ПРАКТИЧЕСКАЯ РАБОТА № 3

Тема: Компоновка и элементы управления в WPF.

Цель: освоить основные контейнеры компоновки и элементы управления в технологии WPF, создать главный интерфейс приложения **«Музыка+»**, обеспечивающий визуальное отображение музыкальных данных из соответствующей базы данных.

Перечень оснащения и оборудования, источников: ПК, раздаточный материал, система управления базами данных, среда разработки Microsoft Visual Studio.

Задание: определить структуру проекта и разработать форму авторизации и главное окно приложения.

№ 1. Структура Вашего проекта может отличаться, в зависимости от того, какой подход к разработке Вы используете.

Если рассматривать фреймворк WPF, то есть 2 подхода к разработке программы:

1. Code-behind: подход к разработке приложения, который опирается на событийное программирование. Реакции всех элементов на экране можно представить в виде отдельных событий, которые могут быть вызваны пользователем напрямую.

2. MVVM: полноценная архитектура, согласно которой логика работы приложения делится на 3 основных компонента: модель (Model), представление (View) и модель представления (ViewModel). Данная архитектура позволяет разделить логику отображения данных от функциональной части приложения, что делает программный код чище и лёгким для чтения (за счёт отделения функций от самого окна).

Можно выделить следующие ключевые различия между этими подходами:

1. В Code-Behind вся логика работы окон хранится в скриптах «*.xaml.cs», в то время как в MVVM архитектуре вся рабочая логика находится в модели представления окна.

2. Также реакция всех элементов представляет собой событий в Code-Behind и напрямую связывается с программным кодом окна, а в MVVM все привязки задаются через механизм DataBinding и комманды.

3. MVVM легче тестировать из-за независимости программной логики от окна.

Определим базовые шаблоны структуры проекта в зависимости от подхода к разработке программы.

В Code-Behind основными элементами проекта будут являться сами окна и дополнительные программные блоки, выделенные в отдельные классы. Таким образом можно определить примерный шаблон структуры проекта для Code-Behind следующим образом:

```
MyWpfApp/
├── MainWindow.xaml
├── MainWindow.xaml.cs
├── Resources/
│   └── ...
├── Views/
│   ├── UserView.xaml
│   └── UserView.xaml.cs
├── Models/
│   └── User.cs
└── Services/
    └── DataService.cs
```

Данная структура включает в себя следующие составляющие:

1. MainWindow – основное окно приложения.
2. Resources – директория, в которой хранятся ресурсы приложения (изображения, файлы, документы и др.).
3. Views – окна приложения.
4. Models – модели, основанные на разработанной БД.
5. Services – дополнительные программные модули, которые могут быть переиспользованы в рамках проекта несколько раз.

В MVVM структура несколько сложнее, но при этом она предполагает разделение основных компонентов на соответствующие директории. Таким образом можно определить следующую директорию:

```
MyWpfApp/
├── Resources/
│   └── ...
├── Views/
│   ├── MainWindow.xaml
│   └── MainWindow.xaml.cs
├── ViewModels/
│   ├── MainViewModel.cs
│   ├── ViewModelBase.cs
│   └── RelayCommand.cs
├── Models/
│   └── User.cs
├── Services/
│   └── DataService.cs
└── Converters/
    └── BoolToVisibilityConverter.cs
```

Данная структура проекта включает в себя:

1. Resources – ресурсы приложения (изображения и др.).

2. Views – представления, то есть сами окна, UserControl-элементы и страницы.

3. ViewModels – модели представления, хранящие логику работы приложения.

4. Models – модели, основанные на таблицах БД.

5. Services – дополнительные программные модули.

6. Converters – конвертеры значений, нужны для привязки данных.

Стандартом разработки на WPF в настоящее время считается MVVM-архитектура, но Вы можете разрабатывать приложение и в Code-Behind, если считаете такой подход более рациональным.

На основе выбранного подхода сформируйте структуру Вашего проекта и создайте 2 новых окна: Auth (авторизация) и Menu (главное меню приложения).

№ 2. В WPF предусмотрено несколько элементов для формирования компоновки на экране: Grid, StackPanel, WrapPanel, DockPanel и Canvas.

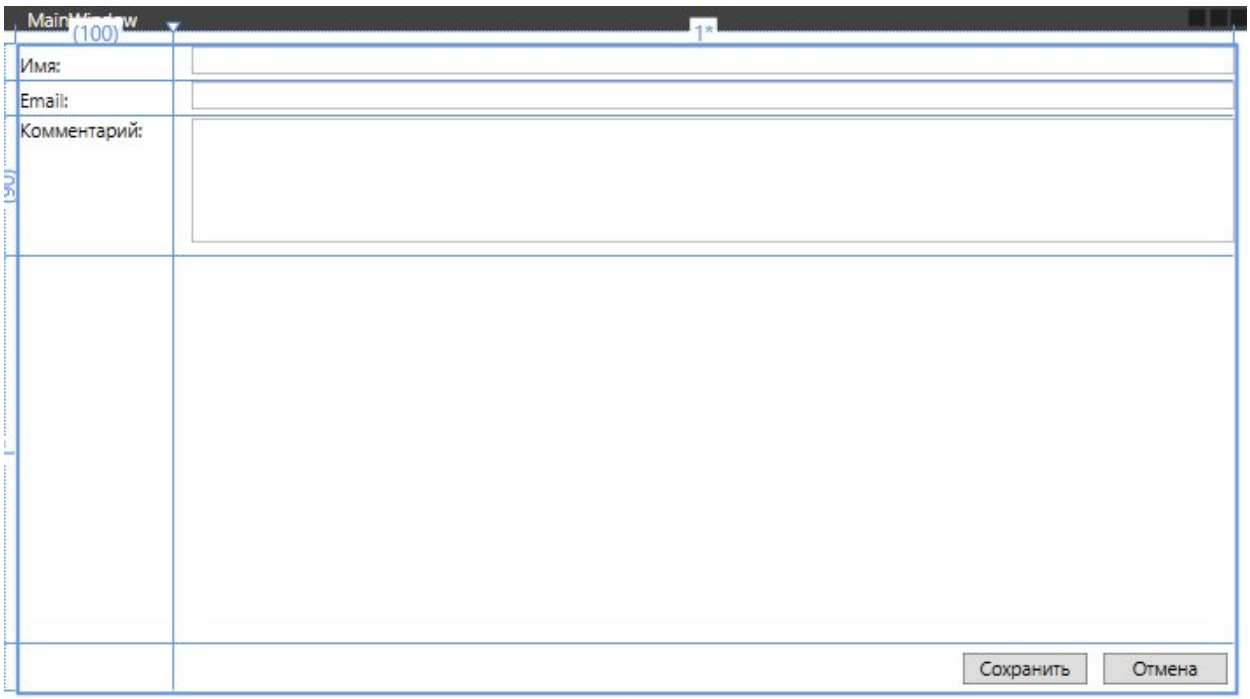
Основной принцип компоновки в WPF: UI строится как дерево элементов, где контейнеры (панели) содержат другие элементы (включая другие панели). С соответствующей структурой формы можно ознакомиться во вкладке «Структура документа».

В WPF при компоновке и расположении элементов внутри окна стоит придерживаться следующих принципов:

1. Нежелательно указывать явные размеры элементов. Размеры должны определяться контейнерами.

2. Нежелательно указывать явные позицию и координаты элементов внутри окна. Позиционирование элементов всецело должно быть прерогативой контейнеров.

Grid – это одна из самых используемых панелей компоновки в WPF. Она позволяет создавать сложные и адаптивные пользовательские интерфейсы, разбивая доступное пространство на строки (Rows) и столбцы (Columns), в которые можно размещать элементы.



Чтобы задать структуру сетки, используются коллекции **RowDefinitions** и **ColumnDefinitions**.

Каждая строка задается с помощью вложенного элемента **RowDefinition**, который имеет открывающий и закрывающий тег. При этом задавать дополнительную информацию необязательно.

Для определения нового столбца в Grid необходимо использовать элемент **ColumnDefinition**. Указанное количество объектов данного класса формирует соответствующее количество столбцов в таблице.

```
<Grid.RowDefinitions>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
</Grid.RowDefinitions>
```

```
<Grid.RowDefinitions>
    <RowDefinition/>
    <RowDefinition/>
    <RowDefinition/>
</Grid.RowDefinitions>
```

Для строки можно установить высоту, а для колонки ширину.

Для изменения высоты реализован атрибут **Height**.

Для изменения ширины, в свою очередь, необходимо задействовать атрибут **Width**.

Для определения размеров сетки (строк и столбцов) можно выделить 3 основные типа размеров: фиксированный, автоматический и относительный.

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="2*"/>
    <RowDefinition Height="30"/>
</Grid.RowDefinitions>
```

```
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="*"/>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="500"/>
</Grid.ColumnDefinitions>
```

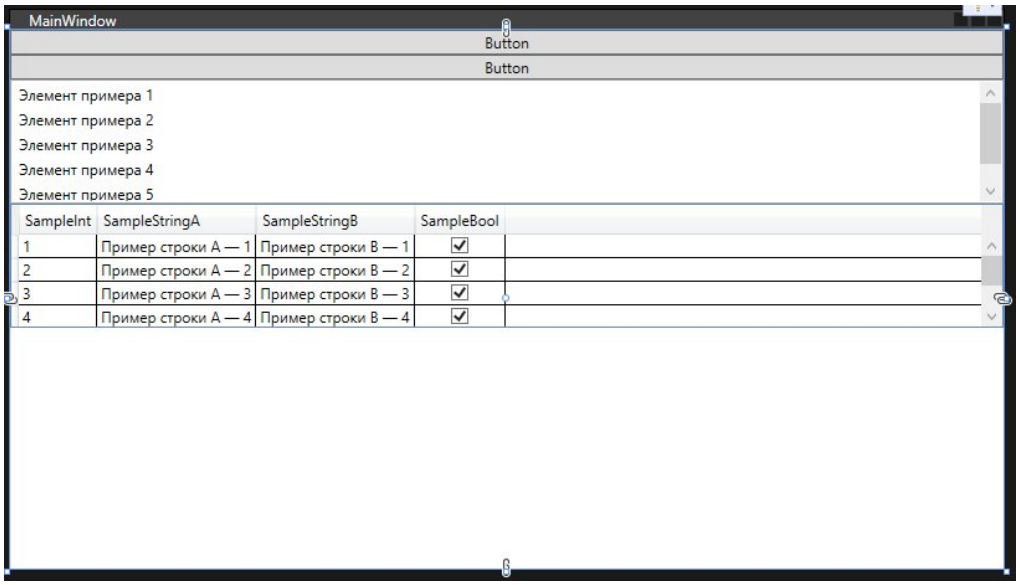
Не рекомендуется указывать явные размеры объектов внутри контейнеров, в том числе и внутри Grid. Но допускается определение минимальных значений с помощью следующих свойств:

1. **MinWidth / MaxWidth** – определяют минимальную и максимальную ширину объекта.
2. **MinHeight / MaxHeight** – определяют минимальную и максимальную высоту объекта.

Данный перечень свойств располагается в категории «Макет».

Помимо Grid в WPF есть и другие, более простые, элементы компоновки, например StackPanel.

StackPanel – одна из самых простых панелей в WPF. Предназначена для линейного размещения дочерних элементов – либо вертикального друг под другом, либо горизонтально в ряд. Работает по аналогии с такой структурой данных, как стек (Stack).

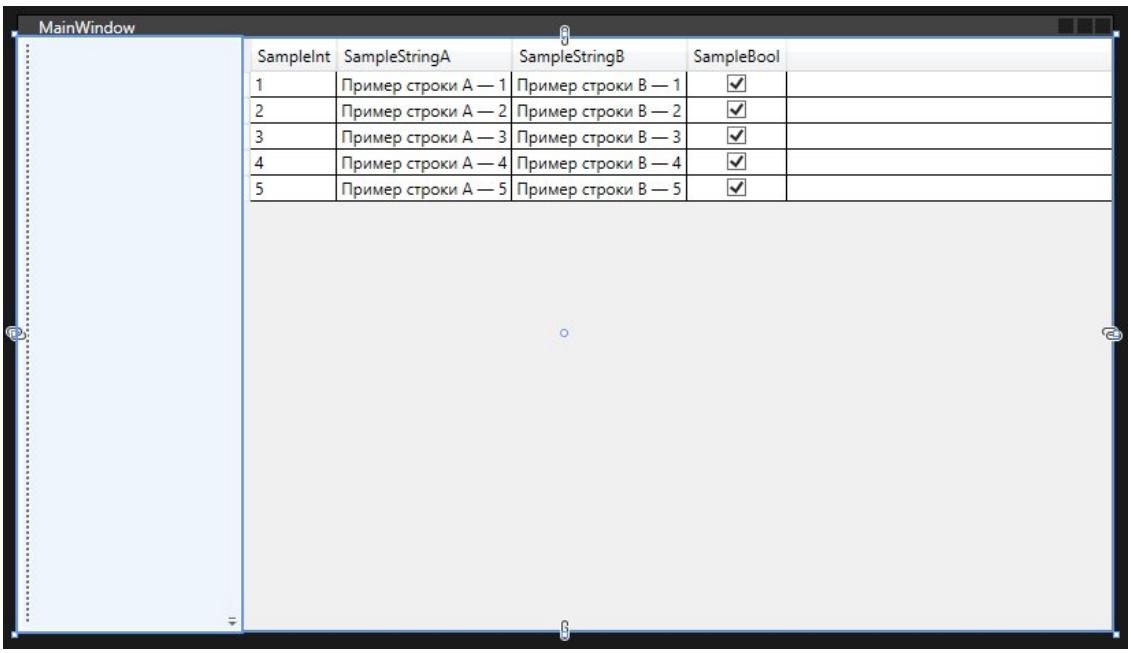


Основное свойство StackPanel – направление расположения элементов (Orientation). Находится во вкладке «Макет».

Допускаются только 2 значения: **Vertical** (сверху вниз) и **Horizontal** (слева направо).

У StackPanel есть разновидность, предлагающая больше возможностей, но работающая схожим образом – **WrapPanel**.

WrapPanel – это одна из панелей компоновки в WPF, которая располагает дочерние элементы в строку или столбец, автоматически перенося их на следующую строку или столбец, когда места больше нет.



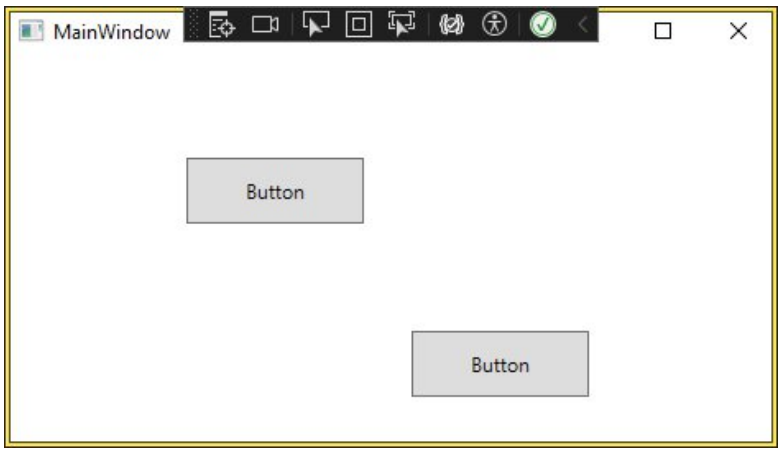
DockPanel работает по следующему принципу:

1. Элементы поочерёдно прикрепляются к указанной стороне.
2. Каждый прикреплённый элемент занимает всё доступное пространство по своей оси.
3. Последний элемент заполняет остаток – это обычно основная рабочая область.

Контейнер DockPanel особенно удобно использовать для создания стандартных интерфейсов, где верхнюю и левую часть могут занимать какие-либо меню, нижнюю - строка состояния, правую - какая-то дополнительная информация, а в центре будет находиться основное содержание.

Canvas – это одна из панелей компоновки в WPF, **предназначенная для абсолютного позиционирования** дочерних элементов с помощью точных координат X и Y.

В отличие от других панелей, Canvas не управляет размерами и расположением элементов автоматически – разработчик сам задаёт, где и как они должны находиться.



Каждый дочерний элемент позиционируется с помощью следующих свойств:

1. **Canvas.Top** – расстояние от верхнего края.
2. **Canvas.Bottom** – расстояние от нижнего края.
3. **Canvas.Right** – расстояние от правого края.
4. **Canvas.Left** – расстояние от левого края.

```
<Canvas>
    <Button Content="Button" Canvas.Left="110" Canvas.Top="61"
    <Button Content="Button" Canvas.Left="247" Canvas.Top="166"
</Canvas>
```

Определите структуру окна авторизации, выберите наиболее подходящие элементы компоновки и создайте окно авторизации. Также реализуйте главное меню приложения.

Пример окна авторизации: