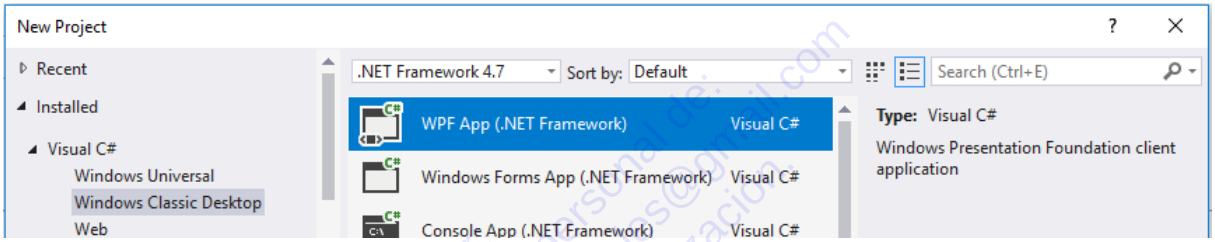
Creando métodos Esperables (Awaitable Methods)

Un ***Método Esperable (Awaitable Method)*** es aquel que permite ser ejecutado de forma asíncrona y esperar a que finalice su ejecución sin bloquear el hilo que lo invoca.

Ejercicio
Creando métodos Esperables (Awaitable Methods)

Realiza los siguientes pasos para conocer la forma de crear ***Métodos Esperables***.

1. Crea una aplicación WPF utilizando la plantilla ***WPF App (.NET Framework)***.



2. Dentro del archivo ***MainWindow.xaml***, remplaza el elemento ***<Grid>*** por el siguiente código.

```
<StackPanel>
    <Label x:Name="lblResult"/>
    <Button x:Name="btnGetResult"
            Content="Obtener resultado" Click="btnGetResult_Click"/>
</StackPanel>
```

   El código anterior simula una aplicación sencilla que consiste en un botón y una etiqueta.

   Si un usuario hace clic en el botón, el método ***btnGetResult_Click*** será ejecutado en el hilo de la interfaz de usuario.

3. Agrega la siguiente instrucción al inicio del archivo ***MainWindow.xaml.cs*** para importar el espacio de nombre de la clase ***Thread***.

```
using System.Threading;
```

4. Agrega el siguiente código en la clase ***MainWindow*** para simular un método que obtiene de una base de datos el nombre de un producto a partir de su ID.

```
string GetProductName(int ID)
{
    // Simulamos un proceso de larga duración
    Thread.Sleep(10000);
    return "Chai";
}
```

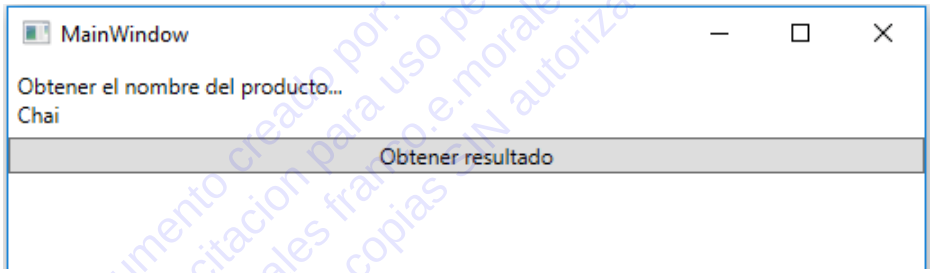
5. Agrega el siguiente código dentro de la clase ***MainWindow*** para definir el manejador del evento ***Click*** del botón ***btnGetResult***.

```
private void btnGetResult_Click(object sender, RoutedEventArgs e)
{
    lblResult.Content = "Obtener el nombre del producto...";
    var ProductName = GetProductName(1);
    lblResult.Content += Environment.NewLine + ProductName;
}
```

El código anterior hace un llamado al método GetProductname para obtener el nombre del producto con ID = 1.

6. Ejecuta la aplicación y haz clic en el botón ***Obtener resultado***. Puedes notar que la aplicación se ejecuta de manera síncrona. El hilo principal se queda congelado hasta que la ejecución del método finalice para poder continuar, esto lo puedes notar si durante ese tiempo intentas mover o cambiar el tamaño de la ventana.

Al finalizar la ejecución del método, el resultado es mostrado.



7. Agrega el modificador ***async*** al método ***btnGetResult_Click*** para permitir que pueda ser ejecutado de forma asíncrona.

```
private async void btnGetResult_Click(object sender, RoutedEventArgs e)
```

8. Agrega el operador ***await*** en la línea de código que invoca al método que obtiene el nombre del producto. El operador ***await*** es utilizado para esperar a que una instancia de la clase *Task* haya terminado su ejecución sin bloquear el hilo que la inicia.

```
private async void btnGetResult_Click(object sender, RoutedEventArgs e)
{
    lblResult.Content = "Obtener el nombre del producto...";
    var ProductName = await GetProductName(1);
    lblResult.Content += Environment.NewLine + ProductName;
}
```

Puedes notar ahora el mensaje de error que indica que el tipo ***string*** devuelto por el método ***GetProductName*** no contiene una definición de ***GetAwaiter***.

```
var ProductName = await GetProductName(1);
'string' does not contain a definition for 'GetAwaiter' and
```

El operador *await* espera un tipo de dato que pueda ser "*Esperable*" *("Awaitable")*. Un tipo de dato "*Awaitable*" expone un método ***GetAwaiter*** que devuelve un "*awaiter*" válido que es utilizado para esperar la ejecución de una tarea. Los tipos de datos *awaitables* comúnmente utilizados son ***Task*** y ***Task<Tresult>***.

Normalmente, si deseamos crear un método asíncrono que podamos esperar a que termine su ejecución utilizando el operador ***await***, el método debe devolver un objeto ***Task* o *Task<TResult>***.

Cuando convertimos un método síncrono a un método asíncrono, debemos seguir la siguiente guía:

- Agregar el modificador ***async*** a la declaración del método.
- Cambiar el tipo de resultado devuelto:
  - Si el método síncrono no devuelve un valor, al convertirlo en asíncrono debe devolver un objeto ***Task***.
  - Si el método síncrono devuelve un valor ***T***, al convertirlo en asíncrono debe devolver un objeto ***Task<T>***.
- Modificar la lógica del método para utilizar el operador ***await*** con cualquier operación de larga duración.

En nuestro caso el método síncrono ***GetProductName***, devuelve un tipo ***string***, por lo tanto, siguiendo la guía, para convertirlo en un método asíncrono "*esperable*", debemos cambiar el tipo de retorno ***string*** por ***Task<string>***.

9. Agrega el modificador ***async*** a la declaración del método ***GetProductName*** y cambia el valor devuelto ***string*** por ***Task<string>***.

```
async Task<string> GetProductName(int ID)
```

10. Modifica la lógica del método para utilizar el operador ***await*** con una tarea que ejecute la operación de larga duración.

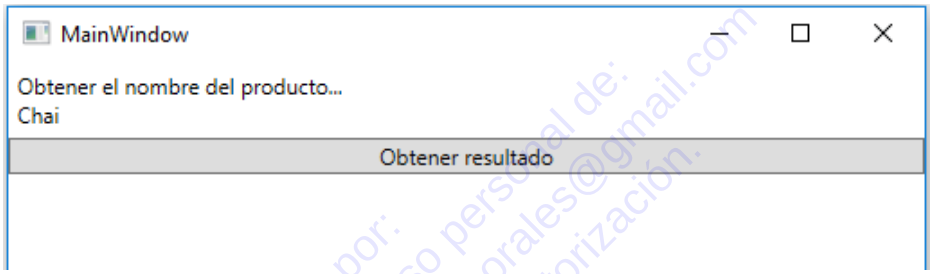
```
async Task<string> GetProductName(int ID)
{
    string Result = await Task.Run<string>(() =>
    {
        // Simulamos un proceso de larga duración
        Thread.Sleep(10000);
        return "Chai";
    });
```

```
    return Result;
}
```

Puedes notar que la instrucción ***await GetProductName(1)*** ahora ya no muestra el error. Tenemos ahora un método asíncrono "esperable".

11. Ejecuta la aplicación y haz clic en el botón ***Obtener resultado***. Puedes notar que la aplicación se ejecuta de manera asíncrona. El hilo principal ya no se queda congelado debido a que ya no espera que la ejecución del método finalice para poder continuar, esto lo puedes notar si durante ese tiempo intentas mover o cambiar el tamaño de la ventana, ahora ya es posible.

    Al finalizar la ejecución del método, el resultado es mostrado.



De la misma forma en que podemos tener métodos síncronos que devuelven un tipo de dato, también es posible tener métodos síncronos que no devuelvan algún valor y que podríamos querer convertirlos en métodos asíncronos *esperables*. Veamos un ejemplo de esto.

12. Agrega el siguiente código para definir un método que busca un nombre de producto y lo muestra en la etiqueta ***lblResult***.

```
void ShowName(int ID)
{
    // Simulamos un proceso de larga duración
    Thread.Sleep(10000);
    string ProductName = "Chang";

    lblResult.Content += ProductName;
}
```

Un método asíncrono puede devolver ***void***, sin embargo, normalmente esto sólo es usado por los manejadores de eventos. Siempre que sea posible, debemos devolver un objeto ***Task*** para permitir a los invocadores utilizar el operador ***await*** con el método.

13. Agrega el siguiente código al final del método ***btnGetResult_Click***. En nuestro caso, esa sería la forma para invocar al método ***ShowName*** una vez que lo hayamos convertido en un método asíncrono *esperable*.

```
private async void btnGetResult_Click(object sender, RoutedEventArgs e)
{
    lblResult.Content = "Obtener el nombre del producto...";
    var ProductName = await GetProductName(1);
    lblResult.Content += Environment.NewLine + ProductName;
    await ShowName(2);
}
```

Si un método síncrono devuelve ***void*** (en otras palabras, no devuelve un valor) como en nuestro ejemplo, para convertirlo en un método asíncrono *esperable*, debemos agregar el modificador ***async*** a la declaración del método, cambiar el tipo de retorno ***void*** por ***Task*** y modificar la lógica del método para utilizar el operador ***await*** con cualquiera de las operaciones de larga duración.

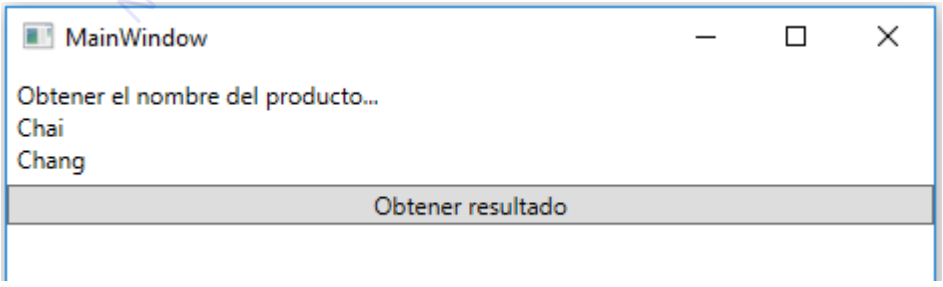
14. Modifica el método ***ShowName*** para convertirlo en un método asíncrono *esperable*.

```
async Task ShowName(int ID)
{
    string ProductName = await Task.Run<string>(() =>
    {
        // Simulamos un proceso de larga duración
        Thread.Sleep(10000);
        return "Chang";
    });

    lblResult.Content += Environment.NewLine + ProductName;
}
```

15. Ejecuta la aplicación y haz clic en el botón ***Obtener resultado***. Puedes notar que la aplicación se ejecuta de manera asíncrona. El hilo principal no se queda congelado mientras se obtienen los resultados, esto lo puedes probar si durante ese tiempo intentas mover o cambiar el tamaño de la ventana y notas que es posible.

    Al finalizar la ejecución del método, el resultado es mostrado.



En este ejercicio, mostramos la forma de crear métodos ***esperables (awaitables***).

Para obtener más información sobre cómo devolver valores en métodos asíncronos, se recomienda consultar el siguiente enlace:

Async Return Types (C#)
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/async-return-types

Para obtener más información sobre cómo implementar tipos *Awaitables*, se recomienda consultar el siguiente enlace:

await anything;
https://blogs.msdn.microsoft.com/pfxteam/2011/01/13/await-anything/