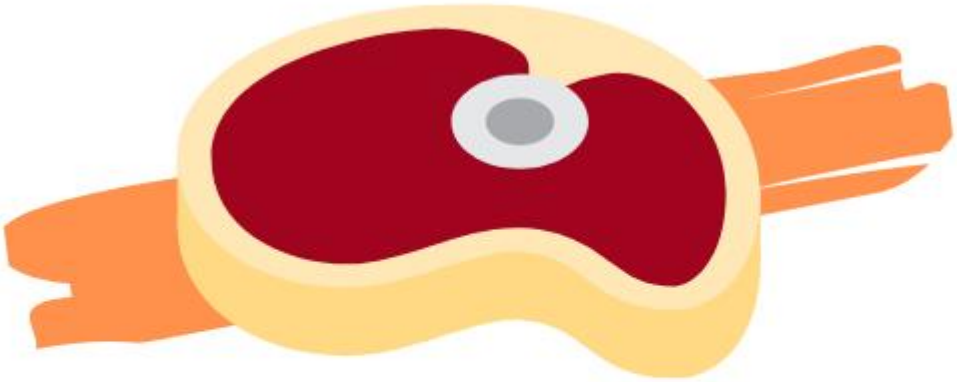

Microsoft®
.NET

ESQUEMA CREACIÓN APP CON .NET

DTO

- Añadir el interfaz **INotifyPropertyChanged** para que se notifiquen los cambios.
- Clase pública.
- Añadir este tipo de getters/setters

```
public string Apellidos
{
    get { return _apellidos; }
    set
    {
        this._apellidos = value;
        this.PropertyChanged(this, new PropertyChangedEventArgs("Apellidos"));

    }
}
```

DISEÑO PANTALLAS

- Anclar con shift.
- Nombre elementos.

ENLAZAR PANTALLAS

- Abrir desde main: instanciar diálogo en el botón y ejecutar .show().
- Volver a main: this.close()

AÑADIR LÓGICA

- Clase pública.
- Tendrá la lista pública -> ObservableCollections<Patatas> con { get; set } OJO A LOS GET/SET
- Constructor que inicie la lista.
- Hay que pasar la lógica a los diálogos:
    - El constructor tiene que recibir un objeto Lógica.
- Ventana principal:
    - Instancia de lógica, inicial (la que se pasa a los diálogos).

MOSTRAR LISTA EN DATAGRID

- Binding propio para el componente, en el constructor: DataGridLibros.DataContext = lógica;
- En el XAML del componente: ItemsSource=”{Binding Path=listaLibros}”
- Evitar placeholder combobox = CanUserAddRows = False

ALTA DE OBJETO EN DIÁLOGO

- Se instancia un objeto en el diálogo. Se inicializa en el constructor, sin parámetros.
- Establecer como contexto del dialogo, el objeto (libro, patatas…) -> this.DataContext = libro
- En el XAML, hacer bindings en los input del diálogo -> Text="{Binding Path=Titulo}" // SelectedDate="{Binding Path=FechaEntrada}" [así, se relacionan los componentes con la instancia del objeto que nos interesa]
- Implementar botón aceptar/insertar:

- onClick, llamar al método que nos interese (lógica.aniadirLibro())
- Para vaciar campos:
  - Iniciar un nuevo libro vacío.
  - Añadirlo al contexto: this.DataContext = libroVacio;
  - O cerrar ventana.)

MODIFICAR

- Comprobar que hay algo seleccionado en el componente del que cogemos los datos (DataGridLibros.SelectedIndex ¡=-1…) y recogemos el item seleccionado en un nuevo objeto (DataGridLibros.SelectedItem)
- Implementar ICloneable en el objeto -> en Clone() -> return this.MemberwiseClone();
- Se pasa al constructor la posición y el item clonado, y se muestra.
  [en vez de trabajar sobre el objeto se leccionado, hace una copia y, al aceptar, guarda el nuevo objeto en la posición inicial. Si no, al hacer cambios, se modificaría en el datagrid y se almacenarían dichos cambios al pulsar en cancelar]

VALIDAR FORMULARIO

- Va en el objeto (libro, patata, cabeza…).
- Implementar IDataErrorInfo:
  - public string Error => " ";
  - public string this[string columnName] :

```
public string this[string columnName]
{
    get
    {
        String result = "";
        if (columnName == "Título")
            if (string.IsNullOrEmpty(_titulo))
                result = "Debe introducir un título";
        if (columnName == "Autor")
            if (string.IsNullOrEmpty(_autor))
                result = "Debe introducir un autor";
        return result;
    }
}
```

- En el XAML del diálogo de alta:
  - Añadir al binding de los campos:
    - Text="{Binding Path=Titulo, NotifyOnValidationError=True, ValidatesOnDataErrors=True}"
- Para que el botón aceptar esté desactivado hasta que se valide el formulario:
  - En el XAML del diálogo, en cada campo, añadir esta función:
    - Validation.Error="validation_Error"
  - Implementarla en el cs del diálogo:
    - Instanciar contador de errores.
    - Escribir función:

```
private void validation_Error(object sender, ValidationErrorEventArgs e)
{
    if (e.Action == ValidationErrorEventAction.Added)
    {
        contadorErrores++;
    }
    else
    {
        contadorErrores--;
    }

    if (contadorErrores == 0)
    {
        btnInsertar.IsEnabled = true;
    }
    else
    {
        btnInsertar.IsEnabled = false;
    }
}
```

```
private void cbDatos_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    ProductoDTO productoSeleccionado = (ProductoDTO)cbDatos.SelectedItem;
    this.refresh();
}
```

```
private void refresh(ProductoDTO producto)
{
    if (producto.Stock > 0)
    {
        btnSale.Visibility = Visibility.Visible;
        btnEntra.Visibility = Visibility.Visible;
    }
    else
    {
        btnSale.Visibility = Visibility.Hidden;
        btnEntra.Visibility = Visibility.Visible;
    }
}
```