**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ»**

**Факультет «ИКТ»**

**Дисциплина**

«Создание программного обеспечения инфокоммуникационных систем»

**Лабораторная работа 1**

**«**Анализ предметной области. Создание проекта WPF**»**

**Выполнил:**

студент группы K3415

Лобченко Д. Р.

**Проверил**:

Осипов Н. А.

Санкт-Петербург

2021

**Практическое задание 1. Создание приложений WPF**

**Упражнение 1. Создание приложения Windows**

**Цель задания:**

Создать простое WPF приложение. Познакомиться с конструктором и редактором XAML. Протестировать базовые элементы пользовательского интерфейся.

**Шаги выполнения работы:**

1. Создадим новый проект WPF Application с именем «WpfHello».

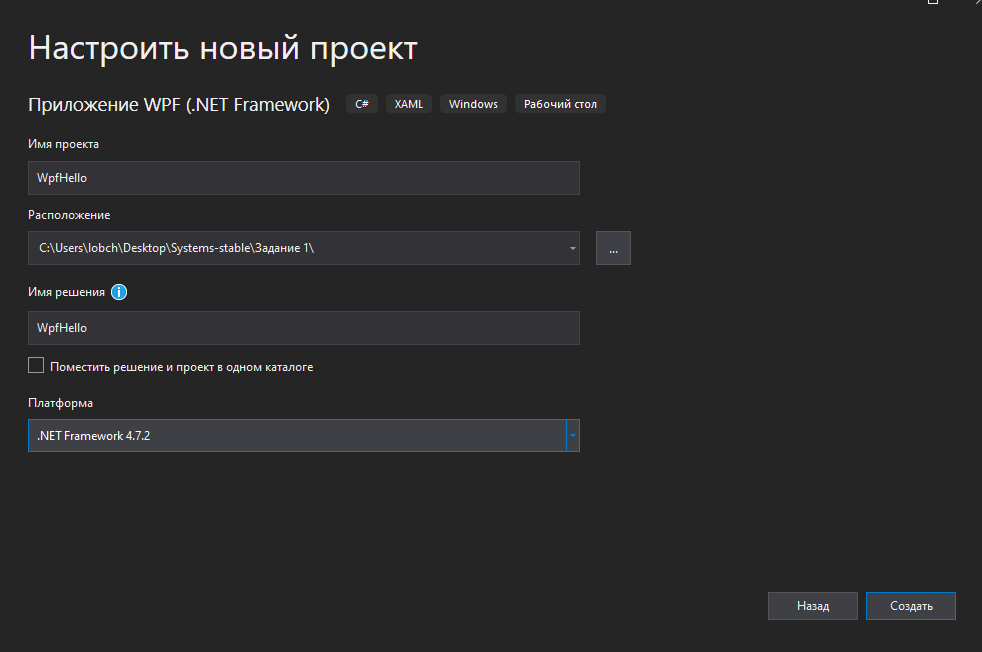


Рисунок 1 - окно создания нового приложения в Visual Studio.

1. Используя панель инструментов (Toolbox) определим на форме элементы пользовательского интерфейса **Textbox**, **Button** и **Label** в соответствии с заданием.

**Textbox** – элемент пользовательского интерфейса, позволяющий конечному пользователю вводить простой текст. Элемент поддерживает однострочное (по умолчанию) и многострочное представление.

**Button** - элемент пользовательского интерфейса, представляющий базовую кнопку. При нажатии на кнопку вызывается событие «Click» и соответствующий обработчик события.

**Label** - элемент пользовательского интерфейса, представляющий метку, которая может включать в себя текст и любые элементы управления. Важной особенностью метки является возможность использовать клавиши быстрого доступа для связи с другими элементами формы.

1. Определим свойства добавленных элементов в соответствии с заданием.

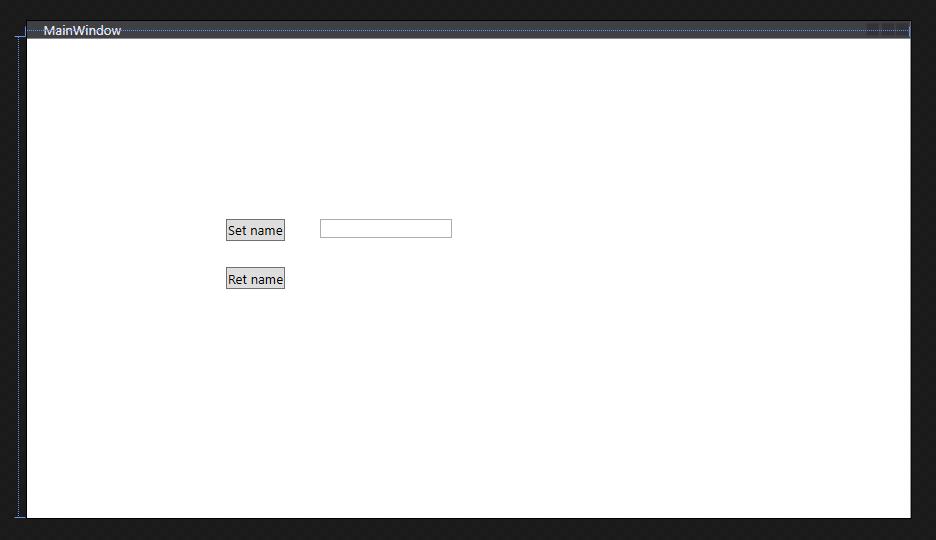


Рисунок 2 - Определенный пользовательский интерфейс в конструкторе.

1. Определим обработчики нажатия (Click) на созданные кнопки.

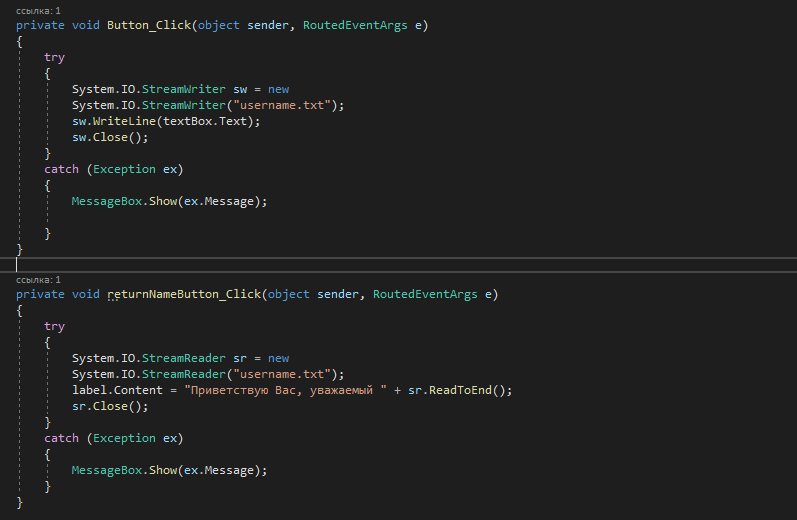


Рисунок 3 - определение обработчиков события Click для кнопок.

1. Проверим работу созданного приложения.

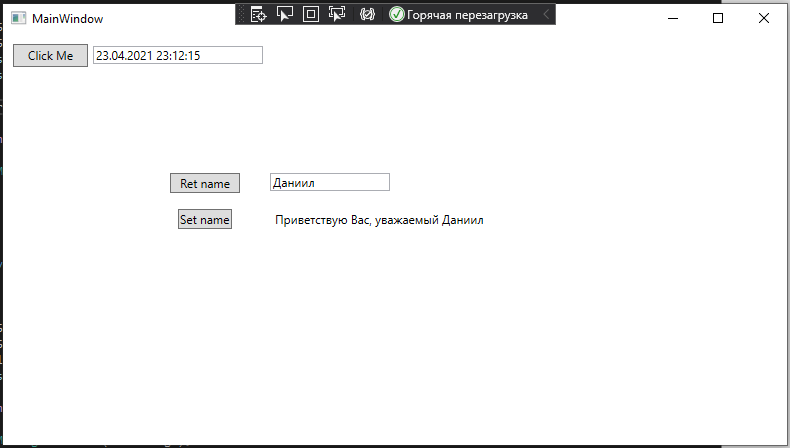


Рисунок 4 - Проверка работоспособности приложения

**Практическое задание 1. Создание приложений WPF**

**Упражнение 2. Создание навигационного приложения**

1. Создадим новый проект WPF Application с именем «WpfHelloNav».
2. В соответствии с заданием, изменим тег Window на NavigationWindow в режиме разметки XAML. Удалим тег <Grid>

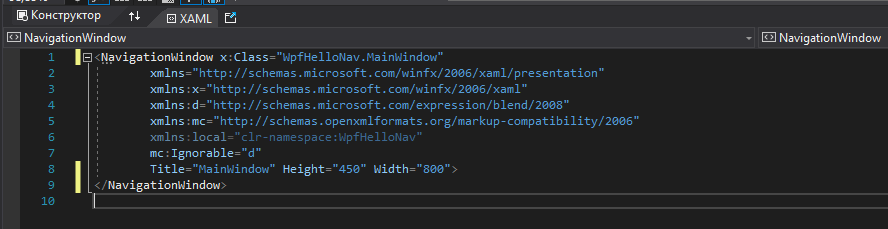


Рисунок 5 - Редактирование формы в режиме разметки XAML.

1. Отредактируем класс формы в режиме кода, заменив базовый класс Window на NavigationWindow.

Класс NavigationWindow более или менее похож на базовый класс Window, за исключением кнопок навигации "вперед" и "назад", которые отображаются в строке сверху.

1. Запустим и протестируем приложение.

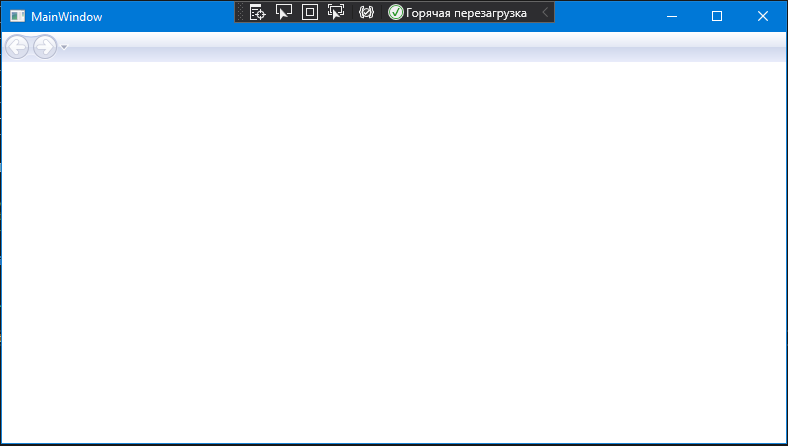


Рисунок 6 - Интерфейс приложения (NavigationWindow)

1. Добавим в приложение две страницы WPF.
2. Для главного окна установим свойство Source со значением Page1.xaml, чтобы она отображалась при запуске приложения.
3. На страницу Page1 добавим кнопку с обработчиком события Click, определенным в задании.
4. Построим и запустим приложение, чтобы протестировать его работу.

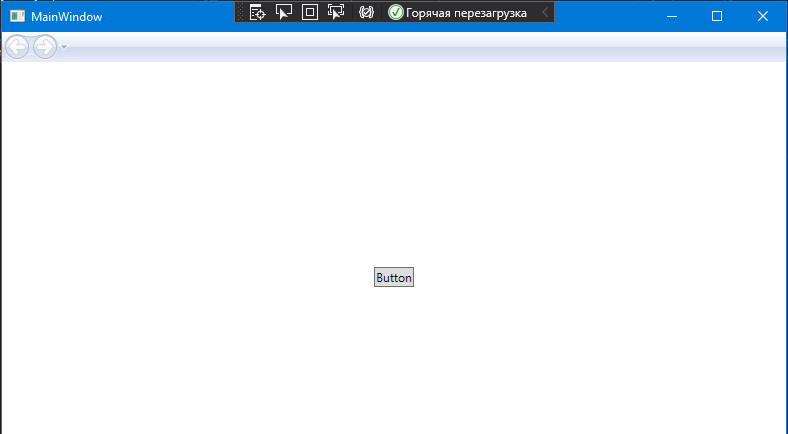


Рисунок 7 – приложение «WpfHelloNav». Страница 1

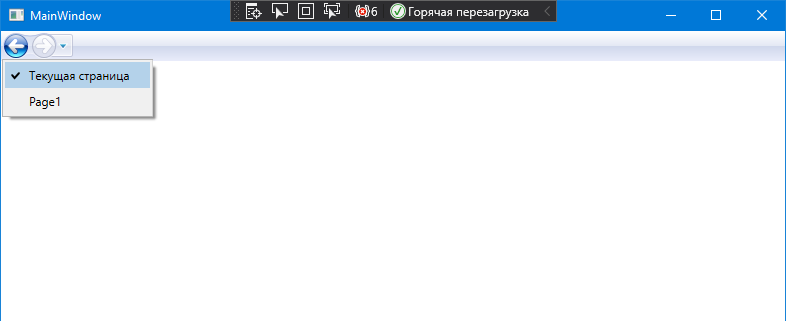


Рисунок 8 - приложение «WpfHelloNav». Страница 2

**Практическое задание 1. Создание приложений WPF**

**Упражнение 3. Реализация обработки событий**

1. Откроем проект первого упражнения.
2. Отредактируем код приложения в соответствии с заданием.
3. Добавим для текстового поля обработчик события ввода текста.
4. Добавим в обработчик события Click первой кнопки код, активирующий интерактивность второй кнопки.



Рисунок 9 - Измененный код формы.

1. Запустим и протестируем приложение.

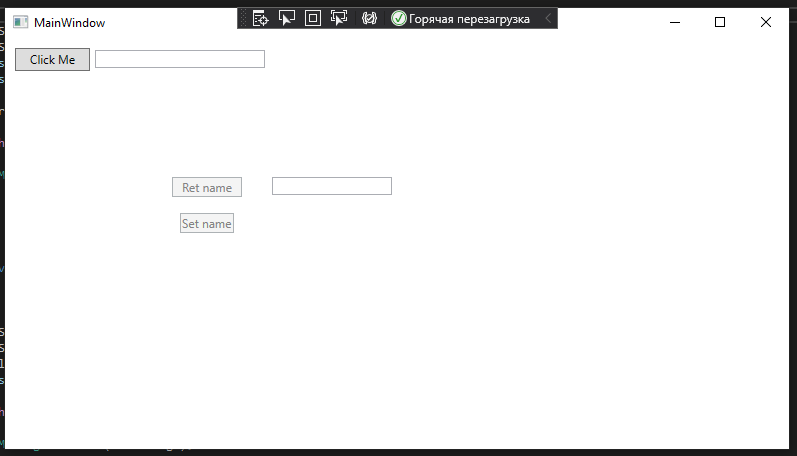


Рисунок 10 - До ввода имени кнопка задания имени неактивна.

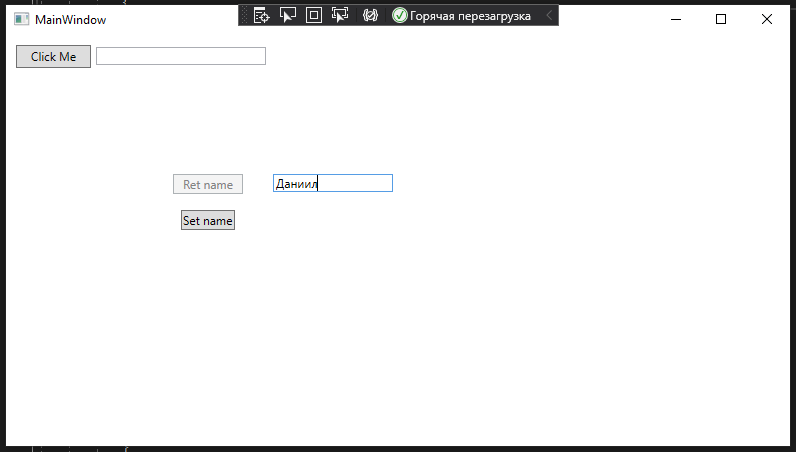


Рисунок 11 - После начала ввода кнопка становится активна.

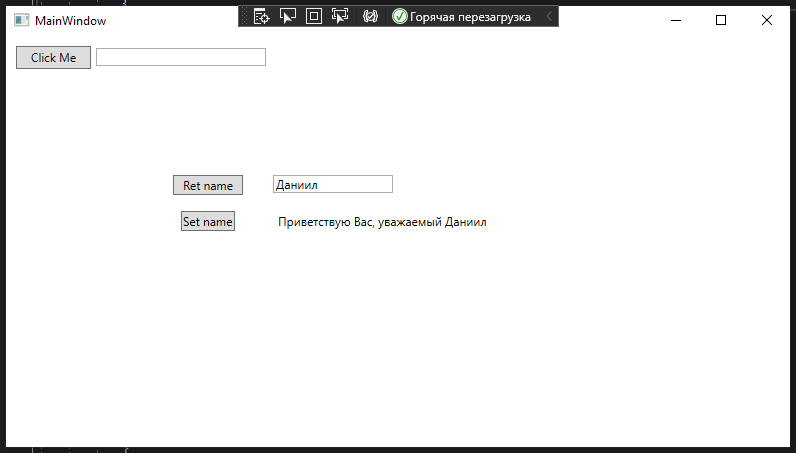


Рисунок 12 - После задания имени активируется кнопка отображения приветствия

**Практическое задание 1. Создание приложений WPF**

**Упражнение 4. Использование редактора XAML**

1. Откроем проект предыдущего упражнения.
2. Изменим корневой элемент в файле MainWindow.xaml на **Canvas**.
3. Отредактируем файл в соответствии с заданием, добавим элементы **Button** и **TextBox**.
4. Для кнопки добавим обработчик события Click и определим его в коде формы.
5. Построим и протестируем приложение. В текстовом поле отображается текущая дата и время.



Рисунок 13 - тестирование приложения

**Практическое задание 2. Маршрутизированные события**

**Упражнение 1. Создание обработчика события**

В данном упражнении рассматривается свойство некоторых событий всплывать вверх по визуальному дереву.

Такие события будут всплывать вверх (распространяться) по визуальному дереву от исходного элемента, пока не будут обработаны или не достигнут корневого элемента. Это позволяет обработать их любому родительскому элементу, стоящему в логическом дереве выше исходного.

1. Создадим новый проект WPF Application с именем «WpfRoutedEvent\_2\_1».
2. Определим пользовательский интерфейс приложения.
3. Определим свойства добавленных элементов в соответствии с заданием.

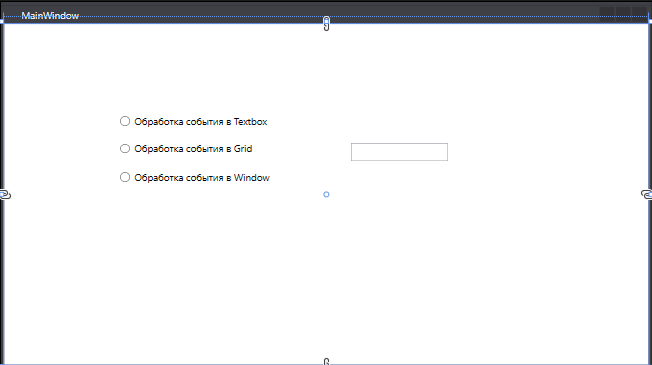


Рисунок 14 - Определенный интерфейс приложения

1. В представлении XAML зададим свойства TextChanged и TextBoxBase.TextChanged для элементов **TextBox, Grid и Window.**
2. Определим обработчики событий в классе формы.
3. Построим и протестируем приложение.

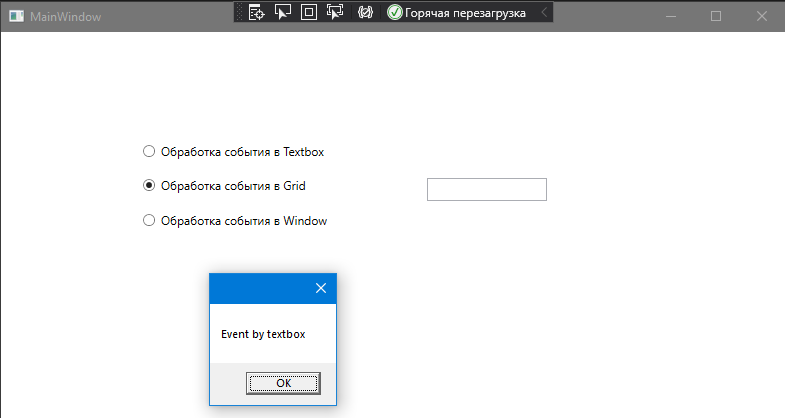


Рисунок 15 - Обработка события обработчиком элемента TextBox

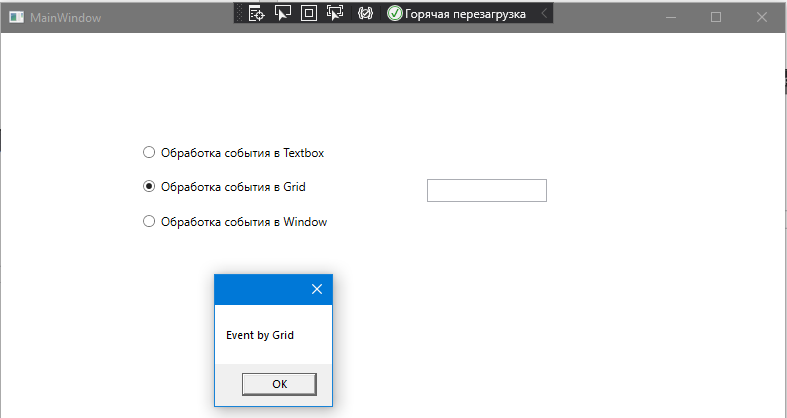


Рисунок 16 - Обработка события обработчиком элемента Grid

**Практическое задание 2. Маршрутизированные события**

**Упражнение 2. Использование маршрутизированных событий**

1. Создадим новый проект WPF Application с именем «WpfRoutedEvent\_2\_2».
2. Определим пользовательский интерфейс приложения.
3. Определим свойства добавленных элементов в соответствии с заданием.
4. Добавим обработчик события клика по кнопке на элемент StackPanel, содержащий добавленные кнопки.
5. Построим и протестируем приложение.

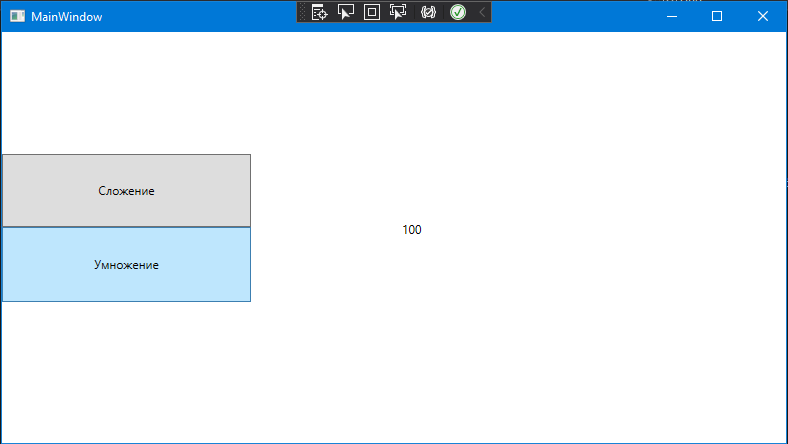


Рисунок 17 - Состояние приложения до нажатия кнопок.

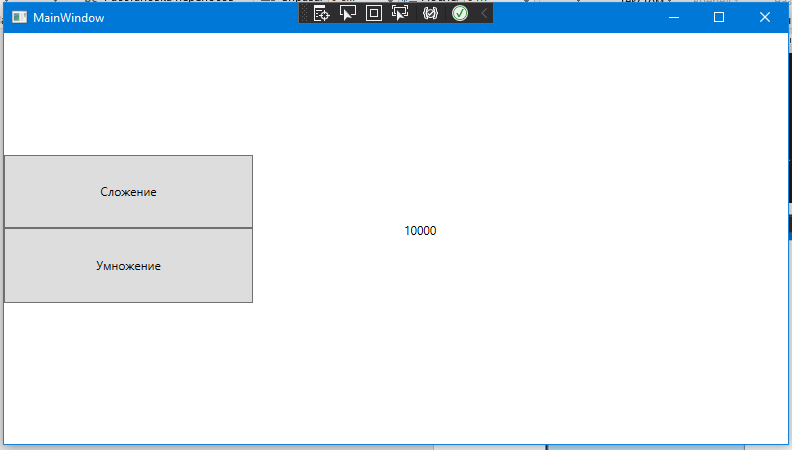


Рисунок 18 - Состояние приложения после умножения.

**Практическое задание 3. Работа с командами**

**Упражнение 1. Применение встроенных команд**

1. Создадим новый проект WPF Application с именем «NCommand».
2. Определим пользовательский интерфейс приложения.
3. Определим свойства добавленных элементов в соответствии с заданием.
4. Определим пункты меню и зададим для них соответствующие команды.
5. Добавим элементы пользовательского интерфейса TextBox и RichTextBox.
6. Запустим и протестируем приложение.

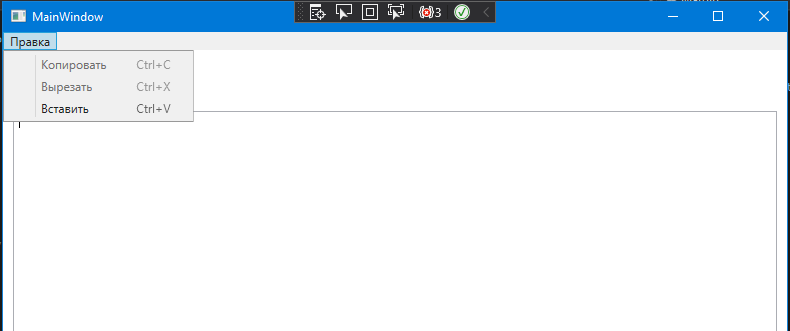


Рисунок 19 - Тестирование приложения

**Практическое задание 3. Работа с командами**

**Упражнение 2. Создание и использование настраиваемой команды**

1. Откроем проект предыдущего упражнения.
2. Добавим в проект новый класс – CustomCommands.
3. Отредактируем ранее созданный класс CustomCommands в соответствии с заданием.



Рисунок 20 - класс CustomCommands

1. Отредактируем элемент меню, добавив новый элемент MenuItem с заголовком «Launch».
2. Отредактируем пользовательский интерфейс, добавив элементы Button и CheckBox со свойствами, определенными в условии лабораторной работы.
3. Отредактируем класс формы, добавив привязку CommandBinding для команды Launch, обработчики событий Launch\_Handler и LaunchEnabled\_Handler.



Рисунок 21 - класс MainWindow

1. Соберем и протестируем приложение.

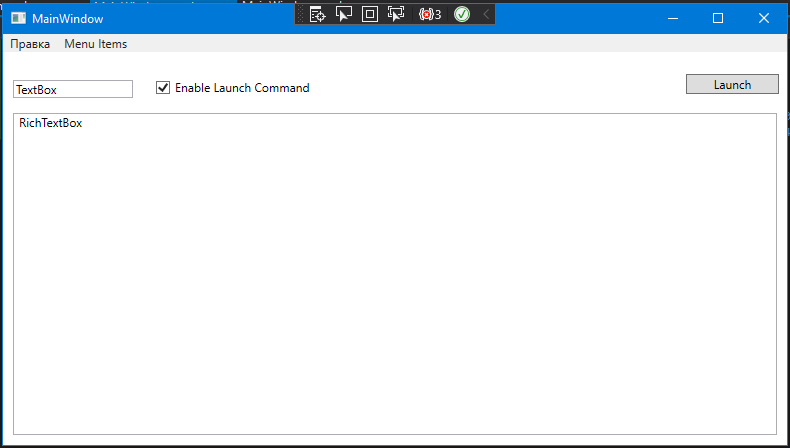


Рисунок 22 - Тестирование приложения

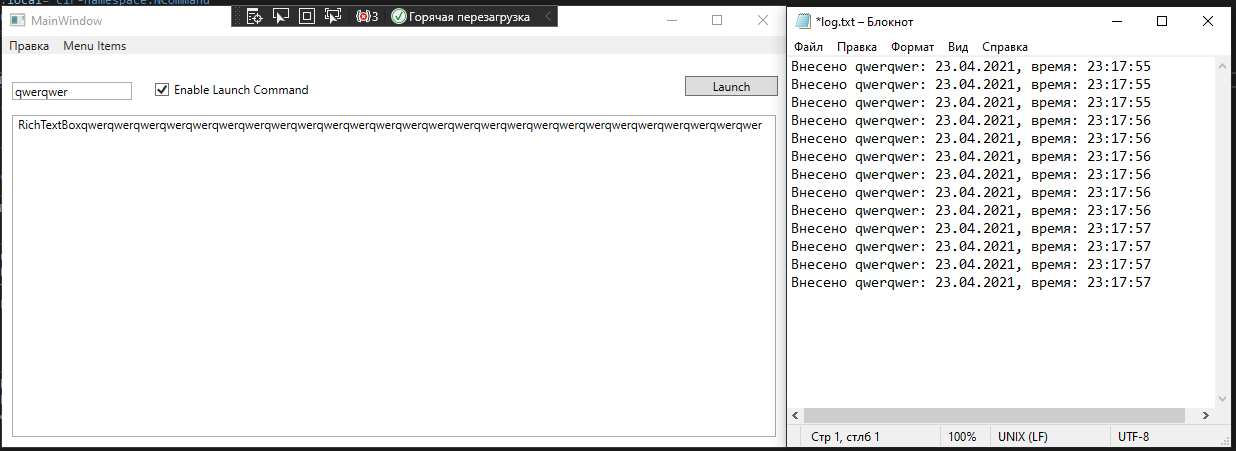


Рисунок 23 - Проверка сохранения логов в файл

**Вывод**

В данной лабораторной работе были рассмотрены:

* Процесс создания WPF приложения в среде разработки Visual Studio
* Работа с элементами пользовательского интерфейса, их создание и настройка посредством возможностей Toolbox, а также непосредственно через редактирование XAML файлов проекта.
* Работа с событиями в WPF приложении, делегирование событий, а также особенность событий всплывать по дереву компонентов.
* Работа с командами, использование встроенных команд и создание новых.

Таким образом можно отметить, что разработка WPF приложения в среде Visual Studio имеет большую вариативность, элементы могут добавляться и редактироваться как непосредственно в конструкторе формы с использованием вкладки Toolbox, предоставляющей обширный набор элементов, так и через код XAML файлов.

События же делятся на прямые, туннельные и пузырьковые. Обработка пузырьковых и туннельных событий возможна с использованием паттерна делегирования обработки на элементы верхнего уровня.

В WPF кроме обработки событий приложение может взаимодействовать с пользователем с помощью команд. Команды представляют собой механизм выполнения какой-нибудь задачи в процессе выполнения которой все действия объединяются в одну команду. Использование команд помогает сократить объем кода и использовать одну и ту же команду для нескольких элементов управления в различных местах программы. Таким образом, команды позволяют абстрагировать набор действий от конкретных событий конкретных элементов.