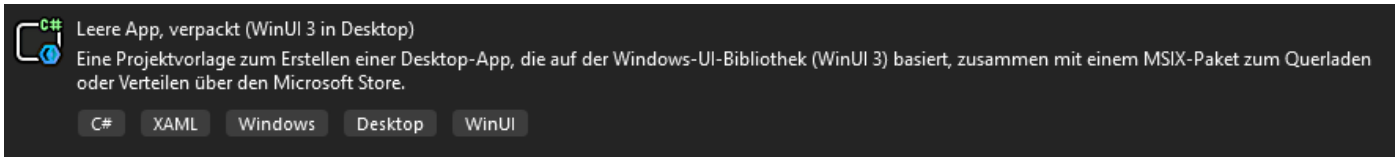
- TicTacToe in C#
  - 1. Projekt erstellen
  - 2. Schaltflächen erstellen
  - 3. Methode Button_Click implementieren
  - 4. Designanpassungen
  - 5. Gedanken für die Implementierung
  - 6. Tipps für die Implementierung
  - Und jetzt du!
  - Zusatzaufgaben wenn das noch zu langweilig war

TicTacToe in C#

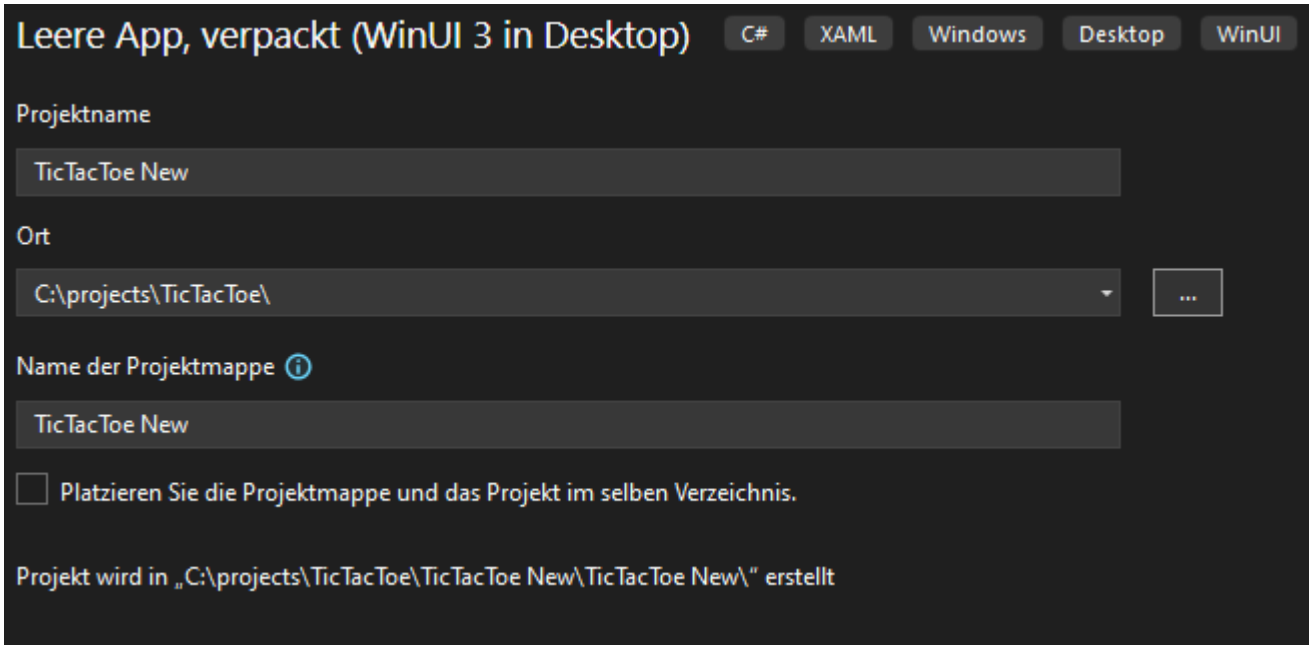
In diesem Tutorial werden wir ein einfaches TicTacToe-Spiel in C# erstellen. Hierfür benötigen wir Visual Studio und grundlegende Kenntnisse in C#.

1. Projekt erstellen

In Visual Studio ein neues Projekt mit der Vorlage `Leere App, verpackt (WinUI 3 in Desktop)` erstellen.



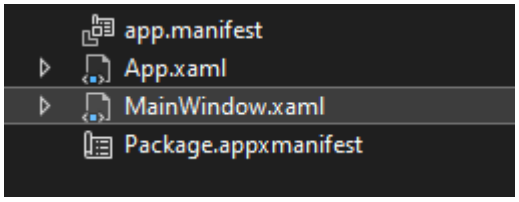
Im nächsten Schritt einen Projektnamen und Speicherort definieren.



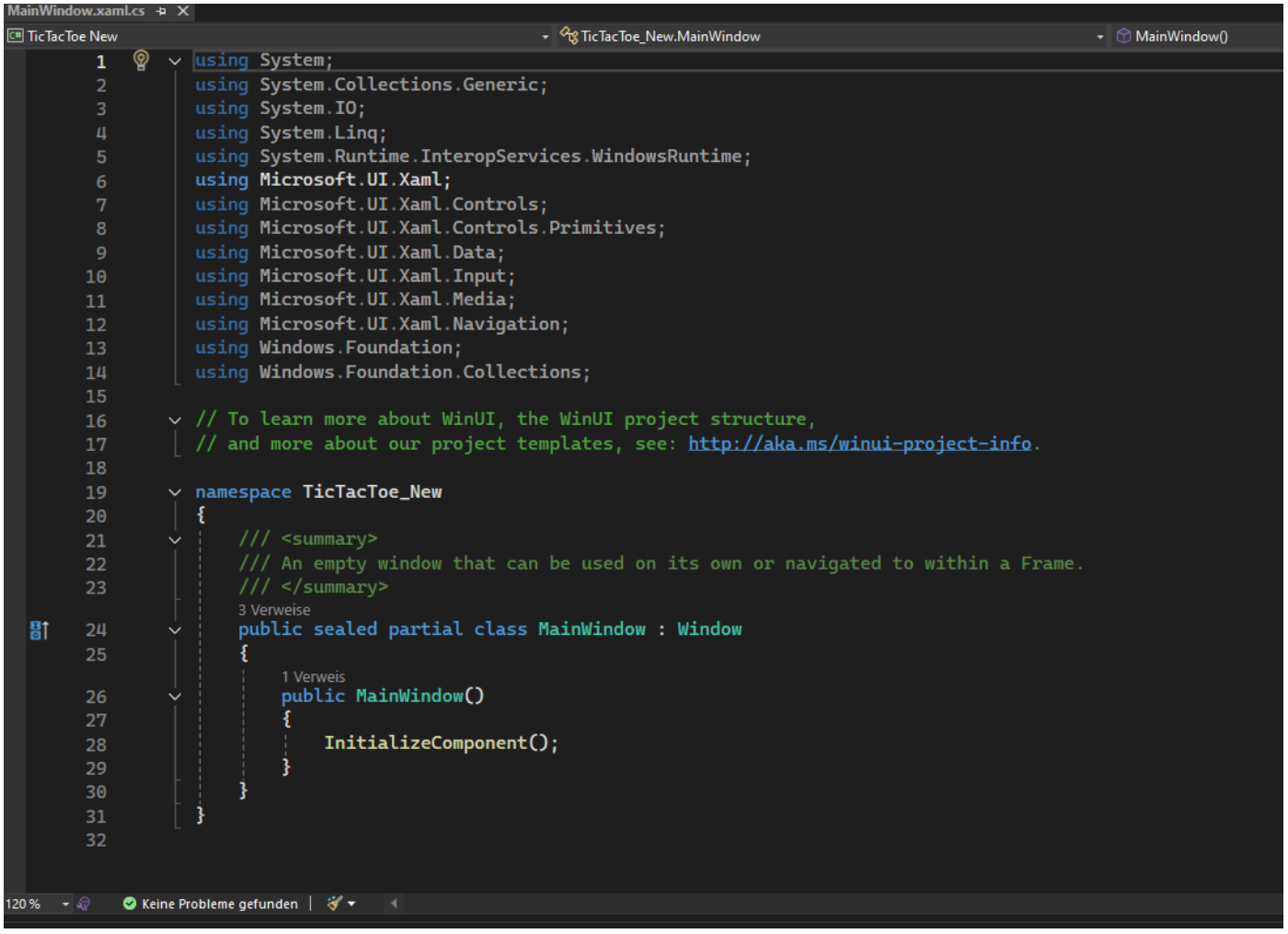
Jetzt auf den `Erstellen`-Button klicken.

2. Schaltflächen erstellen

Im `Projektmappen-Explorer` auf `MainWindow.xaml` doppelklicken.



Der View sollte jetzt ungefähr so aussehen:



Über einen erneuten Doppelklick auf `MainWindow.xaml` gelangt man in den XAML-Code

XAML ist eine Markup-Sprache, die verwendet wird, um Benutzeroberflächen zu erstellen. Hierdurch wird sozuagen die Struktur der Benutzeroberfläche definiert.

Der XAML-Code sollte wie folgt aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<Window
    x:Class="TicTacToe_New.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:TicTacToe_New"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Title="TicTacToe New">

    <Window.SystemBackdrop>
        <MicaBackdrop />
    </Window.SystemBackdrop>

    <Grid>
```

```
    </Grid>
</Window>
```

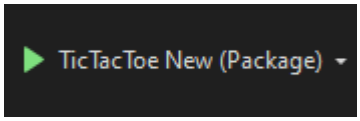
Im `Grid`-Element werden wir die Schaltflächen für das TicTacToe-Spiel erstellen.

```
<!-- Tictactoe -->
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="Auto"/>
</Grid.ColumnDefinitions>
<Button x:Name="Button1" Grid.Row="0" Grid.Column="0" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button2" Grid.Row="0" Grid.Column="1" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button3" Grid.Row="0" Grid.Column="2" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button4" Grid.Row="1" Grid.Column="0" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button5" Grid.Row="1" Grid.Column="1" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button6" Grid.Row="1" Grid.Column="2" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button7" Grid.Row="2" Grid.Column="0" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button8" Grid.Row="2" Grid.Column="1" Click="Button_Click"
FontSize="36"/>
<Button x:Name="Button9" Grid.Row="2" Grid.Column="2" Click="Button_Click"
FontSize="36"/>
```

Was bedeutet der Code?

- `Grid.RowDefinitions` und `Grid.ColumnDefinitions` definieren die Anzahl der Zeilen und Spalten im Grid.
- `Button`-Elemente definieren die Schaltflächen für das TicTacToe-Spiel.
- `Grid.Row` und `Grid.Column` definieren die Position der Schaltflächen im Grid.
- `Click="Button_Click"` definiert die Methode, die aufgerufen wird, wenn die Schaltfläche geklickt wird.
- `FontSize="36"` definiert die Schriftgröße der Schaltflächen.
- `x:Name` definiert den Namen der Schaltfläche.

Über den Play-Button kann das Programm gestartet werden:



Diese Schaltflächen sind jetzt im Moment aber noch sehr klein. Das müssen wir ändern.

Im ersten Schritt müssen wir die Breite und Höhe der Spalten ändern. Statt `Auto` setzen wir `*`:

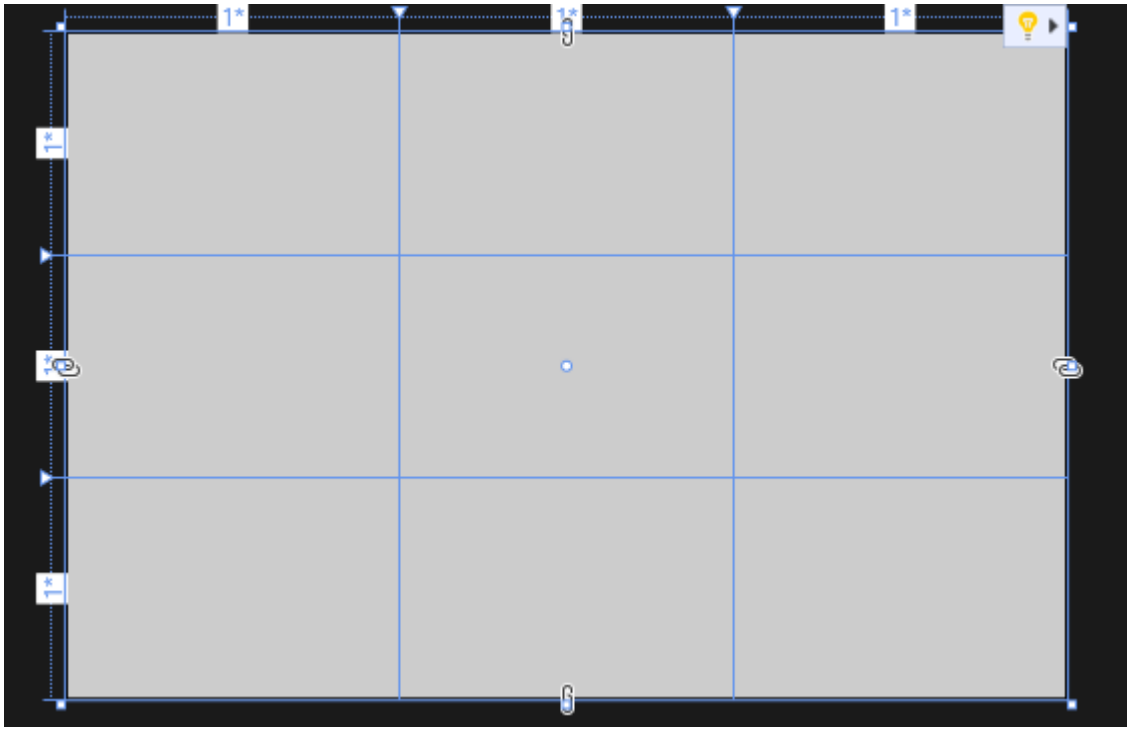
```
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
```

Dies ändert unseren View so dass die Spalten und Zeilen gleichmäßig verteilt sind.

Jetzt müssen sich nur noch die Buttons anpassen hierfür verwenden wir `HorizontalAlignment` und `VerticalAlignment`:

```
<Button x:Name="Button1" Grid.Row="0" Grid.Column="0" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button2" Grid.Row="0" Grid.Column="1" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button3" Grid.Row="0" Grid.Column="2" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button4" Grid.Row="1" Grid.Column="0" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button5" Grid.Row="1" Grid.Column="1" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button6" Grid.Row="1" Grid.Column="2" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button7" Grid.Row="2" Grid.Column="0" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button8" Grid.Row="2" Grid.Column="1" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
<Button x:Name="Button9" Grid.Row="2" Grid.Column="2" Click="Button_Click"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch" FontSize="36"/>
```

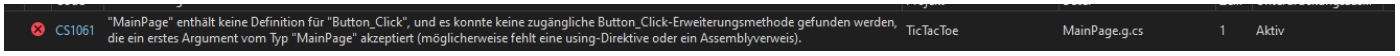
Jetzt sehen die Buttons so aus:



Man sieht, dass die Buttons jetzt die gesamte Breite und Höhe der Zellen einnehmen und somit ein Spielfeld über das gesamte Fenster erstellt wird.

Wir können jetzt mal die App starten und sehen, dass die Buttons auch funktionieren.

Nun sollten wir einen Fehler sehen, da die Methode `Button_Click` noch nicht implementiert ist.



Diesen Fehler gilt es jetzt zu beheben.

3. Methode `Button_Click` implementieren

Über rechtsklick -> `Code anzeigen` gelangen wir in den Code-Behind der `MainPage.xaml`: **`MainPage.xaml.cs`**.

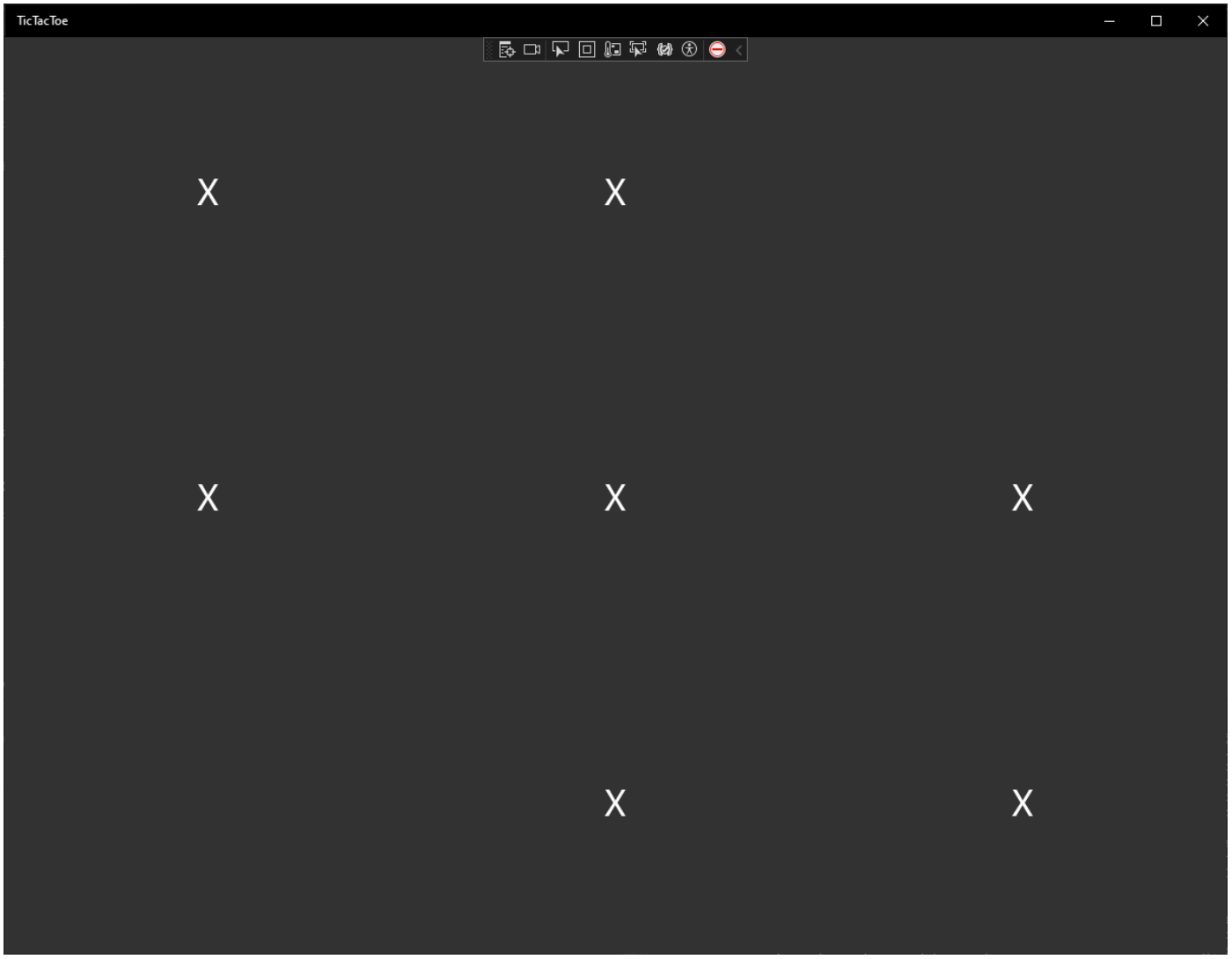
Hier implementieren wir die Methode `Button_Click`:

```
namespace TicTacToe
{
    /// <summary>
    /// Eine leere Seite, die eigenständig verwendet oder zu der innerhalb eines
Rahmens navigiert werden kann.
    /// </summary>
```

```
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            Button button = (Button)sender;
            button.Content = "X";
        }
    }
}
```

Starten wir jetzt das Programm und klicken auf einen Button, sollte sich der Inhalt des Buttons auf `X` ändern.



Wir merken jetzt aber, dass man die einzelnen Buttons nicht von einander unterscheiden kann. Das müssen wir ändern.

4. Designanpassungen

Wir wechseln wieder auf die `MainPage.xaml` und fügen den Buttons ein Style hinzu:

```
BorderThickness="10,10,10,10" BorderBrush="White"
```



5. Gedanken für die Implementierung

- Wir müssen die Methode `Button_Click` so anpassen, dass abwechselnd `X` und `O` gesetzt wird.
- Wir müssen eine Methode implementieren, die prüft, ob ein Spieler gewonnen hat.
- Wir müssen eine Methode implementieren, die prüft, ob das Spiel unentschieden ist.
- Wir müssen eine Methode implementieren, die das Spiel zurücksetzt.

6. Tipps für die Implementierung

Und jetzt du!

Jetzt bist du an der Reihe! Implementiere die oben genannten Punkte und erstelle ein voll funktionsfähiges TicTacToe-Spiel in C#.

Zusatzaufgaben wenn das noch zu langweilig war

- Implementiere eine Methode, die den Gewinner anzeigt.
- Ändere die Farbe der Schaltflächen, wenn jemand ein Zeichen setzt.
- Programmiere eine "KI"
- Implementiere einen "Neues Spiel"-Button
- Vergrößere das Spielfeld

