WWW.UOC.EDU

Proyecto Fin de Carrera

GUI de sistema de preajuste para rotativas

Javier Poza

13/06/2011

Memoria

Contenido

1. DESCRIPCIÓN DEL PROYECTO

1.1 Introducción

Este proyecto tiene como objetivo el desarrollo de una aplicación de escritorio cuya finalidad es resolver una necesidad real, la de proporcionar una interfaz gráfica para poder controlar el sistema que gestiona las máquinas que componen una rotativa para que la impresión de las publicaciones alcance la mayor calidad posible.

Para la elaboración de dicha aplicación se ha optado por usar la tecnología Microsoft .NET Framework 4.0 y la tecnología Windows Presentation Foundation.

La aplicación debe de permitir al usuario realizar unas ciertas tareas que son:

- Alta y bajas de usuarios.
- Gestión de grupos.
- Gestión de parámetros generales.
- Gestión de arquitectura de la rotativa.
- Mantenimiento de curvas.
- Gestión de trabajos de impresión.

La aplicación forma parte de un sistema con dos elementos: un servidor programador en c++ para la plataforma Linux y un cliente que es el objeto el cual estamos tratando. El servidor queda fuera del alcance de este proyecto y será tratado como un elemento externo con el cual tenemos un contrato con los requerimientos que él tiene hacia el cliente y los requerimientos que el cliente ha de tener hacia él.

El proyecto ha sido llevado a cabo en varias etapas que corresponden a las entregas requeridas por el Trabajo de final de carrera, que son:

- Entrega 1 (PEC 1): Plan de trabajo
- Entrega 2 (PEC 2): Análisis y diseño
- Entrega 3 (PEC 3): Implementación
- Entrega 4 (Final): Memoria y presentación

El presente documento perteneciente a la entrega final recoge todas las anteriores entregas, las amplía, corrige y actualiza con la solución finalmente adoptada.

El producto final está previsto ponerlo en producción para el segundo cuarto del actual año, 2011 en una rotativa ubicada en Tarragona.

Este proyecto ha sido realizado por Javier Poza para la empresa “3TControl Precision Systems” para la cual es empleado y bajo la tutela de Juan Carlos Gonzalez Martín, consultor de la UOC.

1.2 Justificación

En las rotativas el papel y la tinta son las principales materias primas que se usan para la impresión de publicaciones. Los costes de la impresión de una publicación serán por tanto directamente proporcionales a la cantidad usada de dichos elementos.

Por otro lado el éxito en la impresión de una publicación como puede ser un periódico no es algo sencillo ya que se imprime a una gran velocidad con un conjunto de máquinas que no son precisas y que tienen muchos puntos de ajuste. Al imprimir una publicación habrá pues que ajustar todos esos puntos de ajuste de forma manual para que la publicación tenga una calidad aceptable. Esto provoca que normalmente en el arranque de una impresión los primeros ejemplares se descarten ya que se usan para conseguir ajustar la máquina a la publicación que se está imprimiendo. Una vez se imprimen los ejemplares correctamente se validan y se empiezan a contar como ejemplares buenos.

Debido a la velocidad de impresión (entre 30.000 y 50.000 ejemplares / hora) el tiempo de ajuste inicial supone una pérdida de recursos elevado.

Con un preajuste inicial conseguiríamos minimizar este tiempo y por tanto ahorrar en materias primas.

Otro efecto deseado es el de conseguir objetividad en la calidad del producto. Como se ha explicado anteriormente los ajustes los realizan los operarios encargados de realizar la publicación. Estos toman muestras, las analizan y subjetivamente deciden los parámetros a ajustar en las máquinas. Con un preajuste automático conseguiríamos que las publicaciones sean más fieles a las fuentes originales y por tanto el aumento de calidad será considerable.

También reduciríamos el tiempo de impresión, factor crucial en el sector prensa ya que tan pronto se imprimen los periódicos se cargan y se distribuyen para cumplir con los plazos de tiempo.

Por último una de las ventajas obtenidas es que con un sistema de preajuste podemos visualizar el estado de las zonas de tinta, que son los componentes que se controlan, y ver si es necesario cambiar alguna por desgaste o fallo facilitando enormemente las labores de mantenimiento.

Existen otros productos en el mercado destinados a satisfacer la misma necesidad, pero son sistemas que implementan mucha más funcionalidad encareciendo el producto final y lo hacen más complicado y por lo tanto, menos atractivo. Además estos productos también tienen sus deficiencias.

Este producto nace de la necesidad de tener un producto que haga únicamente la tarea a la que está destinada con un coste moderado, sea sencillo, fácil de mantener y cumpla su función principal más eficientemente que las otras herramientas.

1.2.1 Retorno de la inversión

Para justificar el desarrollo del proyecto se expone un estudio realizado sobre la justificación de la compra del producto final. Para ello nos basaremos en el retorno de la inversión (ROI). Se ha estimado que en torno a 12 meses se recupere la inversión. Para ello nos basaremos en los costes del papel reales para el año 2010 y su coste al desecharlo al inicio de la impresión, la etapa de ajuste:

- Media de páginas por periódico: 40 pág.
- Gramaje medio del papel: 45 gr.
- Peso medio de un periódico: 73 gr.
- Número medio de publicaciones por día: 20
- Ahorro de ejemplares por preajuste por cada arranque: 300 ejemplares.
- Ahorro por día: 438.000 gr.
- Precio de papel: 625€ / tonelada
- Ahorro de un día: 273.75 €
- Precio sistema de preajuste: 100.000 €
- ROI: 365 días.

Es decir, que tras el primer año no percibiremos mejoras económicas, pero tras el segundo año el ahorro habrá sido de más 100.000 € y cada año posterior será superior al anterior teniendo en cuenta que el precio del papel sube cada año.

1.3 Objetivos

El objetivo de este proyecto es el desarrollo de una interfaz gráfica para la gestión del sistema de preajuste de una rotativa usando tecnología .NET de Microsoft mediante la cual el usuario pueda configurar el sistema para que tras configurarlo funcione correctamente de forma autónoma y por otro pueda realizar un cierto mantenimiento.

Por otro lado y dado la doble naturaleza del proyecto (didáctica y comercial), se detallarán objetivos a nivel de conocimiento que el autor deberá conseguir para la consecución del proyecto:

- Profundizar en tecnología *.NET*.
- Conocimiento de las posibilidades de *Visual Studio 2010*.
- Aprender a realizar la capa de diseño con *WPF* que usa estructura *XAML*.
- Diseñar e implementar una base de datos en *SQL Server 2010*.
- Implementar y usar tecnología *LINQ*.
- Implementar la capa de negocio mediante *LINQ To SQL*.
- Uso de librerías externas como *WPF Toolkit* para mostrar gráficos.
- Repaso y puesta en práctica de asignaturas estudiadas en la carrera como *Estructura de la información* o *Ingeniería del software*.
- Toma de datos y análisis y estimación del proyecto.
- Implementar un sistema que facilite el multi-idioma.

1.4 Alcance global del proyecto

El alcance de este proyecto engloba el análisis, diseño e implementación de la aplicación cliente del sistema de preajuste, la aplicación servidor quedará excluida.

Se divide en varias fases: planificación, análisis, desarrollo y documentación y presentación.

La fase de planificación finalizará cuando se obtenga un planteamiento general del proyecto y unas fechas estimadas para las distintas fases y subfases.

La fase de análisis finalizará cuando se obtenga una serie de requerimientos tanto funcionales como técnicos de lo que se debe desarrollar, así como cualquier diagrama necesario que despeje cualquier posible duda que pueda surgir en el desarrollo y que pueda hacer que se replantee aspectos ya implementados. En el caso de este producto serán diagramas de clases, diagrama de base de datos, de casos de uso y pantallas de cómo debe de ser la interfaz de usuario.

La fase de desarrollo finalizará cuando se obtenga el ejecutable que proporcione una interfaz gráfica que permita gestionar los principales aspectos del sistema preajuste y cumpla con los requisitos que se detallan en el presente documento, así como la base de datos implementada.

La fase de documentación finalizará con la elaboración del presente documento.

La fase de presentación finalizará cuando se elabore y exponga una presentación del proyecto.

El proyecto deberá seguir las fases en el orden expuesto y finalizará con la última fase.

1.5 Resultados esperados

Los resultados esperados y entregados son:

- Aplicación de escritorio: Implementación del sistema. Se obtendrá un código fuente que podrá ser compilado sin requerir dependencias externas en Visual Studio 2010
- Base de datos: Ficheros para importar en Base de datos SQL Server 2008 R2
- Manual de usuario: Manual en castellano muy intuitivo que enseña a manejar la aplicación con numerosas capturas de pantalla.
- Memoria del proyecto: El presente documento que contiene toda la información referente al proyecto
- Presentación.

1.6 Planificación

1.6.1 Planificación inicial

La planificación estimada es la siguiente:

Nombre de tarea	Duración	Comienzo	Fin
Inicio del proyecto	0 días	mié 02/03/11	mié 02/03/11
Definición y planificación del proyecto	**15 días**	**mié 02/03/11**	**mié 16/03/11**
Ejecución del proyecto	**68 días**	**jue 17/03/11**	**lun 23/05/11**
Análisis y diseño	**26 días**	**jue 17/03/11**	**lun 11/04/11**
Especificación requisitos	4 días	jue 17/03/11	dom 20/03/11
Diagrama Entidad-Relación Base de datos	3 días	lun 21/03/11	mié 23/03/11
Modelo relacional	2 días	jue 24/03/11	vie 25/03/11
Diagrama de casos de uso	3 días	sáb 26/03/11	lun 28/03/11
Diagrama de clases	2 días	mar 29/03/11	mié 30/03/11
Diseño protocolo comunicación	3 días	vie 01/04/11	dom 03/04/11
Diseño de interfaz de usuario	6 días	lun 04/04/11	sáb 09/04/11
Elaboración documento PEC2	2 días	dom 10/04/11	lun 11/04/11
Corrección documentación	1 día	lun 11/04/11	lun 11/04/11
Entrega PEC2	0 días	lun 11/04/11	lun 11/04/11
Implementación	**42 días**	**mar 12/04/11**	**lun 23/05/11**
Gestión de usuarios y autenticación	4 días	mar 12/04/11	vie 15/04/11
Gestión de publicaciones	5 días	sáb 16/04/11	mié 20/04/11
Gestión de torres de impresión	5 días	jue 21/04/11	lun 25/04/11
Gestión de parámetros del sistema	2 días	mar 26/04/11	mié 27/04/11
Gestión de curvas	8 días	jue 28/04/11	jue 05/05/11
Informes	5 días	vie 06/05/11	mar 10/05/11
Estado del sistema y gestión de ficheros TIFF	6 días	mié 11/05/11	lun 16/05/11
Mensajería con servidor	2 días	mar 17/05/11	mié 18/05/11
Gestión del idiomas	1 día	jue 19/05/11	jue 19/05/11
Puebas unitarias	2 días	vie 20/05/11	sáb 21/05/11
Pruebas de integración	1 día	dom 22/05/11	dom 22/05/11
Entrega PEC3	0 días	lun 23/05/11	lun 23/05/11
Cierre del proyecto	**21 días**	**mar 24/05/11**	**lun 13/06/11**
Elaboración Memoria	11 días	mar 24/05/11	vie 03/06/11
Corrección memoria	1 día	sáb 04/06/11	sáb 04/06/11
Elaboración presentación	7 días	dom 05/06/11	sáb 11/06/11
Corrección presentación	1 día	dom 12/06/11	dom 12/06/11
Entrega final	0 días	lun 13/06/11	lun 13/06/11

Tabla 1.6.1.1 – Planificación inicial

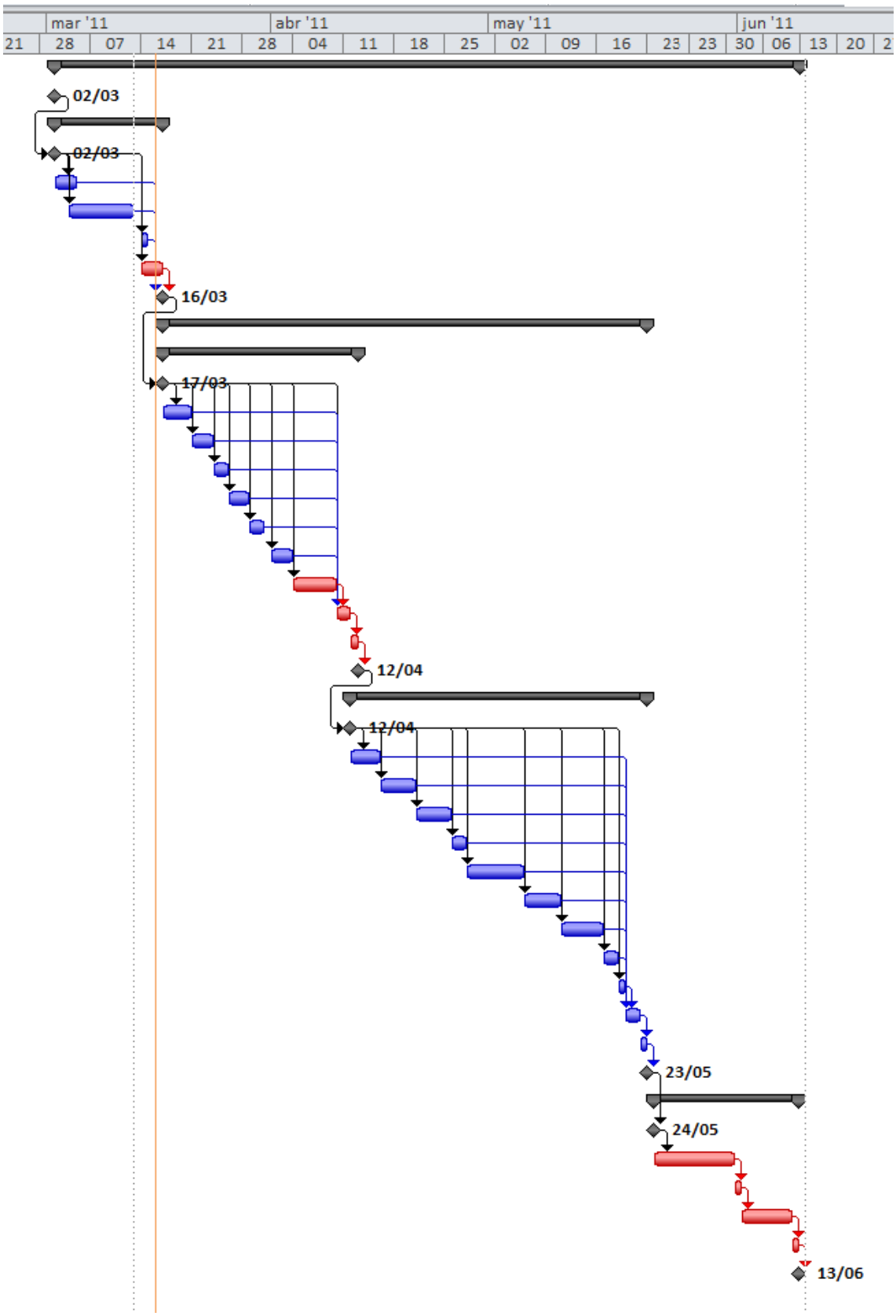

Figura 1.6.1.1 – Diagrama de Gantt

1.6.2 Planificación final

La planificación final es parecida a la inicial. Los hitos principales, que coinciden con la entrega de las PECS, se han mantenido, lo que sí ha variado los las fechas de las tareas internas.

Los cambios más apreciables son que se sobreestimaron algunas tareas del Análisis y Diseño pero se subestimo el coste del diseño de la interfaz de usuario.

Por otro lado en la etapa de implementación se subestimo el estado del sistema y gestión de ficheros TIFF.

El resultado final es el siguiente:

Nombre de tarea	Duración	Comienzo	Fin
GUI de sistema de preajuste para rotativas	**103 días?**	**mié 02/03/11**	**lun 13/06/11**
Inicio del proyecto	0 días	mié 02/03/11	mié 02/03/11
Definición y planificación del proyecto	**15 días**	**mié 02/03/11**	**mié 16/03/11**
Definición y planificación del proyecto - Comienzo	0 días	mié 02/03/11	mié 02/03/11
Recopilación de información	3 días	mié 02/03/11	vie 04/03/11
Elaboración de plan de proyecto	9 días	vie 04/03/11	sáb 12/03/11
Instalación de software	1 día	lun 14/03/11	lun 14/03/11
Estudio de software	3 días	lun 14/03/11	mié 16/03/11
Entrega PEC1	0 días	mié 16/03/11	mié 16/03/11
Ejecución del proyecto	**67 días?**	**jue 17/03/11**	**lun 23/05/11**
Análisis y diseño	**27 días?**	**jue 17/03/11**	**mar 12/04/11**
Análisis y diseño - Comienzo	0 días?	jue 17/03/11	jue 17/03/11
Especificación requisitos	4 días	jue 17/03/11	dom 20/03/11
Diagrama Entidad-Relación Base de datos	2 días	lun 21/03/11	mar 22/03/11
Modelo relacional	0 días	mié 23/03/11	mié 23/03/11
Diagrama de casos de uso	2 días	mié 23/03/11	jue 24/03/11
Diagrama de clases	2 días	vie 25/03/11	sáb 26/03/11
Diseño protocolo comunicación	1 día	dom 27/03/11	dom 27/03/11
Diseño de interfaz de usuario	13 días	lun 28/03/11	sáb 09/04/11
Elaboración documento PEC2	2 días	dom 10/04/11	lun 11/04/11
Corrección documentación	1 día	mar 12/04/11	mar 12/04/11
Entrega PEC2	0 días	mar 12/04/11	mar 12/04/11
Implementación	**41 días?**	**mar 12/04/11**	**lun 23/05/11**
Implementación - Comienzo	0 días?	mar 12/04/11	mar 12/04/11
Gestión de usuarios y autenticación	3 días	mar 12/04/11	jue 14/04/11
Gestión de publicaciones	2 días	jue 14/04/11	vie 15/04/11
Gestión de torres de impresión	4 días	sáb 16/04/11	mar 19/04/11
Gestión de parámetros del sistema	2 días	mié 20/04/11	jue 21/04/11
Gestión de curvas	8 días	vie 22/04/11	vie 29/04/11
Informes	2 días	sáb 30/04/11	dom 01/05/11

Estado del sistema y gestión de ficheros TIFF	15 días	lun 02/05/11	lun 16/05/11
Mensajería con servidor	1 día	mar 17/05/11	mar 17/05/11
Gestión del idiomas	2 días	mié 18/05/11	jue 19/05/11
Puebas unitarias	2 días	vie 20/05/11	sáb 21/05/11
Pruebas de integración	1 día	dom 22/05/11	dom 22/05/11
Entrega PEC3	0 días	lun 23/05/11	lun 23/05/11
Cierre del proyecto	**20 días?**	**mar 24/05/11**	**dom 12/06/11**
Cierre del proyecto - Comienzo	0 días?	mar 24/05/11	mar 24/05/11
Elaboración Memoria	11 días	mar 24/05/11	vie 03/06/11
Correción memoria	1 día	sáb 04/06/11	sáb 04/06/11
Elaboración presentación	7 días	dom 05/06/11	sáb 11/06/11
Corrección presentación	1 día	dom 12/06/11	dom 12/06/11
Entrega final	0 días	lun 13/06/11	lun 13/06/11

Tabla 1.6.2.1 – Planificación final

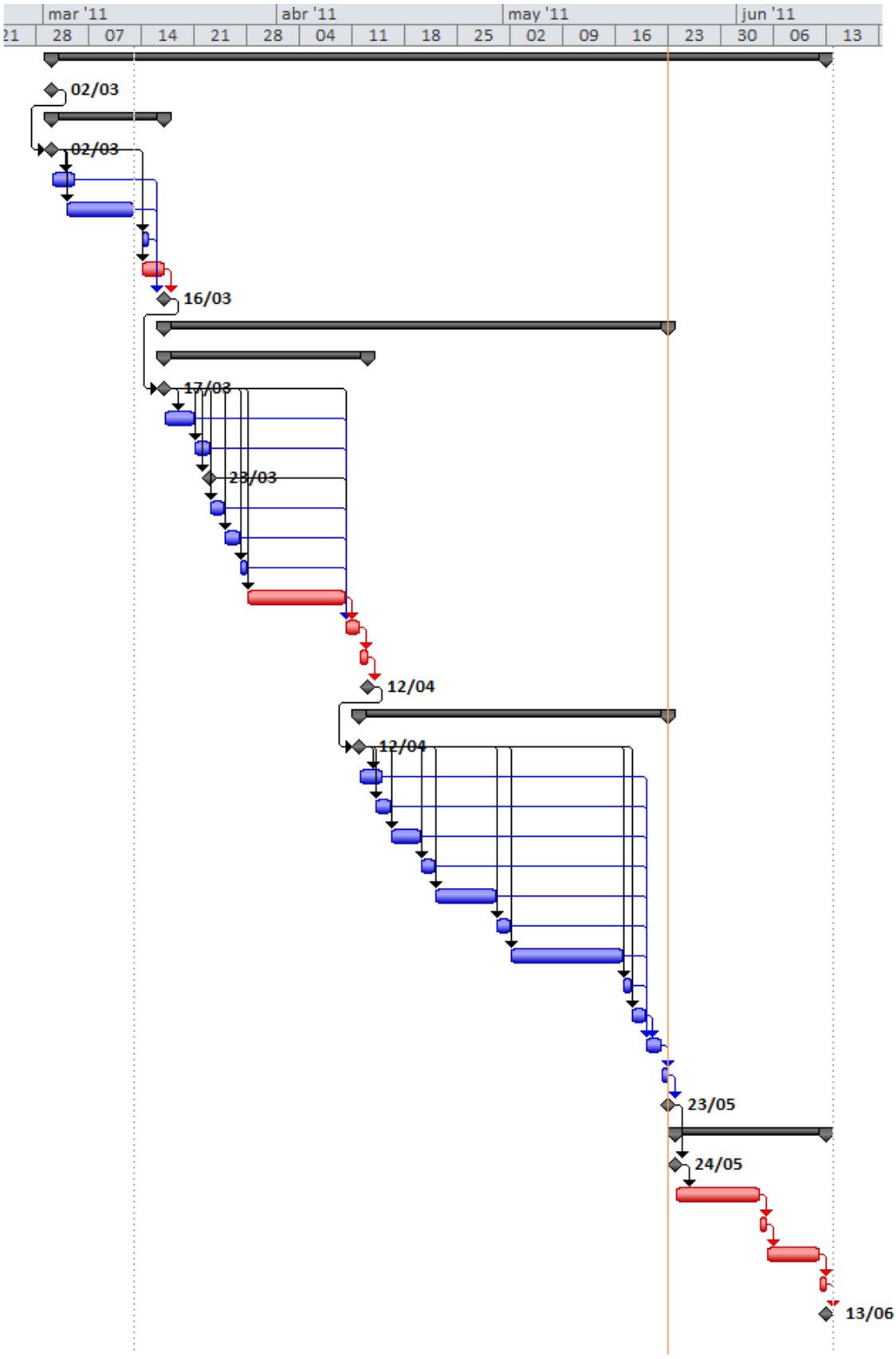

Figura 1.6.2.1 – Diagrama de Gantt

2. ANÁLISIS Y DISEÑO

2.1 Alcance y descripción general del sistema

Para comprender mejor el alcance de este proyecto expondré una breve explicación del funcionamiento de una rotativa. En una rotativa se imprimen publicaciones como diarios, revistas, etc. Estas publicaciones son enviadas a la rotativa en un formato concreto para que sean impresas, concretamente en PDF, con unas especificaciones concretas (conjunto de colores CMYK, número de páginas par múltiplo de 4, etc.).

Una vez la rotativa ha recibido los PDF tiene que adaptarlos a las "impresoras" que son las máquinas que imprimen las publicaciones. Estas máquinas se componen principalmente de dos partes bien diferenciadas:

- Plegadoras: Son las que encuadernan las hojas de la publicación una vez impresas.
- Torres de impresión: Son las que mediante chorros de tinta imprimen en el papel la publicación.

Estas torres de impresión son las que nos interesan llegados a este punto. Cada torre tiene varias zonas por donde expulsa chorros de tinta denominadas tinteros, de una manera muy similar a las impresoras convencionales de chorro de tinta. Cada torre de impresión tendrá entonces 4 hileras de tinteros, una para el color cian, otra para el magenta, otra para amarillo y otra para el negro.
Como se puede apreciar las torres de impresión se basan en el conjunto de colores CMYK, que usa dichos colores como primarios para formar el resto de colores. Dependiendo de cuanta tinta expulse cada tintero hará que el ojo humano vea un color u otro. Esta cantidad de tinta es regulable desde que no expulse nada de tinta, el 0%, hasta un máximo que es el del color primario puro, el 100%

En la siguiente figura podemos apreciar el color que forma cada color primario con la intersección con otro primario:

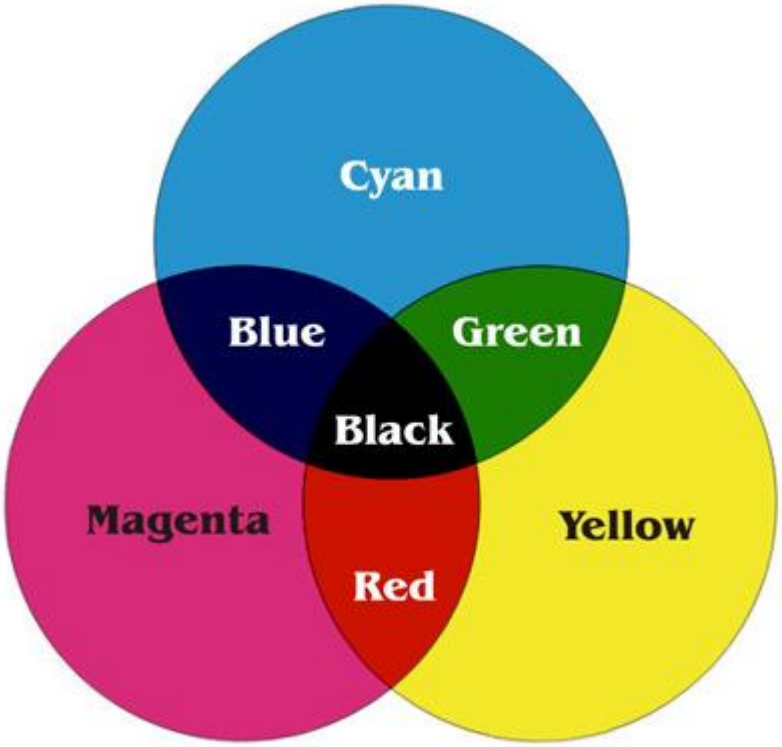

Figura 2.1.1 – CMYK

Por ejemplo si un tintero del color cian es abierto al máximo y se imprime sobre un papel que tenía el color magenta también impreso al máximo y en el que el chorro de amarillo está completamente cerrado dará el color azul que se visualiza en la figura. Pero si variamos la cantidad de apertura de alguno de los dos tinteros dará o bien un cian más tenue o bien un magenta más tenue y por tanto otro tipo de azul. Y si abriéramos algo el Amarillo daría un color grisáceo.

Cabe destacar que para mejorar el negro puro sin tener que gastar los 3 tipos de tintas y teniendo en cuenta que muchas publicaciones se imprimen en escala de grises, se utiliza una tinta negra ya mezclada como un color primario más.

De esto se puede destacar que el resultado de una impresión y su fidelidad al documento original, los PDF, es algo hasta cierto punto subjetivo, por lo tanto no es algo que vaya a estar completamente bien o completamente mal. Depende de la percepción del impresor de la mezcla producida por los colores. Este irá variando la apertura de los tinteros a lo largo de la impresión para aplicar correcciones y mejorar la fidelidad (un azul demasiado oscuro volverlo más claro, un marrón muy claro hacerlo más oscuro, etc.).

Volviendo a los fichero PDF originales, una vez la rotativa tiene dichos ficheros mediante unas herramientas específicas los descompone en los 4 colores por separado y con cada descomposición genera un fichero separada en formato TIFF. Estos ficheros ocupan mucho (entre 10 y 30 mb) y son de gran resolución. Una vez generados los imprimen sobre unas superficies especiales llamadas planchas e insertan cuatro por torre de impresión, uno por cada color.

Estas planchas hacen que sobre unas zonas del papel se imprima el color al que corresponde mientras que en otras no se imprima formando las imágenes, letras, formas, etc.

Ahora ya solo es cuestión del operario aplicar más o menos tinta para que cuando se mezcle con otros colores produzca el color del tono deseado.

Uno de los problemas reside en que las máquinas imprimen a gran velocidad y por tanto hasta que no se han impreso bastantes ejemplares el operario no es capaz de calibrar la máquina para conseguir el color adecuado.

Teniendo en cuenta esto, la idea sobre la que se sostiene el proyecto es la de analizar los mismos ficheros TIFF que se usan para imprimir planchas, extraer información sobre cada color y aplicar una apertura inicial para los tinteros con el objetivo de reducir el número de ejemplares desechados por falta de fidelidad con el original y que solo son usados para el ajuste inicial. Es decir, se pretende conseguir una aplicación de preajuste.

Hay que tener en cuenta que esta información extraída de los ficheros TIFF aunque sea muy precisa si se aplica directamente sobre el sistema puede no ser realista ya que cada tintero tiene su ciclo de vida y de desgaste de tal manera que para que de un cierto tono de color es posible que se tenga que aplicar a los valores analizados una transformación. Por ejemplo si el análisis de TIFF indica que se tiene que abrir un tintero al 50% y ese tintero está muy gastado es posible que para obtener un resultado similar con un tintero ideal al 50% este tintero desgastado se tenga que abrir un 70%.

Para realizar esta aplicación se han definido dos aplicaciones: un servidor (encargado de analizar los ficheros TIFF y aplicar apertura de tinteros a la máquina entre otras cosas) y un cliente que es la aplicación que realmente es el objetivo de este trabajo de fin de carrera.

El cliente será explicado más detalladamente a lo largo del documento, pero a grandes rasgos deberá servir para gestionar los parámetros del sistema de preajuste y visualizar el estado del mismo. El usuario final deberá ser el operario encargado de la impresión.

2.2 Requisitos funcionales

2.2.1 Usuarios, grupos y acceso al sistema

- Req. 1.1: El sistema deberá exigir al usuario una identificación para entrar al sistema consistente en un usuario y contraseña.
- Req. 1.2: Se deberá proporcionar unos menús y pantallas de gestión de usuarios y grupos donde se permita listar, eliminar, modificar y crear nuevos usuarios.
- Req. 1.3: El sistema solo permitirá la gestión de usuarios y grupos a los usuarios y grupos ya dados de alta en el sistema y que tengan permiso para acceder a este modo.
- Req. 1.4: Cualquier usuario, tenga los permisos de acceso que tenga, será capaz de visualizar sus datos de usuario y gestionar su contraseña de acceso al sistema.
- Req. 1.5: Un usuario solo puede pertenecer a un grupo. En el caso de que de un grupo se deshabilite los usuarios también se deshabilitarán y en el caso de que un grupo se

borre sus usuarios no se borrarán, tan solo se deshabilitarán hasta que sean reasignados a un grupo válido.

2.2.2 Parámetros del sistema

- Req. 2.1: El sistema deberá permitir a los usuarios cambiar ciertos valores que afectan a todo el sistema. Para ello se habilitará un una opción el en menú y una serie de ventanas que en general solo atacarán a base de datos y enviarán un mensaje al servidor para avisar que se han refrescado valores.
- Req. 2.2: Deberá permitir configurar opciones referentes al módulo de escaneo que se encuentra en la parte de servidor como las rutas de acceso o si debe guardar imágenes reescaladas.
- Req. 2.3: Se deberá habilitar una pantalla de donde el usuario pueda cambiar las opciones de parseo para la identificación de cada fichero. Deberá controlar que el fichero acabe en ".tif".
- Req. 2.4: Se habilitará una ventana para la configuración de los diferentes perfiles de ficheros TIFF que pueden llegar. Allí se pedirá información sobre las características físicas de cada fichero TIFF y se permitirá listar, crear, modificar y borrar perfiles.
- Req. 2.5: Existirá otra ventana donde podamos gestionar las publicaciones existentes y podamos crear, listar, modificar y borrarlas. Estas publicaciones servirán para la identificación de los nombres de los ficheros en el parseo y así poder asociarlas a un trabajo concreto en el futuro.
- Req. 2.6: Se deberá permitir al usuario cambiar diversas opciones como rutas de acceso a imágenes en miniatura, directorio de backups y de entrada por SMB.
- Req. 2.7: Se necesitará que el grupo al que pertenece el usuario tenga permisos específicos para poder acceder a esta gestión de parámetros del sistema, que en principio está pensada a los usuarios administradores.
- Req. 2.8: Cada vez que se cambien datos en base de datos la aplicación enviará un mensaje por el protocolo UDP al servidor para que refresque parámetros.

2.2.3 Gestión de la configuración de la rotativa

- Req. 3.1: El sistema deberá permitir listar, modificar, crear y eliminar elementos que pertenezcan a la estructura física de la rotativa. Estos elementos son "líneas", "plegadoras" y "torres".
- Req. 3.2: Cuando se añadan o modifiquen líneas se permitirá que el usuario asocie dichas líneas con plegadoras ya existentes previamente. Se debe permitir que una línea pueda no estar asociada a ninguna plegadora. Así en el caso de que el usuario cree una línea antes que una plegadora, puede guardar dicha línea, crear la plegadora y finalmente modificar la línea para asociarla. Si se elimina una línea se desasociará de las plegadoras asociadas antes de eliminarla.
- Req. 3.3: De la misma manera que con las "líneas", se debe de habilitar una serie de ventanas que permitan gestionar las "plegadoras". Estás permitirán la introducción de

datos propios de dichos elementos y además asociarlas a otros elementos del tipo "torre". El comportamiento en este sentido será similar al que ocurre con "líneas" y "plegadoras".

- Req. 3.4: Otra opción será la de gestión de "torres". En este caso no gestionará ninguna asociación y se gestionarán únicamente parámetros del elemento.
- Req. 3.5: Se debe permitir que una torre pueda ir asociada a varias plegadoras o a ninguna, que una plegadora pueda tener asociadas varias torres o ninguna y que pueda ir asociada a varias líneas o ninguna y que una línea pueda tener asociadas varias plegadoras o ninguna.
- Req. 3.6: Se necesitará que el grupo al que pertenece el usuario tenga permisos específicos para poder acceder a esta gestión de configuración de la rotativa, que en principio está pensada a los usuarios administradores.
- Req. 3.7: Cada vez que se cambien datos en base de datos la aplicación enviará un mensaje por el protocolo UDP al servidor para que refresque parámetros.

2.2.4 Gestión de trabajos y ficheros

- Req. 4.1: El sistema deberá permitir crear, modificar y listar trabajos.
- Req. 4.2: Un trabajo será la unión de varios elementos en un tiempo concreto. Por ejemplo, una cierta torre de impresión para una cierta publicación en una fecha concreta. Es por ello que el sistema deberá controlar que no se producen duplicidades en el tiempo pues no se puede imprimir una misma publicación, edición y subproducto dos veces en la misma fecha. Antes de guardar datos tiene que validar que se cumple está regla. No se podrá borrar un trabajo, solo deshabilitar. Entonces sí permitirá volver a crear otro igual, pero el desactivado nunca se podrá volver a activar.
- Req. 4.3: El sistema deberá, además de permitir cambiar las opciones de un trabajo, visualizar información de las páginas asociadas a cada publicación (mediante el parseo del nombre) así como el estado del porcentaje de páginas recibidas y procesadas por el servidor.
- Req. 4.4: El sistema deberá ser capaz de permitir que el usuario manualmente fuerce asociamientos de páginas que no han sido asociadas por el servidor automáticamente. En este caso se mandará al servidor un mensaje UDP indicándole los datos necesarios para que lo asocie.
- Req. 4.5: Se deberá permitir listar y ver estado de ficheros de entrada en el sistema y que han sido procesados por el servidor. Esto será solo informativo, no permitiéndose cambios de ningún tipo.
- Req. 4.6: Cuando se visualice información de un fichero TIFF, si existe, se mostrará la miniatura de dicho fichero.
- Req. 4.7: Cuando se visualice información de un fichero TIFF, si existe miniatura y fichero TIFF original se permitirá descargarlo.
- Req. 4.8: Cuando se visualice información de un fichero TIFF, si este está asociado a algún trabajo se debe mostrar información del mismo y crear un acceso a el mismo. De lo contrario no se mostrará dicha información.

- Req. 4.9: Se debe permitir que el usuario introduzca en el sistema un fichero TIFF propio. Para ello se habilitará una ventana de selección de fichero. Esto lo copiará por SMB a una ruta donde el servidor lea ficheros.
- Req. 4.10: Tanto para el listado de ficheros como el de trabajos se proporcionara una serie de filtros por parámetros para que sea más fácil la búsqueda.
- Req. 4.11: Tanto el listado de ficheros como el trabajos se actualizarán periódicamente para mantener el estado lo más síncrono y actualizado posible. Para ello empleará el parámetro de tiempo de refresco del servidor, que es el tiempo que tarda en volver a verificar si hay ficheros nuevos en la carpeta de entrada, como tiempo de refresco. El refresco se podrá forzar de manera manual.
- Req. 4.11: Se necesitará que el grupo al que pertenece el usuario tenga permisos específicos para poder acceder a esta gestión de trabajos.

2.2.5 Gestión de curvas

- Req. 5.1: El sistema deberá permitir crear, modificar, eliminar y listar perfiles de curvas.
- Req. 5.2: El usuario podrá crear tantos perfiles como desee (por ejemplo, uno para cada fabricante de tintas, para cada tipo de papel, etc.). Para el sistema el motivo deberá dar igual, será el usuario el que tenga que ponerle un nombre identificativo esclarecedor.
- Req. 5.3: Cada perfil tendrá asociada 3 curvas comunes: la inicial, la de tolerancia superior y la de tolerancia inferior. El usuario podrá gestionar estos valores para ajustarlas al perfil.
- Req. 5.4: Cada perfil mantiene una curva distinta (a no ser que coincida por casualidad) para cada tintero de cada torre. Por lo tanto se debe de poder realizar una búsqueda filtrada que permita localizar curvas y ver su forma.
- Req. 5.5: Se debe permitir al usuario modificar los valores de las curvas para cada tintero. Se habilitará una pantalla de edición de la curva para una cierta curva seleccionada en la búsqueda.

2.2.6 Idiomas

- Req. 6.1: Habrá varios idiomas disponibles y cada usuario guardará el suyo preferido en su perfil.
- Req 6.2: El idioma por defecto será el inglés.

2.3 Requisitos no funcionales

- Req. 7.1: Se debe de minimizar el número de opciones y caminos para realizar una tarea al mínimo.

- Req. 7.2: El sistema debe de ser intuitivo de tal forma que una persona habituada al entorno industrial de imprenta pero no a la aplicación sepa orientarse sin necesidad de ayuda o muy poca.
- Req. 7.3: Flexibilidad. Se requiere que ante un problema se puedan buscar alternativas. Para ello se requiere que se puedan parametrizar el máximo número de elementos posibles

2.4 Arquitectura del sistema

El sistema constará de dos partes que son cliente y servidor.

Los requerimientos por parte del servidor quedan fuera de este proyecto pero se ha establecido que sea un servicio programado en C++ para Linux con posibilidad de socket y base de datos.

Los requerimientos por parte de la interfaz de usuario son que sea desarrollado usando tecnología Microsoft .NET , y para la base de datos SQL Server 2008 R2.

2.4.1 Arquitectura del Software

Para llevar a cabo la interfaz de usuario se ha decidido que se realizará en *Visual Studio 2010 Professional* usando *.NET Framework 4.0* y lenguaje *C#*. Para la base de datos se usará *SQL Server 2008 R2 Developer*. Para concretar un poco más el modo y los requerimientos de cada tarea se propone el siguiente desglose y el software a usar:

- Gestionar publicaciones a imprimir y selección de torres: Esta sección será un conjunto de menús y pantallas cuyos valores serán tomados y almacenados en base de datos. Se usará para el diseño de la capa de presentación la tecnología *WPF*.

- Ver el estado del proceso de escaneo: Esta sección abarcará una o varias pantallas que indicaran el proceso de escaneado de los ficheros por parte del servidor, para ello leerá de base de datos información y la mostrará. Al igual que el apartado anterior se usará para el diseño de la capa de presentación la tecnología *WPF*.

- Asignar de forma manual un fichero TIFF a una publicación: Otra pantalla cuyo desarrollo no supone diferencia con las anteriores salvo en la lógica, por lo tanto usaremos la misma tecnología.

- Ver las curvas de los tinteros gráficamente: Este apartado contendrá una pantalla para selección de curva y se accederá a otra pantalla en la que se mostrará una o varias gráficas. Para las gráficas se usara el componente *WPF Toolkit* que proporciona un

conjunto de utilidades y librerías para la generación de las mismas en entorno *.NET Framework 4.0* o superior.

- Parámetros del sistema: Se realizarán varias pantallas y menús al igual que el resto de ventanas en *C#* y *WPF*.

- Acceso mediante usuarios con distintos perfiles: Al iniciar la aplicación se pedirá usuario y contraseña que se verificará en base de datos y según el perfil se permitirá acceso a unas cosas u a otras. Se usará la librería de cifrado y resumen que ofrece *.NET Framework* y así se guardará solo el resumen de la contraseña mediante el algoritmo MD5.

- Informes del sistema: Para generar los informes del sistema se usará *Microsoft ReportViewer* para mostrar los informes generando informes *RDLC*.

- Gestión multi-idioma (castellano e inglés): Se usará un fichero de recursos para cada idioma disponible. Se implementará mediante *WPF Localize Extension*.

Para hacer funcionar esta aplicación se requerirá un equipo con *Windows XP SP4, Windows Vista* o *Windows 7* con el paquete *.NET Framework 4.0 redistributable* instalado.

Los requisitos de software para la aplicación servidor será un equipo de tipo servidor con Ubuntu 10.10, las librerías GCC, la librería LIBTIFF y la librería LIBPNG instaladas.

Los requisitos de software para la SQL Server 2008 será de Windows XP SP4 o superior. Puesto que es preferible que la base de datos este centralizada con el servidor, que tendrá sistema operativo Linux y por tanto no será posible instalar SQL Server, se ha optado por usar una máquina virtual usando la aplicación Oracle VirtualBox con Windows XP instalado como sistema operativo huésped. La solución podría resultar ineficiente si fueran importantes los tiempos de acceso a la base de datos, pero en nuestro caso no hay consultas con cálculos pesados, donde más cuello de botella se puede formar es en el acceso a disco y eso va a ser prácticamente igual con máquina virtual que sin máquina virtual.

2.4.2 Arquitectura del Hardware

- Para la aplicación GUI desarrollada en .NET será necesario un equipo mínimo de Pentium IV o equivalente. El espacio necesario para alojar la aplicación será de 10 a 20 Mb. Será necesario también que la memoria del sistema sea como mínimo 256 MB.
  Por último se requiere que tenga conexión Ethernet para comunicarse con el resto de sistemas.

- Para la aplicación servidor se requerirá de un equipo Core2Duo o superior ya que la carga en el proceso de ficheros será alta. Se requerirá mínimo 2 Gb de RAM para poder manejar

el sistema operativo, la aplicación servidor y la máquina virtual y 2 Tb de disco duro para poder guardar histórico de fichero TIFF.
Requerirá conexión Ethernet.

2.5 Casos de uso

El diagrama general de casos de uso es el siguiente:

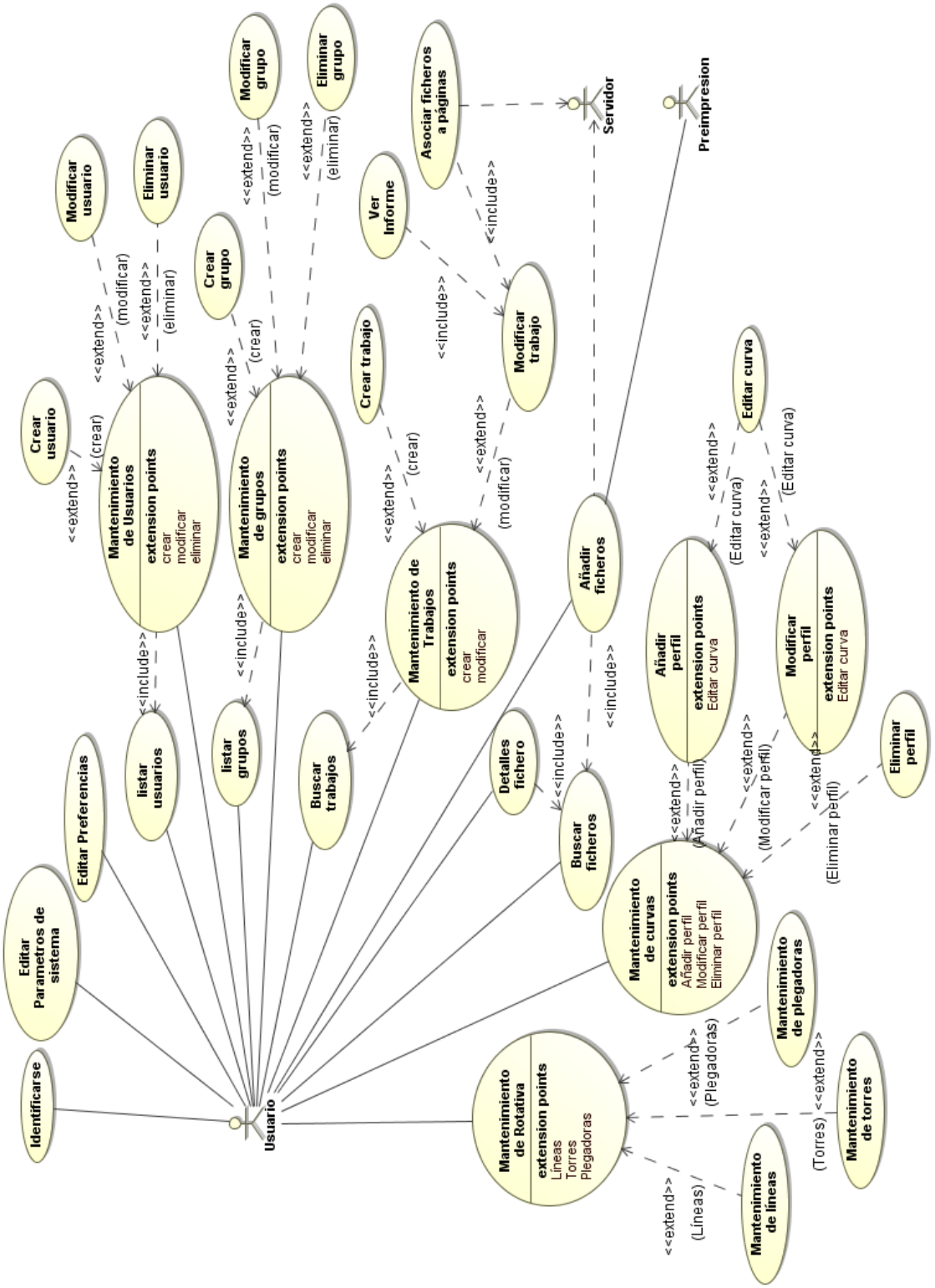

Figura 2.5.1 – Casos de uso

Los actores son:

- **Usuario**: Puede ser el jefe de operarios, un operario, etc.
- **Preimpresión**: Es el grupo de personas y sistemas que convierte los ficheros PDF de los que se compone una publicación a ficheros TIFF en los 4 colores CMYK.
- Servidor: La aplicación servidor que se comunica con la GUI de preajuste.

Para facilitar su comprensión se subdividirá en diagramas que abarquen zonas más concretas.

2.5.1 Entrada al sistema, preferencias y parámetros del sistema

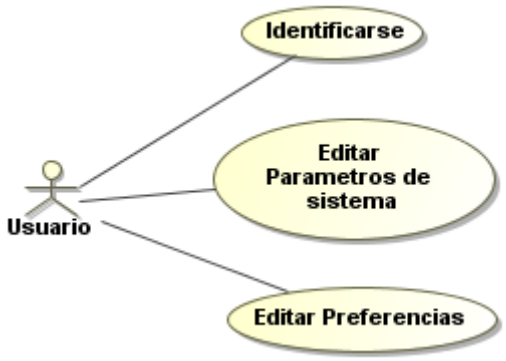

Figura 2.5.1.1 – Subdiagrama 1

El usuario podrá Identificarse, editar parámetros del sistema editar preferencias.
Corresponden con los apartados 2.11.1, 2.11.3 y 2.11.4 respectivamente.

2.5.2 Gestión de usuarios

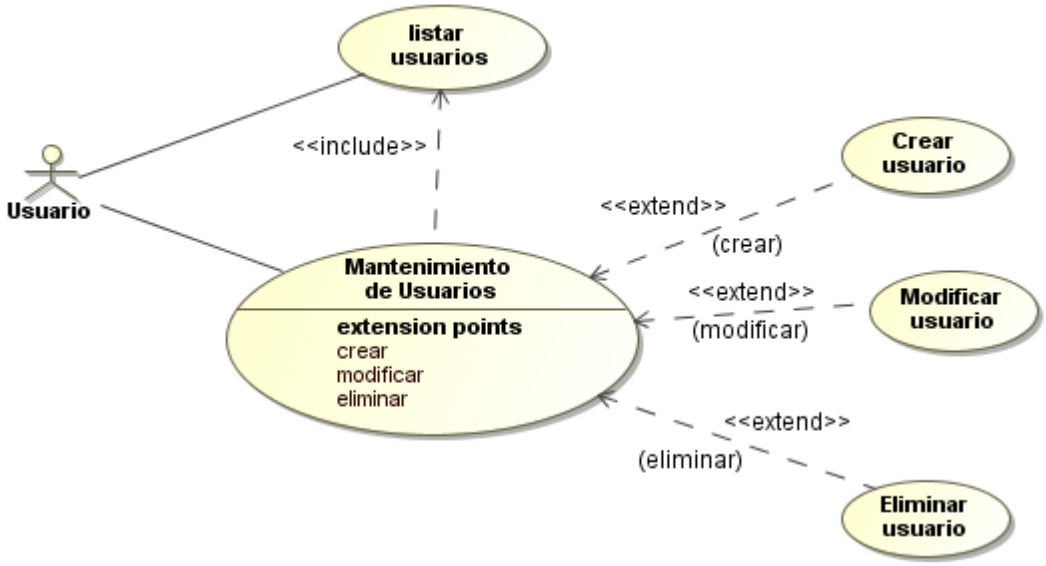

Figura 2.5.2.1 – Subdiagrama 2

El usuario podrá listar y mantener los usuarios. Para efectuar el mantenimiento que permite crear, modificar y eliminar se tendrá que haber accedido desde un listado de usuarios previo. Corresponde con el apartado 2.11.12.

2.5.3 Gestión de grupos

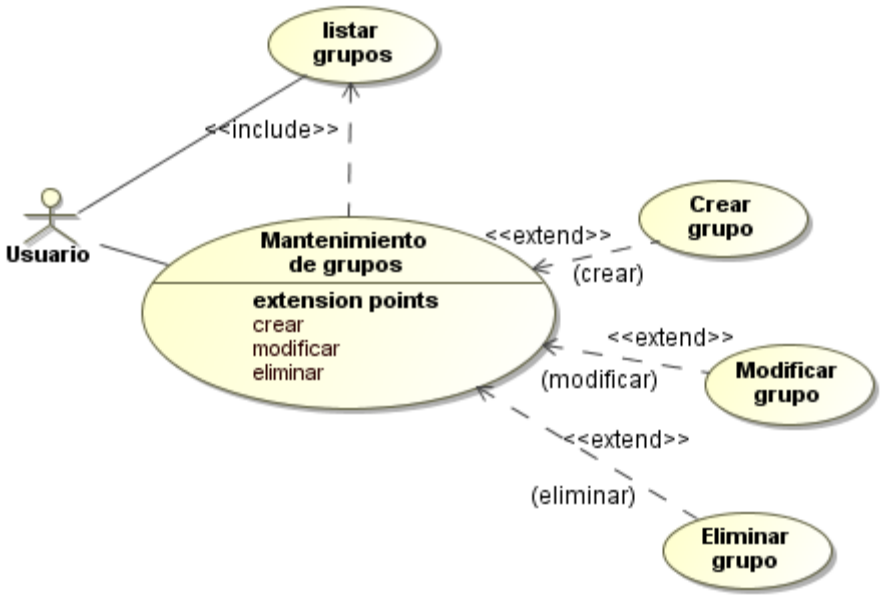

Figura 2.5.3.1 – Subdiagrama 3

El usuario podrá listar y mantener los grupos (y sus permisos). Para efectuar el mantenimiento que permite crear, modificar y eliminar se tendrá que haber accedido desde un listado de grupos previo. Corresponde con el apartado 2.11.13.

2.5.4 Gestión de trabajos y ficheros

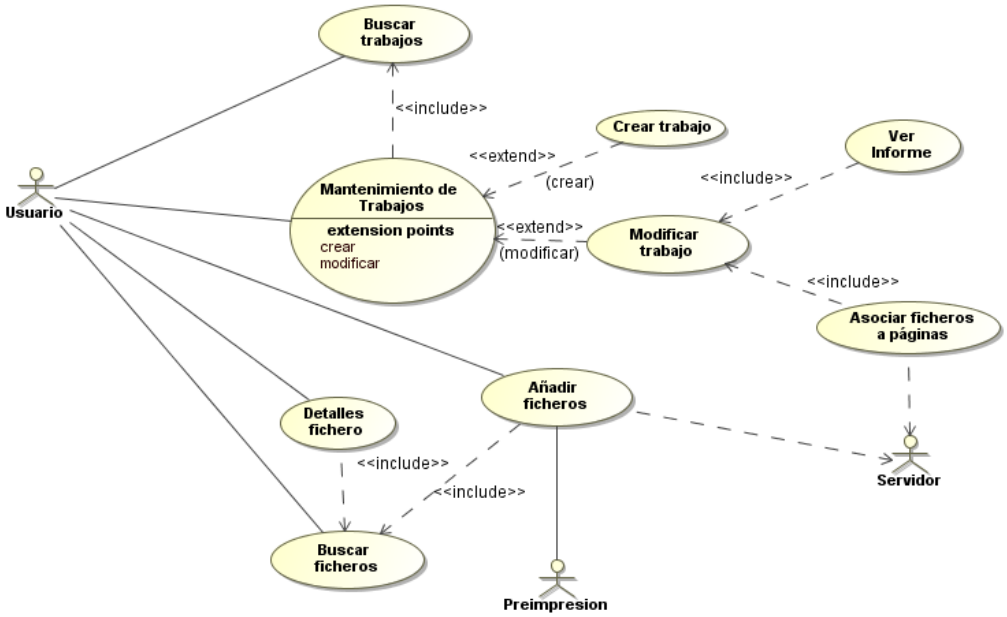

Figura 2.5.4.1 – Subdiagrama 4

El usuario podrá listar mantener trabajos. Para efectuar el mantenimiento es necesario haber accedido a la búsqueda de trabajos. Solo se permite ver informe desde la ventana de modificación de trabajo. Así como la de asociar ficheros a páginas. El servidor será informado y será quien lleve a cabo a esta tarea.

También el usuario podrá ver detalles de un fichero previa búsqueda y listado y enviar al servidor un fichero para que este lo añada. Preimpresión también tiene la capacidad de enviar ficheros al servidor. Corresponde con los apartados 2.11.5, 2.11.6, 2.11.7, 2.11.8.

2.5.5 Gestión de curvas

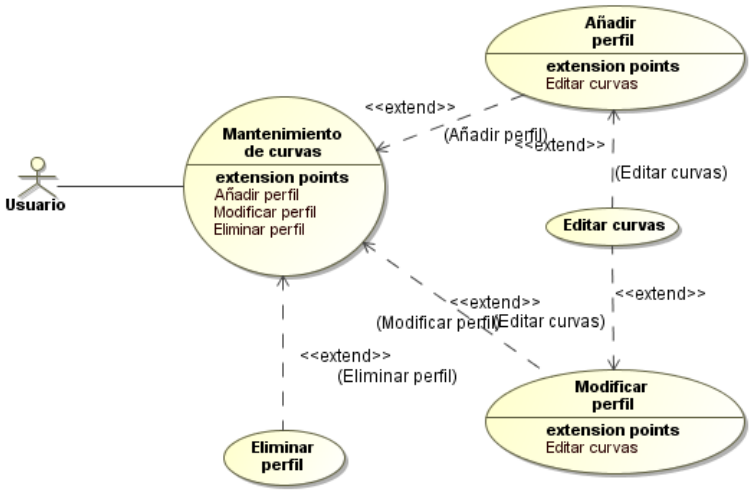

Figura 2.5.5.1 – Subdiagrama 5

Un usuario podrá añadir, modificar y eliminar perfiles de curvas que hacen las veces de contenedores de curvas. Por otro lado toda curva va asociada a un perfil. Por lo tanto para que editar curvas el usuario tiene que añadir o modificar el perfil de curvas correspondiente. Corresponde con el apartado 2.11.10

2.5.6 Gestión de Rotativa

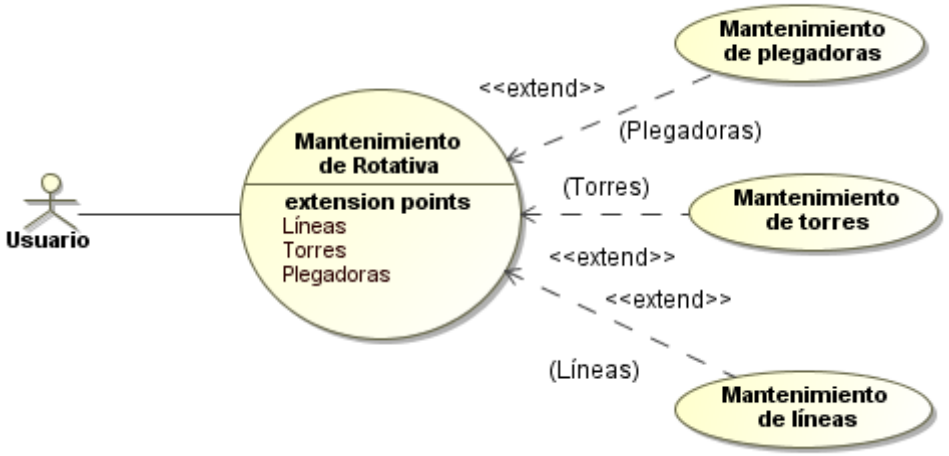

Figura 2.5.6.1 – Subdiagrama 6

El usuario podrá gestionar la rotativa y sus componentes: líneas, plegadoras y torres. Una línea tendrá asociadas múltiples plegadoras y torres y una plegadora tendrá asociada múltiples torres. Desde el mantenimiento de cada elemento se podrá modificar está asociación. Corresponde con el apartado 2.11.11

2.6 Base de datos

Se ha construido el diagrama E-R usando el método “Patas de gallo”:

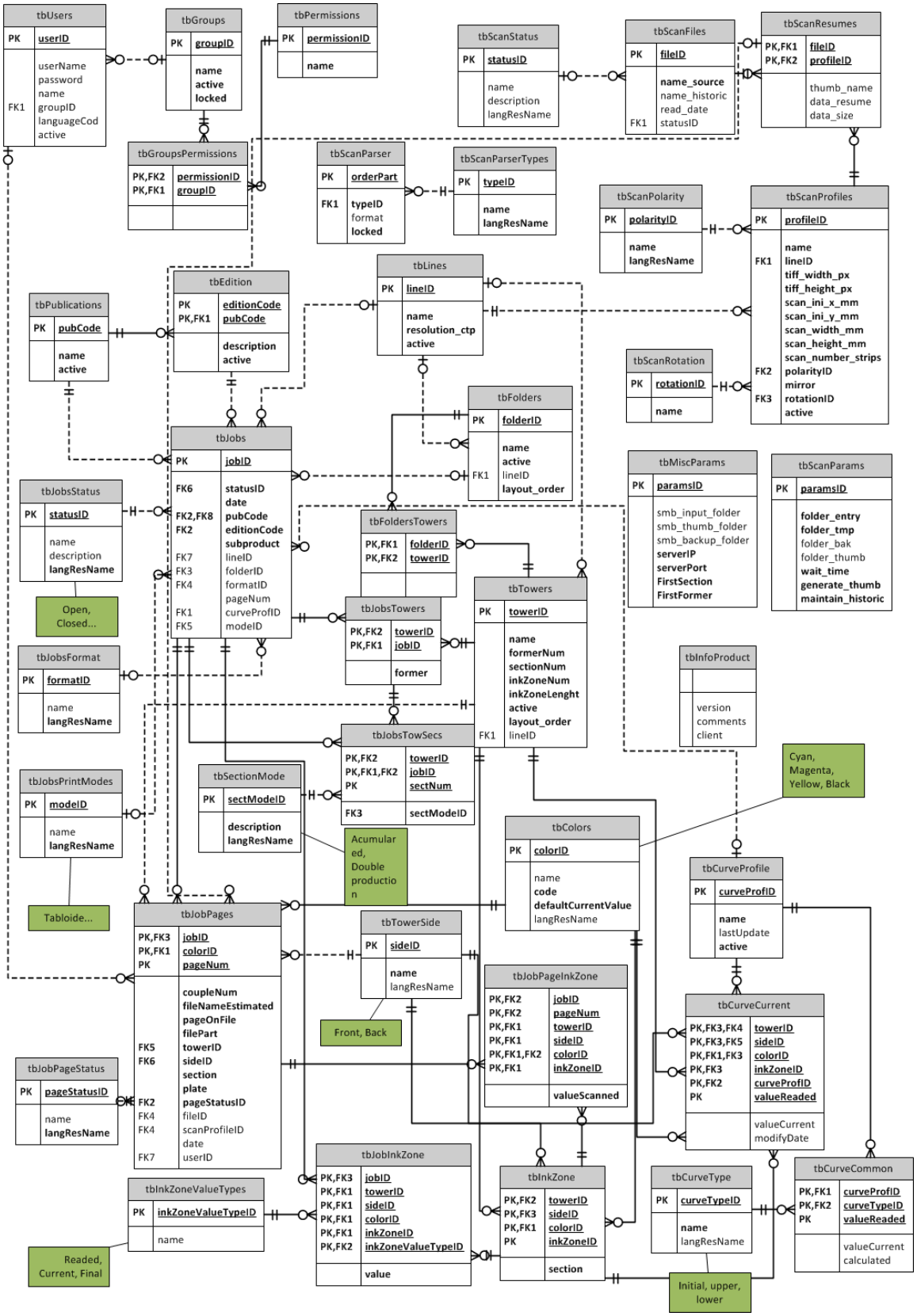

Figura 2.6.1. Diagrama E-R

El diseño de la base de datos sería el siguiente:

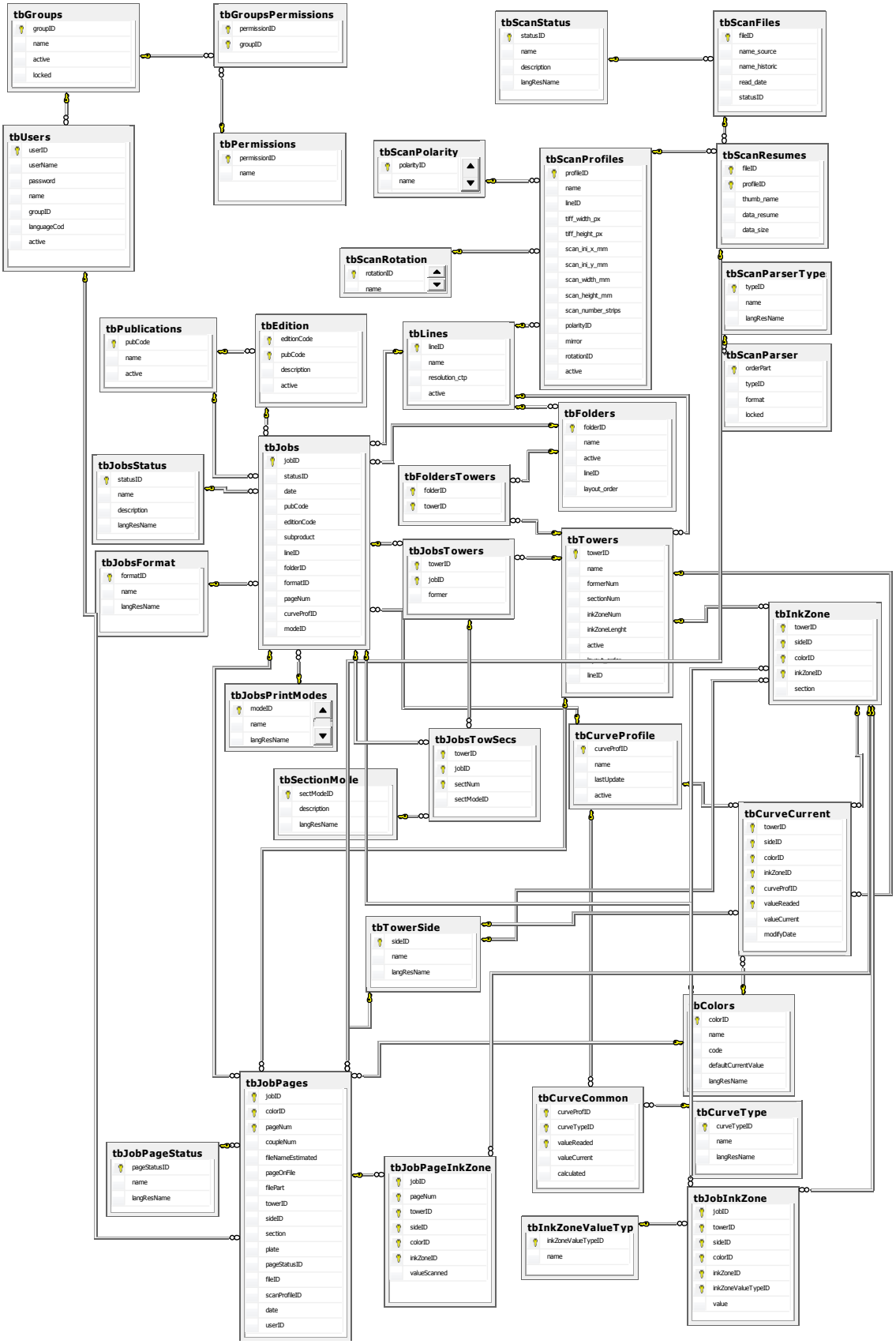

Figura 2.6.2. Diseño base de datos

2.7 Arquitectura de la aplicación

La arquitectura del sistema será la siguiente:

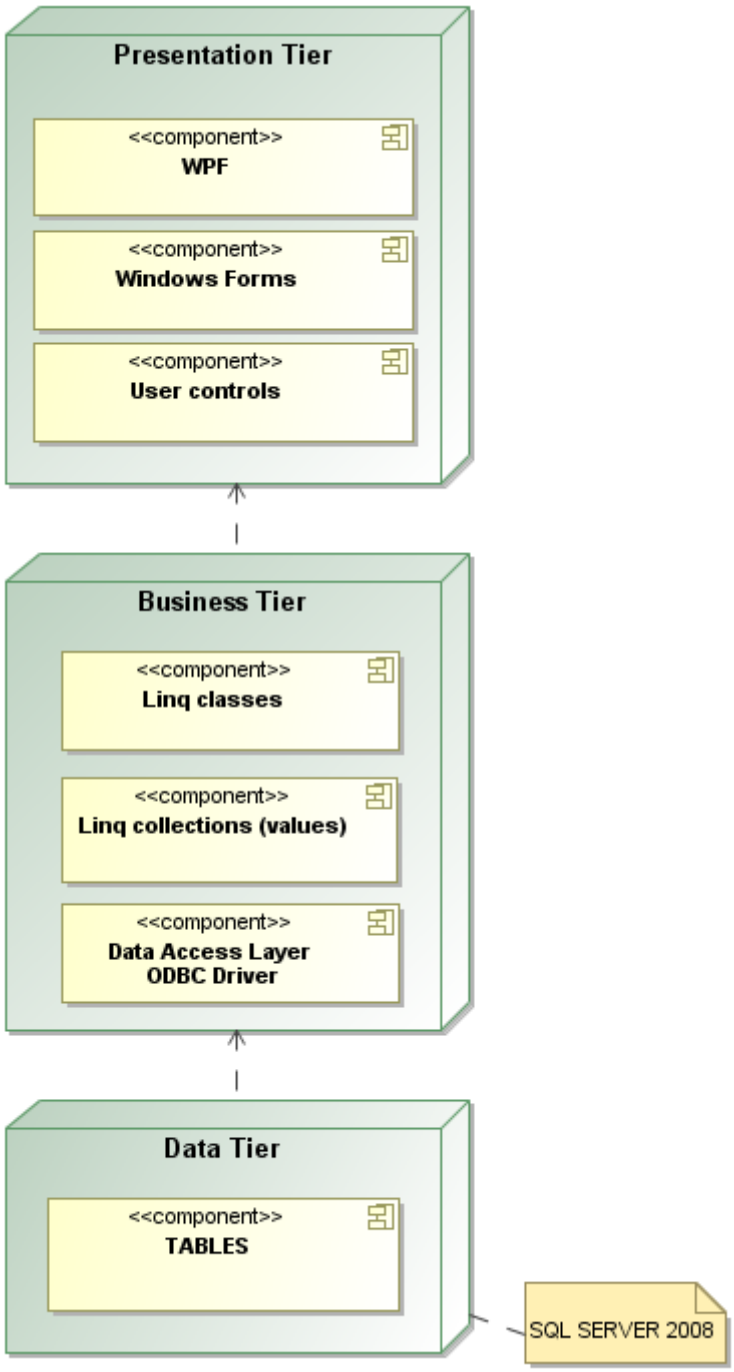

Figura 2.7.1. Diagrama de arquitectura de la aplicación

La capa de presentación estará formada por los controles *WPF*, los controles creados para la aplicación o librerías externas utilizadas y los componentes *Windows Forms* (utilizados para el *Report Viewer*). La capa de negocios estará generada principalmente por *Linq to SQL*. Por último la capa de datos serán las tablas determinadas en servidor *SQL Server 2008*. No contendrá otros componentes como procedimientos almacenados en este caso.

2.8 Arquitectura general del sistema

La arquitectura del sistema será la siguiente:

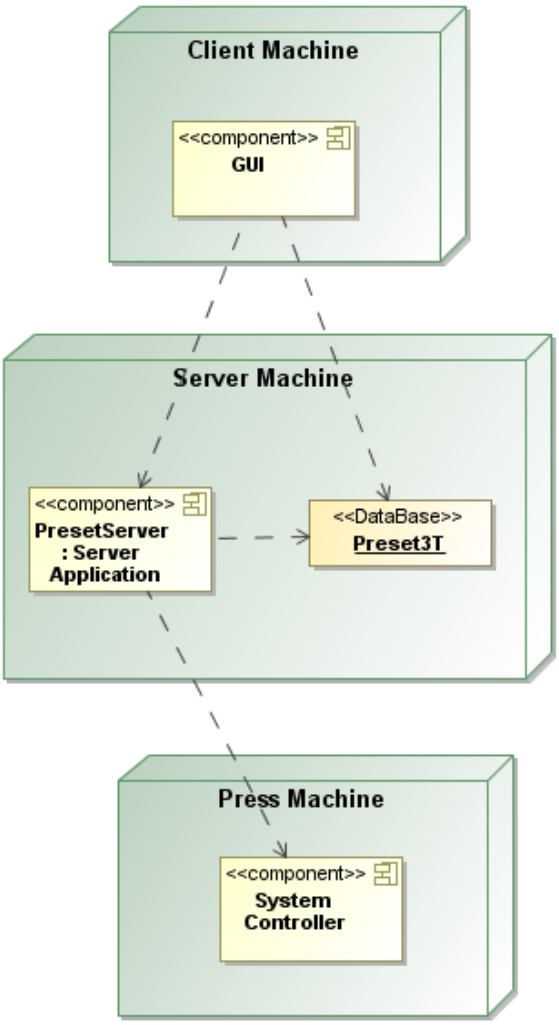

Figura 2.8.1. Diagrama de arquitectura

La *GUI* se comunicará con la aplicación servidor y también conectará con base de datos para escritura y lectura. Así mismo la aplicación servidor también conectará con base de datos para escritura y lectura y se comunicará con la rotativa para mandar los ajustes finales para la apertura de tinteros.
Para la *GUI* el elemento *Press Machine* será transparente pues de la comunicación con este se encarga en exclusiva el servidor mediante un protocolo *Ethernet* definido con el fabricante de la rotativa o mediante órdenes directas a motores por medio de componentes electrónicos *PLC*.

La aplicación servidor y la base de datos en principio está pensado que se ubiquen en la misma máquina pero no es estrictamente necesario: puede estar en la máquina de la *GUI* o incluso en una tercera máquina.

2.9 Diagrama de clases

Para la implementación se usará el modo *Linq to SQL* por lo que el diagrama de clases coincidirá exactamente con el de base de datos:

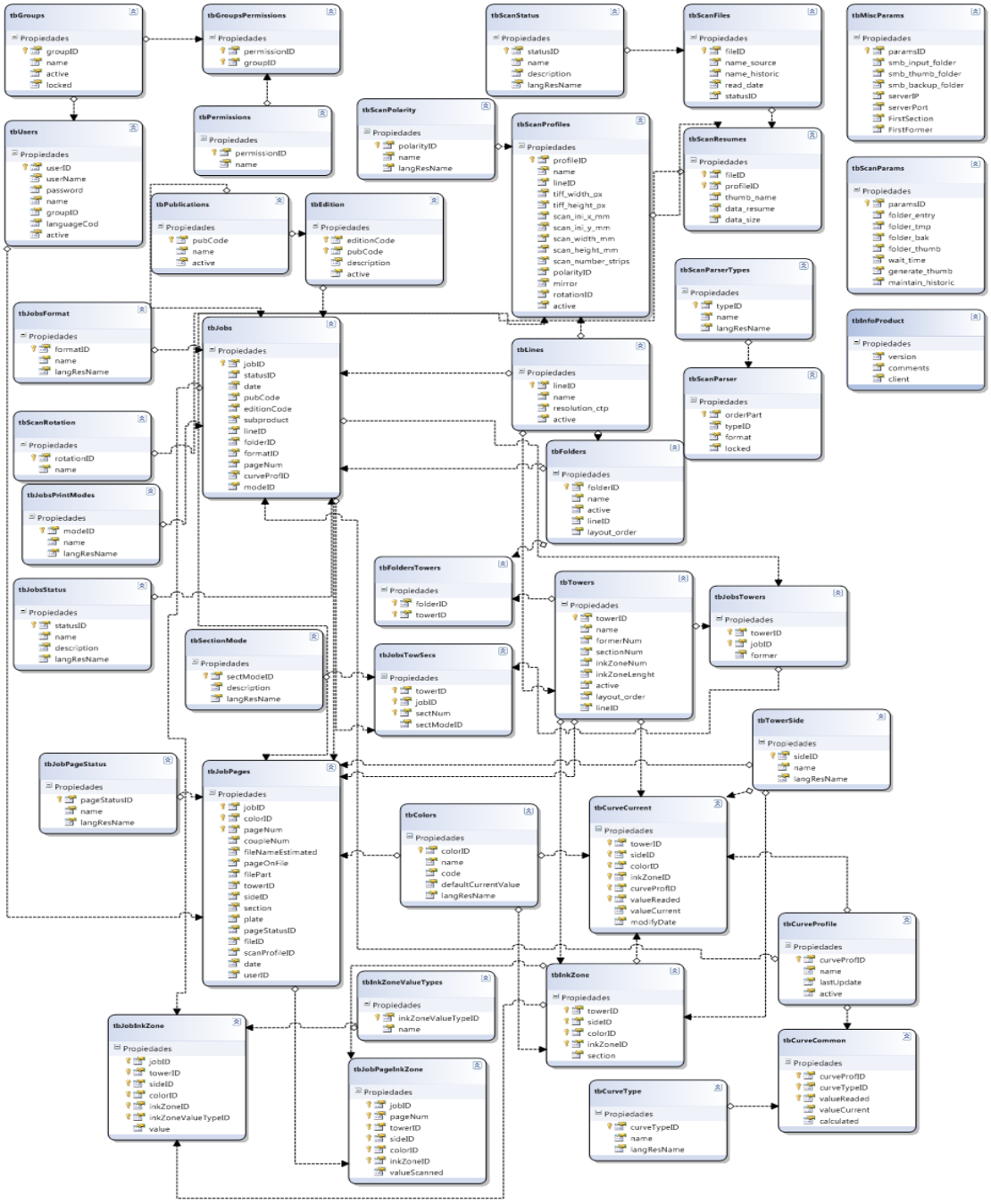

Figura 2.9.1. Diagrama de clases

2.10 Comunicación con servidor

Es necesario que la aplicación GUI que se detalla en este documento se comunique con la aplicación servidor. Para ello se ha optado por el envío de telegramas usando el protocolo no orientado a conexión UDP.

La encapsulación se realizará siguiendo la siguiente estructura:

Byte	Descripción
0x00	Tipo de mensaje
0x01	
0x02	
0x03	
0x04…0x0XX	Datos del mensaje

Tabla 2.10.1 – Estructura de mensaje

Tipo de mensaje: Identificador del tipo de mensaje que se está enviando.
Datos del mensaje: Información referente al mensaje que se está enviando.

Los mensajes resultantes y la estructura de los mismos se detallan en los siguientes puntos.

2.10.1 Comunicación GUI → Servidor

La siguiente tabla muestra un resumen de los mensajes necesarios:

Tipo de mensaje	ID de tipo de mensaje
Petición de actualización de parámetros	1
Refresco de trabajo	2
Asociar fichero a trabajo	3

Tabla 2.10.1.1 – Mensajes posibles desde la GUI hacia la aplicación servidor

Pasamos a detallar dichos mensajes a continuación:

Petición de actualización de parámetros

Byte	Valor	Descripción
0x00	1	Tipo de mensaje
0x01		
0x02		
0x03		

Tabla 2.10.1.2 – Mensajes de petición de actualización de parámetros

Refresco de trabajo

Byte	Valor	Descripción
0x00	2	Tipo de mensaje
0x01		
0x02		
0x03		
0x04	0xXXXXXXXX	- Id del trabajo
0x05		
0x06		
0x07		

Tabla 2.10.1.3 – Mensajes de petición de refresco de trabajo

Asociar fichero a trabajo

Byte	Valor	Descripción
0x00	3	Tipo de mensaje
0x01		
0x02		
0x03		
0x04	0xXXXXXXXX	Id del trabajo
0x05		
0x06		
0x07		
0x08	0xXXXXXXXX	Página identificativa del fichero
0x09		
0x0A		
0x0B		
0x0C	0xXXXXXXXX	ID del color
0x0D		
0x0E		
0x0F		
0x10	0xXXXXXXXX	ID del fichero
0x11		
0x12		

0x13		
0x14	0xXXXXXXXX	ID del usuario
0x15		
0x16		
0x17		

Tabla 2.10.1.4 – Mensajes de petición de asociación de fichero a trabajo

2.10.2 Comunicación Servidor → GUI

Actualmente sin necesidad de recepción de mensajes por parte de la aplicación GUI, aunque queda abierta la posibilidad de que en el futuro se amplíe la funcionalidad de la aplicación y sea necesaria.

2.11 Diseño de la interfaz de usuario

2.11.1 Inicio de sesión

Lo primero que veremos nada más arrancar la aplicación será esta pantalla compuesta por dos pestañas donde introduciremos nuestros datos de usuario.

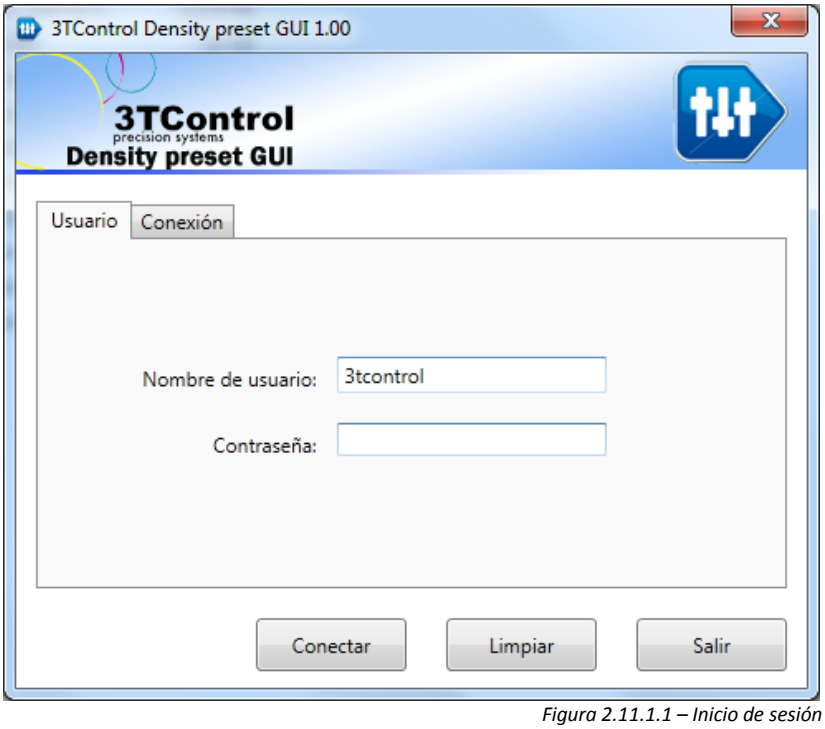

Figura 2.11.1.1 – Inicio de sesión

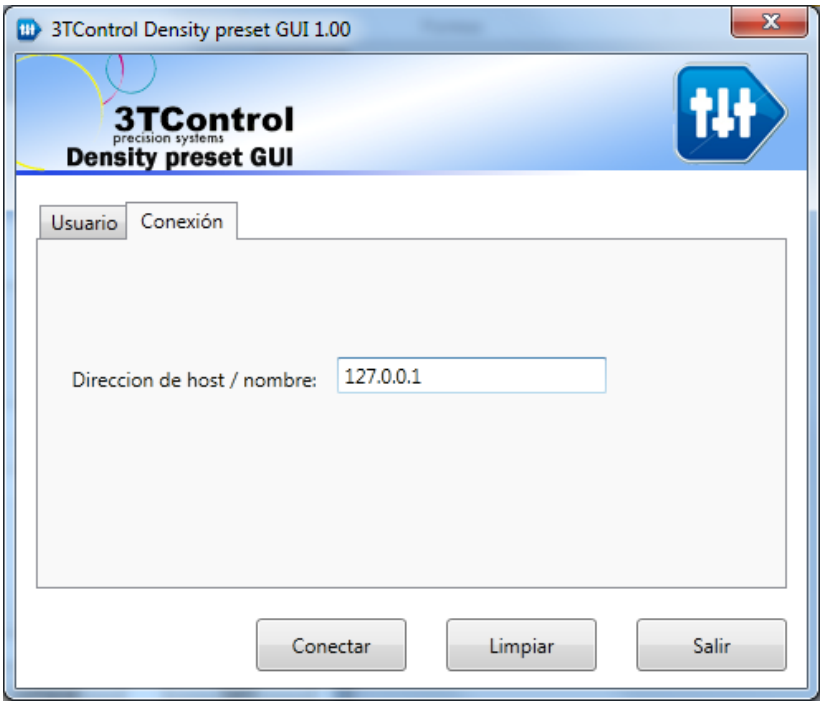

Figura 2.11.1.2– Inicio de sesión

2.11.2 Pantalla principal y menús

Una vez nos hallamos registrado correctamente en el sistema accederemos a la pantalla principal:

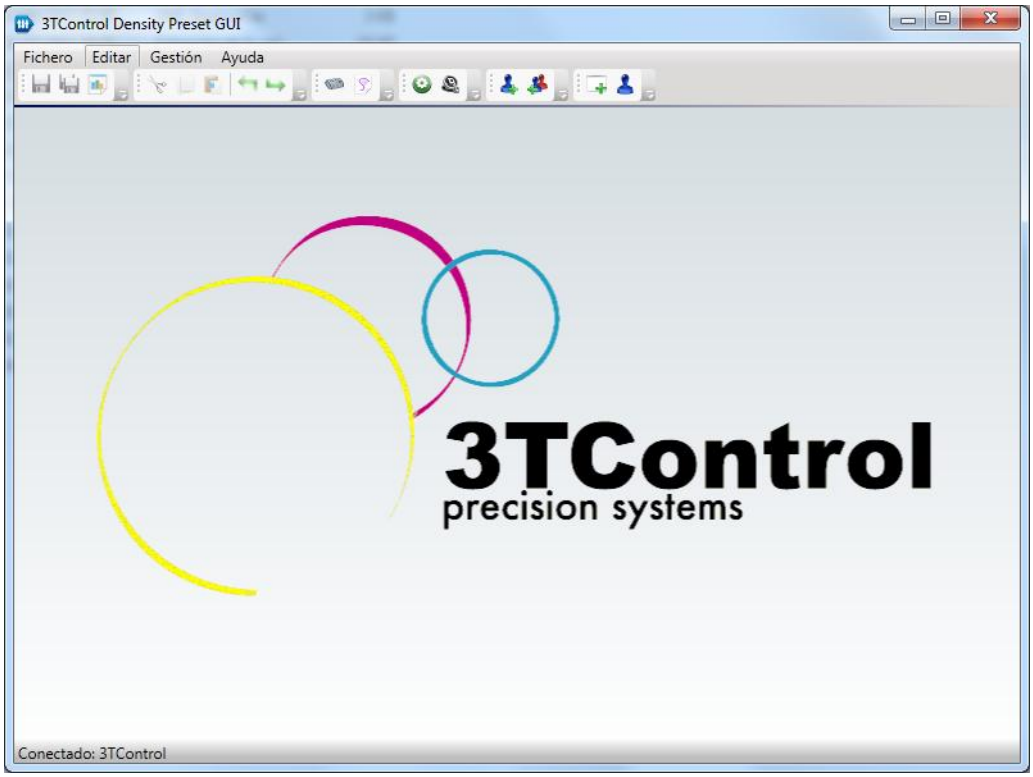

Figura 2.11.2.1 – Pantalla principal

A ella se le irán añadiendo pestañas para cada operación.

El menú superior consta de las siguientes opciones que estarán habilitadas o no dependiendo de la pestaña y la situación de la misma.

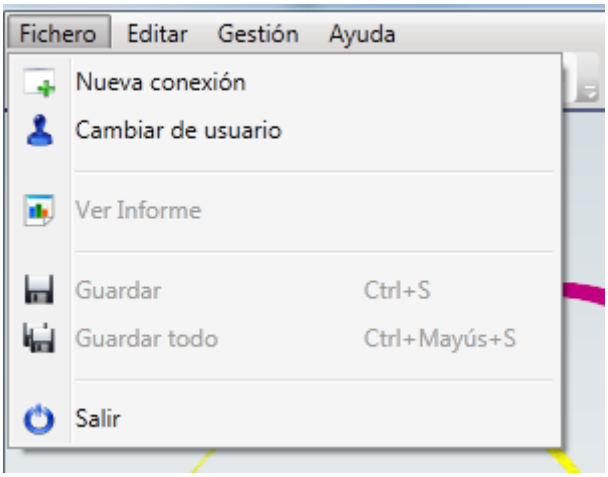

Figura 2.11.2.2 – Pantalla principal, menú "Fichero"

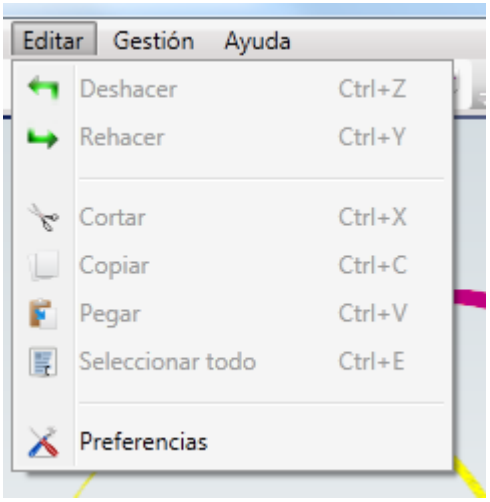

Figura 2.11.2.3 – Pantalla principal, menú "Editar"

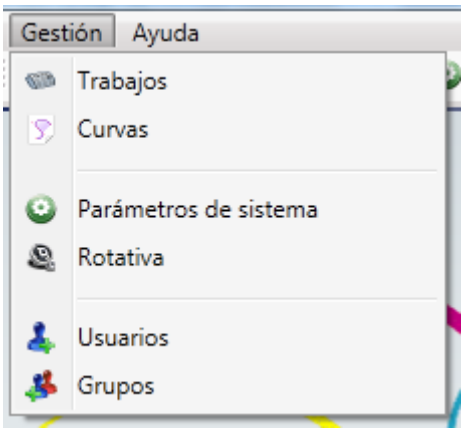

Figura 2.11.2.4 – Pantalla principal, menú "Gestión"

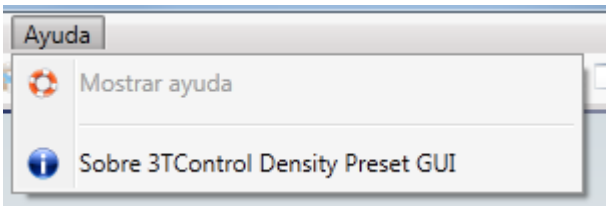

Figura 2.11.2.5 – Pantalla principal, menú "Ayuda"

2.11.3 Preferencias

Esta ventana es en la que los usuarios podrán gestionar su contraseña y su idioma preferido por defecto.

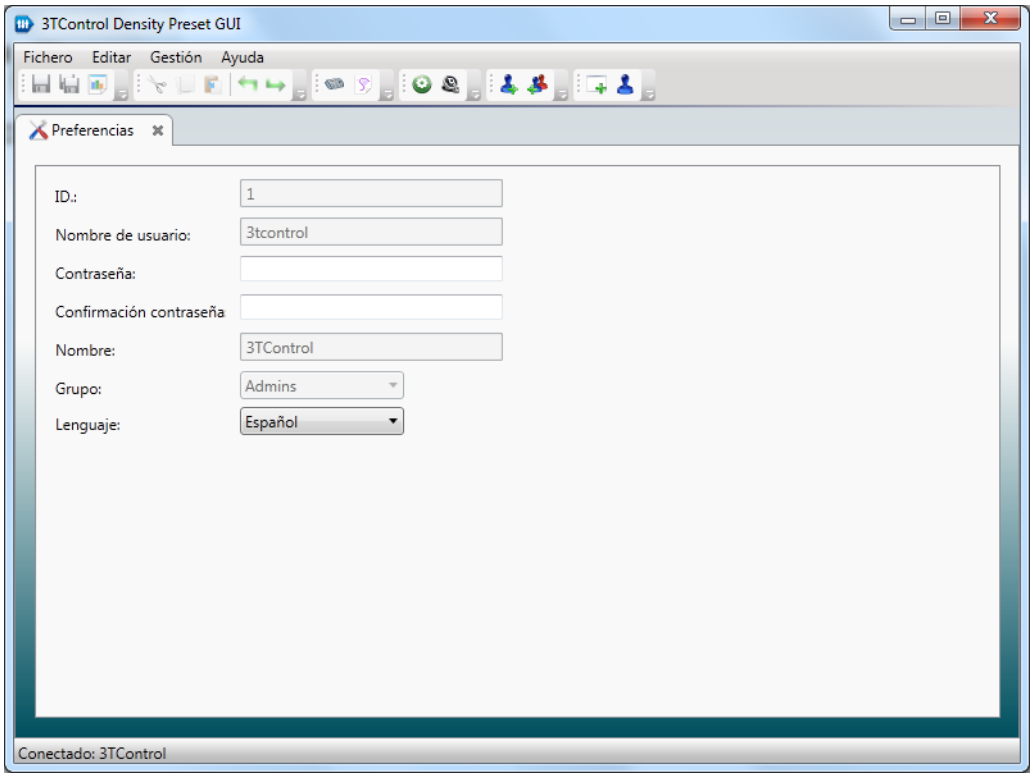

Figura 2.11.3.1– Pestaña de preferencias

2.11.4 Parámetros del sistema

Esta ventana consta de varías pestañas según la finalidad de los parámetros. La primera pestaña sirve para ajustar los parámetros para "parsear" el nombre de los ficheros, así cuando se cree un trabajo podremos asociar el fichero entrante automáticamente al trabajo creado sin tener que hacerlo manualmente

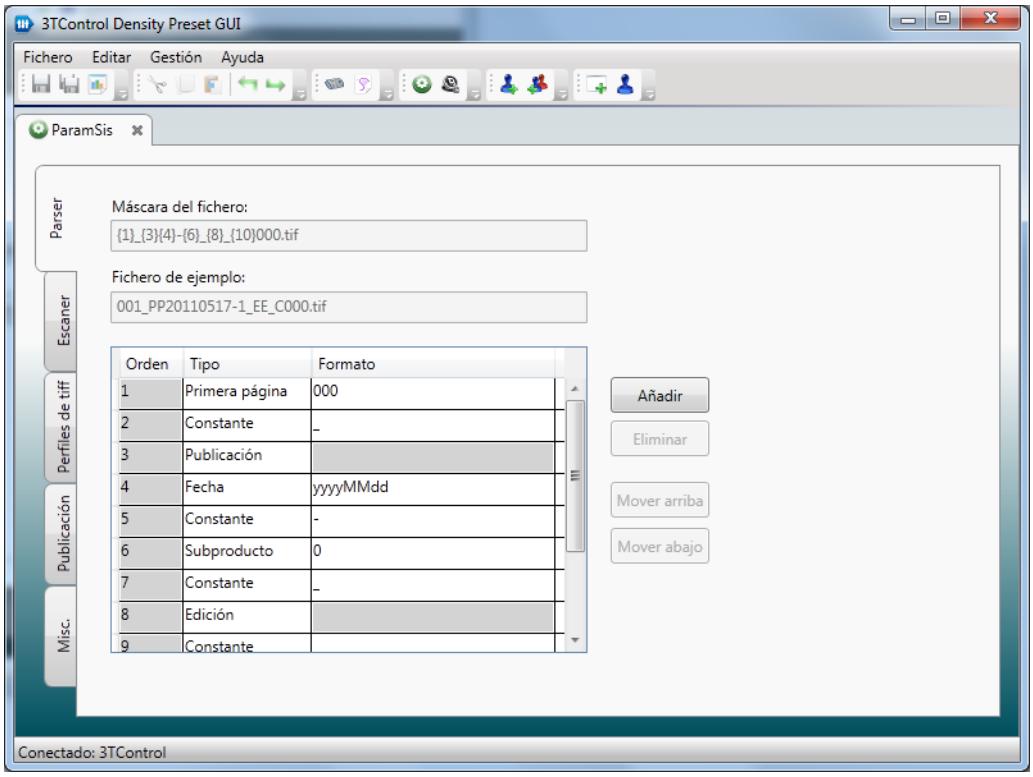

Figura 2.11.4.1 – Pestaña parámetros del sistema, opciones de Parser

La siguiente pestaña corresponde a parámetros del escaneo como directorios de entrada, temporales, de "backup" de ficheros TIFF, de miniaturas. Tiene que ser rutas locales al servidor, por ejemplo "\etc\input". El tiempo de espera es el intervalo de tiempo que tardará en verificar si se han incluido nuevos ficheros TIFF en la carpeta de entrada.

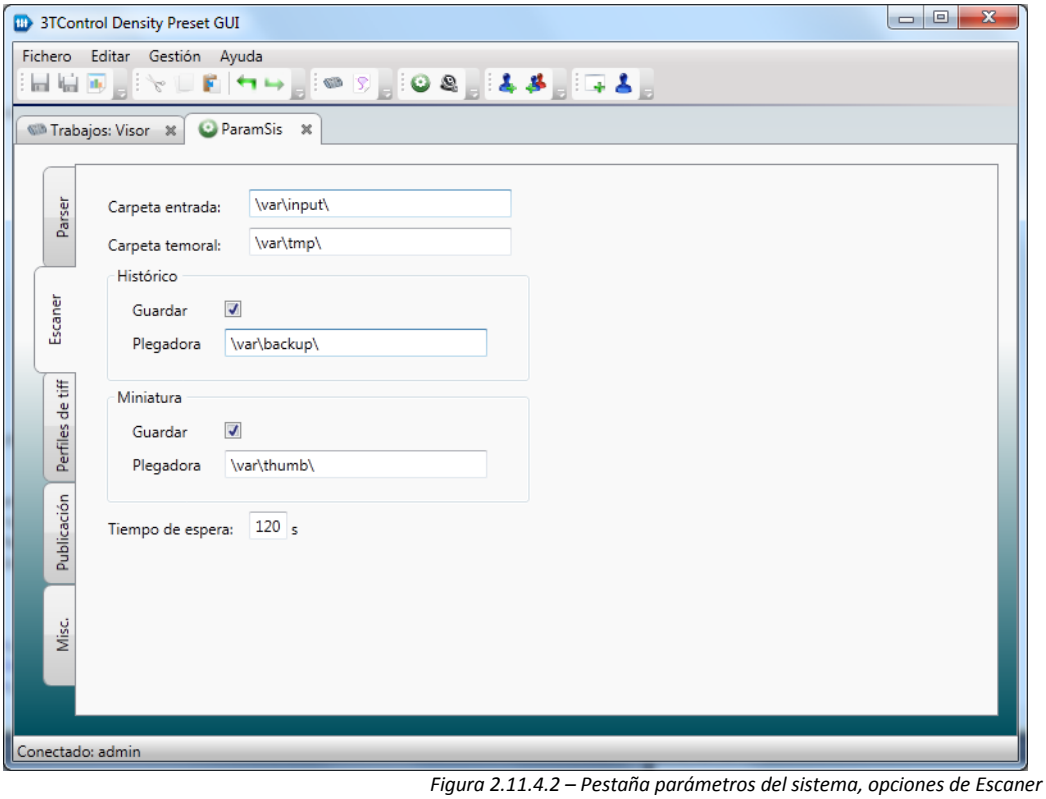

Figura 2.11.4.2 – Pestaña parámetros del sistema, opciones de Escaner

La siguiente pantalla muestra el listado de los perfiles de ficheros TIFF disponibles.

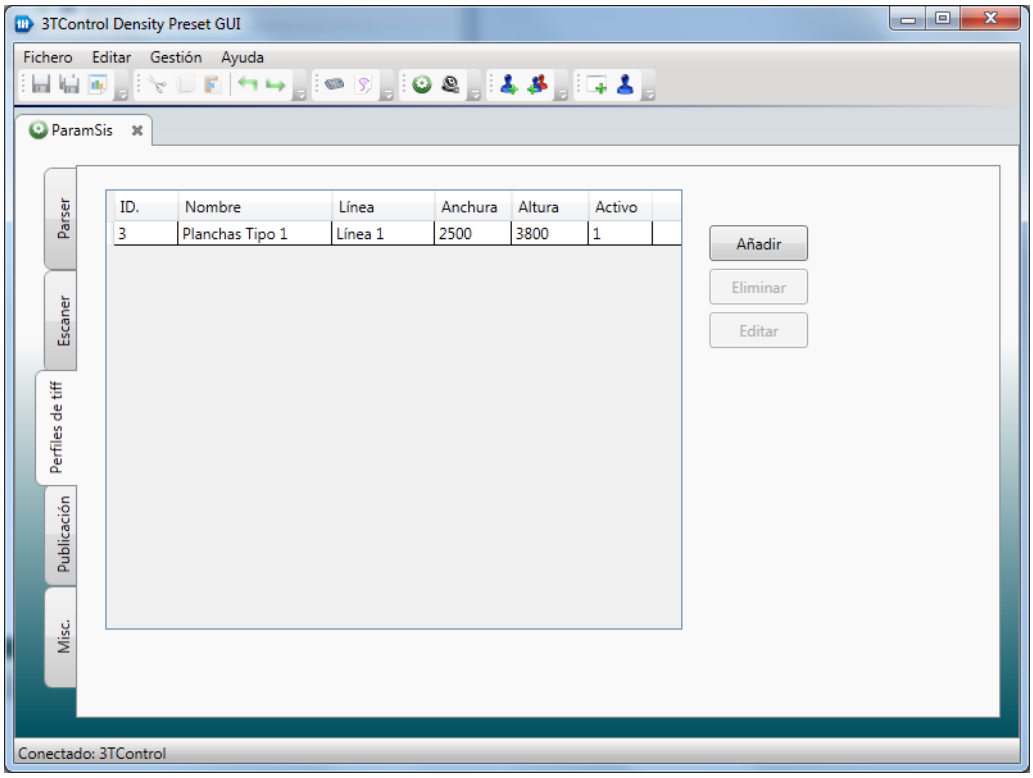

Figura 2.11.4.3 – Pestaña parámetros del sistema, opciones de perfiles de ficheros TIFF, buscador

Cuando pulsemos el botón “Add” nos llevará a la siguiente ventana:

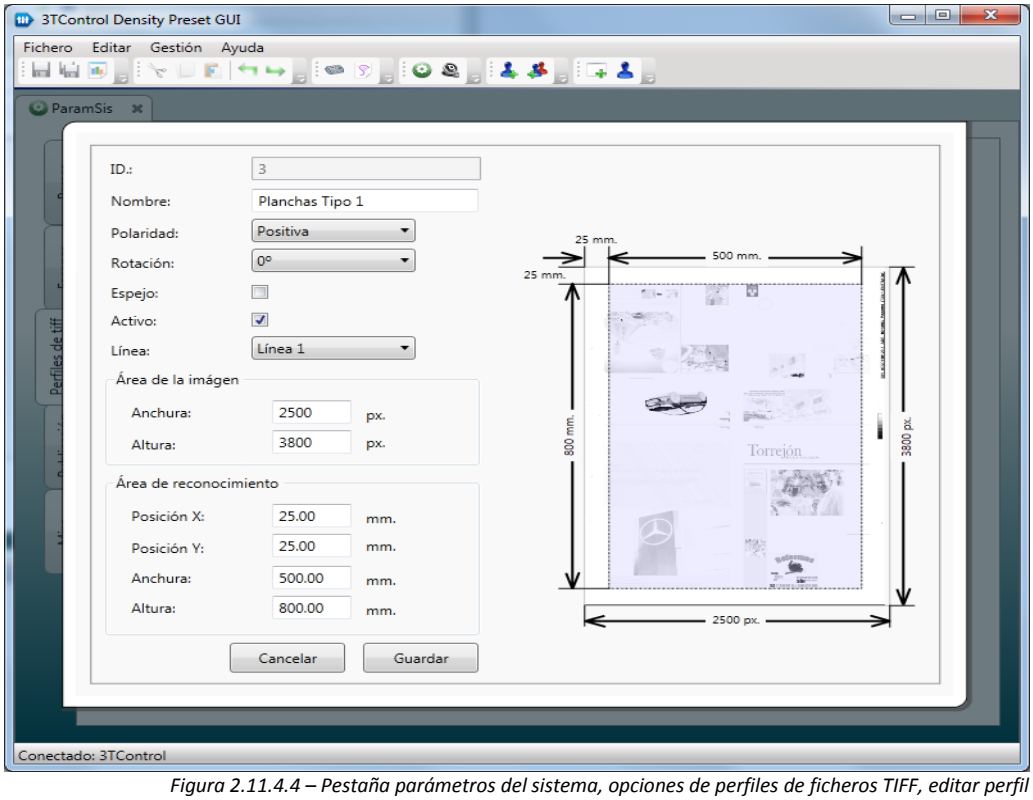

Figura 2.11.4.4 – Pestaña parámetros del sistema, opciones de perfiles de ficheros TIFF, editar perfil

Donde configuraremos las características físicas de los ficheros TIFF asociados a ese perfil.

Nos podrán llegar ficheros TIFF de diferentes tamaños con diferentes márgenes. El mantener perfiles nos servirá para que el servidor sepa que dado un tamaño de ancho y alto de fichero TIFF lo asocie a un perfil en concreto y leamos dentro de los márgenes establecidos para el mismo.

Las unidades de medida de pixeles para dimensiones y milímetros para márgenes están elegidas así ya que así es como trabajan habitualmente los futuros usuarios del sistema.

La conversión de mm a pixeles viene dada por el tamaño de la CTP, parámetro de "Línea", de ahí que se exija asociar el perfil a una línea.

La formulación que permite convertir mm en pixeles es la siguiente:

$$medida\_px = \frac{tamaño\_mm}{tamaño\_ctp}$$

La siguiente pestaña muestra las publicaciones existentes.

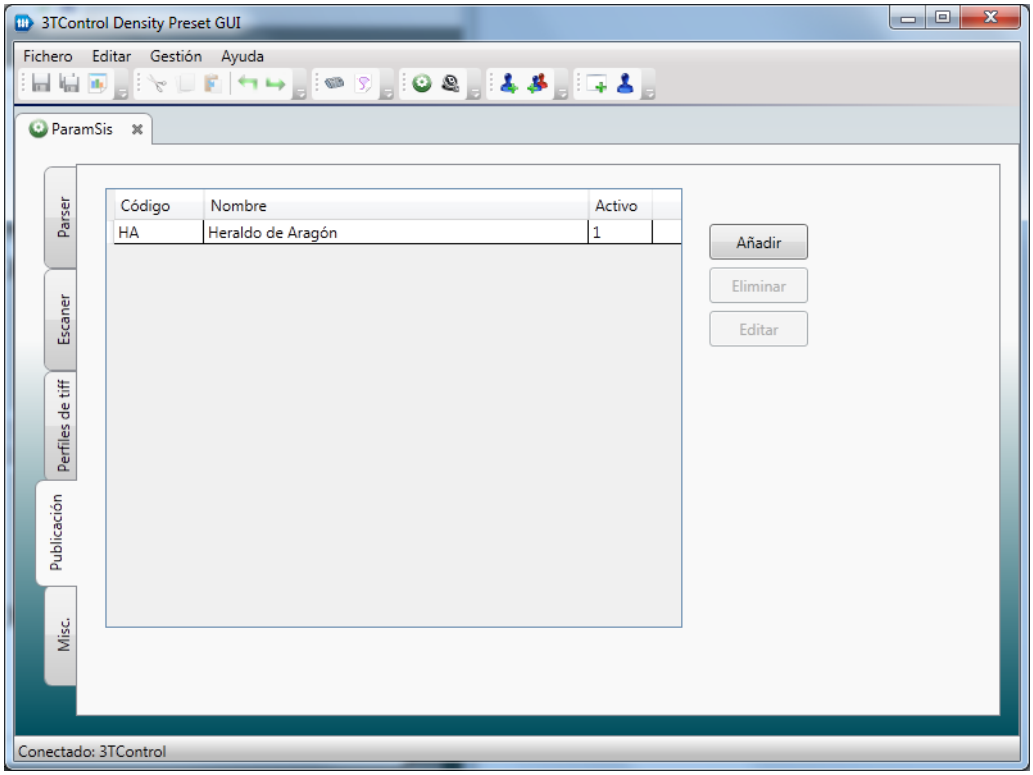

Figura 2.11.4.5 – Pestaña parámetros del sistema, opciones publicaciones, listado de publicaciones

Cuando pulsemos “Añadir” o “Editar” accederemos a la siguiente ventana:

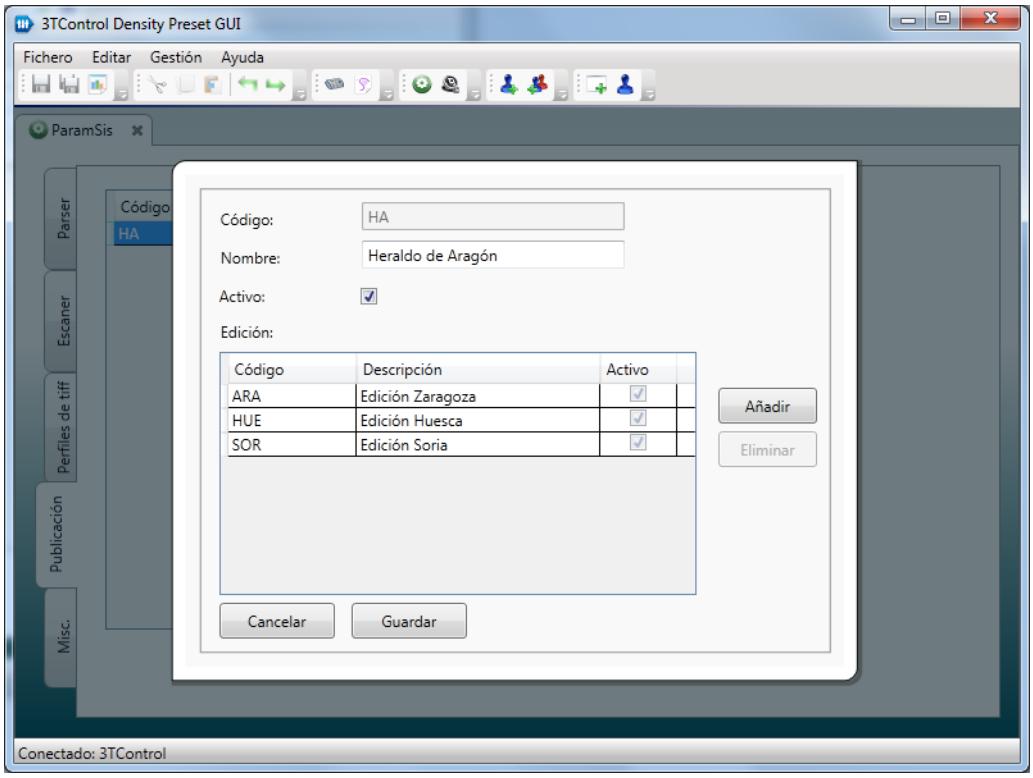

Figura 2.11.4.6 – Pestaña parámetros del sistema, opciones publicaciones, edición de publicación

Aquí configuraremos la publicación, que quedará definida por su código. También podremos gestionar las ediciones y subproductos.

Por último la última subpestaña de parámetros de sistema es para configurar diversos parámetros como rutas “SMB” (o “Samba”) donde tomar las miniaturas y dirección y puerto del servidor para mensajería.

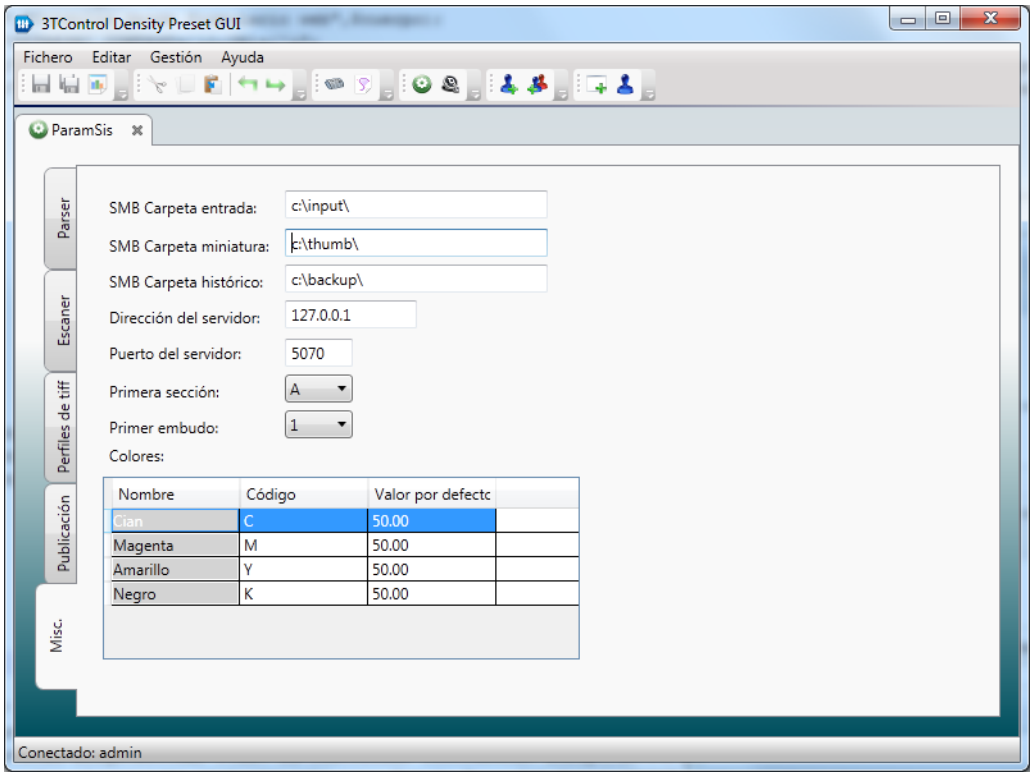

Figura 2.11.4.6 – Pestaña parámetros del sistema, opciones miscelánea

2.11.5 Gestión de trabajos: Buscador

Desde está ventana localizaremos y veremos el estado de los trabajos existentes. Tenemos la posibilidad de añadir filtros para localizar más fácilmente el trabajo que buscamos y muestra el estado y porcentaje de cada trabajo. Se actualiza cada cierto intervalo de tiempo.

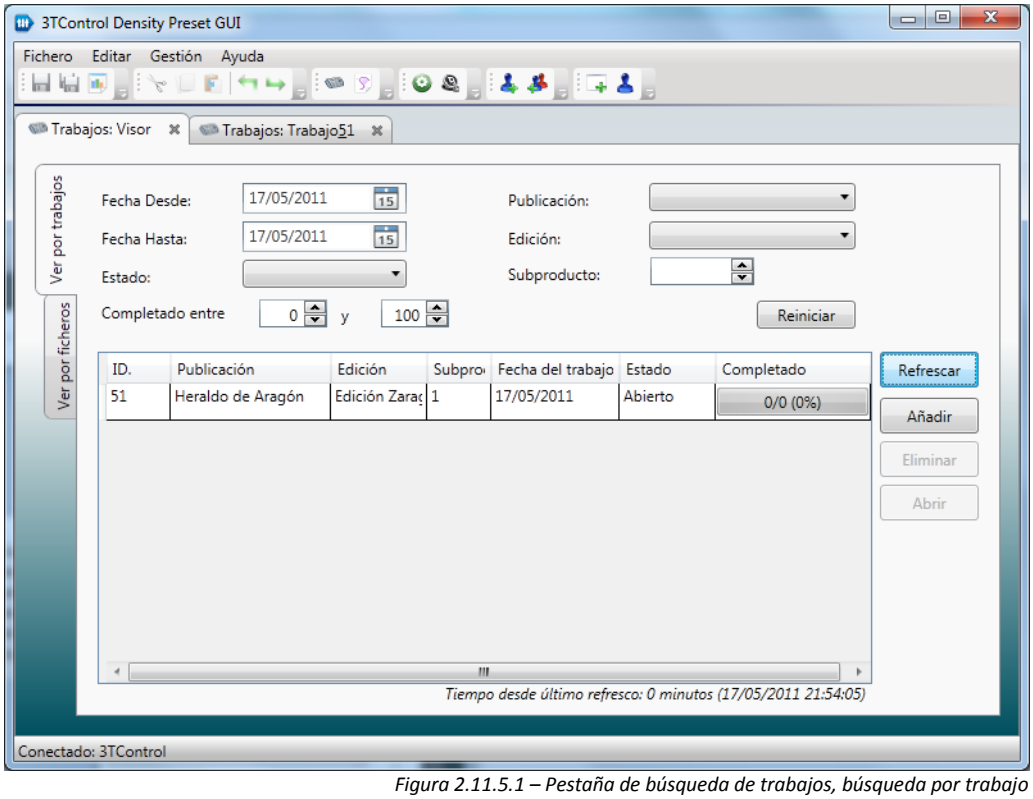

Figura 2.11.5.1 – Pestaña de búsqueda de trabajos, búsqueda por trabajo

En la segunda pestaña podemos localizar ficheros que han sido procesados por el servidor y ver información relativa a ellos como su estado.

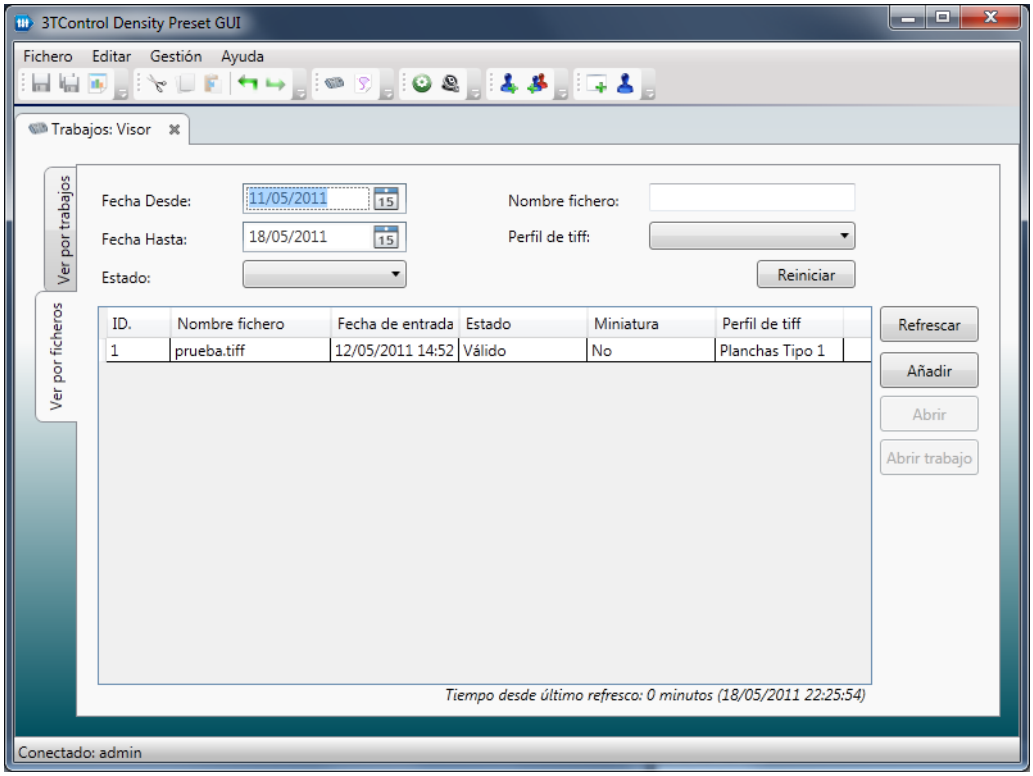

Figura 2.11.5.2 – Pestaña de búsqueda de trabajos, búsqueda por fichero

Además desde esta ventana pulsando "Añadir" podremos subir uno o varios ficheros. Los cambios no serán instantáneos pues el servidor tiene que procesarlos.

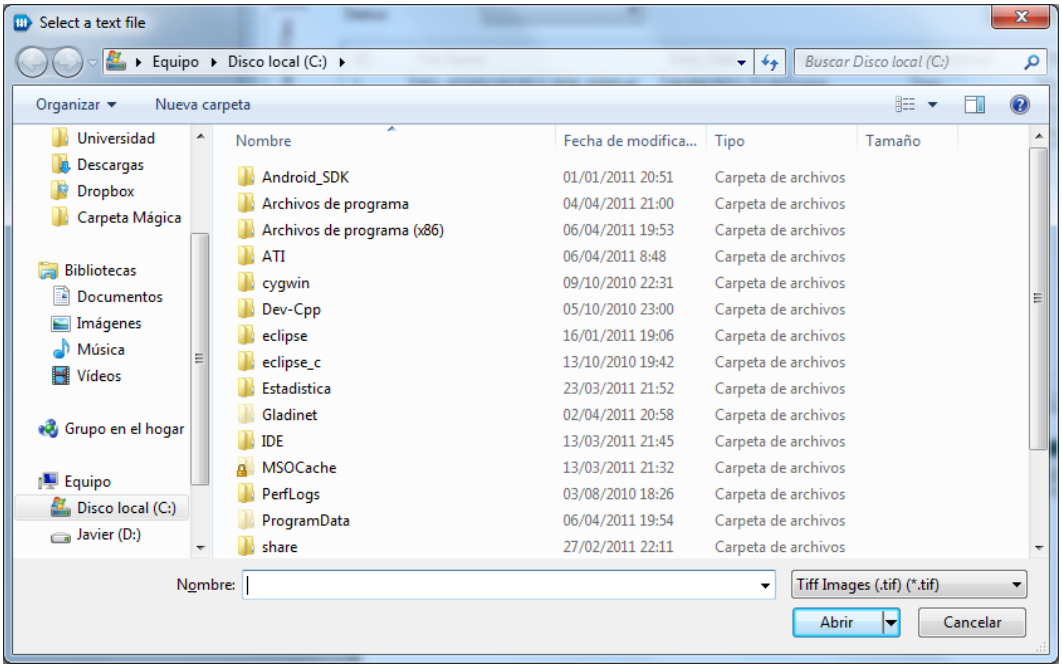

Figura 2.11.5.3 – Pestaña de búsqueda de ficheros, ventana emergente de adición de ficheros

2.11.6 Gestión de trabajos: Nuevo/Modificar trabajo

Desde aquí podrémos modificar los trabajos o crear uno nuevo. Seleccionaremos la publicación, la configuración de la rotativa donde se imprimirá y un perfil de curvas.

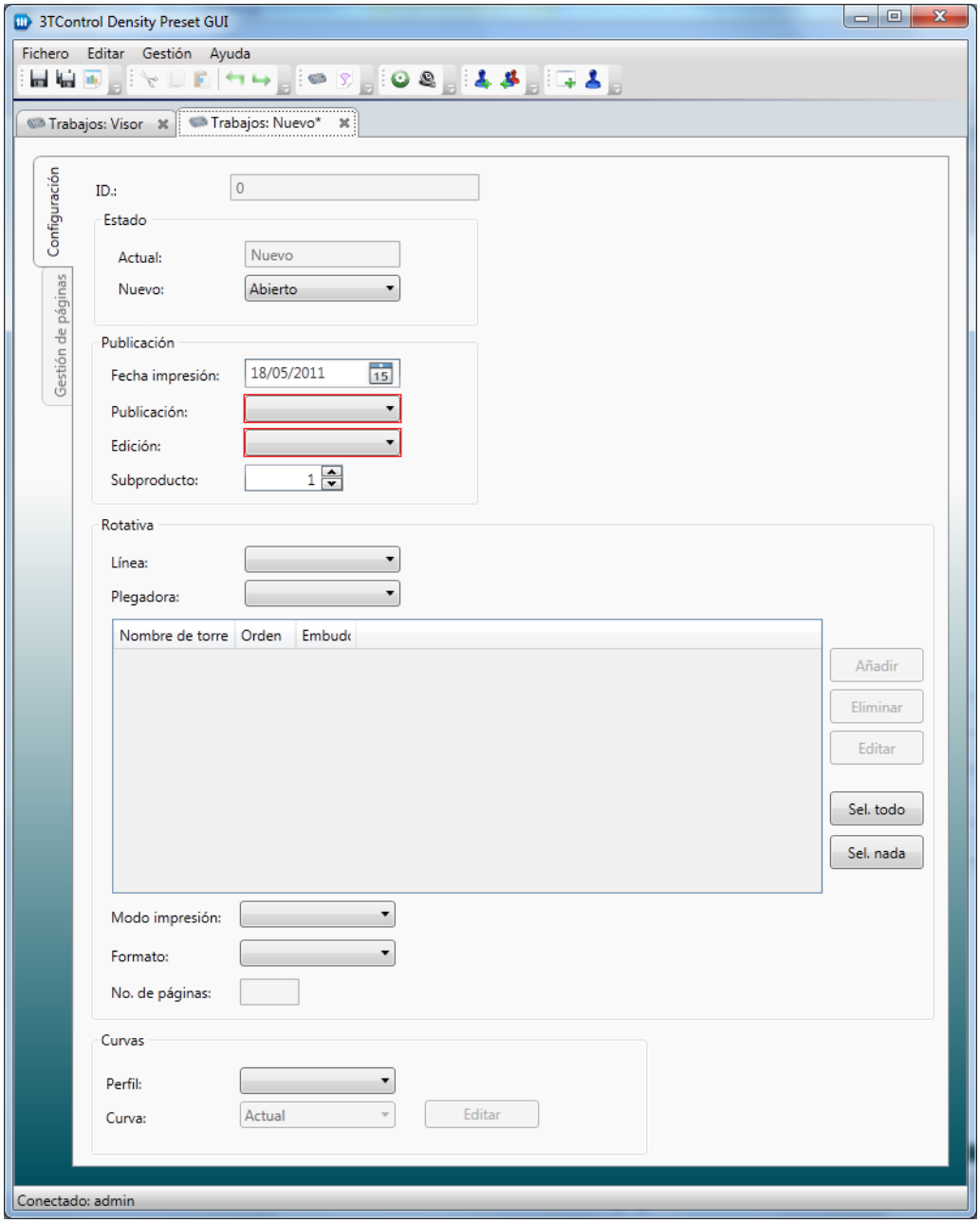

Figura 2.11.6.1 – Pestaña de edición de trabajo, configuración del trabajo

Si pulsamos el botón añadir dentro de la sección “Rotativa” podrémos añadir las torres con las que se imprimirá el trabajo así como el embudo usado y las secciones.

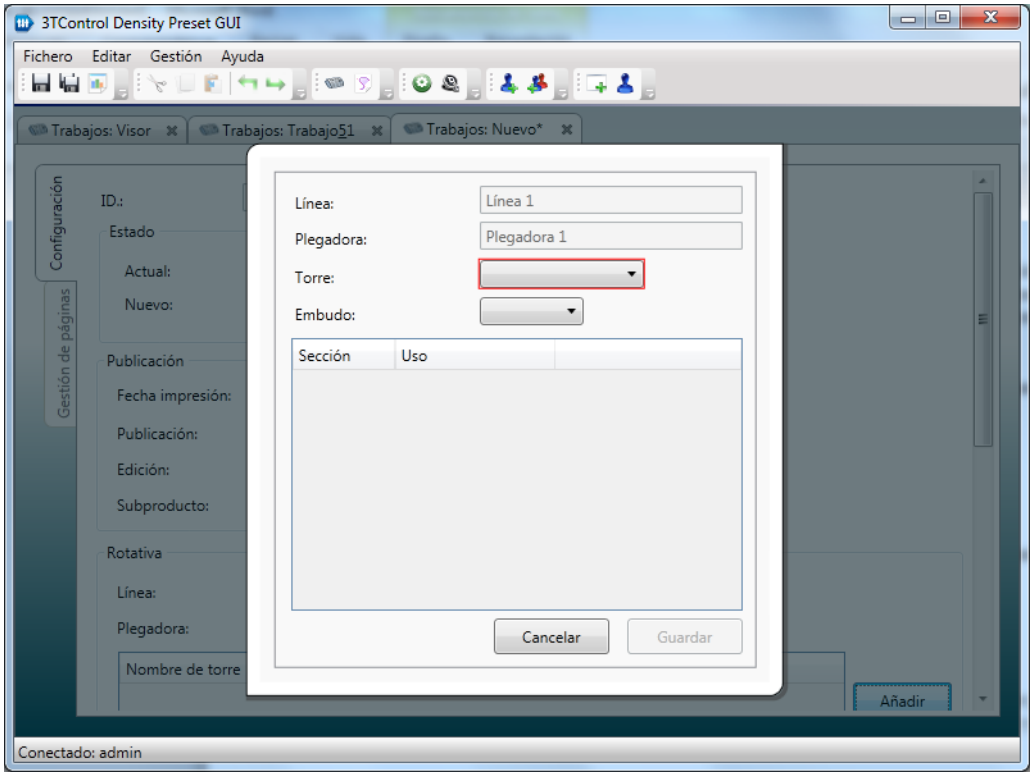

Figura 2.11.6.2 – Pestaña configuración del trabajo, añadir torres al trabajo

Desde la siguiente pestaña verémos el estado de cada una de las páginas y del trabajo en general, indicando en el listado, además de con un campo “estado”, con un color de fila si existe información para el fichero o no. Si existe la fila estará en verde, si no en amarillo y si ha entrado una fichero que no encaja con ningúna página estará en rojo, colores que coinciden con las barras de estado superiores.

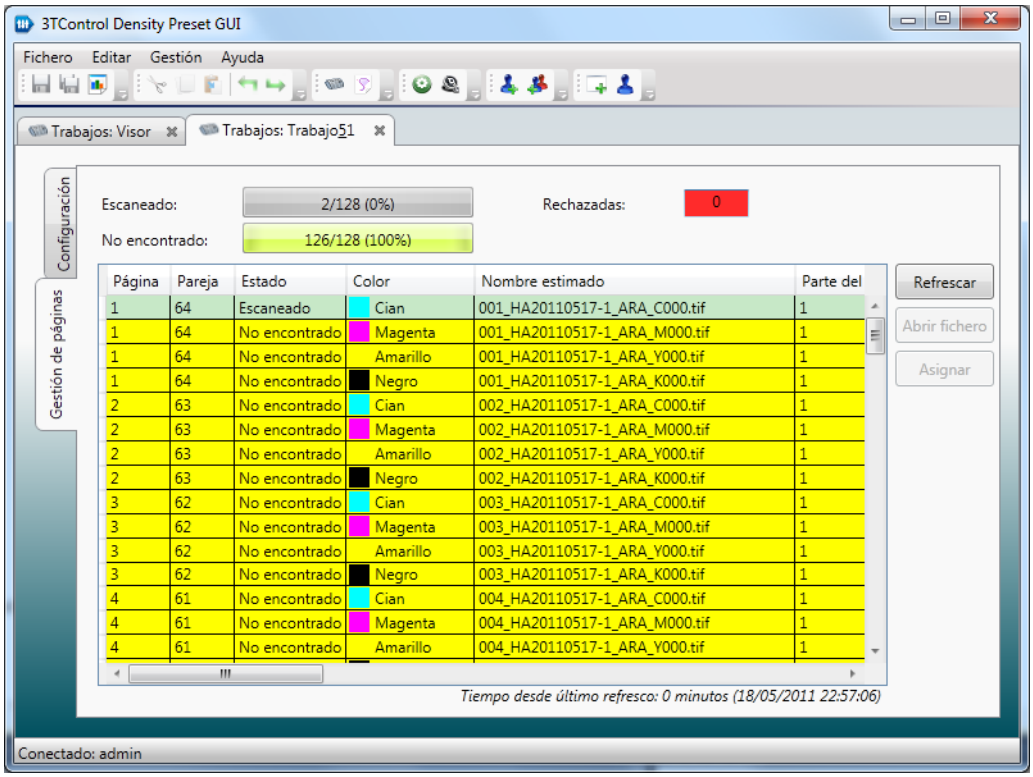

Figura 2.11.6.3 – Pestaña de edición de trabajo, páginas asociadas a un trabajo.

Además si seleccionamos una página y pulsamos “Assign” podrémos cambiar la asignación de una página (asignada ya o no) a un fichero que nosotros queramos. Para ello se abre el siguiente pop-up que nos permite la búsqueda del fichero:

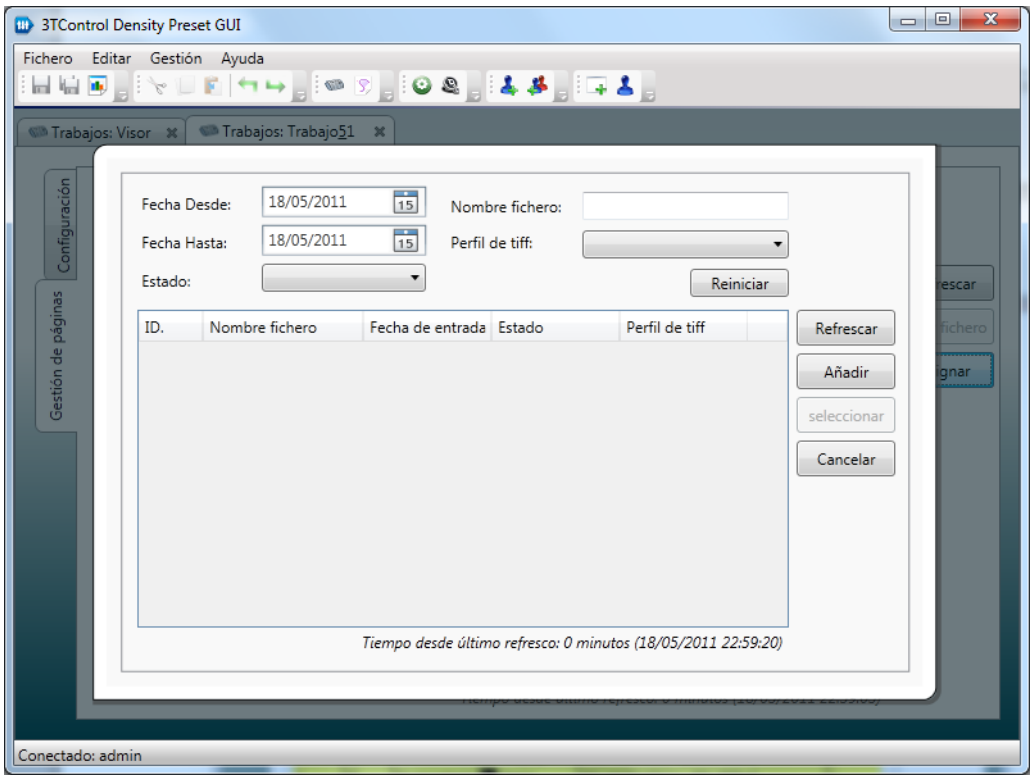

Figura 2.11.6.4 – Pestaña de edición de trabajo, ventana emergente búsqueda y selección de fichero.

2.11.7 Gestión de trabajos: Ver informe

Es la ventana del visor de informes para los trabajos

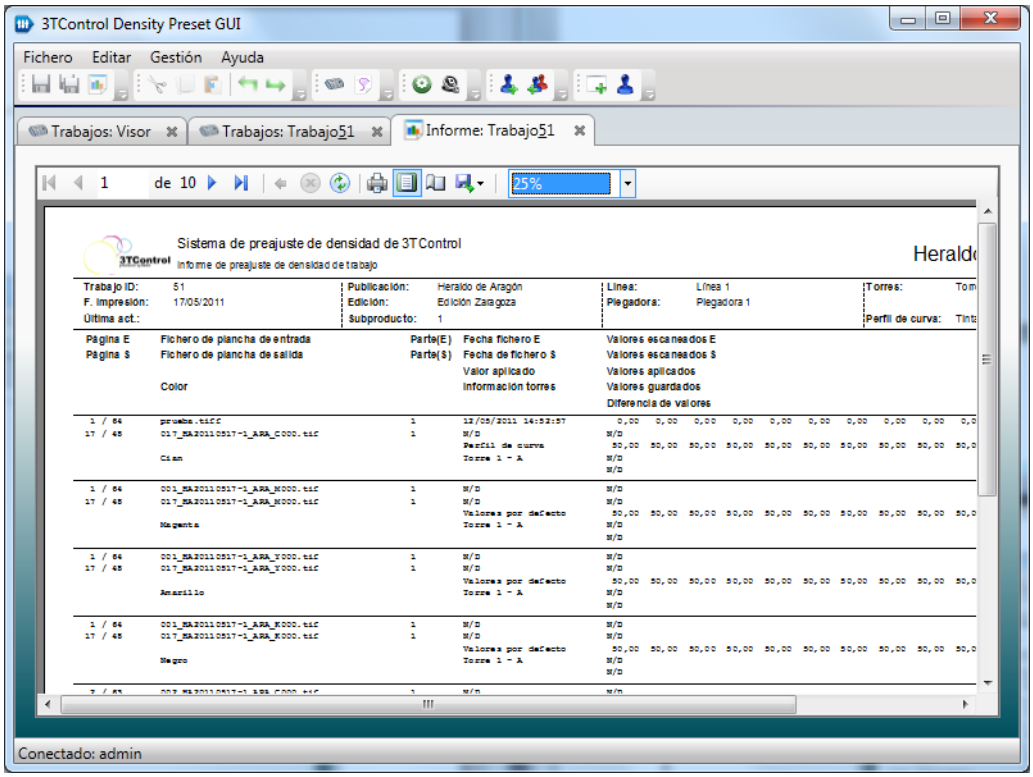

Figura 2.11.7.1 – Pestaña de informe

2.11.8 Ver información de fichero

La página de detalle del fichero es la siguiente.

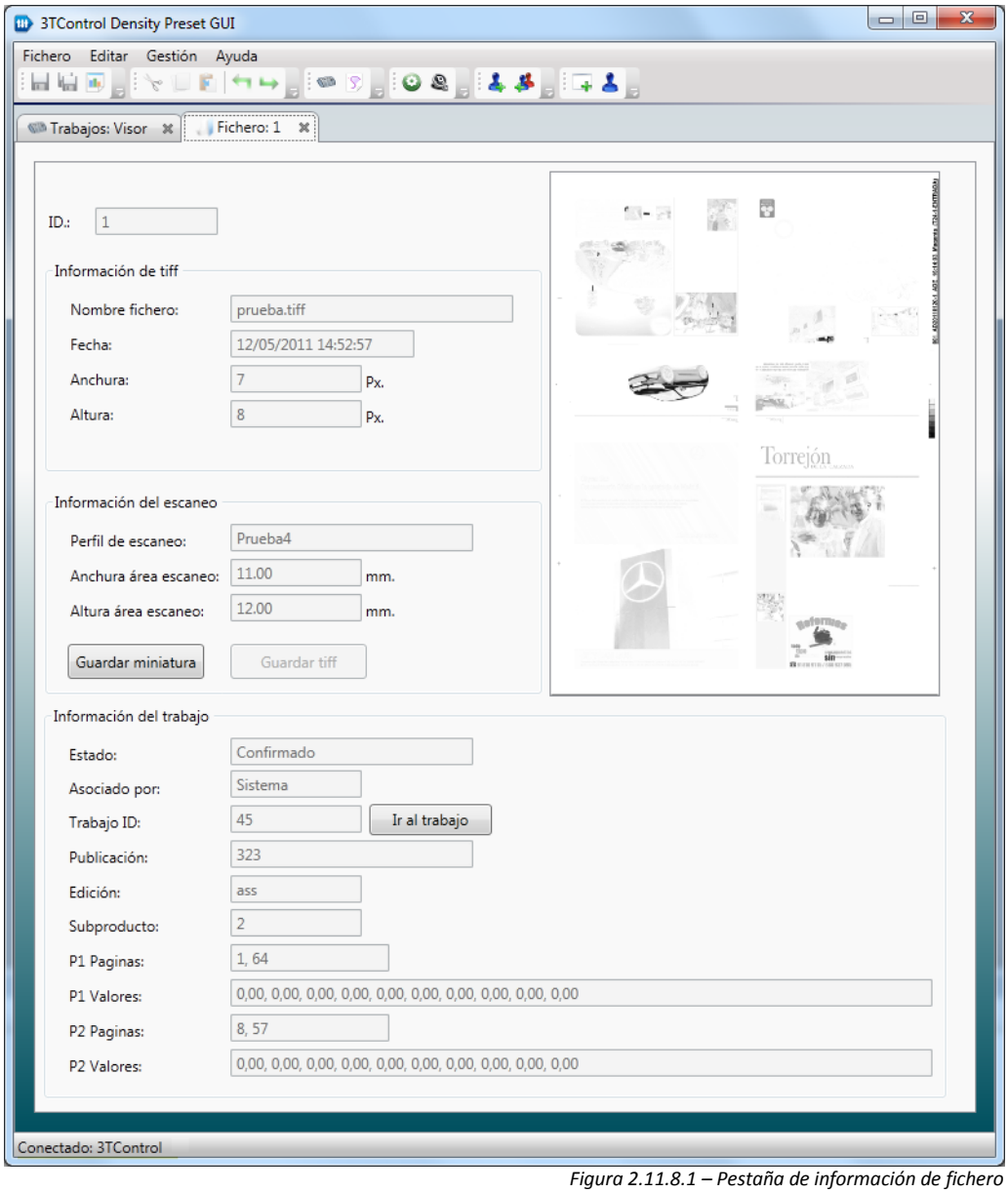

Figura 2.11.8.1 – Pestaña de información de fichero

No se puede modificar nada, tan solo visualizar. También tenemos la opción de recuperar el fichero TIFF original o descargar la miniatura si existe.

Por otro lado mostrará la imagen si existe y se ha podido recuperar.

2.11.9 Gestión de curvas: Buscador de perfiles

Esta página mostrará el listado de perfiles. Permite la búsqueda por nombre. Con ella podemos localizar un perfil de curvas y editarlo o añadir uno nuevo.

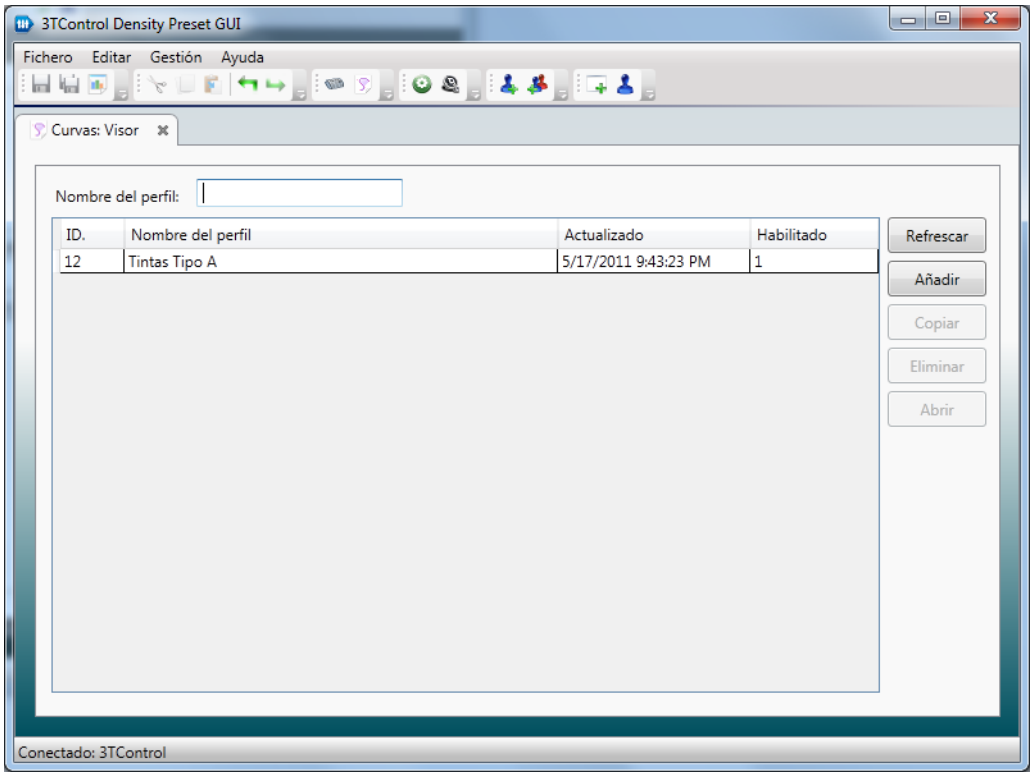

Figura 2.11.9.1 – Pestaña de búsqueda de perfiles de curvas

2.11.10 Gestión de curvas: Perfil nuevo / modificar existente

Cuando en la pantalla del buscador de perfiles de curvas pulsemos “Añadir” o “Eliminar” nos llevará a la siguiente ventana que esta subdividida en 3 pestañas. En la primera editaremos información referente al nombre y si está activo.

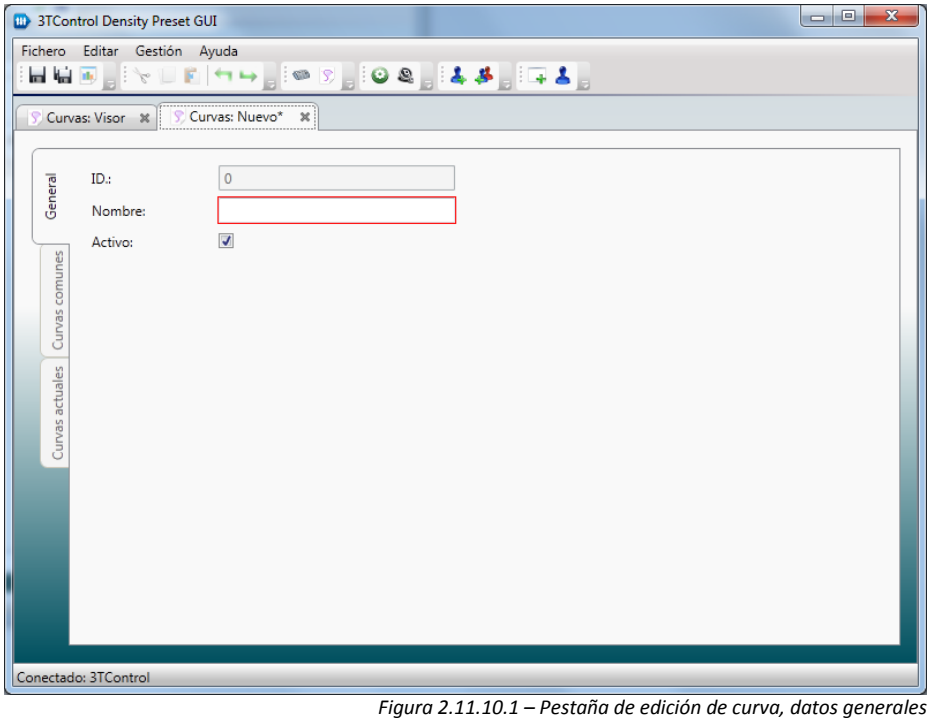

Figura 2.11.10.1 – Pestaña de edición de curva, datos generales

En la segunda ventana editaremos información referente a las curvas comunes que se aplicarán al resto de curvas.

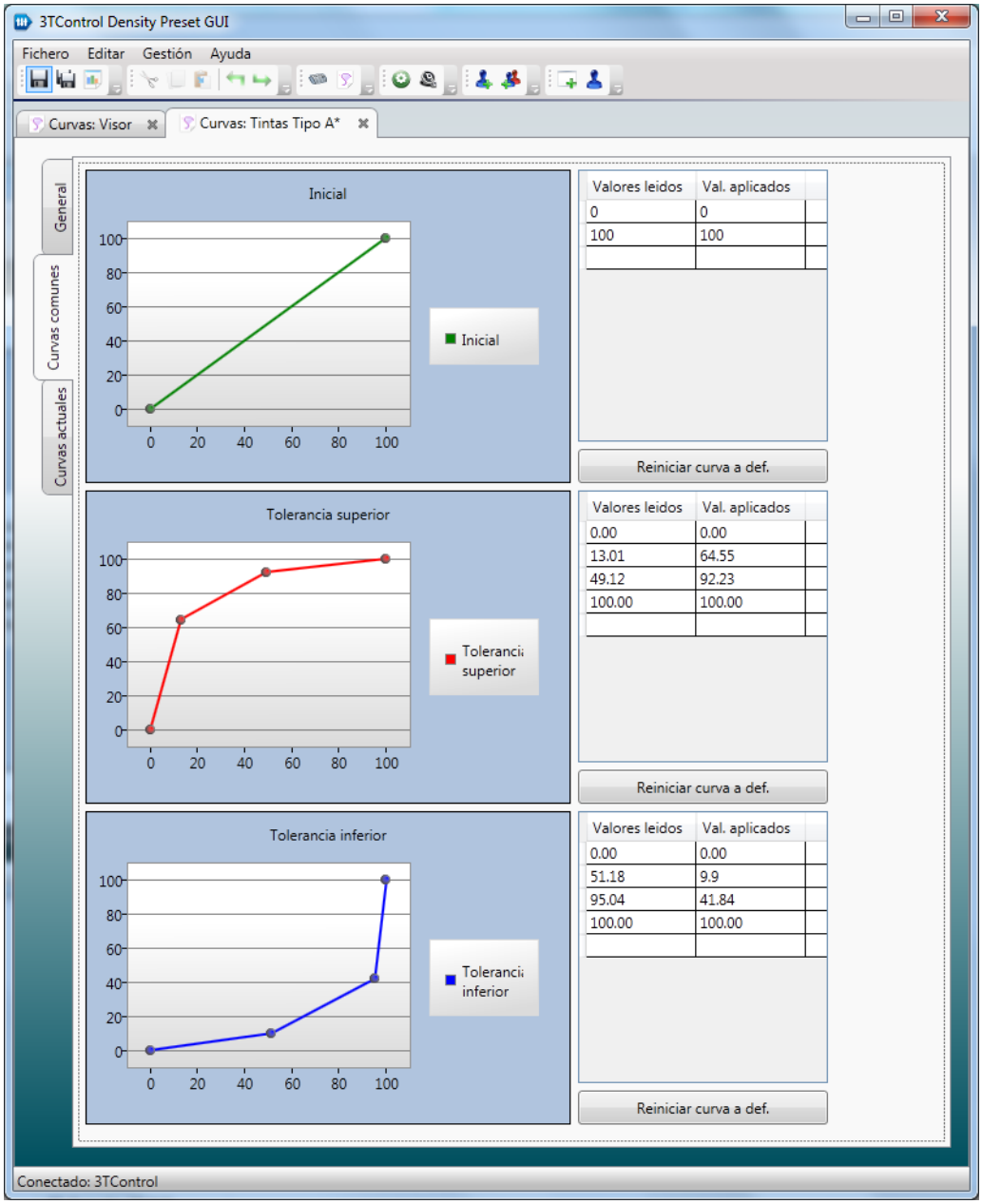

Figura 2.11.10.2 – Pestaña de edición de curva, curvas generales

Por último, en la pestaña “curvas actuales” podremos buscar y visualizar las curvas que tiene este perfil. Recordemos que pueden ser muchos resultados, así que se ha colocado un paginador en la parte superior e inferior para facilitar ver los resultados.

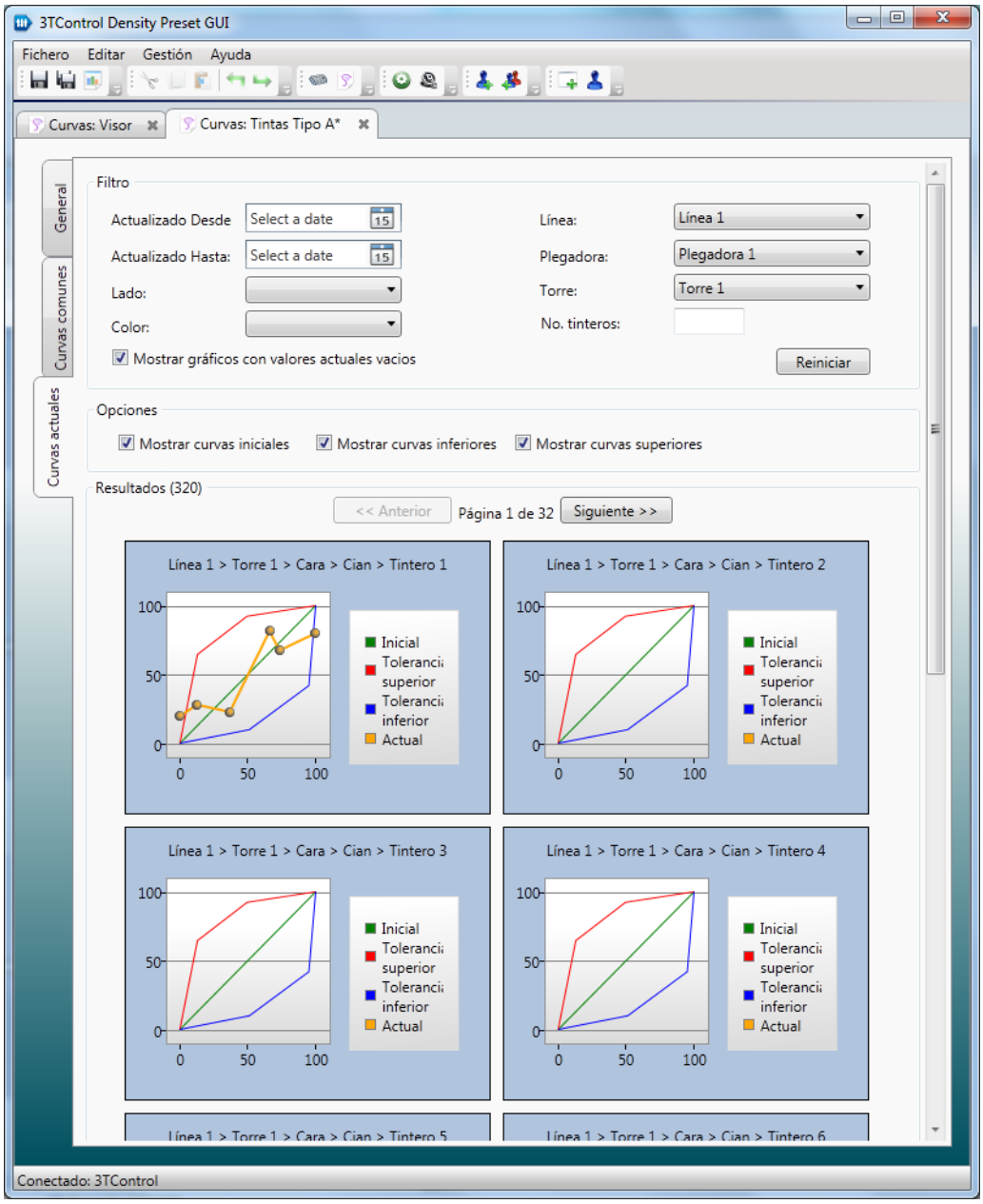

Figura 2.11.10.3 – Pestaña de edición de curva, búsqueda y visualización de curvas actuales para cada tintero

Si pulsamos en la pantalla anterior sobre alguna de las gráficas nos aparecerá el siguiente pop-up:

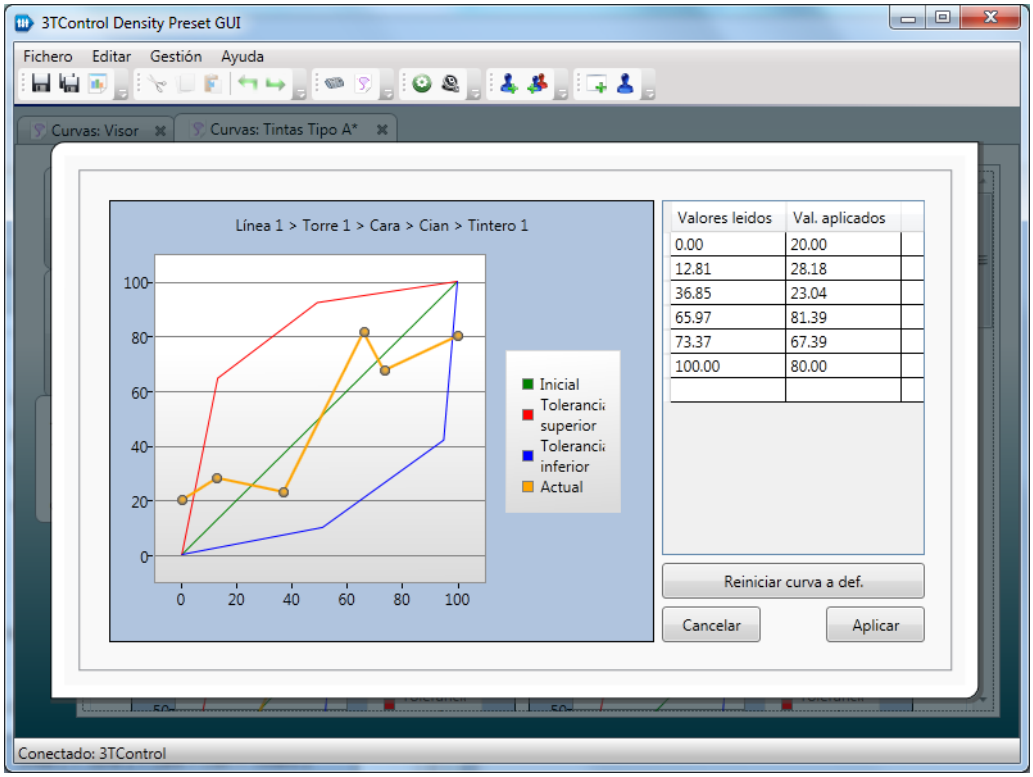

Figura 2.11.10.4 – Pestaña de edición de curva, ventaja emergente de edición de una curva

Aquí podremos editar la curva actual para ese tintero o restaurarla al valor por defecto.

2.11.11 Gestión de la rotativa

Desde las siguientes pantallas daremos de alta los diferentes componentes que forman una rotativa: líneas, plegadoras y torres.

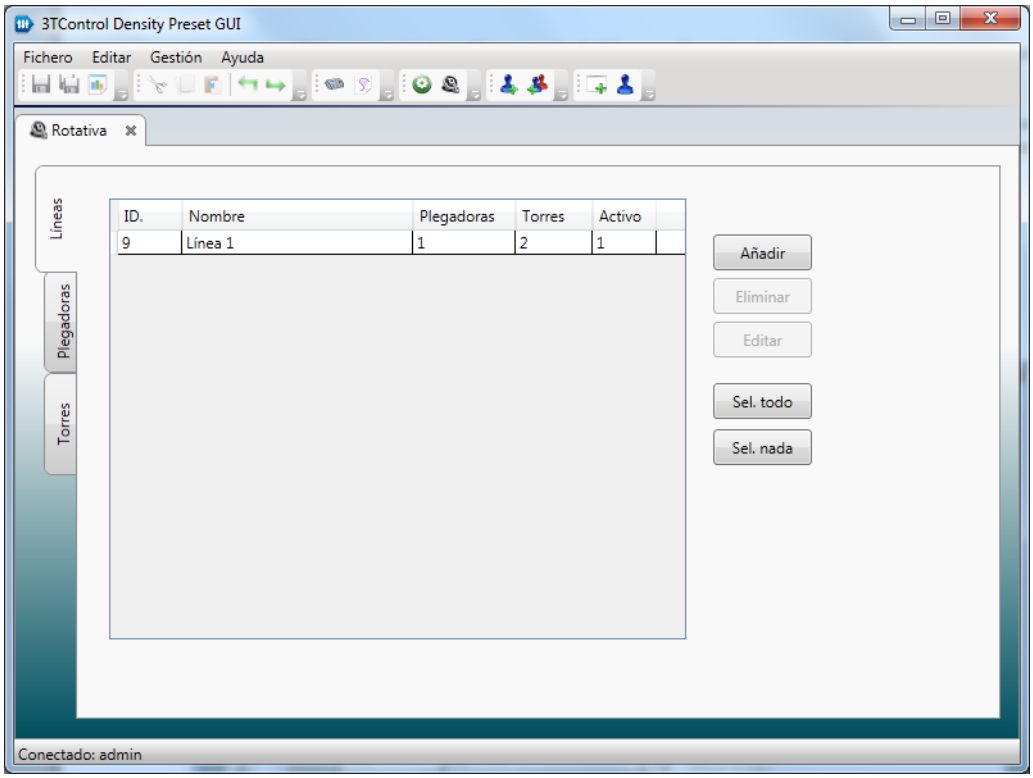

Figura 2.11.11.1 – Pestaña de edición de rotativa, listado de líneas

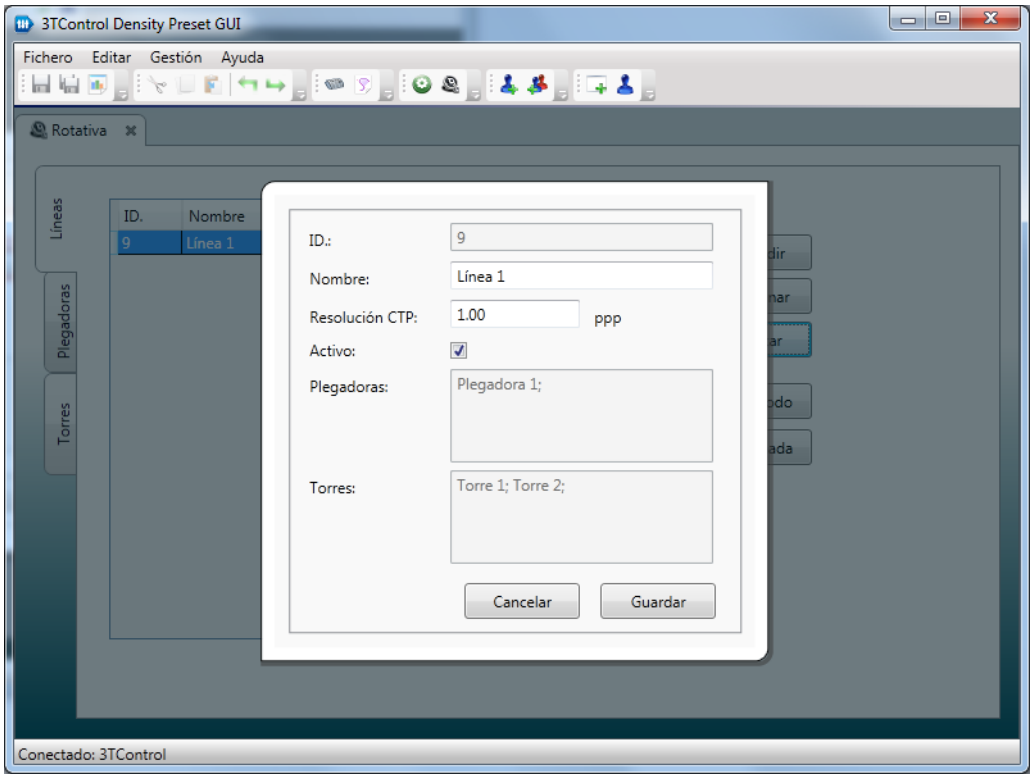

Figura 2.11.11.2 – Pestaña de edición de rotativa, edición de línea

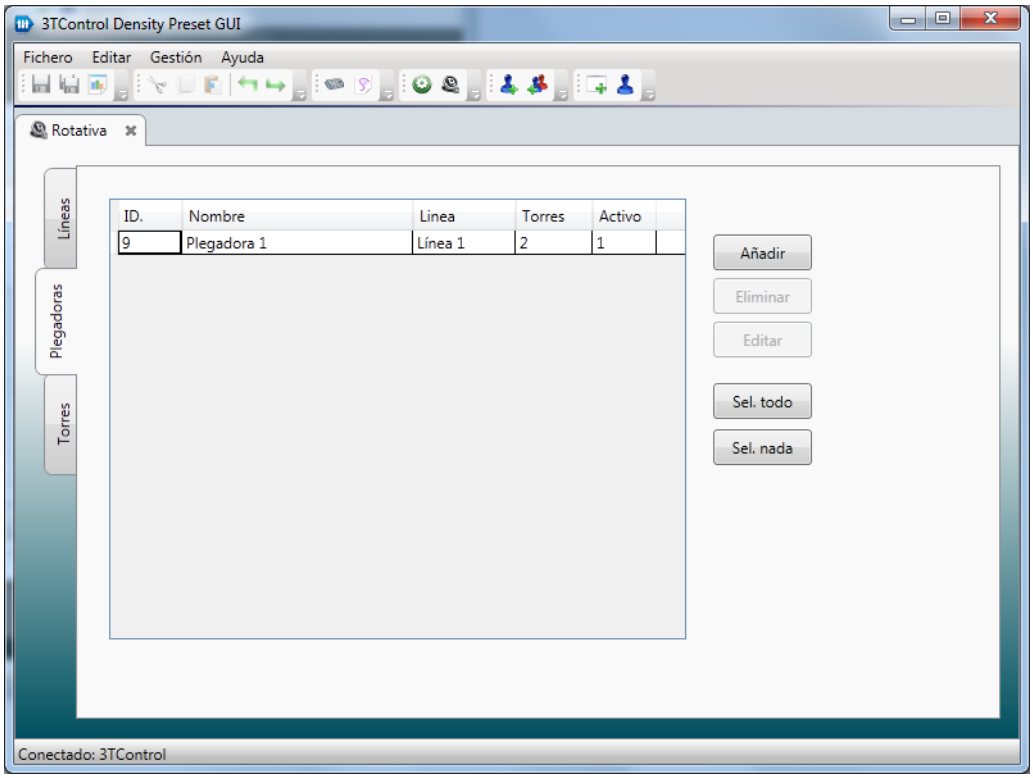

Figura 2.11.11.3 – Pestaña de edición de rotativa, listado de plegadoras

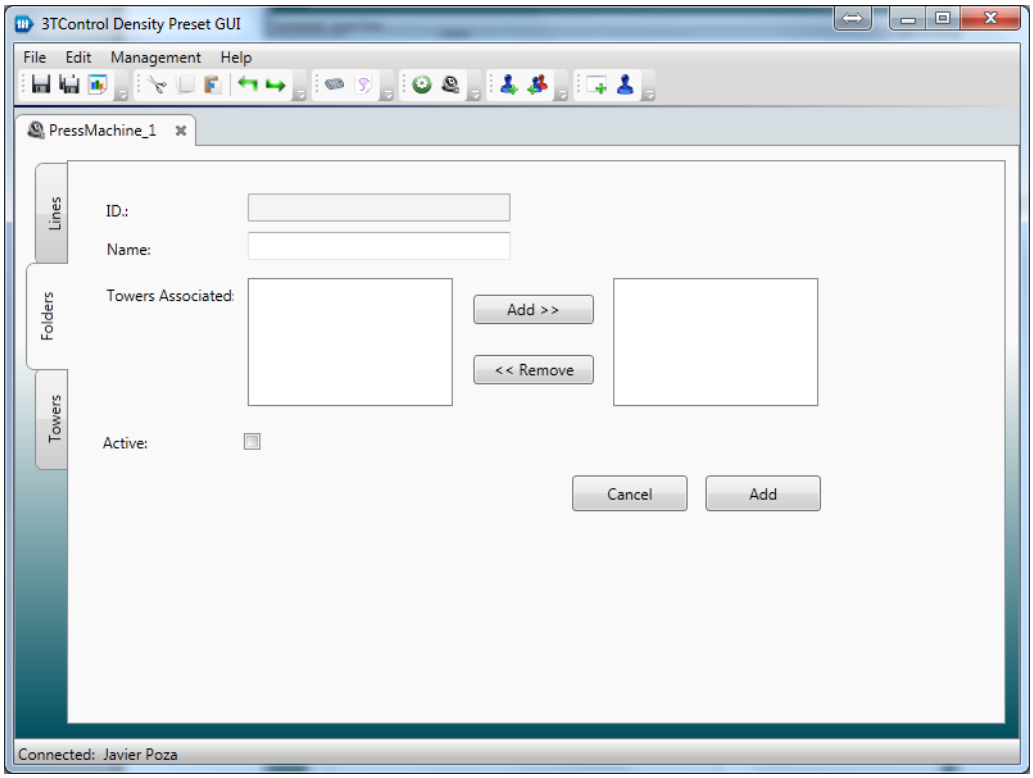

Figura 2.11.11.4 – Pestaña de edición de rotativa, edición de plegadora

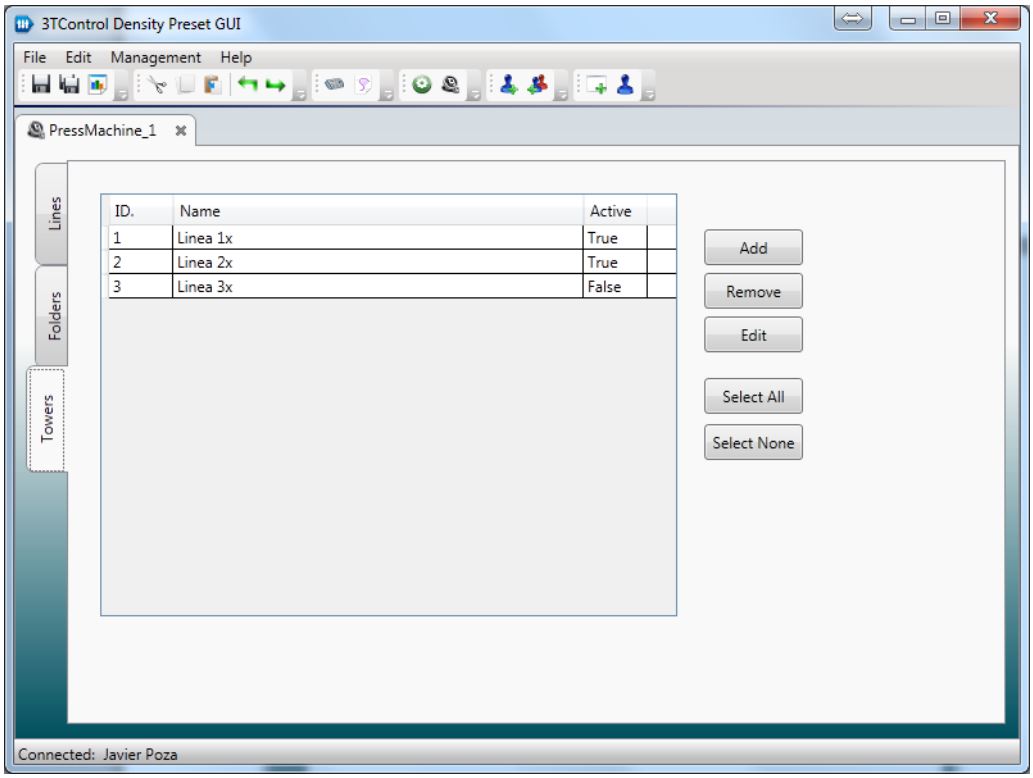

Figura 2.11.11.5 – Pestaña de edición de rotativa, listado de torres

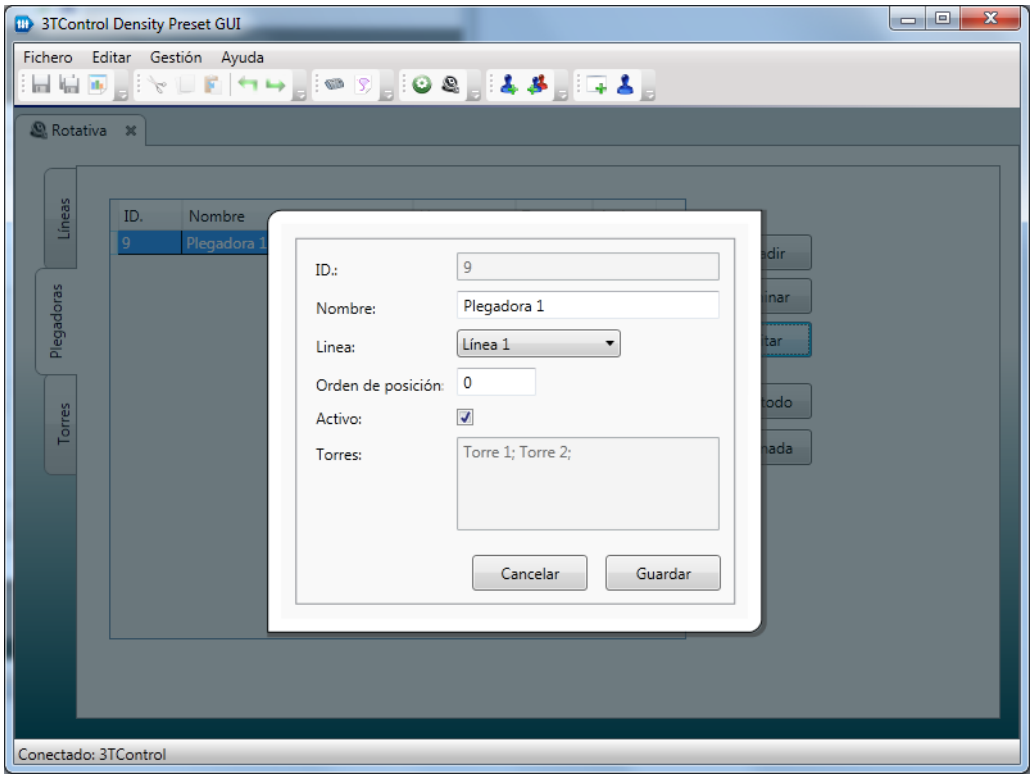

Figura 2.11.11.6 – Pestaña de edición de rotativa, edición de torre

2.11.12 Gestión de usuarios

Desde las siguientes ventanas el administrador podrá gestionar los usuarios:

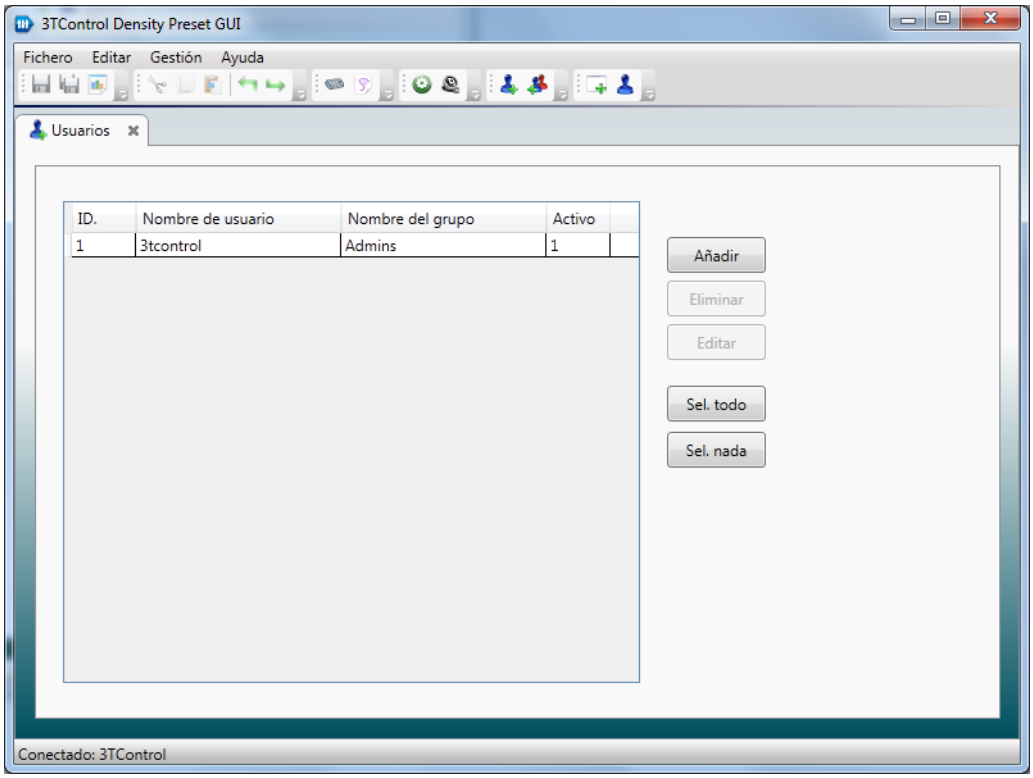

Figura 2.11.12.1 – Pestaña de edición de usuarios, listado de usuarios

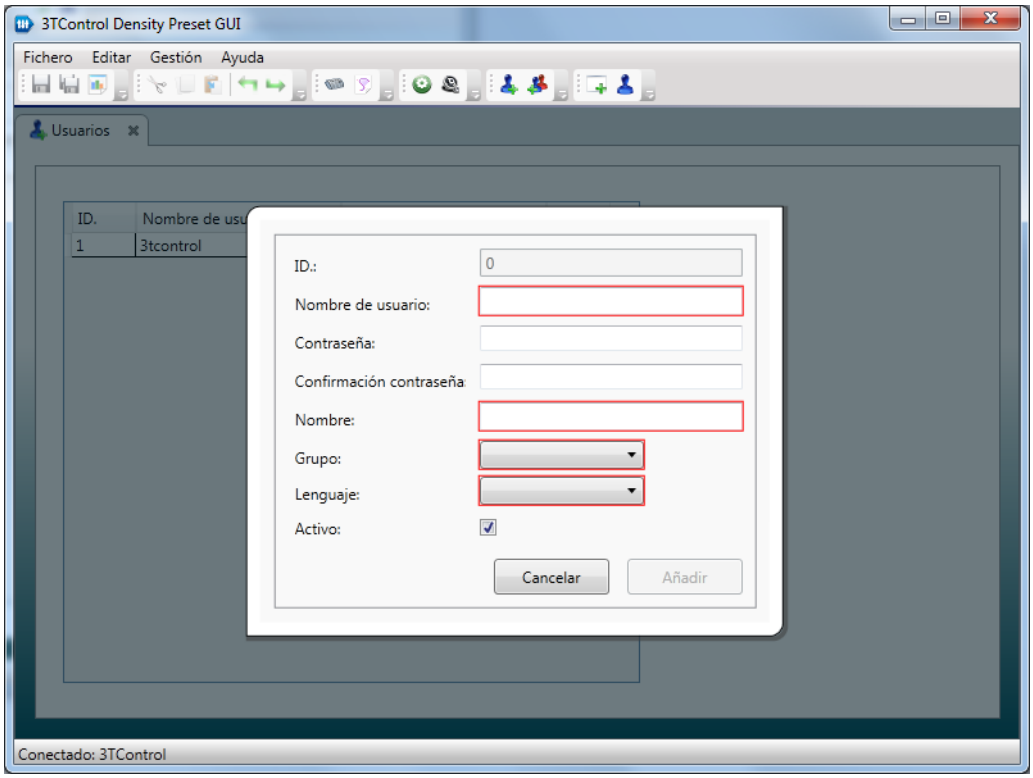

Figura 2.11.12.2 – Pestaña de edición de usuarios, edición de usuario

2.11.13 Gestión de grupos

Desde las siguientes ventanas el administrador podrá gestionar los grupos:

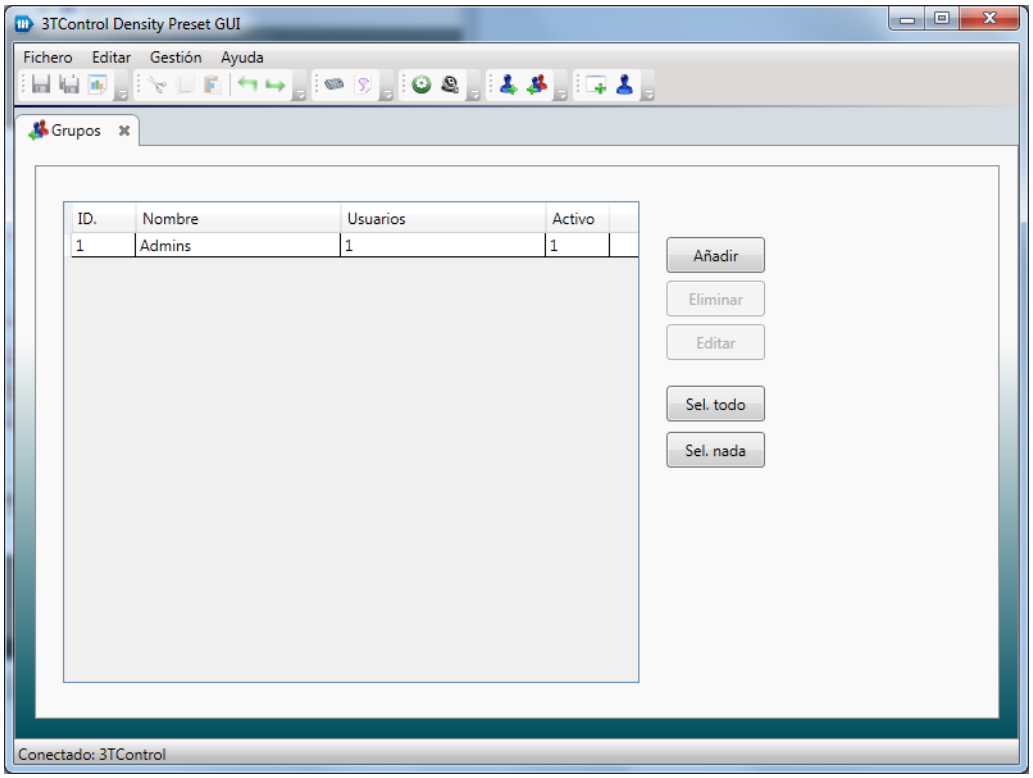

Figura 2.11.13.1 – Pestaña de edición de usuarios, listado de grupos

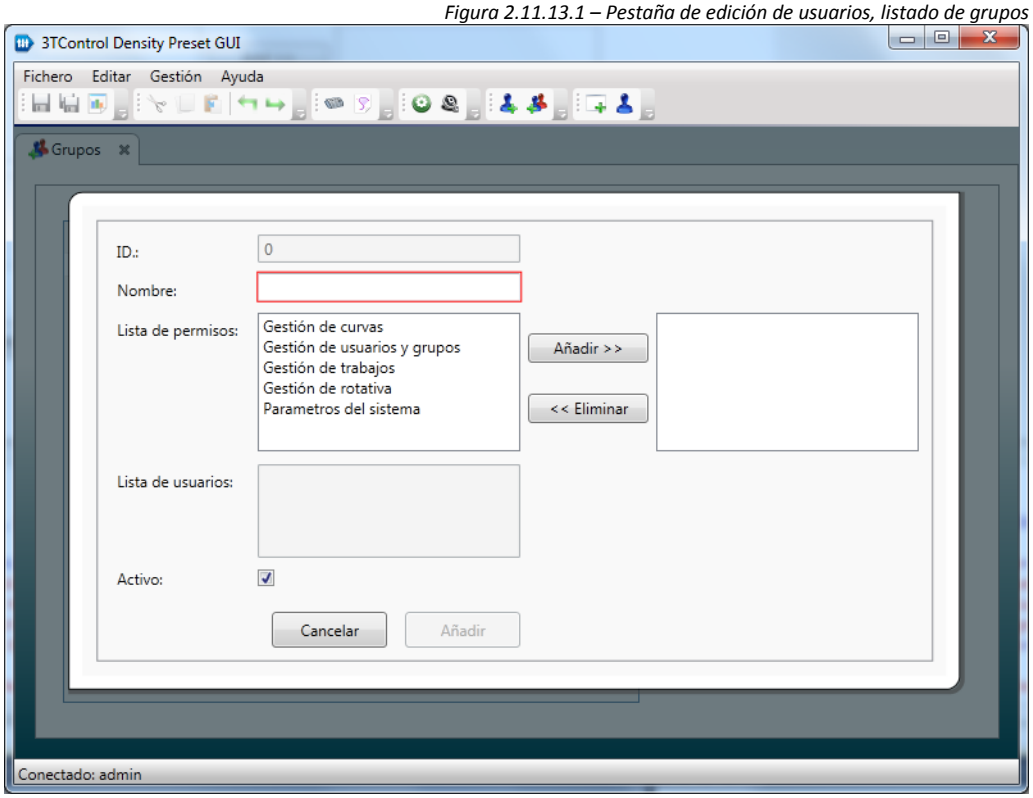

Figura 2.11.13.2 – Pestaña de edición de usuarios, edición de grupo

2.11.14 About box

Figura 2.11.14.1 – Ventana emergente "About box"

3. IMPLEMENTACIÓN

3.1 Software utilizado

Entorno de desarrollo:

- Microsoft Visual Studio 2010 Ultimate
- Microsoft Windows 7 Ultimate
- Microsoft SQL Server Management Studio Express

Liberías y controles:

- Microsoft .NET Framework 4.0
- LinqKit
- WPF Toolkit
- WPF Toolkit Extended
- WPF Localize Extension
- Linq to SQL
- Report Viewer

Componentes servidor:

- Microsoft SQL Server 2008 R2

3.2 Capa de datos

Para gestionar la capa de datos se ha usado *Linq to SQL* ya que era el método más sencillo de implementar y de usar y puesto que la base de datos tiene un tamaño considerable en cuanto a cantidad de tablas y de campos ha acelerado mucho el desarrollo.

El fichero que contiene todas las clases generadas por el método indicado se llama *Preset3TDB.dbml*.

3.3 Lógica de negocio

Puesto que prácticamente toda la lógica se centra en la gestión de información en la base de datos se ha usado las clases creadas para *Linq* y las relaciones entre tablas de la base de datos. También se ha implementado cierta lógica de negocio sobre las propias clases de los formularios. Hubiera sido más correcto crear clases específicas para esto, pero por problemas de tiempo se ha hecho así.

Cabe destacar que la lógica de negocio que más confusión puede generar es la de la compaginación:

```
private void createPages()
{

    // Primero tenemos que tener una lista ordenada. El orden es el siguiente: Primero las
    // torres cuyo orden es superior al de la plegadora seleccionada y despues las que tienen
    // un orden inferior.
    #region jobTowersOrdered
    List<tbJobsTowers> jobTowersOrdered = (from p in jobEdit.tbJobsTowers
                                  where p.tbTowers.layout_order > jobEdit.tbFolders.layout_order
                                  orderby p.tbTowers.layout_order
                                  select p).ToList();

    jobTowersOrdered = jobTowersOrdered.Union<tbJobsTowers>((from p in jobEdit.tbJobsTowers
                                                             where p.tbTowers.layout_order < jobEdit.tbFolders.layout_order
                                                             orderby p.tbTowers.layout_order
                                                             select p).ToList()).ToList();

    #endregion

    //Obtenemos el rango de secciones maximo.
    #region firstSection , lastSection, maxSections
    int firstSection = jobEdit.tbJobsTowSecs.Where(o=> o.sectModeID != (int)App.SectionMode.NotUsed).Min(o => o.sectNum);
    int lastSection = jobEdit.tbJobsTowSecs.Where(o => o.sectModeID != (int)App.SectionMode.NotUsed).Max(o => o.sectNum);
    int maxSections = lastSection - firstSection + 1;
    #endregion

    //Obtenemos el rango de embudos.
    #region firstFormer , lastFormer, maxFormers
    int firstFormer = jobEdit.tbJobsTowers.Min(o => o.former);
    int lastFormer = jobEdit.tbJobsTowers.Max(o => o.former);
    int maxFormers = lastFormer - firstFormer + 1;
    #endregion

    //Obtenemos el número de páginas
    #region totalPages
    int totalPages = calculateTotalPages();
    #endregion

    //Obtenemos el estado por defecto
    #region defaultStatus
    tbJobPageStatus defaultStatus = db.tbJobPageStatus.Where(o=>o.pageStatusID==(int)App.JobPageStatus.Missing).First();
    #endregion

    //Obtenemos el objeto de color negro
    #region colorBlack
    tbColors colorBlack = db.tbColors.Where(o=>o.colorID==(int)App.TowerColor.Black).First();
    #endregion

    //Obtenemos si es doble producción
    #region doubleProduction
    bool doubleProduction = jobEdit.modeID == (int)App.PrintMode.DoubleProduction;
    #endregion

    //Obtenemos si es doble producción
```

```
    #region broadSheet
    bool broadSheet = jobEdit.formatID == (int)App.PrintFormat.Broadsheet;
    #endregion

    //Obtenemos el esqueleto del nombre de fichero:
    #region fileSkeleton & pageformat
    string fileSkeleton = buildFileNameSkeleton();
    string pageformat = db.tbScanParser.Where(o => o.typeID == (int)App.ParserType.Type_FirstPage).First().format;
    #endregion

    //Generamos las paginas con la compaginación
    #region Pagination
    int pageCounter = 1;
    for (int i = 1; i <= 2; i++) // Planchas (i = 1 o i = 2)
    {
        for (int n = 1; n <= maxFormers; n++) // Embudos
        {
            for (int j = 0; j < maxSections; j++) // Secciones
            {
                List<tbJobsTowers> jobTowerList = jobTowersOrdered;
                List<tbTowerSide> sideList = db.tbTowerSide.OrderBy(o=>o.sideID).ToList();

                if (j % 2 == 1) // Si es sección impar
                {
                    jobTowerList = jobTowersOrdered.Reverse<tbJobsTowers>().ToList();
                    sideList = db.tbTowerSide.OrderByDescending(o => o.sideID).ToList();
                }

                foreach (tbJobsTowers jobTower in jobTowerList)
                {
                    if (jobTower.former == firstFormer + n - 1)
                    {
                        // Comprueba que la sección está activa
                        if (jobTower.tbJobsTowSecs.Where(o => o.sectNum == j && o.sectModeID !=
(int)App.SectionMode.NotUsed).Count() > 0)
                        {

                            // Cara / Retiracion (Front / Back)
                            foreach (tbTowerSide l in sideList)
                            {
                                if (!doubleProduction || i == 1)
                                {
                                    int firstPagePlate = pageCounter;
                                    // Si la sección es impar
                                    if ((j % 2) == 1)
                                    {
                                        // Buscamos que se imprime en la misma torre, cara, sección pero en la otra plancha
para tomar su
                                        // pagina inicial esperada.
                                        IEnumerable<tbJobPages> results = jobEdit.tbJobPages.Where(o => o.tbTowers ==
jobTower.tbTowers &&
                                                                                                   o.tbTowerSide == l &&
                                                                                                   o.section == j-1 &&
                                                                                                   o.filePart == 1 &&
                                                                                                   o.plate == i);
                                        if (results.Count() > 0)
                                        {
                                            tbJobPages plate1 = results.First();
                                            firstPagePlate = results.First().pageOnFile;
                                        }
                                    }

                                    bool hasColor = sectionIsColor(jobTower, l, j);
                                    foreach (tbColors color in db.tbColors)
                                    {
                                        if (hasColor || color == colorBlack)
                                        {
                                            string filename = string.Format(fileSkeleton,
                                                                            firstPagePlate.ToString(pageformat,
CultureInfo.InvariantCulture),
                                                                            color.code);
                                            int couple = 0;
                                            if (!broadSheet)
                                                couple = (totalPages - pageCounter) + 1;


                                            tbJobPages newPage = new tbJobPages()
                                            {
                                                tbJobs = jobEdit,
                                                tbColors = color,
                                                tbTowers = jobTower.tbTowers,
                                                tbTowerSide = l,
                                                section = j,
                                                plate = i,
                                                tbJobPageStatus = defaultStatus,
                                                pageNum = pageCounter,
                                                coupleNum = couple,
                                                filePart = (j%2)+1,
                                                pageOnFile = firstPagePlate,
                                                fileNameEstimated = filename

                                            };
                                            jobEdit.tbJobPages.Add(newPage);
                                        }
                                    }

                                    pageCounter++;
                                }
```

```
                        }
                    }
                }
            }

        }
    }
    #endregion

}
```

Este método está basado en una cierta lógica que se cumple siempre en todas las rotativas y que determina por donde se imprimirá cada página en un trabajo. Se basa principalmente en el *Layout Order* establecido en los elementos plegadora y torre de la rotativa

Otras clases que implementan lógica son: *Msg2Srv*, *Utils*, *Validation* y *MyCommands*

3.4 Capa de presentación

Se ha realizado en *WPF* usando diseño sobre *XAML*. Los ficheros son:

VcCurversEdit.xaml, VcCurvesList.xaml, VcCurvesModify.xaml, VcFileSearch.xaml, VcFilesModify.xaml, VcFoldersModify.xaml, VcGroupModify.xaml, VcGroupsList.xaml, VcJobsAddTower.xaml, VcJobsList.xaml, VcJobsModify.xaml, VcJobsReport.xaml, VcLinesModify.xaml, VcPreferences.xaml, VcPressMachine.xaml, VcPublicationsModify.xaml, VcSysParam.xaml, VcTiffProfilesModify.xaml, VcUsersList, VcUsersModify.xaml, Login.xaml, MainWindow.xaml, WPFAboutBox1.xaml

Se ha usado un tema común para todos estos ficheros que modifican la apariencia y comportamiento de algunos objetos (aunque la mayoría no están redefinidos y por tanto se comportan como por defecto). El fichero *MainTheme.xaml*

Por último se ha usado como informe un objeto *rdlc* abierto por un objeto de la clase *reportViewer*. Este objeto inicialmente no está pensado para *WPF* sino para *Windows Forms* por lo que se ha tenido que insertar en una ventana *WPF* un objeto *Windows Form* y dentro de este el objeto *ReportViewer*.

3.5 Consideraciones

- Se ha implementado la funcionalidad multi-idioma usando *WPF Localize Extension*. Esta extensión me permite localizar textos en ficheros de recursos *resx*. Su funcionamiento es sencillo, lee el idioma actual de la aplicación y si existe un fichero *ResLang.XX.resx* donde *XX* es el código de dicho idioma se usa ese fichero, de lo contrario se usa *ResLang.resx* que es el de por defecto en inglés. Como principal ventaja que aporta que se puede ver en tiempo de diseño los textos ya extraídos de los ficheros de recursos y aplicarlos directamente en el fichero *XAML*. También me

permite la extracción desde código *c#*. Tiene control de errores si no se encuentra una traducción concreta en un diccionario, etc.

- La implementación por pestañas se ha hecho inspirado en programas como *Visual Studio* o *Notepad++*. Para ello ha sido necesario modificar por estilos el comportamiento, estructura y apariencia de los objetos *TabItem* de la zona de trabajo principal para poder añadirle un botón de cierre. Se ha diseñado para que el objeto hijo siempre sea un objeto *ViewControl*, lo que ha posibilitado que para el título del *TabItem* (*Header*) se haya podido hacer binding sobre la variable *title* que está siempre presente en objetos *ViewControl* y así delegar su gestión a la propia clase *ViewControl*. También esto ha servido para que se pueda llamar a métodos presentes en esta clase como los de guardar datos, control de cambios, etc.

- Se ha creado una clase específica *ViewControl* que gestione el titulo e implemente métodos necesarios para guardar y tratar formularios. Como se ha especificado en el punto anterior esto me ha servido para que aunque desconozcamos que formulario sea se pueda llamar siempre al mismo método guardar u acceder al título, etc.

- También se ha implementado un sistema de *popups* mostrando una capa semitransparente con la ventana. En un principio se pensó en *popups* comunes pero estos son estéticamente mejores y también eliminan ventanas simultáneas que pueden confundir al usuario. Esta clase es la clase *PopUpLayer*. También se ha implementado una pequeña animación para indicar al usuario que no puede realizar otra operación mientras el *popup* este activo.

- Se ha implementado una clase para guardar *Logs* llamada *ErrorLog*. Dicha clase está descargada de *codeproject.com* y está ligeramente modificada para que se ajuste a las necesidades de la aplicación. Es creada al iniciar la aplicación y se llama al método de escritura en fichero en el evento *OnFirstChanceException* guardando en el fichero *exceptions.xml* todos los errores.

- Se ha implementado también una barra de progreso circular en la clase *CircularProgressBar* para mostrar en los tiempos de carga. La implementación correcta sería realizando la operación deseada en otro hilo (creando otro *thread*, mediante el uso de *BackgroundWorkers*, etc.) dejando el hilo principal dedicado a animar esta barra circular. La realidad es que algunos puntos no está implementado así por cuestión de tiempo y se ejecuta en el mismo hilo que la operación provocando que no se produzca la animación.

3.6 Conclusiones de la implementación

Se ha empleado bastante tiempo desarrollando la capa de presentación y en el aprendizaje de *WPF*. *WPF* es una tecnología que proporciona más facilidades al desarrollador que *Windows*

Forms ya que permite el diseño de pantalla mediante programación XML, más directa y facilitando que el orden y la estructura sean más a gusto del desarrollador.
También se ha empleado bastante tiempo en el diseño de estilos sobre los objetos ya que el abanico de posibilidades es muy amplio y si no se sabe exactamente lo que se hace nunca se consigue lo esperado. Creo que el diseño de elementos en pantalla es sencillo, pero aplicar una hoja de estilos requiere de mayor tiempo de aprendizaje.

Por otro lado hubiera sido más correcto, como ya he comentado anteriormente, organizar un poco mejor la capa de lógica de negocio e incluso añadir alguna capa más.

Otro aspecto en el que se ha empleado bastante tiempo es el diseño y estructura de la base de datos, aspectos muy importantes en la lógica de negocio, que si bien no tiene dificultad técnica entrañan bastante dificultad lógica.

Por último en la implementación he detectado distintos fallos y carencias en el tratamiento de base de datos creado por *Linq to SQL* que he tenido que ir corrigiendo, así como distintos fallos y dificultades en el uso de *binding* sobre objetos.

También cabe destacar que aplicación se pensó en un principio para ser usada con un único *DataContext*, pero debido a la que es imposible separar operaciones y discriminar por ventanas o pestañas, se ha tenido que instanciar múltiples *DataContext*.

En resumen, las conclusiones generales de la implementación son que se ha consumido bastante tiempo en la capa de presentación y en la capa de datos, que WPF es una tecnología muy potente, que el uso de *binding* síncronos con colecciones algunas veces dan muchos problemas y que los *DataContext* creados por el *Linq to SQL* no son perfectos y tienen carencias y fallos que hay que conocer y salvar.

4. EVALUACIÓN DE COSTES

Ya que se trata de un proyecto real se evaluaran los costes en base a lo que realmente ha costado a la empresa.

El salario es aproximadamente de 13 € la hora. Teniendo en cuenta que cada día se trabaja 8 horas el coste estimado será el siguiente:

Nombre de tarea	Duración	Nombres de los recursos	Costo
GUI de sistema de preajuste para rotativas	**103 días?**		**10.920,00 €**
Inicio del proyecto	0 días	Javier Poza	0,00 €
Definición y planificación del proyecto	**15 días**		**1.664,00 €**
Definición y planificación del proyecto - Comienzo	0 días	Javier Poza	0,00 €
Recopilación de información	3 días	Javier Poza	312,00 €
Elaboración de plan de proyecto	9 días	Javier Poza	936,00 €

Instalación de software	1 día	Javier Poza	104,00 €
Estudio de software	3 días	Javier Poza	312,00 €
Entrega PEC1	0 días	Javier Poza	0,00 €
Ejecución del proyecto	**67 días?**		**7.176,00 €**
Análisis y diseño	**27 días?**		**2.808,00 €**
Análisis y diseño - Comienzo	0 días?	Javier Poza	0,00 €
Especificación requisitos	4 días	Javier Poza	416,00 €
Diagrama Entidad-Relación Base de datos	2 días	Javier Poza	208,00 €
Modelo relacional	0 días	Javier Poza	0,00 €
Diagrama de casos de uso	2 días	Javier Poza	208,00 €
Diagrama de clases	2 días	Javier Poza	208,00 €
Diseño protocolo comunicación	1 día	Javier Poza	104,00 €
Diseño de interfaz de usuario	13 días	Javier Poza	1.352,00 €
Elaboración documento PEC2	2 días	Javier Poza	208,00 €
Corrección documentación	1 día	Javier Poza	104,00 €
Entrega PEC2	0 días	Javier Poza	0,00 €
Implementación	**41 días?**		**4.368,00 €**
Implementación - Comienzo	0 días?	Javier Poza	0,00 €
Gestión de usuarios y autenticación	3 días	Javier Poza	312,00 €
Gestión de publicaciones	2 días	Javier Poza	208,00 €
Gestión de torres de impresión	4 días	Javier Poza	416,00 €
Gestión de parámetros del sistema	2 días	Javier Poza	208,00 €
Gestión de curvas	8 días	Javier Poza	832,00 €
Informes	2 días	Javier Poza	208,00 €
Estado del sistema y gestión de ficheros TIFF	15 días	Javier Poza	1.560,00 €
Mensajería con servidor	1 día	Javier Poza	104,00 €
Gestión del idiomas	2 días	Javier Poza	208,00 €
Puebas unitarias	2 días	Javier Poza	208,00 €
Pruebas de integración	1 día	Javier Poza	104,00 €
Entrega PEC3	0 días	Javier Poza	0,00 €
Cierre del proyecto	**20 días?**		**2.080,00 €**
Cierre del proyecto - Comienzo	0 días?	Javier Poza	0,00 €
Elaboración Memoria	11 días	Javier Poza	1.144,00 €
Correción memoria	1 día	Javier Poza	104,00 €
Elaboración presentación	7 días	Javier Poza	728,00 €
Corrección presentación	1 día	Javier Poza	104,00 €
Entrega final	0 días	Javier Poza	0,00 €

El coste final para la empresa será de 10.920 €.

5. TRABAJO FUTURO

La ampliación modificación más directa que existe es adaptarlo para usarlo con bases de datos de tipo *MySQL*. Esto es debido a que no siempre se va a poder disponer de un servidor *SQL* Server.
La mejor opción y que tiene compatibilidad casi total con *Linq to SQL* es una librería desarrollada por *http://www.alinq.org* llamada *Linq to MySQL*. Si usando esta librería generaramos clases con nombres similares no habría que modificar prácticamente nada pudiendo generar versiones compatibles para ambas plataformas.

Otra mejora sería el uso de una librería mejor que *WPFToolkit* para generar los gráficos. Por ejemplo *http://www.visiblox.com/* permite generar gráficos con curvas de Bezier y además consumen menos recursos.

Otra posible mejora es incluir mensajes informativos del resultado de las operaciones efectuadas.

Por último una idea que fue descartada debido a la falta de tiempo y a que no es necesaria (aunque útil) es un visor de *logs* del servidor. El servidor registrará sus acciones en un fichero *log* que puede estar compartido por *SMB*. La aplicación podría mostrar de forma organizada este fichero *log* y que automáticamente se vaya refrescando, así el usuario sabrá que está haciendo el sistema exactamente.

6. CONCLUSIONES

El desarrollo de esta aplicación me ha permitido aprender y conseguir mucha experiencia en la plataforma *.NET* de *Microsoft*.

Gracias a este proyecto he descubierto *Linq*, *Linq to SQL* y *WPF*.

Sobre *Linq* me parece una tecnología realmente buena. Me ha permitido realizar selecciones sobre colecciones de una manera muy intuitiva y eficiente, como lo haría con una consulta *SQL*. Además he probado bastantes selecciones con condiciones complejas, uniones, etc. sin haber mirado documentación en concreto previamente, simplemente probando y han funcionado perfectamente. Es realmente sencillo e intuitivo.

Sobre *Linq to SQL* me parece sorprendente. Normalmente creaba las clases y colecciones de forma manual. Cuando probé la tecnología *Linq to SQL* quede realmente sorprendido y aliviado, pues ya vi en ese momento la cantidad de carga de trabajo que me iba a quitar, además lo sencillo que era de utilizar.

He detectado bugs (muchos de ellos ya reportados a Microsoft) y en ocasiones ha realmente frustrante al tener problemas al realizar *binding*, pero en general he quedado muy satisfecho con esto.

Con *WPF* también he quedado gratamente sorprendido. Al principio resulto un tanto complicado y extraño, pero pronto conseguí ir dominándolo hasta el punto de preferirlo a *Windows Forms*. La curva de aprendizaje puede ser algo lenta ya que es un cambio muy brusco pero una vez llegas a cierto nivel es bastante rápido diseñar, más que con *Windows Forms*.

También me ha sorprendido el *IDE Visual Studio 2010*. Hasta ahora había probado otros IDES como *SharpDevelop*, *VS2002*, *Visual Studio 6*, etc. Realmente es un entorno muy amigable para el programador simplificando la tarea de *debug*. La gestión de herramientas, pestañas y menús es realmente buena.

Por último he mejorado mi técnica y metodología en *c#*, que unido a la plataforma *.NET* me parece lo mejor para desarrollar hoy en día.

En resumen puedo decir que la plataforma de desarrollo *VS2010 con .NET Framework 4.0* usando *WPF* es bastante productiva y eficiente una vez consigas:

- Defenderte con *WPF*.
- Utilizar *binding* correctamente (tanto desde *WPF* como c#)
- Conocer las deficiencias y carencias de *Linq to SQL* y como salvarlas

Estos tres puntos son clave para desarrollar un proyecto como este de una manera productiva.

7. BIBLIOGRAFIA

Libros:

- **Jordi Conesa Caralt, Àngels Rius Gavidia, Jordi Ceballos Villach, David Gañán Jiménez** (2010). *Introducción a .NET*. Editorial UOC.
- **Miguel Katrib Mora, Mario del Valle Matos, Iskander Sierra Zaldivar, Yamil Hernández Saá** (2010). *Windows Presentation Foundition*. Editorial Netalia.

Artículos electrónicos:

- **.NET Localization**: http://ondotnet.com/pub/a/dotnet/2002/09/30/manager.html
- **Validation in Windows Presentation Foundation:** http://www.codeproject.com/KB/WPF/wpfvalidation.aspx
- **Chart Controls for WPF ship in the Toolkit:** http://blogs.msdn.com/b/wpfsdk/archive/2009/06/25/new-chart-controls-for-wpf.aspx

- **WPF Validation Error Disappear Inside TabControl When Switching TabItems**: http://karlshifflett.wordpress.com/2008/02/19/wpf-validation-errors-disappear-inside-tabcontrol-when-switching-tabitems/
- **A Guided Tour of WPF:** http://www.codeproject.com/KB/WPF/GuidedTourWPF_1.aspx
- **WPF Basic Data Binding FAQ:** http://blogs.msdn.com/b/wpfsdk/archive/2006/10/19/wpf-basic-data-binding-faq.aspx
- **Data Binding in WPF:** http://msdn.microsoft.com/en-us/magazine/cc163299.aspx
- **Linq to SQL DataContext Lifetime Management:** http://www.west-wind.com/weblog/posts/2008/Feb/05/Linq-to-SQL-DataContext-Lifetime-Management
- **Using ObservableCollection with Linq:** http://jimlynn.wordpress.com/2008/12/09/using-observablecollection-with-linq/

Webs consultadas:

- **MSDN**: http://msdn.microsoft.com
- **Biblioteca UOC**: http://biblio.uoc.es
- **Soporte Microsoft**: http://support.microsoft.com