _backgroundWorker.RunWorkerCompleted -=
BackgroundWorker_RunWorkerCompleted;
            _backgroundWorker.DoWork -= BackgroundWorker_DoWork;
            _backgroundWorker.ProgressChanged -=
BackgroundWorker_ProgressChanged;

            _backgroundWorker?.Dispose();
        }

    #endregion

```

```

#region Command Methods

private void Open(object parameter)
{
    if (parameter is FileEntityViewModel
fileEntityViewModel)
        DirectoryOrFileOpened?.Invoke(this, new
OpenDirectoryEventArgs(fileEntityViewModel));
}

private void OpenNewTab(object parameter)
{
    if (parameter is object[] {Length: 2} parameters &&
        parameters[0] is TabsViewModel mainViewModel &&
        parameters[1] is FileEntityViewModel
fileEntityViewModel)
    {
        mainViewModel.OnOpenNewTab(fileEntityViewModel);
    }
}

#endregion

#region Private Methods

private void BackgroundWorker_RunWorkerCompleted(object?
sender, RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled)
    {
        _backgroundWorker.RunWorkerAsync();
    }
}

private void BackgroundWorker_ProgressChanged(object?
sender, ProgressChangedEventArgs e)
{
}

private async void BackgroundWorker_DoWork(object?
sender, DoWorkEventArgs e)
{
    if (CurrentDirectoryPathName == ExplorerEr.RootName)
    {
        var group = "Папки";

        var specialFolders = new[]
        {
            Environment.GetFolderPath(Environment.SpecialFolder.MyVideos),

```

```

Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
,
Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
Environment.GetFolderPath(Environment.SpecialFolder.MyPictures),
Environment.GetFolderPath(Environment.SpecialFolder.MyMusic)
    };

    foreach (var specialFolder in specialFolders)
    {
        await _synchronizationHelper.InvokeAsync(()
=>
        {
            DirectoriesAndFiles.Add(new
DirectoryViewModel(new DirectoryInfo(specialFolder))
            {
                Group = group
            });
        });
    }

    group = "Устройства и диски";

    foreach (var logicalDrive in
Directory.GetLogicalDrives())
    {
        await _synchronizationHelper.InvokeAsync(()
=>
        {
            DirectoriesAndFiles.Add(new
LogicalDriveViewModel(logicalDrive)
            {
                Group = group
            });
        });
    }

    return;
}

var directoryInfo = new
DirectoryInfo(CurrentDirectoryPathName);

var bw = sender as BackgroundWorker;

try
{
    var directories =
directoryInfo.EnumerateDirectories()

```

```

        .OrderBy(d => d.Name, new
NaturalSortComparer());

        foreach (var directory in directories)
        {
            if (bw.CancellationPending)
            {
                e.Cancel = true;

                return;
            }

            await _synchronizationHelper.InvokeAsync(()

=>
            {
                DirectoriesAndFiles.Add(new
DirectoryViewModel(directory));
            });
        }

        foreach (var fileInfo in
directoryInfo.EnumerateFiles().OrderBy(f => f.Name, new
NaturalSortComparer()))
        {
            if (bw.CancellationPending)
            {
                e.Cancel = true;

                return;
            }

            await _synchronizationHelper.InvokeAsync(()

=>
            {
                DirectoriesAndFiles.Add(new
FileViewModel(fileInfo));
            });
        }
    }
    catch (Exception ex)
    {
        //TODO: Try Exception
    }
}

#endregion
}
}

```

Приложение В.23
Файл « GridFilesPresenterViewModel.cs»

```

using ExplorerER.FilePresenters.Base;

namespace ExplorerER.FilePresenters
{
    public class GridFilesPresenterViewModel :
BaseFilesPresenter
    {
        public
GridFilesPresenterViewModel(DirectoryTabItemViewModel
directoryTabItemView, string directoryPathName) :
        base(directoryTabItemView, directoryPathName)
        {
        }
    }
}

```

Приложение В.24

Файл « TileFilesPresenterViewModel.cs»

```

using ExplorerER.FilePresenters.Base;

namespace ExplorerER.FilePresenters
{
    public class TileFilesPresenterViewModel :
BaseFilesPresenter
    {
        public
TileFilesPresenterViewModel(DirectoryTabItemViewModel
directoryTabItemView, string directoryPathName) :
        base(directoryTabItemView, directoryPathName)
        {
        }
    }
}

```

Приложение В.25

Файл « DirectoryTabItemViewModel.cs»

```

using System;
using System.Diagnostics;
using System.IO;
using ExplorerER.Base;
using ExplorerER.FileEntities;
using ExplorerER.FileEntities.Base;
using ExplorerER.FilePresenters;
using ExplorerER.FilePresenters.Base.Core;

namespace ExplorerER
{

```

```

    public class DirectoryTabItemViewModel :
ChromerTabItemViewModel
    {
        #region Private Fields

        private readonly IDirectoryHistory _history;

        private string _searchText;
        private bool _isTilePresenter;
        private bool _isGridPresenter;

        #endregion

        #region Public Properties

        public ISynchronizationHelper SynchronizationHelper {
get; }

        public bool IsTilePresenter
        {
            get => _isTilePresenter;
            set
            {
                _isTilePresenter = value;

                if (value)
                {
                    IsGridPresenter = false;
                    OpenDirectory();
                }
                else if (!_isGridPresenter)
                {
                    IsTilePresenter = true;
                }
            }
        }

        public bool IsGridPresenter
        {
            get => _isGridPresenter;
            set
            {
                _isGridPresenter = value;

                if (value)
                {
                    IsTilePresenter = false;
                    OpenDirectory();
                }
                else if (!_isTilePresenter)
                {
                    IsGridPresenter = true;
                }
            }
        }
    }

```

```

        }
    }

    public string SearchText
    {
        get => _searchText;
        set => SetSearchText(value);
    }

    public string CurrentDirectoryFileName =>
_history.Current.DirectoryPath;

    public IFilesPresenter? FilesPresenter { get; set; }

#endregion

#region Commands

    public DelegateCommand AddBookmarkCommand =>
ExplorerEr.Instance.BookmarksManager.AddBookmarkCommand;

    public DelegateCommand MoveBackCommand { get; }

    public DelegateCommand MoveForwardCommand { get; }

#endregion

#region Constructor

    public DirectoryTabItemViewModel(ISynchronizationHelper
synchronizationHelper,
        string directoryPath,
        string directoryName)
    {
        SynchronizationHelper = synchronizationHelper;

        _history = new DirectoryHistory(directoryPath,
directoryName);

        MoveBackCommand = new DelegateCommand(OnMoveBack,
OnCanMoveBack);
        MoveForwardCommand = new
DelegateCommand(OnMoveForward, OnCanMoveForward);

        Header = _history.Current.DirectoryPathName;
        _searchText = _history.Current.DirectoryPath;

        IsTilePresenter = true;

        _history.HistoryChanged += History_HistoryChanged;
    }

```

```

#endregion

#region Public Methods

public void OpenBookmark(string path)
{
    var attr = File.GetAttributes(path);

    if (attr.HasFlag(FileAttributes.Directory))
        Open(new DirectoryViewModel(new
DirectoryInfo(path)));
    else
        Open(new FileViewModel(new FileInfo(path)));
}

#endregion

#region Commands Methods

private void Open(FileEntityViewModel parameter)
{
    if (parameter is DirectoryViewModel
directoryViewModel)
    {
        SearchText = directoryViewModel.FullName;
        Header = directoryViewModel.Name;

        _history.Add(SearchText, Header);

        OpenDirectory();
    }
    else if (parameter is FileViewModel fileViewModel)
    {
        new Process
        {
            StartInfo = new
ProcessStartInfo(fileViewModel.FullName)
            {
                UseShellExecute = true
            }
        }.Start();
    }
}

private bool OnCanMoveForward(object obj) =>
_history.CanMoveForward;

private void OnMoveForward(object obj)
{
    _history.MoveForward();

    var current = _history.Current;

```

```

        SearchText = current.DirectoryPath;
        Header = current.DirectoryPathName;

        OpenDirectory();
    }

    private bool OnCanMoveBack(object obj) =>
_history.CanMoveBack;

    private void OnMoveBack(object obj)
    {
        _history.MoveBack();

        var current = _history.Current;

        SearchText = current.DirectoryPath;
        Header = current.DirectoryPathName;

        OpenDirectory();
    }

#endregion

#region Private Methods

    private void OpenDirectory()
    {
        if (FilesPresenter != null)
        {
            FilesPresenter.DirectoryOrFileOpened -=
FilePresenterOnDirectoryOrFileOpened;
            FilesPresenter.Dispose();
        }

        FilesPresenter = _isGridPresenter
            ? new GridFilesPresenterViewModel(this,
CurrentDirectoryFileName)
            : new TileFilesPresenterViewModel(this,
CurrentDirectoryFileName);

        FilesPresenter.DirectoryOrFileOpened +=
FilePresenterOnDirectoryOrFileOpened;
    }

    private void
FilePresenterOnDirectoryOrFileOpened(object? sender,
OpenDirectoryEventArgs e)
    {
        Open(e.FileEntityViewModel);
    }

    private void History_HistoryChanged(object? sender,
EventArgs e)

```

```

        {
            MoveBackCommand?.RaiseCanExecuteChanged();
            MoveForwardCommand?.RaiseCanExecuteChanged();
        }

        private void SetSearchText(string text)
        {
            _searchText = text;
        }

        #endregion
    }
}

```

Приложение В.26

Файл « TabsViewModel.cs »

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.Linq;
using ExplorerER.Base;
using ExplorerER.FileEntities;
using ExplorerER.FileEntities.Base;

namespace ExplorerER
{
    public class TabsViewModel : BaseViewModel
    {
        #region Private Fields

        private readonly ISynchronizationHelper
        _synchronizationHelper;

        #endregion

        #region Public Properties

        public ITabClient InterTabClient { get; }

        public ObservableCollection<ChromerTabItemViewModel>
        TabItems { get; }

        public ChromerTabItemViewModel? CurrentTabItem { get;
        set; }

        public IReadOnlyCollection<MenuItemViewModel> Bookmarks
        => ExplorerEr.Instance.BookmarksManager.Bookmarks;

        public Func<DirectoryTabItemViewModel> Factory { get; }

```

```

        #endregion

        #region Commands

        public DelegateCommand CreateNewTabItemCommand { get; }
        public DelegateCommand OpenTabItemInNewWindowCommand {
get; }
        public DelegateCommand DuplicateTabCommand { get; }
        public DelegateCommand CloseAllTabsCommand { get; }

        #endregion

        #region Constructor

        public TabsViewModel(ISynchronizationHelper
synchronizationHelper, ITabClient tabClient,
            IEnumerable<DirectoryTabItemViewModel> init)
        {
            _synchronizationHelper = synchronizationHelper;
            InterTabClient = tabClient;

            CreateNewTabItemCommand = new
DelegateCommand(OnCreateNewTabItem);
            OpenTabItemInNewWindowCommand = new
DelegateCommand(OnOpenTabItemInNewWindow,
OnCanOpenTabItemInNewWindow);
            DuplicateTabCommand = new
DelegateCommand(OnDuplicate);
            CloseAllTabsCommand = new
DelegateCommand(OnCloseAllTabs, CanCloseAllTabs);

            TabItems = new
ObservableCollection<ChromerTabItemViewModel>(init);

            Factory = CreateTabVm;

            TabItems.CollectionChanged +=
TabItemsOnCollectionChanged;
        }

        #endregion

        #region Public Methods

        public void OnOpenNewTab(FileEntityViewModel
fileEntityViewModel, bool isSelectNewTab = false)
        {
            if (fileEntityViewModel is DirectoryViewModel
directoryViewModel)
            {
                var tab = new
DirectoryTabItemViewModel(_synchronizationHelper,
directoryViewModel.FullName,

```

```

        directoryViewModel.Name);
        TabItems.Add(tab);

        if (isSelectedNewTab)
            CurrentTabItem = tab;
    }
}

private void OnCreateNewTabItem(object? obj)
{
    if (obj is not DirectoryTabItemViewModel
directoryTabItem)
        return;

    var tab = new
DirectoryTabItemViewModel(_synchronizationHelper,
        ExplorerEr.RootName, ExplorerEr.RootName);
    TabItems.Add(tab);
}

private bool OnCanOpenTabItemInNewWindow(object? obj) =>
TabItems.Count > 1;

private void OnOpenTabItemInNewWindow(object? obj)
{
    if (obj is not DirectoryTabItemViewModel
directoryTabItem)
        return;

    TabItems.Remove(directoryTabItem);

    ExplorerEr.Instance.OpenTabInNewWindow(directoryTabItem);
}

private void OnDuplicate(object? obj)
{
    if (obj is not DirectoryTabItemViewModel
directoryTabItem)
        return;

    TabItems.Add(new
DirectoryTabItemViewModel(_synchronizationHelper,
        directoryTabItem.CurrentDirectoryFileName,
directoryTabItem.Header));
}

private bool CanCloseAllTabs(object? obj) =>
TabItems.Count > 1;

private void OnCloseAllTabs(object? obj)
{

```

```

        if (obj is not DirectoryTabItemViewModel
directoryTabItem)
            return;

        var removedItems = TabItems.Where(i => i !=
directoryTabItem).ToList();

        foreach (var item in removedItems)
            TabItems.Remove(item);
    }

#endregion

#region Private Methods

    private DirectoryTabItemViewModel CreateTabVm() =>
        new(_synchronizationHelper, ExplorerEr.RootName,
ExplorerEr.RootName);

    private void TabItemsOnCollectionChanged(object? sender,
NotifyCollectionChangedEventArgs e)
    {
        OpenTabItemInNewWindowCommand.RaiseCanExecuteChanged();
        CloseAllTabsCommand.RaiseCanExecuteChanged();
    }

#endregion
}
}

```

Приложение В.27

Файл « ExplorerEr.cs»

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using ExplorerER.FileEntities;
using ExplorerER.FileEntities.Base;

namespace ExplorerER
{
    /// <summary>
    /// Основное приложение проводника chromER
    /// </summary>
    public sealed class ExplorerEr
    {
        private readonly ISynchronizationHelper
_synchronizationHelper;
        private readonly ITabClient _tabClient;
    }
}

```

```

        private readonly Action<TabsViewModel, Point>
_windowFactory;

        #region Singleton

        public static ExplorerEr Instance { get; private set; }

        public static void CreateChromer(ISynchronizationHelper
helper, ITabClient tabClient,
        Action<TabsViewModel, Point> windowFactory)
        {
            if (Instance == null)
            {
                CultureInfo currentCulture = new("Ru-ru");
                CultureInfo.CurrentCulture = currentCulture;
                CultureInfo.CurrentUICulture = currentCulture;

                Instance = new ExplorerEr(helper, tabClient,
windowFactory);
            }
        }

        public static string RootName = "Мой компьютер";

        #endregion

        #region Public Properties

        //public TabsViewModel TabsViewModel { get; }

        /// <summary>
        /// Менеджер иконок
        /// </summary>
        public IIconsManager IconsManager { get; }

        /// <summary>
        /// Менеджер закладок
        /// </summary>
        public IBookmarksManager BookmarksManager { get; }

        public DelegateCommand OpenNewWindowCommand { get; }

        #endregion

        #region Constructor

        private ExplorerEr(ISynchronizationHelper
synchronizationHelper,
            ITabClient tabClient,
            Action<TabsViewModel, Point> windowFactory)
        {
            _synchronizationHelper = synchronizationHelper;
            _tabClient = tabClient;

```

```

        _windowFactory = windowFactory;

        var converter = new ExtensionToImageFileConverter();

        OpenNewWindowCommand = new
DelegateCommand(OnOpenNewWindow);

        IconsManager = new IconsManager(converter);
        BookmarksManager = new BookmarksManager(converter);
    }

    #endregion

    public TabsViewModel
CreateMainViewModel(IEnumerable<DirectoryTabItemViewModel>
initItems)
        => new(_synchronizationHelper, _tabClient,
initItems);

    public void OpenTabInNewWindow(DirectoryTabItemViewModel
directoryTabItemView)
    {
        var mainViewModel = CreateMainViewModel(new
DirectoryTabItemViewModel[0]);

        mainViewModel.TabItems.Add(directoryTabItemView);

        _windowFactory.Invoke(mainViewModel, new Point(24,
24));
    }

    private void OnOpenNewWindow(object parameter)
    {
        if (parameter is FileEntityViewModel
fileEntityViewModel)
        {
            if (fileEntityViewModel is DirectoryViewModel
directoryViewModel)
            {
                var myCompTabVm = new
DirectoryTabItemViewModel(
                    _synchronizationHelper,
                    directoryViewModel.FullName,
                    directoryViewModel.Name);
                OpenTabInNewWindow(myCompTabVm);
            }
        }
    }

    public interface ITabClient
    {
    }

```

