UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

Aplicación de escritorio para el escaneo, comparación y visualización de diferencias entre imágenes de textos

Proyecto Final de Carrera

Ingeniería Informática

Autor: María Cabezuelo Sepúlveda

Director: José Vicente Busquets Mataix

Codirector: Guillermo José Gimeno

Julio 2014

Resumen

Desarrollo de una aplicación de escritorio a medida para una imprenta que se encargara de autenticar a trabajadores de la empresa, facilitarles obtener de un pdf una imagen mapa de bits y escanear un documento con dicha imagen impresa, comparar ambas imágenes, encontrar las diferencias y presentarlas de manera fácil e intuitiva.

Development of a desktop application made specifically for a printing house and that will be able to identify the printing house workers, help them get a bitmap image from a PDF and to scan a document with the mentioned image printed, compare both images and find the differences and after finding them, make a report in PDF and show the user all the differences in a clear way.

Palabras clave: Pygsa, PrintControl, c#, .Net, XML, aplicación escritorio, escaneo documentos.

Agradecimientos

A mi familia por darme apoyo y amor todos estos años.
Al conversatorio y otros amigos por los buenos y malos tiempos, pero más por los buenos.
A Muse y Imagine Dragons por darme las horas de motivación necesarias para acabar.

Tabla de contenidos

Tabla de figuras

1 Introducción

En este capítulo presentamos la aplicación de escritorio "PrintControl" desarrollada en la empresa PYGSA para la imprenta catalana Ibérica Gráfica donde se quiere poder obtener una imagen de un PDF, escanear el impreso equivalente a susodicha imagen y comparar ambas para encontrar diferencias, todo de manera calibrable y sencilla para los operarios.

1.1 Motivación

Hoy en día cualquier producto debe tener un folleto con instrucciones o una etiqueta añadida al producto donde expliquen la composición del mismo, como usarlo, que hacer en caso de que se use de forma incorrecta, etc.

En el caso de los medicamentos esto es aún más importante puesto que dentro de la caja con el medicamento en cuestión debe estar el prospecto médico, donde a parte de lo dicho previamente, deben indicar en qué casos no tomarlo y otras contraindicaciones importantes para la salud. Que los prospectos médicos sean tan importantes implica que las erratas que se puedan encontrar en ellos no sean aceptables.

Se usan traductores humanos profesionales para traducirlo a varios idiomas y que haya el mínimo número de malentendidos posibles; se usa papel fino para que quepa la máxima cantidad de información en un solo papel facilitando así la lectura completa del prospecto; Pero cuando se imprimen documentos existe siempre la posibilidad del fallo de máquina. Una mota de polvo en los cabezales de una impresora, mover el papel antes de que se seque la tinta o un mal posicionamiento del papel pueden ocasionar que haya fallos de impresión, y muchas veces, especialmente cuando el idioma no es el propio de los operarios de la imprenta, no es detectable a simple vista por el ojo humano.

Cuando una imprenta comete uno de estos errores y no se da cuenta suele ocasionar pérdidas de dinero y tiempo importantes puesto que suelen hacer tiradas de impresión de cientos de miles documentos, todos con el mismo error. Esto hace que algunas imprentas intenten contratar software a para ayudarles a encontrar dichos errores.

Existen algoritmos de comparación de imágenes y de texto en el mercado que pueden adaptarse a las necesidades de estas imprentas, pero generalmente en las imprentas habrá rotación de operarios que deban poder usar este software fácilmente y poder visualizar de manera rápida donde están los errores y si son suficientemente importantes como para cancelar la impresión de los documentos. La complejidad de muchos de estos programas junto a la imposibilidad de añadir funcionalidades que puedan hacerles falta a los operarios o supervisores de la imprenta lleva cada vez más a la contratación de software a medida que se haga alrededor de estos algoritmos de comparación y que se acople al máximo a las necesidades de la imprenta.

En nuestro caso, la imprenta Ibérica Gráfica era la encargada de algunos prospectos médicos para la farmacéutica Novartis (entre otras) y en varios idiomas (entre ellos a destacar por la diferencia de grafías el español, turco, coreano y chino) y tras varias reuniones se acordó realizar un programa que funcionaría sobre un algoritmo de comparación que había sido ya contratado fuera de la

empresa para que ambos componentes pudieran funcionar con máxima eficiencia y facilitando su uso y entendimientos a los operarios, que luego nos pedirían otras funcionalidades si se daba el caso.

1.2 Objetivos

El propósito de este proyecto es la implementación de una aplicación de escritorio que ayudará a escanear documentos y obtener su imagen original de un documento pdf, comparar ambos y mostrar las diferencias al operario. Esta aplicación será comercializada íntegramente por PYGSA sistemas y aplicaciones bajo el nombre de printControl, y los cambios realizados sobre la misma dependerán de las necesidades del cliente, en este caso Ibérica Gráfica.

Los pasos del proyecto son los siguientes:

- Negociación con el cliente de las características que tendrá la aplicación.
    - Generación de la hoja de requerimientos.
- Obtención del software encargado de comparar las imágenes de texto.
- Obtención del scanner y aprendizaje de su uso (hardware & software)
- Obtención de materiales necesarios (impresiones de documentos, documentos en pdf, etc.)
- Creación de la versión alfa, que será mostrada al cliente.
- Modificación del proyecto con las aportaciones del cliente
- Creación de la versión beta y inicio fase de testeo
    - Testear la aplicación y obtener información
    - Entregar al cliente para que pueda testearla y dar información
    - Discusión con empresa y cliente sobre puntos críticos, mejoras importantes y mejoras para un futuro teniendo en cuenta la información obtenida.
- Creación de la versión final del producto.
- Adaptación del producto al cliente (introducir usuarios con sus respectivos grados de acceso a la aplicación, crear sistema de ayuda online mediante teamviewer, explicar cómo se usa la aplicación, etc.)

Las tareas de desarrollo, a grandes rasgos, son las siguientes:

- Creación del formulario principal donde se desarrollará el grueso de la aplicación.
    - PictureBoxes donde se cargaran las imágenes tanto escaneada como abierta desde la carpeta de imágenes.
    - Opciones de umbrales, tamaños y separación de píxeles necesarios para la función de comparación.
    - Sistema de mascaras.
    - Jerarquía de carpetas donde guardar y obtener imágenes, formatos, etc.
- Sistema de escaneo y paso de parámetros al escáner.
    - Velocidad de escaneo de documentos.
    - Archivo de configuración con las opciones que el escáner tomará.
- Creación del formulario de autenticación con uso de archivos externos en XML donde se guardarán los usuarios.
    - Autenticación de usuarios con nombre y contraseña.

    - Distintos tipos de usuario.
    - Opciones de añadir y quitar usuarios.
- Creación del sistema de lectura, escritura y guardado de formatos en XML.
    - Formatos guardan todas las opciones del sistema para posterior uso repetido.
- Creación del formulario de opciones
    - Logotipo para los PDFs.
    - Resolución del escáner.
    - Apertura y recorte del PDF para obtener imagen.
    - Log del sistema.
    - Multidocumento.
- Sistema de visualización de errores y creación de PDFs.
    - Comparación de errores uno a uno en ambos documentos.
    - Imagen general de los errores en el documento.

2. Especificación de requisitos

Cuando se nos encargó el proyecto, lo primero que fue necesario fue reunirnos con los encargados de la empresa para que llegáramos a redactar una lista con los requisitos funcionales y no funcionales del sistema, es decir, los comportamientos y requisitos del sistema.

2.1 Requisitos Funcionales

Los requisitos funcionales definen una función concreta del sistema y/o de uno de sus componentes. Para establecer un requisito funcional es necesario establecer las entradas que tiene dicha funcionalidad, comportamiento y salidas que genera.

2.1.1 Requisitos Funcionales de la aplicación

Código del Requisito	**Nombre del requisito**
LoginGeneral	Autenticación de usuario
Descripción o características	
Antes de poder usar la aplicación el usuario debe autenticarse en la aplicación con un nombre de usuario y contraseña.	

Código del Requisito	**Nombre del requisito**
LoginNiveles	Niveles de privilegios al autenticarse
Descripción o características	
Al autenticarse, los usuarios estarán organizados desde Operarios (con el menor nivel de privilegios), hasta Supervisores (con el máximo nivel de privilegios), pasando por Mantenimiento que será el nivel intermedio.	

Código del Requisito	**Nombre del requisito**
Privilegios	Privilegios de cada nivel
Descripción o características	
Operarios: Proceso completo de detección de errores, no pueden añadir usuarios, no pueden crear formatos y no pueden acceder al menú de opciones. Mantenimiento: Mantienen privilegios de operarios, pero también pueden crear formatos y acceder a opciones. No pueden añadir usuarios nuevos. Supervisores: Mantienen privilegios de Mantenimiento, pero también pueden añadir y borrar usuarios de la aplicación.	

Código del Requisito	**Nombre del requisito**
LoginAñadir	Añadir usuarios que puedan autenticarse
Descripción o características	
Los usuarios Supervisores pueden añadir otros usuarios y establecer su nivel y contraseña.	

Código del Requisito	**Nombre del requisito**
Velocidad	Establecimiento de velocidad de escaneo
Descripción o características	
La empresa escaneará todo tipo de papeles, incluidos algunos de gramaje muy fino. Es necesario poder establecer velocidad de escaneo baja para que no se arrugue el papel al escanear.	

Código del Requisito	**Nombre del requisito**
Previsualizar	Previsualización
Descripción o características	
Desde el menú principal se podrá previsualizar el escaneo y escoger la área a escanear en caso de usar escáneres planos.	

Código del Requisito	**Nombre del requisito**
Escanear	Escaneo
Descripción o características	
Desde el menú principal se podrá escanear documentos de manera fácil a una resolución de 400dpi.	

Código del Requisito	**Nombre del requisito**
Formato	Guardado y cargado de formatos
Descripción o características	

Todas las opciones seleccionadas en el programa para el escaneo podrán ser guardadas en un documento de formato y cargar cualquier documento de formato que deseemos.

Código del Requisito	**Nombre del requisito**
PDF	Guardar comparación en PDF
Descripción o características	
Debemos poder guardar en PDF los resultados de la comparación.	

Código del Requisito	**Nombre del requisito**
Recorte	Recortar PDF
Descripción o características	
Para la comparación será necesario poder escoger un pdf y de este sacar una imagen (de una página en concreto) que será la equivalente a la imagen a escanear.	

Código del Requisito	**Nombre del requisito**
Logo	Uso de logo personalizado
Descripción o características	
Los PDFs que se creen deben poder llevar un logo a elección del operario.	

Código del Requisito	**Nombre del requisito**
Multidocumento	Escaneo de imagen con varios documentos
Descripción o características	
Algunas de las hojas escaneadas contienen dentro varias copias del mismo prospecto en el PDF, debemos poder comparar tanto si la imagen escaneada es de un solo documento o de varios.	

Código del Requisito	**Nombre del requisito**
OCR	Detección de caracteres
Descripción o características	
Detección de regiones para poder leer pesos, tamaños o palabras concretas	

Código del Requisito	**Nombre del requisito**
Códigos	Detección de códigos
Descripción o características	
Detección de códigos e interpretación de los mismos, ya sean códigos de barras o códigos QR.	

Código del Requisito	**Nombre del requisito**
Máscara	Establecimiento de máscaras
Descripción o características	
Se deben poder establecer una serie de regiones, que llamaremos máscaras, que se indicarán al algoritmo de comparación y que este ignorará a la hora de ver las diferencias entre documentos.	

Código del Requisito	**Nombre del requisito**
OpcionesComparación	Establecer características de comparación.
Descripción o características	
El algoritmo de comparación que usamos tiene una serie de características configurables, que son el Tamaño del defecto mínimo para considerarse defecto, el Umbral de tolerancia y el Umbral de similitud que funcionan juntos para saber cómo de diferente debe ser lo encontrado respecto a lo esperado para considerarse error y el desplazamiento, es decir, el espacio donde buscar lo esperado en caso de no encontrarlo.	

Código del Requisito	Nombre del requisito
VisualizacionGrande	Visualización de errores en documento
Descripción o características	
Debemos poder ver los errores en la imagen escaneada de forma general para poder localizarlos fácilmente en la hoja impresa.	

Código del Requisito	Nombre del requisito
VisualizacionPequeña	Comparación de errores
Descripción o características	
Debemos poder ver los errores de forma muy localizada y clara puesto que en el documento las letras pueden ser muy pequeñas y difíciles de ver dónde está el error.	

2.1.2 Requisitos Funcionales de la interfaz de usuario

Código del Requisito	Nombre del requisito
MenuLogin	Menu de Login
Descripción o características	
El menú de login o autenticación tiene que estar por separado del menú principal, y es donde los usuarios indicarán que tipo de usuario son, su nombre y su contraseña. También será accesible desde el menú principal para poder añadir usuarios o cambiar de usuario en cualquier momento.	

Código del Requisito	Nombre del requisito
MenuPrincipal	Menú Principal
Descripción o características	
Menú que se abrirá inmediatamente después de autenticarse y que es donde se escanearán y compararán imágenes, y desde donde accederemos a los otros menús.	

Código del Requisito	**Nombre del requisito**
MenuPrevisualización	Menú de Previsualización de escaneado
Descripción o características	
Antes de escanear podemos darle a previsualizar para ver como se ve la imagen y para seleccionar la zona a escanear en caso de tener un escáner plano.	

Código del Requisito	**Nombre del requisito**
MenuErrores	Menú visualización de errores
Descripción o características	
Cuando hayamos hecho la comparación, debemos poder ver los errores que se han detectado (en caso de que se haya hecho) tanto en global como error por error y esto se hará en un menú independiente, para que puedan obtener el PDF sencillamente o ver los errores según si tienen tiempo o no	

Código del Requisito	**Nombre del requisito**
MenuOpciones	Menú de Opciones
Descripción o características	
Abrir en un menú independiente las distintas opciones que se han pedido para que el menú principal no esté recargado y para que solo los usuarios de nivel Mantenimiento y Supervisor puedan cambiar las opciones.	

Código del Requisito	**Nombre del requisito**
LoginAñadirSitio	Añadir usuarios de login
Descripción o características	
Debemos poder añadir usuarios sin tener que iniciar el programa principal y cuando estemos dentro del programa principal	

Código del Requisito	**Nombre del requisito**
PrincipalImagenes	Mostrar imagenes escaneo
Descripción o características	
Tanto la imagen obtenida del pdf como la imagen recién escaneada deben estar mostradas siempre para asegurarse que son la misma	

Código del Requisito	**Nombre del requisito**
PrincipalBotonesImagenes	Botones de funcionalidad de imagenes
Descripción o características	
Bajo cada imagen a mostrar debe haber un botón de Abrir imágenes que irá a la carpeta de imágenes de la aplicación, un botón de Escanear que empezara a escanear y un botón de Previsualizar, que abrirá el menú de previsualizar.	

Código del Requisito	**Nombre del requisito**
PrincipalOpcionesComparacion	Opciones de comparación en principal
Descripción o características	
Para que los operarios, que solo tienen acceso al menú principal, puedan configurar las opciones de comparación, estas deben de estar en el menú principal.	

Código del Requisito	**Nombre del requisito**
OpcionesCodigosOCRRecorte	Códigos, OCR y Recorte en opciones
Descripción o características	
Tanto la detección de códigos, zonas OCR y recorte de imagen de pdf estarán en opciones y se harán de forma gráfica sobre la imagen que se haya escaneado, o abierto en el caso del recorte de PFC	

2.2 Requisitos No Funcionales

1. La aplicación debe funcionar en un ordenador con 8GB de memoria RAM, sistema operativo Windows 7 de 64 bits, y con un procesador i5.

2. La aplicación debe estar compilada para una arquitectura de 32bits para poder funcionar con la función de comparación.

3. La fecha de entrega de la versión beta debe ser 5 meses después de la contratación del mismo.

4. El tiempo de un ciclo entero, es decir, escaneo, comparación y obtención del pdf con los resultados debe ser menor a 5 minutos por documento con una resolución de 400 dpi en el escáner.

5. El escáner deber poder escanear documentos que vayan desde 0.4 a 0.8 gramos de grosor.

6. El escáner debe ser un escáner de tambor y que pueda escanear documentos de tamaño A2.

7. Se autenticará al usuario mediante usuario y contraseña a indicar por el cliente.

8. La licencia para usar la aplicación estará en un USB que se entregará al cliente junto a otra copia de seguridad y será necesario para poder comparar documentos.

9. En todo momento se mantendrá control de quien hace las cosas para futura referencia en caso de necesitarlo por temas de seguridad.

3. Análisis, diseño y versión Beta

Una vez resueltos la mayoría de los requisitos que se nos habían indicado como importantes, y habíamos hecho la interfaz para todos, creamos la versión Beta del proyecto, a un mes de la fecha de entrega que se nos habla pedido

3.1 Cambio en los requisitos tras la versión Beta

3.1.1 Cambios en la aplicación

Para la versión beta algunos de los requisitos especificados aún no se habían implementado, estos requisitos habían sido estimados de baja importancia por el cliente, así que se planearon tener para la versión final.

Para cumplir los requisitos de tiempo, y tras probar la versión beta, algunos de estos requisitos fueron dejados para versiones futuras.

Por el contrario, tras probar algunas de las funcionalidades que nos pedían descubrieron que estas no eran suficientes para la finalidad que ellos querían conseguir, o que por el contrario eran demasiado.

A continuación detallamos que requisitos se añadieron, modificaron o dejaron para versiones futuras de la aplicación y por qué.

3.1.1.1 Requerimientos añadidos

Con una resolución de 400dpi, que era lo que originalmente se nos pedía, el cliente descubrió que, en algunos idiomas, el error era de menos de 1mm y que al escanear a 400dpi no se detectaba este error, se nos pidió que aumentáramos la resolución de escaneado a 600 y 720 dpi, lo cual nos trajo problemas de memoria como veremos en posteriores capítulos.

También se nos pidió que añadiéramos resoluciones menores en caso de querer hacer comparaciones rápidas y no muy exhaustivas de documentos.

Código del Requisito	**Nombre del requisito**
EscaneoVariasResoluciones	Elección de varias resoluciones al escanear.
Descripción o características	
Escaneo a varias resoluciones: 150, 300, 400, 600, 720 dpi	

En la versión beta, al crear el PDF resultado y recortar una imagen, automáticamente se le daba el nombre de la fecha junto al formato que se estaba usando en el caso del PDF (y un número de identificación) y en el caso de la imagen recortada, el nombre del pdf, junto a la página.

El cliente nos dijo que preferían ser ellos los que pusieran el nombre de los archivos, puesto que cada prospecto tenía un número de identificación, y preferían tener todos los archivos ordenados según éste número.

Código del Requisito	**Nombre del requisito**
DenominaciónDeArchivos	Denominación de archivos
Descripción o características	
Cuando se crea un PDF con resultados o se recorta una imagen es el cliente el que le da un nombre específico.	

3.1.1.2 Requerimientos dejados para futuras versiones

Los requisitos con código ñOCRò y ñCódigosò fueron implementados usando unas herramientas que acompañaban al algoritmo de comparación por si se querían usar. En Pygsa se decidió integrarlas en la aplicación por si a futuros clientes les era necesario, pero por falta de tiempo y porque en los prospectos no había necesidad de leer ninguna área con texto o decodificar ningún código, se decidió inhabilitar estas opciones, aunque aún estuvieran presentes, y testearlas y corregir posibles errores en un futuro.

3.1.2 Cambios en la interfaz

Una vez probada la interfaz el cliente nos pidió una serie de cambios que les facilitaran el uso o que añadiera una facilidad extra que no habíamos pensado.

3.1.2.1 Requerimientos añadidos o modificados

Algunos de los documentos que necesitaban escanear eran demasiado grandes o demasiado estrechos como para poderlos escanear con las letras en horizontal con comodidad usando el escáner de tambor. Se nos pidió que, una vez se escaneara el documento o se sacara la imagen del PDF y se mostrara en pantalla, pudieran ellos rotar las imágenes 90 grados en cualquier dirección, puesto que uno de los requisitos de la función de comparación es que ambas imágenes tengan el texto en el mismo sentido.

Código del Requisito	**Nombre del requisito**
PrincipalRotación	Rotar imágenes
Descripción o características	
Donde mostramos las imágenes en el menú principal deben haber botones de rotación de imágenes para que podamos ajustar ambas sin tener que reescanear, en caso de que la imagen escaneada y la del pdf tengan sentidos distintos (el algoritmo de comparación.	

En la visualización de errores, teníamos la imagen grande del documento donde marcábamos donde estaban los errores, y dos imágenes (la del documento original y la del escaneado) que mostraba exactamente que error había.

Los operarios trabajar con esto nos dijeron que no era suficiente, puesto que a veces el documento era tan grande y el error tan pequeño que no conseguían localizarlo en el documento original (era importante para ellos saber dónde estaban en la visión global del documento). Se nos pidió implementar en este menú de previsualización un zoom que mostrara donde estaba el error desde una vista más cercana y que si ellos movían el ratón por el documento también les mostrará una imagen más cercana de esa zona.

Código del Requisito	**Nombre del requisito**
VisualizacionZoom	Zoom para visualizar errores
Descripción o características	
En el menú de visualización de errores debemos añadir un zoom en forma de imagen dinámica al lateral para poder ver mejor los errores.	

La función de comparación admitía 4 valores para hacerla más o menos tolerante ante los fallos, estas se añadieron a la beta como se nos había pedido, tras probar estas opciones nos dijeron que su uso resultaba complicado y lioso, así que se nos pidió que añadiéramos tres ágrupacionesô de estas opciones según fueran de ôBajaô, ôMediaô y ôAltaô tolerancia ante los fallos, es decir, si elegían la opción de Baja los valores automáticamente se ajustaban para que las diferencias más mínimas fueran consideradas un error, mientras que por el contrario en Alta detectaríamos menos errores.

Código del Requisito	**Nombre del requisito**
ResumenCaracterísticas	Agrupación de características de comparación
Descripción o características	
Definición de tres opciones de comparación según lo tolerantes que sean con los fallos.	

En documentos con una gran cantidad de errores se hacía tedioso tener que visualizar cada uno de los errores, se nos pidió poder saltar de página y poder ir a un error en concreto a la hora de visualizar los errores.

Código del Requisito	**Nombre del requisito**
ConcretaciónErrores	Especificación de error a visualizar
Descripción o características	
En el menú de visualización de errores quieren poder escoger la página y el error concreto, a parte de la opción previa de ir pasando de error en error.	

Uno de los requisitos no funcionales era el de que para poder comparar documentos era necesario introducir una memoria USB que contenía la licencia. Se nos pidió que detectáramos que la licencia estuviera conectada y en caso de que no que avisáramos al usuario de la aplicación puesto que no siempre recordaban que debían conectarla.

Código del Requisito	**Nombre del requisito**
AvisoLicencia	Aviso de conexión de la licencia USB
Descripción o características	
Detectar si la licencia está conectada o no y avisar al usuario en caso negativo.	

3.2 Diagramas de actividad

Se ha utilizado el lenguaje de modelos UML (Unified Modeling Language) para facilitar la organización y estructuración de la aplicación mediante un diagrama de interfaz, el cual muestra los distintos menús y cómo se relacionan entre sí, y que ayuda a tener una visión global de la aplicación; Y de un diagrama de flujo, que nos sirve para ver los distintos pasos que seguiría un usuario desde que accede al menú principal hasta que sale de la aplicación.

3.2.1 Diagrama de interfaz

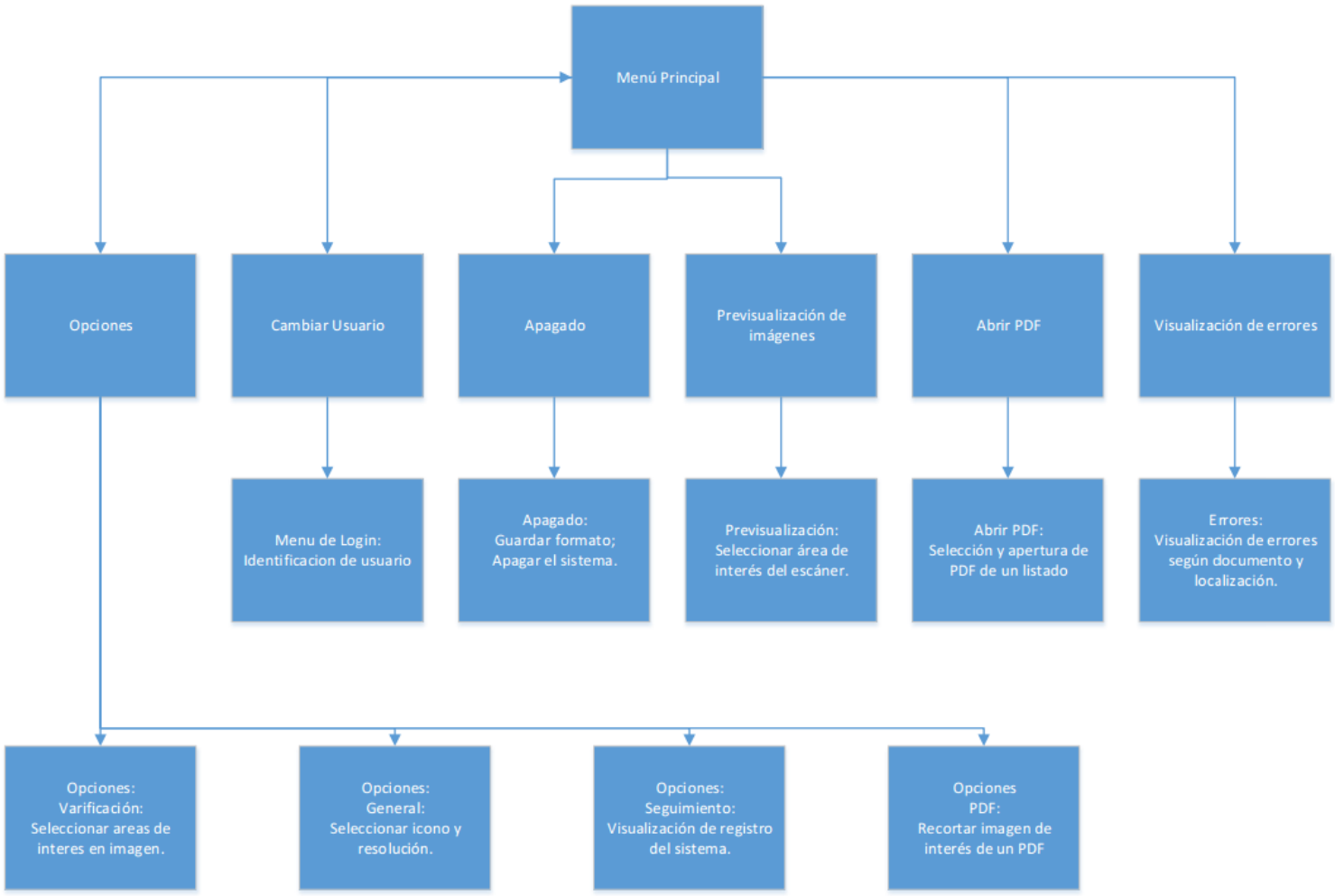

Figura 1: Diagrama de interfaz

En el diagrama de interfaz de la Figura 1 podemos apreciar que el grueso de la aplicación sucede en el menú principal, donde a parte de la funcionalidad principal también es la raíz de la mayoría de menús y submenús.

La mayoría de los menús solo tiene un nivel de complejidad, es decir, al pulsar el botón correspondiente abriremos un menú nuevo que tendrá una única funcionalidad y cuando se termina de usar se cierra.

La única excepción de esto es el menú de opciones, que como veremos en los siguientes apartados, tiene cuatro pestañas que agrupan a su vez una serie de funcionalidades en concreto.

Al tener agrupados así los menús el usuario no tiene que avanzar por complejas ramas de interfaz hasta llegar a la funcionalidad que desea y también facilita bloquear ramas entera de funcionalidades según los privilegios del usuario, como es el caso del operario, que no puede acceder al menú de opciones ni ninguno de sus submenús.

3.2.2 Diagrama de flujo

En la Figura 2 podemos ver el diagrama de flujo, que nos indica los pasos que sigue un usuario (con los permisos adecuados) desde que accede al menú principal después de autenticarse hasta que sale de la aplicación, también se muestra las decisiones que se toman y las consecuencias que ellas tienen.

Se han simplificado los las consecuencias de las tareas, de forma que si por ejemplo queremos añadir una máscara no describimos todo el proceso de la creación de máscaras sino que lo agrupamos bajo la acción de ñDefinir máscarasò

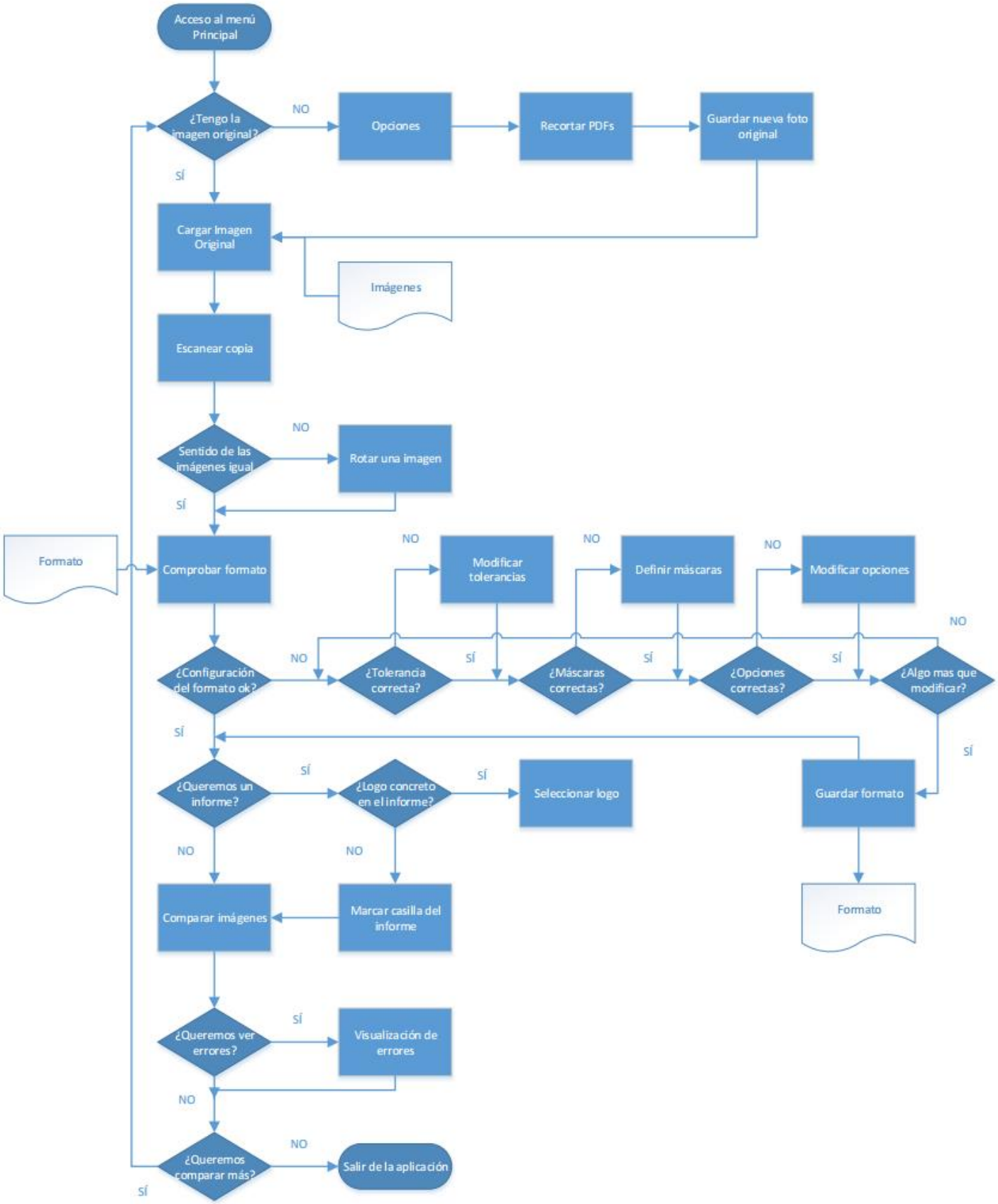

Figura 2: Diagrama de flujo

3.3 Estructura de documentos XML

XML es un Lenguaje de Etiquetado Extensible (eXtensible Markup Language) muy simple, pero estricto, que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML (HyperText Markup Language) pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información.

3.3.1 Autenticación

Para la autenticación de usuarios se han creado archivos XML que contendrán los usuarios admitidos por la aplicación, todos los usuarios tendrán un nombre, una contraseña y un nivel de privilegios. Cuando intenten autenticarse en la aplicación comprobaremos en este archivo si están en la lista y si su contraseña es correcta, y a continuación les daremos los permisos apropiados.

```
<printControl>
 <operario>
   <nombre>supervisor</nombre>
   <password>supervisor1234</password>
   <nivel>3</nivel>
 </operario>
 <operario>
   <nombre>pygsa</nombre>
   <password>pygsa1234</password>
   <nivel>3</nivel>
 </operario>
 <operario>
   <nombre>pepe</nombre>
   <password>0000</password>
   <nivel>2</nivel>
 </operario>
 <operario>
   <nombre>juan</nombre>
   <password>1234</password>
   <nivel>1</nivel>
 </operario>
 <operario>
   <nombre>juanita</nombre>
   <password>juanita</password>
   <nivel>1</nivel>
 </operario>
</printControl>
```

Entre `<printControl></printControl>` estarán contenidos todos los operarios admitidos en el sistema.

Cada usuario estará definido dentro de las etiquetas `<operario></operario>`, y tendrá cada uno una etiqueta `<nombre>` que será su identificador en el sistema, un `<password>` que será su contraseña y un `<nivel>`, que indicará su nivel de privilegios. Nivel 1 será el operario normal, nivel 2 serán los operarios de mantenimiento y por último nivel 3 indicara el este usuario es un supervisor.

Este archivo estará oculto y protegido contra lectura en los ordenadores de los clientes para que nadie pueda leer los usuarios y contraseñas, en un futuro será encriptado para que no haya ningún problema de seguridad.

3.3.2 Formato

La función de comparación requiere de una serie de datos que a continuación explicaremos y que servían para configurarla. Estos datos son completamente modificables por el usuario de la aplicación y pueden variar perfectamente de comparación a comparación. El cliente nos pidió que hubiera una forma de guardar y cargar dichos datos, puesto que quizás tuvieran que usar el mismo tipo de comparación muchas veces y no querían tener que reconfigurar la aplicación entera cada día. Otros de los datos guardados no están directamente relacionados con la comparación, pero si con opciones de la aplicación como el uso de un logotipo en concreto cuando se crea el PDF con los resultados.

Para esto decidimos crear unos ľformatosò que los usuarios de Mantenimiento y Supervisores pudieran crear, configurando la aplicación y guardando. Y que todos los usuarios pudieran cargar y usar en cualquier momento, creados en XML por su fácil integración y uso con .NET.

```
<formato>
 <Inspeccion>1</Inspeccion>
 <WSize>6</WSize>
 <Threshold>80</Threshold>
 <MaxOffset>12</MaxOffset>
<ImagenOriginal>C:\pygsavision\printControl\Imagenes\IMG.bmp</ImagenOriginal>
 <OCR>0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 </OCR>
 <Mascaras>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 -1 -1 -1 -1 </Mascaras>
 <checkOCR>False</checkOCR>
 <checkCodigo>False</checkCodigo>
 <checkMascara>False</checkMascara>
 <resolucion>2</resolucion>
 <rutaLogo>C:\pygsavision\printControl\Logos\logoPDF.png</rutaLogo>
 <blanca>1</blanca>
 <posQR>536 717 915 802</posQR>
 <posLote>409 813 540 880</posLote>
 <posPN>714 813 895 914</posPN>
 <posPK>433 890 560 966</posPK>
 <posFC>378 1007 569 1082</posFC>
 <posPVP>651 950 913 1105</posPVP>
 <posCodigo1>35 372 325 865</posCodigo1>
 <posCodigo2>54 800 322 1103</posCodigo2>
 <bQR>True</bQR>
 <bLote>True</bLote>
 <bPN>True</bPN>
```

```
 <bPK>True</bPK>
 <bFC>True</bFC>
 <bPVP>True</bPVP>
 <bCodigo1>True</bCodigo1>
 <bCodigo2>True</bCodigo2>
 <multidocumento>True</multidocumento>
 <Threshold1>80</Threshold1>
</formato>
```

Entre las etiquetas de `<formato></formato>` situaremos el resto de etiquetas a leer, que indicarán el principio y final del archivo de formato.

<Inspeccion> se traduce a una serie de valores según cuán de tolerante queremos que sea la inspección, y que junto a `<WSize>`, que es el tamaño mínimo del defecto para tenerlo en cuenta, `<Threshold>` y `<Threshold1>`, que son los umbrales de tolerancia y similitud respectivamente, es decir, a partir de cuánta diferencia entre lo esperado y lo encontrado se considera error, y `<MaxOffset>`, que es el desplazamiento máximo en el que es aceptable se haya movido la impresión, se le darán como parámetros a la función de comparación.

Consideramos que si están usando el mismo formato es porque están comparando varias imágenes escaneadas con una misma imagen original, por eso guardaremos la ruta a dicha imagen en `<ImagenOriginal>`.

La función de comparación requería que las coordenadas de cualquier área, ya sea un código o una máscara, vinieran dadas con cuatro números int separados, pero en el caso de las máscaras y OCR, como podían ser varias, con un vector de ints que acabará en cuatro números -1 cuando ya no hubieran más coordenadas, y con un máximo de 10 para las máscaras, y de tres para el OCR, indicadas con `<Mascaras>` y `<OCR>` respectivamente. Con `<checkOCR>` y `<checkMascara>` que son booleanos indicamos si debemos aplicar las coordenadas guardadas o no de manera automática, podremos cambiar esto desde los checkbox de la aplicación.

Las distintas resoluciones del proyecto se guardan en un enum y por tanto se guardan los valores 0, equivalente a 150 dpi, 1 a 300dpi, 2 a 400, 3 a 600 y 4 a 720 dpi. Estos valores se usan siempre así en el código y se guardan en `<resolucion>`.

Otra de las opciones cambiables es el logotipo que el cliente quiere que salga en los informes. Por defecto está el de la empresa Pygsa, pero esto puede ser cambiado y guardado en `<rutaLogo>` que será la ruta absoluta a la imagen de logo.

Por último en opciones también tenemos dos datos extra, el primero es la etiqueta <blanca>, es decir, si la imagen que estamos comparando tiene los márgenes blancos y por tanto nada que indique a partir de qué punto se acaba el documento y empieza el color de fondo del escáner, que es blanco. La otra opción a marcar es <multidocumento> que podemos marcarla o no según si lo que vamos a escanear es un documento con varias repeticiones de la misma imagen o si es solo una.

Aunque no se usa por ahora, también prevenimos para casos futuros y guardamos si necesitamos detectar códigos, con <checkCodigo>, y las coordenadas de los mismos <posCodigo1> y <posCodigo2>. También guardamos las áreas OCR del Lote, Peso, Precio Venta al público, etc. que incluimos en la aplicación pero que no han sido testeado ni se usan por ahora, como hemos dicho, <posQR>, <posLote>,<posPN>,<posPK,<posFC> y <posPVP>. Esto esta explicado mejor en el apartado 4.2.4 de esta memoria.

4. Implementación

4.1 Entorno de desarrollo

La aplicación en cuestión será desarrollada en .NET íntegramente, usando Visual Studio 10 y c# para la programación de la misma y XML para el sistema de autenticación de los usuarios, todo en Windows y para Windows.

Por restricciones de la librería de comparación de imágenes y de la computadora en la imprenta, la plataforma objetivo para el sistema será x86.

En cuanto al sistema de desarrollo de la aplicación, se usará como herramienta de gestión del proyecto de software Mercurial, concretamente TortoiseHg, tanto para compartir el proyecto con otros desarrolladores de la empresa que quieran mantenerse al día con el proyecto, como para plasmar los hitos del proyecto y las tareas necesarias para cada hito del mismo, así los clientes podrán ver en la web en qué punto está el desarrollo.

4.1.1 Forms en .NET

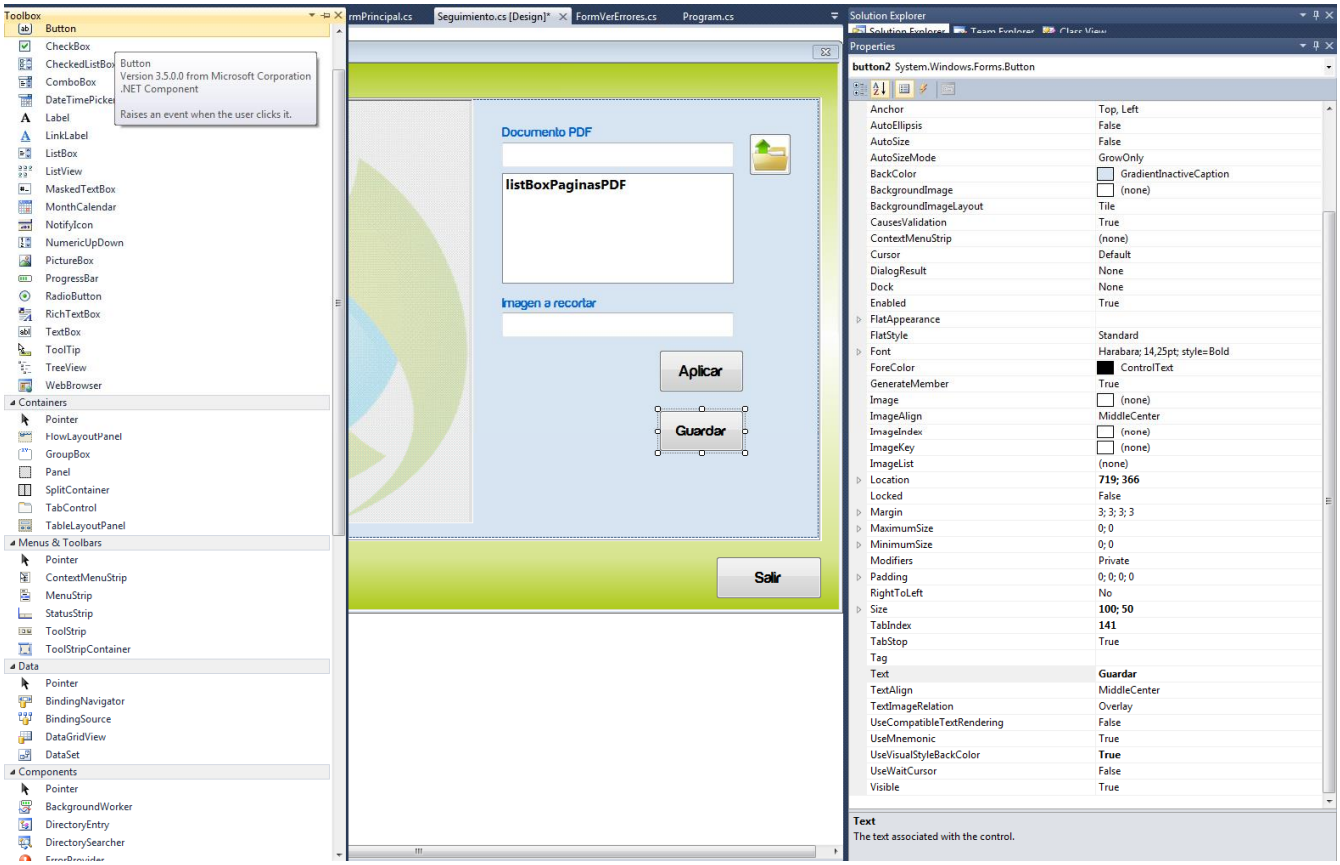

Figura 3: Windows Forms

.NET está perfectamente integrado con Visual Studio 10, por tanto el propio entorno de desarrollo nos proporciona las herramientas necesarias para programar usando los Windows Forms, que es la interfaz de programación de .NET y que envuelve el API de Windows.

Una Form es una ´hoja en blancoô donde podemos añadirle las herramientas que necesitemos del toolkit administrado, y que al añadirlas automáticamente se crearan las funciones y callbacks necesarios y que relacionan las diversas herramientas de un mismo form entre sí.

Este toolkit puede ser ampliado instalando otras librerías que estén integradas con .NET también y se añadirán en una nueva pestaña con el nombre de esta librería nueva las nuevas herramientas.

En la Figura 3 podemos ver un ejemplo de cómo se ve Windows Forms cuando estamos diseñando un menú. En el centro vemos un form al que hemos añadido color y otras herramientas, y estamos añadiendo el botón ´Guardarô Este botón ha sido seleccionado del toolbox de la izquierda que podemos ver en la figura 4

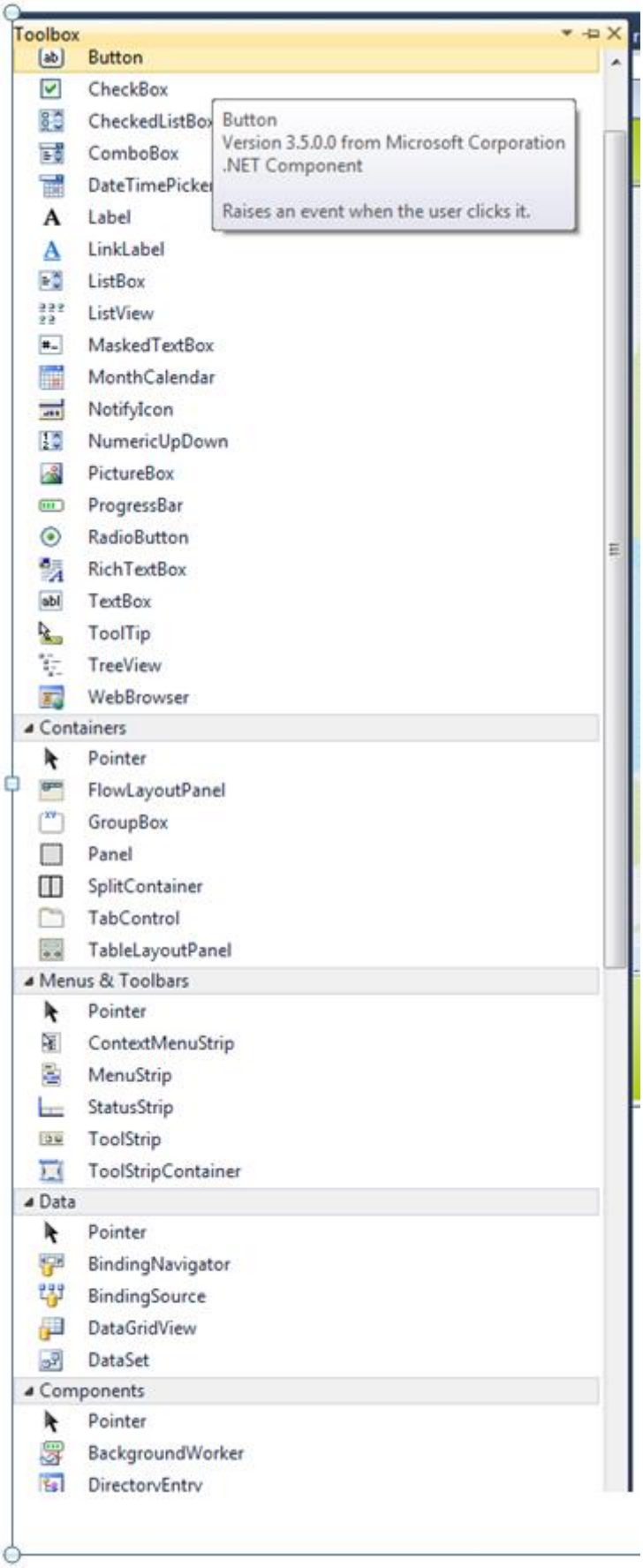

Figura 4: Toolbox

Podemos apreciar la diversidad de herramientas que tenemos a nuestra disposición y todas funcionan a grandes rasgos de la misma forma.

Al seleccionar y dejar en la posición que queramos la herramienta en cuestión, en este caso un botón, el código básico se añade al archivo ôDesignerô que lo relaciona con los otros elementos de la form.

En este caso el botón esta en el menú de Seguimiento, concretamente en la pestaña de PDF. En la figura 5 vemos cómo la pestaña de PDF añade todos los elementos (herramientas) que necesita, incluido nuestro botón guardar que es el ôbutton2ô Y como el button2 se define con un tamaño, posición, nombre, texto, etc. de forma automática al añadirlo al form.

```
//
// tabPagePDF
//
this.tabPagePDF.BackColor = System.Drawing.SystemColors.GradientInactiveCaption;
this.tabPagePDF.Controls.Add(this.button2);
this.tabPagePDF.Controls.Add(this.buttonAplicarRecorte);
this.tabPagePDF.Controls.Add(this.textBoxImagenARecortar);
this.tabPagePDF.Controls.Add(this.label6);
this.tabPagePDF.Controls.Add(this.buttonAbrirPDF);
this.tabPagePDF.Controls.Add(this.listBoxPaginasPDF);
this.tabPagePDF.Controls.Add(this.textBoxPDF);
this.tabPagePDF.Controls.Add(this.label7);
this.tabPagePDF.Controls.Add(this.pictureBoxImagenARecortar);
this.tabPagePDF.Location = new System.Drawing.Point(4, 29);
this.tabPagePDF.Name = "tabPagePDF";
this.tabPagePDF.Padding = new System.Windows.Forms.Padding(3);
this.tabPagePDF.Size = new System.Drawing.Size(907, 517);
this.tabPagePDF.TabIndex = 4;
this.tabPagePDF.Text = "PDF";
//
// button2
//
this.button2.Location = new System.Drawing.Point(719, 366);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(100, 50);
this.button2.TabIndex = 141;
this.button2.Text = "Guardar";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);
//
```

Figura 5: código del designer

En la última línea de la figura 5 vemos un callback. Cuando hacemos doble click sobre el botón en el form creamos automáticamente una función callback de click ôbutton2_Clickôvacía que se añadirá al botón de forma automática como podemos ver en esta última línea.

Este callback vacío se añade al código del form, no del designer, y por tanto podemos editarlo sin que pueda perderse al regenerarse el código o al actualizarse de forma automática como podría suceder con el código de la figura 5 que está en el Designer.

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        string nombreRecorte = Prompt.ShowDialog("Nombre Imagen", "Guardar Recorte");
        Bitmap b0 = new Bitmap(salida);
        string ruta = Util.mRutaImagenes + "\\" + nombreRecorte + ".bmp";
        b0.Save(ruta);
        b0.Dispose();
        salida = "";
    }
    catch { MessageBox.Show("No se ha guardado la imagen recortada"); }
}
```

Figura 6: Callback del botón

En la figura 6 podemos ver el código del callback del botón guardar que se llamará al hacer click sobre el botón. Sencillamente obtenemos el nombre que el usuario quiere darle a la imagen recortada mediante una caja de diálogo, guardamos la imagen con dicho nombre y borramos los elementos intermedios para aligerar la memoria.

Cabe mencionar que aparte de añadir el botón de forma automática, relacionarlo con los otros elementos y añadir callbacks, con Windows Forms también tenemos la opción de acceder a toda esta información y modificarla sin tener que abrir el código del Designer. Esto se hace desde la misma Form a la derecha, donde tenemos las propiedades del elemento seleccionado y que podemos editarlas directamente desde aquí, y éstas se actualizarán en el documento Designer mostrado en la figura 5.

En la figura 7 podemos ver las propiedades del botón, que a parte de las mostradas en el código también incluyen elementos de diseño como color, imágenes de fondo, tipografía, donde alinear la imagen y texto, etc.

SI se modifica algún parámetro que no está incluido en el código del designer, este se añadirá al código de forma automática con la elección del usuario.

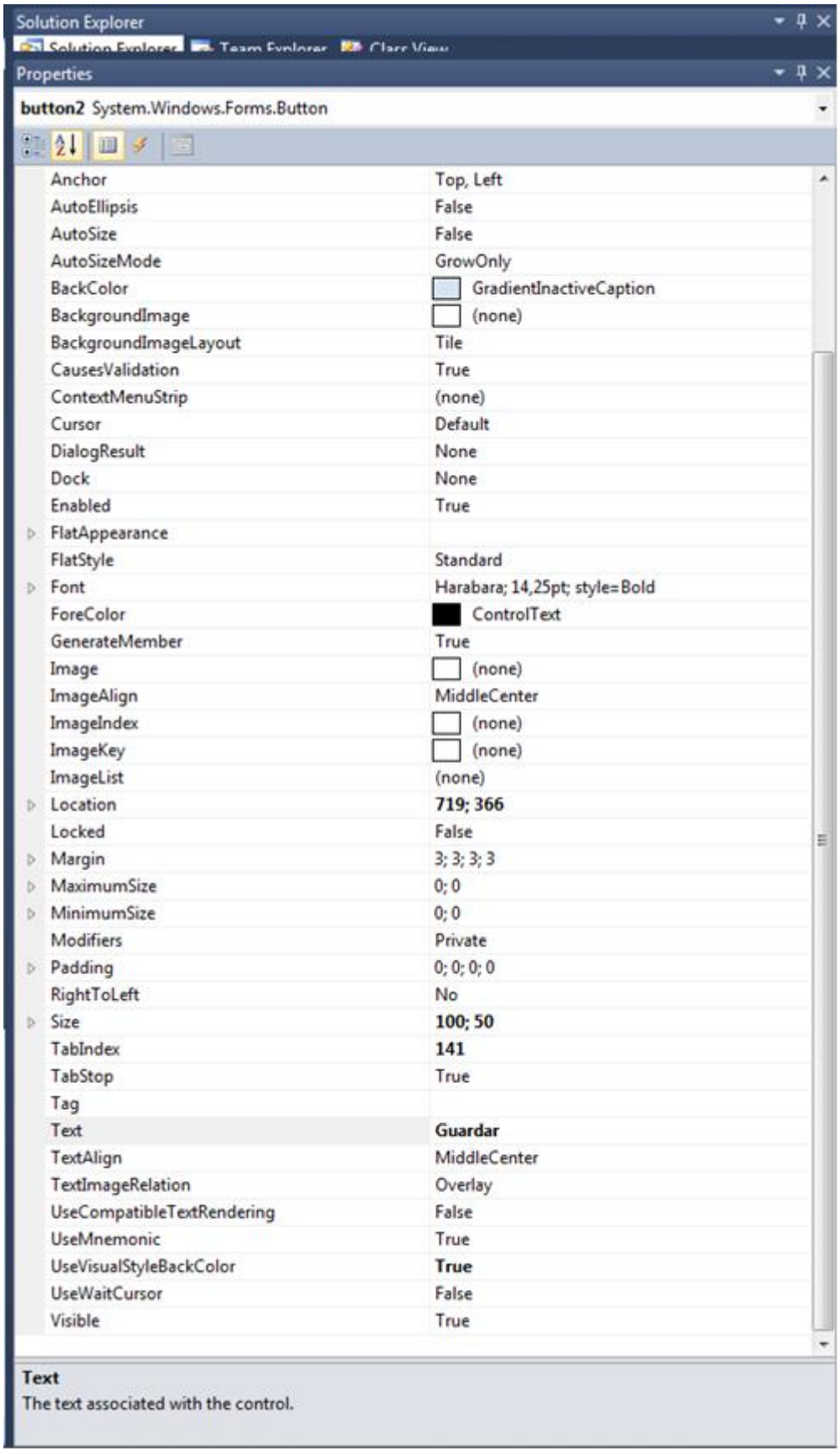

Figura 7: Propiedades de una herramienta

4.2 División de la aplicación en Windows Forms

4.2.1 Autenticación

Figura 8: Autenticación de Supervisor

Cuando iniciamos la aplicación lo primero que vemos es el menú de Login, separado de la aplicación principal como se especifica en ôMenuLoginõ y para que un usuario deba identificarse antes de poder acceder al menú principal, tal y como se indica en ôLoginGeneralô El usuario debe seleccionar entre los tres tipos de usuario que hay en ôLoginNivelesô Los permisos de los tres niveles están explicados en el requisito de ôPrivilegiosô Cuando el usuario inicia la aplicación y creamos el form de autenticación automáticamente cargamos el XML con los datos de los usuarios, cuando el usuario elija el tipo de usuario filtraremos el XML basándonos en esto, y luego al introducir nombre y contraseña se comprobaran que sean las tres cosas correctas y se le asignaran los permisos correspondientes;

en el caso del supervisor antes de pulsar Aceptar, si el usuario y contraseña son correctos, se desplegará la ventana de Login tal y como mostramos en la Figura 8 para poder añadir usuarios incluso antes de arrancar la aplicación, tal y como se nos pedía en el requisito ôLoginAñadirSitioô

Figura 9: Form de Login desde form Principal

Una vez en la aplicación principal, hay un botón en la esquina superior derecha que podemos usar para abrir otra vez el form de Login y así poder cambiar de usuario o en el caso del supervisor añadir nuevos usuarios desde dentro del programa, tal y como también se nos pide en ôLoginAñadirSitioô, en la Figura dos podemos ver el acceso al form de login desde el menú principal.

Para poder acceder a este XML y al de formato definimos una serie de simples get/set para poder obtener de forma rápida los parámetros que nos interesan, como Nivel, Password y Nombre de usuario por ejemplo.

```
public string Pass
{
    get { return pass; }
    set { pass= value; }
}
```

Estos campos los hemos usado para guardar la información del XML en nuestra clase FormatoLogin, de forma que cuando el usuario escribe un usuario y contraseña podemos comprobar si son correctos de forma fácil.

```
      if (mFichLogin.usuarioRegistrado(usuario))
      {
            //Después de consultar si el user está registrado,
             //se obtiene también el pass y el nivel


                  passBD = mFichLogin.Pass;
                  nivelBD = mFichLogin.Nivel;


                  if ((passBD == textBoxContrasenya.Text) && (nivelBD ==
      nivelCad))
                  {
                  //lógica de acceso al programa principal o de ampliación
                  de la interfaz.
                    }
      }
```

Aunque para este acceso tan rápido, primero hemos tenido que parsear la información. Para esto .NET pone a nuestra disposición la librería de `System.Xml` que nos facilita la tarea a la hora de abrir el archivo y sacar la información del mismo.

Por ejemplo, para comprobar que un usuario esta registrado hacemos lo siguiente:

```
public bool usuarioRegistrado(string user)
{
      bool encontrado = false;
      XmlDocument doc = new XmlDocument();
     doc.Load(mNombreFichero);

      //abrimos y cargamos el archivo


     XmlElement empleados = doc.DocumentElement;
     XmlNodeList empresa = Doc.GetElementsByTagName(Util.mNombreEmpresa);

      //en una misma empresa pueden haber varios grupos de usuario que
puedan abrir o no el programa. Obtenemos cuales son con mNombreEmpresa


      XmlNodeList listaEmpleados=    (XmlElement)
      empresa[0]).GetElementsByTagName("operario");


      //Obtenemos una lista de todos los Operarios
      //y para cada operario ahora debemos sacar la información


     foreach (XmlElement nodo in listaEmpleados)
     {
      if (nodo.FirstChild.InnerText == user)
           {
                  //Cuando encontramos el usuario que buscamos
                  //obtenemos su password y nivel de privilegios
            XmlNodeList nPass = Nodo.GetElementsByTagName("password");
```

```
                pass = nPass[0].InnerText;
            XmlNodeList nNivel = nodo.GetElementsByTagName("nivel");
                nivel = nNivel[0].InnerText;
              //y los guardamos en la clase como pass y nivel para
              //poder obtenerlos con el get/set


             encontrado = true;
             break;
            }
     }


     return encontrado;
}
```

Si abrimos el XML con modo escritura también podemos modificar el password asignándole al nodo en cuestión el nuevo password y guardando el XML.

A parte de obtener la información de los XML, esta librería también nos permite crear XMLs nuevos pudiendo darle el formato necesario sin necesidad de escribirlo a mano, esto nos sirve sobre todo para crear varios formatos a cargar con configuraciones distintas.

```
public void crearFicheroInicial()
{


       using (XmlTextWriter xmlTW = new XmlTextWriter(mNombreFichero,
Encoding.UTF8))
     {
             xmlTW.Formatting = Formatting.Indented;


            xmlTW.WriteStartDocument();


            xmlTW.WriteStartElement(Util.mNombreEmpresa);
            xmlTW.WriteEndElement();


            xmlTW.WriteEndDocument();


            xmlTW.Close();
       }
}
```

4.2.2 Principal

Figura 10: Form Principal

Una vez el usuario se autentica en la aplicación, independientemente del tipo de usuario tiene acceso al programa principal. El programa principal está hecho en su propio form, tal y como mostramos en la Figura 10, y es donde se desarrolla la principal acción de la aplicación, esto esta especificado en el requerimiento ôMenuPrincipalõ

En caso de ser un operario el botón de herramientas señalizado por dos herramientas cruzadas en la parte superior derecha estaría bloqueado, tal como sus privilegios especifican en el requerimiento de ôPrivilegiosõ, para el resto de caso estaría habilitado para poder acceder al menú de opciones, como se especifica en el requerimiento de ôMenuOpcionesõ

El botón de ayuda abrirá una ventana que avisara al usuario que debe contactar con PYGSA Sistemas y Aplicaciones en caso de tener problemas, informando del número de teléfono y de la opción de abrir TeamViewer para asistencia remota.

Y por último el de apagar que es para cerrar el programa, y si lo desea, cerrar windows, preguntando previamente si desea guardar los formatos. Esto no era un requerimiento del usuario, pero si un comportamiento estándar de las aplicaciones que vimos necesario incluir.

4.2.2.1 Opciones de comparación y escaneo

Bajo de los botones de la derecha tenemos el panel de las opciones de comparación, donde el usuario puede modificar manualmente estos valores para hacer más o menos tolerante la búsqueda de errores y que están explicados con más detalle tanto en el requisito de ó*OpcionesComparación*ô como en la explicación de donde se guardan dichos valores en la sección 3.3.2, que estos valores estuvieran en el menú principal y no en opciones también fue pedido de forma explícita para que los operarios, que no pueden acceder al menú de opciones, también pudieran modificar esto, según el requisito de ó*PrincipalOpcionesComparación*ô

Figura 11: Selección de tolerancias del Form Principal

Tras comprobar que la modificación de estos valores se hacía tediosa por la necesidad de modificar y testear si era suficiente, proceso que tardaba varios minutos, se nos pidió añadir un desplegable en la interfaz de forma que el usuario solo tuviera que escoger el tipo de tolerancia y los valores se modificarán solos, este requerimiento, ó*ResumenCaracteristicas*, es post-beta, y sencillamente requirió un cambio en la interfaz, puesto que los valores son los mismos, como podemos ver en la Figura 11.

Figura 12: Detalle de la interfaz de principal

Bajo de estos valores tenemos las máscaras, como vemos en la Figura 12, que ahora mismo están en versión beta pero puesto que nos han dicho desde la imprenta del cliente que no las van a usar porque quieren ver documentos enteros, no han sido modificadas.

Cuando indicamos que vamos a añadir una máscara nueva mediante el checkbox de añadir, lo que hacemos en habilitar la captura de posiciones del ratón cuando hacemos click sobre la imagen de escaneado.

```
private void pictureBoxOriginal_MouseDown(object sender, MouseEventArgs e)
{
      if (checkBoxAnyadirMascara.Checked && (indMascara < numMascaras)
           {
                     rectMascara.Width = 0;
                     rectMascara.Height = 0;
                     rectMascara.X = e.X;
                     rectMascara.Y = e.Y;
            }
   }
```

Cuando hacemos click sobre la imagen Original, si tenemos activadas las máscaras, capturaremos la posición mediante el evento de ratón denominado como e que indica la posición.

```
private void pictureBoxOriginal_MouseMove(object sender, MouseEventArgs e)
{
           if (checkBoxAnyadirMascara.Checked && indMascara < numMascaras)
           {
                  if (e.Button == MouseButtons.Left)
                  {
                     Cursor.Current = Cursors.Cross;
                     ControlPaint.DrawReversibleFrame(RectangleToScreen(rect
      Mascara), Color.Black, FrameStyle.Dashed);


                      rectMascara.Width = e.X - rectMascara.X;
                      rectMascara.Height = e.Y - rectMascara.Y;


                      coordX1 = rectMascara.X;
                      coordY1 = rectMascara.Y;
                      coordX2 = e.X;
                      coordY2 = e.Y;
                  }
            }
  }


private void pictureBoxOriginal_MouseUp(object sender, MouseEventArgs e)
{
      if (checkBoxAnyadirMascara.Checked && indMascara < numMascaras)
      {
            anchoImagenOriginal = pictureBoxOriginal.Image.Width;
            altoImagenOriginal = pictureBoxOriginal.Image.Height;


            xScale = ((float)pictureBoxOriginal.Width) /
      anchoImagenOriginal;
            yScale = ((float)pictureBoxOriginal.Height) /
      altoImagenOriginal;


            posMascara[indMascara].xl = (int)((coordX1 / xScale)/
      scaleFactor);
            posMascara[indMascara].yt = (int)((coordY1 / yScale)/
      scaleFactor);
            posMascara[indMascara].xr = (int)((coordX2 / xScale)/
      scaleFactor);
            posMascara[indMascara].yb = (int)((coordY2 / yScale)/
      scaleFactor);


            mFichConfig.coordenadasMascara[(indMascara * 4)] =
      (int)(posMascara[indMascara].xl / scaleFactor);
            mFichConfig.coordenadasMascara[(indMascara * 4) + 1] =
      (int)(posMascara[indMascara].yt / scaleFactor);
            mFichConfig.coordenadasMascara[(indMascara * 4) + 2] =
      (int)(posMascara[indMascara].xr / scaleFactor);
            mFichConfig.coordenadasMascara[(indMascara * 4) + 3] =
      (int)(posMascara[indMascara].yb / scaleFactor);


            indMascara++;
```

```
                textBoxNumMascaras.Text = indMascara.ToString();
        }
}
```

Si no soltamos el ratón, cuando arrastramos dibujamos una caja alrededor de la zona y cuando finalizamos transformamos esa posición de inicio y de fin según como hayamos reducido la imagen, como veremos a continuación. Estas posiciones son relativas a las imágenes de tamaño completo que son las que usa la función de comparación, se guardan y borran directamente del XML de formato como explicamos en el punto 3.3.2. La posición X e Y de la esquina superior izquierda y de la inferior derecha formaran una ócajaôque al pasarle a la función de comparación marcará como óignorableôy por tanto no buscará errores dentro.

Podemos crear hasta 10 zonas a enmascarar, que es el límite establecido por la función de comparación mientras tengamos marcado el checkbox de añadir máscaras.

Cuando pulsamos el botón de borrar todas, como el nombre indica, borramos todas las máscaras. Finalmente con el checkbox de Máscara indicamos si vamos a usar las máscaras o no, puesto que se pueden tener guardadas en el xml de formato y las cargaremos cada vez que carguemos el formato. Con esto cumplimos el requisito de ómascaraô El sistema actual como hemos dicho está en fase beta puesto que solo poder borrar y activar todas las máscaras a la vez no es cómodo, pero como decimos en la sección de trabajo futuro se mejorará en siguientes versiones.

Continuando por la derecha, después de las máscaras nos encontramos con las opciones de la velocidad del escáner, que como explicamos en el requisito de óVelocidadô es necesario poder regular la velocidad del rodillo del escáner. El cliente va a escanear desde documentos en tamaño A4 hasta documentos de tamaño A2, y con varios gramajes. En el caso de los documentos muy finos, la presión de las ruedas que recogen el papel y lo mueven para que se escanee es demasiado si va demasiado deprisa, por eso hemos añadido 3 velocidades, que con la función de `_SetScannerParameter("SCN_SPEED".ToCharArray(), "45".ToCharArray())` de la librería del escáner podemos cambiar. La propiedad del escáner a modificar, en este caso SCN_SPEED, indica que lo que queremos modificar es la velocidad y el 45 cuál es la velocidad en una escala del 1 al 100. Los mismos valores han sido establecidos por prueba y error y por preferencia del cliente.

4.2.2.2 Generación de PDF

Una de las funcionalidades más importantes para el cliente es la generación de informes en formato PDF con los datos obtenidos como se nos pedía en el requerimiento óPDFô Cuando pulsamos el botón de Comparar, si tenemos cargadas dos imágenes en los pictureBox centrales, comprobaremos si el checkbox de ñGenerar PDF resultadoò ha sido marcado, y en caso afirmativo, creamos un BackgroundWorker que es como llama .Net a los hilos que se ejecutarán de forma paralela a la función principal de la aplicación y que pueden

ser cancelados, comprobar su estado y ver si han finalizado. Al iniciar la comparación bloqueamos la interfaz gráfica de la aplicación y desbloqueamos cuando termina. Cuando el hilo del PDF termina pedirá al usuario por medio de una ventana nueva el nombre que quiere darle al informe, tal y como se nos pidió en el requerimiento pos-beta 0DenominacionDeArchivosô y se guarda en la carpeta de Resultados que creamos al instalar la aplicación.

```
BackgroundWorker bwPDF = new BackgroundWorker();
bwPDF.WorkerSupportsCancellation = true;
bwPDF.WorkerReportsProgress = true;
bwPDF.DoWork += new DoWorkEventHandler(bwPDF_DoWork);
bwPDF.ProgressChanged += new
            ProgressChangedEventHandler(bwPDF_ProgressChanged);
bwPDF.RunWorkerCompleted += new
      RunWorkerCompletedEventHandler(bwPDF_RunWorkerCompleted);
```

Necesitamos poder cancelar el PDF en caso de que, por ejemplo, las imágenes sean incorrectas o no estén ambas en el mismo sentido de escritura. La función de comparación no podrá realizar su trabajo y por tanto nosotros debemos cancelar la creación del informe.

Llamaremos al DoWork, que es donde se realiza la creación del informe, y que irá recogiendo los datos que devuelva la función de comparación y añadiéndolos al informe, en nuestro caso lo que recogerá serán los datos de los errores y los añadirá a una tabla donde pondrá el número de error, el trocito de imagen correspondiente al error en la imagen original sacada de el pdf y el trocito de error de la imagen escaneada.

Al inicio del documento pondremos la fecha y hora, que formato era el que usábamos y quien era el operario que realizó este informe, un pequeño resumen de cuantos errores e imágenes en el documento escaneado se han encontrado, y por último el logotipo especificado en opciones, tal y como se nos pedía en los requerimientos de 0PDFô y de 0Logoô

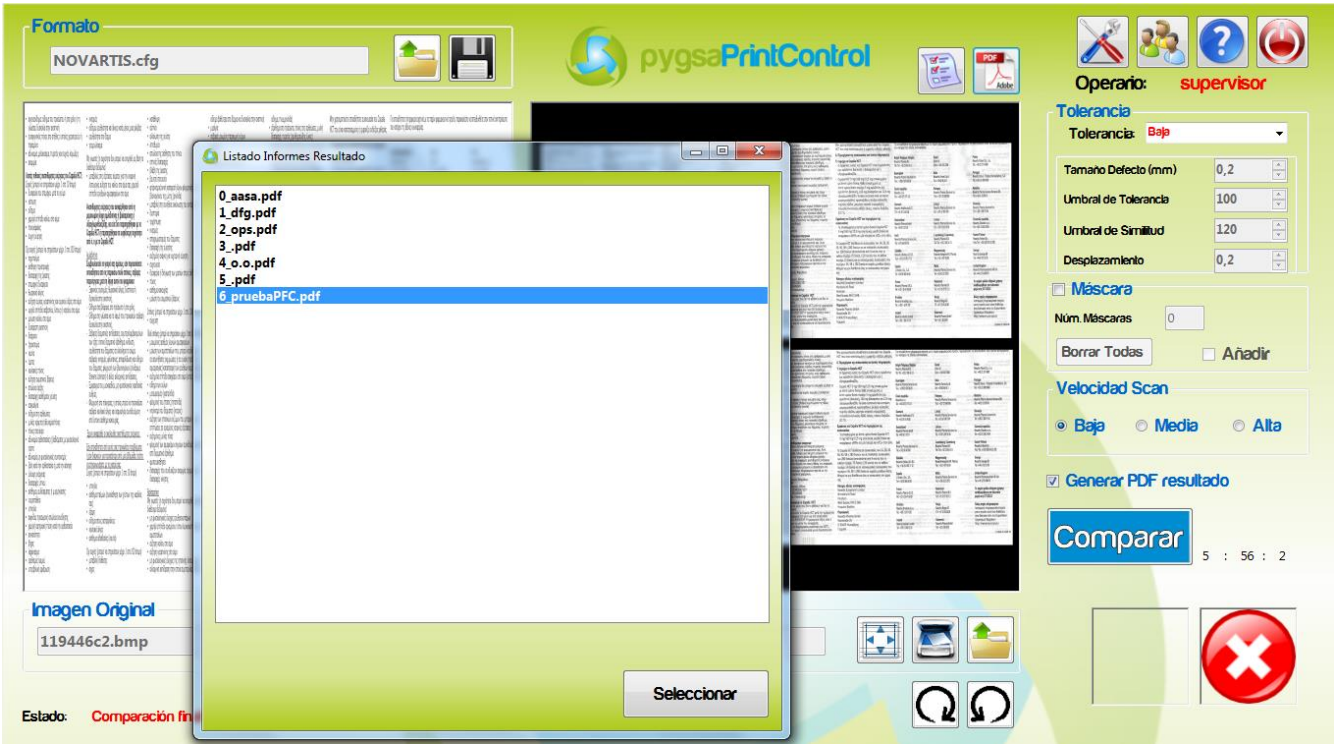

Figura 13: Control de PDFs

Una vez introducido el nombre del informe explicado antes se guardará en la carpeta de Resultados, y podremos abrirlo desde la aplicación cuando pulsamos el botón con la imagen del PDF de Adobe arriba de la segunda imagen, que abrirá la ventana que podemos ver en la Figura 13 y podremos seleccionar el PDF a abrir y abrirlo.

4.2.2.3 Formato

Figura 14: Formato

Situándonos ahora en la esquina superior izquierda tenemos la sección de formato. Como explicamos en el requerimiento ôFormatoõ es necesario que cuando el cliente quiera escanear el mismo tipo de documento, o al menos documentos con características muy similares a lo largo de varios días, pueda hacerlo de forma fácil sin tener que cada día vez que cargue el mismo documento poner a mano cada una de las opciones, para esto creamos un documento XML con los datos que le pasaremos tanto a la función de comparación como que usaremos nosotros en la aplicación y se guarda.

El form principal es que el mantiene el registro local de los datos del formato, al crear el form la primera vez, cargamos una única vez el XML y obtenemos los datos, ningún otro menú abrirá el archivo de configuración. Si pulsamos el botón de cargar un XML de formato nuevo, borraremos este registro del form principal y cargaremos los nuevos datos.

Cuando abrimos cualquier menú, pasamos al nuevo form que estemos creando los datos pertinentes sacados del fichero de configuración que pueda necesitar, luego mediante eventos y delegados guardamos estos valores.

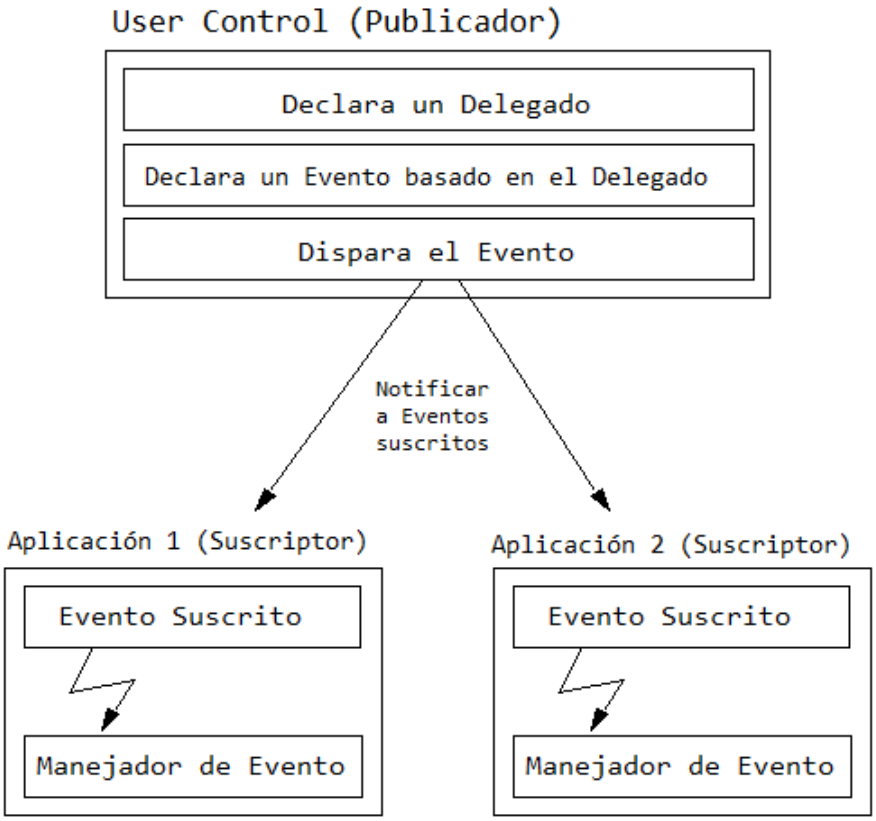

Figura 15: Sistema Evento-Delegado.

En nuestro caso, los forms que pueden modificar los datos del archivo de configuración, y que no son el form principal son los que declaran los delegados y los eventos de modificación de datos, y en el form principal, que es quien crea estos nuevos forms, después de crearlos se suscribe a estos eventos.

```
FormSettings fs = new FormSettings(mRutaLogo, indiceRes, etiquetaBlanca,
multiDocumento, CBconOCR, rutaOriginal, posCodigos, bCodigosActivados);


fs.eventoResolucion += new FormSettings.delegadoResolucion(fs_evento);
fs.eventoRutaLogo += new FormSettings.delegadoRutaLogo(fs_eventoRutaLogo);
fs.eventoBlanca += new FormSettings.delegadoBlanca(fs_eventoBlanca);
```

Por ejemplo, en el ejemplo de arriba, el form principal al pulsar el botón de opciones crear el nuevo menú y les pasa los datos que necesitará. De este form puede obtener cuales son los eventos que existen, en este caso la opción de cambiar la resolución, la ruta al logo que queremos usar para el pdf y si la etiqueta es blanca o no. En caso de que esto sucediera, mediante los delegados creados en opciones, le decimos cual va a ser el manejador o handler de este evento, y por ejemplo en el caso de la ruta del logo

```
void fs_eventoRutaLogo(string ruta)
{
     mRutaLogo = ruta;
     mFichConfig.rutaLogo = ruta;
}
```

Sencillamente guardamos en local la ruta para poder acceder a ella más fácilmente o volverla a pasar en caso de volver a abrir el form de opciones, y luego lo guardamos en el fichero de configuración.

Con este sistema evitamos tener que abrir y cerrar en cada menú el archivo de configuración, con los posibles problemas de sobreescritura de datos y de la inestabilidad que esto produciría. También nos ahorramos tener que en cada menú volver a crear el principal para poder acceder a las declaraciones locales y modificarlas.

Cuando pulsamos el botón de guardar o salimos del sistema se nos da la opción de guardar el formato, siempre desde el form principal que cogerá los datos locales que ya ha introducido generalmente al fichero de configuración y lo guardará con el nombre que le digamos.

Son estas variables locales las que le pasamos a la función de comparación cuando pulsamos el botón de comparar.

4.2.2.4 Imágenes a comparar

Figura 16: PictureBox de la imagen Original

Debajo de sistema de formato tenemos las dos PictureBox, que son una de las herramientas que nos proporciona .Net para el uso de imágenes, y aunque se podría cambiar, la primera imagen siempre será la original obtenida del PDF mientras que la segunda será la que obtenemos al escanear, situadas aquí para rápido acceso de todos los usuarios tal y como se nos pide en el requisito ôPrincipalImagenesô

Estas PictureBox estarán vacías cuando accedemos a la aplicación, debajo de ellas están los botones de previsualización, escaneo y cargado y así también

cumplimos con el requisito de "PrincipalBotonesImagenes", que requería que esto pudiera hacerse por todos los usuarios y que fuera rápido poder hacer esto puesto que es la funcionalidad principal de la aplicación.

Cuando pulsamos previsualización abrimos el form de previsualización explicado en el apartado 4.2.3, cuando escaneamos, si el escáner está encendido, bloquearemos la interfaz y esperaremos a que la función de escaneado acabe, entonces guardaremos en disco la imagen y esta será la ruta que le daremos a la función de comparación. Los PictureBox por cómo funcionan internamente tienen unos tiempo de cargado de imágenes un poco elevados, intentar mostrar las imágenes completas fue un error puesto que muchas veces la aplicación se quedaba sin memoria antes de haber podido cargar las imágenes, esto lo solucionamos reduciendo el tamaño de la imagen puesto que al ser tan grandes y los PicturBox en comparación tan pequeños no se puede apreciar la calidad de la imagen. Según el tipo de resolución las imágenes serán más o menos grandes, que usaremos nosotros también para definir el factor de reducción. Esto lo explicamos más a fondo en el apartado 5.2 de este documento.

Bajo de los botones que hemos comentado tenemos dos botones más, que como indican sus flechas, sirven para rotar las imágenes en una dirección u otra. Necesitamos los botones, tal y como explicamos en el requisito "PrincipalRotacion" porque la función de comparación necesita que el texto de ambas imágenes este en el mismo sentido, pero puede suceder que por el tamaño del papel u otras circunstancias haya sido escaneada una imagen en otro sentido.

Estos botones cargaran la imagen grande por necesidad puesto que es imperativo que la función de comparación reciba la ruta de la imagen en el sentido correcto, no que nosotros enseñemos al usuario la imagen girada, para eso usando los Image de .NET cargamos y rotamos con dos simples funciones.

```
System.Drawing.Image img =System.Drawing.Image.FromFile(ruta);
img.RotateFlip(RotateFlipType.Rotate90FlipNone);
img.Save(ruta, System.Drawing.Imaging.ImageFormat.Bmp);
```

Y ya tenemos la imagen cargada, rotada y guardada. Luego volvemos a pasar esta nueva imagen por la función de reducción y es la que mostraremos por pantalla con los PictureBox.

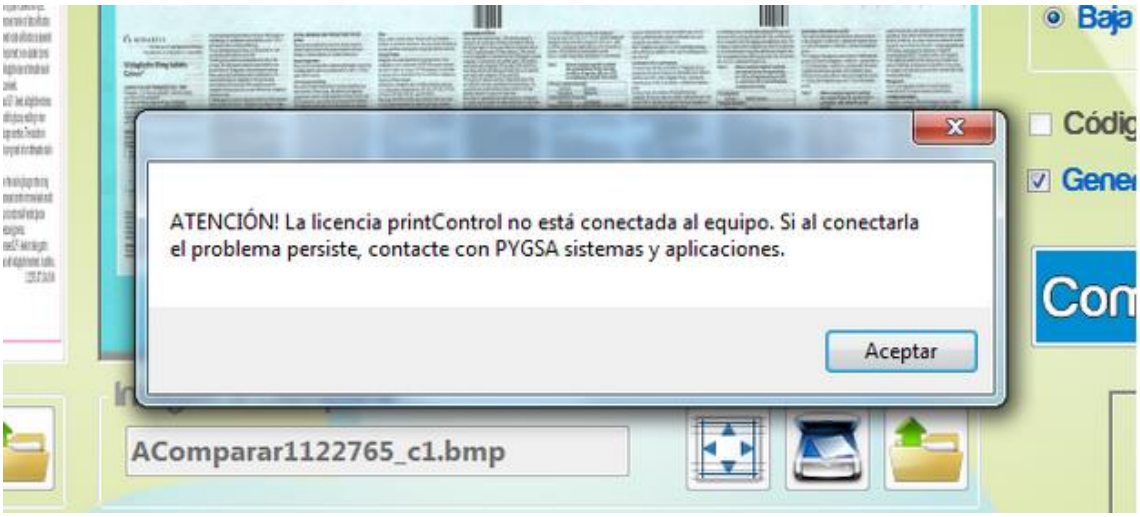

Figura 17: Control de licencia

Como último detalle, podemos abrir el menú principal y ver los PDFs, abrir opciones y otros detalles varios, pero es necesario, por seguridad como se nos pidió en ó AvisoLicenciaô que para comparar y escanear tengamos conectado el USB con la licencia y que si no se detecta avisemos al usuario como vemos en la Figura 17.

4.2.3 Previsualización

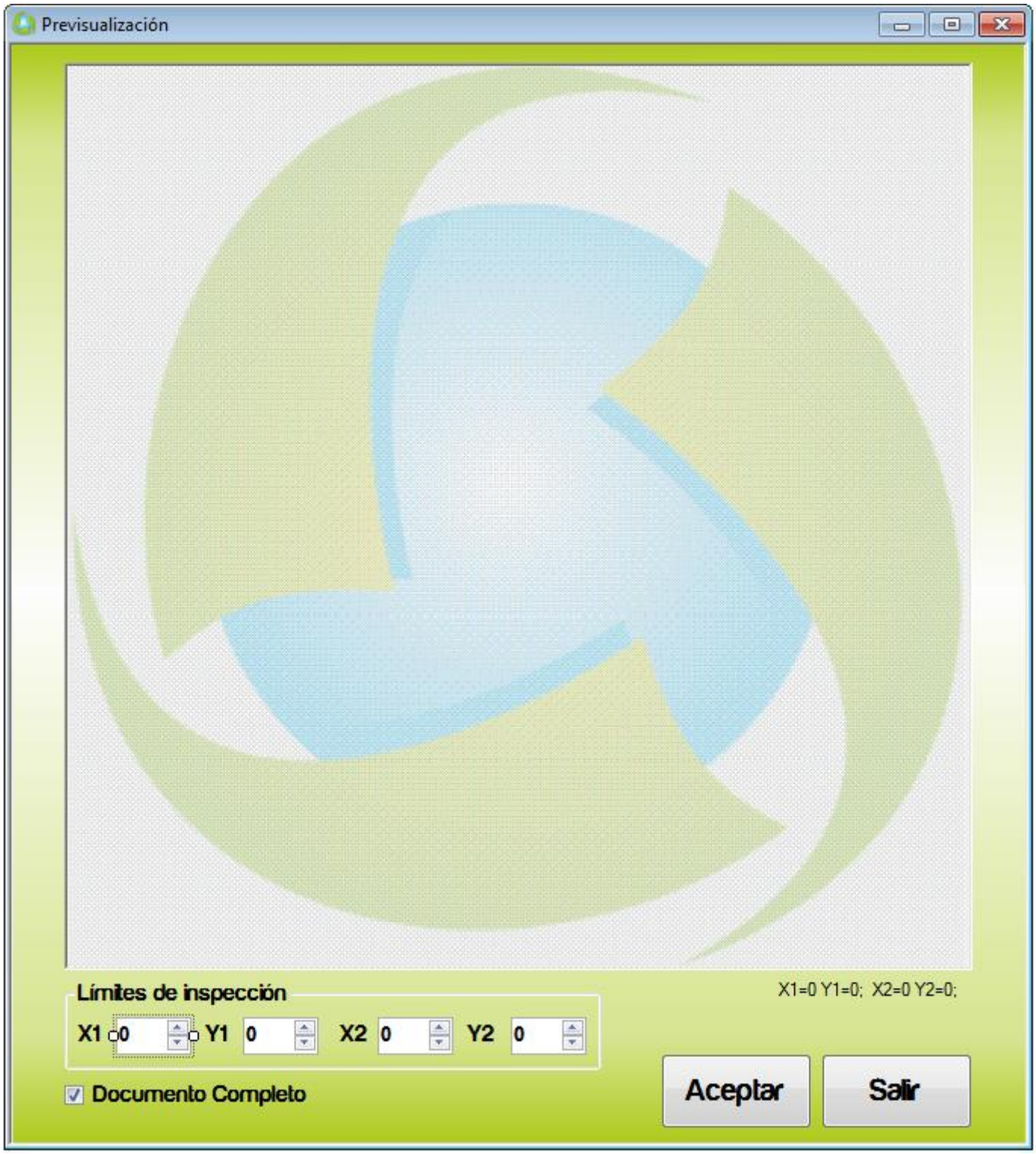

Figura 18: Menú de previsualización.

El cliente, en una primera instancia, quería escanear documentos pequeños o el PDF con la imagen original que no tuvieran en formato digital, sino ya impreso, con un escáner de plato y nos pidió la opción de que no se escaneara toda la zona

puesto que solo quería la región central del texto, ignorando márgenes que solían contener número, título y otras informaciones que ellos no iban a imprimir.

Para esto sencillamente creamos el ó1MenuPrevisualizarô donde escaneamos el documento a la resolución especificada en el menú de opciones y lo mostramos en una PictureBox donde el usuario puede dibujar un rectángulo con la zona que quiere escanear, o la introduce manualmente modificando los límites de la inspección que podemos ver en la Figura 18, y una vez aceptados se los pasamos al escáner de plato

```
_SetScannerParameter("SCN_ROI_LEFT".ToCharArray(), "0".ToCharArray());
_SetScannerParameter("SCN_ROI_RIGHT".ToCharArray(),
      anchoEscaner.ToString().ToCharArray());
_SetScannerParameter("SCN_ROI_TOP".ToCharArray(), "0".ToCharArray());
_SetScannerParameter("SCN_ROI_BOTTOM".ToCharArray(),
      altoEscaner.ToString().ToCharArray());
```

Que en este ejemplo tendría el origen en la esquina superior izquierda y el ancho y alto estarían definidos por anchoEscaner y por altoEscaner.

Esto solo funciona en el escáner de plato, puesto que el escáner de rollo una vez empieza a escanear un documento no para hasta que se acabe el documento y no puede escanear solo una zona, si estuviera configurada una zona sencillamente la ignoraría.

Para facilitar el escaneado completo hemos añadido la un checkBox ñDocumento Completoòque borra los límites explicados antes.

Aunque esta funcionalidad quedó desfasada cuando empezaron a usar el escáner de rollo se nos pidió mantenerla por si algún día necesitaban hacer un escaneo más rápido y pequeño.

4.2.4 Opciones

4.2.4.1 Verificación

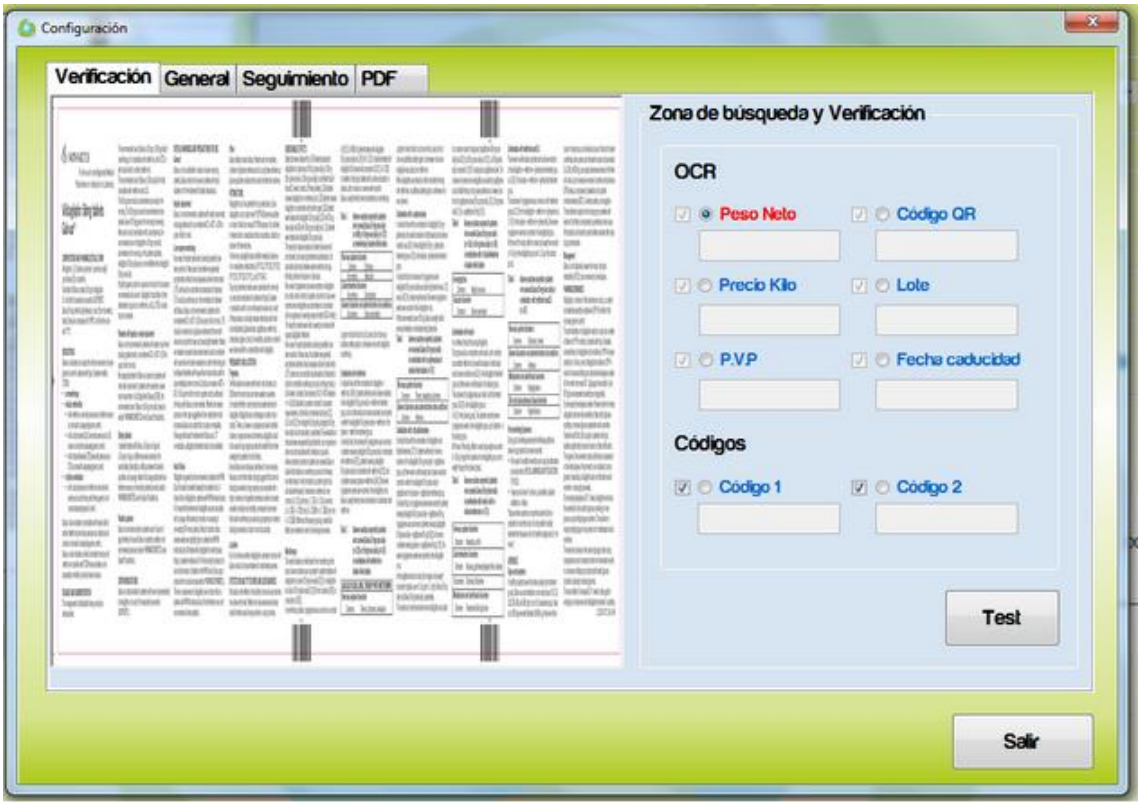

Figura 19: Opciones; Verificación

Cuando pulsamos el botón de opciones se abre un menú nuevo con varias pestañas. Este es el ôMenuOpcionesô que se nos pidió, con el control de privilegios de usuario dicho en los requisitos.

La primera pestaña es la de verificación, que contendría a la izquierda una PictureBox de la imagen escaneada también reducida por el método comentado anteriormente; si no se ha escaneado nada se avisará al usuario y se pondrá la imagen por defecto, que es la última que se escaneo. En esta pestaña se completarían los requisitos de ôOCRô y ôCódigosô que están implementados pero no testeados, y que se han dejado para futuras versiones. Para seleccionar las regiones de interés se usa el mismo método que en las máscaras, es decir, tomando como inicio el primer click del ratón, dibujando un rectángulo con el fin marcado por la posición del ratón al soltar, y transformando estas coordenadas según se haya reducido la imagen.

4.2.4.2 General

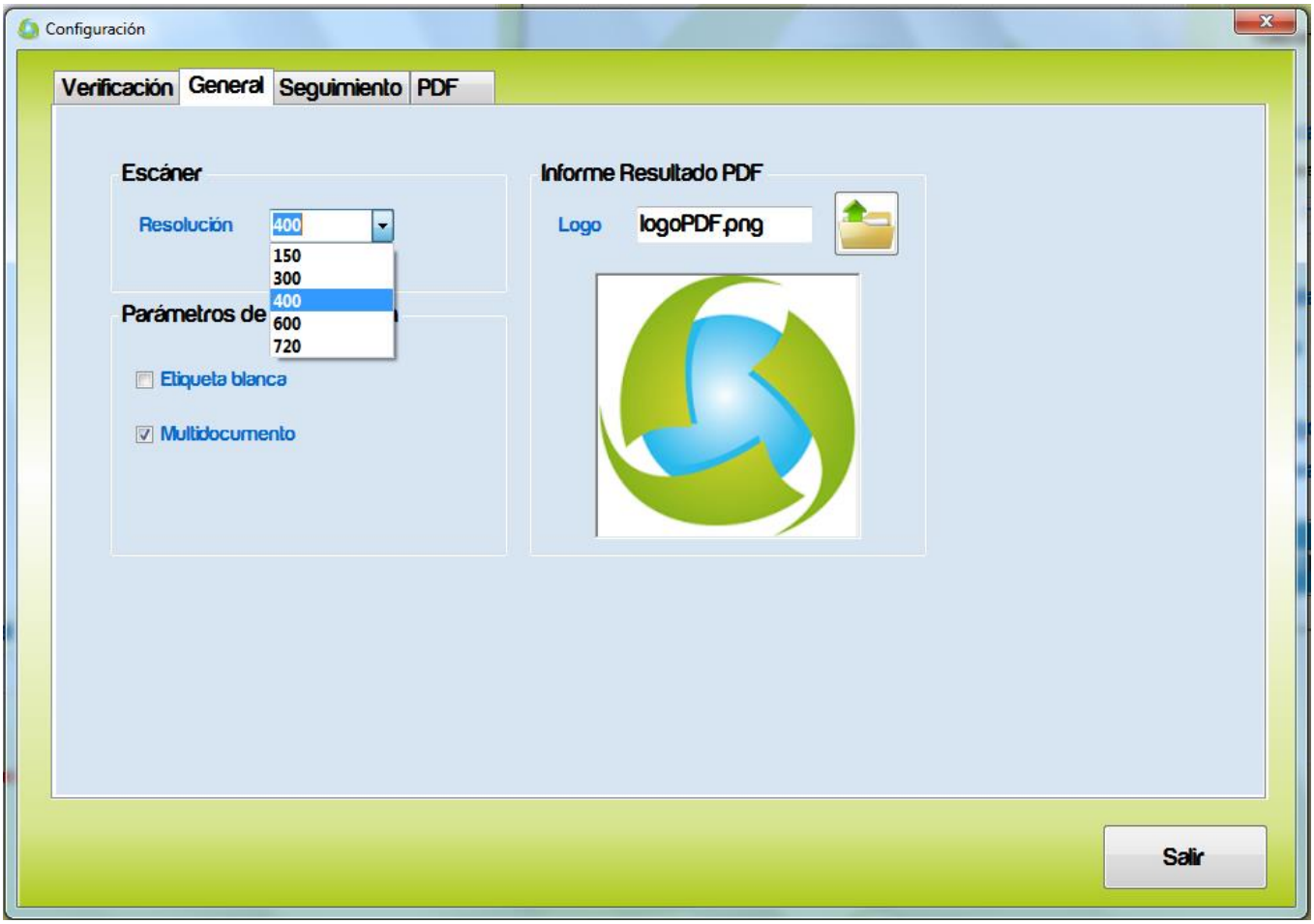

Figura 20: Opciones; General

La siguiente pestaña contiene en primer lugar las distintas resoluciones. Al principio solo se nos pedía a 400dpi pero hemos agregado las otras por el requisito post-beta de ó EscaneoVariasResolucionesô Este valor será pasado a continuación al escáner con

```
_SetScannerParameter("SCN_DPI ".ToCharArray(), dpiResolucion.ToCharArray());
```

para que la imagen resultado tenga esta resolución, a la par que se guardará también en el sistema para que al transformar el pdf en imagen también se haga a esta resolución, puesto que si ambas imágenes a comparar no tienen la misma resolución podrían salir muchos falsos positivos cuando busca errores (por no tener el mismo tamaño, por estar unos más borrosos que otros, etc.).

Aquí también indicamos si es una etiqueta blanca, es decir, si no hay diferencia entre el fondo del escáner y el documento en caso de documentos pequeños. Necesitamos indicarle esto a la función de comparación porque según si todo es blanco o no necesitara mapear el documento siguiendo los bordes, en caso de no

ser etiqueta blanca, o siguiendo las líneas de texto, en caso de sí ser etiqueta blanca.

El multidocumento, cuyo requisito tiene el mismo nombre, está marcado por defecto puesto que el cliente suele imprimir varios prospectos en una misma hoja, pero en caso de que quisieran escanear documentos ya cortados de uno en uno podrían indicarle esto a la función de comparación, que lo usa para cortar en una primera instancia el documento en varios o no, y así evitar que se corte la hoja escaneada en sitios que no debería porque la función de comparación ha entendido que hay varias hojas.

En último lugar podemos ver en la Figura 20 que aquí también indicamos la ruta, con su respectiva previsualización, del logotipo que se mostrará en los informes que se generen al comparar.

4.2.4.3 Seguimiento

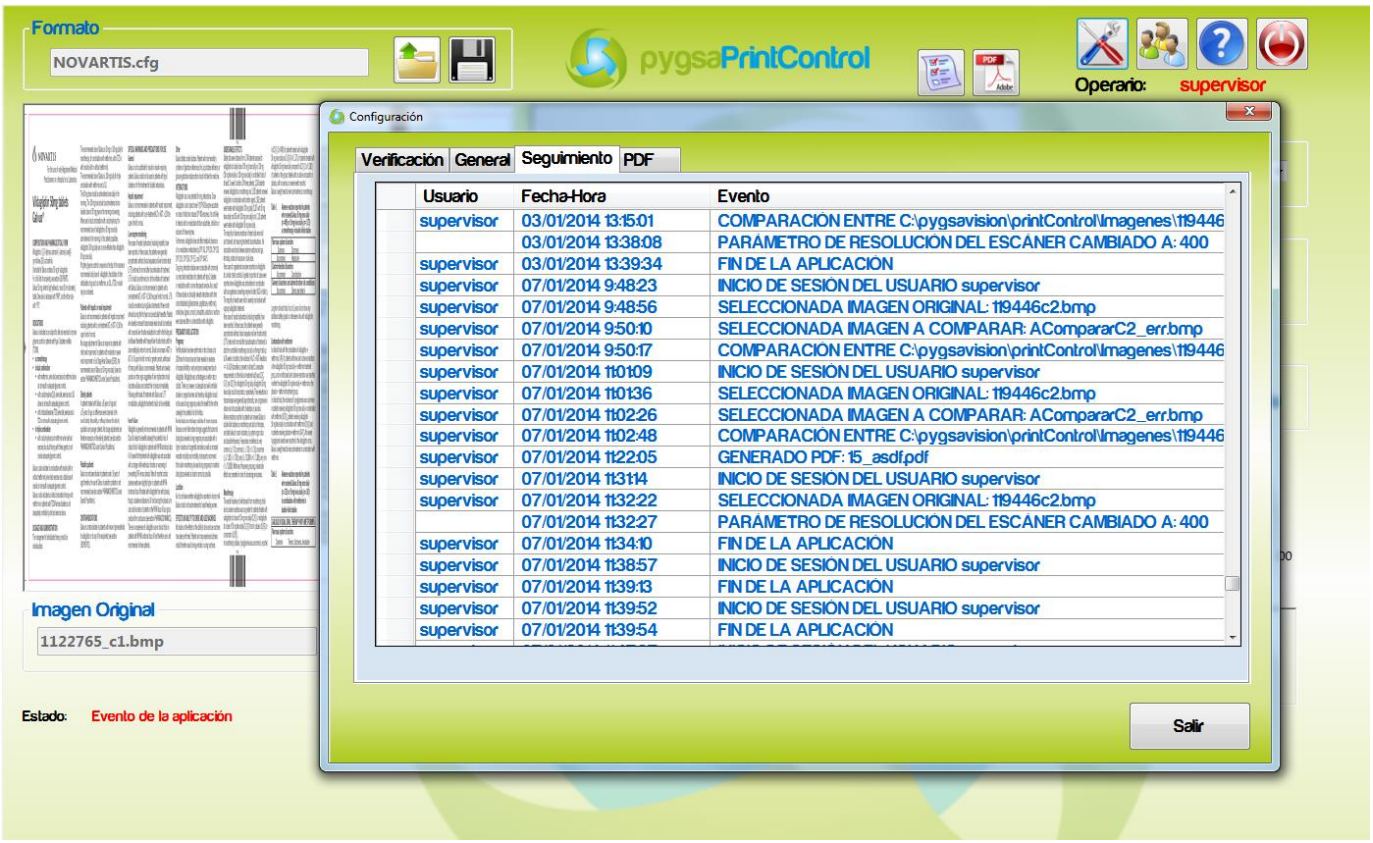

Figura 21: Opciones; Seguimiento

El seguimiento mostrado en la Figura 21 no se nos pidió específicamente, pero por seguridad y por facilidad para nosotros de saber que ha pasado decidimos poner en puntos claves del código mensajes que guardamos en un archivo de texto plano y que sencillamente dice quien, cuándo y qué ha pasado en cada momento de uso de la aplicación.

Se nos pidió que mantuviéramos esta funcionalidad para que así los supervisores supieran que informes se habían generado y por qué operario y que sirviera este log como consulta rápida.

4.2.4.4 PDF

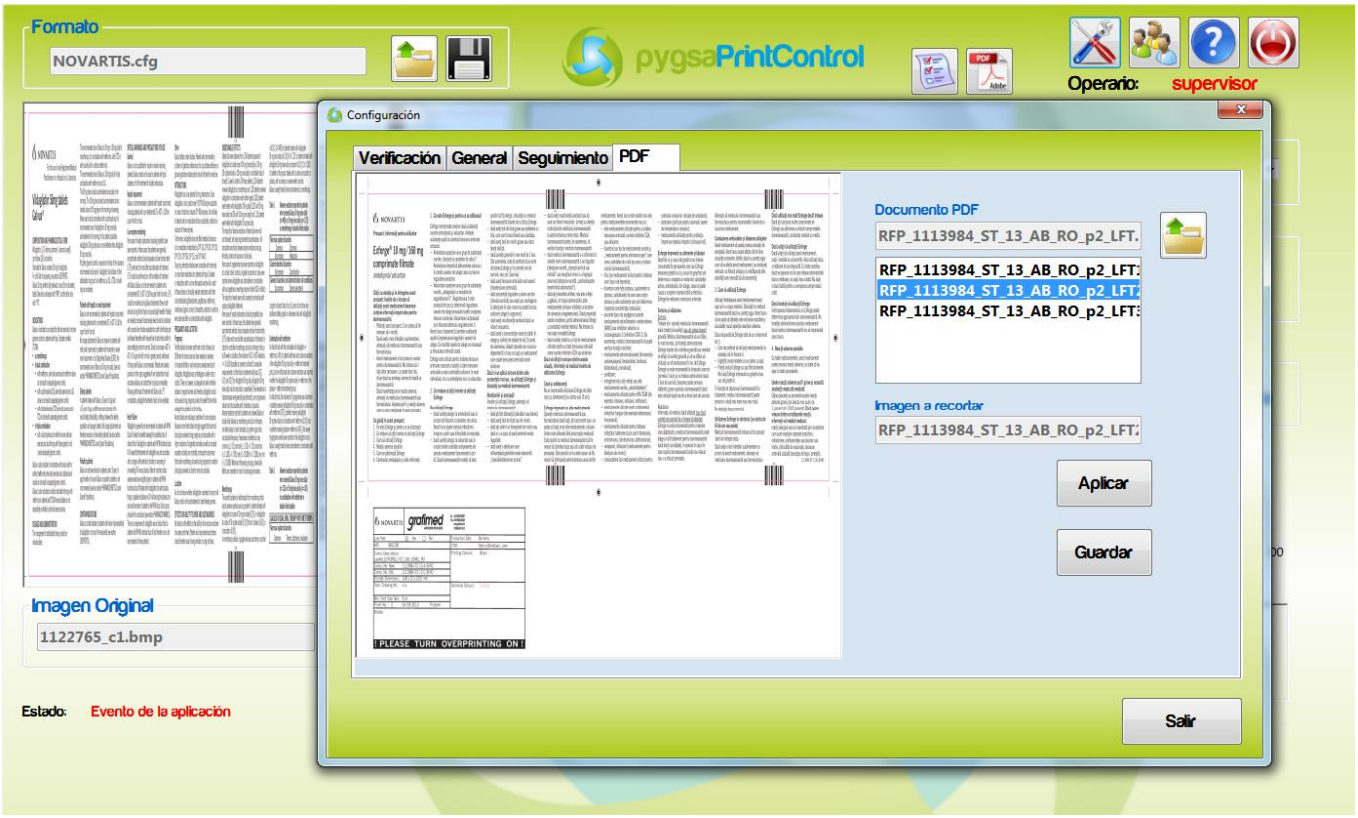

Figura 22: Opciones; PDF apertura

Por último en opciones tenemos la pestaña de recorte de PDF, que es la otra funcionalidad esencial de nuestra aplicación. Primeramente pedimos al usuario que seleccione el PDF que del que quiere sacar la imagen del documento a comparar. PYGSA sistemas y aplicaciones tenía ya en uso una librería para transformar un pdf en una imagen bmp, que es la que se pidió que se usará en el escáner también. Se selecciona la ruta a la imagen y se le pasa a la librería que transformará las varias páginas en imágenes bmp. Como sabemos el nombre del PDF buscar estas imágenes, con el mismo nombre pero con el número de página al final, es fácil. Así que las cargamos en la lista y cuando el usuario selecciona una de las imágenes hacemos el mismo procedimiento de reducir la calidad para que el PictureBox no consuma demasiada memoria.

El usuario a continuación puede seleccionar la zona a recortar del mismo modo que seleccionábamos las máscaras, esta región marcada y modificada según la escala será la del recorte si así lo quiere el usuario. SI pulsa aplicar haremos dos cosas, en primer lugar crearemos un Rectángulo (estructura de .Net) al que le pasaremos la posición y tamaños seleccionados, y dibujaremos la imagen siguiendo ese cuadrado mostrando al usuario lo que teóricamente vería al recortar y sustituyendo la imagen anterior. Como se nos pidió en

1DenominaciónDeArchivosô el usuario entonces, si quiere recortar la imagen de ese modo pulsará ógguardarôy a continuación la aplicación le pedirá un nombre, como podemos ver en la Figura 23.

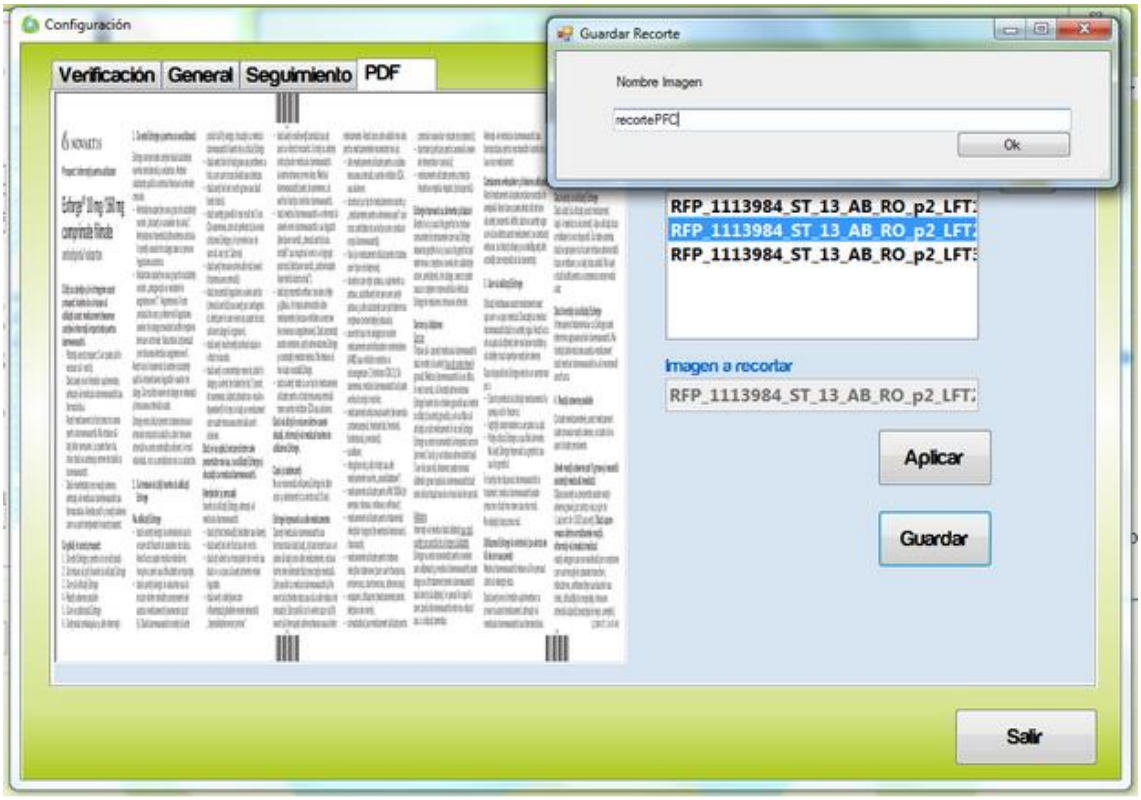

Figura 23: Opciones; PDF recorte

Al guardar es cuando nosotros hacemos a segunda parte del proceso, que es transformar las coordenadas del recorte a su equivalente en la imagen de tamaño completo, cargaremos internamente sin mostrar en ningún momento la imagen completa, crearemos la nueva del tamaño del recorte y le especificaremos cual es la zona que queremos guardar en esa imagen nueva con el nombre que le ha dado el usuario.

Después todas las imágenes intermedias deben ser borradas y descargadas para liberar el máximo espacio y memoria posible.

4.2.5 Visualización de errores

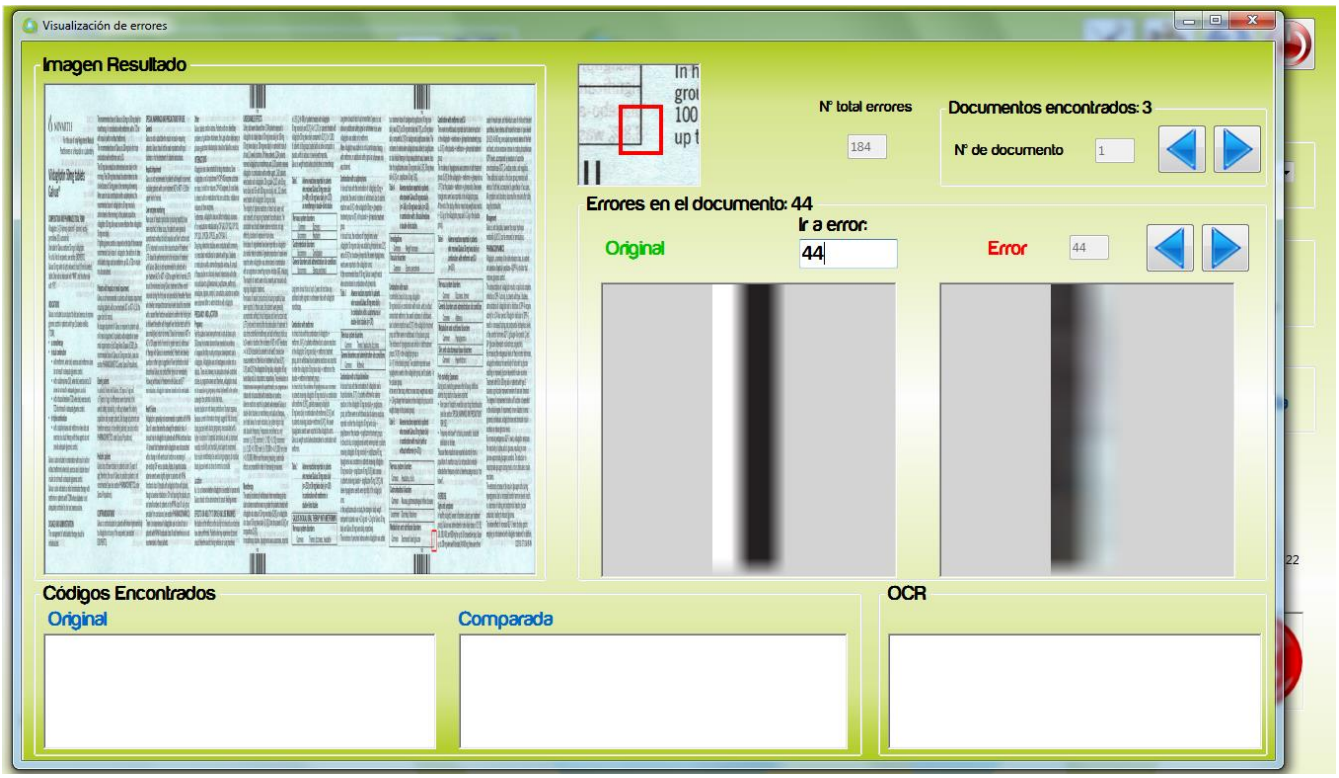

Figura 24: Visualización de errores

El último de nuestros menús es el de visualización de errores, que como el nombre bien indica cumple el requerimiento de ôMenuErroresõ Si el documento escaneado es multidocumento, es decir, en la misma hoja escaneada estaba varias veces repetido el mismo documento, este será dividido y mostrado hoja por hoja en el PictureBox de la izquierda. Con el control de ñDocumentos encontrados:ò de arriba a la derecha podemos pasar entre los varios documentos. Esto es así porque, tal y como se nos pide en ôVisualizacionGrandeõnecesitamos tener una vista global del documento y donde se encuentran los errores. Aunque sí se puedan ver donde están los errores con un recuadro rojo (abajo a la derecha en la Imagen Resultado de la Figura 24) pero no se ve lo suficientemente de cerca como para distinguir el error, para eso tenemos el requerimiento de ôVisualizacionPequeñaõ que son las dos PictureBox de la derecha, donde se ve exactamente cuál es el error y que hay en el documento original y en el escaneado.

La función de comparación nos devuelve las posiciones y tamaños de los errores que ha encontrado, nosotros sencillamente dibujamos un cuadro rojo alrededor de esta posición para la imagen grande, y hacemos un rect del tamaño especificado en ambas imágenes que ya tenemos cargadas en el sistema en el menú principal y es lo que mostramos en las dos ventanas de visualización pequeña de errores. Podemos pasar estos errores de uno en uno con los botones, o podemos ir a un error en concreto con la ôConcretacionErroresõen la que sencillamente, al tener guardadas las posiciones en un array, vamos a la posición especificada y cargamos ese error.

En este menú se nos pidió también tener un zoom, puesto que para errores muy pequeños en el documento grande no se veía apenas el recuadro rojo, y en la visualización pequeña resultaban muy abstractos como para reconocer qué era el error. Cuando el usuario pasa por encima de la PictureBox del documento, llamada picBoxDoc se pasa a la función del zoom la posición del ratón, y usando un método parecido al que usamos para mostrar los errores, mostramos la zona de alrededor de ratón ampliada.

```
private void specificZoomedImage(float xl, float yt)
{
      //Calculamos el tamaño del recuadro de zoom y de la cruz del centro
        int zoomWidth = picBoxDoc.Width / 4;
        int zoomHeight = picBoxDoc.Height / 4;
        int halfWidth = zoomWidth / 2;
        int halfHeight = zoomHeight / 2;


      //Creamos un bitmap donde vamos a dibujar el trozo de imagen del zoom
        Bitmap tempBitmap = new Bitmap(zoomWidth, zoomHeight,
                                       PixelFormat.Format24bppRgb);


      //Le damos formato al bitmap y lo preparamos para cargar la region de
      interes
        Graphics bmGraphics = Graphics.FromImage(tempBitmap);
        bmGraphics.Clear(Color.White);
        bmGraphics.InterpolationMode = InterpolationMode.HighQualityBicubic;


      //Calculamos la escala del zoom
        float xScale = ((float)picBoxDoc.Width)/picBoxDoc.Image.Width;
        float yScale = ((float)picBoxDoc.Height)/picBoxDoc.Image.Height;


        int posX = (int)(xl - halfWidth);
        int posY = (int)(yt - halfHeight);


      //Dibujamos de la imagen del documento, un rectanculo del tamaño de la
      picture box del zoom, de la region de interes del documento con el
      resultado.
        bmGraphics.DrawImage(picBoxDoc.Image,
                        new Rectangle(0, 0, zoomWidth, zoomHeight),
                        new Rectangle(posX, posY, zoomWidth, zoomHeight),
                        GraphicsUnit.Pixel);


      //Y esta region seleccionada antes es la que mostramos en la caja del
      zoom.
        pictureBoxZoom.Image = tempBitmap;
        pictureBoxZoom.Refresh();
 }
```

5. Evaluación y pruebas

Una vez realizadas todas las modificaciones pertinentes a la aplicación decidimos testear el programa para ver tiempos.

Haciendo un ciclo completo, es decir, registrarse en la aplicación, abrir un documento PDF, recortar la imagen y cargarla en la pictureBox correspondiente, escanear un documento, comparar las imágenes y crear un formato nos quedábamos sin memoria a partir del tercer ciclo completo seguido. Esto lo solucionamos como explicamos en las secciones 5.1 y 5.2.

Tras aplicar los cambios en la memoria necesarios, hicimos 25 ciclos completos y no se quedó sin memoria, por lo cual asumimos que no habría problema en un dia de trabajo normal para el cliente, y así ha sido.

Respecto a los tiempos, vimos que había mucha varianza según el tamaño del documento y la resolución de escaneo. El nivel de comparación también introducía variaciones de tiempos, pero estas eran como máximo de 30 segundos y solo en caso de documentos de tamaño A2 así que las hemos considerado despreciables para lo que nos concierne.

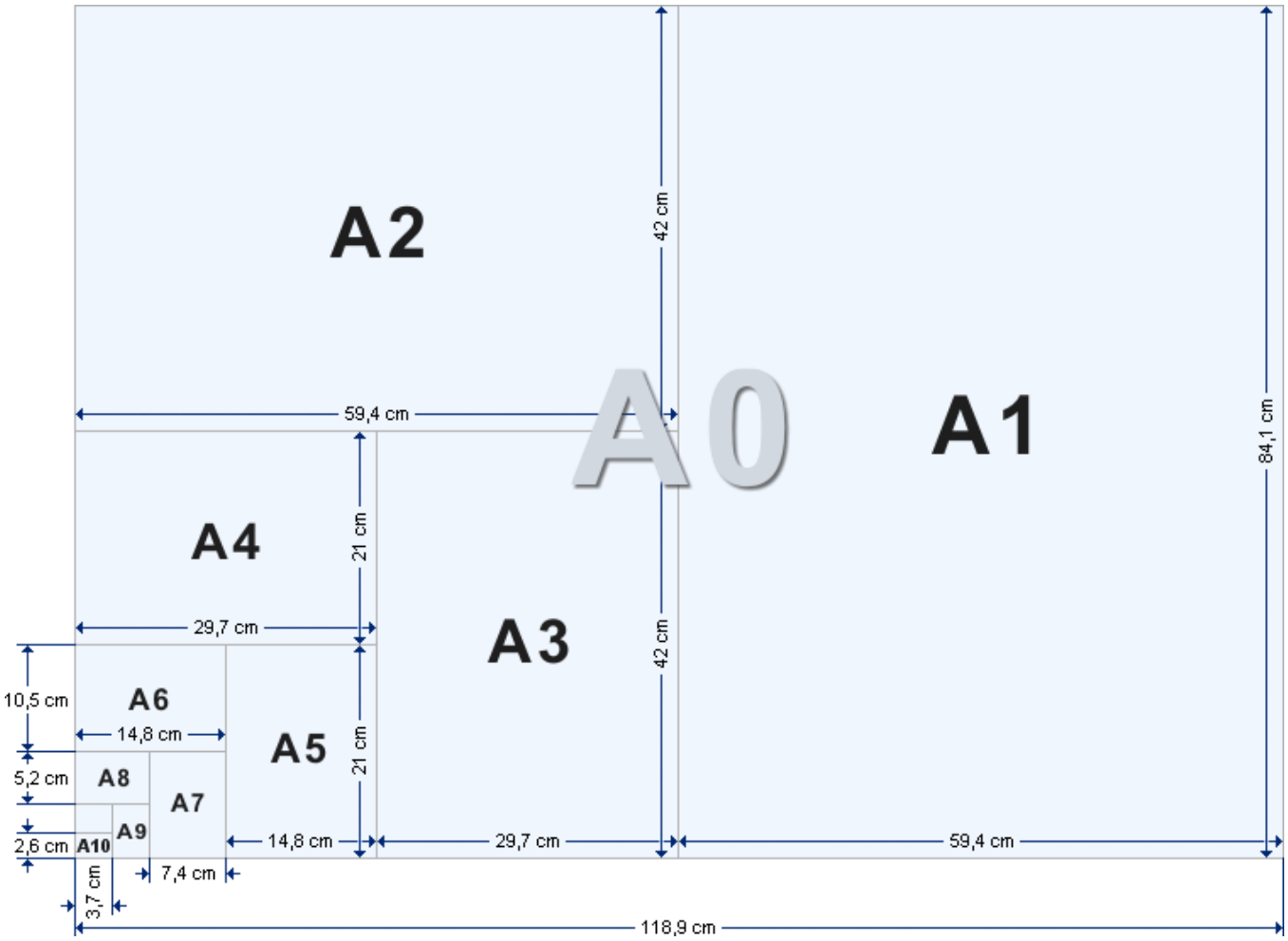

Figura 25: Tamaños de papel.

Aquí presentamos una tabla de los tiempos actuales que el cliente ha considerado válidos y que es lo que está usando. Van a trabajar de normal con imágenes de Din A2 como mucho, y en algunas ocasiones tamaño A3. El tamaño A4 e inferior ha sido incluido puesto que si quizás este cliente no, futuros clientes pueden necesitarlo.

Tiempos	<=300dpi	400dpi	>=600dpi
Din-A4	Comparación: 0.5min Ciclo Entero: 1.5min	*Comparación: ~0.6min* Ciclo Entero: 2min	Comparación: 0.75min Ciclo Entero: 3min
Din-A3	Comparación: 0.7min Ciclo Entero: 2min	Comparación: 1min Ciclo Entero:3.5min	Comparación: 1.5min Ciclo Entero: 5min
Din-A2	Comparación: 1.5min Ciclo Entero:3.5min	Comparación: 2min Ciclo Entero: 5.5min	Comparación: 3.5min Ciclo Entero: 7min

Lo que más tiempo consumía, a parte de la comparación en sí, era el cargado de los pdfs para resoluciones altas, y el recorte y guardado de la imagen en cuestión,

puesto que para poder recortar necesita cargar previamente en memoria todas las páginas del pdf y esto a resoluciones altas sobre todo es muy costoso.

Se nos pidió en los requerimientos que tardásemos menos de 5 minutos para que los operarios tuvieran tiempo de hacer varias comparaciones en un día, puesto que el uso normal de la aplicación será de A2 a resolución media de 400dpi y se tarda aproximadamente 5.5 minutos si se recorta imagen cada vez, y como de normal pueden recortar varias imágenes a la vez antes de escanear o compararán varios documentos impresos con la misma imagen se nos han aceptado estos tiempo como válidos.

5.1 Problemas con tamaño de imágenes

Como hemos dicho en la sección 4.2.2, para mostrar las imágenes al usuario usamos las PictureBox de .Net, y ellas se encargan de cargar las imágenes. El problema de esto era que una imagen cualquiera de las que íbamos a usar para comparar podía ser de 10000x8000 pixeles y de 200MB de tamaño.

El programa debía tener cargado en todo momento dos de estas imágenes en su menú principal sin poder liberarlas hasta que se cargaran las siguientes, y si era necesario rotarlas, esa imagen se crearía de nuevo, rotaria y volvería a cargar (era necesario crearla de nuevo puesto que la función de comparación no ofrecía la posibilidad de indicar rotación, solo de coger la ruta a una imagen concreta).

También cargábamos las imágenes cuando abríamos los menús de opciones y visualización de errores.

Tener que cargar y mantener tantas imágenes tan grandes hacía que los tiempos de carga de las mismas imágenes se dispararan, y el no poder liberar la memoria hasta que acabáramos todo el proceso implicaba que en algún momento la aplicación consumía toda la memoria asignada a la misma.

Para solucionar este problema nos dimos cuenta que las imágenes cargadas solo servían de referencia para el usuario puesto que imágenes tan grandes en un espacio tan pequeño resultaban ilegibles. Decidimos que, a la hora de mostrar una imagen, primero la procesaremos junto a un factor de reducción relativo a los dpi del escáner (cuanto más elevado es este número, mayor es el tamaño de la imagen y por tanto mayor debe ser el factor de escalado.

Cuando debemos mostrar una de las imágenes lo único que tenemos guardado es la ruta hasta dicha imagen, por tanto no es necesario mostrar y borrar las imágenes grandes, simplemente el resultado de la imagen escalada.

Para hacer eso usamos la siguiente función:

```
public void modificaResolucionImagen(string rutaImagenEntrada, string
rutaImagenSalida, double escala)
```

```
{
       //Obtener el tamaño de la imagen, para esto la cargamos primero
        Bitmap b0 = new Bitmap(rutaImagenEntrada);


       //Calculamos nuevo tamaño respecto al de la imagen original y lo
       multiplicamos por la escala
        int newWidth = (int)(b0.Width * escala);
        int newHeight = (int)(b0.Height * escala);


       //Creamos una imagen nueva que enseñaremos por pantalla, y para que
       ocupe menos usamos un formato de pixeles menos y menos colores.
        Bitmap newImage = new Bitmap(newWidth,
       newHeight,   System.Drawing.Imaging.PixelFormat.Format16bppRgb555);


       //Dibujamos la nueva imagen con el nuevo tamaño, con calidad alta pero
       //interpolando los pixeles para que hayan menos pero muestren
       //aproximadamente lo mismo
                using (Graphics graphics = Graphics.FromImage(newImage))
                {
                   graphics.CompositingQuality =
             CompositingQuality.HighQuality;
                   graphics.InterpolationMode =
             InterpolationMode.HighQualityBicubic;
                   graphics.SmoothingMode = SmoothingMode.HighQuality;
                   graphics.DrawImage(b0, 0, 0, newWidth, newHeight);
                }
       //Guardamos la nueva ruta que usaremos para mostrar la nueva imagen
         newImage.Save(rutaImagenSalida);
       //Borramos las imagenes creadas
         b0.Dispose();
         newImage.Dispose();
}
```

Al principio intentábamos reducir sencillamente el tamaño de la imagen, pero al no indicar como queríamos la interpolación ni el formato de los pixeles corrompíamos la memoria del sistema puesto que no sabía qué hacer con los pixeles sobrantes.

También intentamos no tener que cargar la imagen grande, para esto obteníamos los tamaños mediante un `BmbBitmapDecoder` y accediendo directamente al header de la imagen, de donde sacábamos el tamaño, pero cuando usamos `using (Graphics graphics = Graphics.FromImage(newImage))`, Graphics requiere que tengamos la imagen cargada para poder copiarla a la nueva imagen, por eso al final optamos por cargarla, obtener los datos, copiarla y borrarla.

5.2 Problemas con memoria

Uno de los cambios que se nos pidió sobre la versión beta fue que ampliáramos a 600 y 720 los dpi del escáner porque habían unos errores que podían ocupar 4 o 5 pixeles con una resolución de 400 que era la que teníamos como estándar. Estos errores no eran detectables por la función de comparación, así que debíamos ampliar al máximo la imagen para que los detectara. Esto daba lugar a que una imagen recortada de un pdf con resolución 720dpi fuera de tamaño 17000x8000 de

tamaño y peso 250MB, y el documento escaneado, con la posibilidad de que fueran varias veces esta imagen en un mismo papel, de más de 500MB de peso, y más del doble de tamaño.

Aunque no debíamos mostrar estas imágenes como hemos explicado en el punto 4.2 si debíamos cargarlas internamente, y varias veces, con lo que nos quedábamos sin memoria rápidamente.

Nuestra aplicación es un proceso de 32bits y por tanto tiene 2^32 bit (4GB) de espacio de direcciones. Esto quiere decir que cada puntero tiene un tamaño de 4Bytes y por tanto estamos limitados a 4 millones.

Cuando llegamos al punto de cargar estas imágenes llevamos rato corriendo la aplicación, y mostrando todas las interfaces por pantalla, gran parte de estas direcciones están siendo usadas, quizás con suerte pudiéramos cargar una de las imágenes grandes, pero la probabilidad de que tengamos dos veces el espacio necesario para cargar estas imágenes de 500MB con direcciones de 32 bits todas juntas es muy escaso.

Esto lo solucionamos indicando a Visual Studio que necesitábamos para esta aplicación un espacio de direcciones de tamaño LARGE de forma que pudiera cargar las imágenes el tiempo suficiente para hacer sus copias pequeñas a mostrar por pantalla, recortarlas en caso del pdf y poder mandar a la función de comparación su ruta.

Esto se hizo indicando en las opciones de build

```
call "$(DevEnvDir)..\..\vc\vcvarsall.bat" x86
"$(DevEnvDir)..\..\vc\bin\EditBin.exe" "$(TargetPath)"  /LARGEADDRESSAWARE
```

Que habilita la opción (en la primera línea) e indica al linker que la flag LARGEADDRESSWARE se ha activado (en la segunda línea) y que por tanto el binario que vamos a crear podrá trabajar con un espacio de punteros superior a 2G.

Hacer esto tiene como consecuencia que el bit más alto del puntero será un 1, que significa en complemento a 2 que es negativo, y algunas aplicaciones pueden no estar preparadas para trabajar con punteros ónegativosôpero este no fue nuestro caso.

6. Conclusiones

Con este proyecto aprendí a usar .Net y C#, que aunque no se ven durante la carrera son lo suficientemente similares a los lenguajes y entornos aprendidos como para que la curva de aprendizaje no fuera muy exagerada.

Aunque son fáciles de usar, justamente por esto están muy limitados en algunas áreas que he tenido que aprender a sortear, como tener un recolector de basura propio que se encarga de la descarga de memoria y que no podemos usar nosotros cuando queramos. Ha sido interesante ver la diferencia que marca que un sistema sea tan fácil de usar a cambio de sacrificar otras cosas.

Hablar con un cliente también ha sido completamente nuevo, tener que negociar y entender que prestaciones quieren en su aplicación y cómo las quieren ha sido una buena experiencia, y tener que hacer una aplicación para uso de otros, debiendo hacer una interfaz simple y clara no solo para ti, sino para personas que pueden tener conocimientos escasos de usos de ordenador y tecnología en general, ha sido un reto.

Considero que deber de aplicar lo aprendido durante la carrera a un entorno laboral, debiendo hacer las cosas para otros y con fechas estrictas ha sido muy gratificante.

Por otro lado luchar de forma constante contra la escasez de memoria me ha enseñado muchas cosas y me ha forzado a exprimir mi ingenio para que este forcejeo con la memoria e imágenes fuera totalmente transparente para el usuario.

Y por último el tener que trabajar con el software de terceros, como es la función de comparación, y el hardware ya comprado de la imprenta, como el escáner, me ha obligado a adaptar la aplicación a otras personas mientras se adecuaba a las necesidades del cliente y cumplía unos estándares de calidad por parte de la empresa y del cliente, y a tener de aprender rápidamente su uso mientras programaba la aplicación.

En conclusión, este proyecto me ha servido para experimentar lo que es el mundo laboral y a tener experiencias reales con clientes, no solo en ámbito académico.

7. Trabajo futuro

A día de hoy el cliente está trabajando satisfactoriamente con la aplicación y haciendo sus tareas sin problema alguno, aunque siempre hay cabida para mejoras.

En un futuro esperamos poder migrar la aplicación completamente a sistema de 64bits que es el procesador que tienen en el ordenador que usan para comparar documentos. Para poder hacer esto sería necesario en primer lugar encontrar otra función de comparación que funcione en 64 bits o que la que estamos usando sea actualizada.

También queremos acabar los requisitos que quedaron relegados a futuras versiones, como es el poder detectar y descifrar códigos de barras y QR, al igual que poder detectar y traducir algunas de las zonas OCR como son pesos, fechas, etc.

Esto último está más orientado a futuros clientes, que quieran poder escanear etiquetas de productos como pueden ser de aceites o de carnes y poder saber si el peso, la fecha de caducidad, precio de venta al público, etc. han sido bien escritos y la etiqueta es la correcta.

Para estos futuros clientes también esperamos poder detectar colores y ver si hay diferencia entre la etiqueta impresa y la diseñada originalmente. Aunque para esto también sería necesario que la función de comparación detectara dibujos puesto que la que usamos actualmente está orientada a leer y comparar texto mayoritariamente.

En esta versión, como mejoras futuras para este cliente y otros, nos gustaría poder tener un sistema de autenticación basado en bases de datos SQL de forma que todo sea más seguro y el cliente pueda importar bases de datos propias en vez de incluir uno a uno los usuarios con la aplicación y modificando el documento xml.

También hemos sido notificados que el sistema de máscaras no es todo lo fácil de usar que nos gustaría, así que el próximo sistema debería ser en una ventana separada y donde podamos añadir y quitar las máscaras una a una, y activar una máscara en concreto en vez de activarlas todas y borrarlas todas que es lo que se puede hacer en la actualidad.

Por supuesto la aplicación completa está montada de forma que si algún futuro cliente quiere algo específico a su producto, se pueda añadir sin problema en opciones o modificar el sistema principal para añadirle cualquier otra funcionalidad que quieran ahí.

Referencias

http://msdn.microsoft.com/es-es/magazine/ff796223.aspx
http://blogs.msdn.com/b/calvin_hsia/archive/2010/09/27/10068359.aspx
http://msdn.microsoft.com/en-us/library/wz223b1z.aspx
http://msdn.microsoft.com/es-es/library/cc221403(v=vs.95).aspx
http://www.codeproject.com/Articles/21097/PictureBox-Zoom
http://stackoverflow.com/questions/3911568/wpf-out-of-memory-exception-when-loading-large-amount-of-bitmaps-in-single-insta