MANUAL TÉCNICO BCNCATALOG

To: Bcnvision

Date: 11/08/2022

Tel.:93 719 49 35/ Fax: 93 719 89 31/ E-Mail:

1. Resumen

En este documento se resume cómo se organiza el código de la Aplicación BcnCatalog.

1.1 Requisitios Previos

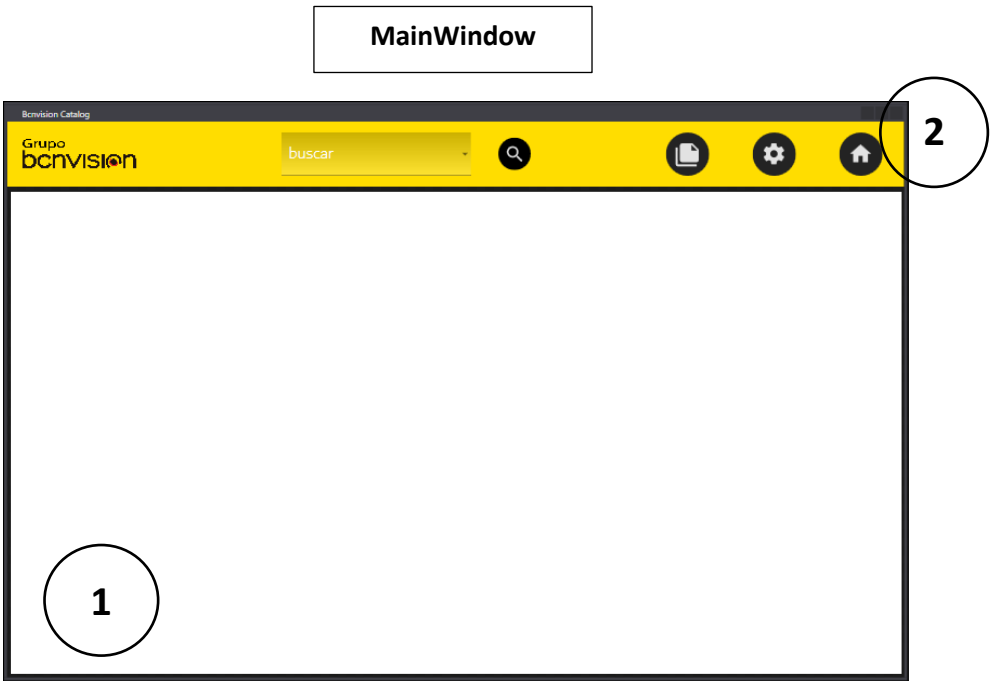
Para usar la aplicación, es necesario tener 2 softwares adicionales que no se instalan automáticamente, :

- **Veracript:** para el encriptado y desencriptado del contenido
- **FreeFileSync:** para la actualización de contenido, copiando desde el servidor.
- **Carpeta BcnCatalog en "H:**"

2. Estructura Básica

El **MainWindow** es la clase principal, una especie de Core encargado de:

- Mostrar los formularios principales en su grid principal (1).
- Contiene el Header, lo que permite controlar las acciones sobre los botones de configuración, settings, versiones… (2)
- Gestiona otros elementos de la aplicación mediante clases, como las rutas (PathManager), los datos serializables (DataXml), el encriptado de contenido…

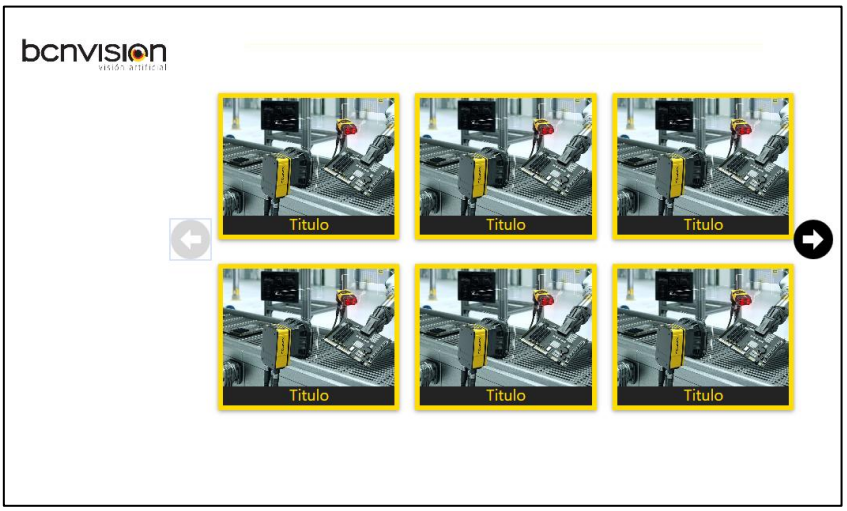


2.1 Windows y Views

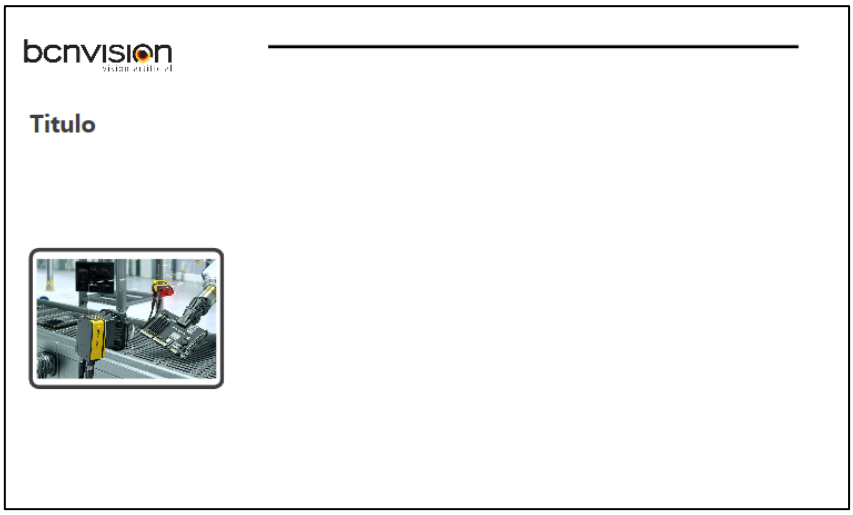
Los elementos que componen la parte visual de la aplicación pueden clasificarse en 2 tipos:

- Por un lado tenemos cuatro Windows que se incrustan en el grid principal del MainWindow, y que están diseñadas para seguir la exploración de carpetas y archivos que el usuario realiza sobre la info que hay en su PC (el contenido). Estas cuatro ventanas son:
  - **WPFMain**: representa que estamos en la carpeta principal, que contiene las 4 carpetas contenedoras de los 4 menús principales que por defecto maneja la aplicación. Muestra 1 de los 4 controles personalizados que se han creado como "introducción" al menú completo en su zona central (1).

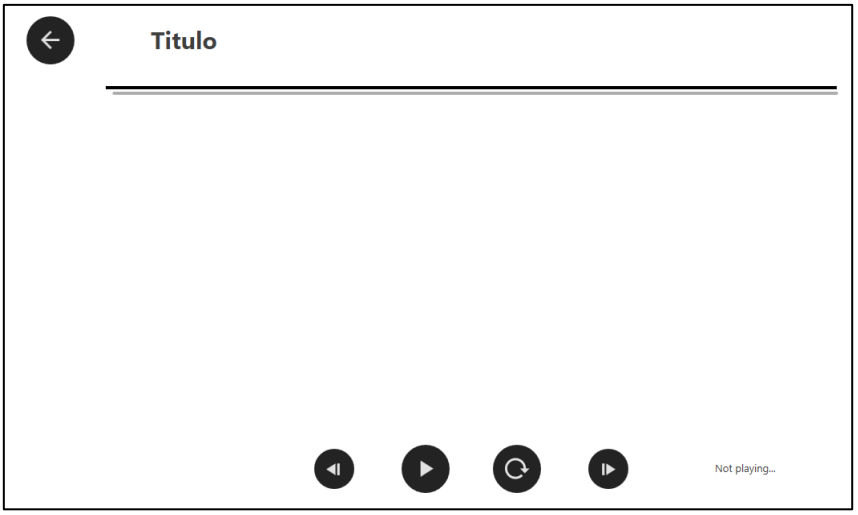
  - **WPFExplorer**: representa que estamos en una subcarpeta que, a su vez, contiene otras carpetas (y no contenido o archivos).



- **WPFContent**: representa que estamos en una subcarpeta que no contiene subcarpetas, sino que contiene archivos (pdfs, htmls, pptxs o mp4). En esta ventana se muestra una lista con todo lo que hay en la carpeta, siempre y cuando sea un archivo de un tipo aceptado.



- **WPFHost**: representa que estamos visualizando un archivo.



➢ Por otro lado, las ventanas que no se incrustan, sino que se muestran de forma independiente, serían las siguientes:

- **WPFBuscador:** Para mostrar los resultados de búsqueda.
- **WPFLanguaje:** Para cambiar el modo de idioma y de empresa en la aplicación. Los cambios de idioma se aplican al instante, los de empresa se configuran al cierre de la aplicación, para la siguiente vez que se inicie.

- **WPFStart:** Ventana que se muestra al inicio, para introducir usuario y contraseña.
- **WPFVersions**: Ventana que permite al usuario comprobar si existe una nueva versión disponible para el contenido, e iniciar la actualización en paralelo.

3. Funcionalidades

En este apartado explico cómo funcionan algunas partes del software:

3.1 Buscador

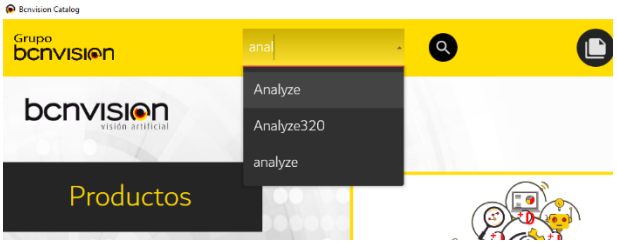
El buscador de la aplicación funciona de la siguiente forma:

- Al inicio de la aplicación, CADA VEZ, se exploran todas las carpetas en busca de archivos .xml, y se extraen todas las Tags en un Dictionary que contiene, además, las rutas de los archivos asociados a cada Tag.

```
//Finalmente vamos a actualizar los tags
data.TagsDataP = ExplorarTagsToJsonFile();
data.ExportTagsData(data.TagsPathXML);
```

(Constructor MainWindow)

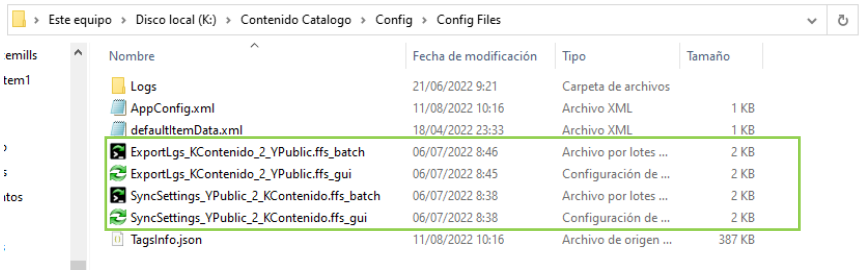
- Con el Dictionary ya creado, cuando buscamos una palabra en el buscador, solo hay que mostrar como opciones del combobox los tags que contengan los caracteres introducidos (a partir de 3 caracteres), y cuando se busca uno, mostrar los achivos asociados a los tags que cumplan esto mismo.



- El método “Buscar()” hace esta selección de archivos extraídos del dictionary, además de corregir en función a la existencia o no de los archivos que pretendemos buscar, volviendo a serializar el archivo json que contiene el dictionary.

3.2 Control de Versiones

El control de versiones de la aplicación consiste en la ejecución de unos archivos preparados, que están en la carpeta del Config que son .batch generados con el software FreeFileSync.



Los archivos son dos: uno para la exportación de Logs (se ejecuta al cierre de la aplicación) y otro para actualizar el contenido desde el servidor (se ejecuta cuando el usuario decide actualizar).

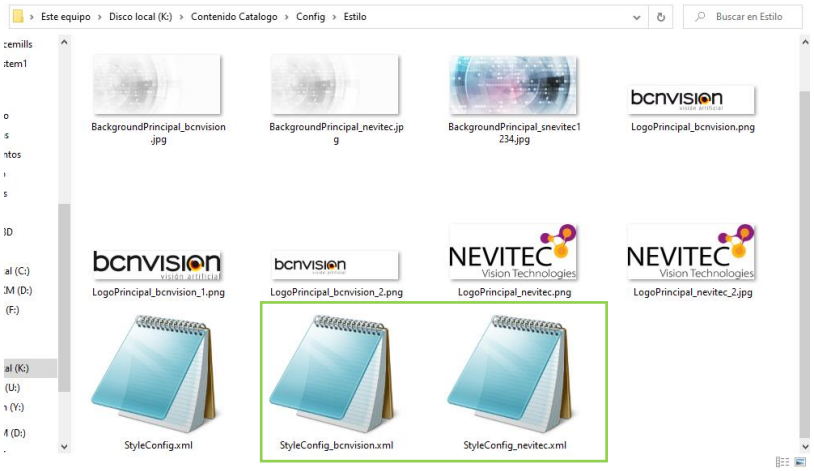
Para cambiar la ruta del servidor que contiene el Contenido del Catálogo, habría que modificar este archivo en el servidor, y pedir a los comerciales que actualicen.

3.3 Configuración: Empresa e Idioma

La ventana de Settings permite seleccionar el modo empresa y modo idioma.

Esto cambiará los títulos de la aplicación al idioma que toque de forma automática. Para que un título tenga traducción (títulos de carpetas), debe existir un archivo .xml asociado, que está dentro de la carpeta, y que comparte nombre con dicha carpeta. Por el contrario, el título que se muestra es el de la carpeta.

Para los títulos de los botones de la pantalla principal y los menús, hay archivos de serialización contenidos en la carpeta Estilo. Estos archivos son independientes por modo de empresa.



El archivo StyleConfig.xml es solo una plantilla.

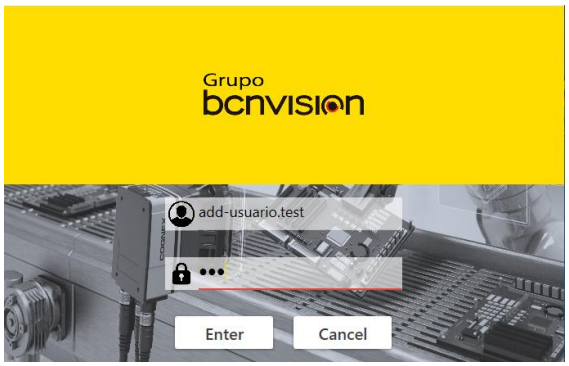
Cabe destacar que toda la carpeta de estilo se extrae del archivo encriptado y se vuelca a la H al inicio de la App, desde donde se carga cuando el usuario se loguea.

Con esto, si en la actualización de la App el equipo de Marketing ha cargado un fondo de pantalla o logo nuevo, se puede sobrescribir en la carpeta encriptada sin conflictos de que los archivos estén siendo usados por la aplicación.

3.4. Control de Usuarios

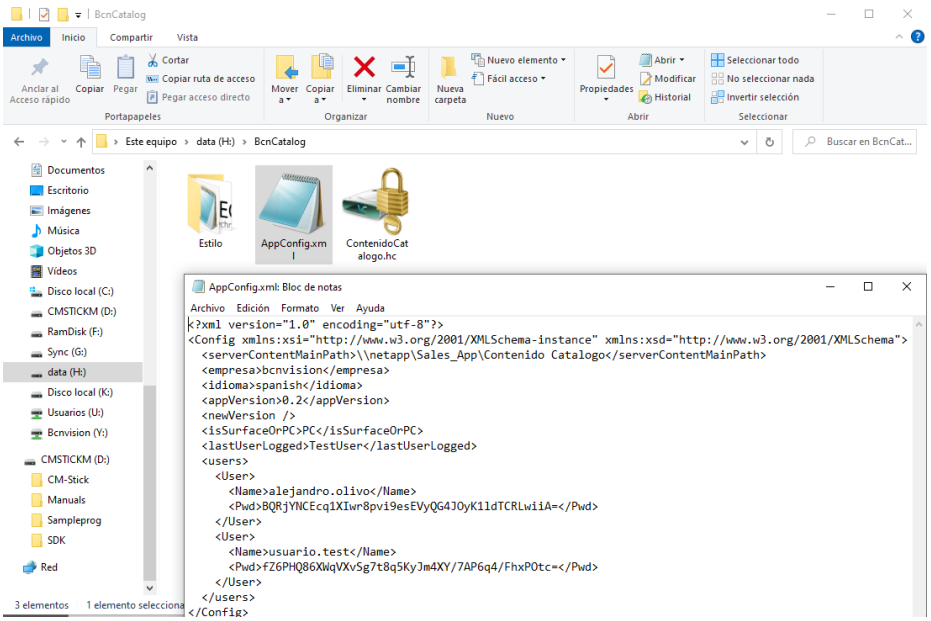
Para crear un usuario nuevo, habrá que:

- En el formulario de inicio, escribir como usuario: Add-*NOMBRE_DE_USUARIO* y la contraseña que se desee poner.



- Para permitir la creación de usuario, nos pedirá una segunda contraseña “de MK”, que será “bcnvision2022”, y que permitirá la creación del usuario. Un comercial no puede crear su propio usuario sin la supervisión de MK o Albert Gimenez.

Para el guardado del usuario y la contraseña, se hace uso de la clase BcnCryptography



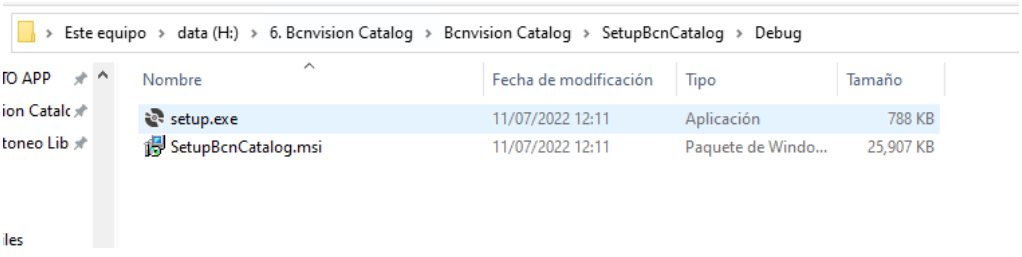
La info está guardada en el App.Config, que está en la ruta H:/BcnCatalog.

En esta ruta encontramos el archivo encriptado, así como la copia de la carpeta estilo comentada anteriormente.

3.5. Proyecto Setup

Dentro de la misma solución, podemos encontrar el proyecto SetupBcnCatalog, para generar el instalable.

OJO. La compilación se hace en la ruta relativa donde esté el código, no a la C.



3.6. Serialización y Clases

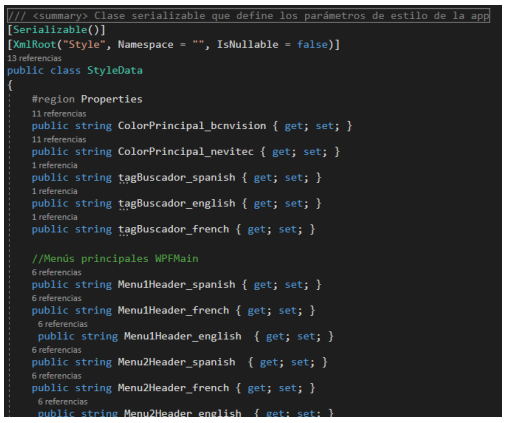
En la aplicación se usan 2 métodos de serializado, todo recogido en las clases DataXML.cs y SerializationManagement.cs:

- Serialización .xml para:
  - Parámetros de Configuración generales: ConfigData

```
/// <summary> Clase serializable que define los parámetros de configuración de l ...
[Serializable()]
[XmlRoot("Config", Namespace = "", IsNullable = false)]
16 referencias
public class ConfigData
{
    #region Properties
    3 referencias
    public string empresa { get; set; }
    13 referencias
    public string idioma { get; set; }
    2 referencias
    public string appVersion { get; set; }
    4 referencias
    public string newVersion { get; set; }
    8 referencias
    public string isSurfaceOrPC { get; set; }
    1 referencia
    public string lastUserLogged { get; set; }
    2 referencias
    public ObservableCollection<User> users { get; set; }

    public string serverContentMainPath = "";
```

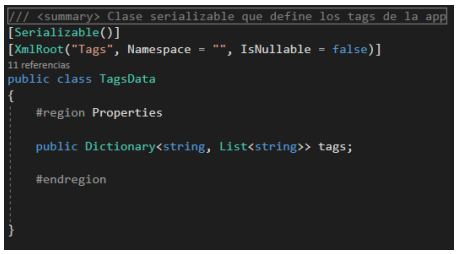
  - Parámetros de estilo: títulos y descripciones por menú, color principal y traducción de tags como “Buscar…”



```
/// <summary> Clase serializable que define los parámetros de estilo de la app
[Serializable()]
[XmlRoot("Style", Namespace = "", IsNullable = false)]
13 referencias
public class StyleData
{
    #region Properties
    11 referencias
    public string ColorPrincipal_bcnvision { get; set; }
    11 referencias
    public string ColorPrincipal_nevitec { get; set; }
    1 referencia
    public string tagBuscador_spanish { get; set; }
    1 referencia
    public string tagBuscador_english { get; set; }
    1 referencia
    public string tagBuscador_french { get; set; }

    //Menús principales WPFMain
    6 referencias
    public string Menu1Header_spanish { get; set; }
    6 referencias
    public string Menu1Header_french { get; set; }
    6 referencias
    public string Menu1Header_english { get; set; }
    6 referencias
    public string Menu2Header_spanish { get; set; }
    6 referencias
    public string Menu2Header_french { get; set; }
    6 referencias
    public string Menu2Header_english { get; set; }
```

  - TAGS: Json serialization



```
/// <summary> Clase serializable que define los tags de la app
[Serializable()]
[XmlRoot("Tags", Namespace = "", IsNullable = false)]
11 referencias
public class TagsData
{
    #region Properties

    public Dictionary<string, List<string>> tags;

    #endregion


}
```

- Info de tags, títulos y descripciones de Items de contenido (asociados a un pdf, video, html…)

```
/// <summary> Clase serializable que define los parámetros asociados a un item d
[Serializable()]
[XmlRoot("Item", Namespace = "", IsNullable = false)]
17 referencias
public class ItemData
{
    8 referencias
    public string TituloES { get; set; }
    8 referencias
    public string TituloIN { get; set; }
    8 referencias
    public string TituloFR { get; set; }
    1 referencia
    public string Tags { get; set; }
    0 referencias
    public string DescripcionES { get; set; }
    0 referencias
    public string DescripcionIN { get; set; }
    0 referencias
    public string DescripcionFR { get; set; }
}
```

3.7. PathManagement.cs

La gestión de rutas de la aplicación se realiza con la clase PathManagement.cs, con un sentido similar al que tiene el FolderManager de Cortex.

Con esta clase podemos ir controlando en qué carpeta estamos, que directorios o archivos contiene dicha carpeta, la carpeta contenedora, así como las rutas globales de la aplicación: dónde está el contenido, dónde está la carpeta de estilo, de configuración, archivos…