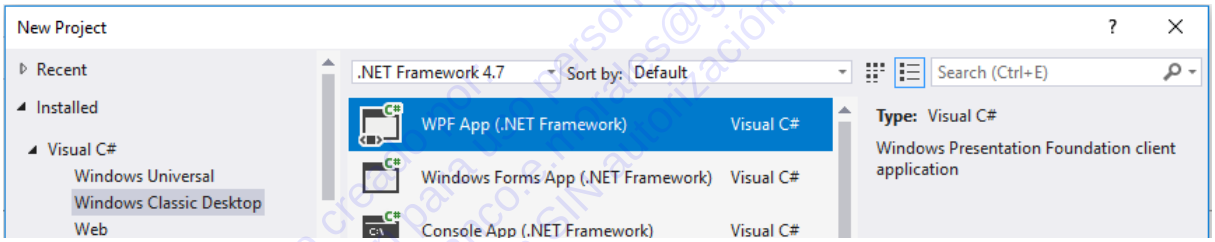
Creando e Invocando Métodos Callback

Si deseamos ejecutar algún tipo de lógica cuando un método asíncrono finalice su ejecución, podemos configurar al método asíncrono para que invoque a un método que contenga la lógica que deseamos ejecutar. El método invocado es conocido como *Método Callback*. Tradicionalmente el método asíncrono proporciona datos al método *Callback* para que este los procese. En una aplicación gráfica, el método *Callback* podría actualizar la interfaz de usuario con los datos procesados.

Ejercicio
Creando e Invocando Métodos Callback

Realiza los siguientes pasos para conocer la forma de crear e invocar ***Métodos Callback***.

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>
    <Button x:Name="GetProducts" Content="Buscar productos" Width="100"
            Click="GetProducts_Click"/>
    <ListBox x:Name="Products" Width="100" />
</StackPanel>
```

Deseamos que cuando el usuario haga clic en el botón, el manejador de evento invoque un método asíncrono que recupere una lista de productos y los muestre en el control ***ListBox***.

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 definir un método asíncrono que simule un proceso de larga duración que obtiene una lista con nombres de productos.

```
async Task GetAllProducts()
{
    var Products = await Task.Run(() =>
    {
        Thread.Sleep(5000);
```

```
            return new List<string>
            {
                "Azucar", "Café", "Leche", "Frijol", "Queso", "Azucar", "Frijol"
            };
        });
    }
```

Como puedes notar, la lista de productos puede contener elementos repetidos así que cuando la recuperación asíncrona de los datos haya finalizado, queremos que sea invocado un segundo método *(método Callback)* que elimine los elementos duplicados del resultado y que muestre el resultado actualizado de la lista de productos.

5. Agrega el siguiente código a la clase ***MainWindow*** para definir un método *Callback* que elimine duplicados de una lista con nombres de productos y los asigne al control *ListBox*.

```
void RemoveDuplicates(List<string> products)
{
    var UniqueProducts = products.Distinct();
    Products.Dispatcher.Invoke(() =>
    {
        Products.ItemsSource = UniqueProducts;
    });
}
```

Para que un método asíncrono (por ejemplo, *GetAllProducts*) pueda invocar a un método *Callback*, podemos incluirle como parámetro un tipo *delegate* donde podamos proporcionar el método *Callback*.

Un método *Callback* típicamente acepta uno o más argumentos y no devuelve un valor, como el caso del método *Callback RemoveDuplicates*.

La firma de los métodos *Callback*, hace que el delegado genérico ***Action<T>*** sea una buena opción para representar un método *Callback* donde ***T*** es el tipo del parámetro del método *Callback*.

6. Modifica la definición del método asíncrono *GetAllproducts* para que acepte un tipo *delegate* como parámetro donde podamos proporcionarle el método *Callback*.

```
async Task GetAllProducts(Action<List<string>> callBack)
```

En este ejemplo, el método *Callback* únicamente recibe un parámetro que es una lista de ***string*** y no devuelve valor. Si el método *Callback* requiriera múltiples argumentos, existen versiones del delegado ***Action*** que aceptan hasta 16 parámetros de tipos.

El método asíncrono puede pasar los datos al método *Callback* para que este los procese.

7. Agrega el siguiente código al final del método ***GetAllProducts*** para invocar al método *Callback* de forma asíncrona.

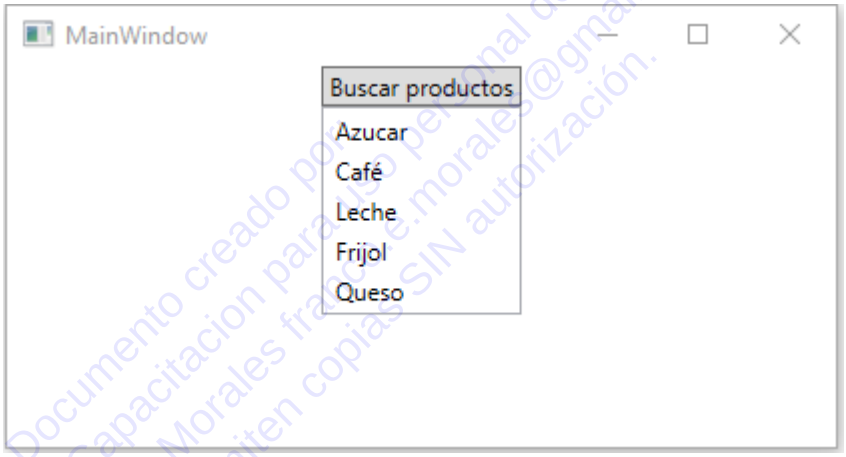
```
await Task.Run(() => callBack(Products));
```

Cuando invoquemos al método asíncrono, podremos proporcionarle el nombre del método *Callback* como un argumento.

8. Modifica el código del método ***GetProducts_Click*** para que pueda ser ejecutado de forma asíncrona e invoque al método ***GetAllProducts*** pasándole como argumento el nombre del método *Callback*.

```
private async void GetProducts_Click(object sender, RoutedEventArgs e)
{
    await GetAllProducts(RemoveDuplicates);
}
```

9. Ejecuta la aplicación y haz clic en el botón ***Buscar productos***. Puedes notar que el resultado es mostrado y no contiene nombres de productos duplicados.



En este ejercicio, mostramos la forma de crear e invocar métodos ***Callback***.