



# ACTIVIDAD 01 – WPF Y XAML

Creación de una aplicación WPF

DESARROLLO RÁPIDO DE APLICACIONES

Por Juan José Camacho Hidalgo  
UNIVERSIDAD DE ALMERÍA

## Contenido

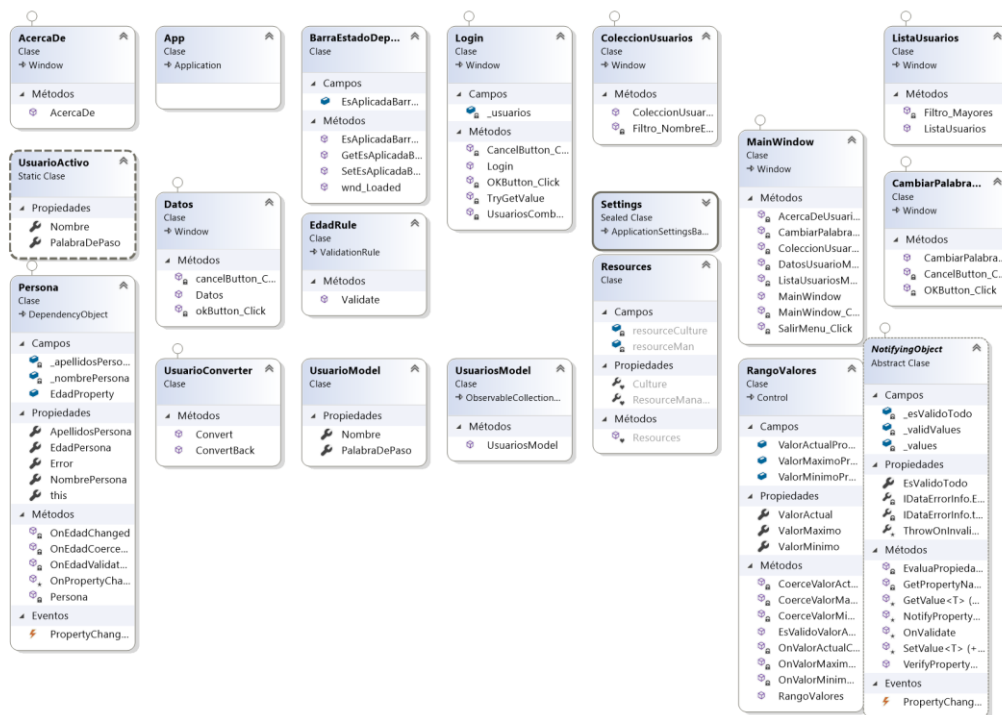
1. Introducción .....	2
2. HolaMundoWpfApplication .....	2
3. ValidacionWpfApplication.....	6

## 1. Introducción

En esta actividad se realiza una aplicación Windows Presentation Foundation (WPF) que incluye XAML (lenguaje de marcas de aplicación extensible), un lenguaje declarativo basado en XML cuyo objetivo es la realización de la interfaz gráfica textual y ordenadamente, y código subyacente (code-behind).

## 2. HolaMundoWpfApplication

El diagrama de clases de la aplicación es el siguiente:



Todo lo escrito en XAML, es referente a la interfaz de usuario MainWindow, Login, ListaUsuarios, Datos, ColeccionUsuarios y CambiarPalabraPaso son las diferentes ventanas que el programa crea, mientras que NotifyingObject, RangoValores, UsuarioActivo, UsuarioModel, BarraEstadoDependencyProperty, son las clases que llevan el funcionamiento de los diferentes componentes de la interfaz.

Para establecer una página como página que se abre por defecto al iniciar la app, usamos la propiedad StartupUri de Application.

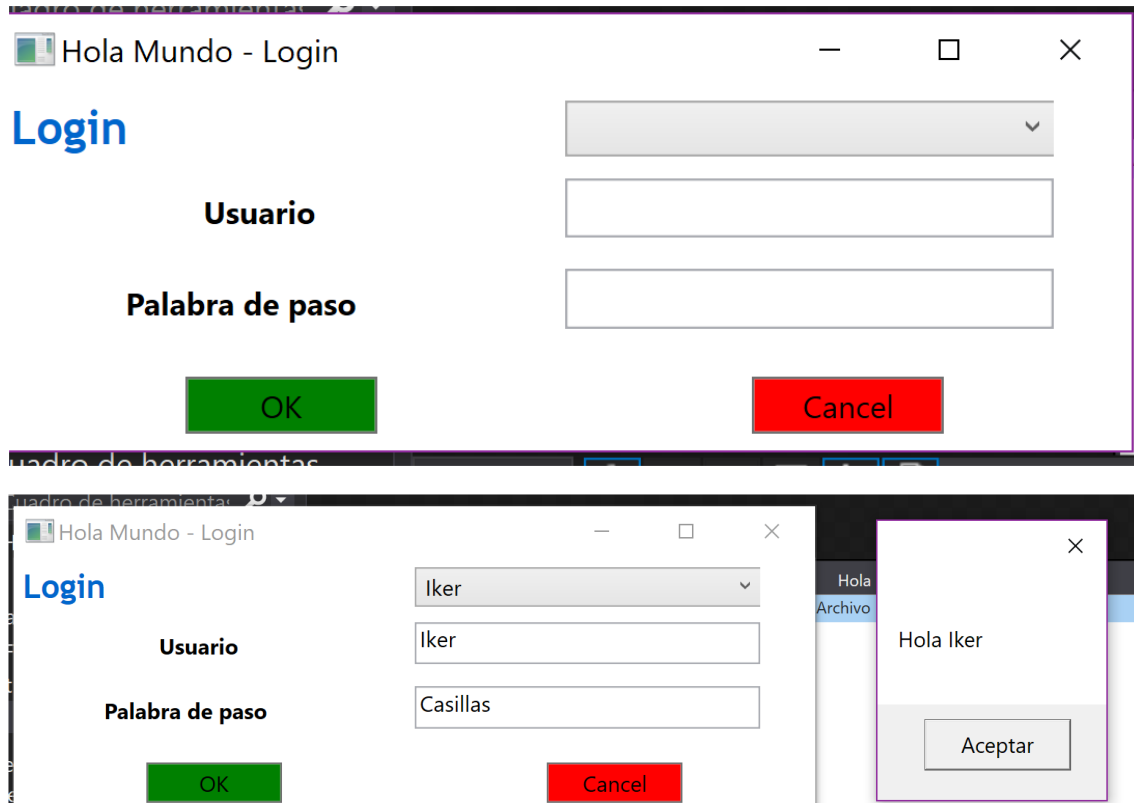
Para el estilo de los componentes de la IU, hacemos uso de los recursos (resources). Cada elemento tiene una propiedad de resource que hay que enlazar mediante una key de acceso al ResourceDictionary de forma única.

En este código se han añadido headerTextStyle, labelStyle, okButtonStyle y cancelButtonStyle. Para ello se han hecho uso de los componentes que usan esos recursos, como button o label, entre otros.

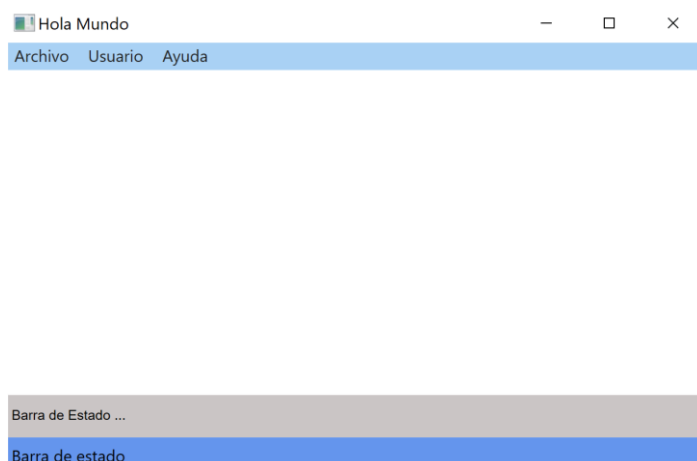
En cuanto a eventos en WPF hay tres tipos: bubbling, tunneling y direct, siendo los dos primeros los más frecuentes. Direct, significa que solo el elemento al que hace referencia

puede ser notificado por el evento, bubbling puede ser cualquier elemento por encima de él notificado, suponiendo un árbol visual de la UI, y tunneling todo elemento por debajo de él.

La clase UsuariosModel crea un listado de usuarios que tienen acceso a la aplicación, incluyendo su nombre y su palabra de paso y en la pantalla de login, en el .cs, se gestionan los errores que tienen que mostrarse si el usuario introducido no pertenece a ese listado.



Si es correcto te da la bienvenida y te redirige hacia el panel de usuario, que viene a ser la clase MainWindow:

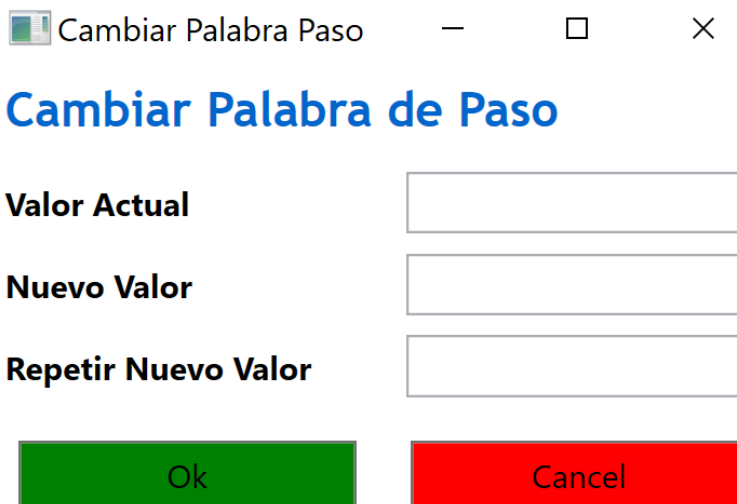


A destacar la creación del menú con la propiedad DockPanel.Dock y las opciones de este con la etiqueta MenuItem. También incluye la clase BarraEstadoDependencyProperty, la cual crea una barra de estado utilizando DependencyProperty, que es un tipo de dependencia propia de

WPF donde lo que hace es gestionar si el usuario cambia el estado, si es así, la modifica, incluso añadiendo elementos nuevos.

La clase CambiarPalabraPaso nos permite cambiar ese valor de entrada de un usuario a la app.

Para ello, se desarrolla un método encargado de ello según los valores del texto introducido y siempre que el valor actual coincida.



**Cambiar Palabra Paso**

**Valor Actual**

**Nuevo Valor**

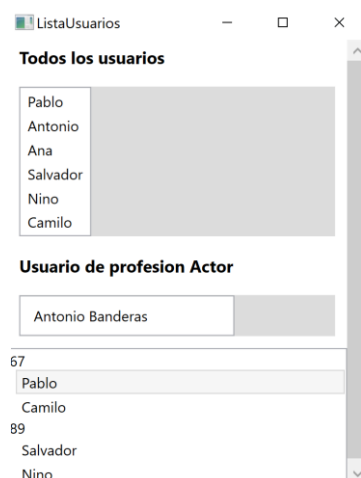
**Repetir Nuevo Valor**

**Ok** **Cancel**

Para referenciar a una clase desde XAML hacemos uso de la siguiente sintaxis:

xmlns : prefijo = "clr-namespace : Espacio de Nombre ; assembly = Nombre del ensamblado".  
Esto es muy útil para obtener la lista de usuarios por ejemplo.

Esta es la clase ListaUsuarios, donde carga los usuarios, y especialmente los actores



**ListaUsuarios**

**Todos los usuarios**

Pablo  
Antonio  
Ana  
Salvador  
Nino  
Camilo

**Usuario de profesion Actor**

Antonio Banderas

67  
Pablo  
Camilo  
89  
Salvador  
Nino

Para los usuarios hace uso de la técnica de binding llamando a uno de los recursos, y obteniendo el nombre:

```

<ListBox ItemsSource="{Binding Source=
{StaticResource xmlListaUsuarios},
XPath=Usuario/Nombre}"/>

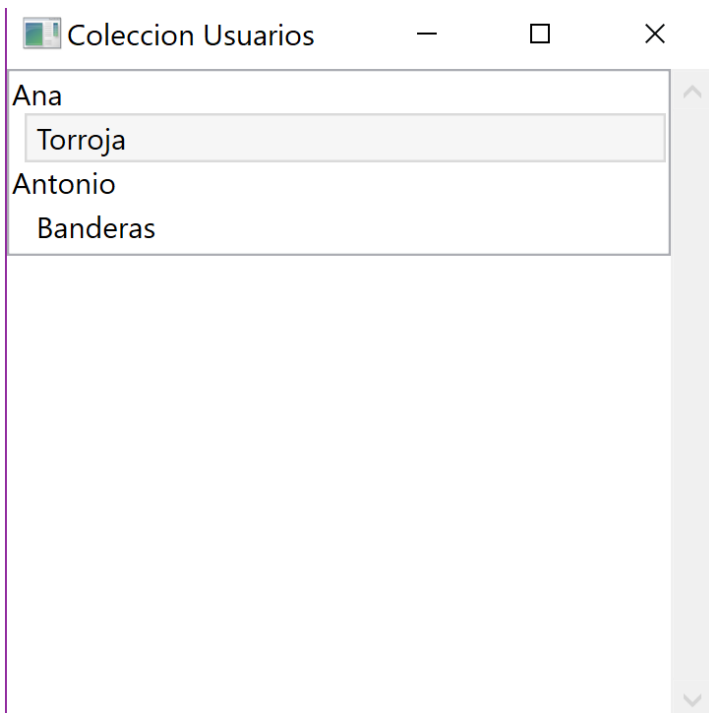
```

Para la clase de colección usuarios hace algo similar:

```

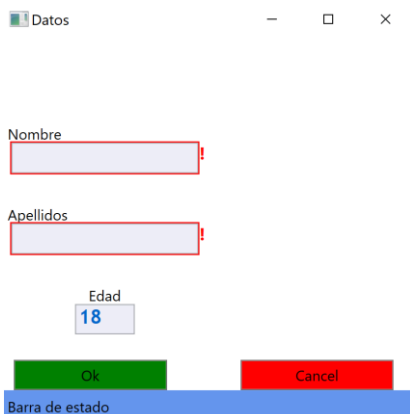
ItemsSource="{Binding Source={StaticResource LoginUsuariosView}}"

```



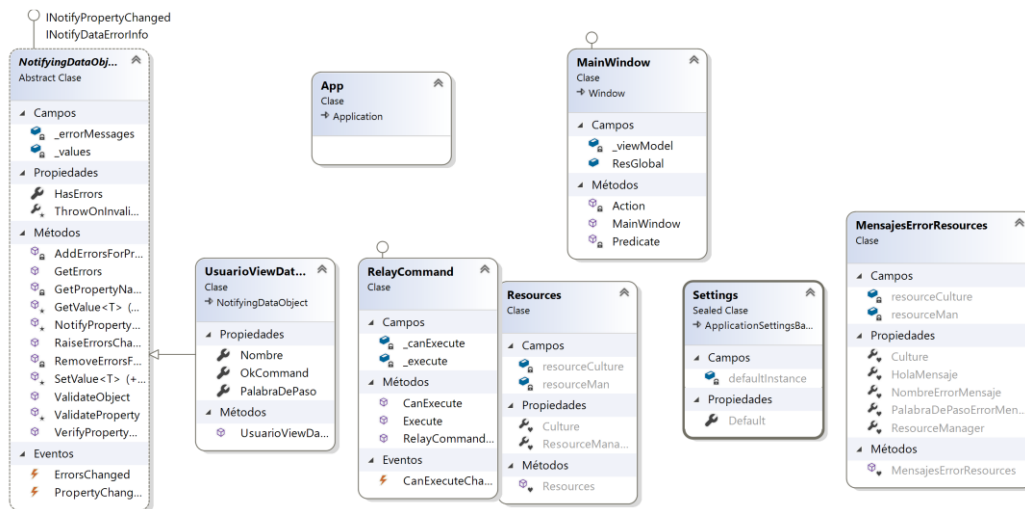
Por último sobre la clase datos, el usuario interacciona insertando nombre y apellidos y la aplicación devuelve una emergente con los datos (también la edad introducida).

Para soportar binding en modo one-way y two-way, la clase persona, incluida en Datos, implementa INotifyPropertyChanged, y los elementos se actualizan dinámicamente.



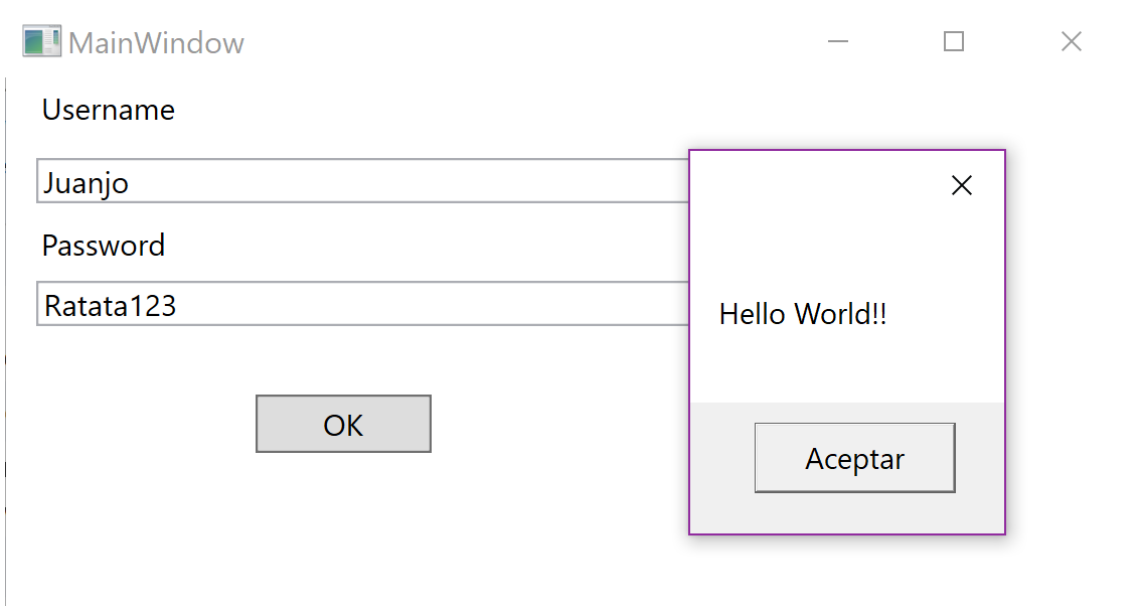
### 3. ValidacionWpfApplication

Aquí disponemos de el diagrama de clases de la aplicación:



Básicamente esta aplicación es un sistema de validación de login, mediante WPF, que valida los datos según nombre de entre 5 y 15 valores alfanuméricos, y contraseña entre 4 y 10 caracteres, una mayúscula y una minúscula al menos, y un dígito, para fomentar la seguridad.

Si introduces correctamente los valores, emerge una ventana diciendo “Hello World!!”.



Para llegar a desarrollar esta aplicación , y como vemos en el diagrama, se ha realizado una clase de MensajesErrorResources, para los estilos que tienen que mostrarse cuando haya algún error en la validación, en forma de exclamación y de color rojo. La clase RelayCommand, tiene como objetivo comprobar si un comando puede ejecutarse, y si es así ejecutar un evento para llevar a cabo el comando. Es un intermediario de comprobación. El modelo de usuario, lo lleva a cabo UsuarioViewDataModel, que se encarga de ofrecer a la aplicación las propiedades y métodos de un usuario cualquiera. La ventana principal de validación es MainWindow, por lo que aquí se muestra todo el xaml llamado desde App que es la ventana ejecutora de la

aplicación. Y por último, `NotifyingDataObject`, se encarga de tratar el estado de los objetos, y gestionar los errores. Para ello implementa dos interfaces, la de `INotifyPropertyChanged`, y `INotifyDataErrorInfo`, una para cuando cambie el formato del texto introducido y otra para comprobar los datos si son correctos con respecto al almacenamiento de datos del usuario.