EXAMEN INTERFACES

ESTILOS:

Antes del Grid inicial, ponemos los recursos del siguiente modo:

-Window.Resources -> En esta etiqueta metemos todos los estilos que vamos a utilizar.

TIPO 1 (Específico).

- Style x:Key=”estilo” TargetType=”Menu” -> En esta etiqueta vamos a meter las instrucciones de un determinado estilo. X:key es el nombre del estilo y TargetType, el tipo de componente al que se le va a aplicar.

-Setter property=”background” value=”orange”; Aquí añadimos una property a la que se le asigna el nombre y en value lo que coge (Fondo naranja en este caso). Puedes tener muchos setter.

Style="{StaticResource estilo}" -> Para asignarlo a un menú será de este modo.

TIPO 2 (Varios tipos de target).

LinearGradientBrush x:key=”gradiante” -> Permite crear un degradado con este nombre

GradientStop Offset=”0” Color “LightGreen”> Nos indica la posición del gradiante y el color de dicha posición, (puedes hacer un degradado de muchos colores diferentes.

GradientStop Offset=”1” Color “Green”>

LLAMARLO EN EL COMPONENTE (En este caso un bloque de texto)

<TextBlock.Background>

<StaticResource ResourceKey="gradiante"></StaticResource>

</TextBlock.Background>

…….

Menú

Para el menú creamos un DockPanel al que le agregamos a la posición superior del mismo el menú. Dentro del menú metemos los MenuItem necesario con su Header correspondente. Si es desplegable dentro del MenuItem metemos otro menúItem, y si va a realizar alguna acción le tenemos que dar un xname y un click (Diferentes, mucho ojo). En caso de que queramos un icono en el menuitem, Añadimos un MenuItem.Icon dentro, y dentro de este, Image Source=”/abrir.png”> Debemos recordar meter las imágenes en la misma carpeta de todos los archivos del proyecto y en EXPLORADOR DE SOLUCIONES/COMUN/SOURCE, elegimos la imagen. No olvidar LastChildFill=”false” en el DockPanel

<Grid>

```
<DockPanel LastChildFill="False">
  <Menu DockPanel.Dock="Top" Width="auto" Height="20" Style="{StaticResource estilo}">
    <MenuItem Header="Acciones">
      <MenuItem x:Name="abre" Header="Abrir.txt" Click="abrir" >
        <MenuItem.Icon>
          <Image Source="/abrir.png"></Image>
        </MenuItem.Icon>
      </MenuItem>
    </MenuItem>
  </Menu>
```

RESTO XML

Podemos incluir otros componentes en el resto de las posiciones del **dockpanel.** En este caso en la parte de abajo metemos un **StackPanel** que incluirá un botón. El stackpanel provoca que los objetos ocupen todo su espacio si no se indica lo contrario.

Grid-Grid.ColumnDefinitions-Grid.RowDefinitions-ColumnDefinition-RowDefinition: Nos permite crear un Grid de x filas y x columnas (Tantas como **ColumnDefinition y RowDefinion** tengamos);

Para **colocar un objeto en el grid**, lo arrastramos, o le damos Grid.Column y Grid.Row (con números, para indicarle la celda que corresponde).

Los **Radiobutton** es necesario meterlos en un **StackPanel** para indicar que pertenecen a un mismo grupo y solo se pueda elegir uno de ellos.

A los **Combobox** se le añaden **Comboboxitem**, tantos como opciones se puedan escoger al respecto. La opción **selectedIndex=”-1”** hace que no haya ninguno seleccionado de antemano, si indicamos un número a partir de 0, cogerá el de esa posición. Al comboBox se le asigna a veces el **evento selectionChanged=”pepito”** que **realiza la acción cuando se cambia de opción** (usaremos un switch, lógicamente).;

Los **ListBox** funcionan con **ListBoxItem** y tienen la posibilidad de aplicar **SelectionMode= “multiple”.** Para elegir varios de ellos.

```
<StackPanel DockPanel.Dock="Bottom" Height="70" Width="auto" Background="Gainsboro">
        <Button x:Name="valida" Content="VALIDAR" Click="validar" Background="White"/>
</StackPanel>
```

```
      <StackPanel Width="475" DockPanel.Dock="Left" Background="LightCyan">
        <Grid>
          <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
          </Grid.ColumnDefinitions>
          <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
          </Grid.RowDefinitions>
          <Label Content="Nombre" Grid.Column="0" Grid.Row="0"></Label>
          <TextBox x:Name="nombre" Grid.Column="1" Grid.Row="0"></TextBox>
<CheckBox x:Name="mayor" Grid.Column="1" Content="Mayor de edad" HorizontalAlignment="Left" Margin="72,0,0,0" Grid.Row="1" VerticalAlignment="Center"/>
          <StackPanel Grid.Column="1" Grid.Row="2">
            <RadioButton x:Name="hombre" Content="Hombre" Width="75"/>
            <RadioButton x:Name="mujer" Content="Mujer" Width="75"/>
          </StackPanel>
        </Grid>
      </StackPanel>
      <StackPanel Width="475" DockPanel.Dock="Right" Background="Beige">
<ComboBox Width="120" Grid.Row="0" Grid.Column="1" x:Name="color" SelectedIndex="-1" SelectionChanged="pepito">
            <ComboBoxItem>Azul</ComboBoxItem>
            <ComboBoxItem>Rosa</ComboBoxItem>
 </ComboBox>
          <ListBox x:Name="deportes" Height="100" Grid.Row="1" Grid.Column="1" SelectionMode="Multiple">
            <ListBoxItem>Baloncesto</ListBoxItem>
            <ListBoxItem>Futbol</ListBoxItem>
          </ListBox> </Grid>.....
```

*************podemos aplicar un DatePicker para trabajar con Fechas**********************

CLASES

Para crear una clase nueva, vamos a PROYECTO/AGREGAR CLASE

A parte de los métodos que sean necesario implementar, es necesario crear los atributos (DateTime para la fecha), el constructor, y los get y sets necesarios. Estos últimos se obtienen haciendo *clic derecho en un atributo/Acciones Rápidas y refactorización/Encapsular campo y usar propiedad.*

```
public class Persona
{
//Atributos
   private string nombre;
   private bool mayor;
   private int edad;
  (private DateTime? fecha=DateTime.Now; o)  private DateTime fecha = new DateTime();
   private List<string> deportes=new List<string>();
//Constructor
   public Persona (string nombre, bool mayor, int edad, DateTime fecha,
List<string>deportes)
   {
      this.Nombre = nombre;
      this.Mayor = mayor;
      this.Sexo = sexo;
      This.Fecha = fecha;
      this.Deportes = deportes;
   }
//Getters y setters.
   public string Nombre { get => nombre; set => nombre = value; }
 }
```

***********Para crear una nueva ventana PROYECTO/AGREGAR VENTANA WPF************

METODO PARA AGREGAR UNA VENTANA

Ayuda help = new Ayuda();

help.Title = "Ventana 2"; -> Permite cambiar el título de la ventana.

help.Show(); --> Método para abrir la ventana creada instanciada anteriormente.

METODO PARA ESCRIBIR EN UN TXT

**********El using nos permite que se continue escribiendo, y no que se sobrescriba. ********** **

private void escribirPersona(Persona per) -> Le pasamos una persona para escribirla

{

using (StreamWriter fichero = File.AppendText("personas.txt"))

{

fichero.WriteLine("Nombre: " + per.Nombre); --> Escribirmos en cada linea los atributos

fichero.WriteLine("Edad: " + per.Edad);

fichero.Write("Sexo: "); -> Aquí sin salto de linea.

if (per.Sexo)

{

fichero.WriteLine("hombre"); - > Según lo que nos devuelva sexo escribe una cosa u otra.

}

else

{

fichero.WriteLine("mujer");

}

fichero.WriteLine("¿Mayor de edad? " + per.Mayor);-> En este caso me escribirá true o false;

fichero.Write("Deportes: ");

for (int i = 0; i < per.Deportes.Count; i++)

{

fichero.WriteLine(per.Deportes[i] + " "); -> Salen más cosas que el deporte seleccionado. (problemas)

}

fichero.Close(); -> Importantísimo no olvidar cerrar el fichero.

}}

METODO PARA LIMPIAR

```
public void limpiar()
{
   nombre.Clear();              -> Textbox o Textblock;
   color.SelectedIndex = -1;          -> Combo o List.
   mayor.IsChecked = false;           -> Radiobutton o checkbox
  }
```

METODO PARA OBTENER UN ID LEYENDO TXT

```
private int obtenerCodigo() -> Obviamente, devuelve un entero;
{
    int cod = 0;
    StreamReader fichero;                    -> Fichero a leer
    string linea;                            -> String que nos recoge la linea a leer.
    if (File.Exists("personas.txt"))         -> Si existe el documento.
    {
       fichero = File.OpenText("personas.txt");      -> Lo abrimos.
       linea = fichero.ReadLine();                   -> Leemos la primera linea
       while(linea != null)                          -> Mientras no se acabe el documento.
       {
          if(linea.StartsWith("Codigo: "))           -> Si la linea empieza por codigo.
          {
             cod=int.Parse(linea.Substring(8));      -> Código pasa a ser el carácter 8;
          }
          linea = fichero.ReadLine();                -> Si no, continuamos leyendo
       }
       fichero.Close();                              -> Al acabar cerramos el fichero.
    }
    cod++;                                           -> Le sumamos 1
```

return cod; -> Y devolvemos el código obtenido.}

METODO PARA ABRIR UNA VENTANA DE DIALOGO, ABRIR UN TXT Y MOSTRARLO

```
private void abrir(object sender, RoutedEventArgs e)
{
  Microsoft.Win32.OpenFileDialog abrirtxt = new Microsoft.Win32.OpenFileDialog();
**instanciamos la ventana para abrir documentos**
  abrirtxt.DefaultExt = ".txt";
  abrirtxt.Filter = "Documentos de texto (.txt)|*.txt";
  Nullable<bool> result = abrirtxt.ShowDialog();
**Filtramos para que sólo se puedan abrir txt y abrimos la ventana**
  if (result == true)    **Si abrimos un txt**
  {
     string documento = abrirtxt.FileName;              **Almacenamos la cadena del documento**
     StreamReader fichero;   **Creamos una variable para leer el fichero**
     fichero = File.OpenText(documento);  ** Fichero nos abre el documento indicado **
     string contenido = "";
     string linea;
     do
     {
        linea = fichero.ReadLine();     ** Mientras haya líneas leemos**
        if (linea != null)
        {
           contenido += " " + linea;     **Y lo añadimos a la variable contenido
        }
     } while (linea != null);
     copia.Text = contenido; **Y en el textbox que queremos, escribimos el contenido. **
     fichero.Close();
```

```
    }
}
```

METODO PARA ABRIR UNA VENTANA DE DIALOGO Y ESCRIBIR UN TXT Y GUARDARLO

```
**Muy parecido al anterior, solo comentamos detalles diferentes**
private void guardar(object sender, RoutedEventArgs e)
{
    Microsoft.Win32.SaveFileDialog guardar = new Microsoft.Win32.SaveFileDialog();
//Cambiamos el OpenFileDialog por SaveFileDialog();
    guardar.DefaultExt = ".txt";
    guardar.Filter = "Documentos de texto (.txt)|*.txt";
    Nullable<bool> result = guardar.ShowDialog();
    if (result == true)
    {
        string documento = guardar.FileName;
        StreamWriter fichero;
//El StreamReader por StreamWriter;
        fichero = File.CreateText(documento);
//El openText por CreateText
        fichero.WriteLine("Soy el mejor");
        fichero.Write("y nadie puede negarlo");
//Y escribimos como ya sabemos
        fichero.Close();
    }
}
```

PREGUNTAR SI SE SALE DE LA APLICACIÓN

```
private void salir(object sender, RoutedEventArgs e)       {
    MessageBoxResult respuesta = MessageBox.Show("¿Deseas salir de la aplicación?", "Salir o escribir", MessageBoxButton.YesNo, MessageBoxImage.Question);
```

```
    if (respuesta == MessageBoxResult.Yes)  {
       Application.Current.Shutdown(); //Salir de la aplicación
    }**Es una ventana que te pregunta y si la respuesta es sí, entonces sales del programa*
```

VALIDACIONES

```
//Vamos a mostrar la numérica por separado.
public bool validando()
{
    bool ok = true;
    if (string.IsNullOrWhiteSpace(nombre.Text))     //En cajas de texto sólo si esta vacio.
    {
       ok = false;
       MessageBox.Show("El nombre no puede estar vacío");
    }
     DateTime fech;
     else if(DateTime.TryParse(Fecha.Text, out fech)){       //Para fechas si es menor a hoy
            if (fech < DateTime.Now.Date)
            {
               ok= false;
      MessageBox.Show("La fecha no debe ser anterior a la actual");
            }
         }
       else if (color.SelectedIndex == -1)                  //Si combobox o list no marcados.
    {
       ok = false;
       MessageBox.Show("Debes elegir uno de los colores");
    }
    else if ((bool)!hombre.IsChecked && (bool)!mujer.IsChecked)
    {                                                       //Si no marcamos ningún radio.
       ok = false;
```

```
        MessageBox.Show("Debes seleccionar un sexo");
    }
     return ok;
}
```

VALIDAR EDAD

****Importante el uso de TryParse para validaciones de todo tipo que no sean cadenas. ****

```
public bool validarEdad()
{
   bool correcto = false;
  if (int.TryParse(edad.Text, out eda) && (eda > 0 && eda < 120))
   {                        **-> Pasamos el texto de la caja edad a la variable int edad...**
      correcto = true;
   }
   return correcto;
}
```

OBTENER UN OBJETO A PARTIR DE LAS CAJAS, COMBOS, RADIOS...

```
private Persona agregarPersona()
{       //Obtenemos cada uno de los datos de las cajas y lo agregamos al constructor
    nom = nombre.Text;
    eda = int.Parse(edad.Text);
    sex = false;
    DateTime? fec = Fecha.SelectedDate;
    if ((bool)hombre.IsChecked) {
       sex = true;
    }
    dep = new List<string>();           //Para obtener la cadena debemos el array
    foreach (var item in deportes.SelectedItems) {
```

```
        string deporte = item.ToString();
        dep.Add(deporte);}
    Persona per = new Persona(nom, may, sex, eda, col, dep);
    return per;}
```

ACTUALIZAR UN FICHERO

```
private void actualizar()
{
   StreamReader fichero;
   StreamWriter auxiliar;
   string n = nomb.Text;                    //Obtenemos el nombre de la caja
   int c;                                          //Este será el codigo de la otra caja
   if (int.TryParse(numero.Text, out c))
   {
      if (File.Exists("Personas.txt"))
      {
         using (fichero = File.OpenText("Personas.txt"))
         using (auxiliar = File.CreateText("auxiliar.txt"))      //necesitamos ambos using
         {
            string linea = fichero.ReadLine();
            while (linea != null)
            {
               auxiliar.WriteLine(linea);
               if (linea.Contains("Codigo: " + c))
               {
                  linea = fichero.ReadLine();
                  auxiliar.WriteLine("Nombre: " + n);
               }//Copiamos linea a linea salvo cuando coincida el codigo que cambia nombre.
               linea = fichero.ReadLine();
```

```
            }
            fichero.Close();
            auxiliar.Close();                    //Cerramos ambos ficheros
            File.Delete("Personas.txt");         //Borramos el antiguo
            File.Move("auxiliar.txt", "Personas.txt"); //Renombramos el auxiliar.
            MessageBox.Show("Usuario actualizado");
        }
      }
   }
}
```

VER INFORMACIÓN EN UNA CAJA DE UNA PERSONA DEL FICHERO

```
private void visualizar()
{
   int x;
   if (int.TryParse(resultado.Text, out x));
   StreamReader fichero;
   string linea;
   if (File.Exists("personas.txt"))
   {
      fichero = File.OpenText("personas.txt");
      linea = fichero.ReadLine();
      String esc = "";                           //Donde almacenaremos la info
      while (linea != null)
      {
         if(linea.StartsWith("Codigo: " + x))          //Leemos y si coincide el codigo.
         {
```

```
            linea=fichero.ReadLine();
            while(linea!=null && !linea.StartsWith("Codigo: "))
            {                   //Escribimos todas las lineas hasta llegar al siguiente código.
               esc += linea + " ";
               linea=fichero.ReadLine() ;
            }
         }
         linea = fichero.ReadLine();
      }
      resultado.Text = esc;      //La caja recibe el texto de nuestra variable.
      fichero.Close();           //Cerramos el fichero.
   }
}
```

NUMERO DE TAREA EN UNA LISTA DE TAREAS

```
private List<Tarea> tareas;
tareas = new List<Tarea>();
---------------------------------------------- -> Hablamos de esta lista de tareas creada como atributo
private int numeroTarea()
      {
         int x = tareas.Count;
         int y = 0;
         if (tareas.Count == 0)
         {
            y = 1; }
         Else {
            y = tareas.Max(t => t.Numero) + 1;  //Coge el número máximo de tarea;}
         return y;}
```

Environment.Exit(0) -> Salir;

listBox1.Items.Add("Elemento 1"); -> Añadir un elemento a un listbox.

int indiceSeleccionado = listBox1.SelectedIndex; -> Selecionamos el indice de la lista que se ha seleccionado.

listBox1.Items.RemoveAt(indiceSeleccionado);-> Borramos el seleccionado.

MOSTRAR Y OCULTAR BOTONES

```
private void mostrarBotones()
    {
       if (tareas.Count == 0)
       {
          elegir.Visibility = Visibility.Hidden;          //Eliminar visibilidad
          eliminar.IsEnabled = false;                     //Deshabilitar
       }
       else
       {
          elegir.Visibility = Visibility.Visible;
          eliminar.IsEnabled = true;
       }
    }
```

---

lista.Items.Clear(); -> Borrar items de una lista.

MÁS FECHAS

DateTime hoy=DateTime.Now; **-> obtener la fecha de hoy**

TimeSpan diferencia = hoy - fec

anti=(int)(diferencia.Days /365.25 **->obtener la antigüedad en años**

VER TAREAS EN UN JLIST

```
private void verTareas(List<Tarea> tareas)
    {
        int n = elegir.SelectedIndex; //Indicamos la opcion del combo que se ha seleccionado.

        if (tareas != null)
        {
            switch (n)
            {
                case 0:         //Recorrer y mostrar cada posición
                    for (int i = 0; i < tareas.Count; i++) {
                        string ur=ponerUrgente(tareas[i]);
                        string ca=ponerLugar(tareas[i]);
                        string re= ponerRealizada(tareas[i]);
                        lista.Items.Add(tareas[i].Nombre + " " + tareas[i].Descripcion + " " + re +" " + ur + "
" + ca + " " + tareas[i].Fecha);
                        eliminar.IsEnabled = true;
                        tarearealizada.Visibility = Visibility.Visible;
                    }
                    break;
                case 1://Este mostrará sólo los casos urgentes.
                    for (int i = 0; i < tareas.Count; i++  {
                        string ur = ponerUrgente(tareas[i]);
                        string ca = ponerLugar(tareas[i]);
                        string re = ponerRealizada(tareas[i]);
                        if (tareas[i].Urgente)
```

```
                lista.Items.Add(tareas[i].Nombre + " " + tareas[i].Descripcion + " " + re  + " " + ur +
" " + ca+" " + tareas[i].Fecha);
                eliminar.IsEnabled = true;
                tarearealizada.Visibility = Visibility.Visible;
            }
            break;
```

ELIMINAR DE UN JLIST

```
private void fulminar(object sender, RoutedEventArgs e)
    {
        int seleccionado = lista.SelectedIndex;                    //lista
        int opc = elegir.SelectedIndex;                            //opcion del combobox
        int num = -1;
        bool enc = false;
        switch (opc)                                               //segun la opcion del combo...
        {
          case 1:          //Eliminamos la tarea que coincida con el num seleccionado siempre y
cuando dicha tarea esté realizada
            for (int i = 0; i < tareas.Count && enc == false; i++)
            {
              if (tareas[i].Urgente)
              {
                num++;          //Este num tiene que coincidir con el de seleccionado para que solo
se elimine ese.
                if(num == seleccionado && tareas[i].Realizada)
                {
                  tareas.Remove(tareas[i]);
                  lista.Items.Clear();
                  verTareas(tareas);
                  MessageBox.Show("Tarea eliminada");
                  enc = true;
```

```
                }
              }
            }
            break;
if (enc == false)//Si no se encuentra, la tarea no estará realizada, por lo que no se elimina.
        {
           MessageBox.Show("No se puede eliminar una tarea no realizada")   }
```

PASAR DE NO REALIZADA A REALIZADA LA TAREA

```
case 4:
   for (int i = 0; i < tareas.Count && enc == false; i++)
   {
      if (tareas[i].Tipo)
      {
         num++;
         if (num == seleccionado && tareas[i].Realizada == false)
         {
            tareas[i].Realizada=true;        //Si coincide y no esta realizada se pasa a realizada al darle
            lista.Items.Clear();
            verTareas(tareas);
            MessageBox.Show("Tarea marcada como realizada");
            enc = true;
         }
      }
   }
   break;
```

APUNTES C#

En **ctrl+alt+l** tenemos el explorador de soluciones (también en ver).

Title="Cursos" Height="450" Width="800"> -> redimensionar el tamaño de la ventana.

En **ctrl+alt+x** o en **ver/cuadro de herramientas**

StackPanel, **margin**, **orientation** **VerticalAligment** **IsReadOnly** **IsEnabled** (atriabutos útiles)

ventana de propiedades también se puede acceder desde **ver/ventana propiedade**s o tecla **F4.**

Para mostrar los cursos, lo que hacemos es **recorrer cada uno de los items de la lista con un for** (lstCursos.Items.Add(cursos[i].Titulo).

Separator simplemente crea una línea de separación en el menú.

Icon: Permite añadirle el icono a la parte superior izquierda.

ResizeMode = NoResizable -> Permite evitar el cambio de tamaño de la ventana.

WindowsStartup location -> Controla la posición de la ventana. Si le indicamos **CenterScreen**, irá al centro.

CANVAS

<Canvas>

<Ellipse Fill="Gainsboro" Canvas.Left="25" Canvas.Top="25" Width="200" Height="200" />

<Rectangle Fill="LightBlue" Canvas.Left="25" Canvas.Top="25" Width="50" Height="50" />

<Rectangle Fill="LightCoral" Canvas.Left="50" Canvas.Top="50" Width="50" Height="50" />

<Rectangle Fill="LightCyan" Canvas.Left="75" Canvas.Top="75" Width="50" Height="50" />

</Canvas>

Colocas **cada objeto en la posición concreta** que indicas con Top, Left, Right...

ZINDEX -> Propiedad que permite colocar objetos uno debajo de otro. Cuanto mayor sea el valor del ZIndex, más arriba irá en el panel. En caso de mismo número, el definido posteriormente gana.

<Ellipse Panel.ZIndex="2" Fill="Gainsboro" Canvas.Left="25" Canvas.Top="25" Width="200" Height="200" />