СМК Ф 7.5.0-01-33

|  |
| --- |
| минобрнауки россии  федеральное государственное бюджетное образовательное учреждение  высшего образования  «ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» |

|  |  |
| --- | --- |
| Институт (факультет) | Институт информационных технологий |
| Кафедра | Математического и программного обеспечения ЭВМ |

**КУРСОВАЯ РАБОТА**

|  |
| --- |
| по дисциплине C#-программирование |

|  |  |
| --- | --- |
| на тему | Разработка программного обеспечения на языке C# |

|  |
| --- |
| Выполнил студент группы 1ПИб-01-1оп-21 |
| *группа* |
| направления подготовки (специальности) |
| 09.03.04 Программная инженерия |
| *шифр, наименование* |
| Сарычева Екатерина Васильевна |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Шаханов Никита Иванович |
| *фамилия, имя, отчество* |
| Доцент, кандидат технических наук |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_\_\_\_\_» \_\_\_ 20\_\_ г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2024 *Год*

# Введение

Темой данной курсовой работы является разработка программного обеспечения на языке программирования C#. В рамках этой работы требуется создать программу с пользовательским интерфейсом, используя технологии WPF (Windows Presentation Foundation), а также применить фреймворк Prism для улучшения архитектуры приложения.

C# является одним из наиболее перспективных языков программирования, поскольку он обладает богатым набором синтаксических конструкций, упрощающих написание кода. Кроме того, C# поддерживает объектно-ориентированный подход, что делает его универсальным инструментом для создания различных типов приложений.

Несмотря на то, что некоторые аспекты синтаксиса C# могут быть не самыми эффективными с точки зрения производительности, этот язык отличается простотой чтения и разработки кода. Важным преимуществом C# является наличие множества библиотек и шаблонов проектирования, что позволяет разработчикам экономить время на написание кода.

Оглавление

[Введение 2](#_Toc168654883)

[1. Выбор и описание предметной области 4](#_Toc168654884)

[2. Обзор основных частей Фреймворка Prism 5](#_Toc168654885)

[2.1. MVVM 5](#_Toc168654886)

[2.2. Модули 6](#_Toc168654887)

[2.3. Навигация 6](#_Toc168654888)

[3. Разработка WPF-приложения 7](#_Toc168654889)

[3.1. Разработка диаграммы классов и диаграммы пакетов с учетом инверсии управления (IoC) 7](#_Toc168654890)

[3.2. Построение оболочки с пользовательским интерфейсом и описанием регионов (представления) 8](#_Toc168654891)

[3.3. Проектирование независимых модулей (модели и модели-представления) 10](#_Toc168654892)

[3.4. Описание функционала кнопок 11](#_Toc168654893)

[Заключение 16](#_Toc168654894)

[Список литературы: 17](#_Toc168654895)

[Приложение 1 18](#_Toc168654896)

[Приложение 2. Руководство пользователя 22](#_Toc168654897)

# Выбор и описание предметной области

**Windows Presentation Foundation (WPF)** — это система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML [1].

WPF используется для создания разнообразных приложений, таких как графические редакторы, системы управления базами данных, игры и многие другие. WPF позволяет разработчикам создавать интерфейсы, которые выглядят современно, легко адаптируются под разные размеры экранов и поддерживают анимацию и другие визуальные эффекты.

WPF использует концепцию "разделения интерфейса и логики", позволяя разработчикам отделить визуальное представление от бизнес-логики, что облегчает поддержку и модификацию приложений [2].

Достоинства WPF:

1. Мощная визуализация: WPF предоставляет широкие возможности для создания красивых и интерактивных пользовательских интерфейсов.

2. Адаптивность: Приложения легко адаптируются под разные разрешения экранов и размеры окон.

3. Богатая графика и анимация: WPF позволяет создавать сложные графические элементы и анимацию, что делает пользовательский интерфейс более привлекательным и информативным.

4. Гибкость: WPF предоставляет разработчикам широкие возможности для настройки и расширения функционала приложений.

5. Интеграция с другими технологиями: WPF легко интегрируется с другими технологиями .NET, что делает его удобным выбором для разработки приложений в экосистеме Microsoft.

WPF использует язык разметки XAML (eXtensible Application Markup Language), который позволяет разработчикам описывать пользовательский интерфейс и его взаимодействие с логикой приложения. Этот язык позволяет легко создавать иерархию элементов интерфейса, их внешний вид и поведение.

XAML представляет собой XML-подобный язык, который описывает структуру элементов интерфейса и их свойства. Разработчики могут использовать XAML для создания иерархии элементов интерфейса, устанавливая для них атрибуты, стили, шаблоны и другие параметры.

XAML связывается с кодом приложения с помощью механизма привязок данных и событий. Это позволяет разработчикам разделять визуальное представление интерфейса от его логики, что упрощает сопровождение и модификацию приложений [3].

Достоинства:

1. Декларативный синтаксис: XAML позволяет описывать интерфейс приложения в виде декларативного кода, что делает его более понятным и удобным для работы.

2. Разделение интерфейса и логики: Использование XAML позволяет разделять визуальное представление приложения от его логики, облегчая поддержку и модификацию.

3. Интеграция с различными технологиями: XAML легко интегрируется с другими технологиями Microsoft, такими как WPF, что делает его универсальным средством разработки для Windows-приложений.

# Обзор основных частей Фреймворка Prism

**Prism — это фреймворк для создания слабосвязанных, ремонтопригодных и тестируемых XAML-приложений в WPF, Xamarin Forms, платформе Uno и WinUI [4].**

Prism способствует повторному использованию кода, позволяя легко разрабатывать, тестировать и интегрировать компоненты и сервисы в одно или несколько приложений. Так же помогает создать приложения, которые легко расширить, управляя зависимостями компонентов, позволяя им легко интегрироваться или заменяться альтернативными реализациями во время выполнения, а также обеспечивая возможность разбить приложение на модули, которые могут быть независимо обновлены и развёрнуты.

Prism помогает создать гибкие приложения, позволяя им легко обновляться по мере разработки новых возможностей.  Поощряет командную разработку, позволяя разрабатывать и даже развёртывать различные части приложения по отдельности.

# MVVM

**MVVM (Model-View-ViewModel) — это архитектурный шаблон, используемый в разработке программного обеспечения для разделения пользовательского интерфейса (UI) от бизнес-логики и данных [5]**.

**Компоненты MVVM:**

1. **Model (Модель)** — отвечает за представление данных и бизнес-логику приложения. Может включать в себя операции с данными, хранение информации и управление состоянием приложения.
2. View (Представление) — отвечает за отображение данных и взаимодействие с пользователем. Это компонент, с которым пользователь взаимодействует, и он визуализирует данные, предоставляемые ViewModel.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

1. **ViewModel (Модель-представления)** — служит посредником между Model и View. ViewModel преобразует данные из Model в формат, который может быть легко отображён в View, и обрабатывает пользовательские действия, перенаправляя их в Model. ViewModel также позволяет реализовать биндинг (связывание) данных между Model и View.

Такое разделение позволяет ускорить разработку и поддерживаемость программы — можно менять один компонент, не затрагивая код другого.

# Модули

Модули в Prism играют ключевую роль в организации приложения на платформе WPF, Silverlight и UWP.

Модули позволяют разделить приложение на логические блоки, что делает код более организованным и легким для поддержки. Они упрощают работу в команде, поскольку разработчики могут работать над разными модулями независимо друг от друга. Модульная структура обеспечивает гибкость и возможность легкого добавления или удаления функциональности приложения.

Модули в Prism представляют собой наборы компонентов (классов, представлений и т. д.), которые связаны с конкретной функциональностью или задачей приложения.

Даже учитывая, что модули должны быть слабо связаны, им свойственно обмениваться друг с другом данными и сообщениями. Есть несколько коммуникационных паттернов для слабо связанных систем, каждый со своими преимуществами и недостатками. Как правило, в конечном счёте, используются комбинации этих паттернов.

# Навигация

Термин «навигация» определён, как процесс, в котором приложение координирует изменения UI в ответ на жесты пользователя, или изменения внутреннего состояния приложения.

Prism также предоставляет набор инструментов для управления навигацией в приложении.

В Prism навигация осуществляется с помощью навигационного сервиса, который предоставляет различные методы для перехода между страницами (View) в приложении [6]. Вот несколько ключевых концепций навигации в Prism:

* NavigationService в Prism предоставляет методы для навигации между страницами. Он позволяет осуществлять навигацию как с параметрами, так и без них.
* Region Navigation поддерживает концепцию регионов, которая позволяет динамически загружать и отображать представления в различных областях приложения.

Регионы (Regions) в Prism представляют собой концепцию, которая позволяет динамически загружать и отображать представления в различных областях приложения. Использование регионов позволяет разделить пользовательский интерфейс на отдельные области, в которых могут отображаться различные представления.

Использование регионов в Prism помогает создавать более гибкие и масштабируемые пользовательские интерфейсы, позволяя легко управлять содержимым различных областей приложения. Они способствуют разделению логики и представления, что делает код более модульным и легким для поддержки и расширения.

# Разработка WPF-приложения

# Разработка диаграммы классов и диаграммы пакетов с учетом инверсии управления (IoC)

Диаграмма пакетов WPF-приложения представлена на рис.1. На ней изображены все пакеты, включая основной – AutomaticMarkup.

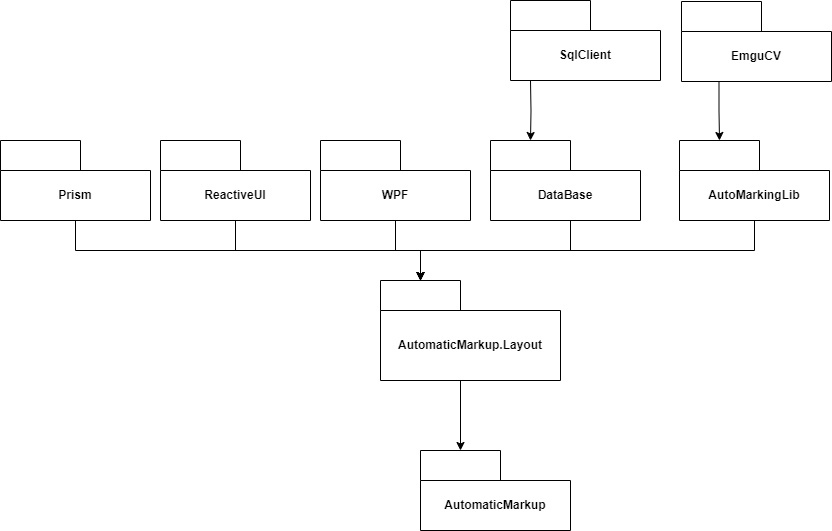


Рис.1. Диаграмма пакетов

Диаграмма классов модуля AutomaticMarkup.Layout представлена на рис.2.

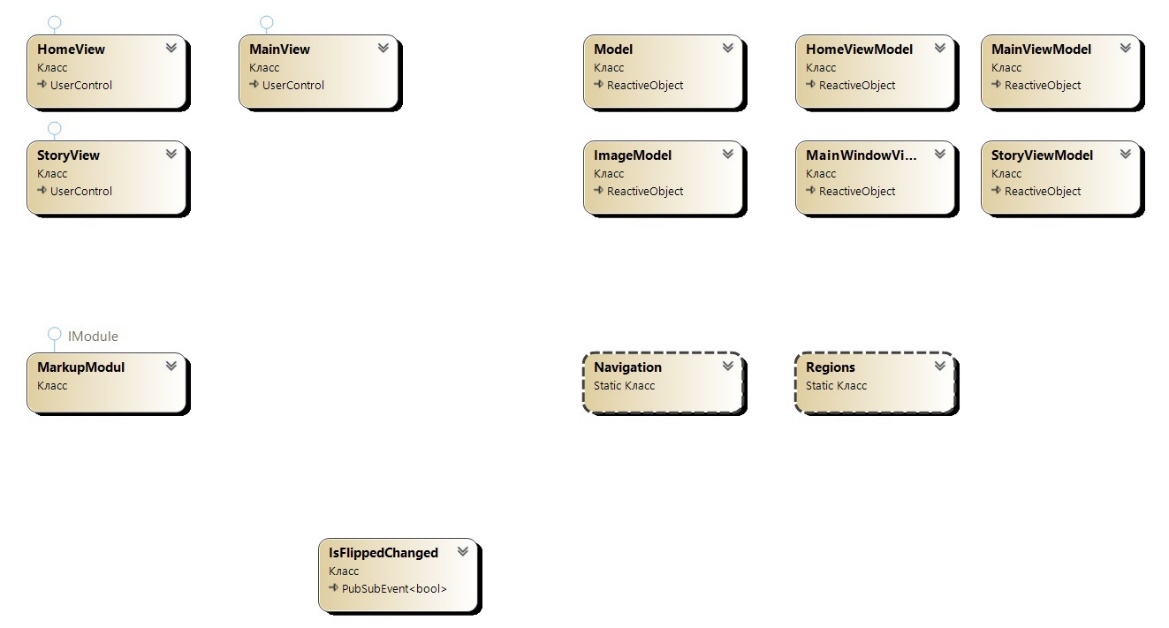


Рис.2. Диаграмма классов

# Построение оболочки с пользовательским интерфейсом и описанием регионов (представления)

Построение оболочки с пользовательским интерфейсом происходит в пакете AutomaticMarkup. В классе MainWindow записываются два региона: MenuRegion, HomeRegion. Они взяты из пакета AutomaticMarkup.Layout.

Регионы MenuRegion, HomeRegion являются основными. В регионе HomeRegion располагаются три элемента Image, в которые будут загружаться изображения необходимые для автоматической разметки.

В регионе MenuRegion находятся кнопки, с помощь которых происходит работа с приложением. Кнопка «Upload» вызывает диалоговое окно загрузки изображений с устройства. Кнопка «MarkUp» осуществляет разметку оригинального изображения используя маску. Кнопка «Save» сохраняет размеченное изображение и файл с координатами разметки. Кнопка «History» осуществляет смену региона HomeRegion на регион SRegion, в котором расположена база данных. Кнопка «Undo» возвращает HomeRegion.

С кодом разметки более подробно можно ознакомиться в приложении 3.

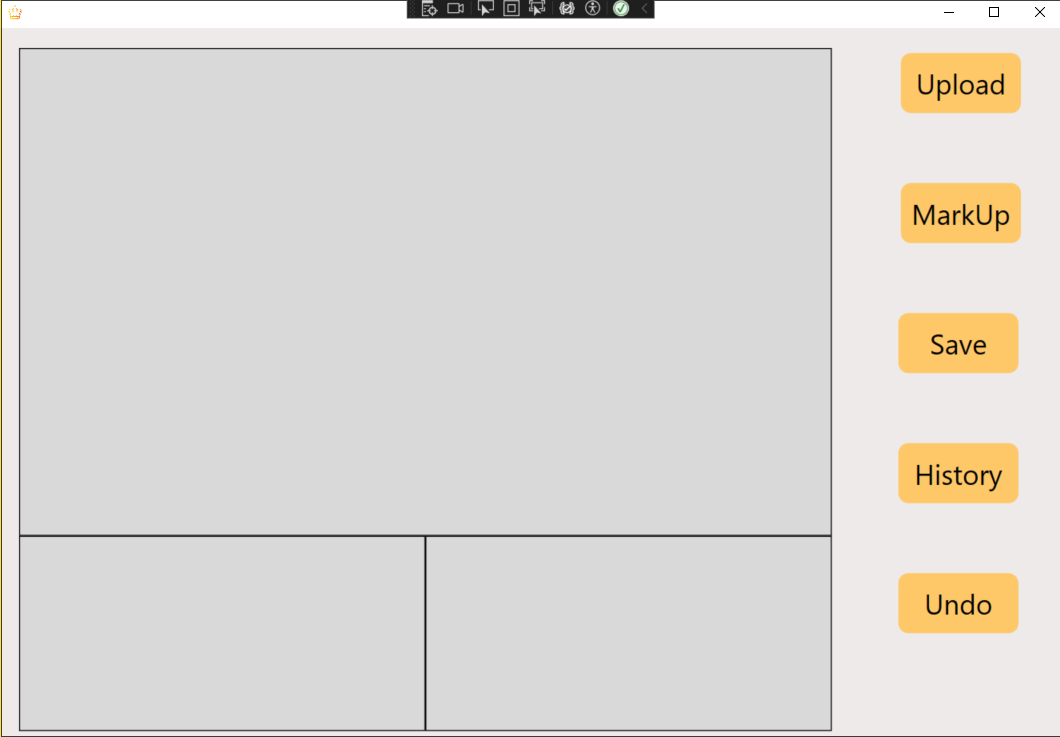


Рис.4. Графический интерфейс MenuRegion

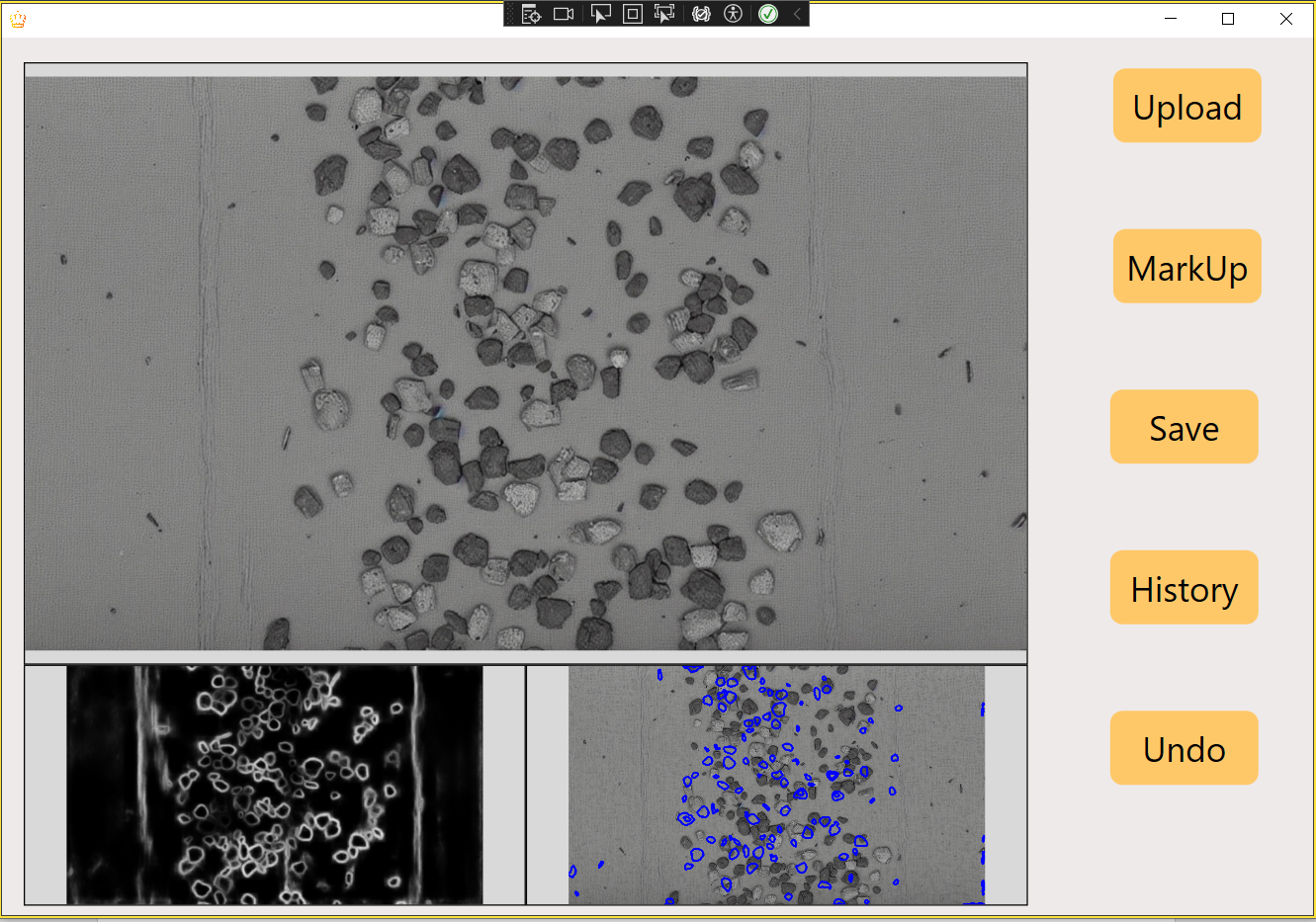


Рис. 5. Графический интерфейс HomeRegion

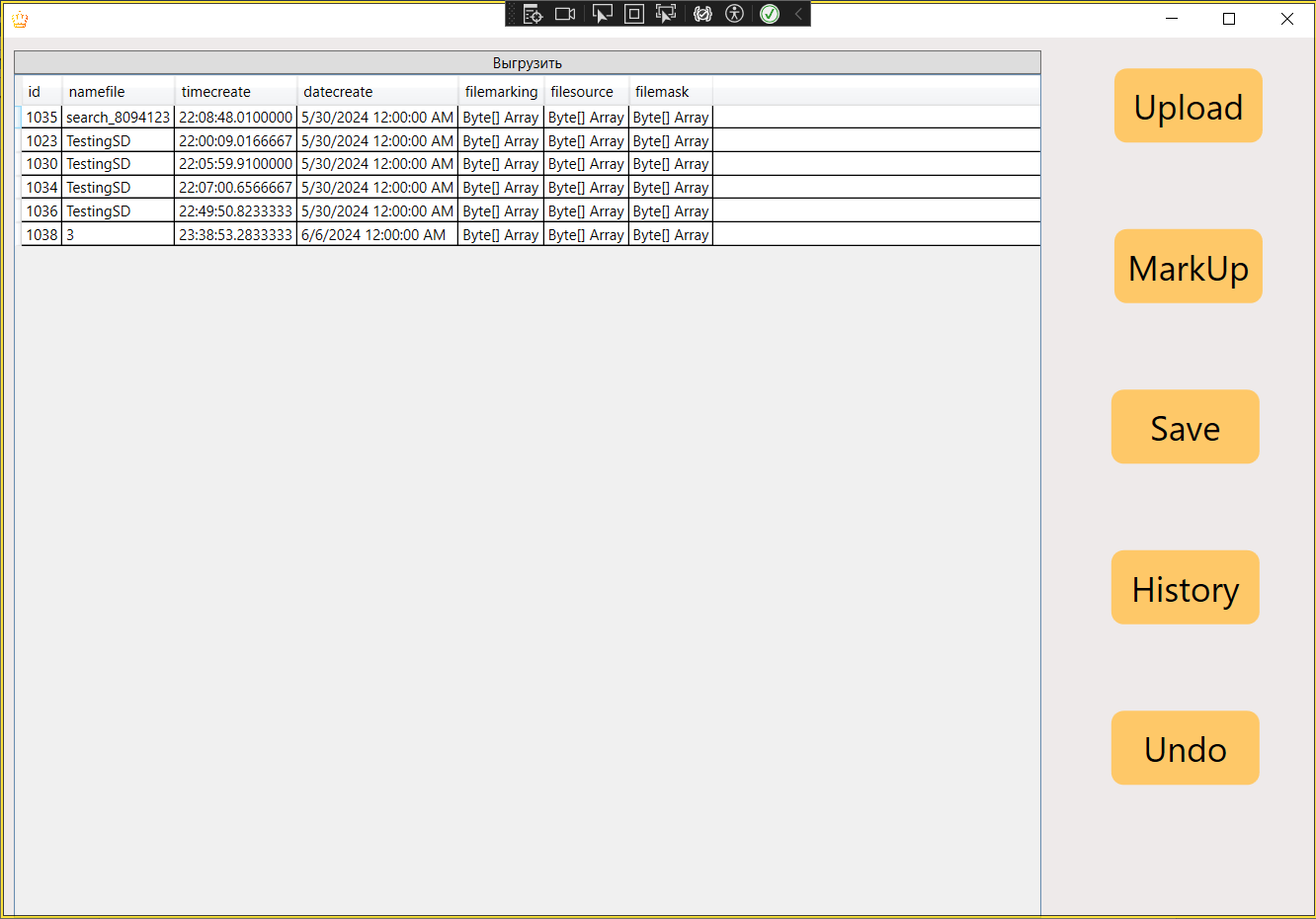


Рис.6. Графический интерфейс SRegion

# Проектирование моделей-представления

Программа состоит из 4 модулей. Основной – AutomaticMarkup, в нем находится основное окно приложения, и модуль AutomaticMarkup.Layout, в котором находятся четыре модели-представления (рис.7).

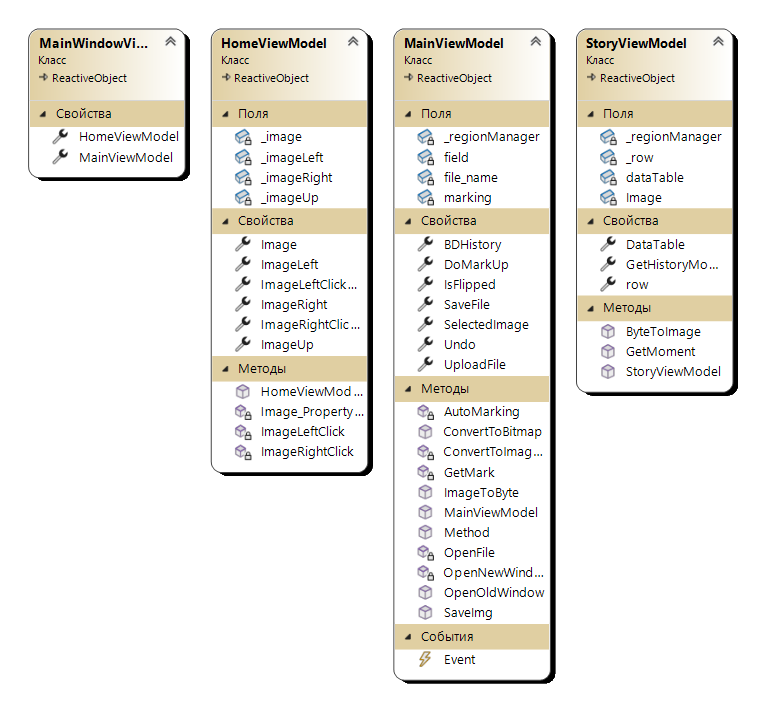


Рис.7. Модели представления

MainWindowViewModel контролирует работу окон, MainViewModel реализует функционал кнопок приложения, HomeViewModel реализует работу с элементами Image, StoryViewModel реализует работу с элементом DataGrig.

# Описание функционала кнопок

Для создания смены регионов необходимо зарегистрировать представления в модуле (рис.8).



Рис.8. Регистрация представлений

После регистрации инициализируются команды для смены представлений в MainViewModel. Пример одной из таких команд представлен на изображении (рис.9-10).

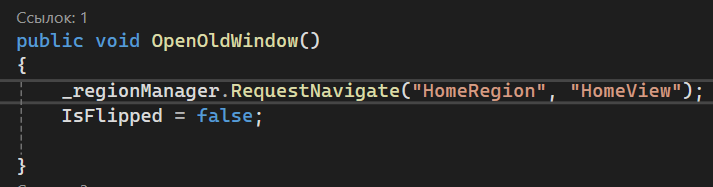


Рис.9. Создание команды OpenOldWindow

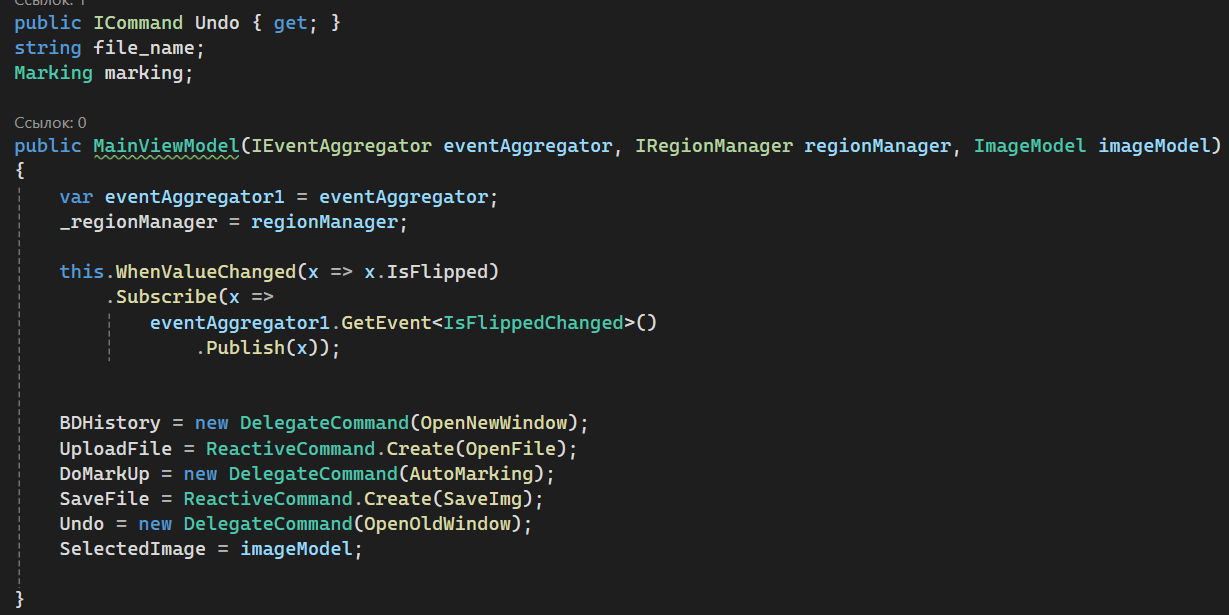


Рис.10. Инициализация команда смены представлений

Смена представлений применяется к кнопкам «Undo» и «History». Пример кнопки «Undo», с функцией смены представления с привязкой к команде Undo, представлен на рисунке ниже (рис.11). Кнопка «History» имеет точно такую же привязку, меняется только само представление, на которое нужно поменять текущее.

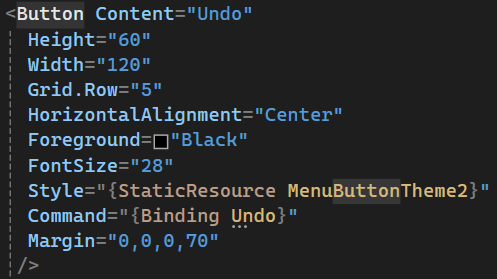


Рис.11. Привязка к кнопке

Кнопка «Upload» осуществляет запуск диалогового окна для выбора файла определенного формата для загрузки в элемент Image (рис.12-13).

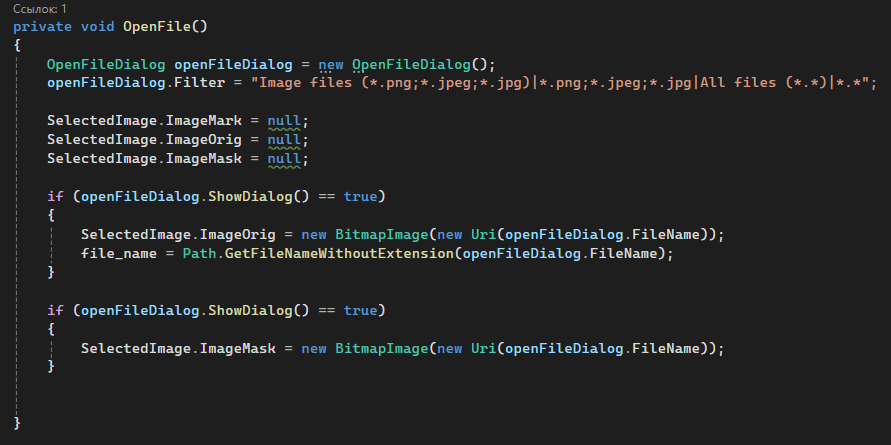


Рис.12. Создание команды OpenFile

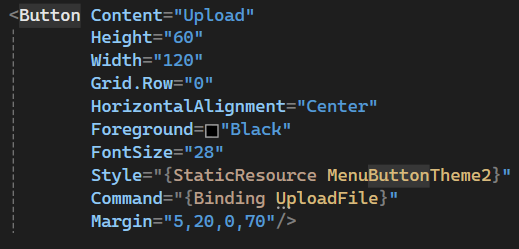


Рис.13. Привязка открытия диалогового окна к кнопке

Кнопка «Save» осуществляет сохранение размеченного изображения и файла с координатами разметки (рис.14-15).



Рис.14.Создание команды SaveImg

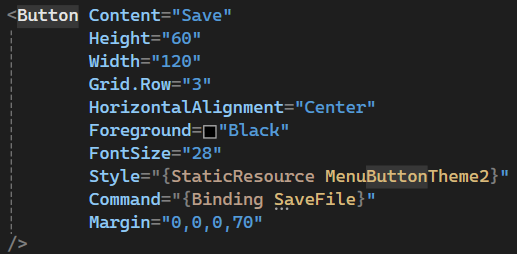


Рис.15. Привязка команды сохранения к кнопке

Для всех кнопок была создана и применена тема MenuButtonTheme2 (рис.16).

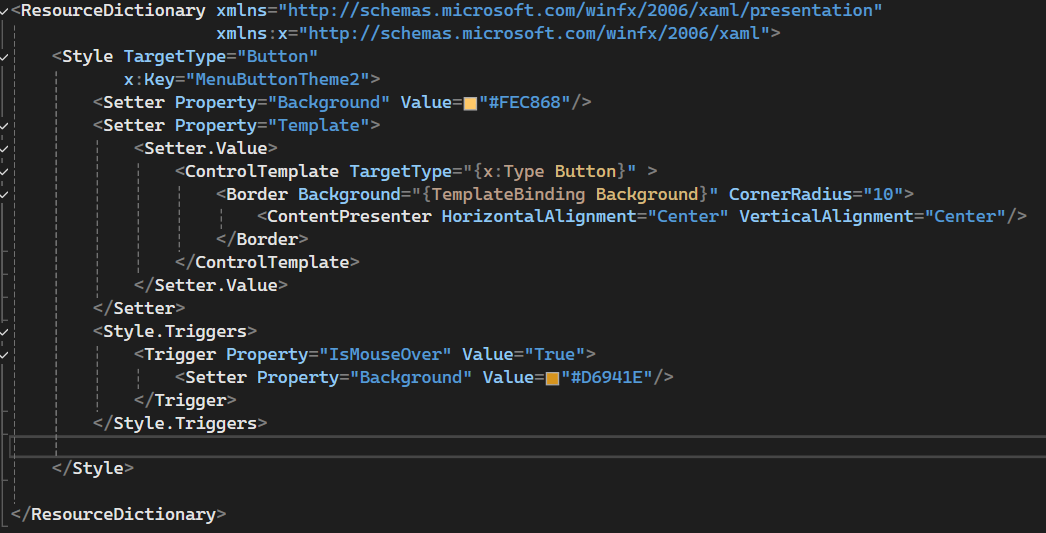


Рис.16. Создание темы для элементов Button

1. Тестирование

В ходе разработки программы проводились тесы на правильное функционирование программы. Данные тестирования представлены в таблице 1.

Таблица 1

Тестирования приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Тестируемый модуль или подпрограмма | Описание тестирование | Тестировщик | Описание результата тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 20.02.2024  15.00 | AutomaticMarkup | Запуск основного окна приложения | Сарычева Е.В. | Ошибки при загрузке |
| 20.02.2024  15.46 | AutomaticMarkup | Повторный запуск  основного окна приложения | Сарычева Е.В. | Успех, ошибок при загрузке не обнаружено |
| 02.03.2024  20.20 | AutomaticMarkup | Тестирование запуска основного окна с графическим представлением кнопок | Сарычева Е.В. | Успех |
| 02.03.2024  14.30 | AutomaticMarkup.Layout | Тестирование кнопки «Upload» | Сарычева Е.В. | При нажатии на кнопку появляется диалоговое окно, изображение не загружается в элемент Image |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 03.03.2024  18.00 | AutomaticMarkup.Layout | Повторное тестирование кнопки «Upload» | Сарычева Е.В. | При нажатии на кнопку появляется диалоговое окно, изображение не загружается в элемент Image |
| 03.03.2024  21.10 | AutomaticMarkup.Layout | Повторное тестирование кнопки «Upload» | Сарычева Е.В. | При нажатии на кнопку появляется диалоговое окно, изображение загружается только в один элемент Image |
| 03.03.2024  21.55 | AutomaticMarkup.Layout | Тестирование кнопки «Save» | Сарычева Е.В. | Осуществляет только сохранение изображения, файлы Json – нет |
| 10.03.2024 | AutomaticMarkup.Layout | Тестирование кнопки «MarkUp» | Сарычева Е.В. | Осуществляет разметку, нужны дополнительные настройки разметки |
| 12.03.2024  20.00 | AutomaticMarkup.Layout | Тестирование кнопки «History» | Сарычева Е.В. | Не выполняется смена регионов |
| 12.03.2024  26.00 | AutomaticMarkup.Layout | Повторное тестирование кнопки «History» | Сарычева Е.В. | Не выполняется смена регионов |
| 20.03.2024  16.00 | AutomaticMarkup.Layout | Повторное тестирование кнопки «Upload» | Сарычева Е.В. | Успех |
| 20.03.2024  20.00 | AutomaticMarkup.Layout | Тестирование кнопки «History» | Сарычева Е.В. | Не выполняется смена регионов |
| 25.03.2024 | AutomaticMarkup.Layout | Тестирование кнопки «MarkUp» | Сарычева Е.В. | Успех |
| 26.03.2024 | AutomaticMarkup.Layout | Тестирование кнопки «History» | Сарычева Е.В. | Смена регионов осуществлена, необходимы доработки |
| 02.04.2024 | MarkupDataBase | Подключение база данных SQL Server Management Studio | Сарычева Е.В. | Успех |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 05.04.2024 | MarkupDataBase | Отображение данных из базы данных | Сарычева Е.В. | Данные отображаются, необходима работа над форматами вывода |
| 26.04.2024  18.00 | AutomaticMarkup.Layout | Тестирование кнопки «History» | Сарычева Е.В. | Успех |
| 26.04.2024  18.40 | AutomaticMarkup.Layout | Тестирование кнопки «Undo» | Сарычева Е.В. | Успех |
| 04.05.2024  18.00 | MarkupDataBase | Тестирование выгрузки изображений из базы данных | Сарычева Е.В. | Загрузка изображения не выполняется |
| 04.05.2024  22.35 | MarkupDataBase | Повторное тестирование выгрузки изображений из базы данных | Сарычева Е.В. | Успех |
| 05.05.2024  21.55 | AutomaticMarkup.Layout | Тестирование кнопки «Save» | Сарычева Е.В. | Успех |
| 15.05.2024  17.00 | MarkupDataBase | Добавление новых записей в базу данных | Сарычева Е.В. | Не удается добавить новые записи |
| 15.05.2024  21.40 | MarkupDataBase | Добавление новых записей в базу данных | Сарычева Е.В. | Успех |
| 20.05.2024 | MarkupDataBase | Обработка исключений | Сарычева Е.В. | Успех |
| 22.05.2024 | AutomaticMarkup.Layout | Тестирование работы модуля в целом | Сарычева Е.В. | Успех |
| 23.05.2024 | AutomaticMarkup | Тестирование работы модуля в целом | Сарычева Е.В. | Успех |
| 24.05.2024 | MarkupDataBase | Тестирование работы модуля в целом | Сарычева Е.В. | Успех |
| 29.05.2024 | MarkupDataBase, AutomaticMarkup,  AutomaticMarkup.Layout | Тестирование работы приложения | Сарычева Е.В. | Успех |

# Заключение

В ходе выполнения данной курсовой работы была написана программа, использующая паттерн MVVM и фреймворк Prism. Программа создавалась в системе для построения клиентских приложений WPF.

Для написания программы потребовалось изучить паттерн MVVM, внедрение зависимостей, навигацию и основы работы с XAML.

# Список литературы:

1. Официальная документация по WPF на сайте Microsoft. 2024. [Электронный ресурс] - URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-8.0 (дата обращения: 1 июня 2024).
2. Десять командментов паттернов WPF. [Электронный ресурс]. – URL: https://www.decosystems.ru/wpf/ (дата обращения: 07.06.2024).
3. Описание языка XAML. [Электронный ресурс]. – URL: https://metanit.com/sharp/wpf/2.php (дата обращения: 07.06.2024).
4. Руководство разработчика Prism. [Электронный ресурс]. – URL: https://habr.com/ru/articles/176851/
5. Руководство разработчика Prism — часть 5, реализация паттерна MVVM [Электронный ресурс]. – URL: https://habr.com/ru/articles/176867/
6. Хабр, статья «Руководство разработчика Prism, навигация на основе представлений (View-Based Navigation)». [Электронный ресурс]- <URL:https://habr.com/ru/post/182052/>

# Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

наименование института (факультета)

МПО ЭВМ

наименование кафедры

C#-программирование

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ,

д. т.н., профессор \_\_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_ г.

Техническое задание на курсовую работу по теме

«Разработка программного обеспечения на языке C#»

Листов 4

Руководитель:

Шаханов Никита Иванович

Ф.И.О. преподавателя

Исполнитель:

студент 1ПИб-01-1оп-20

группа

Сарычева Екатерина Васильевна

Фамилия, имя, отчество

2024 год

Введение

Целью курсовой работы является написание программы с использованием технологии WPF, фреймворка Prism, а также паттерна проектирования MVVM. Так же должны быть применены и описаны модули, внедрение зависимостей и навигация.

Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине "C# - программирование", выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 01.02.2024 года.

Наименование темы разработки: Разработка программного обеспечения на языке C#.

Назначение разработки

Данная программа предназначена для автоматической разметки производственных изображений на основе маски. Так же программа нужна для студентов, которые знакомятся с WPF и применением фреймворка Prism.

Требования к программе

* 1. Требования к функциональным характеристикам:

Программа должна загружать изображения, иметь функционал для создания размеченных изображений и сохранять их, а также иметь доступ к базе данных.

* 1. Требования к надежности:

Работа программного продукта должна быть стабильной, вносимые данные должны быть валидными. При возникновении исключений программа должна обрабатывать ошибки.

* 1. Условия эксплуатации:

Температура окружающего воздуха и влажность должны быть на уровне приемлемых для эксплуатации компьютера.

* 1. Требования к составу и параметрам технических средств:

Для нормальной работы программы необходимы следующие минимальные системные требования:

* тип процессора – intel core i3-560 и выше;
* операционная система – Windows 7 и выше;
* оперативная память – 4gb и выше;
* объем жесткого диска – не менее 20 Гб;
* источник бесперебойного питания.
  1. Требования к информационной и программной совместимости:

Prism был разработан для работы на операционной системе Microsoft Windows 7 или выше. Приложения WPF, созданные с использованием этого руководства, требуют .NET Framework 4.5 и выше. Исходный код программы реализуется при помощи языка C# и встроенных в него функции на базе Visual Studio 2022.

* 1. Требования к транспортированию и хранению

Разработка будет распространяться на съемном носителе, который нельзя подвергать воздействию воды. Вынимать съемный носитель необходимо в безопасном режиме.

Требования к программной документации

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

1. Титульный лист
2. Оглавление
3. Введение
4. Выбор и описание предметной области
5. Обзор основных частей Фреймворка Prism

* MVVM
* Модули
* Внедрение зависимостей (Unity или MEF)
* Навигация
* Интеграция с XAML

1. Разработка WPF-приложения

* Разработка диаграммы классов и диаграммы пакетов с учетом инверсии управления (IoC)
* Построение оболочки с пользовательским интерфейсом и описанием регионов (представления)
* Проектирование независимых модулей (модели и модели-представления)
* Разработка бизнес логики приложения
* Навигация и обмен между частями модулей
* Тестирование

1. Заключение
2. Список литературы
3. Приложения

Стадии и этапы разработки

Стадии и этапы разработки представлены в табл. П1.1

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Этапы разработки | Результат выполнения | Отметка о выполнении |
| Получение технического задания | 13.02.2024 | Составленное технического задания |  |
| Анализ предметной области | 19.02.2024 – 04.04.2024 | Определение объектов предметной области |  |
| Написание программы | 04.04.2024 – 01.05.2024 | Получение работающей и протестированной программы |  |
| Расчётно-пояснительная записка | 01.05.2024 – 11.05.2024 | Написанная РПЗ |  |
| Демонстрация программы | 02.06.2024 | Представлена работающая программа |  |
| Защита курсовой работы | 07.06.2024 | Получен зачет |  |

Порядок контроля и приемки

Приемка и проверка осуществляется при защите курсовой работы.

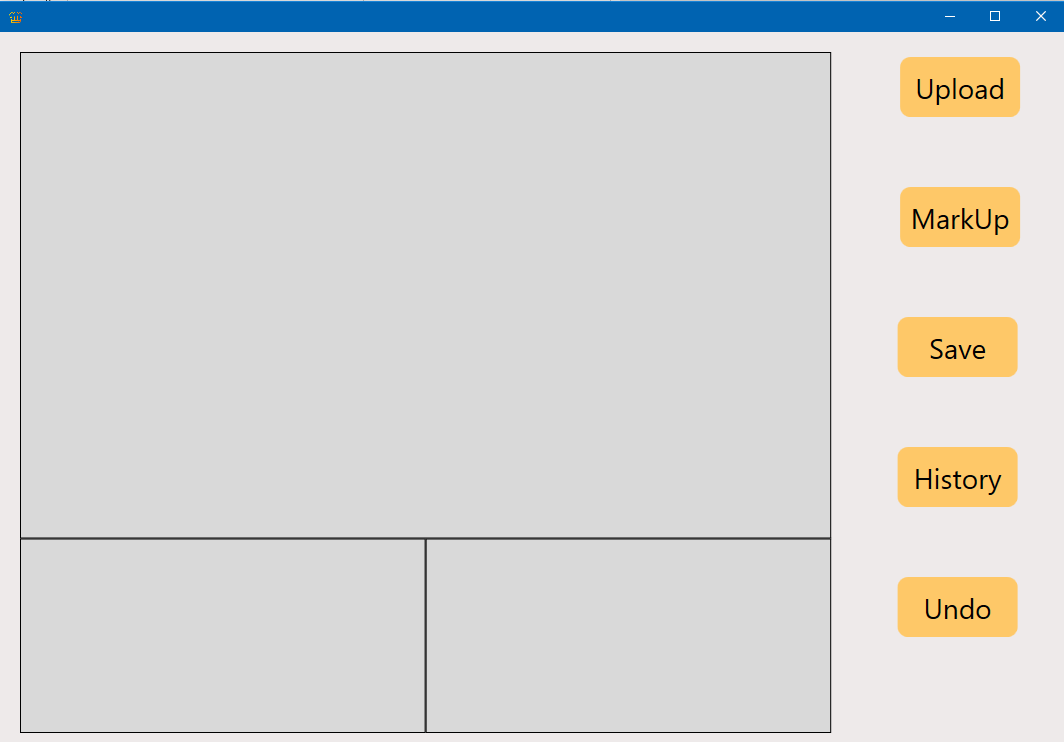
# Приложение 2. Руководство пользователя

Для начала работы с приложением необходимо запустить файл приложения(рис. П3.1)



Рис.П3.1. Запускающий файл

Откроется стартовое окно (рис. П3.2).



3

2

1

Рис.П3.2. Окно приложения при запуске

Пользователь может загрузить изображение и его маску после взаимодействия с кнопкой «Upload». После нажатия на кнопку откроется диалоговое окно для выбора файла(рис.П3.3), выбрав нужное изображение, нужно будет после выбрать и маску для данного изображения. Исходное изображение сразу отобразится в поле 1, а его маска в поле 2 (рис.П3.4).

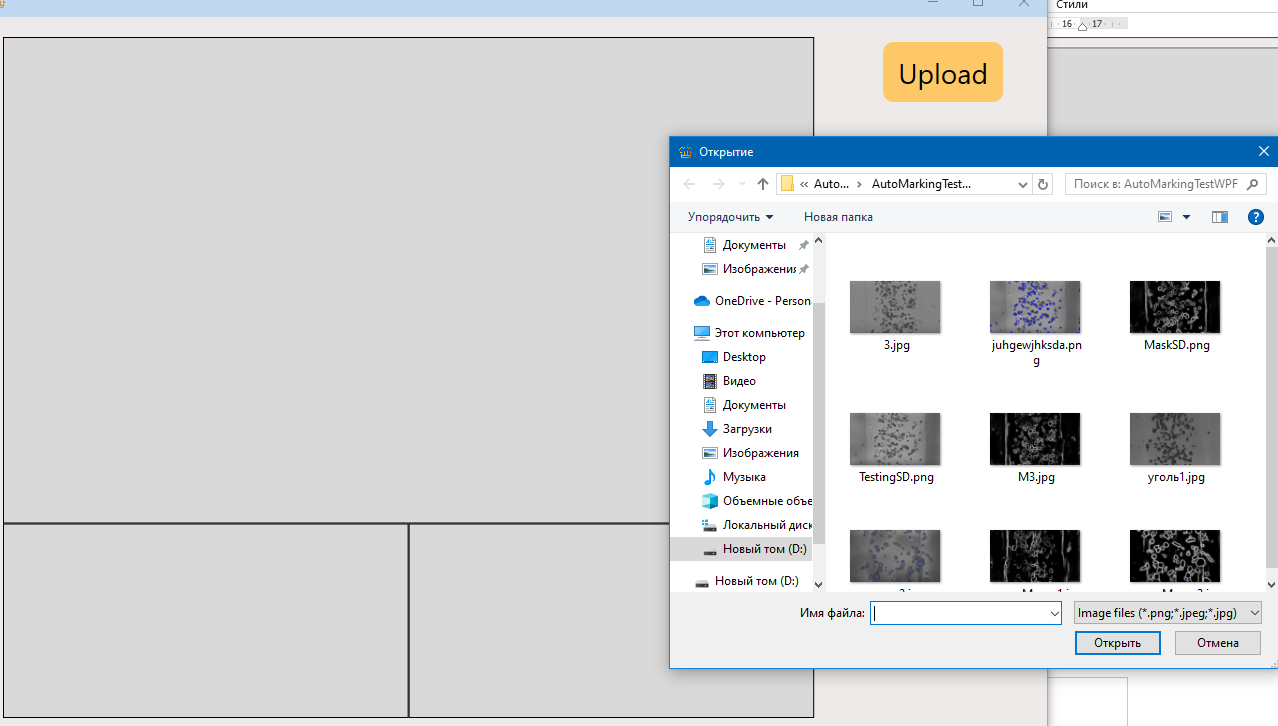


Рис.П3.3. Выбор исходного изображения

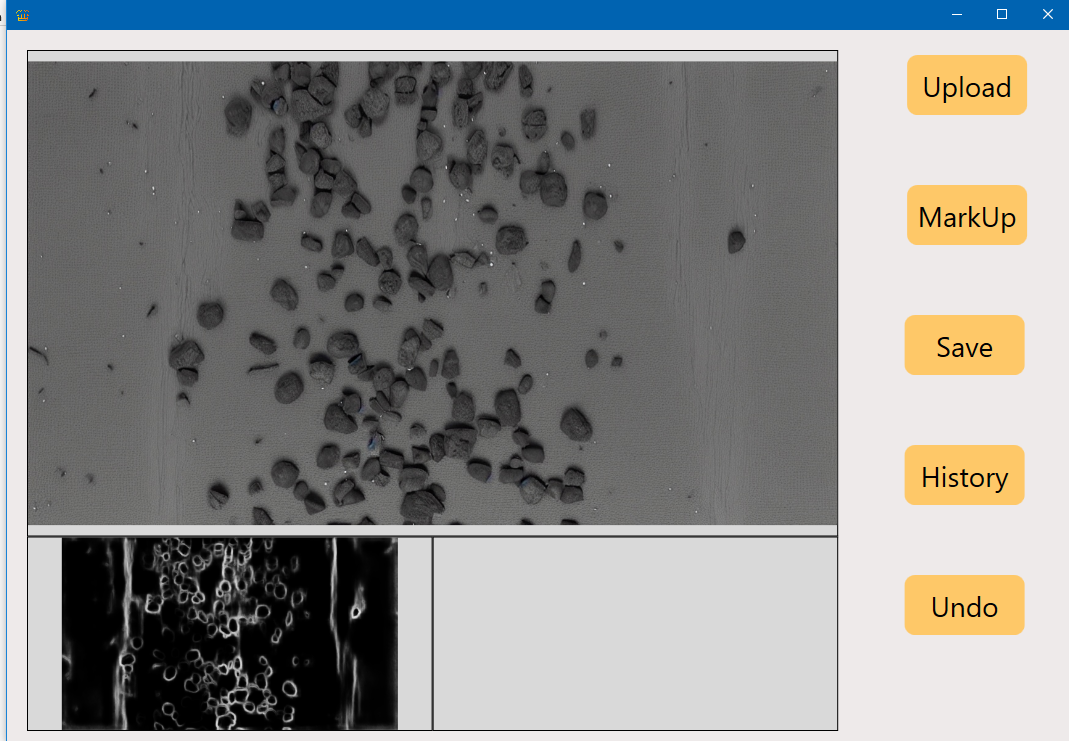


Рис.П3.4. Загруженное изображение и его маска

После выбора нужных исходных файлов пользователь может нажать кнопку «MurkUp» для создания разметки. После завершения операции появится размеченный файл в 3 поле (рис.П3.5).

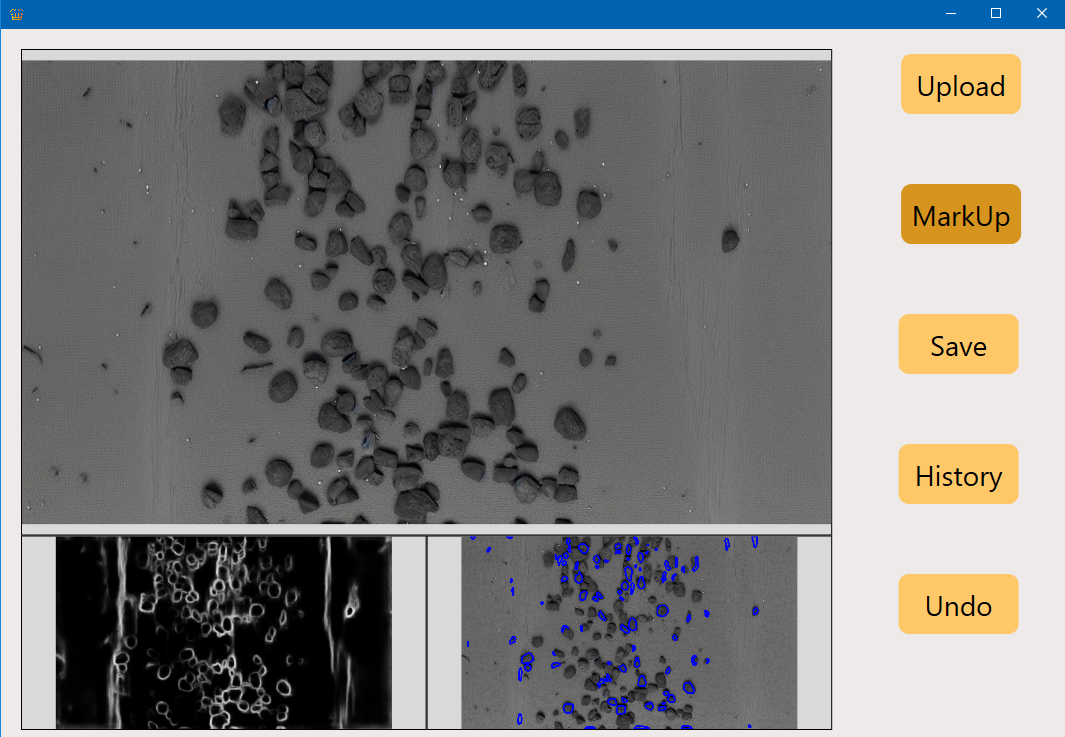


Рис.П3.5. Разметка завершена

Если пользователю необходимо вывести какое-то изображение в поле 1, достаточно нажать на это изображение, и оно поменяется места с изображением из поля 1 (рис.П3.6).

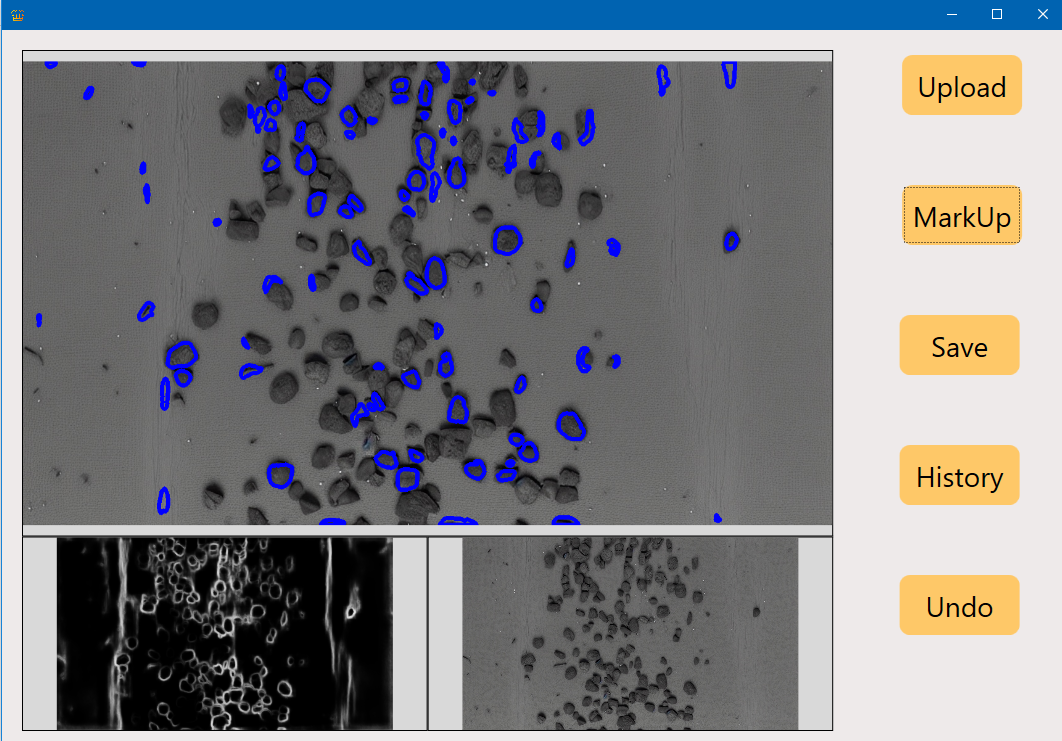


Рис.П3.6. Исходное изображение и изображение с разметкой поменялись местами

Теперь результаты можно сохранить, для этого необходимо воспользоваться соответствующей кнопкой «Save», выбрать папку, в которую необходимо сохранить файл, и как-то назвать. Будет сохранено 2 файла: изображение с разметкой и файл разметки (рис.П3.7).

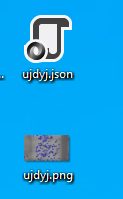


Рис.П3.7. Сохраненные файлы

Также приложение подцеживает историю (рис.П3.8) с помощью которой можно восстановить прежде размеченные файлы.

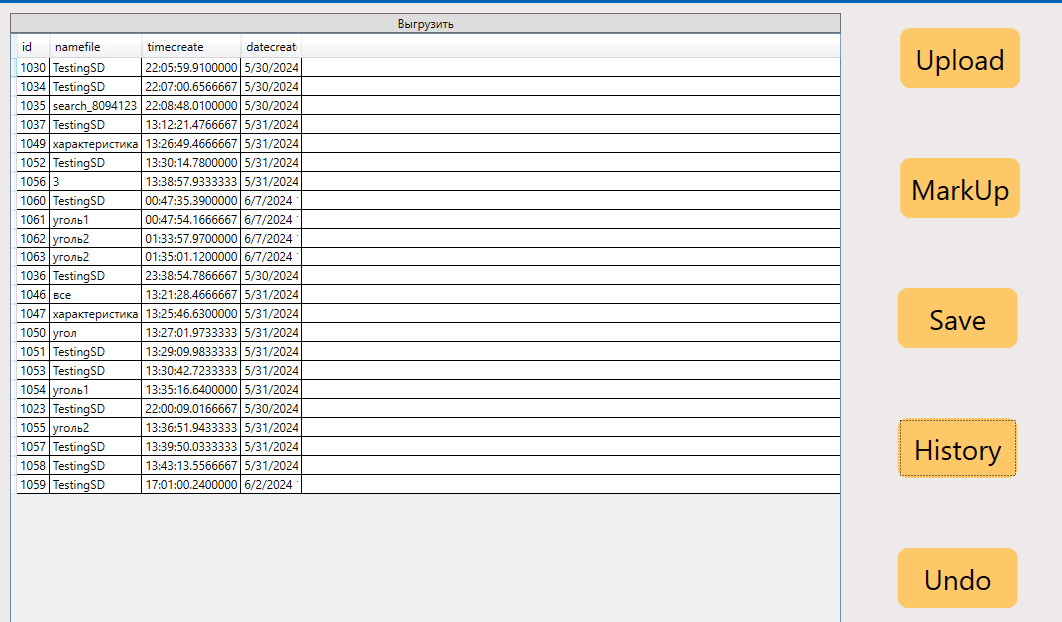


Рис.П3.8. Окно истории

Чтобы восстановить файлы нужно выбрать соответствующую строку в таблице и нажать на кнопку «Выгрузить», она расположена сверху (рис.П3.9). Далее нужно вернуться на прежнюю страницу, нажав на кнопку «Undo», на ней будут отображены восстановленные изображения (рис.П3.10).

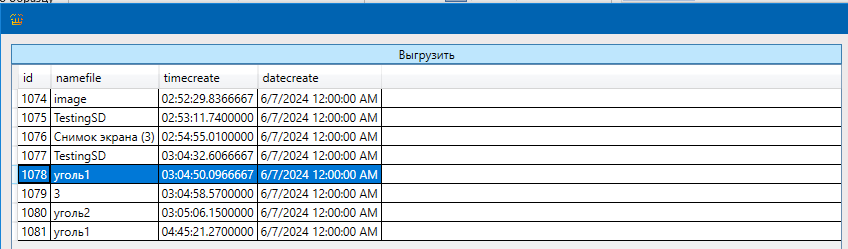


Рис.П3.9. Выбор

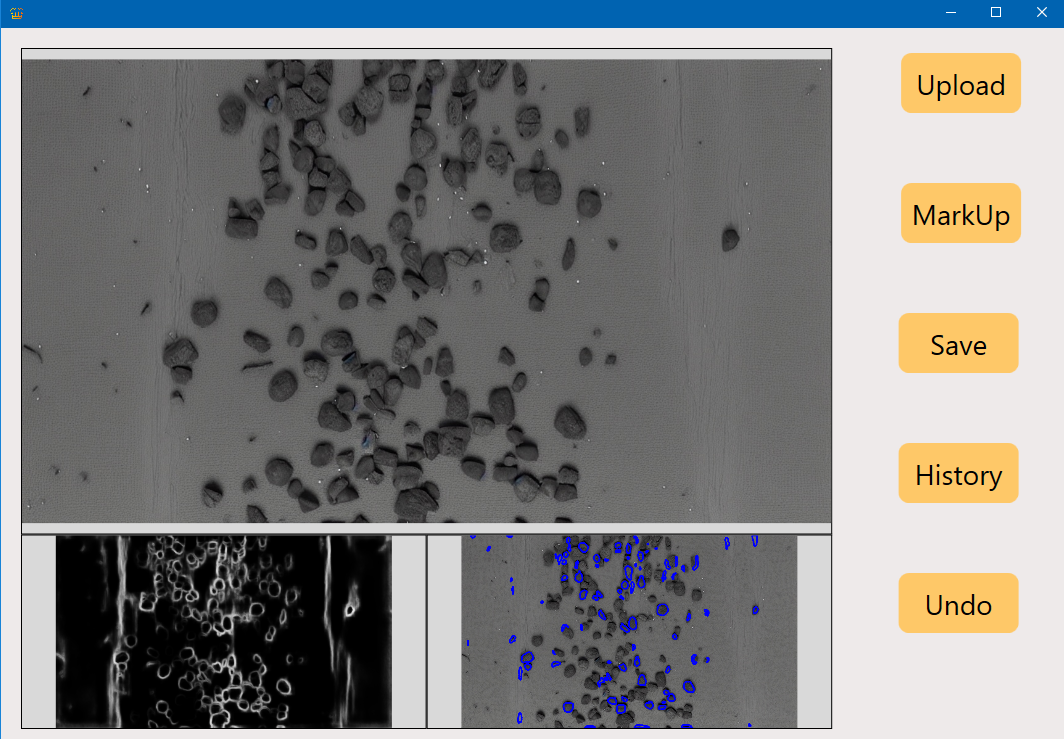


Рис.П3.10. Восстановленные изображения

К руководству создано видео с демонстрацией работы программы (рис.П3.11).



Рис.П3.11. Файл видео с демонстрацией

# Приложение 3. Код программы

<Border Margin="10">

<Border.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,2">

<GradientStop Color="#D9D9D9" Offset="0.0"/>

</LinearGradientBrush>

</Border.Background>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2.5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Border Grid.Row="0"

BorderThickness="1"

BorderBrush="Black">

<Image Source="{Binding ImageUp}"

x:Name="ImageUp"/>

</Border>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Border Grid.Column="0"

BorderThickness="1"

BorderBrush="Black">

<Image Source="{Binding ImageLeft}">

<Image.InputBindings>

<MouseBinding Command="{Binding ImageLeftClickCommand}" MouseAction="LeftClick"/>

</Image.InputBindings>

</Image>

</Border>

<Border Grid.Column="1"

BorderThickness="1"

BorderBrush="Black">

<Image Source="{Binding ImageRight}">

<Image.InputBindings>

<MouseBinding Command="{Binding ImageRightClickCommand}" MouseAction="LeftClick"/>

</Image.InputBindings>

</Image>

</Border>

</Grid>

</Grid>

</Border>

Рис.П3.1. Код HomeView.xaml

<Grid >

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Button Content="Upload"

Height="60"

Width="120"

Grid.Row="0"

HorizontalAlignment="Center"

Foreground="Black"

FontSize="28"

Style="{StaticResource MenuButtonTheme2}"

Command="{Binding UploadFile}"

Margin="5,20,0,70"/>

<Button Content="MarkUp"

Height="60"

Width="120"

Grid.Row="1"

HorizontalAlignment="Center"

Foreground="Black"

FontSize="28"

Style="{StaticResource MenuButtonTheme2}"

Command="{Binding DoMarkUp}"

Margin="5,0,0,70"

/>

<Button Content="Save"

Height="60"

Width="120"

Grid.Row="3"

HorizontalAlignment="Center"

Foreground="Black"

FontSize="28"

Style="{StaticResource MenuButtonTheme2}"

Command="{Binding SaveFile}"

Margin="0,0,0,70" />

<Button Content="History"

Height="60"

Width="120"

Grid.Row="4"

HorizontalAlignment="Center"

Foreground="Black"

FontSize="28"

Style="{StaticResource MenuButtonTheme2}"

Command="{Binding BDHistory}"

Margin="0,0,0,70" />

<Button Content="Undo"

Height="60"

Width="120"

Grid.Row="5"

HorizontalAlignment="Center"

Foreground="Black"

FontSize="28"

Style="{StaticResource MenuButtonTheme2}"

Command="{Binding Undo}"

Margin="0,0,0,70" />

</Grid>

Рис.П3.2. Код MainView.xaml

<UserControl x:Class="AutomaticMarkup.Layout.Views.StoryView"

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

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

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

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

xmlns:local="clr-namespace:AutomaticMarkup.Layout.Views"

xmlns:vm="clr-namespace:AutomaticMarkup.ViewModels"

mc:Ignorable="d"

d:DataContext="{d:DesignInstance vm:StoryViewModel}"

d:DesignHeight="150" d:DesignWidth="300">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Button Grid.Row="0"

Content="Выгрузить"

Command="{Binding GetHistoryMoment}">

</Button>

<DataGrid ItemsSource="{Binding DataTable}"

SelectedIndex="{Binding row}"

AutoGenerateColumns="True" IsReadOnly="True"

Grid.Row="1">

</DataGrid>

</Grid>

</UserControl>

Рис.П3.3. Код StoryView.xaml

<Border Background="#EEEAEA">

<Grid Cursor="Arrow">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="4\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Border Grid.Column="1" Grid.ColumnSpan="1">

<ContentControl prism:RegionManager.RegionName="MenuRegion" Margin="5"/>

</Border>

<Border Grid.Column="0" Grid.Row="1">

<ContentControl prism:RegionManager.RegionName="HomeRegion" Margin="10,10,10,0"/>

</Border>

</Grid>

</Border>

Рис.П3.4. Код MainWindow.xaml

<prism:PrismApplication

x:Class="AutomaticMarkup.App"

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

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

xmlns:prism="http://prismlibrary.com/">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Theme/MenuButtonTheme2.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</prism:PrismApplication>

Рис.П3.5. Код App.xaml