

Оглавление

Тест.....	1
Практика.....	11
Список дел.....	12
Проверка совершеннолетия.....	16
Часы	19
Поиск слов.....	23
Сумма 2х чисел.....	26
Текст из файла	28
Первый различный символ в 2х текстах.....	30
Количество слов в тексте.....	32
Рисование точки по клику.....	34
Изображения из папки	36
Изображение по ссылке.....	38
Визуализация Цвета	40
Список с перемещением	42
Пагинация	44
Номер телефона	47
Сообщения	49

Тест

1. Что такое C# и для чего он используется?
 - a) Язык программирования для создания веб-страниц.
 - b) Язык программирования для разработки приложений на платформе .NET.
 - c) Язык разметки для оформления документов.
2. Как объявляется переменная в C#?
 - a) variable x;
 - b) int x;

- c) declare x;
3. В чем разница между int и float в C#?
- a) `int` используется для целых чисел, а `float` для чисел с плавающей запятой.
 - b) `int` используется для чисел с плавающей запятой, а `float` для целых чисел.
 - c) Нет разницы.
4. Как создать константу в C#?
- a) `const int x = 5;`
 - b) `readonly int x = 5;`
 - c) `constant x = 5;`
5. Что такое массив в C# и как его объявить?
- a) `array x = new array();`
 - b) `int[] x = new int[5];`
 - c) `list<int> x = new list<int>();`
6. Каков синтаксис использования условного оператора if в C#?
- a) `if x > 0 then { /* code */ }`
 - b) `if (x > 0) { /* code */ }`
 - c) `if (x > 0: /* code */)`
7. Как использовать цикл for в C#?
- a) `for (int i = 0; i < 10; i++) { /* code */ }`
 - b) `loop (int i = 0; i < 10; i++) { /* code */ }`
 - c) `foreach (int i = 0; i < 10; i++) { /* code */ }`
8. Что такое оператор switch и как его использовать?
- a) `switch (x) { case 1: /* code */ break; }`
 - b) `if (x) { /* code */ } else { /* code */ }`
 - c) `for (int i = 0; i < x; i++) { /* code */ }`
9. Как объявить функцию (метод) в C#?
- a) `method void MyMethod() { /* code */ }`
 - b) `void MyMethod() { /* code */ }`
 - c) `function MyMethod() { /* code */ }`
10. В чем разница между out и ref в параметрах метода?
- a) `out` используется для передачи данных в метод, а `ref` для получения данных.
 - b) `ref` используется для передачи данных в метод, а `out` для получения данных.
 - c) Нет разницы.
11. Как создать объект класса в C#?
- a) `Class obj = new Class();`
 - b) `new Class obj;`
 - c) `Object.create(Class);`
12. Что такое конструктор и как он используется в C#?
- a) Метод для создания объекта класса, вызывается при его инициализации.
 - b) Метод для удаления объекта класса, вызывается при его уничтожении.
 - c) Метод для изменения состояния объекта класса.
13. Какие основные принципы наследования в C#?
- a) Множественное наследование и полиморфизм.
 - b) Инкапсуляция и абстракция.

c) Иерархия и переопределение.

14. Что такое интерфейс в C# и как его реализовать?

- a) Способ описания класса, не имеющего реализации. Реализуется через ключевое слово `interface`.
- b) Способ описания абстрактного класса. Реализуется через ключевое слово `abstract`.
- c) Способ описания статического класса. Реализуется через ключевое слово `static`.

15. Как обработать исключение (используя блок try-catch)?

- a) `try { /* code */ } catch { /* code */ }`
- b) `if (error) { /* code */ } else { /* code */ }`
- c) `exception (/* code */) { /* code */ }`

16. Какие ключевые слова используются для управления доступом к членам класса?

- a) `public`, `private`, `protected` еще `internal`
- b) `open`, `close`, `read`
- c) `external`, `internal`, `local`

17. Каким образом реализовать множественное наследование в C#?

- a) Используя ключевое слово `multiple`.
- b) Множественное наследование невозможно в C#.
- c) Используя ключевое слово `implements`.

18. Что такое свойства (properties) в C#?

- a) Методы для работы с файловой системой.
- b) Специальные методы для доступа к членам класса.
- c) Переменные, предоставляющие доступ к членам класса.

19. Как использовать индексаторы в C#?

- a) Специальные методы для работы с массивами.
- b) Специальные методы для доступа к элементам коллекции по индексу.
- c) Методы для работы с числами.

20. Как создать статический метод в C#?

- a) `static void MyMethod() { /* code */ }`
- b) `void static MyMethod() { /* code */ }`
- c) `method static void MyMethod() { /* code */ }`

21. Что такое пространство имен (namespace) в C#?

- a) Место для хранения переменных.
- b) Область видимости для именованных элементов, предотвращающая конфликты имен.
- c) Специальная область для хранения комментариев.

22. Как передать параметры в метод по ссылке (by reference)?

- a) Передать параметр с ключевым словом `reference`.
- b) Передать параметр с ключевым словом `ref`.
- c) Передать параметр с ключевым словом `byref`.

23. Как создать перечисление (enum) в C#?

- a) `enum MyEnum { /* values */ }`

b) enum = MyEnum { /* values */ }

c) MyEnum = enum { /* values */ }

24. Как использовать делегаты в C#?

a) delegate MyDelegate() { /* code */ }

b) **delegate void MyDelegate() { /* code */ }**

c) void = delegate MyDelegate() { /* code */ }

25. Что такое события (events) в C#?

a) Методы для обработки исключений.

b) Методы для работы с файлами.

c) **Механизм для уведомления о возникновении событий и их обработки.**

26. Как работает анонимный тип в C#?

a) Тип данных, не имеющий имени и описываемый с помощью ключевого слова anon.

b) **Тип данных, создаваемый на лету без явного определения класса.**

c) Тип данных, используемый для хранения анонимных объектов.

27. Как реализовать свой собственный исключительный тип (exception)?

a) **Создать класс, наследующийся от System.Exception.**

b) Использовать ключевое слово exception.

c) Создать объект класса System.Exception.

28. Как осуществить ввод и вывод данных в консольном приложении C#?

a) console.input() и console.output()

b) **Console.Read() и Console.Write()**

c) input(Console) и output(Console)

29. Как использовать LINQ для запросов к коллекциям в C#?

a) LINQ используется автоматически в запросах к коллекциям.

b) Создать объект LINQ и передать ему коллекцию.

c) **Использовать ключевые слова from, where, select для создания LINQ-запроса.**

30. Что такое асинхронное программирование?

a) Метод программирования, использующий только синхронные вызовы.

b) **Метод программирования, позволяющий выполнять асинхронные (не блокирующие) операции.**

c) Метод программирования, который выполняет все операции параллельно.

31. Как объявляется асинхронный метод в C#?

a) **async void MyMethod() { /* code */ }**

b) void async MyMethod() { /* code */ }

c) **async Task MyMethod() { /* code */ }**

32. Что такое ключевое слово await?

a) Ключевое слово для объявления асинхронного метода.

b) **Ключевое слово для ожидания завершения асинхронной операции.**

c) Ключевое слово для указания, что метод должен быть вызван асинхронно.

33. Какой тип данных возвращает асинхронный метод?

a) void – тоже возвращает

b) **Task**

- c) `async`
- i. ******* (Void, Task, Task<T>)**

34. Как можно организовать последовательность асинхронных операций?

- a) Используя ключевое слово `parallel`.
- b) Используя `Task.Result`.
- c) **Используя ключевое слово `await`.**

35. Что такое Task в контексте асинхронного программирования?

- a) Структура данных для хранения асинхронных операций.
- b) **Класс, представляющий асинхронную операцию и результат её выполнения.**
- c) Тип данных для представления времени выполнения асинхронных методов.

36. Как обрабатывать исключения в асинхронных методах?

- a) Используя блок `try-catch` вокруг всего метода.
- b) Используя ключевое слово `async` для обработки исключений.
- c) **Используя блок `try-catch` вокруг каждой асинхронной операции с `await`.**

37. Что такое асинхронные события (async events)?

- a) События, которые происходят асинхронно в разных потоках.
- b) **Это термин не существует в асинхронном программировании.**
- c) События, которые поддерживают асинхронные делегаты и операции.

38. Как сделать метод асинхронным с возвращаемым значением?

- a) **`async Task<int> MyMethod() { /* code */ }`**
- b) `int async MyMethod() { /* code */ }`
- c) `async MyMethod<int>() { /* code */ }`

39. Как использовать Task.Run для выполнения асинхронной операции в отдельном потоке?

- a) **`Task.Run(() => MyMethod());`**
- b) `MyMethod().RunAsync();`
- c) `Task.Start(MyMethod());`

40. Как отменить асинхронную операцию?

- a) Используя `Task.Abort()`.
- b) **Используя `CancellationToken`.**
- c) Невозможно отменить асинхронную операцию.

41. Что такое async/await паттерн?

- a) **Паттерн для асинхронной реализации с использованием ключевых слов `async/await`.**
- b) Паттерн для синхронной реализации с использованием ключевых слов `sync/wait`.
- c) Паттерн для ожидания выполнения всех асинхронных операций в приложении.

42. Как получить результат из асинхронного метода, возвращающего Task<T>?

- a) `Task<T>.Value`
- b) `await Task<T>`
- c) **`await Task<T>.Result`**

43. Как выполнить несколько асинхронных операций параллельно и дождаться их завершения?

- a) **Используя Task.WaitAll().**
- b) Используя async Task.Parallel().
- c) Используя Task.WhenAll() с await.

44. Что такое ConfigureAwait и для чего он используется?

- a) **Устанавливает, на каком потоке должно продолжаться выполнение после await.**
- b) Устанавливает конфигурацию асинхронной операции.
- c) Отключает конфигурацию для асинхронных операций.

45. Как предотвратить утечки ресурсов при использовании асинхронных операций?

- a) Очистить все ресурсы в блоке finally.
- b) **Использовать using для управления ресурсами.**
- c) Отключить асинхронность.

46. Что такое async void?

- a) Тип возвращаемого значения асинхронного метода.
- b) **Тип, указывающий на отсутствие возвращаемого значения (void) в асинхронном методе.**
- c) Тип асинхронного события.

47. Как передавать параметры в асинхронный метод?

- a) Использовать ключевое слово async перед параметрами метода.
- b) **Параметры передаются так же, как и в синхронных методах.**
- c) Использовать Task.Run для передачи параметров.

48. Как использовать асинхронный конструктор класса?

- a) **Нельзя использовать асинхронные конструкторы.**
- b) Использовать async Task вместо async void.
- c) Использовать асинхронный конструктор, указав async перед ctor.

49. Как определить собственный TaskCompletionSource?

- a) TaskCompletionSource.Create()
- b) **new TaskCompletionSource()**
- c) Task.CompletionSource()

50. Что такое ConfigureAwait(false) и в каком случае его целесообразно использовать?

- a) **Отключает использование потока из пула для продолжения асинхронной операции.**
- b) Включает использование потока из пула для продолжения асинхронной операции.
- c) Отключает асинхронность для операции.

51. Как обрабатывать исключения в асинхронном методе, который возвращает Task<T>?

- a) Используя try-catch вокруг всего метода.
- b) **Используя try-catch вокруг каждого await.**
- c) Используя catch после await.

52. Что такое `async void` метод и когда его следует использовать?

- a) Метод, который выполняется асинхронно и возвращает `void`. Следует использовать для событий.
- b) Метод, который выполняется синхронно и возвращает `void`. Следует использовать для событий.
- c) Метод, который выполняется асинхронно и возвращает `Task`. Следует использовать для событий.

53. Как отменять асинхронные операции с использованием `CancellationToken`?

- a) `CancellationToken.CancelAsync()`
- b) `CancellationToken.ThrowIfCancellationRequested()`
- c) `CancellationTokenSource.Cancel()`

54. Как проверить, завершена ли асинхронная операция?

- a) Использовать `Task.IsCompleted`.
- b) Использовать `Task.IsFinished`.
- c) Использовать `Task.IsDone`.

55. Как обрабатывать несколько исключений в асинхронной операции?

- a) Использовать `catch (Exception ex)`.
- b) Использовать `catch (AggregateException ex)`.
- c) Использовать `catch` после каждого `await`.

56. Как использовать асинхронный делегат (`async delegate`)?

- a) `Task.Run(() => { /* code */ })`
- b) `async delegate { /* code */ }`
- c) `new AsyncDelegate(() => { /* code */ })`

57. Как измерить время выполнения асинхронной операции?

- a) Использовать `DateTime.Now` до и после операции.
- b) Использовать `Stopwatch` класс.
- c) Нельзя измерить время выполнения асинхронной операции.

58. Как прервать выполнение всех асинхронных операций при завершении приложения?

- a) Использовать `Task.WaitAll()`.
- b) Использовать `Task.WhenAll()`.
- c) Использовать `CancellationToken` с отслеживанием завершения приложения.

59. Что означает EF в Entity Framework?

- a) Electronic Forms
- b) Entity Framework
- c) Efficient Functions

60. Какой паттерн программирования реализует Entity Framework?

- a) MVC
- b) MVVM
- c) ORM

61. Что такое "отложенная загрузка" (Lazy Loading) в Entity Framework?

- a) Загрузка данных только при явном запросе.
- b) Автоматическая загрузка всех данных.
- c) Загрузка данных после вызова `SaveChanges`.

62. Как указать первичный ключ в Code First подходе?

- a) [PrimaryKey]
- b) [Key]
- c) [Primary]

63. Каким образом можно удалить запись из базы данных с использованием Entity Framework?

- a) context.Delete(entity)
- b) context.Remove(entity)
- c) context.Entry(entity).Remove()

64. Что такое "Database First" подход в Entity Framework?

- a) Создание базы данных с использованием кода.
- b) Создание кода классов на основе существующей базы данных.
- c) Создание модели базы данных из Entity Framework.

65. Как создать новую базу данных с использованием Entity Framework Code First?

- a) context.CreateDatabase()
- b) context.Database.Create()
- c) context.Database.EnsureCreated()

66. Как указать, что свойство не должно участвовать в маппинге в Code First?

- a) [NotMapped]
- b) [Ignore]
- c) [Exclude]

67. Каким образом можно выполнить "отложенное сохранение" изменений в Entity Framework?

- a) context.SaveChanges()
- b) context.DelayedSaveChanges()
- c) Изменения автоматически сохраняются.

68. Как указать схему для таблицы в Entity Framework Code First?

- a) [Schema("dbo")]
- b) [TableSchema("dbo")]
- c) [Database("dbo")]

69. Как добавить индекс в столбец с использованием Entity Framework Code First?

- a) [Index]
- b) [Key]
- c) [CreateIndex]

70. Что представляет собой класс DbContext в Entity Framework?

- a) Класс для работы с миграциями.
- b) Класс для определения модели данных и взаимодействия с базой данных.
- c) Класс для управления сессиями пользователя.

71. Как удалить все записи из таблицы с использованием Entity Framework?

- a) context.TruncateTable("MyTable")
- b) context.DeleteAll("MyTable")
- c) context.Database.ExecuteSqlCommand("TRUNCATE TABLE MyTable")

72. Как создать индекс для столбца в Entity Framework Code First?

- a) `[Index]`
- b) `[Key]`
- c) `[CreateIndex]`

73. Что такое "Database Context" в Entity Framework?

- a) Объект, предоставляющий доступ к базе данных и управляющий изменениями.
- b) Контекст для работы с изображениями в базе данных.
- c) Объект для выполнения запросов к базе данных.

74. Как указать, что свойство является внешним ключом в Entity Framework Code First?

- a) `[ForeignKey]`
- b) `[Key]`
- c) `[ExternalKey]`

75. Как определить отношение "многие ко многим" в Entity Framework Code First?

- a) `[ManyToMany]`
- b) `[BelongsToMany]`
- c) `[JoinTable]`

76. Как указать, что связь между таблицами является обязательной в Entity Framework Code First?

- a) `[Required]`
- b) `[Mandatory]`
- c) `[NotNull]`

77. Каким образом можно использовать "Raw SQL Queries" в Entity Framework Core?

- a) `context.ExecuteRawSql("SELECT * FROM Table")`
- b) `context.Database.SqlQuery("SELECT * FROM Table")`
- c) `context.RawSqlQuery("SELECT * FROM Table")`

78. Как выполнить "жадную загрузку" связанных данных в Entity Framework Core?

- a) `Include метод.`
- b) `Load метод.`
- c) Автоматическая загрузка всех связанных данных.

79. Какова цель Code-First подхода в Entity Framework?

- a) Автоматическое создание кода из существующей базы данных.
- b) Создание моделей данных сущностей на основе кода, а затем автоматическое создание базы данных.
- c) Визуальное проектирование базы данных.

80. Каким образом Entity Framework обеспечивает работу с транзакциями?

- a) Автоматически управляет транзакциями без дополнительной конфигурации.
- b) Использует явные вызовы методов `BeginTransaction` и `Commit`.
- c) Транзакции не поддерживаются Entity Framework.

Практика

Необходимо разработать программу, содержащую как минимум 3 файла (при необходимости можно добавить свои):

Window.xaml, Window.xaml.cs, WindowViewModel.cs

При разработке программы придерживайтесь паттерна MVVM

Для решения вам могут потребоваться следующие контролы:

<Checkbox>, <Listbox>, <Textbox>, <Image>, <Button>, <Canvas>

<Checkbox>

```
<CheckBox Content="Моя галочка" IsChecked="{Binding IsCheckedValue}" />
```

<Listbox>

```
<ListBox ItemsSource="{Binding ListBoxItems}" SelectedItem="{Binding SelectedListBoxItem}">
    <!-- Для отображения элементов списка используйте DataTemplate -->
    <ListBox.ItemTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding}" />
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

<TextBox>

```
<TextBox Text="{Binding TextValue}" />
```

<Image>

```
<Image Source="/Assets/Image.png" Width="100" Height="100" />
```

<Button>

```
<Button Content="Нажми меня" Command="{Binding ButtonClickCommand}" />
```

<Canvas>

```
<Canvas Width="200" Height="200">
    <Rectangle Canvas.Left="10" Canvas.Top="10" Width="30" Height="30" Fill="Blue" />
    <Ellipse Canvas.Left="50" Canvas.Top="50" Width="30" Height="30" Fill="Red" />
    <!-- Добавьте другие элементы или фигуры, как вам нужно -->
</Canvas>
```

При решении нельзя ссылаться на сторонние библиотеки, за исключением: flurl, community toolkit, reactivex, prism, ef, dapper.

Список дел

Сформируйте контрол со списком дел, которые можно отметить, как завершённые

Создаем структуру данных (модель):

DataModel.cs

```
using System;

namespace ForRR.DataModel
{
    public class ToDoItem
    {
        public string Description { get; set; } = String.Empty;
        public bool IsChecked { get; set; }
    }
}
```

Создаем новый View в папке Views (ToDoListView.axaml с зависимым

ToDoListView.axaml.cs), в зависимости надо добавить

xmlns:vm="using:ForRR.ViewModels", x:DataType="vm:ToDoListViewModel":

ToDoList.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.ToDoListView"
              x:DataType="vm:ToDoListViewModel">
    <DockPanel>
        <ItemsControl ItemsSource="{Binding ListItems}">
            <ItemsControl.ItemTemplate>
                <DataTemplate>
                    <CheckBox Margin="4"
                        IsChecked="{Binding IsChecked}"
                        Content="{Binding Description}"/>
                </DataTemplate>
            </ItemsControl.ItemTemplate>
        </ItemsControl>
    </DockPanel>
</UserControl>
```

ToDoList.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class ToDoListView : UserControl
    {
        public ToDoListView()
        {
            InitializeComponent();
        }
    }
}
```

```

    }
}
}

```

Нужно создать ViewModel и Service, который будет обрабатывать объекты созданной структуры данных.

ToDoListService.cs

```

using System.Collections.Generic;
using ForRR.DataModel;

namespace ForRR.Services
{
    public class ToDoListService
    {
        public IEnumerable<ToDoItem> GetItems() => new[]
        {
            new ToDoItem { Description = "Watch Magnificent Century" },
            new ToDoItem { Description = "Buy some jewellery" }
        };
    }
}

```

ToDoListViewModel.cs

```

using System.Collections.Generic;
using System.Collections.ObjectModel;
using ForRR.DataModel;

namespace ForRR.ViewModels
{
    public class ToDoListViewModel : ViewModelBase
    {
        public ToDoListViewModel(IEnumerable<ToDoItem> items)
        {
            ListItems = new ObservableCollection<ToDoItem>(items);
        }

        public ObservableCollection<ToDoItem> ListItems { get; }
    }
}

```

Теперь в MainWindow.axaml добавляем следующий импорт:

```
xmlns:views = "clr-namespace:ForRR.Views"
```

И добавляем новый View на экран, код такой:

MainWindow.axaml

```

<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:vm="using:ForRR.ViewModels"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:views = "clr-namespace:ForRR.Views"
        xmlns:viewModels="clr-namespace:ToDoList.ViewModels"
        mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

```

```

        x:Class="ForRR.Views.MainWindow"
        x:DataType="vm:MainWindowViewModel"

        Icon="/Assets/avalonia-logo.ico"
        Title="ForRR">

        <views:ToDoListView DataContext="{Binding ToDoList}" />
</Window>

```

MainWindow.axaml.cs

```

using Avalonia.Controls;

namespace ForRR.Views;

public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }
}

```

В главном ViewModel следующий код:

MainWindowViewModel.cs

```

using ForRR.Services;

namespace ForRR.ViewModels
{
    public class MainWindowViewModel : ViewModelBase
    {
        //this has a dependency on the ToDoListService

        public MainWindowViewModel()
        {
            var service = new ToDoListService();
            ToDoList = new ToDoListViewModel(service.GetItems());
        }

        public ToDoListViewModel ToDoList { get; }
    }
}

```

Остальной код на всякий случай:

ViewModelBase.cs

```

using ReactiveUI;

namespace ForRR.ViewModels;

public class ViewModelBase : ReactiveObject
{
}

```

Остальное генерируется автоматически и приводить код App.axaml, ViewLocator.cs, Program.cs и тд смысла нет

Проверка совершеннолетия

Сформируйте контрол с вводом даты, определяющий совершеннолетие пользователя

DateCheckView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.DateCheckView"
              x:DataType="vm:DateCheckViewModel">
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <DatePicker x:Name="BirthDatePicker" Margin="4"
SelectedDateChanged="DatePicker_OnSelectedDateChanged"/>
        <TextBlock Text="{Binding AgeResult}" Margin="4"/>
    </StackPanel>
</UserControl>
```

DateCheckView.axaml.cs (тут добавляется обработчик изменения даты):

```
using System;
using System.ComponentModel;
using Avalonia.Controls;
using ForRR.ViewModels;

namespace ForRR.Views
{
    public partial class DateCheckView : UserControl, INotifyPropertyChanged
    {
        public static DateTime SelectedDate;
        public DateCheckView()
        {
            InitializeComponent();
            DataContext = new DateCheckViewModel();
        }
        private void DatePicker_OnSelectedDateChanged(object? sender,
DatePickerSelectedValueChangedEventArgs e)
        {
            if (sender is DatePicker datePicker && datePicker.SelectedDate !=
null)
            {
                var selectedDateTime =
datePicker.SelectedDate.Value.LocalDateTime;
                if (DataContext is DateCheckViewModel viewModel)
                {
                    viewModel.SelectedDate = selectedDateTime;
                    viewModel.CountAge(viewModel.SelectedDate);
                }
            }
        }
    }
}
```



```

    }
}
}
}

```

DateCheckViewModel.cs

```

using ReactiveUI;
using System;
using System.ComponentModel;

namespace ForRR.ViewModels
{
    public class DateCheckViewModel : ViewModelBase, INotifyPropertyChanged
    {
        private DateTime _selectedDate;
        private string _ageResult;

        public DateTime SelectedDate
        {
            get => _selectedDate;
            set => this.RaiseAndSetIfChanged(ref _selectedDate, value);
        }
        public string AgeResult
        {
            get => _ageResult;
            set => this.RaiseAndSetIfChanged(ref _ageResult, value);
        }

        public void CountAge(DateTime Birth)
        {
            int age = DateTime.Now.Year - Birth.Year;
            if (DateTime.Now.Month < Birth.Month || (DateTime.Now.Month ==
Birth.Month && DateTime.Now.Day < Birth.Day))
            {
                age--;
            }

            AgeResult = age >= 18 ? $"Совершеннолетний: {age}" :
$"Несовершеннолетний: {age}";
        }

        public DateCheckViewModel()
        {
        }
    }
}

```

MainWindowView.axaml – прошлое меняется на это

```
<views:DateCheckView DataContext="{Binding DateCheck}"/>
```

MainWindowView.axaml.cs – без изменений

```
using Avalonia.Controls;
```

```
namespace ForRR.Views;

public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }
}
```

MainWindowViewModel.cs

```
using System.ComponentModel;
using ForRR.Services;
namespace ForRR.ViewModels
{
    public class MainWindowViewModel : ViewModelBase, INotifyPropertyChanged
    {
        public MainWindowViewModel()
        {
            DateCheck = new DateCheckViewModel();
        }
        public DateCheckViewModel DateCheck { get; }
    }
}
```

Часы

Сформируйте контрол, визуализирующий часы

ClockView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.ClockView"
              x:DataType="vm:ClockViewModel">

    <!-- Как-то так -->
    <Canvas>
        <Ellipse Width="200" Height="200" Stroke="Black" StrokeThickness="2"
Canvas.Left="0" Canvas.Top="65"/>
        <Rectangle x:Name="hourHand" Width="5" Height="40" Fill="Black"
Canvas.Left="100" Canvas.Top="130"/>
        <Rectangle x:Name="minuteHand" Width="3" Height="60" Fill="Purple"
Canvas.Left="100" Canvas.Top="110"/>
        <Rectangle x:Name="secondHand" Width="1" Height="70" Fill="Red"
Canvas.Left="100" Canvas.Top="99.5"/>
    </Canvas>
</UserControl>
```

ClockView.axaml.cs (надо отслеживать изменения, так что пишем тут)

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Media;
using Avalonia.Threading;

namespace ForRR.Views
{
    public partial class ClockView : UserControl
    {
        private DispatcherTimer timer;

        public ClockView()
        {
            InitializeComponent();
            timer = new DispatcherTimer();
            timer.Tick += Timer_Tick;
            timer.Interval = TimeSpan.FromSeconds(1);
            timer.Start();
        }

        private void Timer_Tick(object sender, EventArgs e)
        {
            DateTime currentTime = DateTime.Now;
```

```

        int hour = currentTime.Hour % 12;
        int minute = currentTime.Minute;
        int second = currentTime.Second;

        double hourAngle = hour * 30 + minute * 0.5;
        double minuteAngle = minute * 6;
        double secondAngle = second * 6;

        hourHand.RenderTransformOrigin = new RelativePoint(1, 1,
RelativeUnit.Relative);
        hourHand.RenderTransform = new RotateTransform(hourAngle, 0, 0);

        minuteHand.RenderTransformOrigin = new RelativePoint(1, 1,
RelativeUnit.Relative);
        minuteHand.RenderTransform = new RotateTransform(minuteAngle, 0,
0);

        secondHand.RenderTransformOrigin = new RelativePoint(1, 1,
RelativeUnit.Relative);
        secondHand.RenderTransform = new RotateTransform(secondAngle, 0,
0);

    }

}
}

```

ClockViewModel.cs (Вытаскиваем отслеживание)

```

using System;
using System.ComponentModel;
using System.Timers;

namespace ForRR.ViewModels
{
    public class ClockViewModel : INotifyPropertyChanged
    {
        private double _hourHandAngle;
        private double _minuteHandAngle;
        private double _secondHandAngle;

        public double HourHandAngle
        {
            get => _hourHandAngle;
            set
            {
                if (_hourHandAngle != value)
                {
                    _hourHandAngle = value;
                    OnPropertyChanged(nameof(HourHandAngle));
                }
            }
        }

        public double MinuteHandAngle
        {
            get => _minuteHandAngle;

```

```

        set
        {
            if (_minuteHandAngle != value)
            {
                _minuteHandAngle = value;
                OnPropertyChanged(nameof(MinuteHandAngle));
            }
        }
    }

    public double SecondHandAngle
    {
        get => _secondHandAngle;
        set
        {
            if (_secondHandAngle != value)
            {
                _secondHandAngle = value;
                OnPropertyChanged(nameof(SecondHandAngle));
            }
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;

    private void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new
        PropertyChangedEventArgs(propertyName));
    }

    public ClockViewModel()
    {
        var timer = new Timer(1000);
        timer.Elapsed += Timer_Elapsed;
        timer.Enabled = true;
    }

    private void Timer_Elapsed(object sender, ElapsedEventArgs e)
    {
        DateTime currentTime = DateTime.Now;
        int hour = currentTime.Hour % 12;
        int minute = currentTime.Minute;
        int second = currentTime.Second;

        HourHandAngle = hour * 30 + minute * 0.5;
        MinuteHandAngle = minute * 6;
        SecondHandAngle = second * 6;
    }
}

```

MainWindowView.axaml (тупо пишем сюда эту строчку)

```
<views:ClockView DataContext="{Binding Clock}"/>
```

MainWindowViewModel.cs (меняем только основной метод, остальное в предыдущих ответах)

```
public MainWindowViewModel()  
{  
    Clock = new ClockViewModel();  
}  
public ClockViewModel Clock { get; }
```

Поиск слов

Сформируйте контрол, находящий все слова, начинающиеся с выбранной части

WordPartSearchView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.WordPartSearchView"
              x:DataType="vm:WordPartSearchViewModel">
    <StackPanel>
        <TextBox x:Name="SearchTextBox" Margin="10"
        TextChanged="SearchBox_Changed"/>
        <TextBlock Text = "Найденные слова"/>
        <ListBox ItemsSource="{Binding FoundList}"/>
    </StackPanel>
</UserControl>
```

WordPartSearch.axaml.cs

```
using Avalonia.Controls;
using ForRR.ViewModels;

namespace ForRR.Views
{
    public partial class WordPartSearchView : UserControl
    {
        public WordPartSearchView()
        {
            InitializeComponent();
        }

        private void SearchBox_Changed(object? sender, TextChangedEventArgs e)
        {
            if (sender is TextBox txt && txt.Text != null)
            {
                var to_search = SearchTextBox.Text;

                if (DataContext is WordPartSearchViewModel viewModel)
                {
                    viewModel.TextToSearch = to_search;
                    viewModel.FindWords();
                }
            }
        }
    }
}
```

WordPartSearchViewModel.cs

```

using System.Collections.Generic;
using System.ComponentModel;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class WordPartSearchViewModel : ViewModelBase,
    INotifyPropertyChanged
    {
        public List<string> Words = new List<string> { "енот", "енотик",
"помогите", "пожалуйста", "пельмени", "Траляля", "труляля", "хихихиха",
"хачапури"};
        private string _textToSearch;
        private List<string> _foundList = new List<string>();

        public string TextToSearch
        {
            get => _textToSearch;
            set => this.RaiseAndSetIfChanged(ref _textToSearch, value);
        }

        public List<string> FoundList
        {
            get => _foundList;
            set => this.RaiseAndSetIfChanged(ref _foundList, value);
        }

        public void FindWords()
        {
            List<string> filteredWords = new List<string>();

            foreach (string word in Words)
            {
                if (word.ToLower().StartsWith(TextToSearch.ToLower()) &&
TextToSearch != "")
                {
                    filteredWords.Add(word);
                }
            }

            FoundList = filteredWords;
        }
        public WordPartSearchViewModel() {}
    }
}

```

MainWindowView.axaml

```
<views:WordPartSearchView DataContext="{Binding WordPart}"/>
```

MainWindowViewModel.cs

```

public class MainWindowViewModel : ViewModelBase, INotifyPropertyChanged
{
    public MainWindowViewModel()
    {
        WordPart = new WordPartSearchViewModel();
    }
}

```



```
public WordPartSearchViewModel WordPart { get; }  
}
```

Сумма 2х чисел

Сформируйте контрол, находящий сумму 2х чисел

AdditionView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.AdditionView"
              x:DataType="vm:AdditionViewModel">
    <StackPanel>
        <TextBox x:Name = "FirstNumber" Margin="5" Text = "{Binding FirstValue}" />
        <TextBox x:Name="SecondNumber" Margin="5" Text = "{Binding SecondValue}" />
        <Button Content="Результат" Command="{Binding GetAdditionResult}" />
        <TextBlock x:Name="Result" Margin="5" Text="{Binding Result}" />
    </StackPanel>
</UserControl>
```

AdditionView.axaml.cs

```
using System.ComponentModel;
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class AdditionView: UserControl, INotifyPropertyChanged
    {
        public AdditionView()
        {
            InitializeComponent();
        }
    }
}
```

AdditionViewModel.cs

```
using System.ComponentModel;
using ReactiveUI;

namespace ForRR.ViewModels;

public class AdditionViewModel: ViewModelBase, INotifyPropertyChanged
{
    private string _firstValue;
    private string _secondValue;
    private string _result;

    public string FirstValue
    {
        get => _firstValue;
        set => this.RaiseAndSetIfChanged(ref _firstValue, value);
    }

    public string SecondValue
```

```

    {
        get => _secondValue;
        set => this.RaiseAndSetIfChanged(ref _secondValue, value);
    }

    public string Result
    {
        get => _result;
        set => this.RaiseAndSetIfChanged(ref _result, value);
    }

    public void GetAdditionResult()
    {
        if (FirstValue != null && SecondValue != null)
        {
            int a, b;
            int.TryParse(FirstValue, out a);
            int.TryParse(SecondValue, out b);
            if (int.TryParse(FirstValue, out a) && int.TryParse(SecondValue,
out b))
            {
                Result = (a + b).ToString();
            }
        }
    }
    public AdditionViewModel() {}
}

```

MainWindowView.axaml

```
<views:AdditionView DataContext="{Binding Addition}"/>
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    Addition = new AdditionViewModel();
}
public AdditionViewModel Addition{ get; }

```

Текст из файла

Сформируйте контрол, который выводит весь текст файла из переданного пути

FileReaderView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:vm="using:ForRR.ViewModels"
  xmlns:sys="clr-namespace:System;assembly=System.Runtime"
  mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
  x:Class="ForRR.Views.FileReaderView"
  x:DataType="vm:FileReaderViewModel">
  <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
    <TextBox x:Name="FileInput" TextChanged="FilePath_Changed"/>
    <TextBox x:Name="FileOutput" Text="{Binding FileText}"/>
  </StackPanel>
</UserControl>
```

FileReaderView.axaml.cs

```
using Avalonia.Controls;
using ForRR.ViewModels;

namespace ForRR.Views
{
    public partial class FileReaderView : UserControl
    {
        public FileReaderView()
        {
            InitializeComponent();
        }

        private void FilePath_Changed(object? sender, TextChangedEventArgs e)
        {
            if (sender is TextBox txt && txt.Text != null)
            {
                var file_path = FileInput.Text;

                if (DataContext is FileReaderViewModel viewModel)
                {
                    viewModel.FilePath = file_path;
                    viewModel.ReadMyFile();
                }
            }
        }
    }
}
```

FileReaderViewModel.cs

```
using System.ComponentModel;
using System.IO;
```

```

using System.Text;
using ReactiveUI;

// /Users/tatiana/CLionProjects/compMath1/1.txt например это
namespace ForRR.ViewModels
{
    public class FileReaderViewModel : ViewModelBase, INotifyPropertyChanged
    {
        private string _filePath;
        private string _fileText;

        public string FilePath
        {
            get => _filePath;
            set => this.RaiseAndSetIfChanged(ref _filePath, value);
        }

        public string FileText
        {
            get => _fileText;
            set => this.RaiseAndSetIfChanged(ref _fileText, value);
        }

        public void ReadMyFile()
        {
            FileText = File.Exists(FilePath) ? File.ReadAllText(FilePath,
Encoding.ASCII) : "Такого файла нет";
        }
        public FileReaderViewModel() {}
    }
}

```

MainWindowView.axaml

```
<views:FileReaderView DataContext="{Binding FileReader}" />
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    FileReader = new FileReaderViewModel();
}
public FileReaderViewModel FileReader{ get; }

```

Первый различный символ в 2х текстах

Сформируйте контрол, который находит первый различный символ в 2х текстах

TextDiffView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.TextDiffView"
              x:DataType="vm:TextDiffViewModel">
    <DockPanel >
        <Button Content = "Найти различие" Command="{Binding FindDiffs}"/>
        <TextBox x:Name = "TextOne" Text="{Binding FirstText}"/>
        <TextBox x:Name = "TextTwo" Text="{Binding SecondText}"/>
        <TextBlock Text="{Binding DiffResult}"/>
    </DockPanel>
</UserControl>
```

TextDiffView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class TextDiffView : UserControl
    {
        public TextDiffView()
        {
            InitializeComponent();
        }
    }
}
```

TextDiffViewModel.cs

```
using System;
using System.ComponentModel;
using ReactiveUI;

namespace ForRR.ViewModels;

public class TextDiffViewModel: ViewModelBase, INotifyPropertyChanged
{
    private string _firstText;
    private string _secondText;
    private string _diffResult;

    public string FirstText
    {
        get => _firstText;
        set => this.RaiseAndSetIfChanged(ref _firstText, value);
    }
}
```

```

    }

    public string SecondText
    {
        get => _secondText;
        set => this.RaiseAndSetIfChanged(ref _secondText, value);
    }

    public string DiffResult
    {
        get => _diffResult;
        set => this.RaiseAndSetIfChanged(ref _diffResult, value);
    }

    public void FindDiffs()
    {
        int minLength = Math.Min(FirstText.Length, SecondText.Length);

        for (int i = 0; i < minLength; i++)
        {
            if (FirstText[i] != SecondText[i])
            {
                DiffResult = $"Первый различный символ на позиции {i + 1},  

символ 1: {FirstText[i]}, символ 2: {SecondText[i]}";
                return;
            }
        }

        DiffResult = "Тексты полностью совпадают";
    }

    public TextDiffViewModel() {}
}

```

MainWindowView.axaml

```
<views:TextDiffView DataContext="{Binding TextDiff}"/>
```

MainWindowViewModel.cs

```

public class MainWindowViewModel : ViewModelBase, INotifyPropertyChanged
{
    public MainWindowViewModel()
    {
        TextDiff = new TextDiffViewModel();
    }

    public TextDiffViewModel TextDiff { get; }
}

```

Количество слов в тексте

Сформируйте контрол, который считает количество слов во вставленном тексте

CountWordsView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.CountWordsView"
              x:DataType="vm:CountWordsViewModel">
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <TextBox Text = "{Binding MyText}"/>
        <Button Content="Посчитать слова" Command="{Binding CountWordsInText}"/>
        <TextBlock Text="{Binding ResultCount}"/>
    </StackPanel>
</UserControl>
```

CountWordsView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class CountWordsView : UserControl
    {
        public CountWordsView()
        {
            InitializeComponent();
        }
    }
}
```

CountWordsViewModel.cs

```
using System.ComponentModel;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class CountWordsViewModel: ViewModelBase, INotifyPropertyChanged
    {
        private string _myText;
        private string _resultCount;

        public string MyText
        {
            get => _myText;
        }
    }
}
```



```

        set => this.RaiseAndSetIfChanged(ref _myText, value);
    }

    public string ResultCount
    {
        get => _resultCount;
        set => this.RaiseAndSetIfChanged(ref _resultCount, value);
    }

    public void CountWordsInText()
    {
        string text = MyText.Trim();
        int wordCount = 0;
        bool inWord = false;

        foreach (char c in text)
        {
            if (char.IsWhiteSpace(c) || char.IsPunctuation(c))
            {
                inWord = false;
            }
            else if (!inWord)
            {
                wordCount++;
                inWord = true;
            }
        }

        ResultCount = wordCount.ToString();
    }
    public CountWordsViewModel() {}
}

```

MainWindowView.axaml

```
<views:CountWordsView DataContext="{Binding CountWords}"/>
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    CountWords = new CountWordsViewModel();
}
public CountWordsViewModel CountWords { get; }

```

Рисование точки по клику

Сформируйте контрол, который в месте клика рисует красную точку

PointDrawView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.PointDrawView"
              x:DataType="vm:PointDrawViewModel">

    <Canvas PointerPressed="CreatePoint" Background="White">
        <Ellipse x:Name="dot" Fill="Red" Width="10" Height="10"
Canvas.Left="0" Canvas.Top="0" Opacity="0"/>
    </Canvas>
</UserControl>
```

PointDrawView.axaml.cs

```
using Avalonia;
using Avalonia.Controls;
using Avalonia.Input;

namespace ForRR.Views
{
    public partial class PointDrawView : UserControl
    {
        private Canvas pointDrawer;
        public PointDrawView()
        {
            InitializeComponent();
            pointDrawer = this.FindControl<Canvas>("PointDrawer");
        }

        private void CreatePoint(object? sender, PointerPressedEventArgs e)
        {
            if (sender is Canvas canvas)
            {
                var position = e.GetPosition(canvas);
                SetCanvasPosition(dot, position);
            }
        }

        private void SetCanvasPosition(Control control, Point position)
        {
            control.SetValue(Canvas.LeftProperty, position.X);
            control.SetValue(Canvas.TopProperty, position.Y);
            control.Opacity = 1;
        }
    }
}
```

PointDrawViewModel.cs

```
using System.ComponentModel;
```

```
namespace ForRR.ViewModels
{
    public class PointDrawViewModel : ViewModelBase, INotifyPropertyChanged
    {
        public PointDrawViewModel(){}
    }
}
```

MainWindowView.axaml

```
<views:PointDrawView DataContext="{Binding PointDraw}" />
```

MainWindowViewModel.cs

```
public MainWindowViewModel()
{
    PointDraw = new PointDrawViewModel();
}
public PointDrawViewModel PointDraw { get; }
```

Изображения из папки

Сформируйте контрол, визуализирующий все картинки из папки
изображения пользователя

ImageReaderView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.ImageReaderView"
              x:DataType="vm:ImageReaderViewModel">
    <StackPanel>
        <TextBox x:Name="DirPathText" Text="{Binding DirPath}"/>
        <Button Content="Показать фото" Command="{Binding GetPhotos}"/>
        <ListBox x:Name="ImagesFromDir" ItemsSource="{Binding Photos}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <Image Source="{Binding}" Width="100" Height="100"/>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </StackPanel>
</UserControl>
```

ImageReaderView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class ImageReaderView : UserControl
    {
        public ImageReaderView()
        {
            InitializeComponent();
        }
    }
}
```

ImageReaderViewModel.cs

```
using System.Collections.Generic;
using System.ComponentModel;
using Avalonia.Media.Imaging;
using System.IO;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class ImageReaderViewModel : ViewModelBase, INotifyPropertyChanged
    {
        private string _dirPath;
```

```

private List<Bitmap> _photos = new List<Bitmap>();

public string DirPath
{
    get => _dirPath;
    set => this.RaiseAndSetIfChanged(ref _dirPath, value);
}

public List<Bitmap> Photos
{
    get => _photos;
    set => this.RaiseAndSetIfChanged(ref _photos, value);
}

public void GetPhotos()
{
    var imgPaths = Directory.GetFiles(DirPath, "*.png");

    List<Bitmap> ph = new List<Bitmap>();
    foreach (var path in imgPaths)
    {
        var bitmap = new Bitmap(path);
        ph.Add(bitmap);
    }

    Photos = ph;
}

public ImageReaderViewModel() {}
}

```

MainWindowView.axaml

```
<views:ImageReaderView DataContext="{Binding ImageReader}"/>
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    ImageReader = new ImageReaderViewModel();
}

public ImageReaderViewModel ImageReader { get; }

```

Изображение по ссылке

Сформируйте контрол, визуализирующий изображение по переданной ссылке

ImageLoaderView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.ImageLoaderView"
              x:DataType="vm:ImageLoaderViewModel">
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <TextBox x:Name="Link" Text="{Binding ImageLink}"/>
        <Button Content="Посмотреть" Command="{Binding LoadFromWeb}"/>
        <Image Source="{Binding ImgSource}" Width="100" Height="100"/>
    </StackPanel>
</UserControl>
```

ImageLoaderView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class ImageLoaderView : UserControl
    {
        public ImageLoaderView()
        {
            InitializeComponent();
        }
    }
}
```

ImageLoaderViewModel.cs

```
using System;
using System.ComponentModel;
using System.Net;
using System.Threading.Tasks;
using Avalonia.Media.Imaging;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class ImageLoaderViewModel : ViewModelBase, INotifyPropertyChanged
    {
        private string _imageLink;
        private Bitmap _imgSource;

        public string ImageLink
        {

```

```

        get => _imageLink;
        set => this.RaiseAndSetIfChanged(ref _imageLink, value);
    }

    public Bitmap ImgSource
    {
        get => _imgSource;
        set => this.RaiseAndSetIfChanged(ref _imgSource, value);
    }

    public async Task LoadFromWeb()
    {
        if (ImageLink != null && ImageLink != " ")
        {
            using (var webClient = new WebClient())
            {
                byte[] data = await webClient.DownloadDataTaskAsync(new
Uri(ImageLink));
                ImgSource = new Bitmap(new System.IO.MemoryStream(data));
            }
        }
    }

    public ImageLoaderViewModel()
    {
    }
}

```

MainWindowView.axaml

```
<views:ImageLoaderView DataContext="{Binding ImageLoader}"/>
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    ImageLoader = new ImageLoaderViewModel();
}

public ImageLoaderViewModel ImageLoader { get; }

```

Визуализация Цвета

Сформируйте контрол, визуализирующий цвет по введенному RGB

ColorScreenView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.ColorScreenView"
              x:DataType="vm:ColorScreenViewModel">
    <DockPanel Background="{Binding ColorHex}">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center" >
            <TextBox x:Name="RedBox" Text = "{Binding RedValue}"
                Watermark="Red"/>
            <TextBox x:Name="GreenBox" Text = "{Binding GreenValue}"
                Watermark="Green"/>
            <TextBox x:Name="BlueBox" Text = "{Binding BlueValue}"
                Watermark="Blue"/>
        </StackPanel>
    </DockPanel>
</UserControl>
```

ColorScreenView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class ColorScreenView: UserControl
    {
        public ColorScreenView()
        {
            InitializeComponent();
        }
    }
}
```

ColorScreenViewModel.cs

```
using System.ComponentModel;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class ColorScreenViewModel: ViewModelBase, INotifyPropertyChanged
    {
        private string _redValue;
        private string _blueValue;
        private string _greenValue;
        private string _colorHex = "#FFFFFF";

        public string RedValue
        {

```



```

        get => _redValue;
        set => this.RaiseAndSetIfChanged(ref _redValue, value);
    }
    public string BlueValue
    {
        get => _blueValue;
        set => this.RaiseAndSetIfChanged(ref _blueValue, value);
    }
    public string GreenValue
    {
        get => _greenValue;
        set => this.RaiseAndSetIfChanged(ref _greenValue, value);
    }
    public string ColorHex
    {
        get => _colorHex;
        set => this.RaiseAndSetIfChanged(ref _colorHex, value);
    }

    public void AddColor()
    {
        int r, g, b;
        if (int.TryParse(RedValue, out r) && int.TryParse(BlueValue, out
b) && int.TryParse(GreenValue, out g))
        {
            ColorHex = $"#{r:X2}{g:X2}{b:X2}";
        }
    }

    public ColorScreenViewModel() {}
}

```

MainWindowView.axaml

```
<views:ColorScreenView DataContext="{Binding ColorScreen}"/>
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    ColorScreen = new ColorScreenViewModel();
}
public ColorScreenViewModel ColorScreen { get; }

```

Список с перемещением

Сформируйте контрол, отображающий список с возможностью переместить выбранный элемент вверх

MoveItemsView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.MoveItemsView"
              x:DataType="vm:MoveItemsViewModel">
    <StackPanel >
        <ListBox x:Name="MovingList" ItemsSource="{Binding MyList}"
        SelectedIndex="{Binding SIndex}"/>
        <Button Content = "Подвинуть вверх" Command="{Binding MoveUp}"/>
    </StackPanel>
</UserControl>
```

MoveItemsView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class MoveItemsView: UserControl
    {
        public MoveItemsView()
        {
            InitializeComponent();
        }
    }
}
```

MoveItemsViewModel.cs

```
using System.Collections.ObjectModel;
using System.ComponentModel;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class MoveItemsViewModel: ViewModelBase, INotifyPropertyChanged
    {
        private ObservableCollection<string> _myList = new
        ObservableCollection<string> {"Item1", "Item2", "Item3", "Item4", "Item5"};
        private int _sIndex = 1;

        public ObservableCollection<string> MyList
        {
            get => _myList;
            set => this.RaiseAndSetIfChanged(ref _myList, value);
        }
    }
}
```

```

    public int SIndex
    {
        get => _sIndex;
        set => this.RaiseAndSetIfChanged(ref _sIndex, value);
    }

    public void MoveUp()
    {
        if (SIndex > 0 && SIndex < MyList.Count)
        {
            var item = MyList[SIndex];
            var tmp = MyList[SIndex - 1];
            MyList[SIndex - 1] = item;
            MyList[SIndex] = tmp;
        }
    }

    public MoveItemsViewModel()
    {}
}
}

```

MainWindowView.axaml

```
<views:MoveItemsView DataContext="{Binding MoveItems}"/>
```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    MoveItems = new MoveItemsViewModel();
}

public MoveItemsViewModel MoveItems { get; }

```

Пагинация

Сформируйте контрол, поддерживающий пагинацию

PagingView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.PagingView"
              x:DataType="vm:PagingViewModel">
    <StackPanel >
        <ListBox x:Name="ItemsListBox" Margin="10" ItemsSource="{Binding PageItems}" SelectionMode="Single" Height="200"/>
        <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Bottom">
            <Button Content="Назад" Margin="5" Command="{Binding GoBack}" />
            <TextBlock Text="{Binding CurrentPage}" Margin="5"/>
            <Button Content="Вперед" Margin="5" Command="{Binding GoNext}" />
        </StackPanel>
    </StackPanel>
</UserControl>
```

PagingView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{
    public partial class PagingView : UserControl
    {
        public PagingView()
        {
            InitializeComponent();
        }
    }
}
```

PagingViewModel.cs

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;

namespace ForRR.ViewModels
{
    public class PagingViewModel: ViewModelBase, INotifyPropertyChanged
    {
        //когда меняется текущая страничка, ловим Property Change, чтобы заново загрузить элементы вместе с этим
        public event PropertyChangedEventHandler PropertyChanged;
        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

```

private int _currentPage = 1;
private const int ElementAmount = 5;
private List<string> ItemList = new List<string>{"Item1", "Item2",
"Item3", "Item4", "Item5", "Item6", "Item7", "Item8", "Item9", "Item10"};

public int CurrentPage
{
    get => _currentPage;
    set
    {
        _currentPage = value;
        LoadItems();
        OnPropertyChanged(nameof(CurrentPage));
    }
}

public void GoNext()
{
    if (CurrentPage * ElementAmount < ItemList.Count)
    {
        CurrentPage++;
    }
}

public void GoBack()
{
    if (CurrentPage > 1)
    {
        CurrentPage--;
    }
}

//Кусок всего списка, доступный для отображения
public ObservableCollection<string> PageItems { get; set; } = new
ObservableCollection<string>();
private void LoadItems()
{
    int startIndex = (CurrentPage - 1) * ElementAmount;
    int count = ElementAmount;
    PageItems.Clear();
    for (int i = startIndex; i < startIndex + count && i <
ItemList.Count; i++)
    {
        PageItems.Add(ItemList[i]);
    }
}

public PagingViewModel()
{
    LoadItems();
}
}

```

MainWindowView.axaml

```
<views:PagingView DataContext="{Binding Paging}"/>
```

MainWindowViewModel.cs

```
public MainWindowViewModel()  
{  
    Paging = new PagingViewModel();  
}  
public PagingViewModel Paging { get; }
```

Номер телефона

Сформируйте контрол, позволяющий ввести только номер телефона

PhoneView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              xmlns:sys="clr-namespace:System;assembly=System.Runtime"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.PhoneView"
              x:DataType="vm:PhoneViewModel">
    <StackPanel >
        <TextBox x:Name="PhoneBox" KeyUp="KeyChecker" TextChanged="Letters"
        MaxLength="12"/>
    </StackPanel>
</UserControl>
```

PhoneView.axaml.cs

```
using Avalonia.Controls;
using Avalonia.Input;

namespace ForRR.Views
{
    public partial class PhoneView : UserControl
    {
        public PhoneView()
        {
            InitializeComponent();
        }

        private void KeyChecker(object? sender, KeyEventArgs e)
        {
            if (sender is TextBox txt)
            {
                string text = txt.Text;
                if (!text.StartsWith('+'))
                {
                    txt.Text = "+";
                    txt.CaretIndex = 1;
                }
            }
        }

        private void LetterChecker(object? sender, TextChangedEventArgs e)
        {
            if (sender is TextBox txt && txt.Text.Length > 1)
            {
                string text = txt.Text;
                char sym = text[text.Length - 1];
                var h = char.IsNumber(sym);
                if (!char.IsNumber(sym))
                {
                    txt.Text = text.Remove(text.Length - 1);
                }
            }
        }
    }
}
```

```
    }  
    }  
    }  
}
```

PhoneViewModel.cs

```
using System.ComponentModel;  
  
namespace ForRR.ViewModels  
{  
    public class PhoneViewModel : ViewModelBase, INotifyPropertyChanged  
    {  
        public PhoneViewModel() {}  
    }  
}
```

MainWindowView.axaml

```
<views:PhoneView DataContext="{Binding Phone}"/>
```

MainWindowViewModel.cs

```
public MainWindowViewModel()  
{  
    Phone = new PhoneViewModel();  
}  
public PhoneViewModel Phone { get; }
```


Сообщения

Сформируйте контрол, визуализирующий сообщения пользователя с его именем и аватаром

MessageModel.cs

```
using System;
using Avalonia.Media.Imaging;

namespace ForRR.DataModel
{
    public class MessageModel
    {
        public string Name { get; set; } = "Tatiana";
        public Bitmap Photo { get; set; } = new
        Bitmap("/Users/tatiana/Desktop/images/hihi.png");
        public string MessageText { get; set; } = String.Empty;
    }
}
```

UserMessageView.axaml

```
<UserControl xmlns="https://github.com/avaloniaui"
              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
              xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
              xmlns:vm="using:ForRR.ViewModels"
              mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"
              x:Class="ForRR.Views.UserMessageView"
              x:DataType="vm:UserMessageViewModel">
    <StackPanel>
        <TextBox Watermark="Напишите сообщение" Text="{Binding InputVal}"/>
        <Button Content="Отправить" Command="{Binding Send}"/>
        <ListBox ItemsSource="{Binding MessageList}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal">
                        <Image Source="{Binding Photo}" Height="50"
Width="50"/>
                        <StackPanel>
                            <TextBlock Text="{Binding Name}"
Foreground="Indigo"/>
                            <TextBlock Text="{Binding MessageText}"/>
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </StackPanel>
</UserControl>
```

UserMessageView.axaml.cs

```
using Avalonia.Controls;

namespace ForRR.Views
{

```

```

public partial class UserMessageView : UserControl
{
    public UserMessageView()
    {
        InitializeComponent();
    }
}

```

UserMessageViewModel.cs

```

using System.Collections.ObjectModel;
using System.ComponentModel;
using ForRR.DataModel;
using ReactiveUI;

namespace ForRR.ViewModels
{
    public class UserMessageViewModel: ViewModelBase, INotifyPropertyChanged
    {
        private ObservableCollection<MessageModel> _messageList = new
ObservableCollection<MessageModel>();
        private string _inputVal;

        public ObservableCollection<MessageModel> MessageList
        {
            get => _messageList;
            set => this.RaiseAndSetIfChanged(ref _messageList, value);
        }

        public string InputVal
        {
            get => _inputVal;
            set => this.RaiseAndSetIfChanged(ref _inputVal, value);
        }

        public void Send()
        {
            var message = new MessageModel();
            message.MessageText = InputVal;
            MessageList.Add(message);
        }

        public UserMessageViewModel()
        {}
    }
}

```

MainWindowView.axaml

```

<views:UserMessageView DataContext="{Binding UserMessage}"/>

```

MainWindowViewModel.cs

```

public MainWindowViewModel()
{
    UserMessage = new UserMessageViewModel();
}

public UserMessageViewModel UserMessage { get; }

```

<https://github.com/dntbthrm/ForRR>