ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA DE TELECOMUNICACIÓ LA SALLE

TREBALL FI DE CARRERA

ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

Análisis y reingeniería de una plataforma de desarrollo y ejecución de aplicaciones de gestión.

ALUMNE

Silvia Mur Blanch

PROFESSOR PONENT

Miriam Cordero Delgado

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. *Silvia Mur Blanch*

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Análisis y reingeniería de una plataforma de desarrollo y ejecución de aplicaciones de gestión.

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

<< Don’t try.

Do. Or do not.

There is no try >>

Dedicado a mis padres, por todos
los motivos imaginables, y a Mari,
que no pudo convertirse en ingeniera.
Dejaste un inmenso vacío en nuestros
corazones. Jamás te olvidaré.

Agradecimientos.

Gracias en primer lugar a Miriam por haber sido una jefa de proyecto comprensiva a la par que exigente, cuando la situación lo requería. Gracias por haber dedicado tantas horas (y paciencia) a resolver mis dudas y a solucionar los problemas que surgieron durante el desarrollo del proyecto. Pero, sobretodo, gracias por haber confiado siempre, más incluso que yo misma, en mí y en mis capacidades.

Gracias a mis padres y a mi familia por haberme apoyado en todo momento, especialmente durante los primeros y más difíciles años de la carrera, animándome a seguir adelante pese a mis dudas.

Gracias a mis compañeros en el área de informática por ser el mejor equipo humano posible… y por los cafés a media mañana. Gracias en particular a los demás miembros del "Equipo A", porque este proyecto es también vuestro, y entre todos hemos trabajado muy duro para sacarlo adelante.

Gracias a mis amigos por haber sido mi vía de escape durante todos estos años en La Salle, porque no todo ha sido estudiar y hacer prácticas aunque parezca mentira.

Gracias especialmente a Christian, que empezó en este proyecto conmigo y juntos trabajamos en él durante su etapa más dura. Gracias por contagiarme tus ganas de aprender, por motivarme a estudiar más y por ser un compañero de trabajo excelente y mejor persona.

Gracias muy en particular a mi "Maestro Jedi", David Vernet, por ser un amigo antes que un profesor, porque siempre estuvo ahí para echarme una mano y tuvo fe en mí. Gracias por descubrirme dos de tus mayores pasiones, que ahora también lo son mías: la montaña y la programación.

Y por último, y no menos importante, gracias a George Lucas… por la Fuerza.

Abstracto.

Estudio de una plataforma E.R.P. obsoleta basada en sistemas de ficheros, para llevar a cabo la corrección de problemas de funcionamiento e incorporación de novedades en la misma. Implementación del conjunto de herramientas necesarias para migrar el sistema de ficheros original a una base de datos, y posterior diseño y desarrollo de una nueva plataforma, altamente dinámica y personalizable, y totalmente compatible con la inicial.

Resumen.

El objetivo de este Trabajo Final de Carrera consiste en el diseño y desarrollo de la nueva versión de una aplicación de planificación de recursos empresariales (E.R.P.) creada a principios de los años 80 y que, por lo tanto, se ha quedado obsoleta.

El proyecto ha constado de varias fases: en la primera se estudió a fondo la plataforma original, se solucionaron algunos problemas de funcionamiento a petición del cliente y se implementaron nuevas funcionalidades. Esta parte se desarrolló en Visual Studio 5 y 6 utilizando el lenguaje C.

El funcionamiento de la plataforma E.R.P. original está basado en sistemas de ficheros: ficheros de texto planos, binarios o encriptados, que tanto pueden definir el entorno de la plataforma como almacenar los datos de sus aplicaciones. La segunda fase consistió en implementar un conjunto de aplicaciones para llevar a cabo la migración del sistema de ficheros original a una base de datos. Esta ha sido la parte más complicada del proyecto, y ha significado el 50% del tiempo total invertido en la realización del mismo.

La última fase del proyecto ha consistido en diseñar e implementar el entorno de la nueva plataforma, desarrollada en Visual Studio .NET 2005 y utilizando el lenguaje C#. Se han aprovechado al máximo las posibilidades que ofrece este entorno de desarrollo para crear una aplicación muy dinámica y personalizable, con un estilo muy moderno y fácilmente adaptable al paso del tiempo, a la vez que totalmente compatible con su antecesora.

Capítulo 1.
Introducción.

Capítulo 1. Introducción.

El presente Trabajo Final de Carrera tiene como objetivo el desarrollo de una aplicación de gestión de uso comercial, destinada al mercado de las pequeñas y medianas empresas, y basada en un software ya existente.

El proyecto Axialsoft, que será como nos referiremos a este proyecto de ahora en adelante, aparece como partes de diversos proyectos, divididos éstos a su vez en distintas fases que podríamos clasificar, a grandes rasgos, en análisis, diseño e implementación, siendo el propósito principal del proyecto en sí el rediseño de un paquete E.R.P. obsoleto y su posterior desarrollo en Visual Studio .NET.

Los inicios del proyecto estuvieron dedicados al estudio del software original, el paquete E.R.P. *Axiwin*, propiedad de la empresa AxialSoft, S.L., la cual empezó a comercializarlo a principios de los años 80. La documentación que nos fue proporcionada era escasa e imprecisa, así que durante los primeros meses se comentó el código fuente y se generó mucha documentación sobre el funcionamiento de la plataforma.

Tras el primer contacto con la plataforma *Axiwin*, la empresa AxialSoft, S.L. proporcionó un listado de problemas de funcionamiento a solucionar, y otro de nuevas funcionalidades a incorporar al E.R.P. Se realizaron sendos análisis de viabilidad que recogían una estimación de la dificultad de implementar las modificaciones solicitadas al código fuente original, y el tiempo requerido para llevarlas a cabo; en muchos casos, determinadas tareas quedaron descartadas por uno de estos dos motivos. Tras la aprobación del cliente empezó la fase de corrección de errores e implementación de novedades, la cual fue ardua y extensa.

Pese a que el estudio teórico previo fue muy extenso, éste proyecto se centra en las fases de diseño y desarrollo más recientes y también, con especial hincapié, en la fase de migración que constituyó la transición entre “lo viejo” y “lo nuevo”.

Siendo el paquete E.R.P. *Axiwin* anterior a la actual y cada vez más extendida práctica de emplear bases de datos en el software de gestión, todo su funcionamiento se basa en ficheros. Dado que la utilización de bases de datos y la completa compatibilidad entre la plataforma *Axiwin* original y la nueva eran dos de los principales requisitos del proyecto, era prioritaria la implementación del software necesario para llevar a cabo la migración de datos.

Pese a que la migración del sistema de ficheros completo se realiza en una única operación y de forma transparente para el usuario, ésta se compone de distintos proyectos bien diferenciados, destinado cada uno de ellos a una parte específica de la migración. La parte que engloba el proyecto Axialsoft es la migración de pantallas y menús. Previo a explicar en qué consiste dicha migración debemos definir los conceptos de "pantalla" y "menú" en el contexto de la plataforma *Axiwin*.

La particularidad de *Axiwin* respecto a otros paquetes E.R.P. que se pueden adquirir en la actualidad es que la plataforma en sí es un autómata sobre el que se ejecutan aplicaciones de gestión desarrolladas por, y en un lenguaje propio de, AxialSoft, S.L. Dichas aplicaciones se componen, a grandes rasgos, de código y pantalla/s, de forma semejante a los formularios de Windows, ofreciendo así un entorno de interacción amigable al usuario. Las pantallas, originalmente, se diseñaban en modo texto, utilizando caracteres especiales para definir los bordes de los "formularios", y ésta información se guarda en ficheros de texto binarios, con la extensión .pan.

La migración de pantallas consistió en interpretar la información que contienen los ficheros .pan e introducirla en las tablas diseñadas específicamente a tal efecto en la base de datos, traduciendo la definición original de la pantalla en los controles equivalentes de los formularios de Windows *label* (etiqueta) y *textbox* (caja de texto). Paralelamente a la implementación del *traductor de pantallas* se implementó el *cargador de pantallas*, que inicialmente se utilizaba para comprobar que los resultados de la migración fuesen satisfactorios, generando formularios a partir de los datos

traducidos en la base de datos, y actualmente se utiliza el mismo código para cargar las aplicaciones migradas en la nueva plataforma.

El fin de la larga y complicada fase de migración supuso un punto de inflexión en el proyecto y cerró una etapa, dado que desde este momento todo empezaría a ser nuevo y estaría, en cierto modo, desvinculado de la plataforma *Axiwin* original. A continuación se llevó a cabo el análisis funcional de la nueva plataforma, durante el cual se diseñaron los formularios principales de la misma y se definieron las líneas generales de funcionamiento. Casi paralelamente al análisis funcional se realizó un primer análisis orgánico de corto alcance, que se limitaba a las tablas inmediatamente necesarias, sin contemplar las necesidades de las futuras fases de desarrollo de la plataforma. Para atenderlas se hizo un segundo análisis orgánico, más completo.

La nueva plataforma *Axiwin* está dividida, de forma bien diferenciada, en dos grandes partes:

- ✓ La *plataforma de cliente* es el paquete E.R.P. en sí que la empresa AxialSoft, S.L., va a comercializar. Tiene todas las funcionalidades que ofrecía el E.R.P. original e incluye muchas novedades, entre ellas la posibilidad de personalizar el aspecto de la plataforma casi por completo. El funcionamiento está enteramente ligado a una base de datos y la carga de aplicaciones es totalmente dinámica.

- ✓ La *plataforma de administrador* la componen un amplio conjunto de aplicaciones que AxialSoft, S.L., utiliza para modificar la plataforma de cliente, añadiendo, quitando o cambiando aspectos de la misma, permitiéndole también generar paquetes de instalación a medida para sus clientes. Entre este conjunto de herramientas del administrador se encuentran el editor de estilos (colores) de la plataforma, el editor de iconos (generador de recursos de imágenes), el editor de idiomas, el editor de formularios (pantallas), el editor de código y su compilador, etc.

Este proyecto engloba la creación del entorno funcional de las plataformas de cliente y administrador, sus principales formularios y algunas de las aplicaciones de la plataforma del administrador, así como el diseño de buena parte de las tablas que constituyen la base de datos.

La plataforma cliente guarda un gran parecido, no tanto estético como funcional, con la plataforma original, en primer lugar porque era un requerimiento del usuario, y en segundo lugar porque esto facilitará la adaptación a los usuarios del E.R.P. *Axiwin* predecesor. En la creación de los nuevos formularios se ha buscado aprovechar al máximo las posibilidades que ofrece la plataforma de desarrollo Visual Studio .NET, apostando por diseños atractivos e incorporando los detalles habituales de la gran mayoría de aplicaciones existentes hoy en día para el sistema operativo Windows, dado que la afinidad con dicho sistema operativo facilitará el aprendizaje a los nuevos usuarios del E.R.P. *Axiwin*.

Esta parte del proyecto fue la más agradecida y la más vistosa, y además nos dio la posibilidad de trabajar por primera vez con la versión más reciente de la plataforma de desarrollo Visual Studio .NET (*Microsoft Visual Studio .NET 2005*), que ofrece total libertad al programador para personalizar y dinamizar al máximo sus aplicaciones. Los más claros ejemplos de aplicación de las múltiples posibilidades de la plataforma .NET los veremos en dos de las aplicaciones de la plataforma de administrador, el *generador de estilos* (*StyleGen*) y el *cargador de estilos* (*StyleLoader*).

El *cargador de estilos* es el encargado de aplicar, a todos los formularios de las plataformas de administrador y cliente, las mismas propiedades predefinidas a los controles más comunes, entre los que se incluyen las etiquetas, las cajas de texto, los botones, etc. El conjunto de propiedades editables de cada control y sus valores se especifica mediante una *hoja de estilos*, que no es más que un fichero .xml en el que cada control está representado por una tabla de datos. La elección del estilo que deberá mostrar la plataforma al ejecutarse, así como la creación y edición de *hojas de estilos*, se realiza desde el *generador de estilos*.

Dado que las *hojas de estilos* tienen una estructura muy sencilla, sería posible modificarlas y crear nuevos estilos empleando directamente un editor de textos, pero el *generador de estilos* ofrece al usuario un entorno de trabajo muy amigable y sencillo de utilizar, que le permite además ver en todo momento el resultado de su labor, obteniendo así una idea aproximada de cómo quedará el estilo una vez se aplique a los formularios de las plataformas de administrador y cliente.

El *generador de estilos* se complementa con el editor de iconos, o *generador de recursos de imágenes* (*ImageResourceCreator*). Desde esta otra herramienta del administrador es posible modificar todos los iconos de las plataformas de administrador y cliente, creando diferentes paquetes de imágenes que posteriormente se asignan a las hojas de estilos, homogeneizando el resultado visual final (p.ej. podemos crear un estilo con distintos tonos de color azul y asignarle un paquete de iconos a conjunto).

La última parte de este proyecto se centra en la implementación del control de acceso a la plataforma de cliente mediante perfiles y permisos de usuario. Esta fue la parte más difícil de diseñar tanto en el análisis orgánico como en el análisis funcional, e igualmente difícil de implementar, dado que tiene mucha interacción con la base de datos, siendo necesario consultar en todo momento, durante una misma sesión, la información contenida en múltiples tablas para autorizar, o no, a los usuarios a utilizar las distintas aplicaciones y herramientas de la plataforma de cliente.

Veremos cómo los permisos por usuario se establecen mediante diversas capas con distintas jerarquías, pudiendo restringir el acceso no sólo a módulos completos y aplicaciones concretas, sino también a campos específicos de cada aplicación. El primer filtro de usuarios lo constituyen los perfiles de acceso y, posteriormente, los permisos se asocian a menús enteros, los cuales se asignan a los usuarios mediante el *editor de menús*, que es otra de las herramientas del administrador.

Los menús definen las opciones que estarán disponibles en la plataforma de cliente para cada módulo de aplicaciones, independientemente de la cantidad total de

aplicaciones existentes en la base de datos. Las aplicaciones que consten en un menú se mostrarán en el árbol explorador del formulario principal de la plataforma de cliente, de modo que, asignando posteriormente los menús a cada usuario, conseguimos limitar su acceso únicamente a las aplicaciones que éstos pueden ver.

Cabe destacar también que en el transcurso de la realización de este proyecto se ha trabajado con 2 lenguajes de programación, C y C#, y hasta 4 plataformas de desarrollo distintas, a saber:

- ✓ Microsoft Visual C++ 5, lenguaje C
- ✓ Microsoft Visual C++ 6, lenguaje C
- ✓ Microsoft Visual Studio .NET 2003
- ✓ Microsoft Visual Studio .NET 2005

Capítulo 2.
Estudio teórico.

Capítulo 2. Estudio teórico.

2.1. Plataforma y lenguaje de desarrollo.

Como hemos visto en el *Capítulo 1. Introducción*, éste proyecto tuvo una primera y extensa fase de estudio de la plataforma software original, el paquete E.R.P. *Axiwin*, que posteriormente incluyó detección y corrección de errores e inclusión de nuevas funcionalidades. En esta primera fase se trabajó sobre el código fuente original del E.R.P., motivo por el cual podríamos decir que, tanto la plataforma de desarrollo como el lenguaje de programación, nos fueron impuestos.

Las fases en las que se centra este proyecto abarcan el diseño e implementación del nuevo E.R.P. *Axiwin*, desarrollado íntegramente sobre la plataforma Visual Studio .NET de *Microsoft*. La elección de esta plataforma de desarrollo se debe atribuir, inicialmente, al Área de informática de Transferencia de Tecnología La Salle, dado que tiene mucha experiencia colectiva acumulada en la utilización de Visual Studio .NET, gracias a los numerosos proyectos realizados con esta plataforma hasta el momento. En este caso, además, el cliente (AxialSoft, S.L.) no tenía preferencias y se dejó aconsejar. Visual Studio .NET es, además, la mejor opción para desarrollar aplicaciones para Windows, ya que el *framework* (máquina virtual) sobre el que se ejecutan sus aplicaciones viene ya integrado en el mismo sistema operativo. Se empezó a trabajar en el proyecto Axialsoft con la versión 2003 de VSNET, y posteriormente fue el cliente quien pidió expresamente que se utilizase la versión más reciente de la plataforma, VSNET2005, cuando ésta prácticamente acababa de salir al mercado. Esto conllevó algunos problemas de compatibilidad entre versiones, al tener que coexistir durante un tiempo los formularios implementados originalmente en VSNET2003 y sus correspondientes versiones migradas a VSNET2005.

En cuanto a la elección de C# como lenguaje de programación fue también el Área de informática de Transferencia de Tecnología La Salle la responsable, por motivos parecidos a los que motivaron la elección de la plataforma de desarrollo Visual Studio .NET. Visual Basic habría sido el otro lenguaje a tener en cuenta, pero se podría

decir que, prácticamente, ni se planteó tal posibilidad, siendo C# el lenguaje escogido desde el primer momento.

C# es un lenguaje muy actual que gana popularidad rápidamente porque combina las mejores cualidades de los lenguajes de programación orientados a objetos y de los lenguajes de programación orientados a eventos. Destaca también C# por su dinamismo, porque cuenta con un interminable repertorio de clases ya implementadas que sigue creciendo día a día, y porque es muy fácil aprender a utilizarlo teniendo conocimientos previos básicos de programación orientada a objetos.

2.2. Sistema Gestor de Base de Datos.

El sistema gestor de base de datos escogido es SQL Server. En las primeras fases de desarrollo de la nueva plataforma *Axiwin* se utilizó SQL Server 2000, y en las más recientes se empezó a trabajar con SQL Server 2005, concretamente con la versión SQL Server Express, que viene ya integrada en Visual Studio .NET 2005.

SQL Server Express es un gestor de base de datos gratuito basado en SQL Server 2005. La principal ventaja que ofrece respecto a SQL Server 2005 es una mayor facilidad de uso, lo que a su vez permite una instalación y configuración más rápidas en los sistemas destino. Por otra parte, al ser una versión gratuita, es algo limitada respecto a SQL Server 2005 en cuanto al número de conexiones simultáneas, pero aún así es suficiente para gestionar la extensa base de datos de la plataforma *Axiwin* sin problemas de fluidez ni de tiempo de acceso.

Dada la magnitud del proyecto que se ha llevado a cabo, la elección del sistema gestor de base de datos era muy importante. Se ha escogido SQL Server porque cumplía los siguientes requisitos:

- ✓ Utilización de *stored procedures*.
- ✓ Sencilla generación y recuperación de *backups*.
- ✓ Total integración con la plataforma de desarrollo. SQL Server está especialmente indicado para trabajar con Visual Studio .NET, dado que ambos productos proceden del mismo fabricante.

A decir verdad, prácticamente no se plantearon otras posibilidades en cuanto a qué sistema gestor de base de datos utilizar, pero habrían sido igualmente válidos:

- ✓ *Oracle* está considerado como uno de los SGBD más completos, destacando su estabilidad, su escalabilidad y el soporte de transacciones. En contra tiene el coste de adquisición, que es muy elevado. Hasta hace poco *Oracle* dominaba el mercado de los SGBDs, actualmente SQL Server y algunos SGBDs de licencia libre, como MySQL o PostgreSQL, representan su más seria competencia.

- ✓ *Informix*, propiedad de IBM desde 2001, es también un SGBD muy completo, especialmente indicado para aplicaciones de *Datawarehouse* y *Datamining*. Su coste de adquisición es igualmente elevado, aunque no tanto como *Oracle*.

- ✓ *MySQL* es el SGBD de código abierto más popular del mercado actualmente; entre sus usuarios más destacados se encuentran *Google*, *Yahoo!*, *Amazon* e incluso la *NASA*. Mientras que, en sus inicios, *MySQL* era un SGBD gratuito, hoy en día es de pago, aunque mucho más económico que la gran mayoría de SGBDs.

2.3. Estudio de mercado.

Hasta la realización de este proyecto del concepto de E.R.P. conocíamos únicamente su nombre y que se refiere a un tipo de software, pero no de qué tipo de software se trataba exactamente, para qué se utiliza, qué tipo de usuarios lo utiliza, etc. E.R.P. son las siglas de *Enterprise Resource Planning* (Planificación de recursos empresariales), y se trata de un software muy completo con el que se puede manejar la producción, logística, distribución, inventario, envíos, facturas, contabilidad, ventas, pagos, producción, administración de inventarios, etc. de una empresa.

Este conjunto de tareas, tan común a todas las empresas, sean grandes, medianas o pequeñas, se ha llevado a cabo durante años siguiendo metodologías propias de cada empresa, en muchos casos empleando el formato papel, en otros casos utilizando aplicaciones software desarrolladas a medida por y para cada empresa.

Los sistemas E.R.P. suponen una ventaja respecto al software a medida principalmente porque son aplicaciones de fácil implantación que se pueden adaptar a las necesidades específicas de cada empresa de forma sencilla ya que, por lo general, se trata de software altamente parametrizable. Los E.R.P.'s ofrecen además herramientas para calcular estadísticas a partir de un conjunto de datos, aportando a la empresa información sobre su desarrollo y ayudándola así a tomar decisiones.

Actualmente hay una gran cantidad de empresas desarrolladoras de software que distribuyen sistemas E.R.P., y éstos se pueden clasificar principalmente por el precio y por el tipo de empresas al que están destinados: las empresas muy grandes necesitarán E.R.P.'s igualmente grandes y caros, que garanticen la cobertura de todas las necesidades de la empresa, mientras que las pequeñas y medianas empresas satisfarán sus necesidades con E.R.P.'s más sencillos y más económicos.

Los sistemas E.R.P. más populares del mercado en la actualidad, destinados a grandes empresas son:

- ✓ *SAP*. Este E.R.P. está considerado hoy en día como "el número 1", además de ser uno de los pioneros de este tipo de software. *SAP* es el sistema E.R.P. más grande, más caro y más vendido en todo el mundo, pero es igualmente complicado de implantar y utilizar. La empresa que lo comercializa, SAP AG, es la empresa desarrolladora de software más grande de Europa, y la tercera del mundo.
- ✓ *Oracle Applications,* de Oracle Corporation, que también comercializa, entre otros, los sistemas gestores de bases de datos *Oracle*.
- ✓ *Microsoft Dynamics,* de Microsoft.
- ✓ *Lawson Software*.
- ✓ *Visma*.

Otros sistemas E.R.P. que podríamos considerar más modestos, destinados a pequeñas y medianas empresas, son:

- ✓ *SP Contaplus,* de la empresa española Sage SP.
- ✓ *LogicSoft.*
- ✓ *Axiwin,* de la empresa AxialSoft, S.L., que se empezó a comercializar a principios de la década de los 80, y en cuyo rediseño y posterior desarrollo se basa este proyecto.

2.4. El E.R.P. Axiwin.

Pese a que el objetivo principal del proyecto Axialsoft siempre fue el diseño y posterior desarrollo del nuevo E.R.P. *Axiwin*, ésta fase sólo ha significado un 50% del tiempo total invertido en este proyecto. A continuación detallamos en qué consistieron las distintas fases dedicadas al estudio de la plataforma software original.

- ***Estudio y documentación del E.R.P.***

La documentación del E.R.P. *Axiwin,* proporcionada por el cliente junto con el código fuente original, era muy escasa e incompleta. El código fuente, implementado en lenguaje C, está formado por múltiples proyectos y centenares de archivos, muchos de los cuales sobrepasan las 5000 líneas de código. Durante la primera fase de este proyecto se revisaron y documentaron todos los archivos del código fuente original, uno a uno. La gran mayoría de las funciones originales tienen nombres poco intuitivos, de modo que fue necesario ayudarse de las pruebas de ejecución y *debug* para descubrir la finalidad las mismas. Hasta después de la fase de documentación no se implementó ni una sola línea de código.

Esta parte del proyecto fue muy dura y pesada, ya que hubo que trabajar con un código ajeno, mal estructurado, poco óptimo, muy complicado y bastante ofuscado. Por otra parte fue de gran ayuda para las fases posteriores conocer la plataforma original a fondo.

- ***Resolución de problemas e incorporación de novedades en el E.R.P. original.***

En los 20 años transcurridos desde que la empresa AxialSoft, S.L. empezó a comercializar el E.R.P. *Axiwin*, sólo se había modificado el código original en dos ocasiones para introducir mejoras. Aunque la intención del cliente era rediseñar la plataforma software original e implementarla de nuevo, había errores importantes en la misma que era preciso corregir, ya que AxialSoft, S.L. cuenta con muchos clientes que utilizan la plataforma día a día.

El cliente proporcionó un listado de problemas de funcionamiento y mejoras solicitadas y a partir de éste se llevaron a cabo el *Análisis de viabilidad de la resolución de problemas* y el *Análisis de viabilidad de la incorporación de novedades*.

El *Análisis de viabilidad de la resolución de problemas* contempla, básicamente, la estimación del tiempo necesario para solucionar los errores de funcionamiento de la plataforma *Axiwin*, mientras que el *Análisis de viabilidad de la incorporación de novedades* consideraba la posibilidad o imposibilidad de implementar las mejoras solicitadas al código fuente original, así como una estimación del grado de dificultad de cada tarea y el tiempo necesario para llevarlas a cabo.

A continuación se comentan algunos de los puntos más importantes del *Análisis de viabilidad de la incorporación de novedades*.

- ✓ *Corrección del control de versiones DEMO.* La empresa AxialSoft, S.L. quería ofrecer la posibilidad de adquirir una versión *DEMO* del E.R.P. *Axiwin*, limitando el uso de la aplicación no a un período de tiempo concreto, como es lo habitual, sino a un máximo de registros por aplicación.

- ✓ *Ampliación de instrucciones del lenguaje propio de Axiwin*. Se deseaba corregir y/o ampliar el funcionamiento de algunas de las instrucciones del lenguaje de programación propio de AxialSoft, S.L.

- ✓ *Iconografía en menús*. La aplicación *Axiwin* original sólo utiliza dos tipos de iconos (libro abierto, libro cerrado) para distinguir las carpetas de las aplicaciones en el menú principal. Se quería añadir un control para asignar iconos específicos a cada entrada menú.

- ✓ *Creación de grupos de listados*. Con la implementación de esta funcionalidad se podrían agrupar los listados según los parámetros deseados por el usuario, pudiendo tenerlos organizados de forma más ordenada e intuitiva.

- ✓ *Ampliación de la pantalla de la aplicación.* La plataforma original muestra un tamaño de pantalla de 640 x 480 lo cual resulta bastante incómodo, ya que el área de trabajo es muy reducida. Se deseaba crear un control de autoconfiguración del tamaño según la configuración local de Windows, aumentando la pantalla hasta 800x600 y 1024x768 píxels.

Las fases de resolución de problemas e incorporación de novedades quedan fuera del ámbito de este proyecto, que se centra en las distintas fases que han compuesto el desarrollo del nuevo E.R.P. *Axiwin*, y por este motivo no se explicarán con más detalle.

- ➢ ***Migración de pantallas y menús.***

La migración de pantallas y menús era uno de los pasos previos imprescindibles a la creación de la nueva plataforma *Axiwin*. Mientras que el diseño y la implementación de las herramientas necesarias para llevar a cabo la migración se detalla en el análisis funcional de este proyecto, fue preciso realizar un estudio teórico previo de los datos de origen que se iban a tratar en la migración.

La definición de las pantallas de las aplicaciones del E.R.P. se guarda en ficheros de texto binarios con extensión .pan, cuya estructura es bastante complicada. Para entender cómo se leen y cómo se escriben los datos en dichos ficheros fue necesario ejecutar el código fuente del generador *Axigen* (el editor de aplicaciones de la plataforma *Axiwin* original) en modo *debug*, para seguir el proceso detenidamente.

Posteriormente, en la fase de migración, ante la evidente dificultad de adaptar el proceso de lectura de los ficheros .pan al lenguaje C#, se utilizó el código original añadiendo las modificaciones pertinentes para tratar los datos a medida que se leían de fichero y se introducían en la base de datos.

Capítulo 3.
Análisis de especificaciones.

Capítulo 3. Análisis de especificaciones.

3.1. Objetivos y alcance del proyecto.

El objetivo principal de este proyecto ha sido diseñar e implementar una nueva versión del paquete E.R.P. *Axiwin*, comercializado por la empresa AxialSoft, S.L., cuya versión original está ya obsoleta. El proyecto se ha llevado a cabo a petición (y bajo la estricta supervisión) de la empresa, que pretende con el nuevo E.R.P. no sólo mantener a sus actuales clientes sino también abrirse a un abanico de mercado más amplio y captar nuevos clientes.

El Área de informática de Transferencia de Tecnología La Salle, encargada de realizar el proyecto, orientó a la empresa AxialSoft, S.L. en términos de plataforma de desarrollo y lenguaje de programación a utilizar. Los requisitos primordiales del cliente eran que el nuevo software funcionase íntegramente utilizando una base de datos, y que fuera totalmente compatible con el paquete E.R.P. original, cuyos datos se distribuyen en un complejo sistema de ficheros.

El alcance de este proyecto comprende desde la fase de migración, previo estudio del sistema de ficheros y el funcionamiento de la plataforma *Axiwin* original, hasta la implementación del nuevo E.R.P.

La migración del sistema de ficheros está dividida en pequeños proyectos, cada uno encargado de migrar un tipo de ficheros específico:

- ✓ Ficheros de datos .dat, ficheros de estructura de los datos .ddx, ficheros de campos .mas y ficheros de índices de los datos .ixx.
- ✓ Ficheros de informes .inf, .lis y .prc.
- ✓ Ficheros de código fuente .cal y ficheros de código compilado .prc y .tab.
- ✓ Ficheros de menú .men.
- ✓ Ficheros de configuración de inicio .ini.
- ✓ Ficheros de pantallas .pan.

Este proyecto se centra en la migración de los ficheros de pantallas .pan. La fase de implementación de la aplicación de migración de pantallas y el estudio de la plataforma software original representan el 50% de este proyecto.

La fase de desarrollo del nuevo E.R.P. *Axiwin* se dividió también en partes bien diferenciadas, implementadas por separado aunque paralelamente, distribuyendo las tareas a realizar entre los distintos técnicos que han formado el grupo de trabajo, según el proyecto de migración de ficheros llevado a cabo por cada uno de ellos en la fase anterior. Los proyectos en los que se descompuso la fase de desarrollo son:

- ✓ Implementación del editor y compilador del código fuente propio de AxialSoft, S.L., y del autómata (intérprete) que lo ejecuta.
- ✓ Implementación del editor de pantallas (formularios).
- ✓ Implementación del generador de listados y plantillas de listados.
- ✓ Implementación del entorno del nuevo E.R.P., del generador de estilos y el generador de recursos de imágenes, así como del editor de menús, editor de perfiles y de editor de permisos de acceso por usuario. Este último conjunto de tareas son las que ha abarcado el presente proyecto.

El desarrollo del entorno de la plataforma empezó con la generación y diseño de los formularios principales; en esta fase apenas se implementó código. Posteriormente se perfiló con más detalle el diseño orgánico de la base de datos a medida que se implementaba código, se añadían nuevos formularios, y los distintos proyectos que componen la plataforma se integraban en una única solución.

3.2. Arquitectura.

El nuevo E.R.P. *Axiwin* se ha desarrollado íntegramente en Visual Studio .NET y utiliza el sistema gestor de bases de datos SQL Server, ambas aplicaciones de Microsoft. Dado que es necesario tener instalada la máquina virtual o *framework* de Microsoft para poder ejecutar las aplicaciones Windows desarrolladas en .NET, esto limita los sistemas operativos compatibles con la nueva plataforma a versiones de Windows posteriores a Windows 95.

- ***Capas funcionales de la plataforma.***

La plataforma *Axiwin* se compone de múltiples proyectos los cuales, en su mayor parte, fueron implementados independientemente unos de otros e integrados posteriormente. Para trabajar más cómoda y ordenadamente, agilizando a la vez el acceso a la base de datos, cada proyecto se ha implementado utilizando *capas* de funcionalidades. Así pues, se puede representar la plataforma mediante el siguiente diagrama:

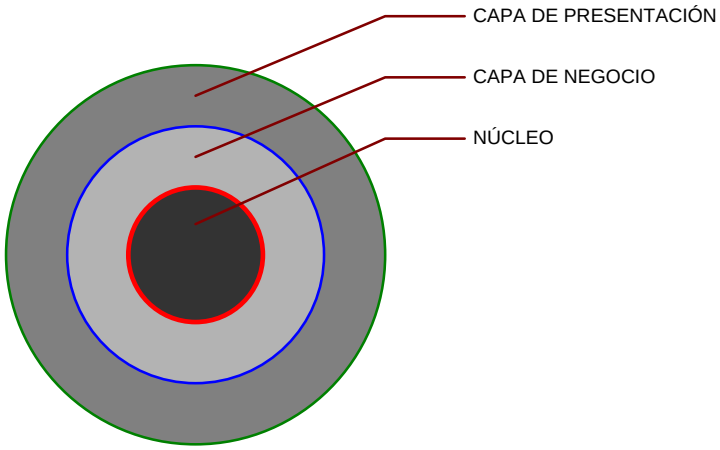


***Figura 3-1**. Diagrama de capas funcionales de la plataforma.*

- ✓ *Capa de presentación*. Se refiere a la implementación de los formularios de la plataforma, es decir, de la interacción con el usuario. También se incluye en esta capa todo lo relacionado con la carga de estilos, imágenes, idiomas, etc.

- ✓ *Capa de negocio*. La nueva plataforma *Axiwin* es totalmente dinámica, motivo por el cual la interacción con la base de datos es muy intensiva. Todos los formularios, en un momento u otro, deben acceder a la base de datos, ya sea para hacer consultas, altas, bajas o modificaciones. Se crearon unas clases genéricas para conectar con la base de datos y ejecutar los comandos SQL habituales; estas clases se utilizan para implementar las capas de negocio de cada proyecto involucrado en la plataforma *Axiwin*. La capa de negocio consiste en implementar una clase destinada, única y exclusivamente, a interactuar con una o varias tablas de la base de datos. La intención es conseguir evitar la replicación de código y simplificar la forma de acceder a un conjunto específico de datos de la base de datos.

- ✓ *Núcleo*. Las aplicaciones del E.R.P. se componen, básicamente, de *código* y *pantallas*. El código está programado en un lenguaje propio de la empresa AxialSoft, S.L., que está compuesto por un conjunto de más de 200 instrucciones que permiten implementar las aplicaciones de forma rápida y sencilla. Existe por lo tanto un compilador para este lenguaje, que genera no un código ensamblador sino un código intermedio, que es lo que la plataforma *Axiwin* interpreta y ejecuta. Como ya se ha comentado, la compatibilidad entre la plataforma anterior y la actual debía ser total, motivo por el cual el compilador ha debido ser rediseñado e implementado íntegramente en lenguaje C#, añadiendo nuevas instrucciones al conjunto original, y ampliando las funcionalidades de las ya existentes. El editor de código y el precompilador son dos herramientas de la plataforma de administrador, mientras que la plataforma de cliente se puede considerar, en sí misma, el intérprete sobre el que ejecutar las aplicaciones de gestión del E.R.P. Este proyecto contempla el diseño y desarrollo del entorno del intérprete y de sus funcionalidades y herramientas estáticas, mientras que el desarrollo del compilador forma parte de otro proyecto. Ambos proyecto se desarrollaron de forma independiente y en paralelo, integrándose posteriormente en una única solución, que es la nueva plataforma *Axiwin*.

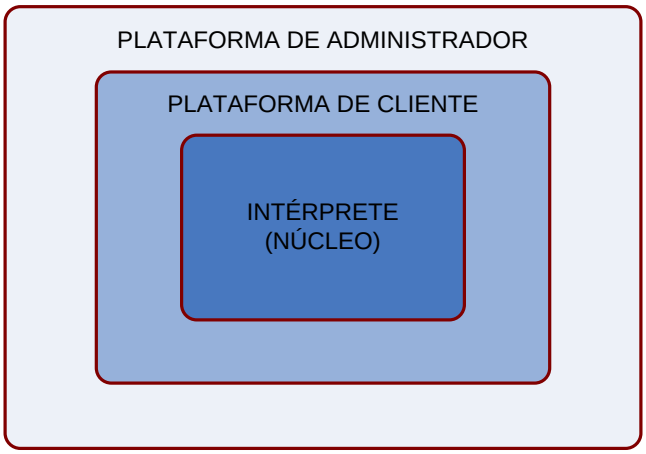


***Figura 3-2**. Diagrama de capas del E.R.P. Axiwin.*

- ***Plataforma de administrador.***

Ésta es la versión *"*extendida*"* de la plataforma *Axiwin,* de la que únicamente dispone la empresa AxialSoft, S.L., y que sus programadores utilizan para diseñar y crear aplicaciones, pantallas, estilos, paquetes de iconos, perfiles de acceso, etc. mediante las herramientas descritas en el *Capítulo 4. Análisis funcional*. También disponen de las herramientas para generar los paquetes de instalación del E.R.P. *Axiwin* a medida de cada cliente (*AxiClient*), y los paquetes de actualización del mismo (*AxiUpdate*). Desde la plataforma de administrador es posible también ejecutar la plataforma de cliente.

- ***Plataforma de cliente**.*

Es el entorno en el que se ejecutan las aplicaciones de gestión del E.R.P. *Axiwin*, que se cargan dinámicamente de la base de datos. Gracias al editor de paquetes de instalación de la plataforma de administrador, el cliente final adquiere una versión del E.R.P. generada a su medida, que contiene exclusivamente los módulos contratados a AxialSoft, S.L., y el conjunto de herramientas para personalizar el aspecto de la plataforma (estilos, idiomas, botonera principal, etc.).

3.3. Requisitos básicos.

3.3.1. Presentación.

Uno de los aspectos más cuidados en el desarrollo de la nueva plataforma *Axiwin*, a petición explícita del cliente, fue la presentación. Es bien sabido que la primera impresión es muy importante, y esto se aplica especialmente al software: el aspecto visual de una aplicación debe ser atractivo a simple vista.

Hoy en día es muy fácil adivinar la "edad" de una aplicación por su presentación, el estilo de sus pantallas e iconos. Con los años la plataforma *Axiwin* anterior había dejado de ser atractiva en un mercado en el que lo que más se vende es lo que entra por los ojos. Por este motivo el objetivo principal a la hora de diseñar el estilo general de la nueva plataforma era hacerlo fácilmente adaptable al paso del tiempo y muy moldeable.

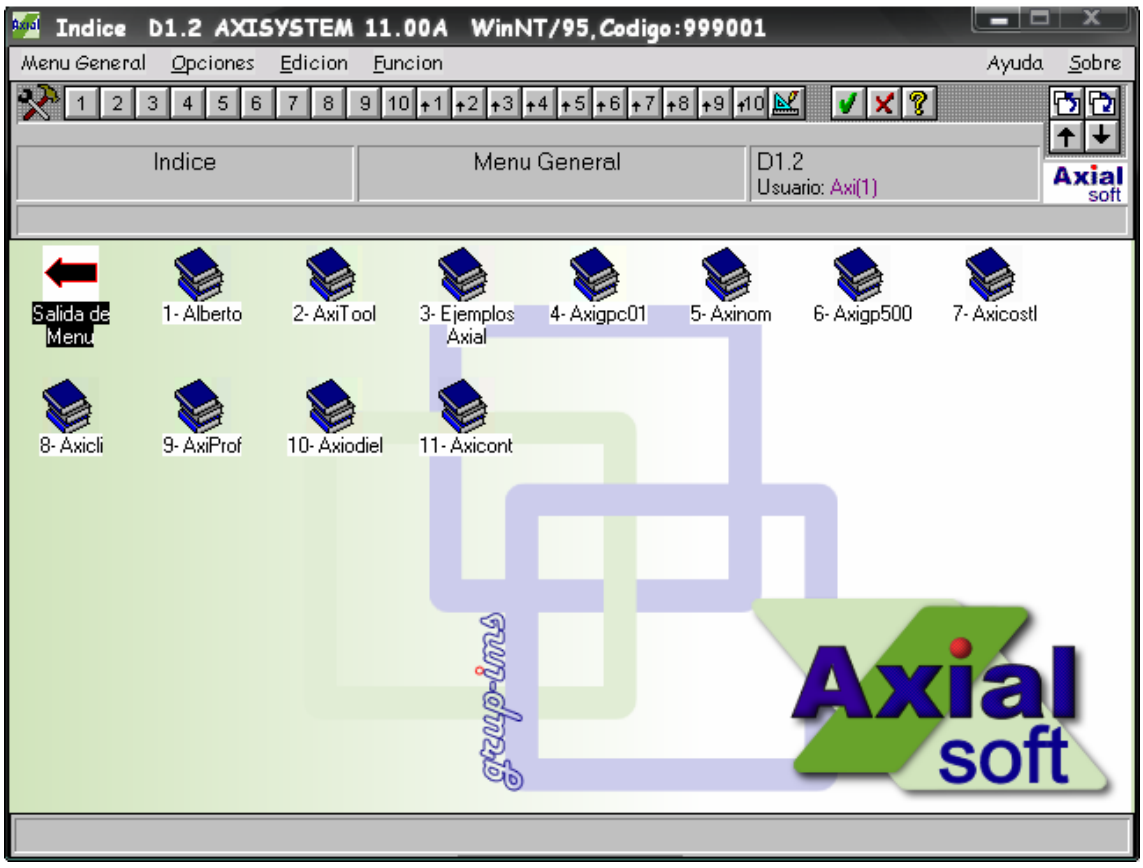


***Figura 3-3**. Pantalla principal de la plataforma Axiwin original.*

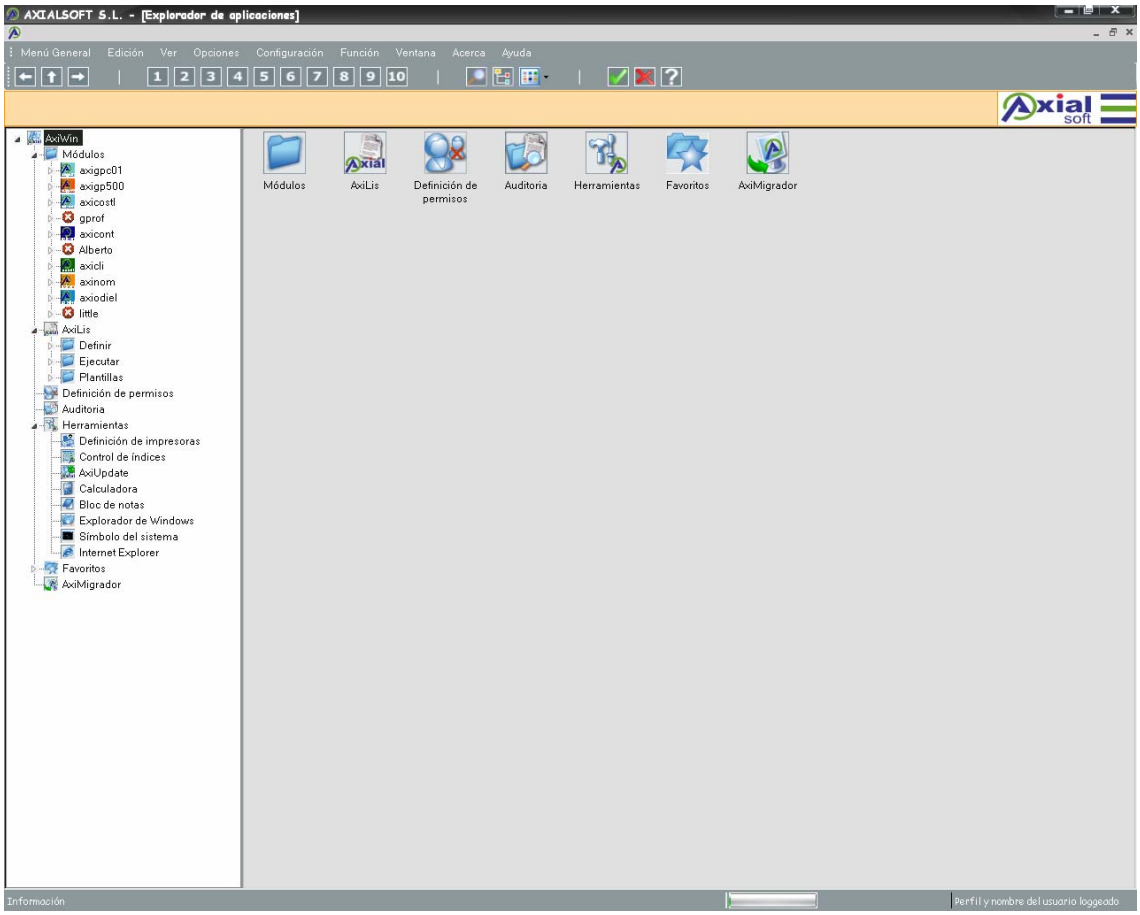


***Figura 3-4**. Pantalla principal de la nueva plataforma Axiwin.*

- ***Hojas de estilos y recursos de imágenes.***

Para poder cargar el aspecto del E.R.P. de forma dinámica, según unos parámetros preestablecidos, se implementaron las *hojas de estilos*. La carga de estilos es un proceso que lee de un fichero .xml (la *hoja de estilos*) la combinación de valores que deben tomar las propiedades estéticas de los controles de un formulario, y aplica éstas a todos los controles de todos los formularios. Esto facilita además la tarea de diseñar formularios, ya que no es necesario editar las propiedades de antemano, pudiendo centrar la atención en la implementación de código.

El generador de estilos es la herramienta que permite crear nuevas *hojas de estilos*, distintas "pieles" para el E.R.P. *Axiwin* que se pueden cargar de forma muy sencilla desde la misma plataforma. Los estilos se complementan con los recursos de imágenes para crear un aspecto visual totalmente dinámica.

Las imágenes de la nueva plataforma (iconos, papeles tapiz, etc.) se cargan dinámicamente mediante los *recursos de imágenes*. Los archivos de recursos son parecidos a archivos de paquetes (.tar), en cuanto a que sirven para "empaquetar" imágenes (y cualquier otro tipo de archivos) en un formato no comprimido. Pero son también archivos encriptados que sólo se pueden leer por código utilizando las clases pertinentes, lo cual impide el libre acceso a las imágenes y, por consiguiente, que éstas puedan ser eliminadas, modificadas o corrompidas.

El generador de recursos de imágenes permite no sólo crear nuevos archivos de recursos sino también modificar los archivos existentes (quitar o añadir imágenes). Los *recursos de imágenes* van ligados a las *hojas de estilos*, y ambos se asocian mediante el generador de estilos. Cada estilo tiene asignado un archivo de recursos de imágenes, de forma que se puedan combinar las propiedades definidas para los controles con iconos diseñados a conjunto.

- ***Creación de nuevos controles.***

Los requisitos del cliente respecto a desarrollar una aplicación altamente personalizable se referían, sobretodo, a ofrecer la posibilidad de modificar las propiedades visibles de los controles. Si bien los controles de VSNET2005 tienen más atributos editables que la versión anterior de la misma plataforma de desarrollo, seguían sin satisfacer los requerimientos del cliente, por lo que propuso la utilización del paquete de controles *NetAdvantage* de la empresa Infragistics, que AxialSoft, S.L. ya había adquirido anteriormente.

Los controles de Infragistics están especialmente diseñados para desarrollar aplicaciones muy vistosas y personalizables, pero son poco intuitivos y la excesiva cantidad de propiedades editables que poseen los hace confusos y difíciles de usar. A eso debemos añadir el hecho de que utilizar estos controles suponía cambiar gran parte del código ya implementado hasta el momento. Por lo que finalmente se optó por implementar nuestros propios controles personalizables, hechos justo a medida de

las necesidades del cliente. Empezamos por añadir algunas propiedades a los controles *label* (etiqueta) y *textbox* (caja de texto).

Los primeros resultados obtenidos agradaron al cliente, motivo por el cual el conjunto inicial de controles se amplió hasta 6, que son los siguientes:

- ✓ *FlatLabel*.
- ✓ *FlatTextBox*.
- ✓ *FlatGroupbox*.
- ✓ *FlatCombobox*.
- ✓ *FlatPanel*.
- ✓ *FlatButton*.

La adición de atributos a los controles implica, en la mayoría de los casos, sobrecargar el método de pintado *WndProc()* de los mismos. Estos controles aportan vistosidad a los formularios del nuevo E.R.P. *Axiwin* y son muy sencillos de usar ya que, una vez generados, se trabaja con ellos en tiempo de diseño como si de los controles de VSNET se tratara.

3.3.2. Gestión de datos.

El E.R.P. *Axiwin* está concebido como una aplicación cliente/servidor. Se detalla más respecto a este modelo de ejecución en el apartado *3.3.3. Comunicación*, pero este hecho repercute también en la gestión de datos de la plataforma.

Al implantar la plataforma *Axiwin* en el escenario del cliente, éste requiere de un servidor local en el que se instalará la base de datos compartida, que contendrá los datos de las aplicaciones del E.R.P. La aplicación se instala por igual en todos los equipos de la red que vayan a ejecutarla en modo cliente; cada PC tendrá, por lo tanto, una base de datos local que almacenará únicamente la información necesaria para poder ejecutar la plataforma. Esta base de datos local será común a todos los equipos cliente.

Entre las tablas comunes a cada equipo de la red local, se encuentran las asociadas a la carga dinámica de aplicaciones (con sus respectivas tablas de pantallas, listados, campos, etc.), las de configuración de idioma, estilos y demás aspectos visuales, y un largo etcétera. Los datos de cada aplicación, en cambio, tan sólo se encuentran en la base de datos del servidor local, de modo que todos los equipos cliente deben conectarse a éste para consultarlos y/o modificarlos (lo que incluye añadir y eliminar registros). De esta forma se agiliza enormemente el proceso de carga de aplicaciones en los equipos cliente, a la vez que se evita la replicación local de datos.

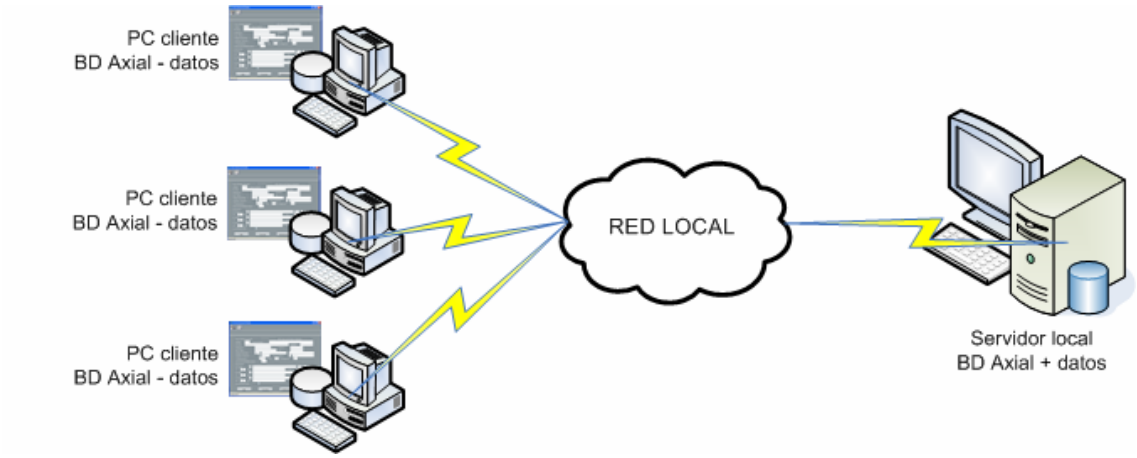


Figura 3-5. *Ejemplo representativo de la distribución de los equipos y los datos en el tipo de escenario concebido para la implantación de la plataforma Axiwin.*

3.3.3. Comunicación.

Como ya se ha comentado, la plataforma *Axiwin* está diseñada para funcionar según el clásico modelo cliente/servidor. Al adquirir el E.R.P. *Axiwin,* los clientes de AxialSoft, S.L. contratan un número predeterminado de puestos de trabajo, que se traduce en PC's que van a tener instalado el E.R.P. y lo ejecutarán en modo cliente. Habrá otro PC (únicamente uno por cliente) que ejercerá de servidor central del cliente, o servidor local, y en el que estará instalada la base de datos compartida (ver punto *3.3.2. Gestión de datos*).

El servidor local contiene en su base de datos los nombres de los usuarios dados de alta en el sistema y, por lo tanto, cada vez que un usuario desea acceder a la aplicación desde uno de los PC's clientes, la autenticación se realiza contra el servidor

local. Por su parte, el servidor local debe autenticarse contra el servidor central de AxialSoft, S.L. para verificar los derechos de ejecución de cada módulo de aplicaciones del E.R.P. *Axiwin*. En esta autenticación interviene una clave de activación proporcionada por AxialSoft, S.L. al cliente, conforme éste ha pagado los servicios contratados.

La comunicación con el servidor central de AxialSoft, S.L. toma mucha relevancia en la nueva plataforma *Axiwin*, no sólo en cuanto a la validación de las claves de activación de los clientes, sino también, y muy especialmente, en la actualización remota de la plataforma. AxialSoft, S.L. sólo debe preocuparse de publicar los paquetes de actualización en su servidor central, y a partir de este punto el proceso de actualización de los servidores locales de los clientes se describe mediante el siguiente diagrama:

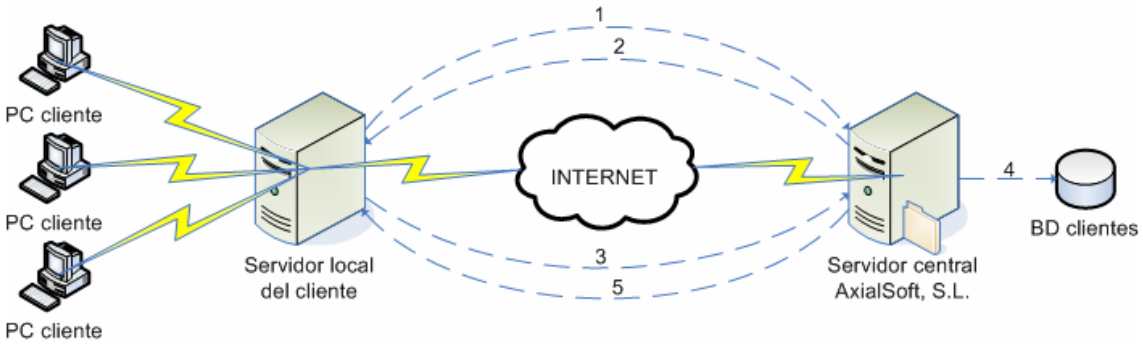


***Figura 3-6**. Diagrama del proceso de actualización de la plataforma Axiwin.*

1) El servidor local realiza la petición de actualización al servidor central de AxialSoft, S.L.
2) El servidor central responde según su disponibilidad de paquetes de actualización.
3) Si la respuesta del servidor central en el paso 2) ha sido afirmativa, el servidor local pide la actualización de un módulo.
4) El servidor central comprueba los módulos que el cliente tiene contratados y que, por lo tanto, le puede ceder.
5) El servidor central envía los paquetes de actualización al servidor local del cliente.

6) El servidor local instala las actualizaciones.

Este mismo proceso de actualización de la plataforma lo realizan los equipos del cliente contra su servidor local.

3.3.4. Normativas.

Desarrollar el nuevo paquete E.R.P. *Axiwin* ha supuesto no tan sólo intentar modernizar y mejorar en todo lo posible a su versión anterior, sino también adaptarlo a las nuevas tendencias en aplicaciones de software y, por supuesto, a las nuevas normativas respecto a las tecnologías de la información (TI's).

La Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal (*LOPD*) recoge en su primer artículo: "*tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar*". Actualmente todas las empresas tienen la obligación de cumplir con esta normativa, aplicando las pautas que la misma dicta sobre la recogida, tratamiento y conservación de datos personales. Las sanciones que se aplican por el incumplimiento de alguna de las obligaciones de la *LOPD* pueden ascender a 600.000€ (100 millones de pesetas cuando la ley fue redactada y aprobada).

En la fase de análisis funcional de la plataforma *Axiwin* se tuvo presente la necesidad de aplicar las normas establecidas por la LOPD mediante la inclusión de una columna más en las tablas de campos de aplicaciones de la base de datos. Ésta columna se utiliza para indicar que el campo en cuestión contiene datos sujetos a la normativa establecida por la LOPD y, por lo tanto, debe recibir un trato especial.

3.3.5. Seguridad.

Al desarrollar la nueva plataforma *Axiwin* se ha hecho especial hincapié en la implementación de un sistema de autenticación robusto e inviolable. Veremos las principales diferencias entre el acceso al E.R.P. original y el nuevo.

- ***Acceso a la plataforma Axiwin original.***

Al adquirir el E.R.P. *Axiwin* original el cliente obtenía de AxialSoft, S.L., junto con el software, una *mochila* que le autorizaba a utilizar la plataforma una vez instalada. La *mochila* es un pequeño dispositivo hardware que se conecta al puerto paralelo del servidor local, contra el que deben autenticarse los equipos clientes. Por este motivo para ejecutar la aplicación como cliente se requiere añadir la IP del servidor local a los parámetros iniciales de ejecución de la aplicación.

Al recibir una petición de autenticación, el servidor local comprueba que no se haya sobrepasado el número de conexiones simultáneas contratado (registrado en la *mochila*), en cuyo caso deniega el acceso a la aplicación. Por otra parte, en caso de no tener conexión, el equipo cliente no puede ejecutar la aplicación al no poder acceder a la *mochila* para autenticarse. Pese a ser un método de autenticación bastante anticuado, el uso de la *mochila* es muy seguro ya que, de ser adquirido el E.R.P. de forma fraudulenta, sería imposible ejecutarlo. Al contratar la plataforma *Axiwin* en modo DEMO, AxialSoft, S.L. proporciona al cliente un *driver* que emula la *mochila*.

En cuanto a la autenticación de usuarios en la plataforma original, el acceso se realiza utilizando un *login* y una contraseña, y el listado de usuarios autorizados en el sistema se guarda en el servidor local.

- ***Acceso a la plataforma Axiwin actualmente.***

Como se ha comentado en el apartado *3.3.4. Comunicación*, en el proceso de autenticación de un usuario en el sistema intervienen el servidor local del cliente, el servidor central de la empresa AxialSoft, S.L., y la clave de activación que ésta entrega al cliente como prueba de que ha pagado por la adquisición del E.R.P. *Axiwin*. Este proyecto se centra en el control de acceso posterior, una vez se ha validado al cliente contra el servidor de AxialSoft, S.L.

El acceso a la nueva plataforma *Axiwin* utiliza también el clásico método de autenticación de usuarios mediante *login* y contraseña. Sólo se permite el acceso a usuarios definidos en la base de datos, y el encargado de definirlos es el administrador

del cliente (quien se encarga del servidor). El control de acceso está formado por muchas capas que verifican los permisos del usuario a distintos niveles, cada cual es más restrictivo que el anterior. El orden en que se determinan los permisos del usuario lo ilustra la figura 3-7.

Una vez se ha comprobado la existencia del usuario en el sistema, se procede a cargar los menús modulares que éste tiene asignados. Los menús modulares se asocian, como su nombre indica, a los módulos disponibles en la plataforma (aquellos adquiridos por el cliente). Cada entrada de menú representa a un directorio o a una aplicación del módulo en cuestión al que se refiere el menú, de modo que éste no solo define el orden de aparición de las aplicaciones en el árbol explorador del formulario principal, sino también qué aplicaciones serán visibles y cuáles no (si no se encuentran en el menú, no son accesibles). El asociar un menú modular a un usuario, automáticamente le otorga permisos de acceso sobre el módulo referenciado, por este motivo el usuario sólo podrá ver, en el árbol explorador, los módulos cuyos menús le hayan sido asignados.

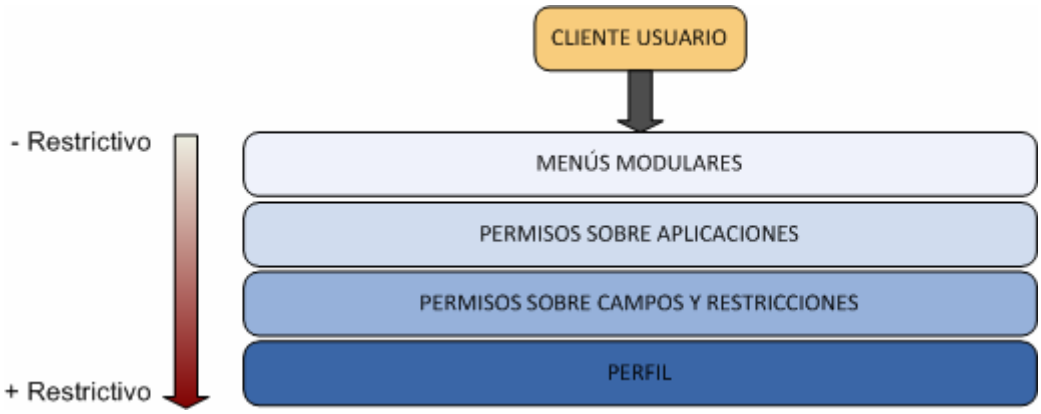


***Figura 3-7**. Orden de aplicación de permisos de acceso a la plataforma Axiwin por usuario.*

Tras este primer filtro se aplican los permisos sobre aplicaciones específicas del módulo. Esto tiene el mismo efecto que omitir la aplicación en la definición del menú, pero de esta forma se ofrece la posibilidad de aplicar un mismo menú base a muchos usuarios a la vez, para después personalizar un poco más los permisos sobre aplicaciones. Esta información por usuario se recoge en la tabla *tblAcceso*.

Una vez está acotado el conjunto de aplicaciones accesible por el usuario, se aplican los permisos específicos sobre los campos de las aplicaciones. De este modo, aunque el usuario pueda acceder a una aplicación, será posible restringirle el acceso a algunos de los campos de la misma, lo que significa que no podrá ni consultarlos ni editarlos, aunque vea sus correspondientes cajas de texto. También es posible permitir a un usuario el acceso a un campo pero restringirle el rango de datos accesible del mismo. Por ejemplo, en aplicación de consulta de fichas de clientes, se podría restringir el tipo de campo alfanumérico referido a los apellidos, permitiendo consultar o editar únicamente el rango de valores entre 'García' y 'Giménez', menor o mayor que 'Rodríguez', igual o diferente a 'Pérez', etc. Los datos de las aplicaciones de la plataforma *Axiwin* pueden ser de tres tipos distintos:

- ✓ Campo alfanumérico.
- ✓ Campo numérico.
- ✓ Campo de fecha.

Las restricciones por campos se recogen en la tabla *tblRestriccion*.

Finalmente, se aplica sobre el conjunto de aplicaciones y campos resultantes de los anteriores procesos de discriminación por permisos el filtro de acceso más restrictivo, que es el perfil o política de acceso del usuario. Los perfiles se definen mediante la combinación de 5 reglas básicas: permiso de lectura, permiso de escritura, permiso de utilización de las herramientas de la plataforma de cliente, permiso de creación y obtención de listados y máximo de altas permitidas. Un conjunto de perfiles genérico sería el siguiente:

Perfil	Lectura	Escritura	Listados	Herramientas	MaxAltas
Consulta	✓	✗	✗	✗	0
Gestión	✓	✓	✗	✓	∞
Root	✓	✓	✓	✓	∞

Los perfiles están predefinidos en la plataforma cliente por el administrador de AxialSoft, S.L., y se recogen en la tabla *tblPerfilAcceso*.

3.3.6. Compatibilidad con plataforma original.

La empresa AxialSoft, S.L. cuenta actualmente con una extensísima cartera de aplicaciones, desarrolladas en su propio lenguaje para la plataforma *Axiwin*. En ningún caso era aceptable que el desarrollo del nuevo E.R.P. implicase tener que volver a implementar dichas aplicaciones, motivo por el cual el requisito más indispensable de este proyecto ha sido la total compatibilidad entre la nueva plataforma y su antecesora.

A fin de hacer posible esta compatibilidad entre plataformas, ha sido necesario implementar un conjunto de herramientas para migrar los datos del E.R.P. original, contenidos y distribuidos en un complejo sistema de ficheros, a la base de datos que el nuevo E.R.P. utiliza en sustitución de dichos ficheros. La utilización de una base de datos es fundamental para dotar a la nueva plataforma del dinamismo y velocidad de acceso a datos que el cliente solicitaba como otro de los requisitos primordiales del proyecto Axialsoft.

La aplicación de migración se compone de distintos proyectos dedicados, cada uno de ellos, a la traducción de una parte específica del sistema de ficheros original. La principal dificultad que ha entrañado la migración de datos ha sido descubrir la estructura interna de los distintos tipos de ficheros, en su mayoría binarios (ver apartado *3.1. Objetivos y alcance del proyecto*).

Tras el proceso de migración es preciso realizar una adaptación de los datos migrados, que consiste en crear nuevas tablas en la base de datos, añadir columnas a las ya existentes, y redistribuir o dar un nuevo formato a la información contenida en las mismas. Este paso es necesario para adecuar los datos originales al diseño orgánico de la nueva plataforma.

Capítulo 4.
Análisis funcional.

Capítulo 4. Análisis funcional.

4.1. Descripción del entorno.

El nuevo E.R.P. *Axiwin* se puede ejecutar en cualquier versión del sistema operativo Windows posterior a Windows 95, ya que es necesario tener instalado el *framework* (máquina virtual) de Visual Studio .NET y Windows 95 no lo soporta.

Tanto la plataforma de administrador como la plataforma de cliente están diseñadas para trabajar a pantalla completa, preferentemente a una resolución de 1024 X 768 píxels. La combinación de colores de ambas plataformas la define la *hoja de estilos* activa en el momento de la ejecución. Del mismo modo, el *set* de iconos cambiará según el estilo aplicado, ya que ambos están vinculados.

Es posible escoger el idioma de la plataforma de cliente en tiempo de ejecución, pero el idioma que se carga al inicio está definido en una tabla de la base de datos que contiene éste y otros parámetros iniciales de la plataforma de cliente.

El E.R.P. *Axiwin* consta de un servidor y N clientes, donde N es el número de puestos de trabajo contratados por el cliente final a AxialSoft, S.L. La plataforma de cliente se ejecuta de forma local pero los usuarios deben autenticarse contra la base de datos del servidor para acceder a la plataforma. Cada plataforma de cliente tiene instalada su propia base de datos, en la que tendrá replicadas todas las tablas de la base de datos del servidor menos las de datos, propiamente, que se accederán de forma remota.

4.2. Mapa principal de navegación.

El entorno gráfico de la plataforma de administrador es más sencillo que el de la plataforma de cliente, ya que es la segunda la que AxialSoft, S.L. distribuye comercialmente, motivo por el cual se ha cuidado con especial atención la presentación de todos los formularios.

Tanto la plataforma de administrador como la plataforma de cliente requieren autenticación al usuario, por lo que el formulario de *login* será el primero que se mostrará al usuario en ambas plataformas.

Al acceder a la plataforma de administrador el usuario puede navegar por las opciones disponibles a través del árbol explorador del formulario principal, o utilizando el doble clic sobre los iconos que aparecen en la parte central del formulario. Este comportamiento emula el funcionamiento del explorador de Windows, ya que es muy sencillo e intuitivo de usar, y los usuarios ya estarán habituados a él. Las herramientas del administrador están separadas por carpetas según su funcionalidad; tenemos por un lado las aplicaciones de gestión gráfica (editor de estilos, editor de iconos, etc.), el conjunto de aplicaciones del generador (editor de código, editor de pantallas, etc.), etc. Es también posible acceder, desde el nivel principal, a la plataforma de cliente (ver Figura 4.1).

La plataforma de cliente ofrece al usuario muchas más opciones a simple vista. El formulario principal consta también de un árbol explorador y una parte central, además de una botonera y un menú muy completo (ver Figura 4.2). El número de módulos (y sus correspondientes aplicaciones) que el usuario pueda ver al acceder a la plataforma estará determinado por los menús modulares que éste tenga asignados. Por este motivo los módulos se cargan dinámicamente en un proceso aparte mientras que el resto de opciones de menú se cargan por código con opciones predefinidas.

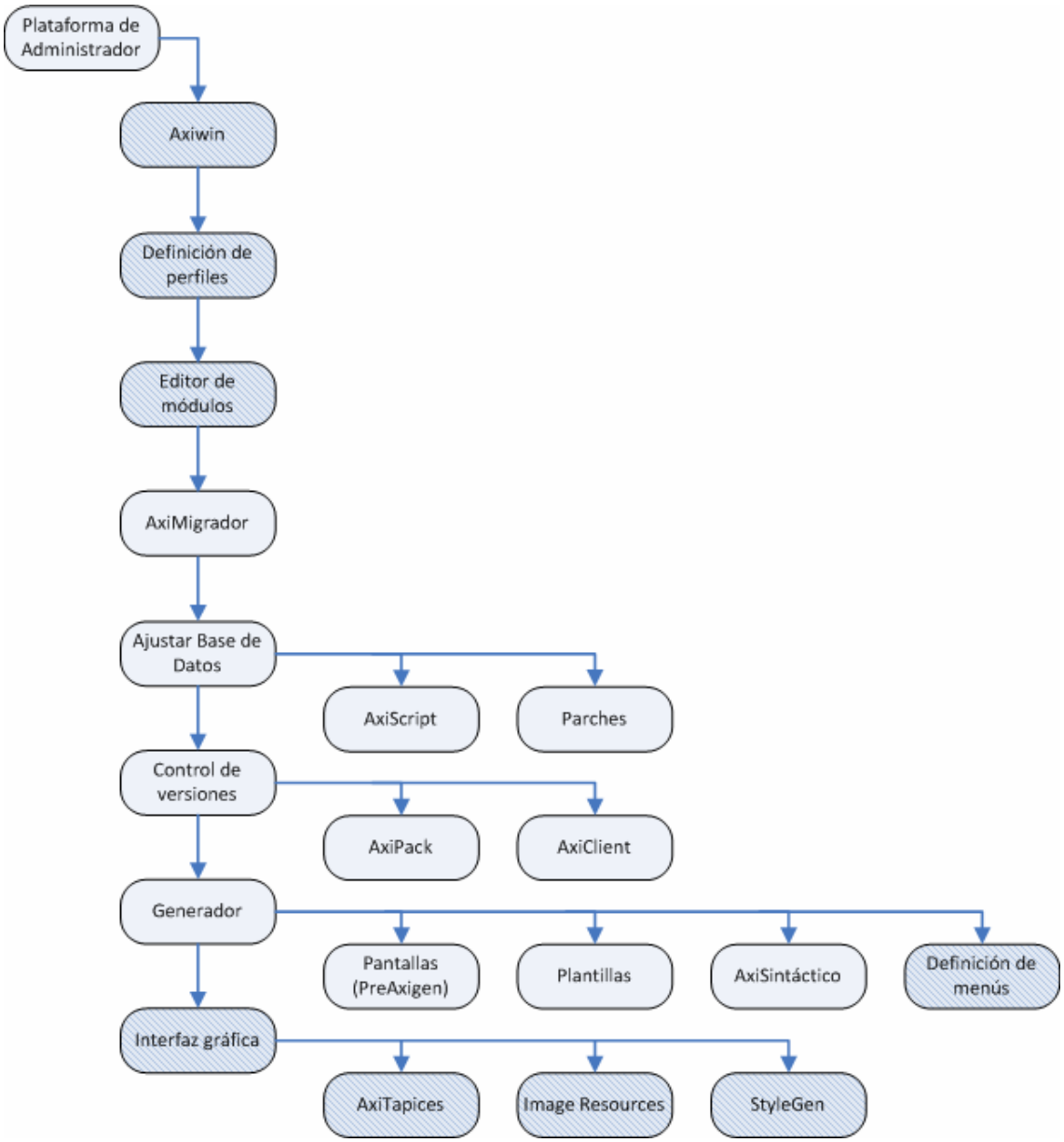


***Figura 4-1**: Mapa de navegación de la plataforma de administrador. Los módulos resaltados se han implementado como parte de este proyecto.*

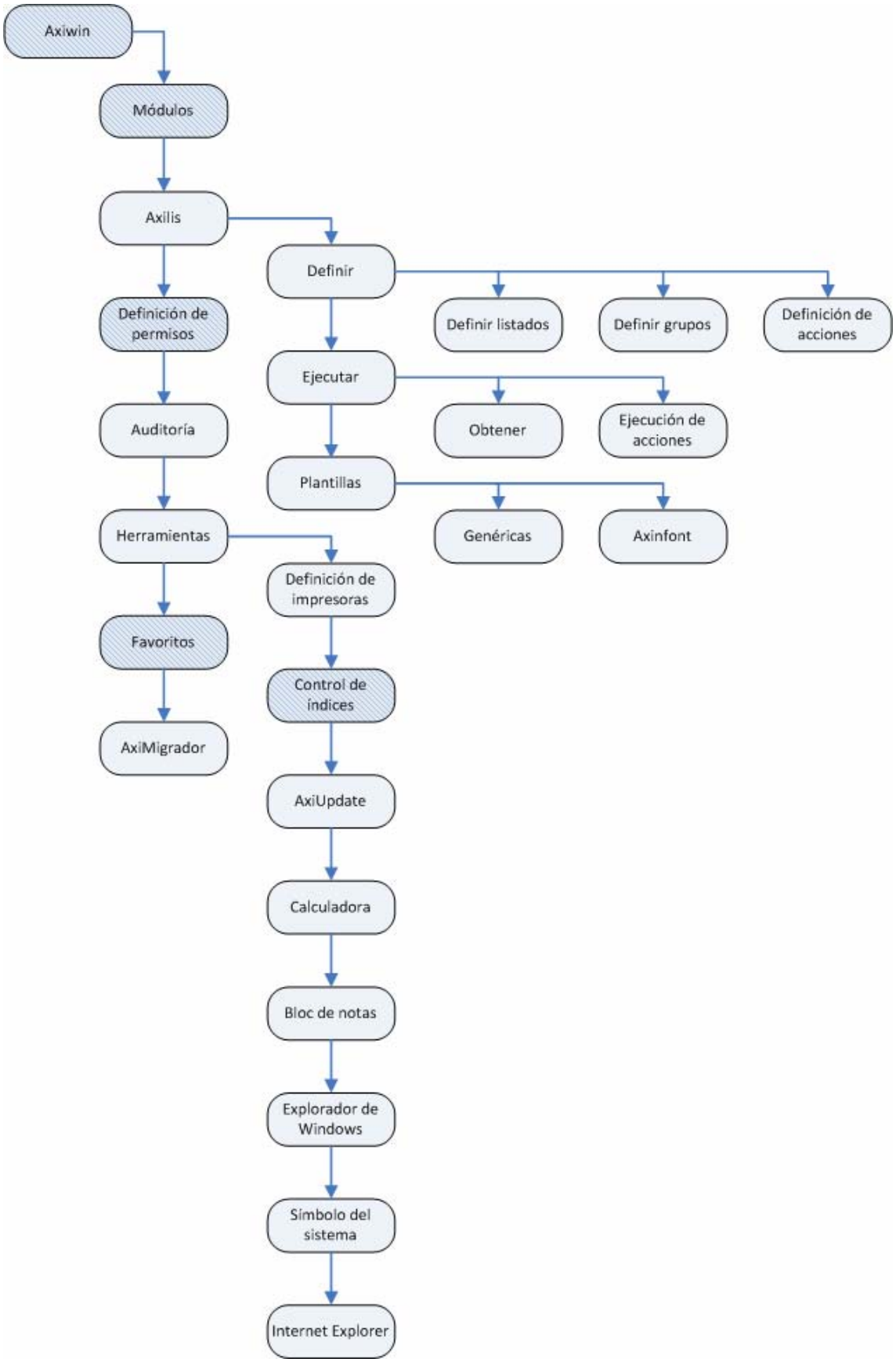


***Figura 4-2**: Mapa de navegación de la plataforma de cliente. Los módulos resaltados se han implementado como parte de este proyecto.*

4.3. Módulos funcionales de la aplicación.

4.3.1. Traductor de pantallas y Cargador de pantallas.

Ambas funcionalidades pertenecen al proyecto *TraductorPANs*, que es el encargado de realizar la migración de pantallas y la carga de formularios desde la base de datos.

4.3.1.1. Traductor de pantallas.

El proyecto *TraductorPANs* se implementó como una aplicación independiente que traducía los ficheros de pantallas .pan de la plataforma *Axiwin* original e introducía la información extraída en las tablas pertinentes de la base de datos. Por este motivo el proceso de traducción de pantallas consta de un solo formulario, poco vistoso, mientras que el código que ejecuta la traducción es muy extenso y complicado.

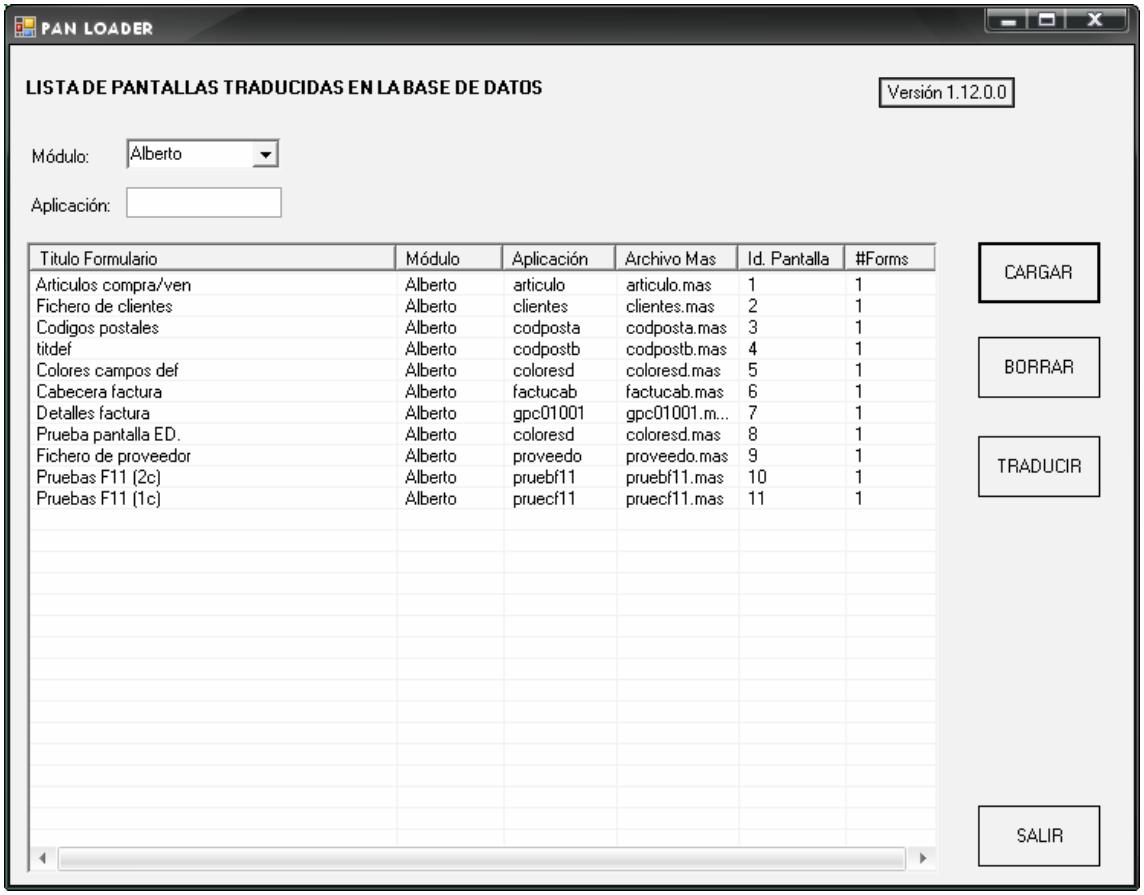


***Figura 4-3**. Formulario principal del proyecto traductor de pantallas, antes de implementar la carga de estilos. Vemos listadas todas las pantallas traducidas hasta el momento.*

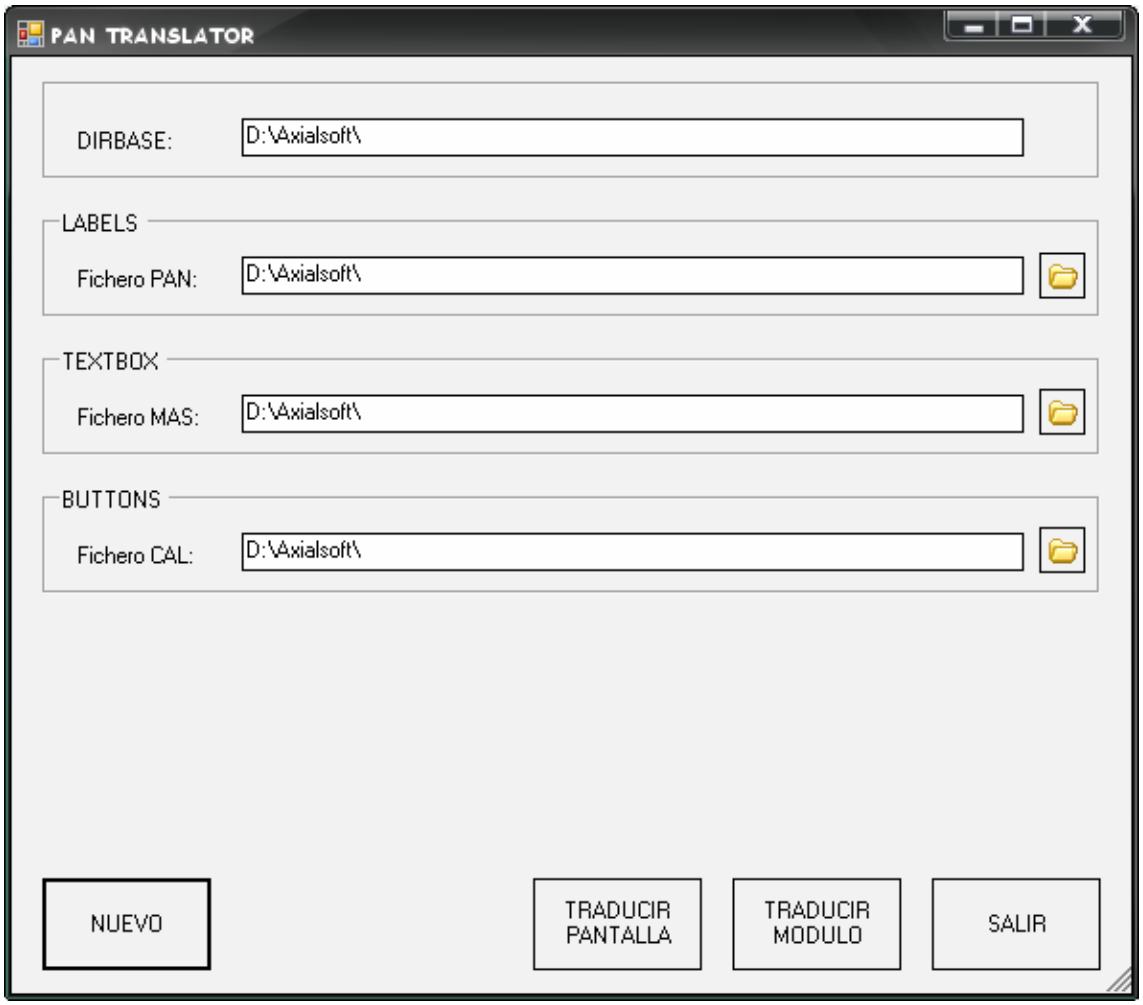


***Figura 4-4**. Formulario de traducción de pantallas antes de implementar la carga de estilos. Permite escoger entre traducir un fichero de pantalla o todo un módulo.*

Los ficheros .pan contienen la información que define las pantallas, o formularios, de las aplicaciones implementadas en el lenguaje propio de AxialSoft, S.L., y esta información está encriptada. El generador de pantallas del E.R.P. *Axiwin* original no es más que un editor de texto en el que los marcos delimitadores y los detalles de los formularios, tales como recuadros, se definen mediante caracteres de la tabla ASCII extendida. La forma en que se escriben estos caracteres en los ficheros de pantallas consiste en indicar primero con un número negativo la cantidad de veces que se repite un mismo carácter seguido del valor ASCII del mismo, que lo define. Las cadenas de caracteres que definen las etiquetas de los formularios se escriben "tal cual", indicando antes de cada cadena de caracteres la longitud de la misma mediante un

número positivo. Así, el proceso de lectura se intercala entre leer números (*shorts* en vez de enteros) y leer *bytes*.

De la lectura de los ficheros .pan extraemos únicamente las etiquetas de los formularios y su ubicación exacta en la pantalla, las dimensiones de la cual son siempre 80 caracteres de ancho por 25 caracteres de alto. Dado que la lectura se realiza en *bytes* y en *shorts* (2 *bytes*), utilizamos un contador para saber el número total de *bytes* leídos hasta el momento durante el proceso de traducción de cada fichero .pan. Si consideramos que la pantalla consta de 2000 caracteres (80 x 25), obtenemos la ubicación de las etiquetas en la pantalla del siguiente cálculo:

- Posición en X: (#*Bytes* leídos) *mod* (Ancho de pantalla)
- Posición en Y: (#*Bytes* leídos) / (Ancho de pantalla)

La posición en X y la posición en Y de cada etiqueta tiene en cuenta, además, un *offset* variable propio de cada pantalla.

***Figura 4-5**. Aspecto de la pantalla clientes.pan desde el editor de pantallas de la plataforma Axiwin original (Axigen).*

Durante el proceso de traducción de una pantalla se genera una reproducción de la misma utilizando caracteres normales y se escribe en un fichero de salida con formato .txt. Este fichero de salida no tiene ninguna utilidad aparente, pero cuando el

resultado de la traducción de una pantalla es erróneo la mayoría de las veces es posible detectar el origen del error con un simple vistazo al fichero de texto generado.

```
                              DATOS_DE_CLIENTES
+------------------------------------------------------------+
|                                                            |
|  ID._Cliente:                                              |
|                                                            |
|  Nombre:                                                   |
|                                                            |
|  Dirección:                                                |
|                                                            |
|  Teléfono:                                                 |
|                                                            |
|  CP:       Población:                                      |
|                                                            |
|            Provincia:                                      |
|                                                            |
|  Descuento:    %                                           |
|                                                            |
+------------------------------------------------------------+
```

***Figura 4-6**. Ejemplo de fichero de texto generado en el proceso de traducción de la pantalla de la figura 4-5.*

***Figura 4-7.** Aspecto de la pantalla agifigen.pan desde el editor de pantallas de la plataforma Axiwin original (Axigen).*

La migración de pantallas no sólo consiste en traducir etiquetas sino también los *checkbox* y las cajas de texto (*textbox*), que representan a los campos del/los fichero/s de datos asociado/s a un formulario. La información sobre qué campos aparecen en pantalla, sus ubicaciones, la longitud de las cajas de texto que los

representan y el máximo de caracteres que éstos pueden contener, entre otros muchos datos, se encuentra en los ficheros .mas.

```
____Ficha_Clientes_General_____________<F8>_Menu_de_Ficheros_Adjuntos_____
__|_Codigo_:_____Razon_Social_:_____________________________________|___
__|_Nombre_:_____________________________________CIF/NIF:_____________|___
__|_Anagrama_.......:_____________________Telefono_:_____________|___
__|_Otros_Tlfnos_...:_____________________Fax_.....:_____________|___
__|_Codigo_Hacienda_:_____________________________________________|___
__|_Contrato_o_Servicio_:_____________Tipo_Empresa_:_____________|___
__|_%_Participacion_....:_____Area:_____Delegacion:_____________|___
__|_Cod.Comunidad/Sociedad:_____________________________________|___
____Direcciones_____________________________________________________
__|_Dir.Fiscal_:_____________________________________________|___
__|_C.P:_____Pob:_____________________________Prov_.:_____|___
__|_Dir.Activi.:_____________________________________________|___
__|_C.P:_____Pob:_____________________________Prov_.:_____|___
__|_Dir.Envio_.:_____________________________________________|___
__|_C.P:_____Pob:_____________________________Prov_.:_____|___
__|_Envio_Etiqueta_S/N_:_____________________________________|___
_____________________________________________________________________
```

***Figura 4-8**. Ejemplo de fichero de texto generado en el proceso de traducción de la pantalla de la figura 4-7.*

Para consultar la información contenida en los ficheros .mas se utilizó una OCX facilitada por la empresa AxialSoft, S.L., que se intentó modificar en más de una ocasión para incorporarle nuevas funciones pero no fue posible. También a partir de los ficheros .mas se migraron los *checkbox* a la base de datos. Otro de los tipos de controles que se traducen en el proceso de migración de pantallas, aunque no se extraigan sus propiedades de los ficheros .pan, son los *datagrids*.

Los *datagrids* reciben el nombre de *lineales* en el E.R.P. *Axiwin* original, y es posible mostrar éstos en un formulario de dos formas distintas: la primera es mediante una instrucción del lenguaje propio de AxialSoft, S.L. en el código asociado a una pantalla; en el segundo caso se requieren dos tipos de ficheros relacionados entre sí: un fichero de líneas y un fichero de cabeceras.

En el primer caso, la migración de pantallas juega un papel muy pequeño, detectando la llamada a la instrucción en cuestión (*ENTLINEAL*), capturando los parámetros que se le pasan e introduciéndolos en la base de datos para que sean

tratados posteriormente, pero este proyecto no contempla esta parte del funcionamiento de la nueva plataforma *Axiwin*.

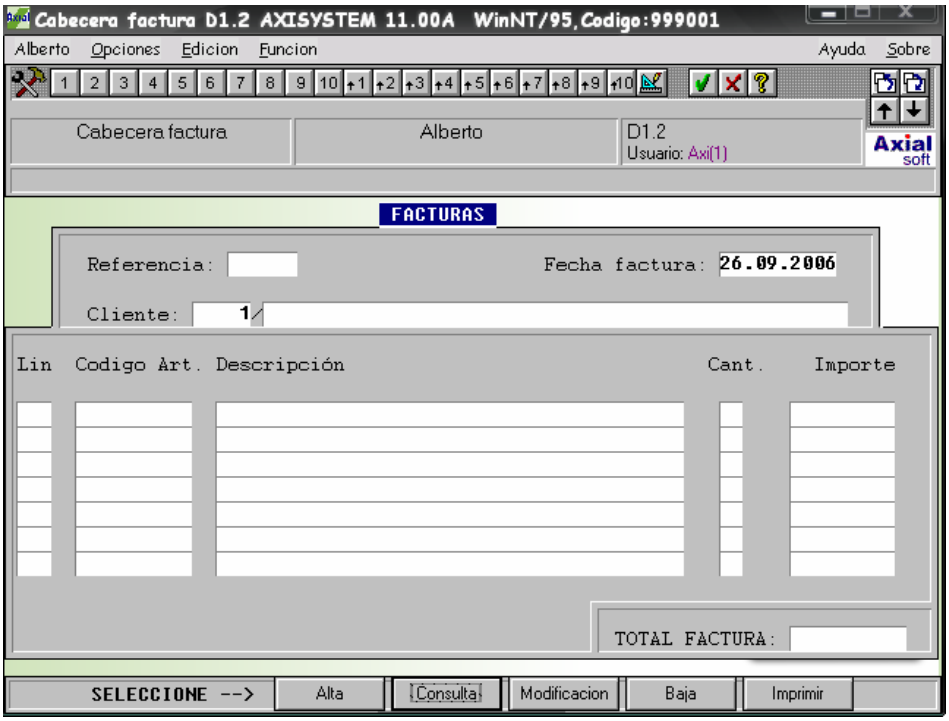


***Figura 4-9**. Pantalla de una aplicación con "lineal", en la plataforma Axiwin original.*

En el caso de los ficheros de cabeceras y los ficheros de líneas, el primero sirve para definir las columnas de un *lineal*, mientras que en el segundo se encuentra la definición de los campos que se asocian a cada columna del *lineal*, así como el número máximas de filas visibles en el formulario. La relación entre estos dos ficheros se detecta en la fase de migración de datos.

Por último, también se lleva a cabo durante la migración de pantallas la traducción de "botones de mantenimiento". Las pantallas están asociadas no sólo a uno o varios ficheros de datos, sino que además, y por lo general, se les asocia también un fichero de código fuente programado en el lenguaje propio de AxialSoft, S.L. En caso de no tener asociado un fichero de código fuente, la aplicación que define la pantalla se accederá en modo *mantenimiento*. Esto significa que será posible acceder a los registros del fichero de datos asociado para hacer altas, bajas, consultas y modificaciones. En caso de tratarse de una aplicación de *mantenimiento*, en el momento de traducir su/s pantalla/s asociada/s se tendrá en cuenta que, cuando se

cargue el formulario migrado, deberán mostrarse los 4 botones correspondientes a la opciones de *mantenimiento* mencionadas.

El código del traductor de pantallas requirió de muchas pruebas y de muchos retoques y añadidos hasta conseguir resultados satisfactorios en todas las traducciones.

4.3.1.2. Cargador de pantallas.

Mientras que el *Traductor de pantallas* introduce los datos de las pantallas migradas en la base de datos, el *cargador de pantallas* es el encargado de leer esos datos y generar formularios, de forma totalmente dinámica, que representen con exactitud a sus respectivas pantallas originales.

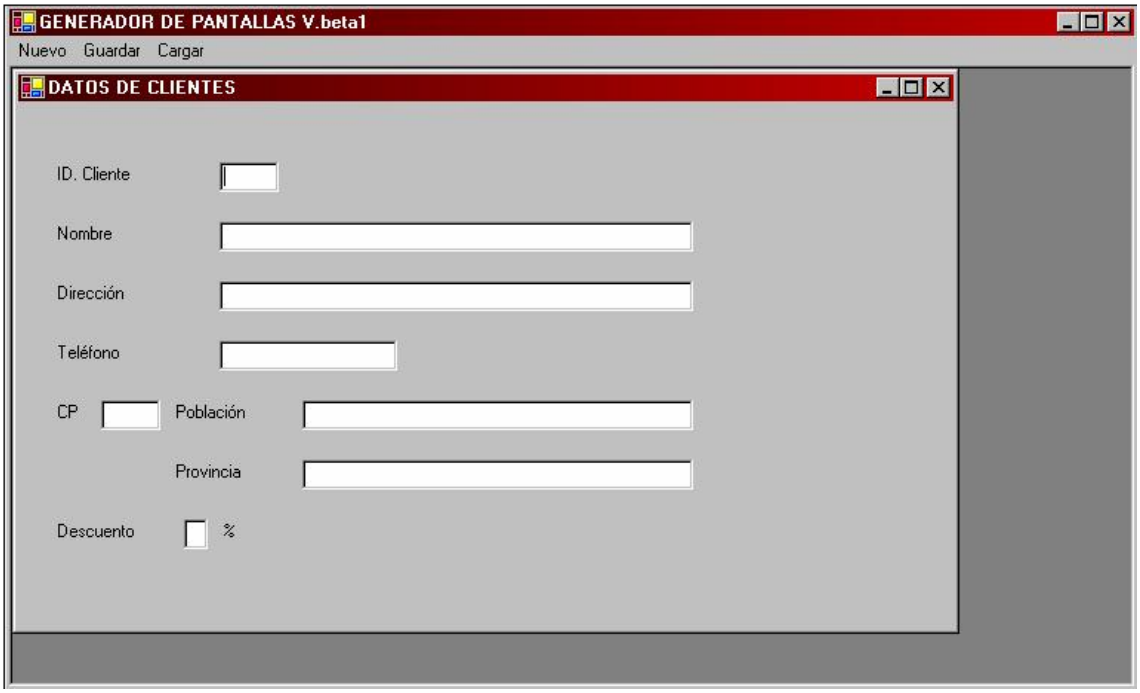


***Figura 4-10**. Esta captura muestra el aspecto del formulario 'Clientes' cargado dinámicamente de la base de datos con la primera versión del Cargador de pantallas.*

El proceso de carga de un formulario consta de muchos pasos y sigue un orden concreto. Se utiliza un formulario genérico sobre el que cargar los controles, el *frmContenedor*, que tiene ya implementados muchos eventos comunes a todas las aplicaciones del E.R.P. *Axiwin*. Los controles no se cargan directamente sobre el formulario contenedor sino sobre un control de tipo panel y, dado que una aplicación

puede tener más de una pantalla, cada una de éstas se carga sobre un panel distinto. Los paneles se añaden al conjunto de controles del formulario contenedor a medida que se van cargando.

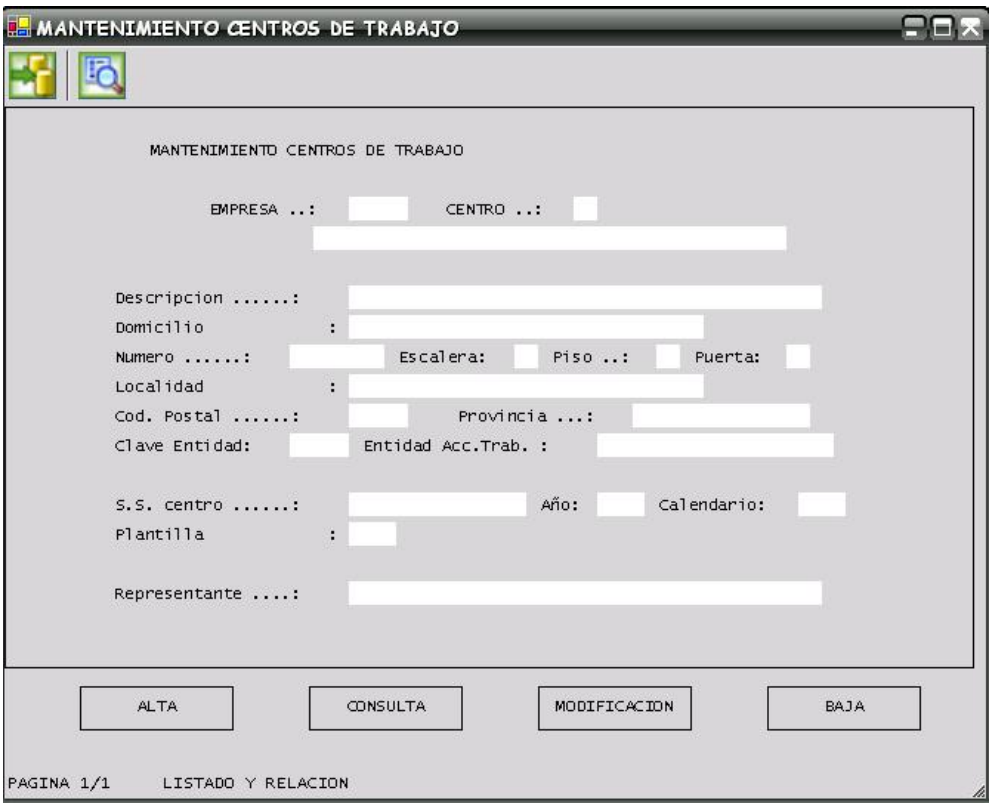


***Figura 4-11**. La carga de botones se añadió en versiones posteriores del Cargador de pantallas. Podemos apreciarlos en este formulario, pero el aspecto visual es muy pobre.*

El proceso de carga de pantallas sigue este orden:

a) Primero de todo se comprueba si la aplicación seleccionada corresponde a un fichero de cabeceras, lo que significa que habrá un *datagrid* en la pantalla. En caso afirmativo es necesario encontrar el correspondiente fichero de líneas para saber cuáles son los campos que componen las columnas del *datagrid*.

b) Si la aplicación asociada tiene más de una pantalla, para cada una se aplican los siguientes pasos.

c) Se cargan las propiedades de la pantalla en un panel.

d) Se cargan los *checkbox* en el panel.

e) Se cargan las cajas de texto en el panel y se asocian a sus campos correspondientes.

f) Se cargan las etiquetas en el panel.

g) Finalmente, si la pantalla tiene un *datagrid*, éste se carga en el panel.

Es necesario cargar las etiquetas después de cargar las cajas de texto porque sino las segundas se superponen a las primeras, en ocasiones cubriéndolas por completo, y el resultado sería insatisfactorio.

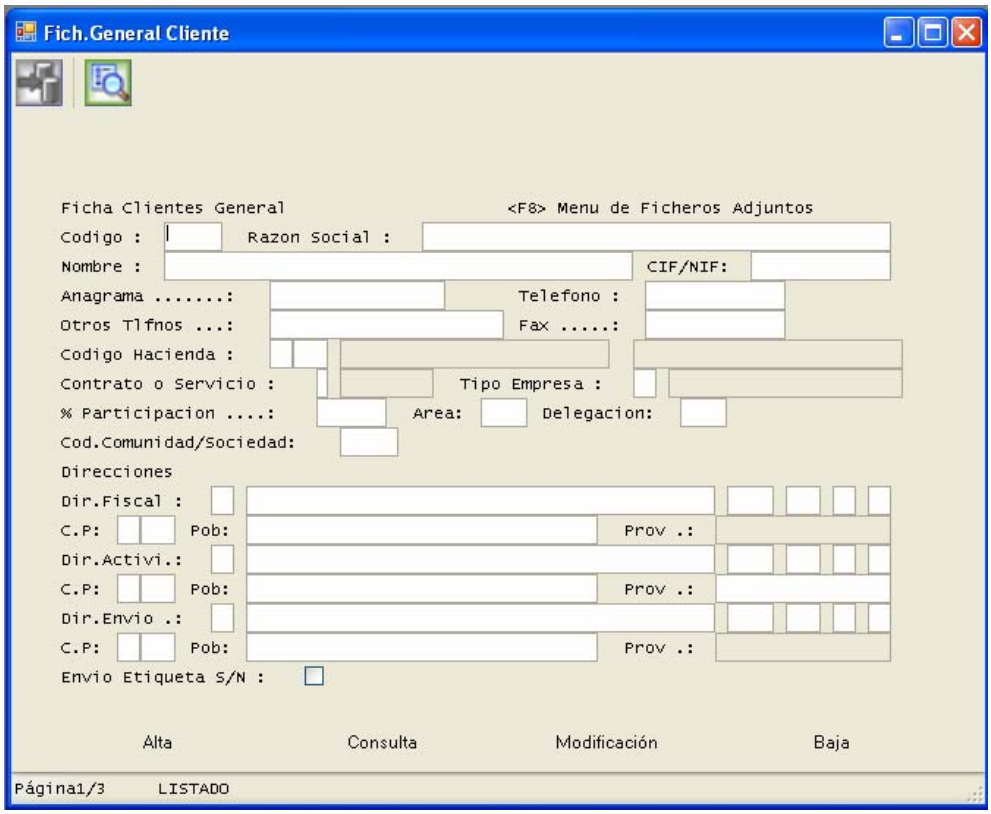


***Figura 4-12**. Nuevo aspecto de la aplicación 'Agifigen', vista en la figura 4-7. En esta versión del Cargador de pantallas todavía no se aplicaban estilos a los controles.*

Tras las primeras pruebas de carga de pantallas a partir de la información contenida en la base de datos se pudo comprobar que los formularios aparecían desplazados respecto a los márgenes establecidos, lo cual nos hizo preguntarnos si acaso las posiciones en pantalla que habíamos calculado tenían valores incorrectos y, por lo tanto, la lectura del fichero .pan estaba mal implementada. Después de echar un vistazo al diseño de pantallas desde el generador de pantallas del E.R.P. *Axiwin* original pudimos observar que nuestros cálculos de posición *x* e *y* se habían realizado respecto

al marco delimitador del formulario y no respecto a la ventana, que es más grande. Fue necesario entonces añadir los *offset* en ancho y altura que se han comentado anteriormente, y que son propios de cada fichero .pan.

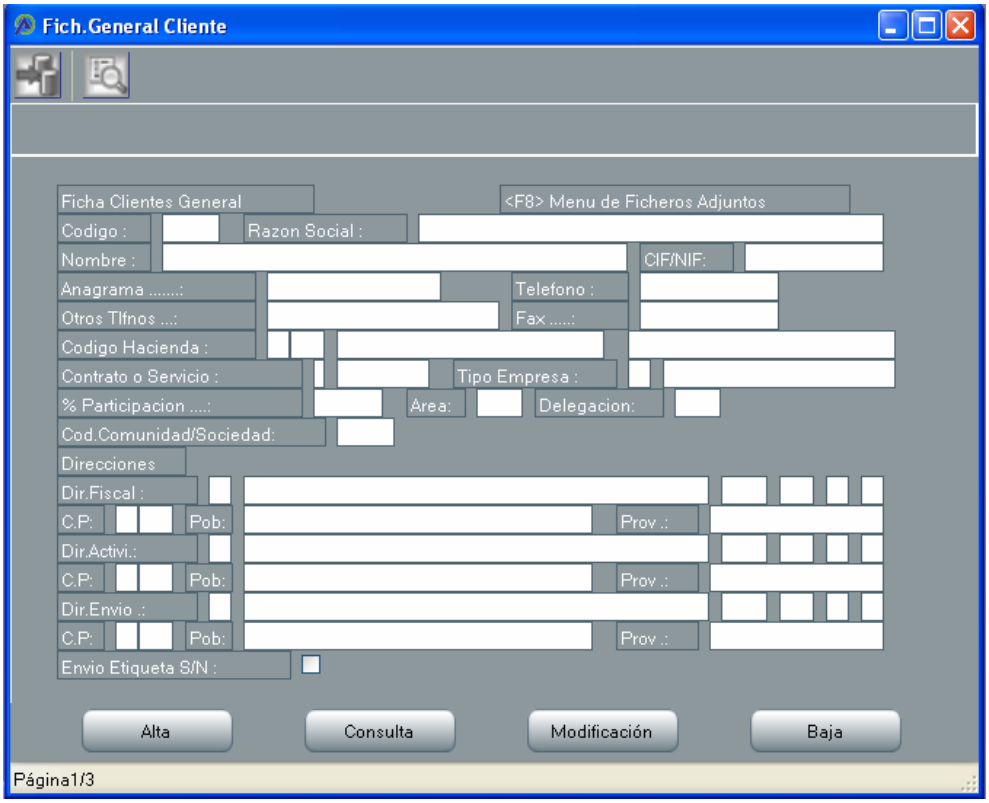


***Figura 4-13**. Otra vez la aplicación 'Agifigen', esta vez aplicando estilos a los controles.*

Pese a haber logrado resultados muy satisfactorios existe un problema con la carga de pantallas que hasta el momento no se ha podido resolver. Si tenemos en cuenta la forma en que se lee el fichero de pantalla .pan y se extraen las posiciones en *x* e *y* de las etiquetas, comprobamos que los tamaños de las etiquetas se calculan a partir del número de caracteres que éstas contienen, por lo que, por poner un ejemplo, una etiqueta con el texto "Hola" tiene longitud 4. Pero en Visual Studio .NET los tamaños de los controles se calculan en píxels, por lo que había que encontrar un multiplicador que, aplicado a la longitud en caracteres, nos diera el ancho equivalente de la etiqueta en píxels. Actualmente se utiliza el tamaño de fuente como multiplicador pero los resultados siguen sin ser del todo perfectos, ya que para ello se tendría que usar en las etiquetas el mismo tipo de fuente que se utilizaba en la plataforma *Axiwin* original, y esto no es posible.

4.3.1.3. Carga de relaciones entre aplicaciones.

Las aplicaciones del E.R.P. *Axiwin* se definían originalmente mediante varios ficheros, repartidos en distintos directorios específicos, distribuyendo la definición de los campos en los ficheros .mas, la de pantallas en los ficheros .pan, la de los datos en los ficheros .dat, etc. El sistema de ficheros está muy bien diseñado, existiendo complejas relaciones entre los ficheros de las distintas aplicaciones de forma parecida a como se definen claves foráneas en las tablas relacionadas de una base de datos. Así pues un campo concreto en una aplicación puede estar referenciado a otro campo de otra aplicación, forzando entre ambos una existencia obligatoria.

Las relaciones entre aplicaciones se migran también a la base de datos, convirtiéndolas en claves foráneas, durante el proceso de migración de los datos y su estructura (definición de los campos). Esta parte, sin embargo, no concierne a la traducción de pantallas, pero sí a la carga de aplicaciones.

En la plataforma *Axiwin* original, al ejecutar una aplicación y posicionar el foco sobre un campo (caja de texto) relacionado con otra aplicación, se indica éste vínculo al usuario mediante un botón que normalmente está *desactivado*. Entonces es posible acceder a la aplicación relacionada mediante la pulsación de éste botón o la tecla *F11*. Este funcionamiento debía mantenerse en el nuevo E.R.P., por lo que se incluyó en la implementación de la carga de aplicaciones.

Dicho funcionamiento se emula en la nueva plataforma *Axiwin* utilizando eventos. En este caso se trata de un evento asociado a una caja de texto, que es la representación en pantalla de un campo de la aplicación a cargar. Por este motivo los campos con relaciones a otras aplicaciones deben detectarse en el momento de la carga de la pantalla a la que pertenecen, ya que la asociación de eventos a controles se tiene que hacer antes de mostrar el formulario.

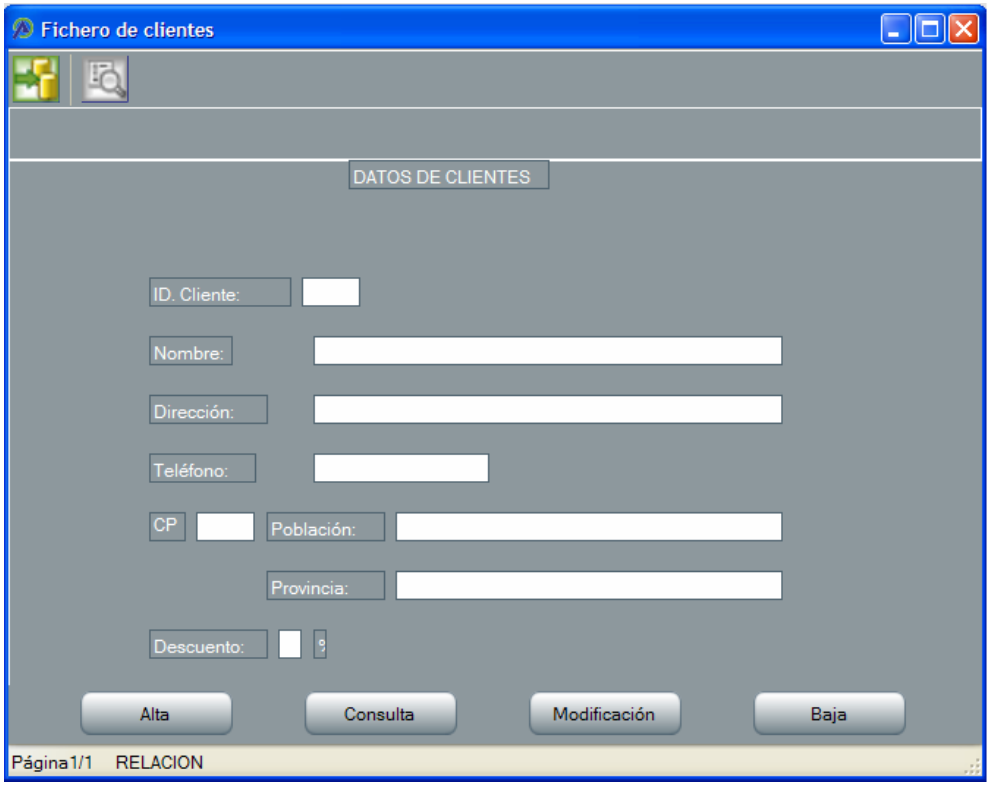


***Figura 4-14**. Podemos apreciar una mejora sustancial en el aspecto actual de la aplicación 'Clientes' respecto al de la figura 4-10. El botón en la parte superior izquierda indica que el campo sobre el que está el cursor (C.P., código postal) está relacionado con otra aplicación; podemos verlo también en la etiqueta de la barra de estado, en la que se lee 'RELACION'. Al pulsar la tecla F11 el formulario relacionado se abre automáticamente (ver figura 4-15).*

La implementación de la detección de relaciones entre campos y la posterior asociación del evento de carga de una aplicación relacionada fue una tarea muy dificultosa. Para la detección de relaciones se consultan directamente las tablas de información de la base de datos (*INFORMATION_SCHEMA*) para obtener todas las claves foráneas de la aplicación a cargar. A partir de las claves foráneas se generan los parámetros de los eventos de carga de aplicación relacionada y éstos se asocian a las cajas de texto dinámicamente en el momento de cargarlas en su correspondiente formulario.

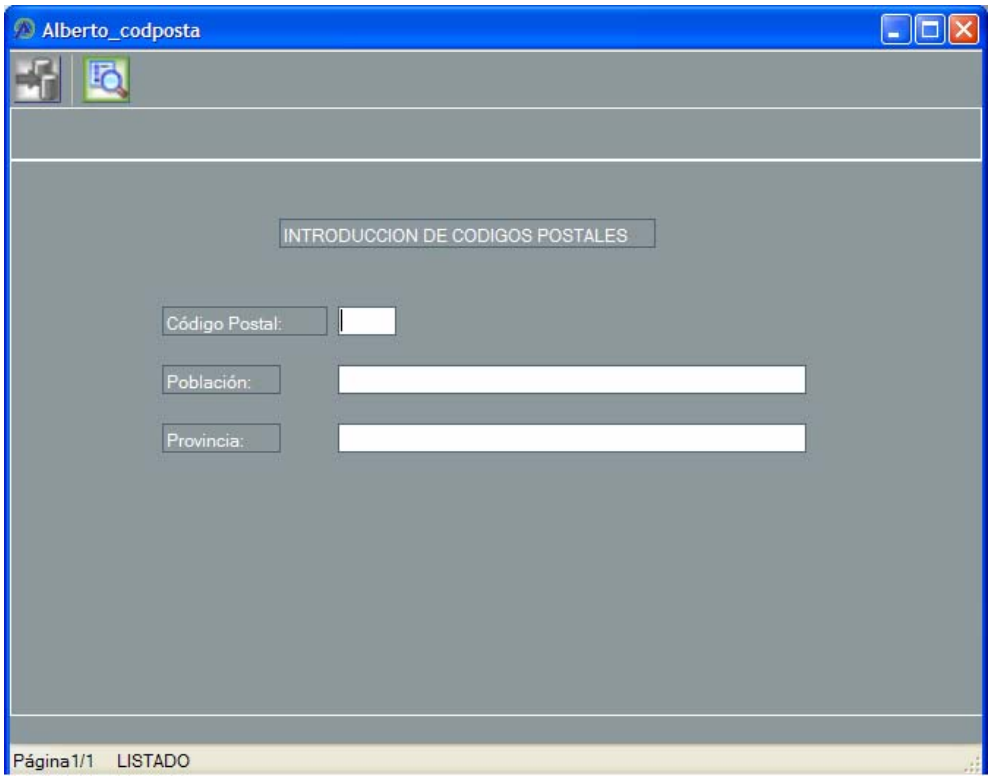


***Figura 4-15**. Al pulsar F11 sobre el campo C.P. en el formulario de la figura 4-14, se abre la aplicación relacionada de 'Códigos postales'.*

4.3.1.4. Creación de nuevos controles.

Los controles de la plataforma de desarrollo Visual Studio .NET 2005, comparados con los de su versión anterior, son mucho más vistosos y tienen muchas más propiedades editables, lo que facilita al programador la creación de formularios visualmente atractivos. Aún así las posibilidades de diseño que ofrecen los controles de VSNET2005 no satisfacían completamente nuestras necesidades ni las del cliente, quien entonces sugirió utilizar el paquete de controles *NetAdvantage*, de la empresa Infragistics.

Infragistics, como muchas otras empresas de desarrollo de software, se dedica a crear y vender controles altamente personalizables, especialmente destinados a la implementación de aplicaciones Windows con Visual Studio .NET. La empresa AxialSoft, S.L. ya había adquirido una versión del paquete *NetAdvantage*, y durante unos días se hicieron pruebas con los nuevos controles, readaptando buena parte del código ya implementado. Ciertamente, los controles de Infragistics permiten personalizar gran parte de las propiedades estéticas de los mismos, pero no son nada

intuitivos y tienen, de hecho, tantas propiedades editables que se vuelven confusos y complicados de utilizar. Se optó entonces por abandonar los controles de Infragistics e implementar los nuestros propios, partiendo de los de VSNET2005 y añadiéndoles las propiedades estrictamente necesarias.

En Internet encontramos exactamente lo que buscábamos: el *FlatTextbox,* una clase de tipo control que hereda los atributos del *Textbox* y sobrescribe la función de pintado *WndProc()*, añadiendo un marco alrededor de la caja de texto (ver *Anexo II*). Mediante la adición de dos nuevos atributos públicos al control se permite la modificación del grosor del marco (*BorderWidth*) y el color del mismo (*BorderColor*). Empleando la misma metodología creamos otros tres controles: *FlatLabel*, *FlatCombobox* y *FlatPanel*.

Posteriormente se crearon los controles *FlatGroupbox* y *FlatButton*. A pesar de su nombre el *FlatGroupbox* no hereda las propiedades del *Groupbox*, sino que se compone de un panel de tipo *FlatPanel* y una etiqueta para crear un control similar a un *Groupbox*. Aunque el resultado visual es muy bueno, el funcionamiento del control *FlatGroupbox* es algo imperfecto e impredecible en tiempo de diseño. Por su parte el control *FlatButton* no es más que un botón completamente plano al que se le aplica una imagen de fondo para crear su forma y darle la sensación de relieve. Se utilizan otras 3 imágenes para crear los efectos de *disabled* (botón inhabilitado), *mouseOver* (botón seleccionado) y *mouseDown* (botón presionado).

A continuación podemos ver un resumen de los controles implementados:

Nombre del control	Control del que hereda	Propiedades añadidas respecto al control original
FlatTextbox	Textbox	✓ BorderColor ✓ BorderWidth ✓ BackColorInactive
FlatLabel	Label	✓ BorderColor ✓ BorderWidth ✓ ImageInactive
FlatPanel	Panel	✓ BorderColor

		✓ BorderWidth
FlatCombobox	Combobox	✓ BorderColor ✓ BorderWidth
FlatGroupbox	FlatPanel + Label	✓ BorderColor ✓ BorderWidth
FlatButton	Button	✓ BackgroundImageActive ✓ BackgroundImageInactive ✓ BackgroundImageMouseDown ✓ BackgroundImageMouseOver ✓ ImageActive ✓ ImageInactive

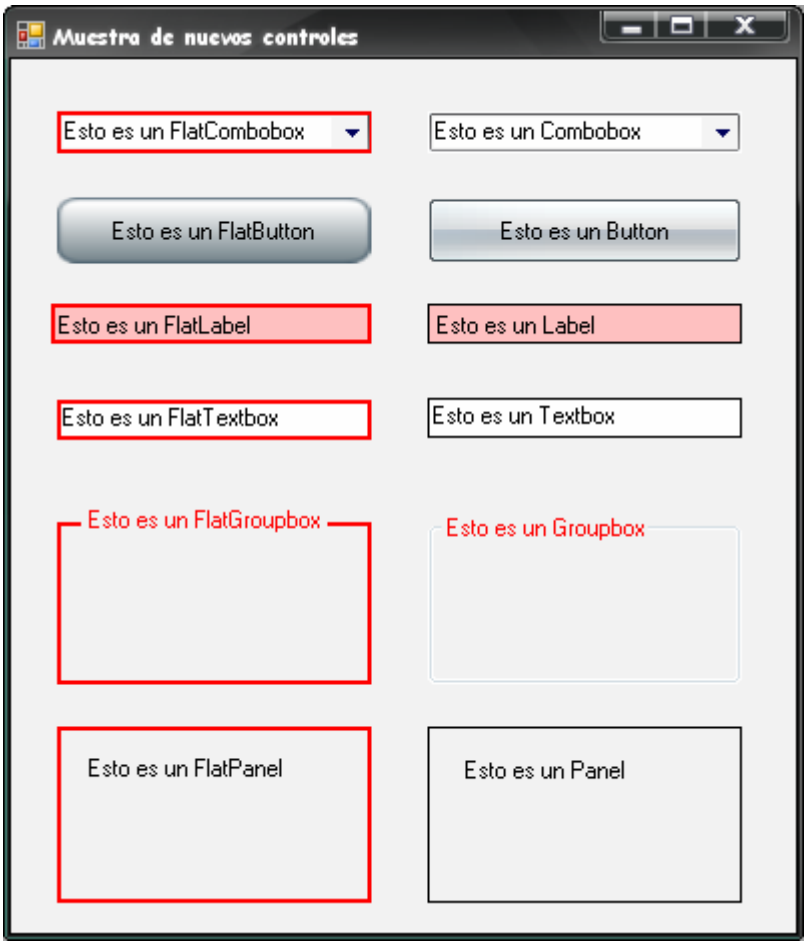


***Figura 4-16**. Formulario de ejemplo en el que podemos ver las principales diferencias entre los nuevos controles y sus homólogos, en los que se han basado.*

4.3.2. Formularios de la plataforma de administrador.

Trataremos en este apartado los formularios que conciernen a las partes de la plataforma de administrador que atañen a este proyecto.

4.3.2.1. Formulario principal y formulario explorador.

La plataforma de administrador fue la que se empezó a diseñar primero, sin una idea demasiado concreta en cuanto a diseño. Dado que aquella era la primera vez que se creaban formularios para una aplicación del estilo del paquete E.R.P. *Axiwin*, no sabíamos si centrarnos más en la vistosidad o en la funcionalidad. La idea de utilizar un formulario principal y un formulario explorador combinados dio muy buen resultado y por este motivo se volvió a implementar este diseño en la plataforma cliente.

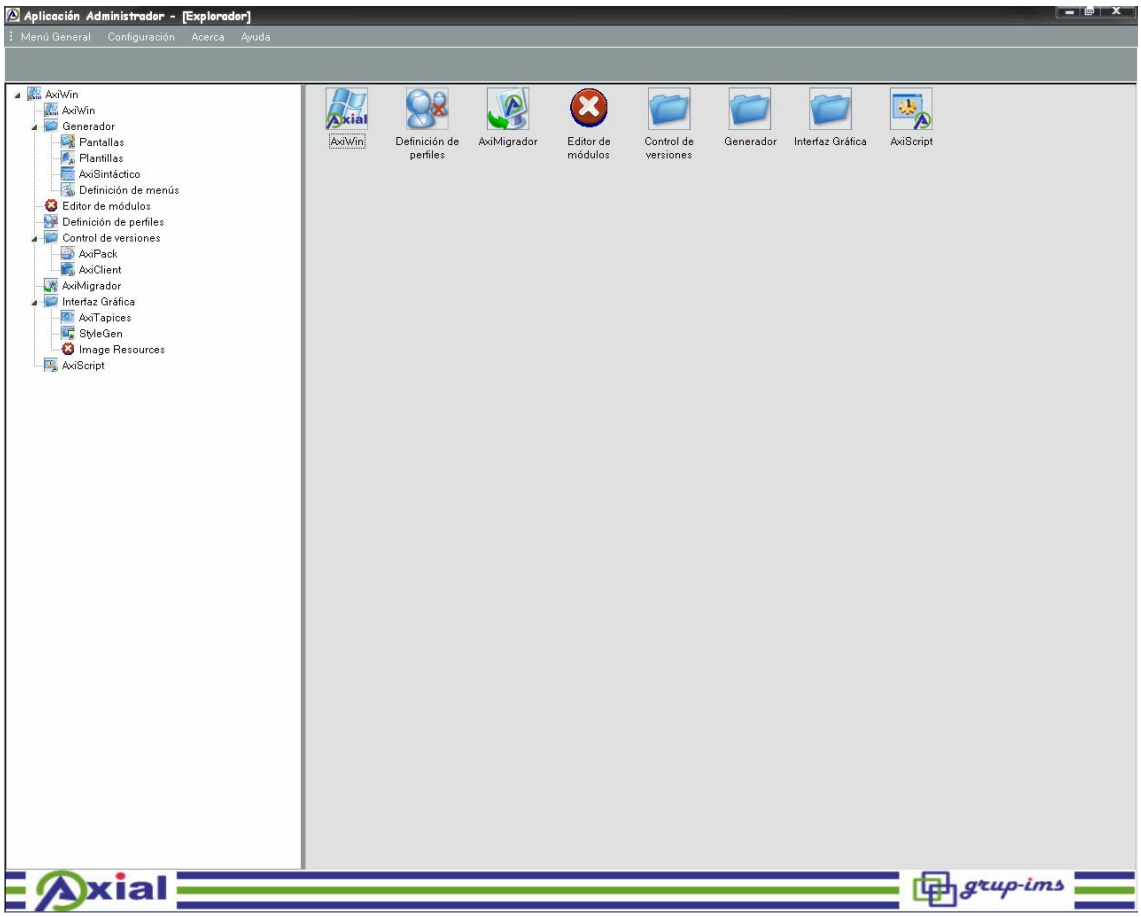


***Figura 4-17**. Formularios 'principal' y 'explorador' de la plataforma de administrador.*

El formulario principal es de tipo *MdiContainer*, y simplemente consta de un menú principal y dos paneles: uno inferior, en el que mostramos una imagen corporativa de tipo cenefa de la empresa AxialSoft, S.L., y otro superior, que se utiliza

para mostrar información al usuario en ocasiones determinadas. El formulario explorador es el primer formulario hijo *MdiChild* del formulario principal, y contiene básicamente un control de tipo *Treeview* (árbol explorador de aplicaciones) y un control de tipo *Listview*. El control más importante es el árbol, ya que en él se carga la información de las herramientas disponibles en la plataforma de administrador.

Al seleccionar un nodo del árbol hay dos opciones:

a) Si el nodo seleccionado tiene hijos se expandirá su contenido en el *Treeview* a la vez que en el *Listview*, que ejerce de control secundario.
b) Si el nodo seleccionado no tiene hijos se tratará de un nodo *hoja*, y estará por lo tanto asociado a una aplicación. Se procede entonces a ejecutar dicha aplicación en formato de diálogo sobre el formulario principal.

Este comportamiento es exactamente el mismo del explorador de Windows. Las aplicaciones de la plataforma de administrador son herramientas para el usuario programador de AxialSoft, S.L.; algunas de estas herramientas son:

- ✓ Generador de estilos.
- ✓ Generador de recursos de imágenes.
- ✓ Editor de perfiles de acceso.
- ✓ Editor de pantallas.
- ✓ Precompilador.
- ✓ Generador de paquetes de instalación y de actualización.
- ✓ Migrador.

En los próximos apartados analizaremos detalladamente las herramientas cuyo desarrollo ha estado incluido en la realización de este proyecto.

4.3.2.2. Generador de estilos - StyleGen.

Desde esta aplicación el usuario puede crear nuevos estilos para las plataformas de administrador y cliente, generando *hojas de estilos.* Podrá también modificar y/o eliminar los estilos existentes. El entorno es muy visual e intuitivo, lo que convierte al *Generador de estilos* en una herramienta sencilla de utilizar.

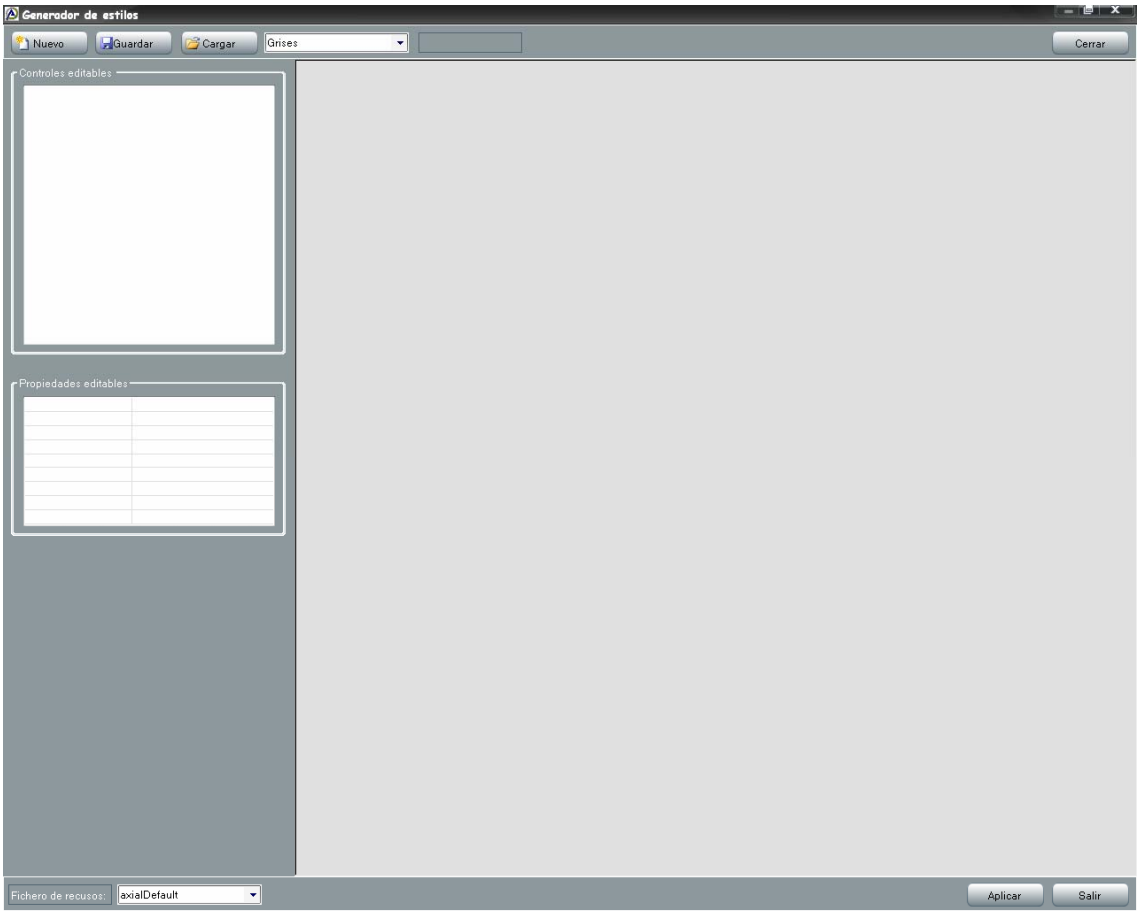


***Figura 4-18**. Pantalla principal del generador de estilos StyleGen.*

- ***Cómo crear un estilo.***

Al pulsar el botón *Nuevo* se abre un pequeño formulario en el que el usuario debe introducir un nombre descriptivo para el estilo (p.ej. 'dark Side') y el nombre identificador del estilo (p.ej. 'sith'). Por supuesto, no se pueden repetir nombres de estilos ya existentes.

***Figura 4-19**. Formulario de creación de nuevo estilo.*

Al pulsar el botón *Aceptar* en la figura 4-19, el fichero de plantilla de estilo se copia con el nuevo nombre y se carga el nuevo fichero en el generador de estilos. ¡Ya podemos empezar a diseñar!

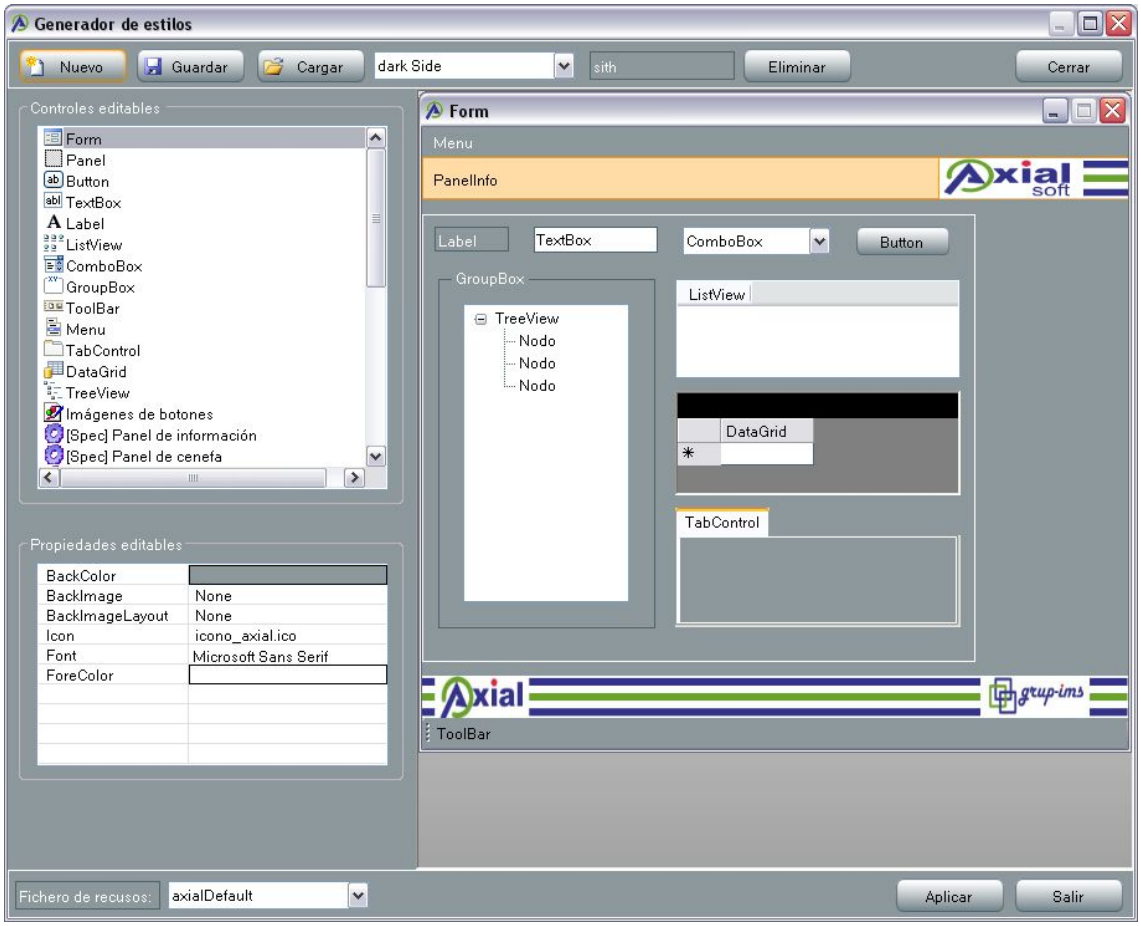


***Figura 4-20**. Para crear un nuevo estilo partimos de la plantilla gris por defecto.*

Los estilos creados se guardan físicamente en un mismo directorio, así como se introducen sus nombres en la base de datos. En la parte superior del formulario podemos observar que hay un control de tipo *Combobox* que contiene los nombres de todos los estilos existentes en el sistema (los que constan en la base de datos). Desde aquí podemos seleccionar un estilo y cargarlo mediante el botón *Cargar*.

El *Combobox* en la parte inferior izquierda del formulario contiene los nombres de todos los archivos de recursos de imágenes existentes, cuyos nombres se encuentran también en la base de datos. Seleccionando un archivo de recursos de imágenes de este *Combobox* lo asignamos al estilo, de manera que la próxima vez que

se cargue el estilo en la plataforma *Axiwin* lo hará usando el paquete de imágenes asociado.

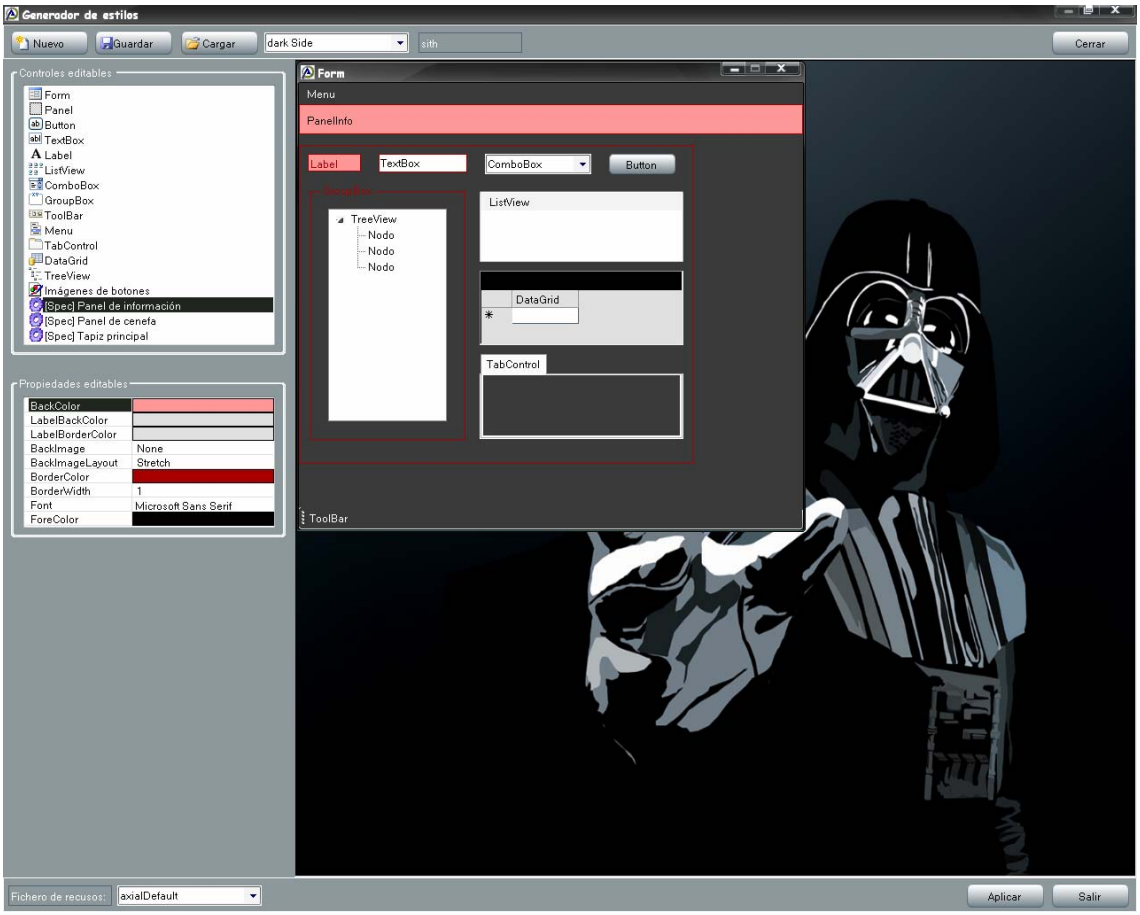


***Figura 4-21**. Partiendo de la plantilla gris original… ¡podemos llegar a esto!*

Mediante el botón *Aplicar* podemos seleccionar el estilo activo, para lo cual deberemos haber cargado el estilo previamente, seleccionándolo del *Combobox* de estilos disponibles y pulsando el botón *Cargar*. La próxima vez que ejecutemos la plataforma de administrador podremos ver el estilo aplicado a todos los controles.

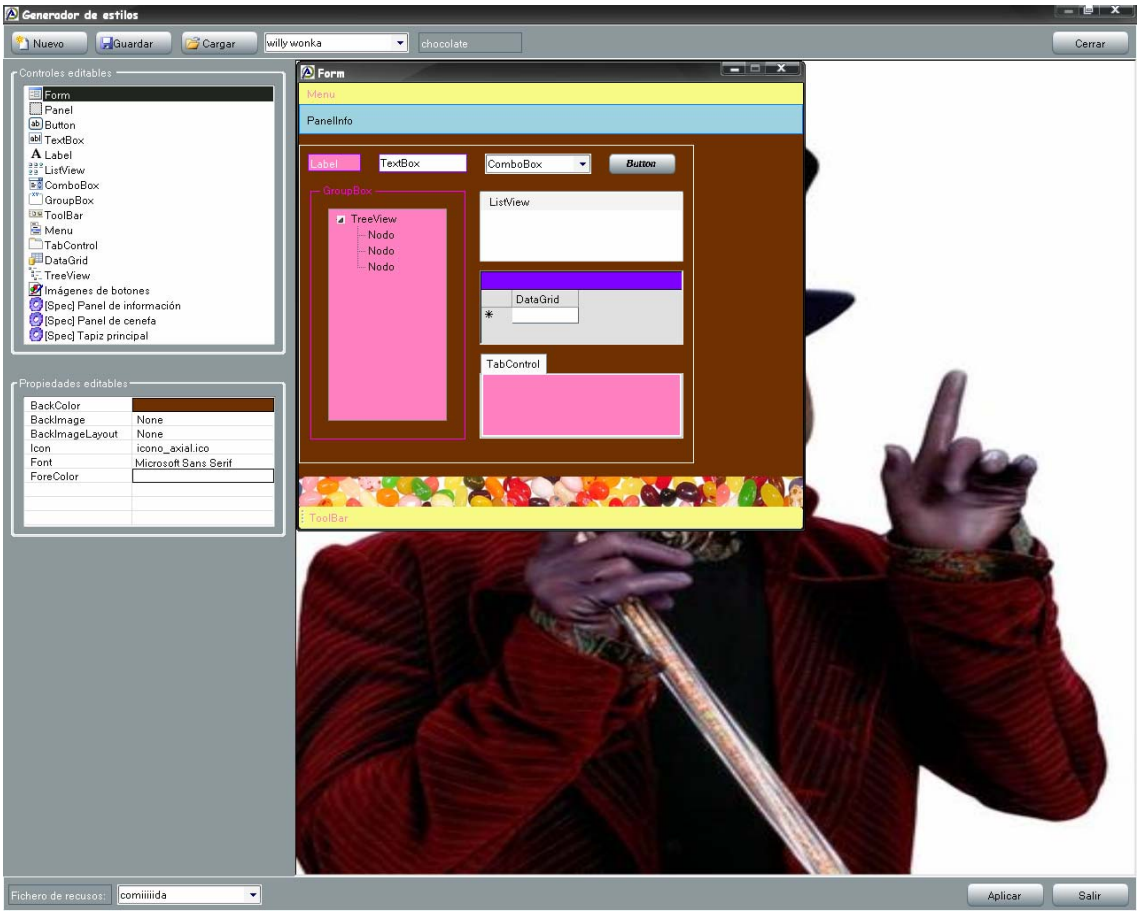


***Figura 4-22**. Estilo cargado en el generador de estilos. Vamos a seleccionarlo como estilo activo pulsando el botón Aplicar.*

Para guardar los cambios antes de salir del generador de estilos es necesario pulsar el botón *Guardar*. Mediante el botón *Cerrar* podemos cerrar el estilo cargado para edición, liberando los recursos del formulario de vista previa.

- ***Modificar un estilo existente.***

Es necesario cargar el estilo primero, y después tan sólo hay que editar las propiedades de los controles normalmente.

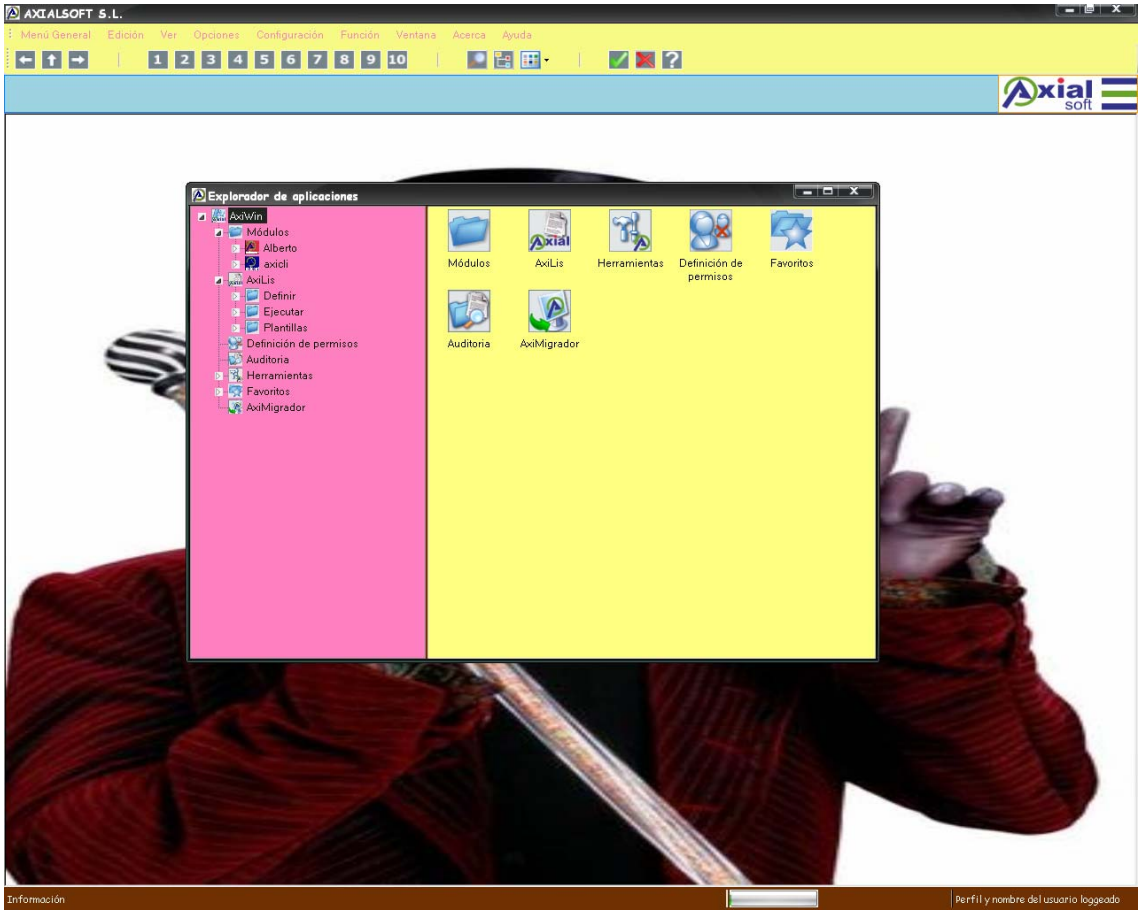


***Figura 4-23**. Aspecto del formulario principal y el formulario explorador del cliente después de aplicar el estilo seleccionado en la figura 4-22.*

4.3.2.3. Generador de recursos de imágenes – ImageResourceCreator.

Todos los iconos, papeles tapiz, y cualquier otro tipo de imágenes que aparecen en las plataformas de administrador y cliente están "empaquetados" y encriptados en un mismo archivo de recursos (.*resources*). De este modo el usuario no tiene fácil acceso a los ficheros de imágenes, y éstos no pueden ser corrompidos.

Los *sets* de iconos están asociados a los estilos para facilitar la tarea de combinar los colores o patrones de ambos. Para crear nuevos *sets* de iconos el usuario

utiliza el *Generador de recursos de imágenes*, aplicación desde la que puede también añadir y/o quitar iconos a archivos de recursos existentes.

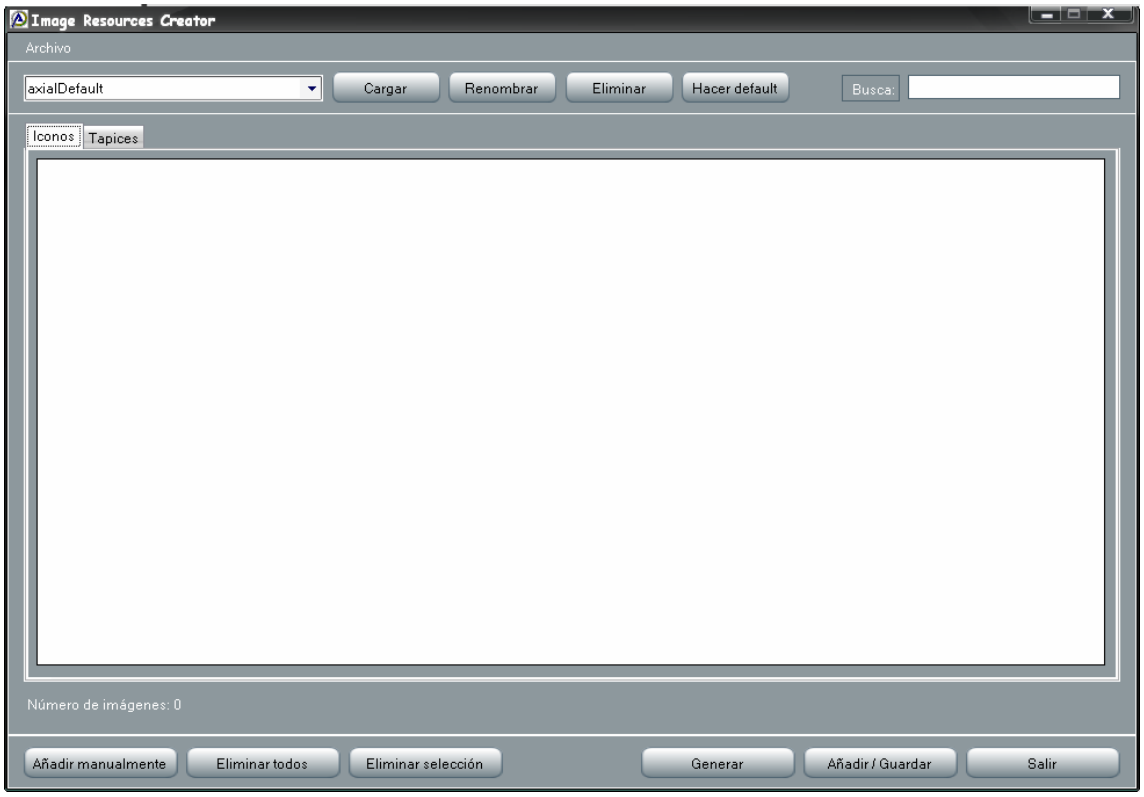


***Figura 4-24**. Pantalla principal del Generador de recursos de imágenes.*

Teniendo en cuenta que los iconos de las plataformas de administrador y cliente se cargan, en la mayoría de los casos, por sus nombres, esto supone un pequeño problema a la hora de generar nuevos paquetes de iconos ya que, de no encontrar los iconos bajo los nombres predeterminados en el momento de la carga, los formularios se cargan sin ellos quedando, en cierto modo, desnudos. Por ello es imperativo, al generar un nuevo paquete de iconos, que exista el número de iconos preestablecido y que se utilicen los nombres predefinidos. Si el usuario se deja algún icono al generar un nuevo archivo de recursos se dará cuenta enseguida, porque donde debería aparecer el icono descuidado aparecerá el icono de error por defecto.

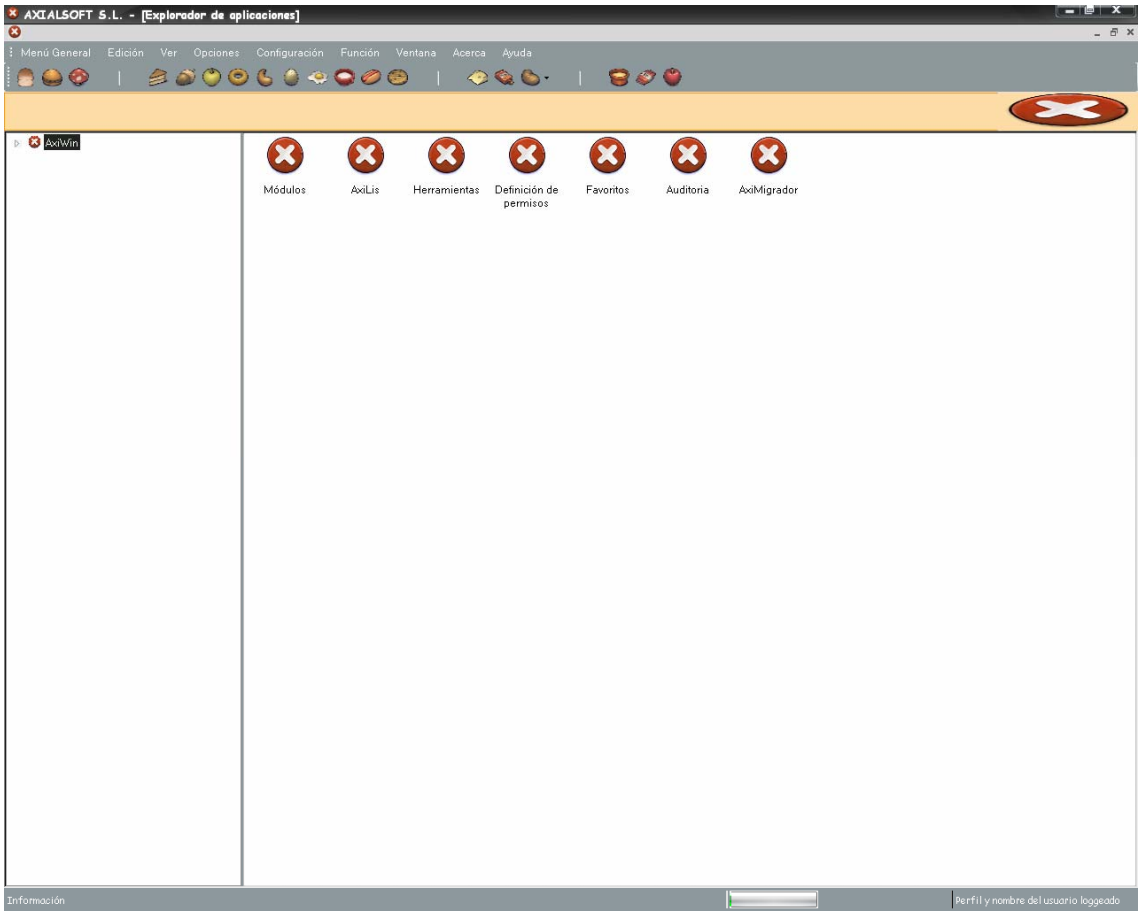


***Figura 4-25**. Las imágenes predefinidas por código que no se encuentren en el archivo de recursos de imágenes activo son sustituidas en tiempo de ejecución por el clásico icono de error, informando así al usuario del problema.*

A diferencia del *Generador de estilos* el *Generador de recursos de imágenes* no dispone de una plantilla por defecto de la que el usuario pueda partir para generar nuevos paquetes de iconos, sino que usará el listado de iconos predefinidos como referencia.

- ***Cómo crear un archivo de recursos de imágenes.***

Al abrir el generador de recursos de imágenes ya podemos empezar a añadir imágenes al control de tipo *Listbox* que ocupa la mayor parte del formulario principal. Para añadir imágenes podemos simplemente seleccionarlas, arrastrarlas con el *mouse* sobre el *Listbox* y soltarlas, o hacerlo mediante la opción de menú *Archivo* → *Añadir imágenes manualmente*. Podemos apreciar que el *Listbox* está contenido en un control de tipo *TabControl* que consta de dos pestañas. Las imágenes añadidas al *Listbox* cuando la pestaña de *Iconos* está seleccionada se añadirán al archivo de recursos de

imágenes como iconos. Si queremos incluir papeles tapiz en el archivo de recursos deberemos hacerlo seleccionando previamente la pestaña de *Tapices* y añadir las imágenes normalmente. Cualquier imagen de tipo tapiz que no añada al *Listbox* de tapices no constará como tal en el archivo de recursos de imágenes generado.

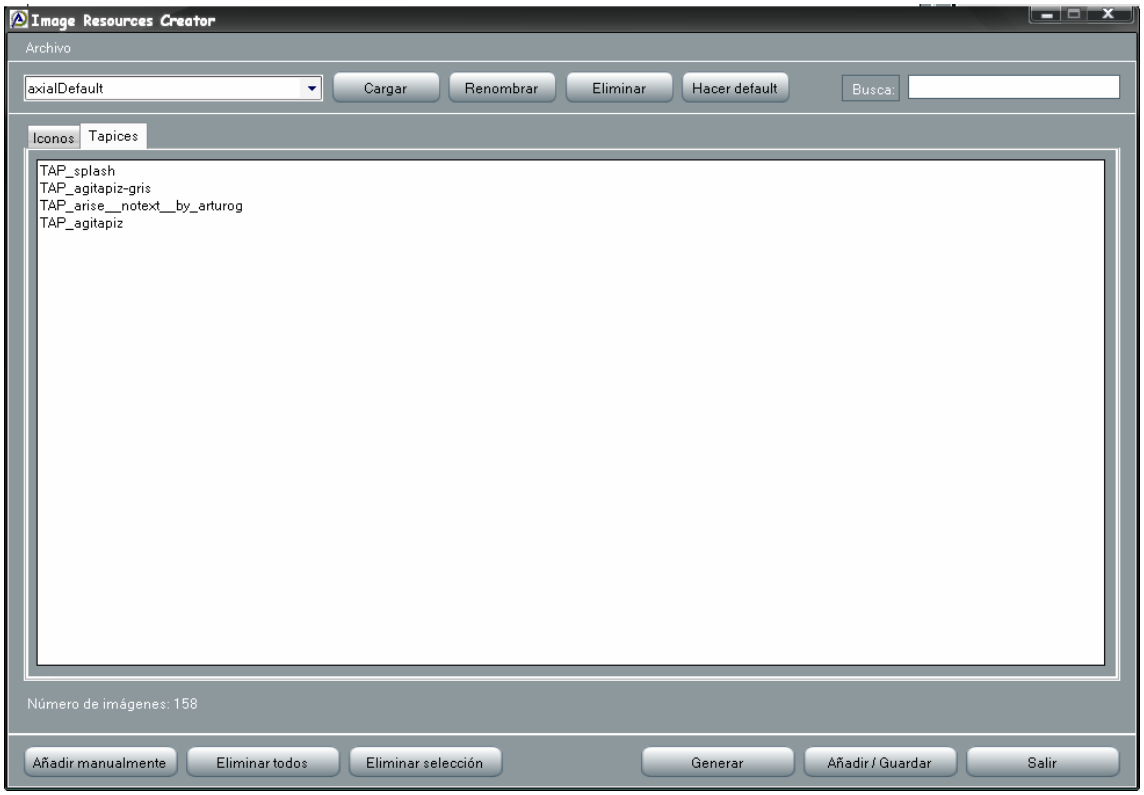


Figura 4-26. *Es necesario seleccionar la pestaña de Tapices para añadir papeles tapiz al archivo de recursos de imágenes.*

Al terminar de añadir imágenes pulsamos el botón *Generar* y se abrirá un pequeño formulario en el que deberemos introducir el nombre del nuevo archivo de recursos de imágenes. La opción de *Generar* permite no sólo crear nuevos archivos de recursos sino también sobrescribir por completo un archivo de recursos existente. Por este motivo si escribimos el nombre de un archivo existente en la base de datos se nos preguntará mediante un diálogo modal si estamos seguros de sobrescribir el archivo de recursos de imágenes (ver figura 4-28).

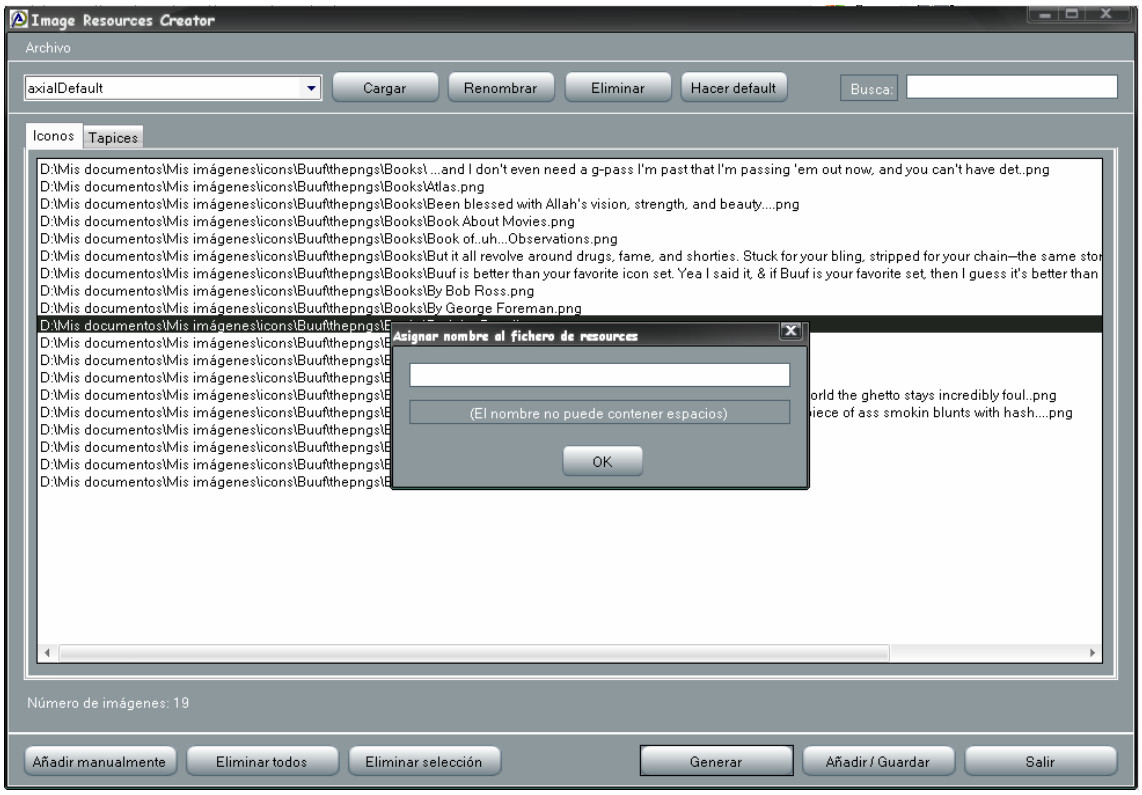


***Figura 4-27**. Al pulsar el botón Generar se abre este pequeño diálogo.*

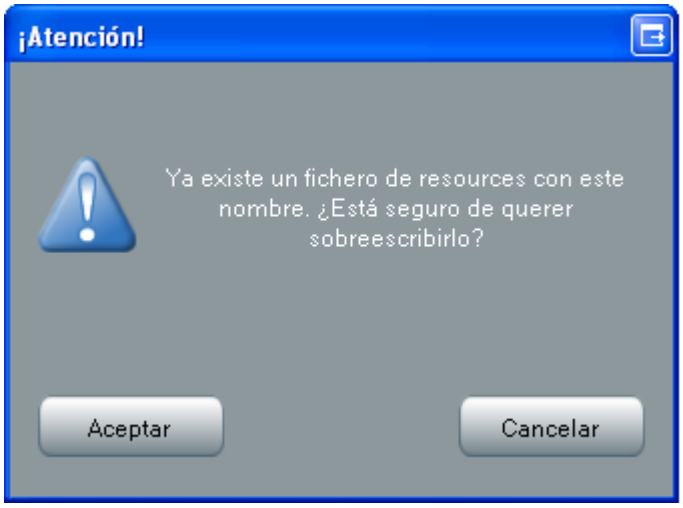


***Figura 4-28**. Diálogo modal para verificar la generación de un archivo de recursos existente.*

- ***Modificar un archivo de recursos de imágenes.***

No es necesario cargar un archivo de recursos existente para añadirle iconos o tapices. Simplemente debemos seleccionar el nombre del archivo de recursos a modificar en el *Combobox* superior del formulario, y añadir las imágenes

normalmente, arrastrándolas sobre el *Listbox* o mediante la opción de menú correspondiente. Al terminar pulsaremos el botón *Añadir/Guardar*.

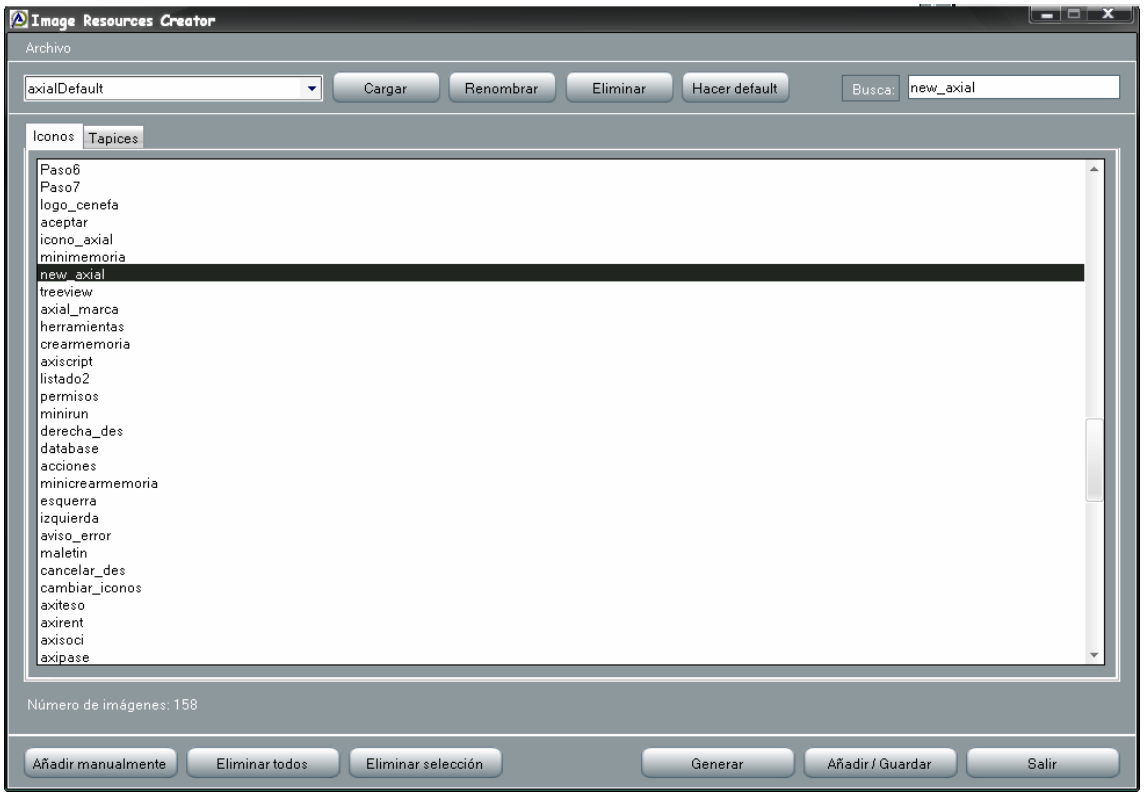


***Figura 4-29**. Vista del formulario principal tras cargar un archivo de recursos de imágenes mediante el botón Cargar.*

Si en cambio lo que queremos es eliminar imágenes del archivo de recursos deberemos cargar el recurso seleccionándolo previamente en el *Combobox* y pulsando el botón *Cargar*. Podemos buscar las imágenes por su nombre utilizando la caja de texto de la parte superior derecha del formulario y eliminarlas de una en una, o eliminar una selección múltiple, mediante el botón *Eliminar selección* (ver figura 4-30).

En el archivo de recursos no pueden existir, como es lógico, dos imágenes con el mismo nombre. Para sustituir una imagen por otra con el mismo nombre será necesario cargar el archivo de recursos, borrar la imagen en cuestión y guardar el archivo de recursos. Acto seguido podremos añadir la imagen de sustitución y volver a guardar el archivo de recursos. Será necesario reiniciar la plataforma para ver los cambios realizados en un archivo de recursos.

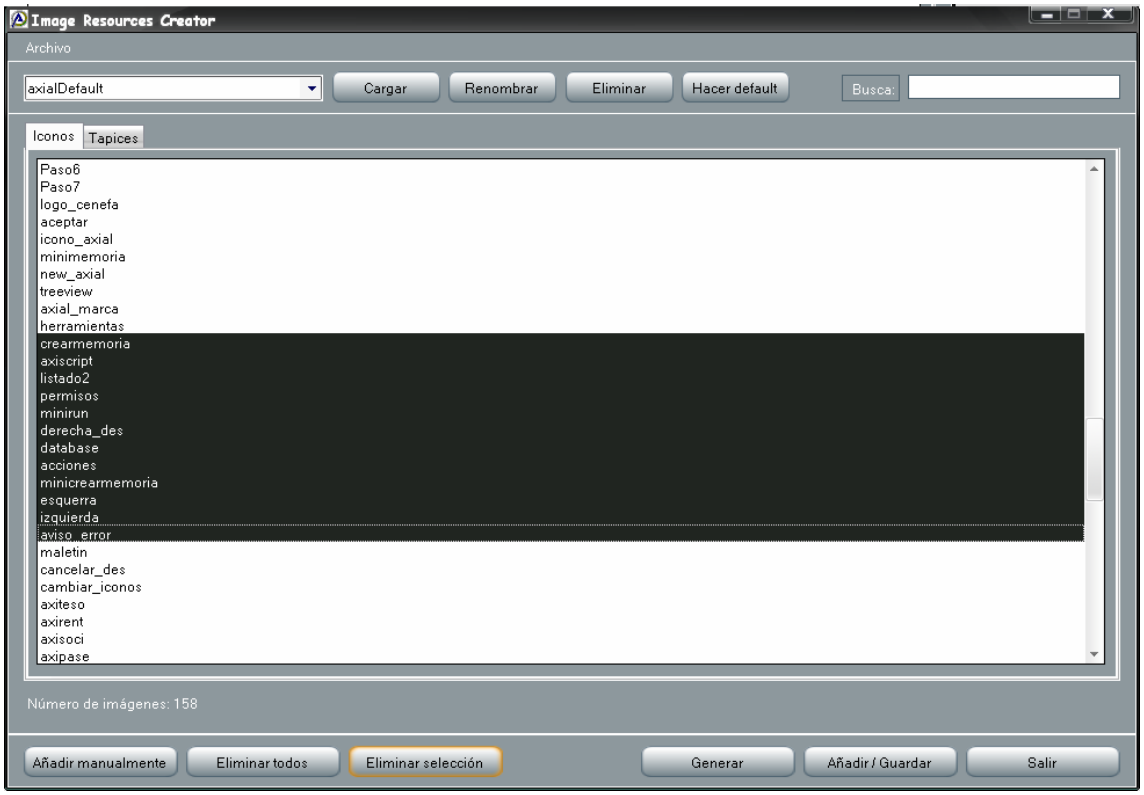


***Figura 4-30**. Eliminación de una selección de imágenes del archivo de recursos.*

4.3.2.4. Herramienta de cambio de tapices.

Esta otra herramienta del administrador permite configurar los papeles tapiz de la plataforma de cliente. Como ya se ha comentado en el apartado anterior, todos los tipos de imágenes que aparecen en las plataformas de administrador y cliente se obtienen del archivo de recursos. Esto incluye por lo tanto a las imágenes de papel tapiz.

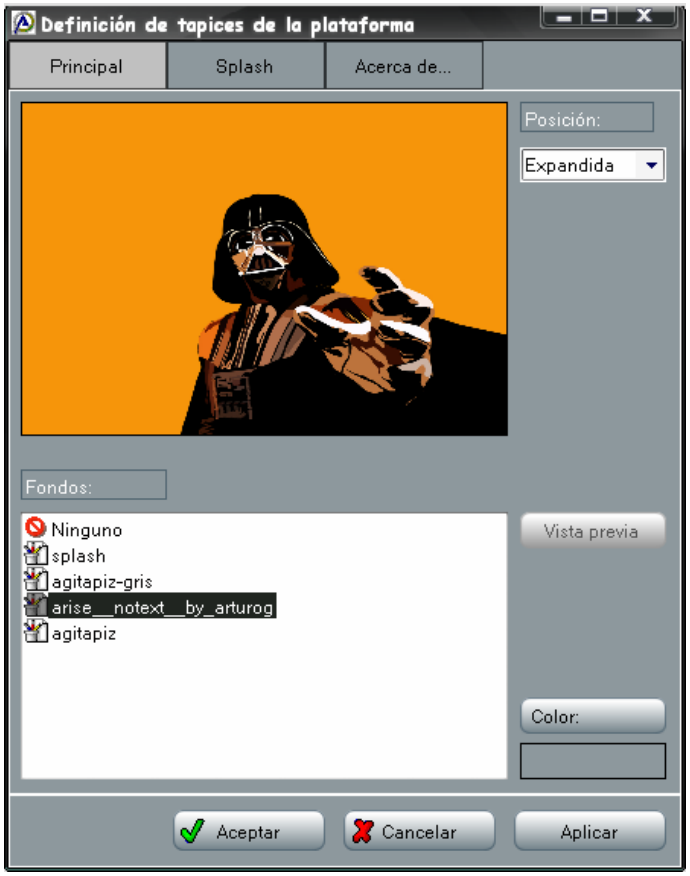


***Figura 4-31**. Pantalla principal de la herramienta de cambio de tapices.*

Los papeles de tapiz disponibles tendrán que estar previamente definidos como tales en el paquete de imágenes (archivo de recursos), en el que se guardan con el prefijo "TAP_".

El formulario de la aplicación de cambio de tapices consta de 3 pestañas que representan las 3 ubicaciones de los papeles tapiz en la plataforma de cliente: fondo del formulario principal, fondo del formulario de *splash* o bienvenida, y fondo del formulario de información de la plataforma (*Acerca de*). El papel tapiz del formulario principal del cliente se define en las *hojas de estilos*, pudiendo únicamente en este caso concreto utilizar imágenes no definidas previamente en el archivo de recursos, sino obtenidas de cualquier ubicación del sistema de ficheros local. Esto no es posible en cambio para los formularios *Acerca de…* y *splash*.

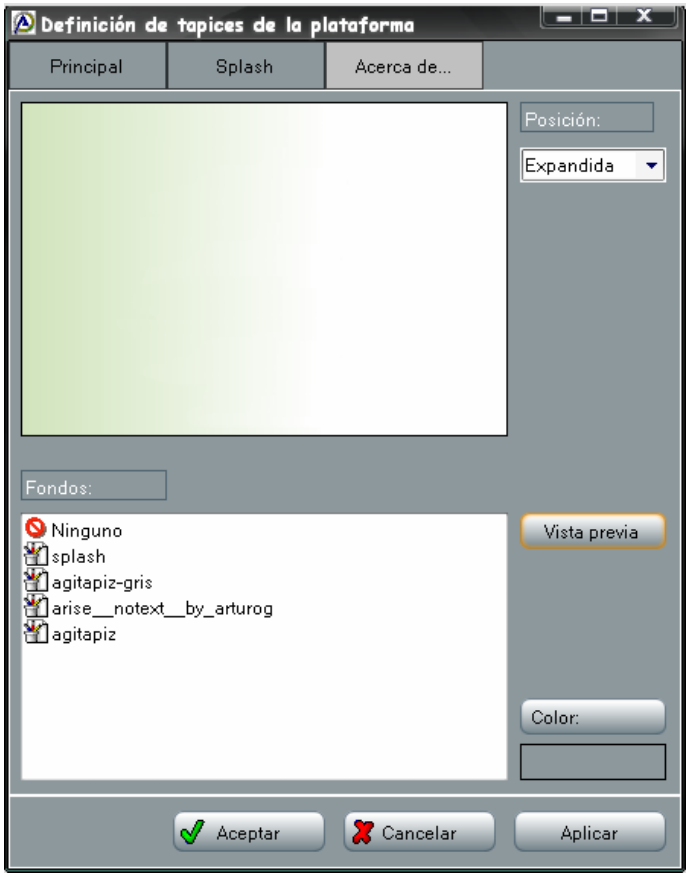


***Figura 4-32**. Vista de la herramienta de cambio de tapices al seleccionar la pestaña 'Acerca de'.*

Desde las pestañas destinadas a seleccionar los tapices de los formularios de *splash* y *Acerca de…* es posible obtener una vista previa de cómo quedará el papel tapiz seleccionado en el formulario en cuestión. Vemos un ejemplo del resultado de pulsar el botón de *Vista previa* en el formulario de la figura 4-32.

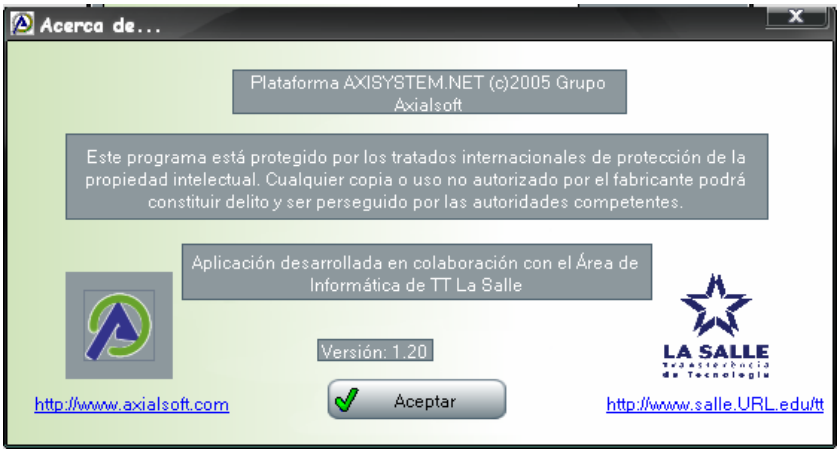


***Figura 4-33**. Vista previa del formulario de Acerca de… tras pulsar el botón de Vista previa en el formulario de la figura 4-32.*

4.3.2.5. Herramienta de edición de menús.

Al acceder a la plataforma de cliente, el usuario únicamente ve en el árbol explorador (y, por consiguiente, en el navegador principal del formulario, que ocupa la parte central del mismo) el conjunto de módulos que tiene asignados, es decir, los módulos sobre los que se le hayan concedido permisos de acceso.

Los permisos sobre los módulos del E.R.P. se conceden a los usuarios mediante la asignación de los menús modulares, los cuales definen el conjunto accesible de aplicaciones del módulo al que representan, así como la disposición de las mismas en el árbol explorador, tras ser cargadas.

El editor de menús permite crear menús modulares y/o modificar los ya existentes, cargándolos previamente de la tabla de la base de datos en la que están almacenados, *tblMenuMen*.

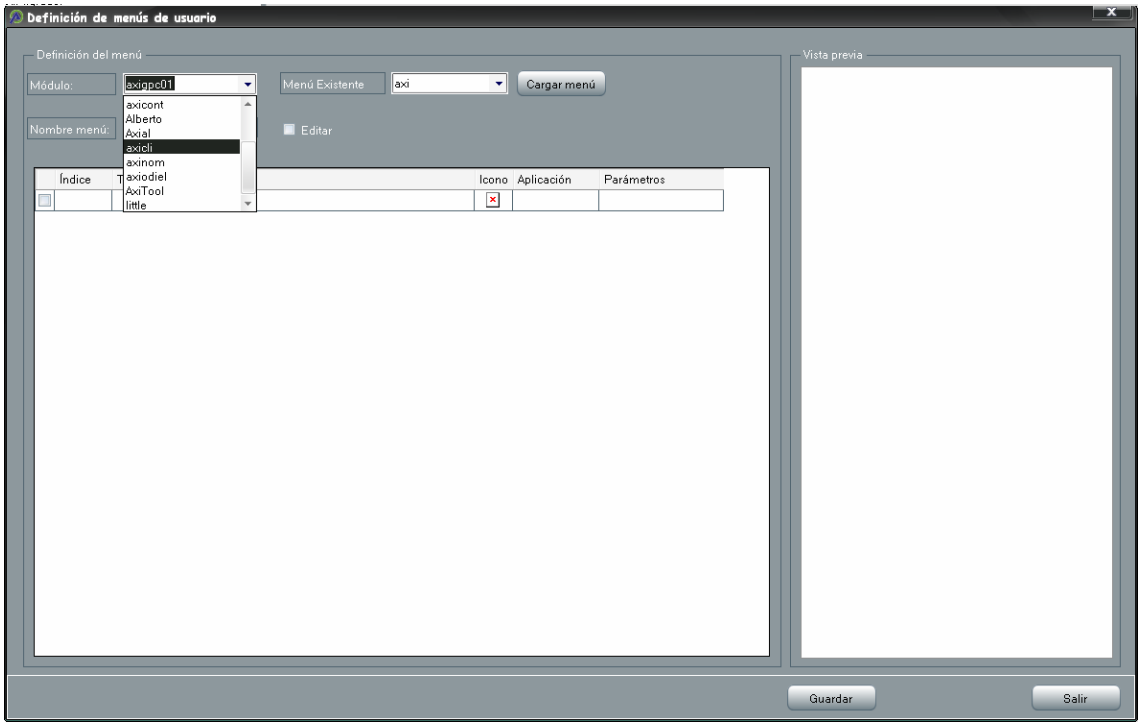


***Figura 4-34**. Formulario principal del editor de menús.*

El formulario principal del editor de menús dispone de un control de tipo *Combobox* en la parte superior izquierda, en el que se cargan los nombres de los módulos disponibles en la plataforma *Axiwin* (ver figura 4-34). El segundo *Combobox*, situado a la derecha del primero, permite seleccionar uno de los menús modulares existentes en la base de datos, y cargarlo en el editor mediante el botón *Cargar menú*.

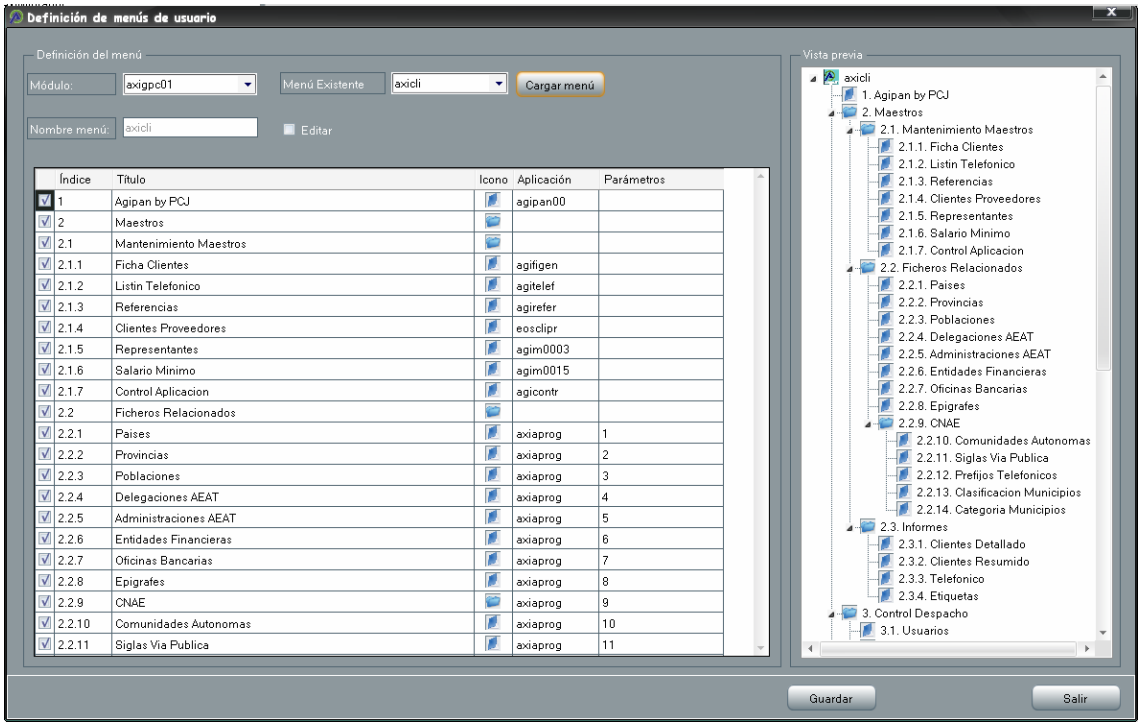


***Figura 4-35**. Aspecto que muestra el editor de menús tras cargar uno de los menús existentes en la base de datos.*

En la parte central del formulario se encuentra el editor de menús propiamente, que no es más que un control de tipo *DatagridView* en el que cada fila define una entrada de menú mediante las propiedades *Índice, Título, Icono, Aplicación* y *Parámetros*. Al dejar el campo *Aplicación* vacío en una entrada de menú, el editor interpreta que ésta define un directorio en vez de un acceso a una aplicación, asignándole automáticamente el icono pertinente (una carpeta cerrada). El árbol explorador a la derecha del formulario muestra en todo momento una vista previa del contenido del *DatagridView*.

- ***Crear o modificar un menú modular.***

Tanto si queremos modificar un menú cargado de la base de datos como si queremos crear uno nuevo, es preciso marcar previamente la casilla *Editar*, ya que por defecto el *DatagridView* está desactivado, impidiendo al usuario escribir en él. Tras activar la casilla de edición podemos empezar a introducir entradas de menú en el *DatagridView*. La caja de texto que se encuentra junto a la casilla *Editar* define el nombre del menú que se está creando o modificando (ver figura 4-36).

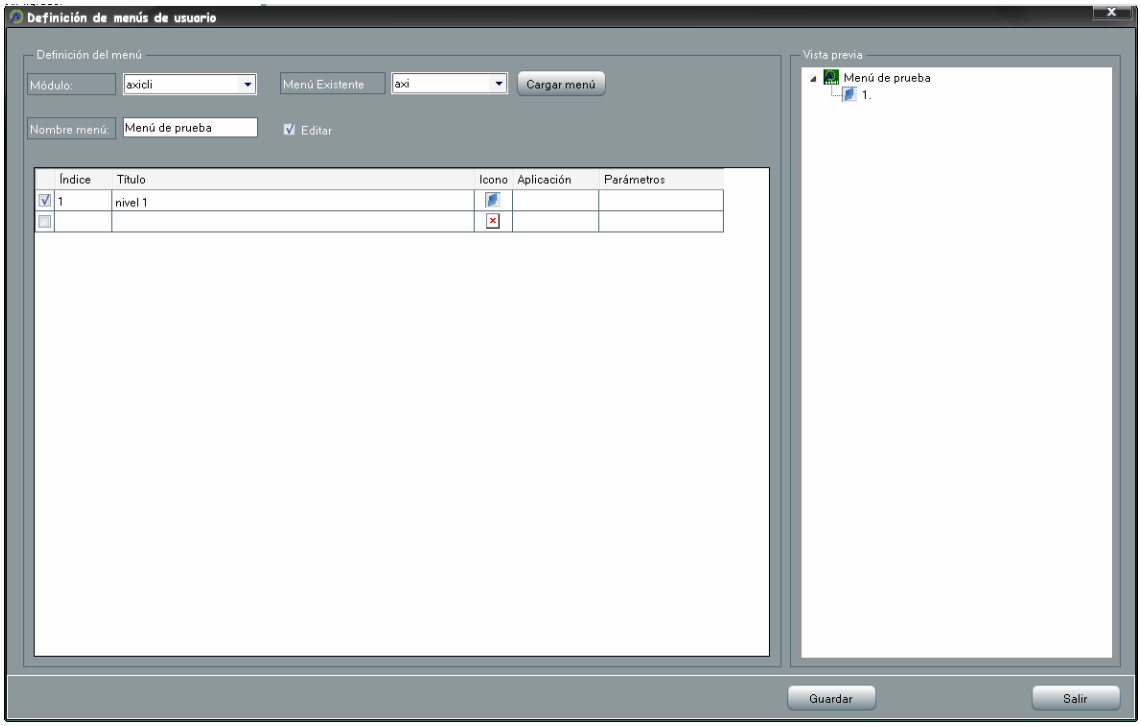


Figura 4-36*. Tras activar la casilla de edición podemos empezar a crear un nuevo menú.*

Dado que la finalidad de esta herramienta es la de permitir la creación/edición de menús modulares, en caso de crear un nuevo menú es necesario asociarlo a un módulo mediante la selección del mismo en el *Combobox* de la parte superior izquierda del formulario. Al terminar de editar el menú es preciso pulsar el botón *Guardar* para que los cambios queden registrados en la base de datos.

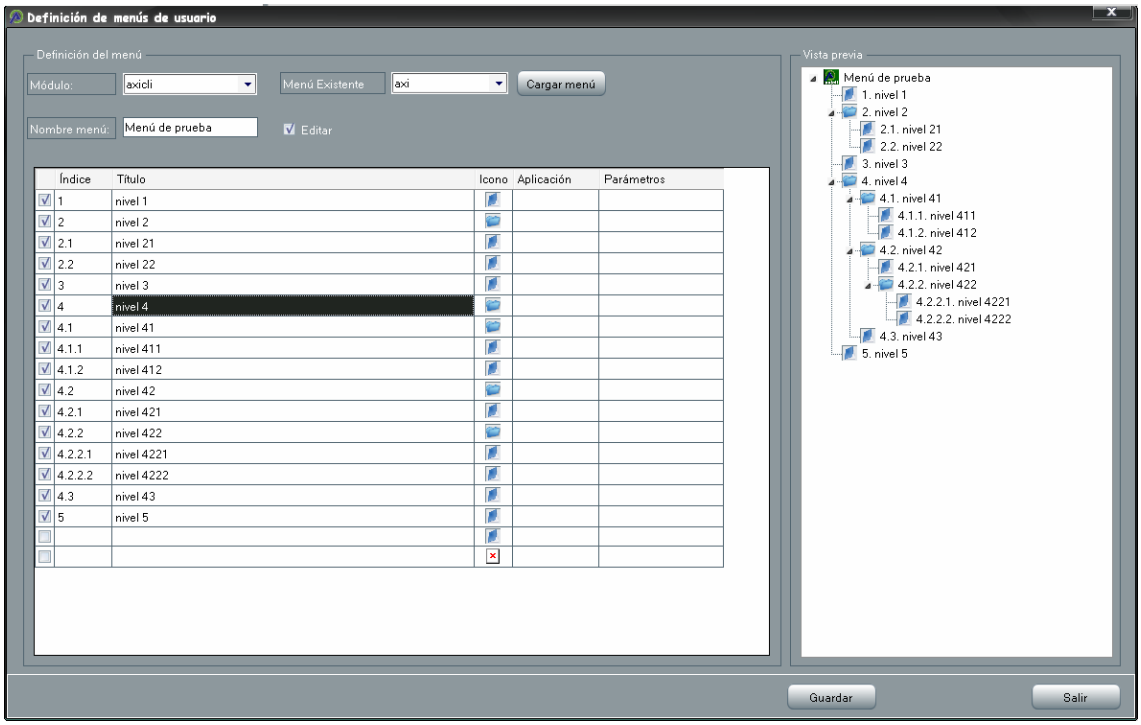


***Figura 4-37**. Ejemplo de creación de un menú modular con el editor de menús.*

4.3.2.6. Herramienta de edición de perfiles.

Los perfiles de usuario definen políticas de acceso a las plataformas de administrador y cliente mediante la configuración y combinación de 5 parámetros los cuales, a su vez, definen conjuntos de operaciones permitidas. Estos parámetros son:

- ✓ *Lectura*. Otorga permiso genérico para consultar los datos de las aplicaciones.
- ✓ *Escritura*. Otorga permiso genérico para realizar altas, bajas y modificaciones de los datos de las aplicaciones.
- ✓ *Listados*. Otorga permiso genérico para definir y obtener listados.
- ✓ *Herramientas*. Otorga permiso genérico para ejecutar el resto las aplicaciones de tipo *herramientas* (auditoria, control de índices, etc.)
- ✓ *MaxAltas*. Define el número máximo de altas permitidas a cada usuario perteneciente al mismo perfil.

***Figura 4-38**. Formulario de edición de perfiles. Los 5 perfiles visibles están predefinidos en la plataforma, pero pueden modificarse y/o eliminarse.*

- ***Creación de un perfil.***

Al pulsar el botón *Crear perfil* aparecerá un diálogo mediante el cual es posible crear un nuevo perfil de acceso, asignándole los permisos iniciales en el momento de crearlo. El nombre del perfil, por supuesto, no puede estar repetido.

***Figura 4-39**. Formulario de creación de un nuevo perfil.*

Los cambios realizados en los perfiles definidos se guardarán automáticamente al salir de la herramienta de edición de perfiles mediante el botón *Aceptar*. Es posible eliminar los perfiles seleccionándolos primero con un simple *clic* y pulsando el botón *Borrar.*

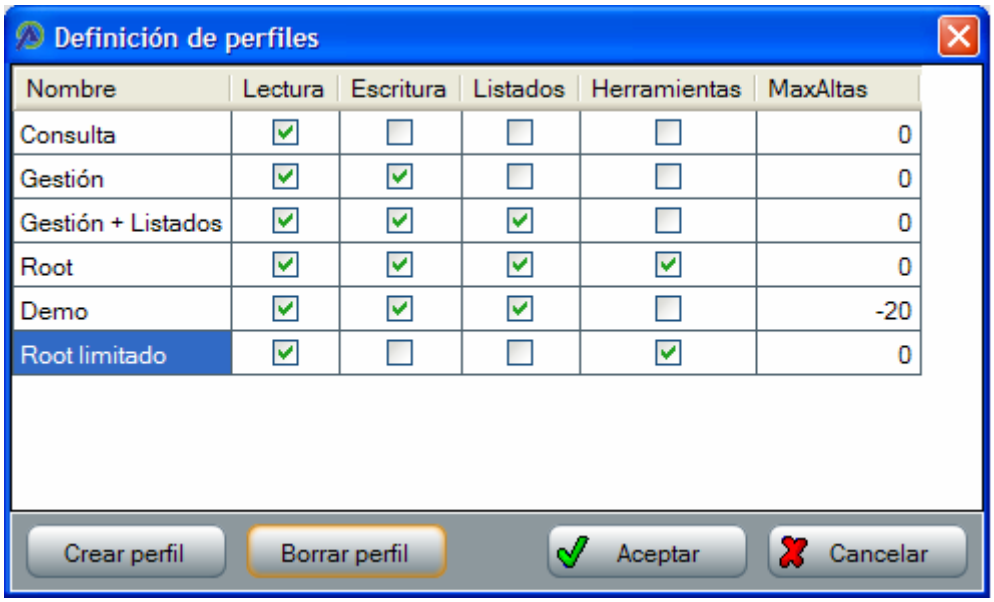


***Figura 4-40**. Tras seleccionar un perfil podemos borrarlo mediante el botón 'Borrar perfil'.*

4.3.3. Formularios de la plataforma de cliente.

Trataremos en este apartado los formularios que conciernen a las partes de la plataforma de cliente que atañen a este proyecto.

4.3.3.1. Formulario de login.

Para poder acceder a la plataforma de cliente, el usuario debe autenticarse mediante la introducción de su *login* (nombre de usuario) y contraseña en este formulario. De esta forma se identifica al usuario en el sistema y se procede a cargar su configuración personal, que contempla opciones tales como idioma de preferencia, teclas de función programadas, favoritos, etc. Tras la autenticación se cargan también, en el árbol explorador del formulario principal, los menús modulares que le hayan sido asignados al usuario, así como sus permisos específicos, y se le aplica el perfil de acceso al que esté ligado, según conste en la tabla *tblUsuario* de la base de datos.

El mismo formulario de *login* se utiliza también para autenticar a los usuarios de la plataforma de administrador.

Figura 4-41. *Formulario de login de las plataformas de administrador y cliente.*

4.3.3.2. Formulario principal del cliente y formulario explorador del cliente.

Pese a que son dos formularios los consideramos como un solo porque siempre se muestran a la vez. El formulario principal del cliente es un formulario de tipo *MdiContainer*, es decir que contiene otros formularios. El formulario explorador se podría considerar como el "recibidor de la casa", ya que desde él el usuario podrá acceder a todas las opciones disponibles en la plataforma de cliente. Este formulario es el primer formulario *MdiChildren* del formulario principal.

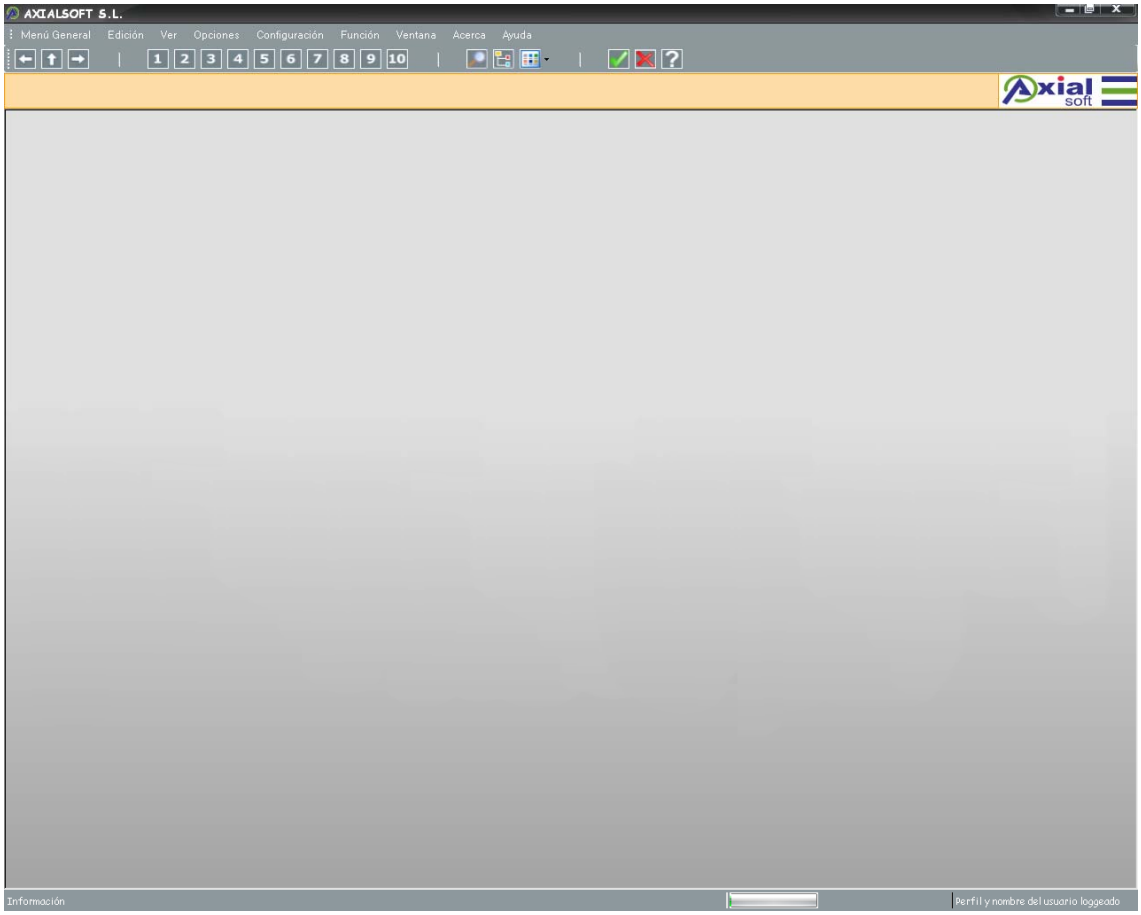


***Figura 4-42**. Formulario principal del cliente sin el explorador. Podemos aplicar un papel tapiz al fondo del MdiContainer, como se puede apreciar en esta captura.*

Las aplicaciones propias de la plataforma de cliente se cargan sobre el formulario principal también como *MdiChildren*, en cambio la gran mayoría de aplicaciones/herramientas se muestran en forma de diálogo.

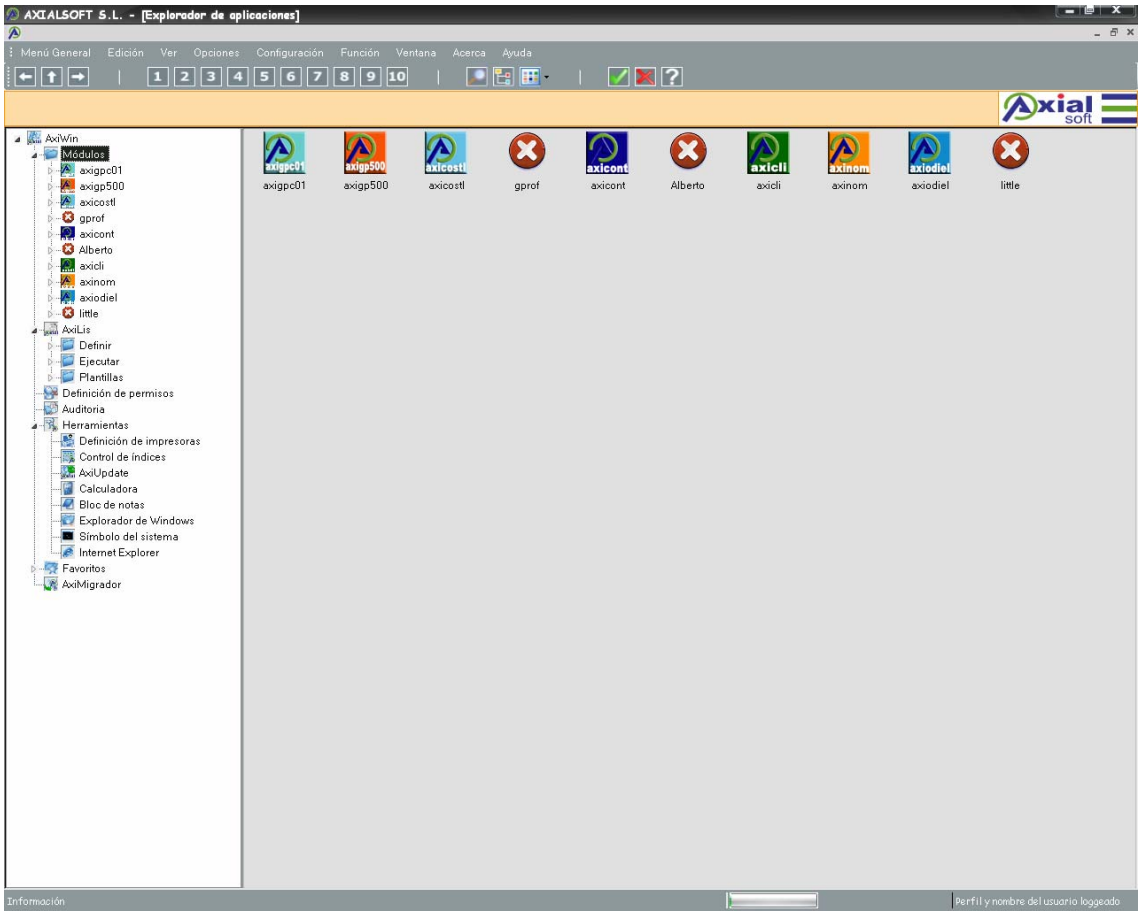


***Figura 4-43**. El formulario explorador del diente es el primer MdiChildren del formulario principal.*

Ya que el usuario deberá pasar inevitablemente por el formulario explorador en muchas ocasiones, nos pareció interesante considerar la posibilidad de que éste formulario nunca se cerrara realmente, aunque el usuario puede hacerlo "desaparecer" de la vista pulsando el botón de cerrar. En realidad, el formulario sólo se oculta, pero sigue estando cargado, y es posible hacerlo reaparecer mediante el botón de la botonera *Mostrar explorador*.

También existe la posibilidad de ocultar y volver a mostrar el árbol explorador de aplicaciones, pese a que ofrece un acceso más rápido que los iconos de la parte central del formulario explorador. Mediante el botón *Mostrar/ocultar árbol explorador* el usuario podrá ganar unos centímetros más en la parte central del formulario explorador.

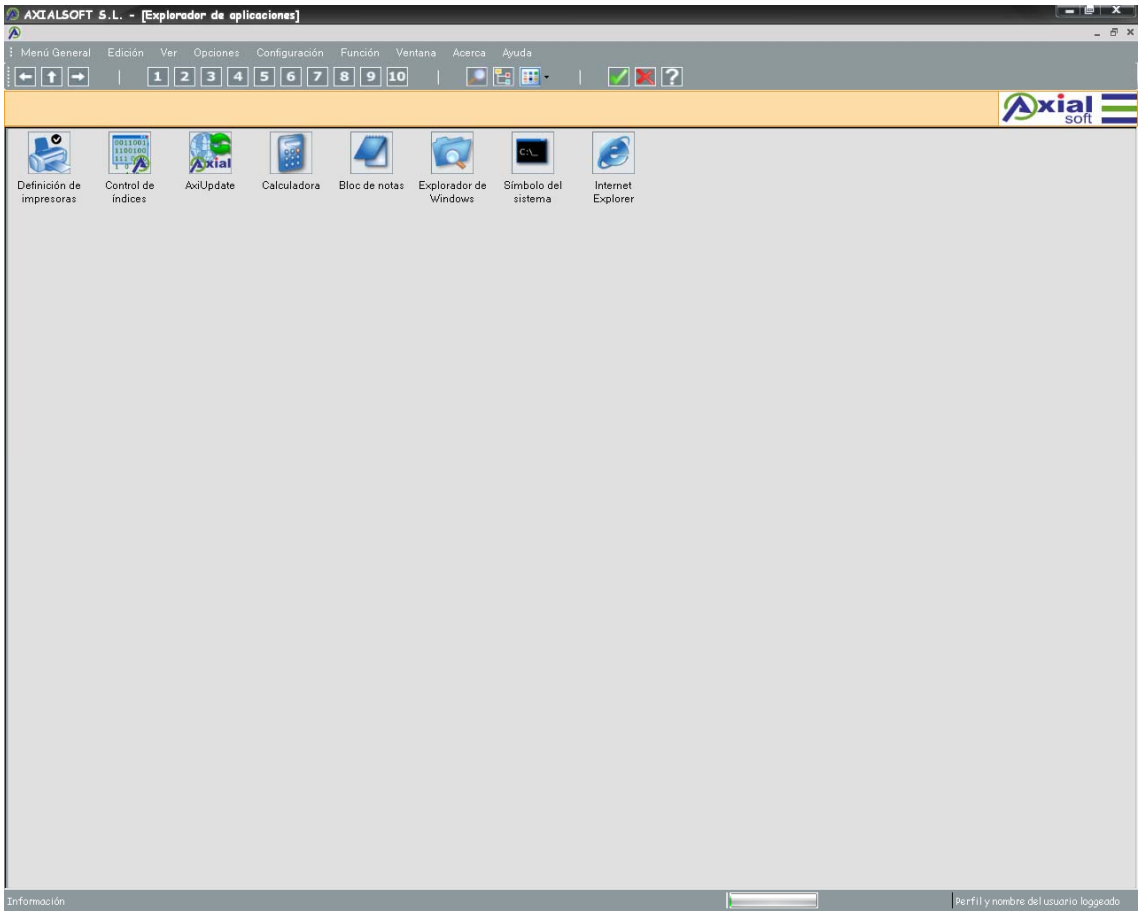


***Figura 4-44**. Vista del formulario explorador sin el árbol explorador de aplicaciones.*

La botonera de la parte superior del formulario principal también se puede ocultar mediante la opción de menú *Ver → Barras de herramientas → Botonera principal* (ver figura 4-45).

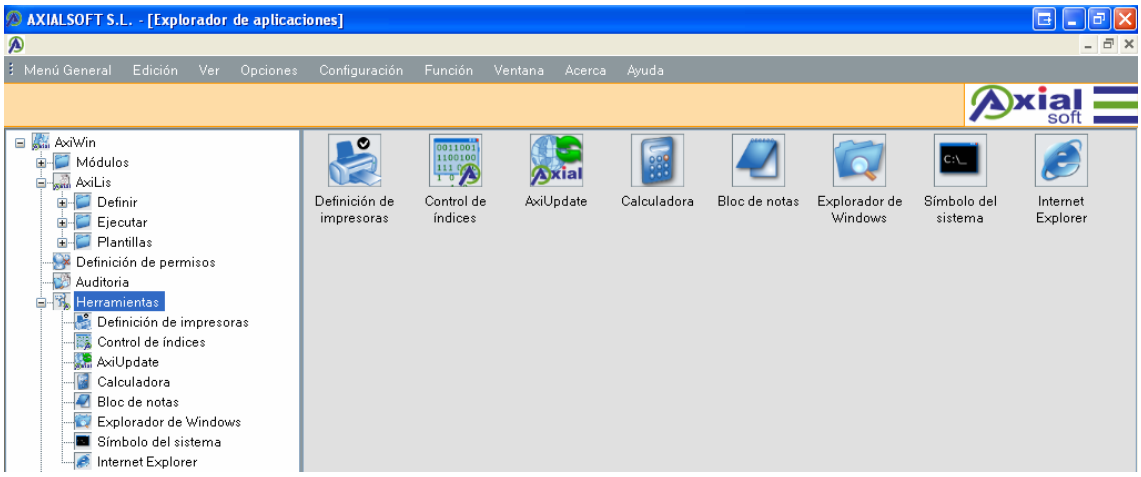


***Figura 4-45**. Vista del formulario explorador sin la botonera principal.*

Es igualmente posible mover la botonera de su lugar por defecto, y colocarla en cualquiera de los otros 3 lados del formulario, mediante la opción de menú *Opciones → Posicionar botonera* (ver figuras 4-47 y 4-48).

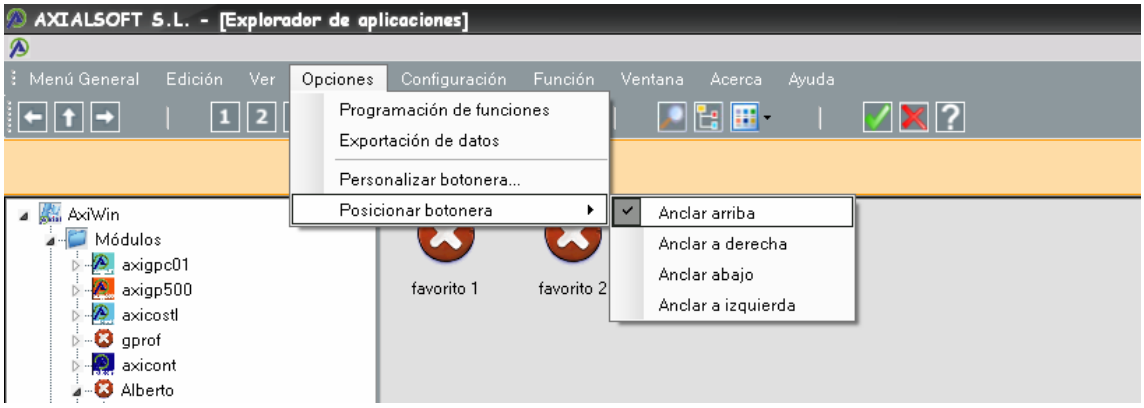


***Figura 4-46**. Mediante este menú podemos cambiar la posición de la botonera en el formulario principal.*

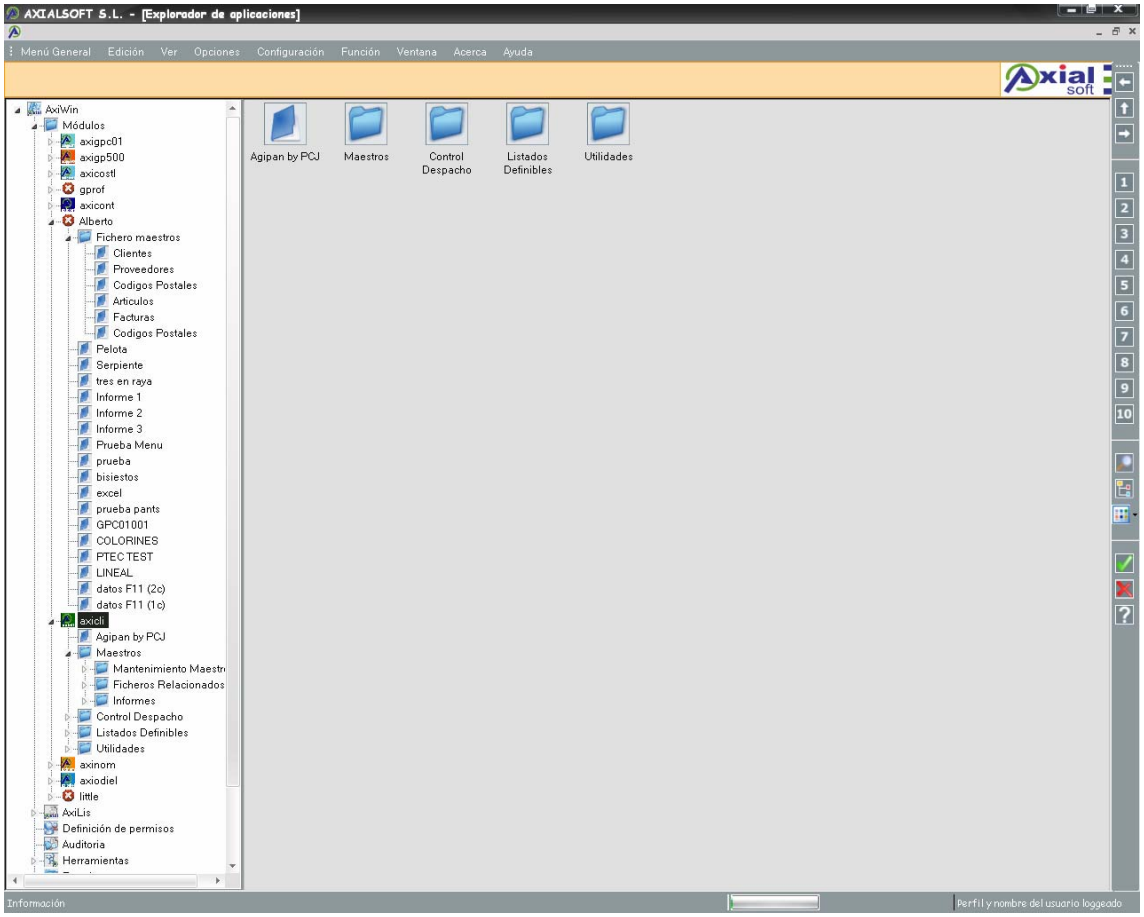


***Figura 4-47**. Aspecto del formulario principal con la botonera anclada a la derecha.*

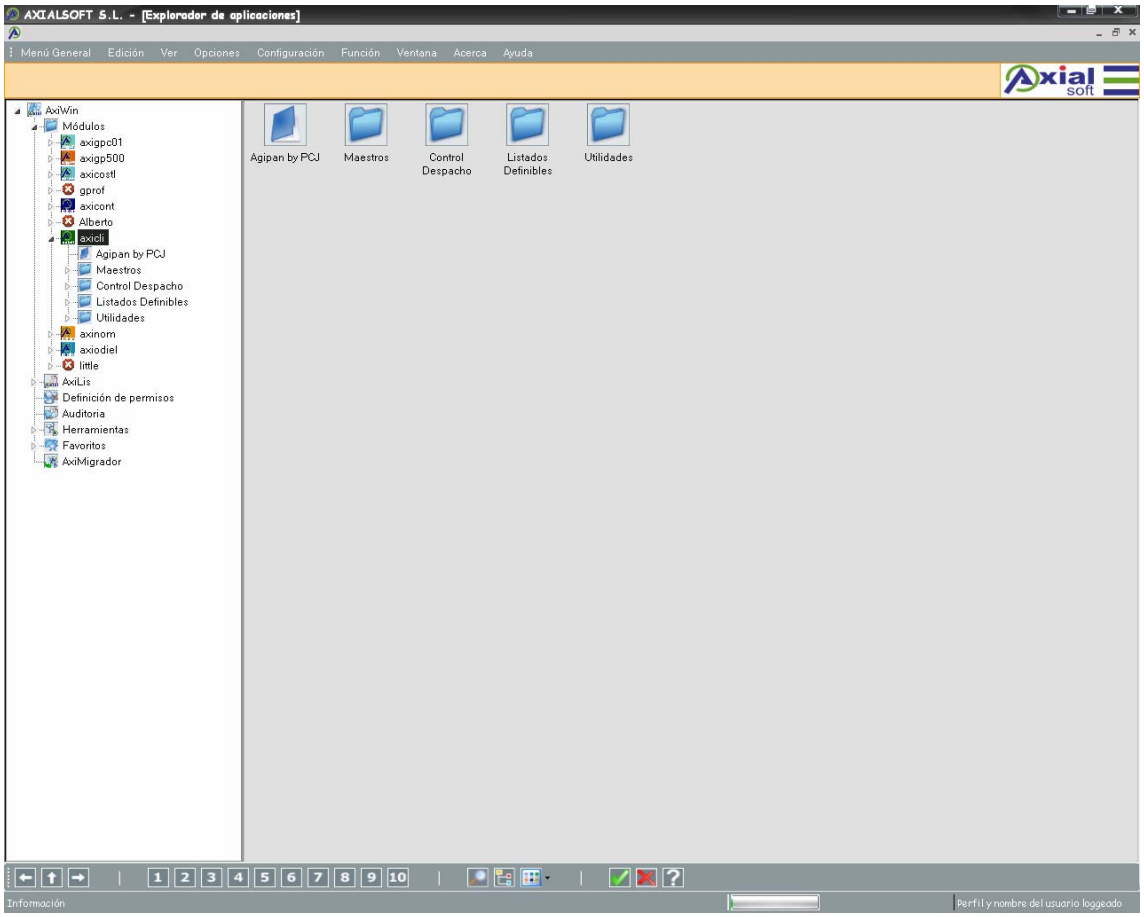


***Figura 4-48**. Aspecto del formulario principal con la botonera anclada abajo.*

La botonera principal es un control completamente personalizable y propio de cada usuario. La primera vez que un usuario se autentica en la plataforma cliente *Axiwin* se le muestra la botonera por defecto (la que podemos ver en la figura 4-43). El usuario puede entonces añadir y quitar botones a la botonera, así como cambiar los botones de lugar, mediante el formulario de personalización de la botonera, al que se accede por el menú *Opciones* → *Personalizar botonera* . El conjunto de botones que se pueden poner en la botonera está predefinido, de forma que cada botón tiene asociada una acción; esta información la recoge la tabla *tblAccionBotonera* en la base de datos. La distribución de botones escogida por el usuario debe guardarse en la base de datos para ser cargada cada vez que dicho usuario se autentique en la plataforma. A tal efecto utilizamos las tablas *tblBotonera* y *tbBotonBotonera*.

***Figura 4-49**. Formulario de personalización de la botonera. El control Listview de la izquierda muestra todos los botones disponibles, mientras que el Listview de la derecha muestra los botones que hay actualmente en la botonera. Mediante los botones Subir y Bajar podemos cambiar el orden de los botones en la botonera.*

La tabla *tblBotonera* contiene tantos registros como usuarios existan en el sistema, ya que cada usuario tiene su propia botonera. Por este motivo botoneras y usuarios están relacionados mediante el identificador numérico del usuario que se encuentra en la tabla *tblUsuario*. La tabla *tblBotonBotonera* contiene la definición del conjunto de botones que tiene cada botonera, relacionando los identificadores de botonera de la tabla *tblBotonera* con los identificadores de acción, o botón, de la tabla *tblAccionBotonera*.

cfBotonera	cfAccBot	orden	bModulo
1	1	0	False
1	2	1	False
1	3	2	False
1	4	4	False

idAccBot	tag	icono
1	32	esquerra
2	33	dalt
3	34	dreta
4	36	un

***Figura 4-50**. Ejemplo del contenido de las tablas tblBotonBotonera y tblAccionBotonera, mediante las cuales se definen los botones de las botoneras definidas en la tabla tblBotonera.*

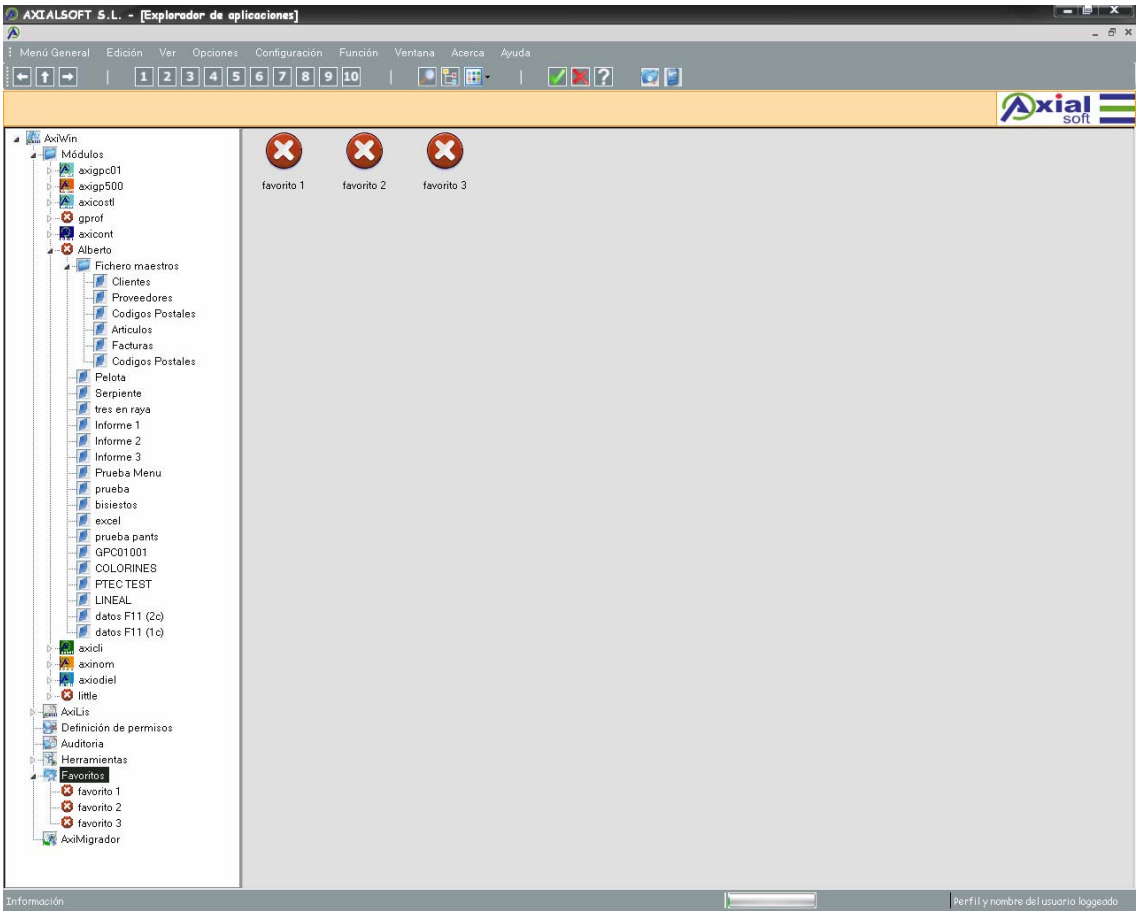


***Figura 4-51**. Podemos apreciar el aspecto de la botonera tras añadir dos botones mediante el formulario de personalización de la botonera, en la figura 4-49.*

4.3.3.3. Herramienta de edición de usuarios y permisos.

Desde el primer nivel del árbol explorador de aplicaciones podemos acceder al formulario editor de usuarios y permisos. Utilizando esta herramienta el usuario administrador del cliente puede definir nuevos usuarios, asignarles menús y editar sus permisos de acceso. Al acceder al editor vemos un listado de los usuarios definidos en el sistema, y los perfiles a los que pertenecen.

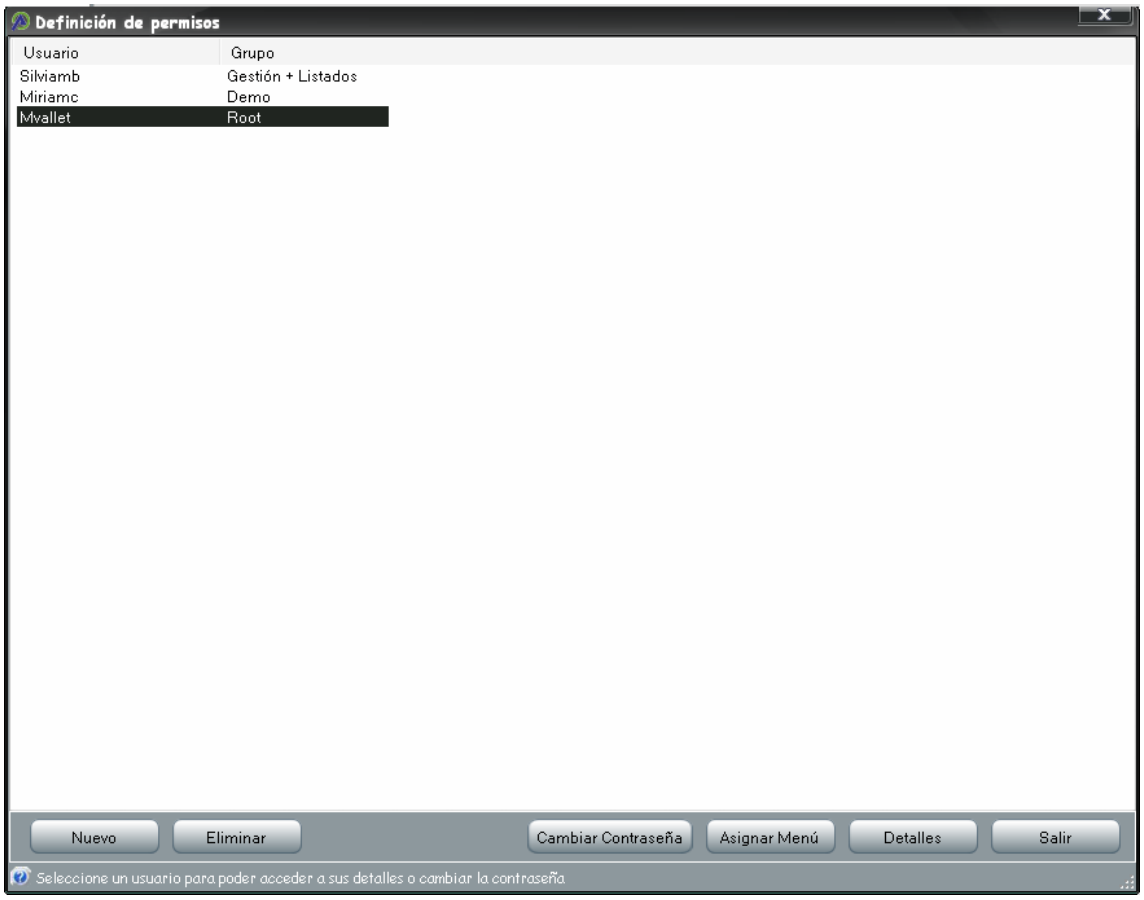


***Figura 4-52**. Pantalla principal del editor de usuarios y permisos.*

Podemos cambiar el perfil de un usuario haciendo *clic* sobre el nombre de su perfil actual, y seleccionando el nuevo perfil del *Combobox* que aparece a continuación (ver figura 4-53). Para ver los permisos de un usuario deberemos seleccionarlo primero en el listado y pulsar el botón *Detalles*. Se abrirá entonces la pantalla de la figura 4-54.

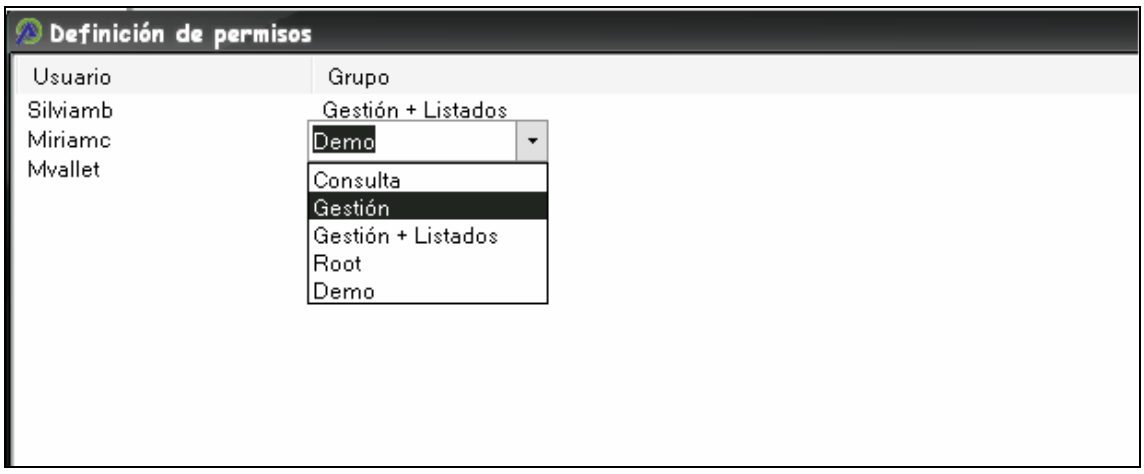


***Figura 4-53**. Ejemplo práctico de cómo cambiar el perfil a un usuario.*

Los permisos del usuario se muestran de forma incremental: en el primer listado, a la izquierda de la pantalla (ver figura 4-54), veremos los módulos a los que el usuario puede acceder según los menús modulares que éste tenga asignados. Al seleccionar un módulo se listarán en el centro de la pantalla las aplicaciones disponibles del mismo, otra vez según la definición correspondiente del menú modular asignado (ver figura 4-55). Podemos en este punto marcar y desmarcar las aplicaciones que estarán accesibles al usuario, ayudándonos si es necesario de los botones de la parte inferior del formulario.

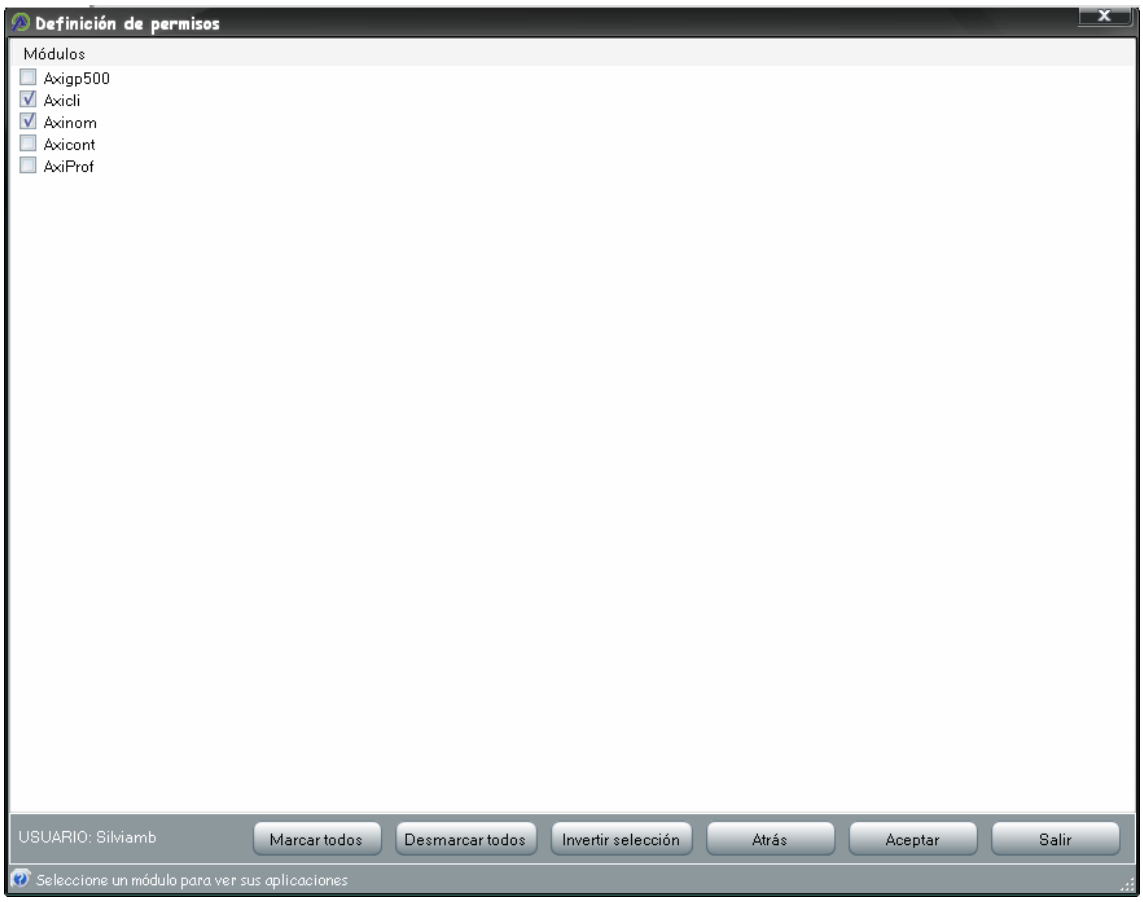


Figura 4-54. *Listado de los módulos asignados a un usuario.*

Al seleccionar un módulo o aplicación aparecerán en la parte inferior del listado correspondiente dos *Checkbox* en los que podremos marcar el tipo de acceso específico (*Lectura* o *Escritura*), que predominará sobre el perfil del usuario, sea cual sea. Además, existe un panel de ayuda en la parte inferior del formulario, que se

muestra al pasar el *mouse* por encima, y se oculta automáticamente al mover el *mouse* fuera del panel (ver figura 4-55).

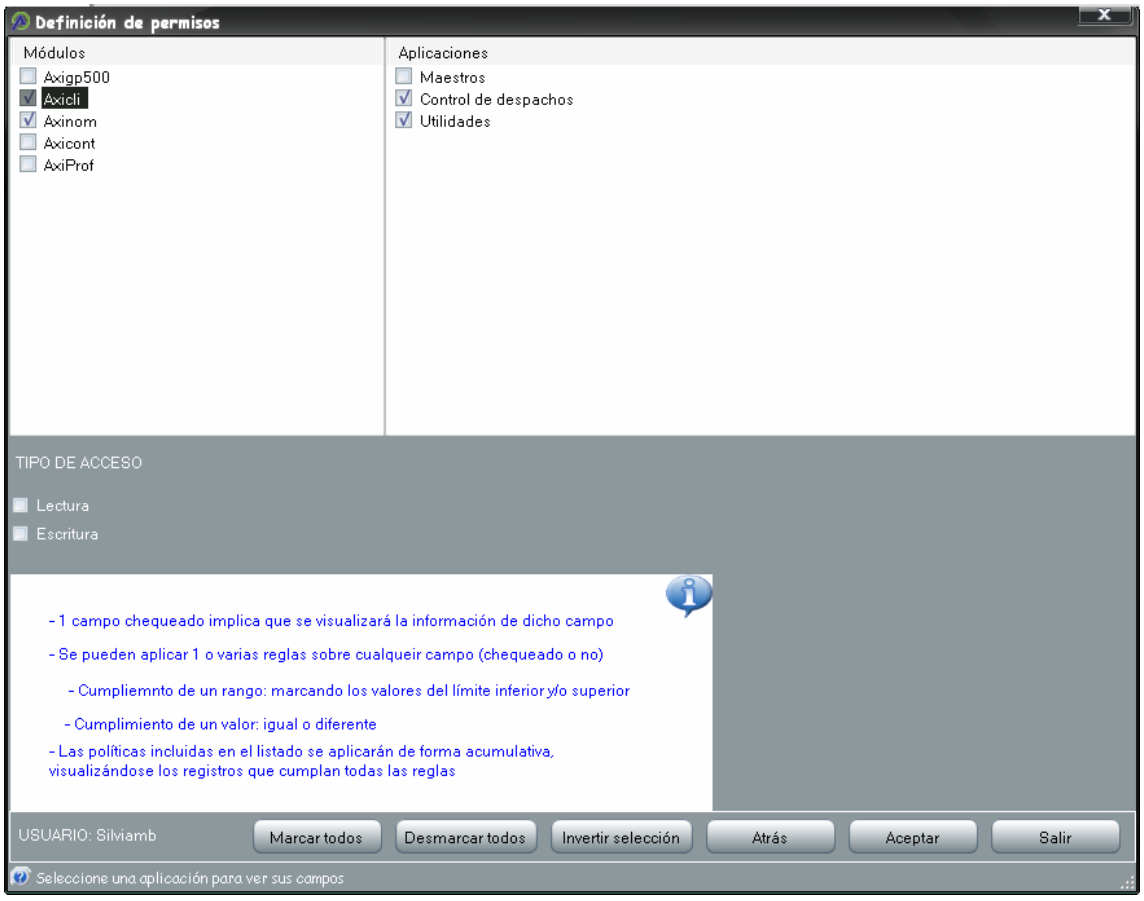


***Figura 4-55**. Listado de aplicaciones del módulo seleccionado, y ayuda despleglable.*

Al seleccionar una aplicación del listado central se mostrarán a la derecha del formulario, en un tercer listado, los campos asociados a dicha aplicación. Al desmarcar un campo éste se hará no accesible al usuario; esto significa que, pese a que el usuario podrá ver la caja de texto que representa al campo en la pantalla correspondiente a la aplicación seleccionada, no podrá consultar los valores del campo en cuestión. P.ej. en un formulario de gestión de fichas de clientes, al desautorizar el campo de 'Apellidos', el usuario podría ver los nombres y otros datos de los clientes, pero no sus apellidos.

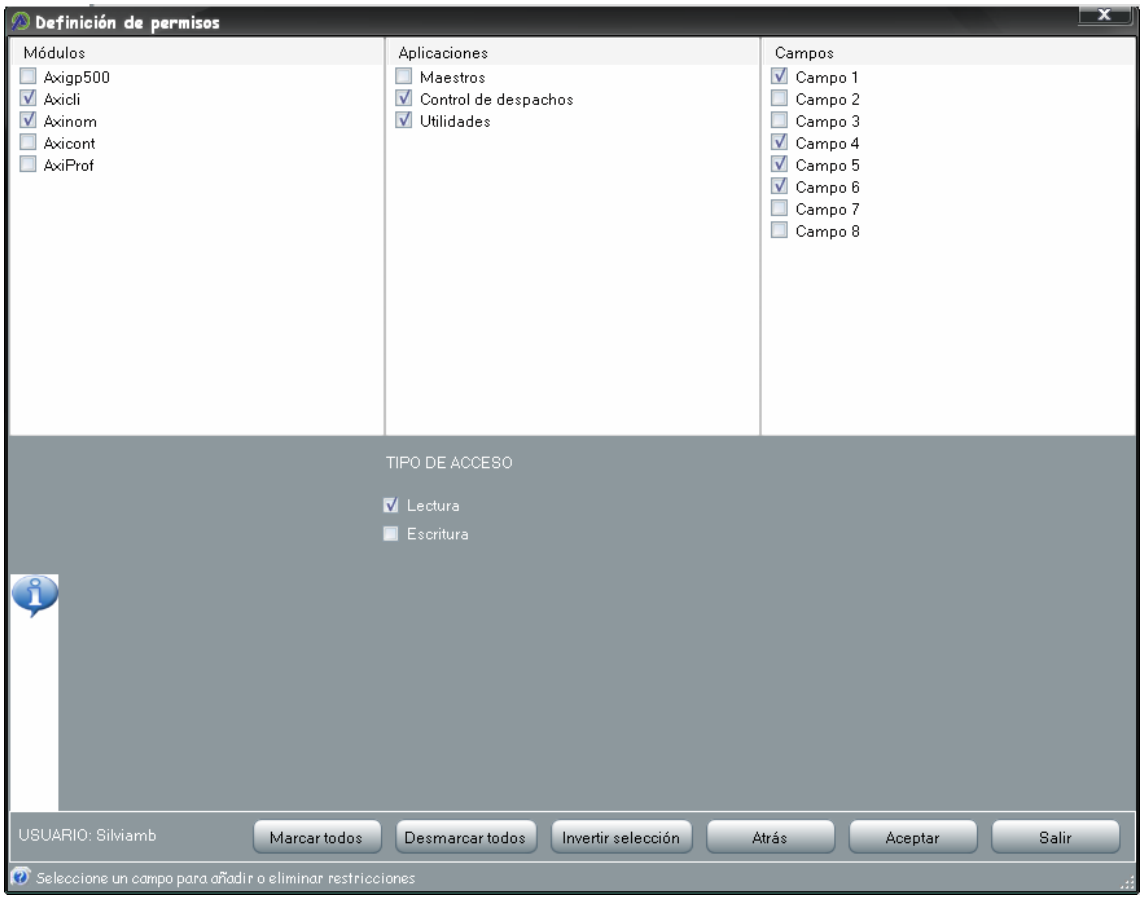


***Figura 4-56**. Al seleccionar una aplicación podremos ver listados sus campos asociados.*

Además, a los campos se les pueden aplicar restricciones, las cuales se guardan en la tabla *tblRestriccion* de la base de datos. Restringir un campo es similar a desautorizarlo: en este caso se permite al usuario acceder al campo, pero el rango de valores del mismo está acotado. Las acotaciones se aplican según el tipo de campo, y utilizando tres reglas. Los tipos de campo posibles son:

- ✓ Campo alfanumérico (string).
- ✓ Campo numérico.
- ✓ Campo de fecha.

Las tres reglas de acotación son:

- ✓ Valor entre un límite inferior y un límite superior.
- ✓ Valor “Igual a”.
- ✓ Valor “Diferente de”.

Entenderemos mejor las acotaciones con las siguientes pantallas de ejemplo.

Al seleccionar un campo aparecerá el menú de acotaciones en la parte inferior del formulario. El menú de acotaciones será uno u otro según el tipo del campo. En este primer caso el tipo del campo es alfanumérico. Suponiendo un caso como el planteado anteriormente con un formulario de gestión de fichas de clientes, si el campo seleccionado se refiere a los 'Apellidos' del cliente, podemos acotar los valores accesibles al usuario entre los apellidos 'García' y 'Martínes', a la vez que excluyendo el apellido 'Giménez'. Las restricciones se añaden al listado mediante los botones  .

En este segundo caso escogemos un campo numérico. Las acotaciones se establecen mediante controles de tipo *NumericUpDown*. Como podemos observar en el listado de restricciones, serán accesibles al usuario los valores comprendidos entre los intervalos '0-500' y '1000-2500' (límites inclusive), diferentes de '250' y '499'.

Se pueden eliminar las restricciones a los campos seleccionándolas del listado correspondiente y pulsando el botón  .

En el último caso vemos cómo se aplican restricciones a un campo de tipo fecha. Para establecer las acotaciones utilizamos un control *DateTimePicker*; al hacer *clic* sobre éste se despliega un pequeño calendario del que podemos escoger una fecha. En este ejemplo las acotaciones aplicadas permiten al usuario acceder a los registros con fechas comprendidas entre el 24 de julio de 2006 y el 7 de agosto de 2006, o iguales al 1 de enero de 2006.

- ***Crear usuario.***

Pulsando el botón *Nuevo* en la pantalla principal de la herramienta de edición de usuarios y permisos (ver figura 4-57) podemos crear un nuevo usuario del sistema. Se mostrará entonces un pequeño formulario que deberemos rellenar con la información del nuevo usuario.

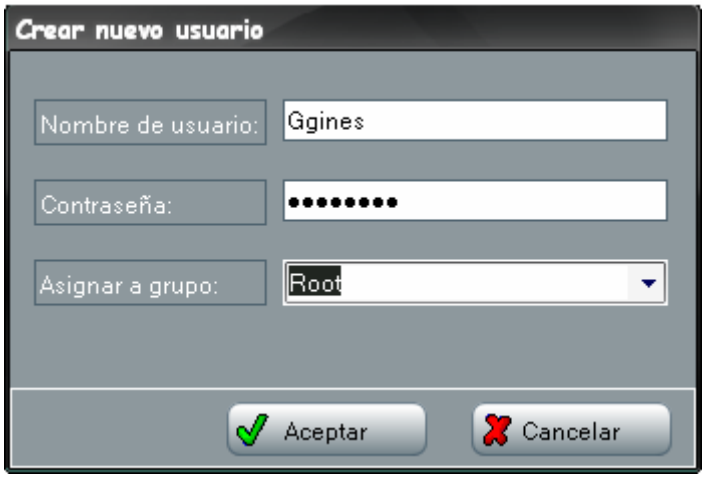


Figura 4-57*. Formulario de creación de nuevo usuario.*

Tras pulsar el botón *Aceptar* ya podemos proceder a editar los permisos específicos del usuario y/o asignarle menús modulares.

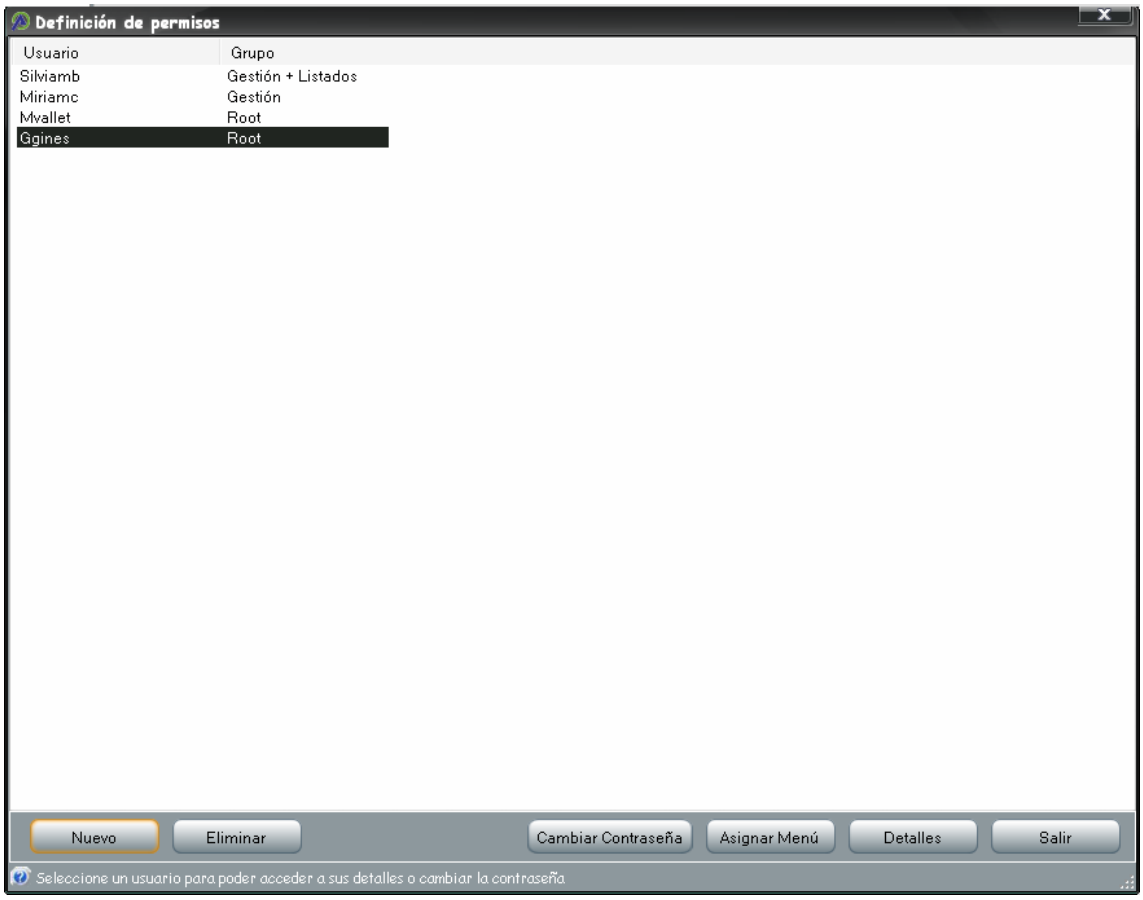


***Figura 4-58**. El usuario recién creado aparece añadido al listado de usuarios.*

- ***Asignar menús a un usuario.***

Al pulsar el botón *Asignar menú*, previa selección de un usuario, se abre un nuevo formulario mediante el cual podemos asignarle menús modulares al usuario en cuestión.

Podemos apreciar en la parte superior derecha del formulario (ver figura 4-59) un control de tipo *Listview* que contiene el listado de menús modulares que el usuario tiene ya asignados.

Para asignar un menú modular existente en la base de datos, debemos seleccionarlo previamente en el *Combobox* situado junto a la etiqueta *Menú existente*. Es posible también cargar el menú seleccionado en el control *DatagridView* de la parte central del formulario, mediante el botón *Cargar menú,* y editarlo convenientemente

antes de asignarlo al usuario. Para poder editar el menú es precio marcar primero la casilla *Editar*.

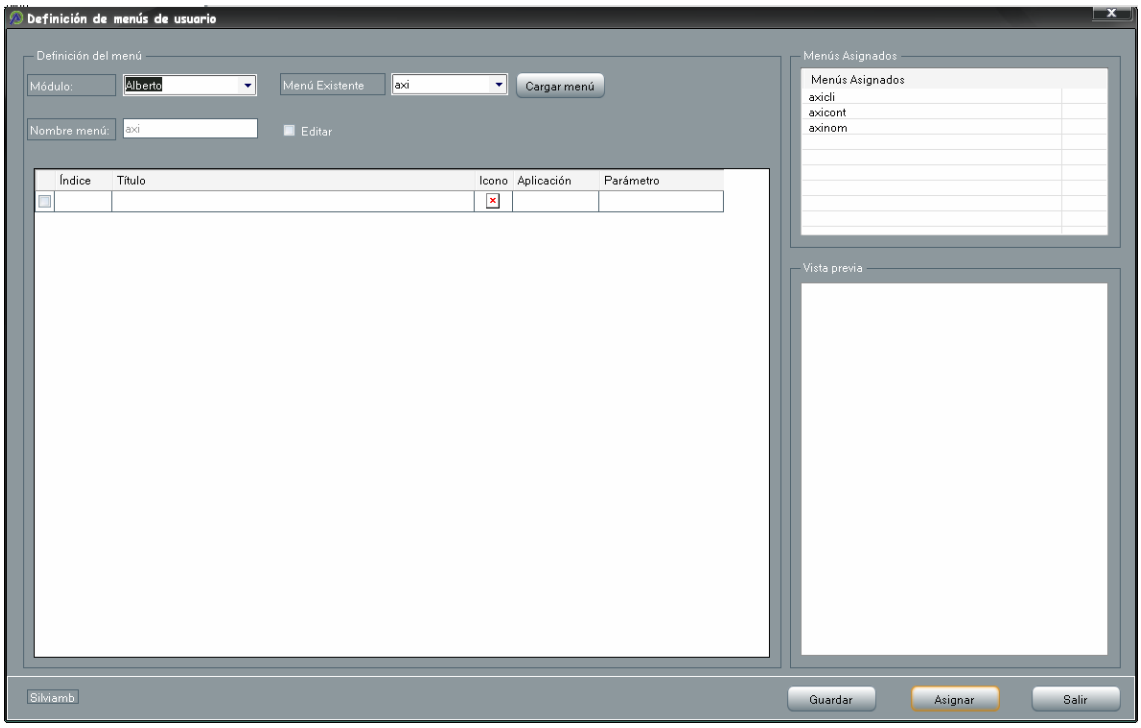


***Figura 4-59**. Formulario de asignación de menús modulares por usuario.*

Capítulo 5.
Análisis orgánico.

Capítulo 5. Análisis orgánico.

5.1. Sistemas de almacenaje.

Como se ha comentado anteriormente en varias ocasiones, el E.R.P. *Axiwin* original trabaja con un extenso y complicado sistema de ficheros. Pese a que el diseño y la implementación son dignos de elogio, el acceso a los datos es de todo menos óptimo. Cuando trabajamos con ficheros pequeños la ejecución sigue siendo fluida, pero inevitablemente los ficheros crecen y crecen hasta tamaños poco habituales y menos aconsejables en ficheros de texto, y el acceso a los datos se ralentiza mucho.

La nueva plataforma *Axiwin* funciona íntegramente sobre una base de datos, aunque se utilizan algunos tipos de ficheros también, pero en ningún caso para guardar datos de las aplicaciones del E.R.P.

5.1.1. Base de datos.

Teniendo en cuenta que la nueva plataforma *Axiwin* es totalmente dinámica, entendiendo por dinamismo la posibilidad de cargarla con distintas propiedades cada vez, según indiquen los valores establecidos en múltiples parámetros de configuración, el volumen de información a manejar es muy grande. Para explicar mejor el diseño de la base de datos, su estructura, vamos a dividirla y clasificar sus tablas según su finalidad.

5.1.1.1. Tablas asociadas a la carga de pantalla de aplicaciones.

La carga dinámica de pantallas de aplicaciones requiere de muchas tablas, relacionadas entre sí, en las que guardamos toda la información asociada a las etiquetas, las cajas de texto, las relaciones con otras aplicaciones, etc. En el diseño original de la base de datos todas las aplicaciones de todos los módulos compartían las mismas tablas, y tras las primeras migraciones exhaustivas comprobamos que las tablas se hacían demasiado grandes. Hicimos un nuevo diseño de la base de datos, distribuyendo los datos en distintas tablas por módulos, a la vez que de esta forma

diferenciábamos las tablas relacionadas con las carga de aplicaciones de las tablas de configuración del sistema. Para llevar a cabo esta tarea se implementó un proyecto de adaptación de la base de datos que debe ejecutarse, preferentemente, tras realizar la migración por primera vez. El siguiente diagrama representa el conjunto de tablas asociadas a la carga de pantallas de aplicaciones, antes de realizar la adaptación.

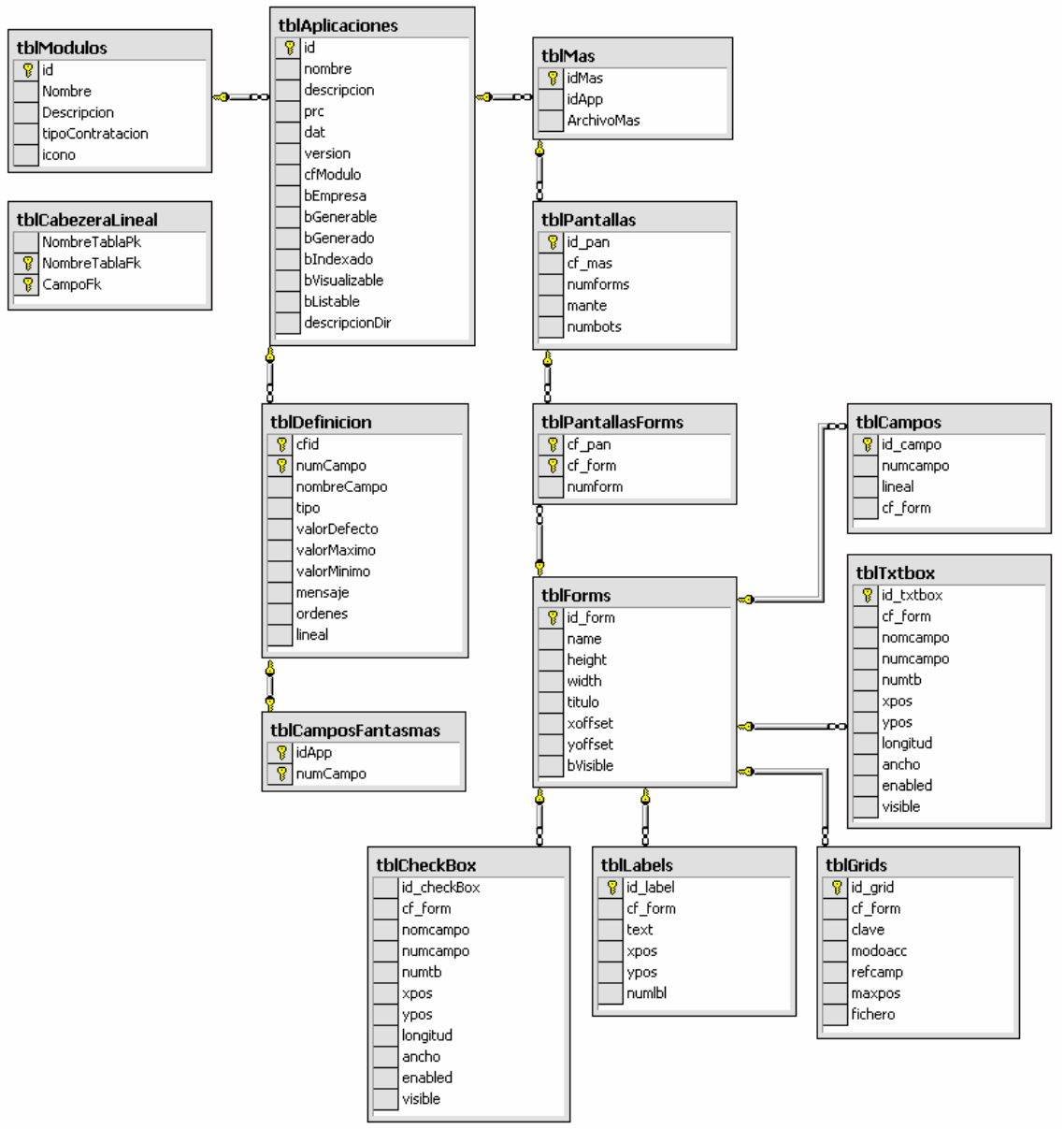


***Figura 5-1**. Diseño orgánico original de las tablas de la base de datos asociadas a la carga de pantallas de aplicaciones.*

En el proceso de adaptación los datos de distribuyen por módulos en tablas cuyos nombres empiezan por "*tbl"* seguido del nombre del módulo (p.ej: "*tblaxinom_Labels*" o "*tblaxicli_Textbox*"). En el proceso de adaptación se crean, además, tablas *extras* que no se contemplaron inicialmente en el diseño orgánico inicial de la base de datos.

Tras la adaptación la base de datos mantiene la estructura original de las tablas módulo a módulo, pero muestra este otro aspecto:

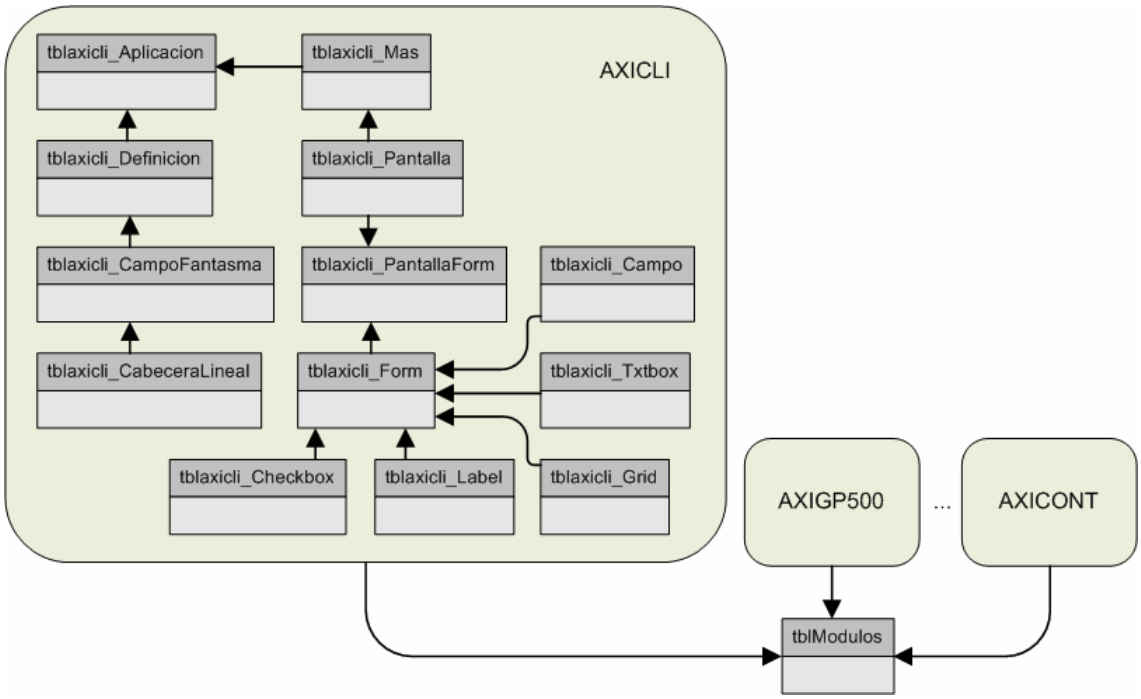


Figura 5-2*. Diseño de las tablas asociadas a la migración de pantallas después de la adaptación.*

Las tablas que intervienen en la carga de pantallas son, principalmente:

- ✓ *tblModulo_Label*. Guardamos la posición de la etiqueta en pantalla (X, Y), el contenido, el estilo específico, y el identificador del formulario al que pertenece.
- ✓ *tblModulo_Textbox*. Guardamos la posición de la caja de texto en pantalla (X, Y), el nombre y número del campo al que está asociado el *textbox* en la aplicación a la que pertenece, la longitud en caracteres, el ancho, el estilo específico, y si está habilitado y/o visible.

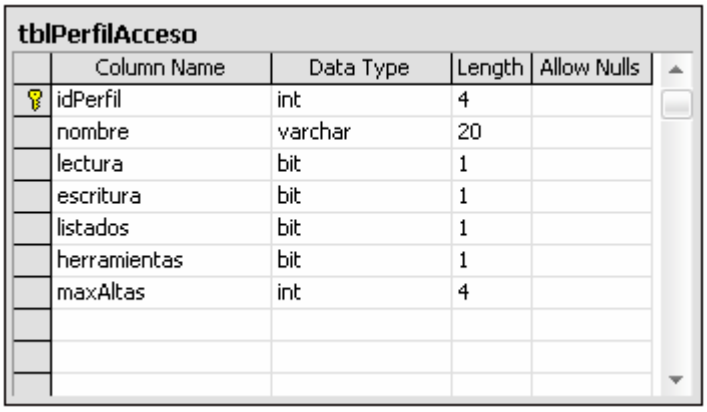
- ✓ *tblModulo_Campo*. En esta tabla guardamos el número de campo (repetido en la tabla *tblModulo_Textbox*) y un campo booleano que nos indica si se trata de un campo lineal o no.
- ✓ *tblModulo_CampoFantasma.* Guarda números de campos asociados a identificadores de aplicaciones. Los "campos fantasma" son cajas de texto que el usuario puede ver pero en las que no puede escribir (algo así como *textbox* inhabilitados).
- ✓ *tblModulo_Pantalla.* Tiene una correspondencia 1 : 1 con las tablas de aplicaciones y de ficheros de definición de campos (.mas) pertinentes. Guardamos el número de formularios (pantallas) de cada aplicación, así como un campo booleano que nos dice si la aplicación lleva una botonera de mantenimiento o no.
- ✓ *tblModulo_PantallaForm.* Esta tabla tiene una correspondencia 1 : N con *tblModulo_Pantallas*, y ejerce de puente con la tabla *tblModulo_Forms*. Cada aplicación tendrá en esta tabla tantas entradas relacionadas como formularios.
- ✓ *tblModulo_Form.* Guardamos el ancho y alto del formulario, el estilo específico, el título del formulario (equivale a la descripción de la aplicación en la tabla *tblModulo_Aplicacion*).

5.1.1.2. Tablas de gestión de permisos y usuarios.

Utilizamos un amplio conjunto de tablas para la gestión de los permisos de acceso a la plataforma de cliente *Axiwin*. No todas estas tablas están necesariamente relacionadas entre ellas, pero sí dedicadas, de un modo u otro, a validar a un usuario en el sistema, y cargar su configuración específica en el proceso.

- ✓ *tblPerfilAcceso*.

  En los sistemas operativos *UNIX* los permisos de un usuario se determinan según los grupos a los éste pertenece. En el caso de la plataforma *Axiwin* se utilizan *perfiles de acceso*, que funcionan de forma parecida a los grupos de *UNIX* con la diferencia principal de que un usuario sólo puede pertenecer a un perfil de acceso.



tblPerfilAcceso

	Column Name	Data Type	Length	Allow Nulls
🔑	idPerfil	int	4	
	nombre	varchar	20	
	lectura	bit	1	
	escritura	bit	1	
	listados	bit	1	
	herramientas	bit	1	
	maxAltas	int	4	

***Figura 5-3**. Diseño de la tabla tblPerfilAcceso.*

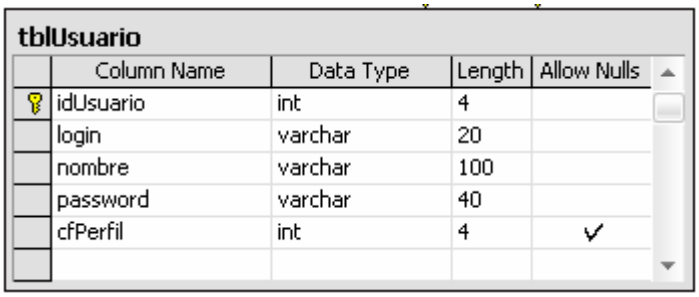
Como hemos visto en el *Capítulo 4. Análisis funcional* los perfiles se definen mediante la combinación de 5 parámetros:

- *lectura*: autoriza al usuario a consultar los datos de las aplicaciones del E.R.P. *Axiwin*. Tipo de dato: booleano.
- *escritura*: autoriza al usuario a realizar altas, bajas y modificaciones en las aplicaciones del E.R.P. *Axiwin*. Tipo de dato: booleano.
- *listados*: autoriza al usuario a definir y obtener listados. Tipo de dato: booleano.
- *herramientas*: autoriza al usuario a utilizar las herramientas del E.R.P. *Axiwin*. Tipo de dato: booleano.
- *maxAltas*: indica el número máximo de registros que el usuario puede dar de alta en cada aplicación del E.R.P. *Axiwin*. Tipo de dato: entero.

Los perfiles se distinguen por un nombre y un identificador autonumérico de la tabla.

✓ *tblUsuario*.

En esta tabla se guarda la información de los usuarios del sistema, sin entrar en detalles personales. Cada usuario queda definido por un *nombre identificador de usuario* (comúnmente llamado *login*), su nombre verdadero (en caso de tratarse de una persona, y no de un usuario genérico), un *password* o contraseña y el identificador del perfil al que pertenece (clave foránea de la tabla *tblPerfilAcceso*). El *password* deberá estar encriptado, pero esta tarea, junto con otras cuestiones de seguridad, no está incluida en este proyecto.



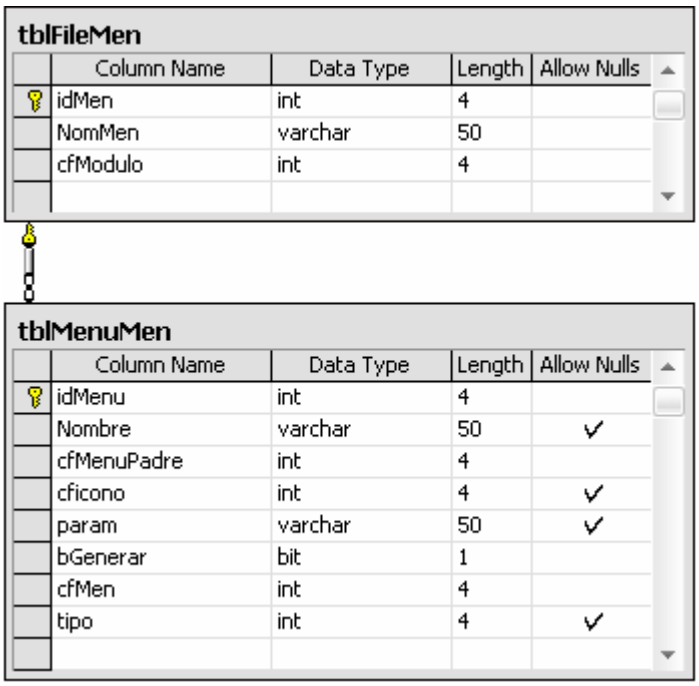
tblUsuario

	Column Name	Data Type	Length	Allow Nulls
🔑	idUsuario	int	4	
	login	varchar	20	
	nombre	varchar	100	
	password	varchar	40	
	cfPerfil	int	4	✓

***Figura 5-4**. Diseño de la tabla tblUsuario.*

- ✓ *tblFileMen* y *tblMenuMen*.

  Entre estas dos tablas se guarda la definición de los menús modulares de la plataforma de cliente *Axiwin*. Ambas tablas se generan durante el proceso de migración de los ficheros de menús .men, y se utilizan posteriormente para guardar la definición de los nuevos menús creados con el editor de menús de la plataforma de administrador (ver *Capítulo 4. Análisis funcional*).



tblFileMen

	Column Name	Data Type	Length	Allow Nulls
🔑	idMen	int	4	
	NomMen	varchar	50	
	cfModulo	int	4	

tblMenuMen

	Column Name	Data Type	Length	Allow Nulls
🔑	idMenu	int	4	
	Nombre	varchar	50	✓
	cfMenuPadre	int	4	
	cficono	int	4	✓
	param	varchar	50	✓
	bGenerar	bit	1	
	cfMen	int	4	
	tipo	int	4	✓

***Figura 5-5**. Diseño de las tablas tblFileMen y tblMenuMen. Tienen una relación 1 : N.*

En la tabla *tblMenuMen* cada registro se corresponde con una entrada de menú, de uno de los menús definidos en la tabla *tblFileMen*. Nos fijaremos en los campos *idMenu* y *cfMenuPadre*, que establecen de forma sencilla la relación entre las entradas de menú que definen los distintos niveles del mismo.

- *idMenu*: número identificador de la entrada de menú. Es autonumérico.
- *cfMenuPadre*: este campo establece una relación lógica con otro registro de la misma tabla. Indica el número identificador de la entrada de menú "padre" o raíz de la que cuelga la presente entrada de menú.

Mediante una función recursiva bastante simple podemos trasladar la definición de un menú en la tabla *tblMenuMen* a un control de tipo *Treeview* (árbol explorador), partiendo simplemente del identificador del nodo o entrada raíz.

✓ *tblAcceso* y *tblRestriccion*.

Como hemos visto en el *Capítulo 4. Análisis funcional*, los permisos de acceso por usuario se aplican en distintas capas, cada una más restrictiva que la anterior:

- Menús modulares asignados al usuario.
- Permisos sobre aplicaciones (módulo a módulo).
- **Permisos sobre campos y restricciones (aplicación a aplicación).**
- Perfil de acceso.

Para implementar los permisos sobre y restricciones se utilizan las tablas *tblAcceso* y *tblRestriccion*. Cada registro de la tabla *tblAcceso* está asociado a un usuario del sistema mediante el identificador autonumérico de la tabla *tblUsuario.* Los otros campos son:

- *idAcceso*: identificador de la norma de acceso.
- *cfModulo*: identificador de uno de los módulos asignados al usuario.
- *cfAplicacion*: identificador de una de las aplicaciones del módulo especificado por el campo anterior.

Cada uno de los registros de la tabla *tblAcceso* equivale a autorizar al usuario a acceder a la aplicación especificada por el campo *cfAplicacion*.

Los *accesos* se pueden complementar con *restricciones*, las cuales están definidas en la tabla *tblRestriccion*. Las restricciones se aplican a campos

de aplicaciones, motivo por el cual la tabla *tblRestriccion* y la tabla *tblAcceso* están relacionadas por el campo *idAcceso*. Las restricciones pueden ser de tres tipos:

- Restricción de *acotación*: limita el rango de valores accesibles.
- Restricción de *exclusión*: valores accesibles deben ser *diferentes de* un valor en concreto.
- Restricción de *obligación*: valores accesibles deben ser *iguales a* un valor en concreto.

Asimismo hay tres tipos de campos en las aplicaciones del E.R.P. *Axiwin:*

- Alfanumérico.
- Numérico.
- Fecha.

Según el tipo de campo la restricción se aplicará de forma distinta, pero en la base de datos todas las restricciones se guardan en forma de cadena de texto, que debe tratarse por código posteriormente. Veamos los campos de la tabla *tblRestriccion*:

- *cfCampo*: identificador del campo dentro de la aplicación, nos servirá para saber de qué tipo de campo se trata.
- *politica*: nombre del tipo de restricción (acotación-diferente-igual). Tipo de dato: cadena.
- *limiteInf*: límite inferior en caso de tratarse de una restricción de acotación, o valor concreto en caso de tratarse de una restricción de exclusión. Tipo de dato: cadena.
- *limiteSup*: límite superior en caso de tratarse de una restricción de acotación, o valor concreto en caso de tratarse de una restricción de obligación. Tipo de dato: cadena.

Cada campo de una aplicación puede tener más de una restricción, o ninguna. En cualquier caso es necesario que exista una entrada en la tabla *tblAcceso* para la aplicación a cuyos campos queramos aplicar restricciones.

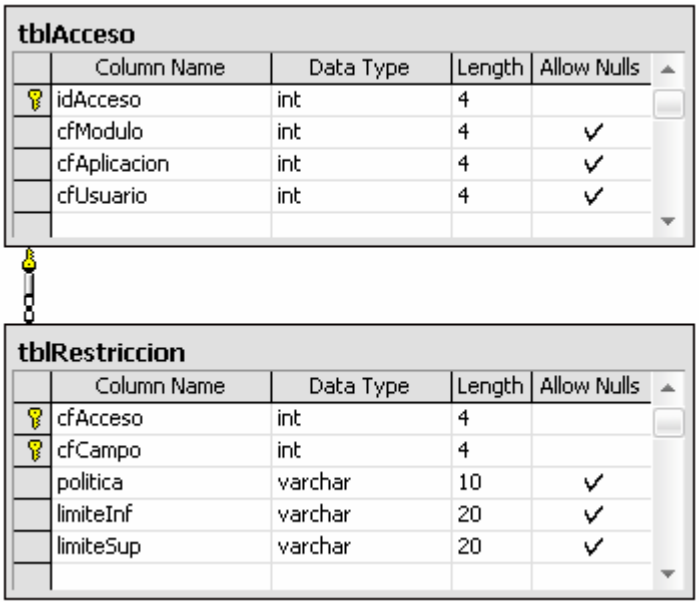


***Figura 5-6**. Diseño de las tablas tblAcceso y tblRestriccion. Tienen una relación 1 : N.*

5.1.1.5. Tablas de configuración de las propiedades de la/s plataforma/s.

- ✓ *tblBotonera*, *tblAccionBotonera* y *tblBotonBotonera*.

  La botonera del formulario principal de la plataforma de cliente es un control 100% personalizable y propio de cada usuario. Inicialmente existe una única botonera por defecto, que es la que cada usuario verá la primera vez que se autentique en el sistema. En la tabla *tblBotonera* existe un registro por cada usuario definido en la tabla *tblUsuario*.

  - *bDefault*: indica si se trata de la botonera por defecto o no, es decir, si el usuario la ha modificado o no. Tipo de dato: booleano.
  - *cfUsuario*: identificador del usuario a quien pertenece la botonesa.

  En la tabla *tblAccionBotonera* se encuentran predefinidas las acciones que el usuario puede añadir en la botonera como accesos rápidos. Entendemos por *acción* el acceso a una herramienta o aplicación que no pertenezca a ninguno de los módulos de *Axiwin*.

  - *tag*: se refiere a la entrada del fichero de idioma que contiene el nombre de la acción.

- *Icono*: nombre del icono que se deberá mostrar en la botonera. El icono se carga del archivo de recursos de imágenes.

Los botones de los que consta una botonera se definen, finalmente, gracias a la tabla *tblBotonBotonera.* Esta tabla asocia un identificador de botonera (*cfBotonera*) con múltiples identificadores de acciones (*cfAccBot*), que serán los botones. Los otros campos son:

- *orden*: asigna al botón una posición dentro de la botonera. Tipo de dato: entero.
- *bModulo*: indica si el presente botón/acción se refiere a una herramienta de la plataforma de cliente o a un módulo de aplicaciones-

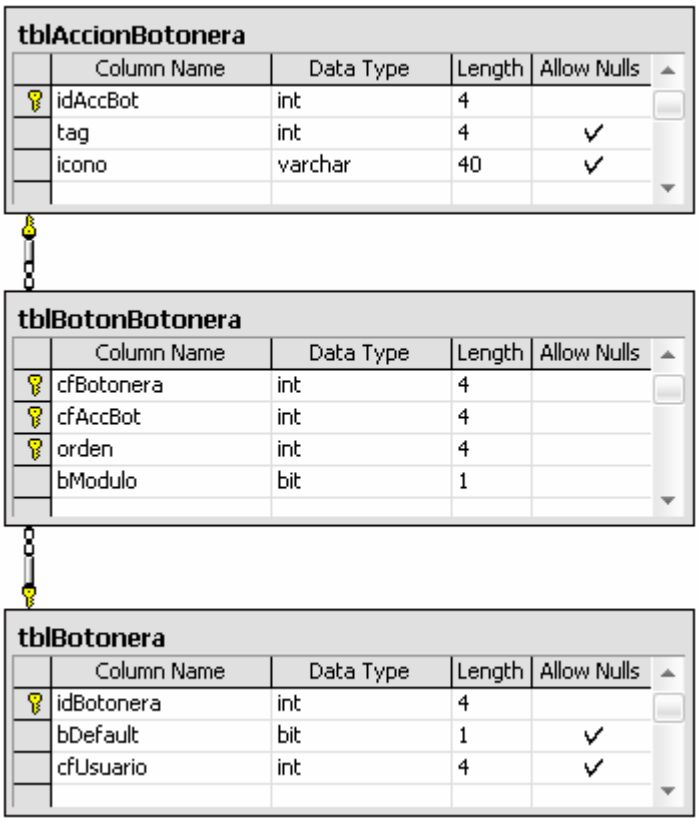


***Figura 5-7**. Diseños de las tablas tblBotonera, tblAccionBotonera y tblBotonBotonera.*

✓ *tblIcono*.

Esta tabla contiene los nombres de todas las imágenes/iconos que se utilizan en los formularios de las plataformas de administrador y cliente. Se puede

considerar como el listado de las imágenes/iconos que **no deberían faltar** en el archivo de recursos de imágenes para visualizar correctamente todos los formularios.

tblIcono

	Column Name	Data Type	Length	Allow Nulls
🔑	idIcono	int	4	
	Nombre	varchar	50	

***Figura 5-8**. Diseño de la tabla tblIcono.*

- ✓ *tblImgResources*.

  Guardamos en esta tabla los nombres de los archivos de recursos de imágenes existentes en el sistema. Deberán haber sido creados con el *Generador de recursos de imágenes* para que la plataforma los detecte y puedan ser utilizados. Actualmente el campo *defecto* no se usa para nada, ya que los paquetes de imágenes se asocian directamente a los estilos. Anteriormente se marcaba uno de los recursos de imágenes como activo, y era éste el que la plataforma cargaba al ejecutarse.

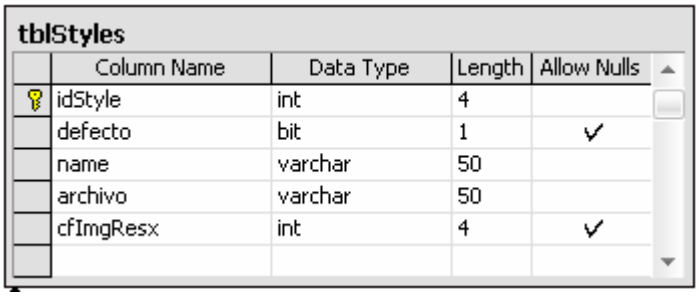
tblImgResources

	Column Name	Data Type	Length	Allow Nulls
🔑	idImgResx	int	4	
	defecto	bit	1	✓
	name	varchar	20	

***Figura 5-9**. Diseño de la tabla tblImgResources.*

- ✓ *tblStyles*.

  Contiene información acerca de las *hojas de estilos* existentes en el sistema. Las *hojas de estilos* se guardan físicamente (son archivos XML) en un directorio propio de la plataforma *Axiwin*, pero para que la plataforma los detecte deben estar definidos en la base de datos.



tblStyles

	Column Name	Data Type	Length	Allow Nulls
🔑	idStyle	int	4	
	defecto	bit	1	✓
	name	varchar	50	
	archivo	varchar	50	
	cfImgResx	int	4	✓

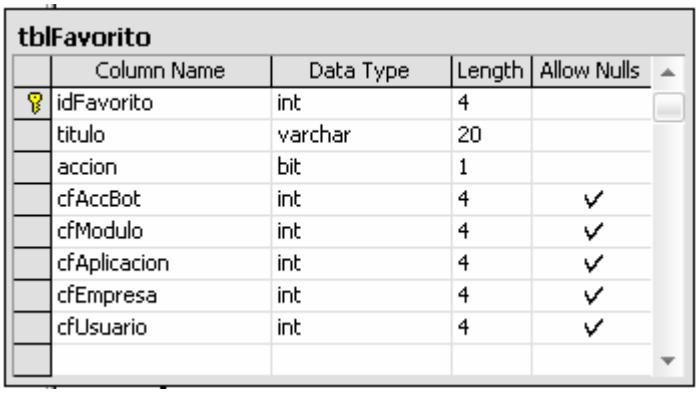
***Figura 5-10**. Diseño de la tabla tblStyles.*

- *defecto*: se utiliza para indicar cuál es el estilo activo, es decir, el que se deberá cargar al ejecutar la plataforma (aunque cada usuario puede escoger un estilo propio). Tipo de dato: booleano.
- *cfImgResx:* identificador del archivo de recursos de imágenes asociado al estilo (clave foránea de la tabla *tblImgResources*).

✓ *tblFavorito*.

El funcionamiento de los *favoritos por usuario* está implementado igual que el de la botonera personalizable.

- *titulo*: nombre/descripción del favorito.
- *accion*: indica si se trata o no de una acción (si no es una acción será un acceso a un módulo o una aplicación).
- *cfAccBot*: identificador de la acción en la tabla *tblAccionBotonera* (si el favorito es una *acción*).
- *cfModulo*: identificador del módulo en la tabla *tblModulos* (si el favorito es un acceso a un módulo).
- *cfAplicacion*: identificador de la aplicación en la correspondiente tabla modular de aplicaciones (si el favorito es un acceso a una aplicación),
- *cfEmpresa*: identificador de la empresa, si se puede aplicar al módulo o aplicación al que hace referencia el favorito.
- *cfUsuario*: identificador del usuario al que pertenece el favorito.



tblFavorito

	Column Name	Data Type	Length	Allow Nulls
🔑	idFavorito	int	4	
	titulo	varchar	20	
	accion	bit	1	
	cfAccBot	int	4	✓
	cfModulo	int	4	✓
	cfAplicacion	int	4	✓
	cfEmpresa	int	4	✓
	cfUsuario	int	4	✓

***Figura 5-11**. Diseño de la tabla tblFavorito.*

5.1.2. Ficheros.

5.1.2.1. Hojas de estilos.

Las *hojas de estilos* definen combinaciones de colores y otras propiedades de un conjunto preestablecido de controles de las plataformas de administrador y cliente. Estos ficheros se generan con el *Generador de estilos* y son de tipo .xml (cada uno con su correspondiente fichero de *schema* .xsd), así que tienen estructura de tabla de datos. Cada control editable incluido en las *hojas de estilos* está representado por su propia tabla. En la siguiente tabla se muestra el conjunto de controles que constan en la *hoja de estilos*, junto con el listado de propiedades editables de cada control. Las propiedades marcadas en color azul fueron añadidas a los controles, ya que por defecto no existen en Visual Studio .NET 2005.

Control	Propiedades editables
Form	✓ BackColor ✓ BackgroundImage ✓ BackgroundImageLayout ✓ Icon ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Panel	✓ BackColor ✓ BorderColor ✓ BorderWidth
Button	✓ BackgroundImage ✓ BackgroundImageActive ✓ BackgroundImageInactive

	✓ BackgroundImageMouseDown ✓ BackgroundImageMouseOver ✓ BackgroundImageLayout ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Textbox	✓ BackColor ✓ BackColorInactive ✓ BorderColor ✓ BorderWidth ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Label	✓ BackColor ✓ BorderColor ✓ BorderWidth ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Listview	✓ BackColor ✓ BackgroundImage ✓ BackgroundImageTiled ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Combobox	✓ BackColor ✓ BackColorInactive ✓ BorderColor ✓ BorderWidth ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Groupbox	✓ BackColor ✓ BorderColor ✓ BorderWidth ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Menu	✓ BackColor ✓ ForeColor ✓ FontName ✓ FontSize

	✓ FontStyle
Toolbar / Toolstrip	✓ BackColor ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
TabControl	✓ BackColor ✓ BackgroundImage ✓ BackgroundImageLayout ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Treeview	✓ BackColor ✓ LineColor ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle
Datagrid	✓ BackColor ✓ AlternatingBackColor ✓ GridColor ✓ CaptionBackColor ✓ CaptionForeColor ✓ ForeColor ✓ FontName ✓ FontSize ✓ FontStyle

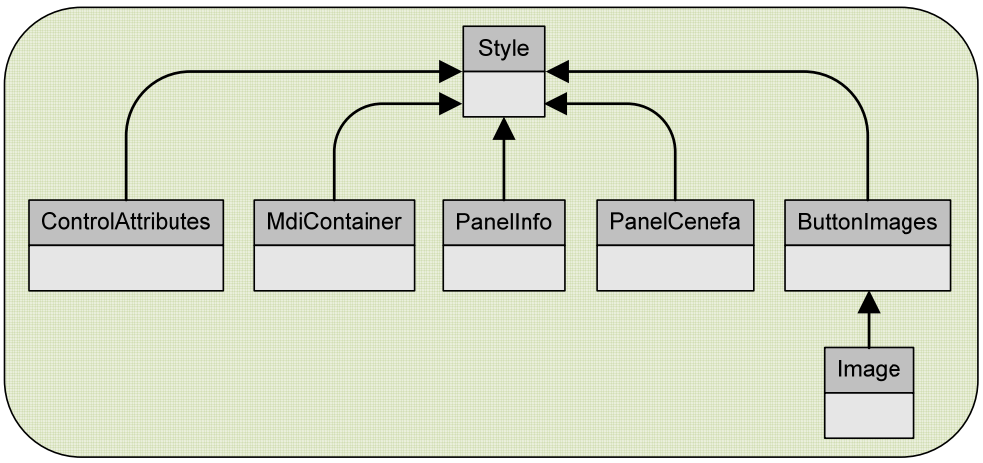


***Figura 5-12**. Diagrama que representa la estructura de una hoja de estilos. Los controles se representan mediante tablas que pertenecen a la tabla ControlAttributes.*

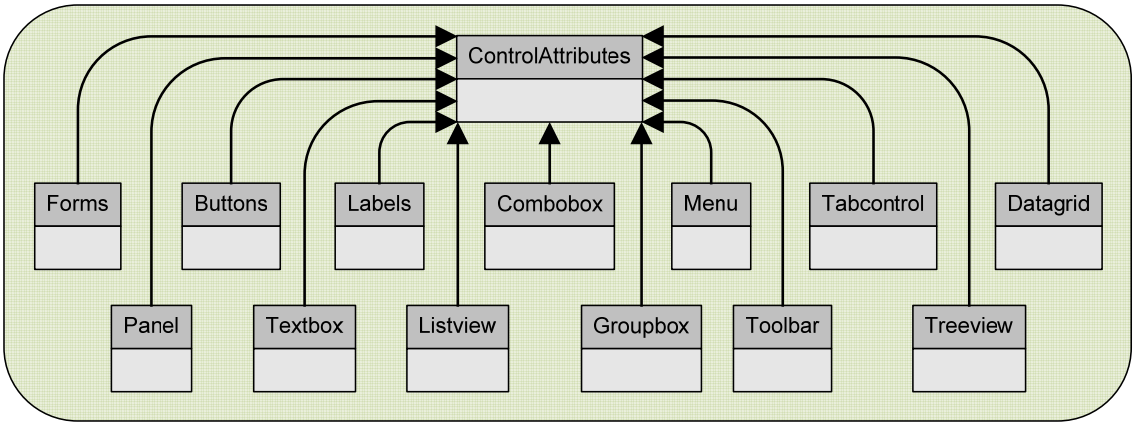


***Figura 5-13**. Diagrama de tablas de la tabla ControlAttributes de una hoja de estilos.*

5.1.2.2. Recursos de idiomas.

Son ficheros de recursos .resx, cuya estructura es también parecida a una tabla de datos. En estos ficheros se definen todos los textos que aparecen en los distintos controles de la plataforma de cliente, tales como etiquetas, botones, menús, mensajes de los *messagebox*, etc., y a cada texto se le asigna un identificador numérico. El fichero .resx de idioma debe ser compilado para generar su correspondiente fichero .resources encriptado, que es el que usa la plataforma de cliente para cargar los textos en los controles según el idioma activo en el momento de la ejecución. Es necesario asignar a la propiedad *Tag* de cada control el identificador numérico del texto que le corresponda según el fichero .resx, ya que de otro modo el texto de dicho control no se cargará.

La utilización de ficheros de recursos .resx es el método más común para implementar la carga de distintos idiomas en una aplicación. Por otra parte es muy pesado trabajar con esta metodología porque hay que poner los correspondientes *Tag's* a los controles en tiempo de diseño, y definir el texto asociado al *Tag* en el fichero .resx.

5.1.2.3. Recursos de imágenes.

Una de las grandes mejoras aportadas por la nueva plataforma *Axiwin* respecto a la original es la inclusión de gran cantidad de imágenes en buena parte de los formularios. Las imágenes son en su mayor parte iconos, proporcionados por el cliente

y diseñados específicamente para el E.R.P. *Axiwin*, que se aplican, y diferencian, a las distintas aplicaciones y herramientas de las plataformas de administrador y cliente ya que, como hemos visto en el *Capítulo 4. Análisis funcional*, los formularios principales de ambas plataformas se asemejan a una ventana de Windows.

También encontramos imágenes como papeles tapiz de algunos formularios, acompañando al texto en algunos botones, etc. En las primeras fases de desarrollo de la nueva plataforma *Axiwin* las imágenes se manejaban fácilmente ya que no eran muchas. Las imágenes se guardaban todas juntas, dispersas en un mismo directorio hasta que su número creció de forma importante y éstas se distribuyeron en distintos directorios que también utilizamos para clasificar las imágenes según su finalidad. Esto conllevó a la larga tener imágenes repetidas entre los diferentes directorios, y el manejo de las mismas en tiempo de diseño se convirtió en un completo caos. Además, el hecho de tener las imágenes en directorios es poco seguro ya que son fácilmente accesibles y cualquiera podría borrarlas, modificarlas o corromperlas.

Tras decidir que había que encontrar una solución al problema de la acumulación de imágenes se plantearon diversas soluciones y finalmente se optó por los ficheros de recursos, pero descubrimos que no podíamos crear ficheros .resx que contuvieran imágenes, ya que estos sólo permiten añadir cadenas de caracteres. Podíamos en cambio crear ficheros .resources (que es lo que se obtiene de “compilar” un fichero de recursos .resx), aunque no sabíamos cómo. Encontramos ayuda en Internet sobre cómo crear ficheros de recursos fácilmente y entonces se implementó el *Generador de recursos de imágenes.*

El fichero de recursos es muy sencillo de usar; se crea utilizando un *ResourceWriter* y se lee utilizando un *ResourceReader* o un *ResourceManager* (éste segundo en nuestro caso). Al guardar objetos en el fichero de recursos asignamos a cada uno un nombre distintivo con el que recuperaremos dicho objeto posteriormente.

Dado que el fichero de recursos está encriptado es imposible acceder a su contenido si no es utilizando el *ResourceReader* o el *ResourceManager*, de modo que una vez creado es como una "caja negra", no podemos saber lo que contiene. Por este motivo no nos es posible mostrar un esquema de la estructura de un fichero de recursos.

5.2. Estructura de clases.

La plataforma *Axiwin* es una única solución de Visual Studio .NET, formada por un total de 48 proyectos distintos, algunos de los cuales se han desarrollado como parte de este Trabajo Final de Carrera. A continuación los veremos con más detalle.

5.2.1. Traductor/cargador de pantallas.

Proyecto que recoge el conjunto de clases y formularios en los que están definidos los métodos destinados a realizar la traducción de los ficheros de pantallas, y la posterior carga dinámica de dichas pantallas de la base de datos.

- ✓ *frmTraductor*. Formulario en el que está definido el procedimiento principal del proceso de migración de pantallas, *TraductorMain()*.
- ✓ *frmContenedor*. Es un formulario de tipo *MdiContainer* que se utiliza como base para cargar las pantallas de aplicaciones de la base de datos. Tiene implementados los eventos de los controles comunes a todas las aplicaciones.
- ✓ *cLabel*s. Definición de los métodos que realizan la traducción de las etiquetas e introducen dicha información en la base de datos.
- ✓ *cTxtbox*. Definición de los métodos que realizan la traducción de las cajas de texto e introducen dicha información en la base de datos.
- ✓ *cCheckBox*. Definición de los métodos que realizan la traducción de los *checkbox* e introducen dicha información en la base de datos.
- ✓ *cButtons*. Definición de los métodos que realizan la traducción de los botones e introducen dicha información en la base de datos.
- ✓ *cGrids*. Definición de los métodos que realizan la traducción de los *datagrids* o *lineales* e introducen dicha información en la base de datos.
- ✓ *cLoader*. Esta clase tiene suma importancia en el marco global de la nueva plataforma *Axiwin*, ya que contiene la definición de los métodos encargados de cargar las pantallas de las aplicaciones a partir de las distintas tablas de la base de datos que recogen la distribución de las etiquetas, las cajas de texto y demás.

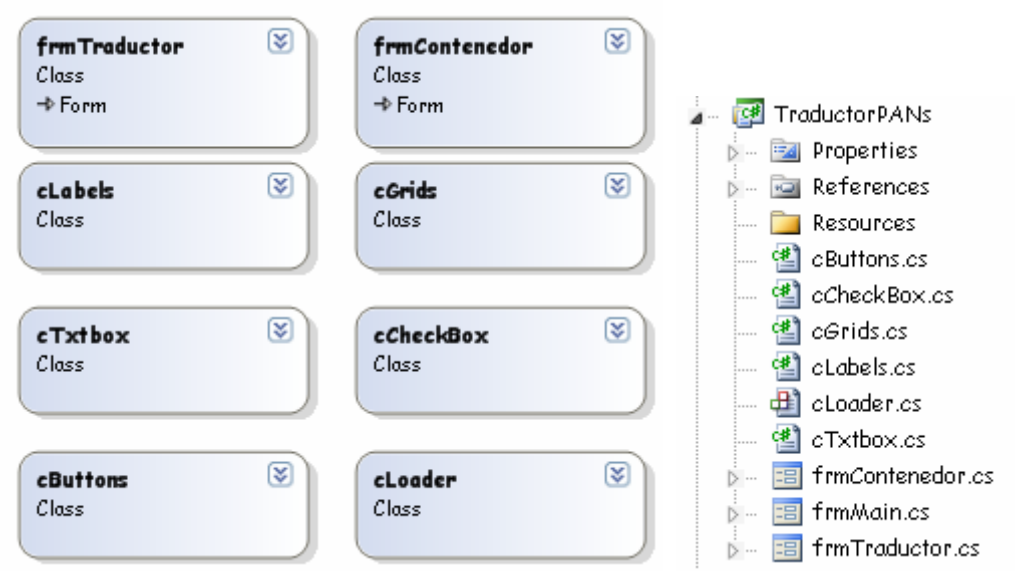


***Figura 5-14**. Conjunto de clases y formularios que componen el proyecto TraductorPANs.*

El proyecto de adaptación de la base de datos implementa el conjunto de métodos destinados a modificar, de un modo u otro, las tablas creadas en el proceso de migración, adecuándolas al diseño orgánico de la nueva plataforma.

- ✓ *frmAdaptador*. Formulario en el que está definido el procedimiento principal del proceso de adaptación de la base de datos.
- ✓ *clsAdaptador*. Definición de los métodos que realizan la adaptación específica de cada tabla de la base de datos.
- ✓ *clsOperaciones*. Clase que tiene implementadas numerosas funciones genéricas que realizan operaciones comunes contra la base de datos, así como otras menos genéricas, implementadas específicamente para el proceso de adaptación.

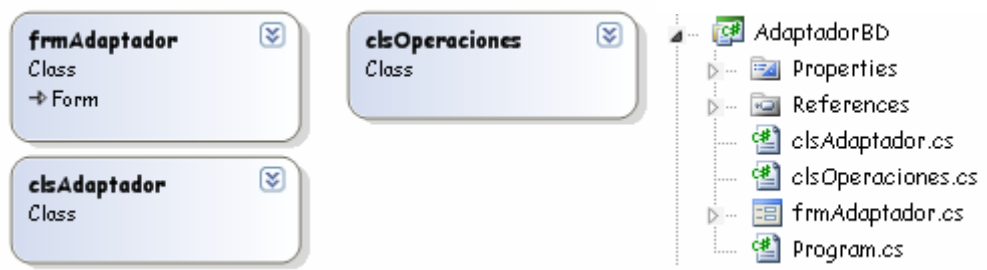


***Figura 5-15**. Conjunto de clases y formularios que componen el proyecto AdaptadorBD.*

5.2.2. Cargador de estilos.

En las clases de este proyecto están definidas las funciones encargadas de cargar el estilo de los formularios de las plataformas de administrador y cliente, a partir de una *hoja de estilos*.

- ✓ *StyleLoader*. Clase principal del cargador de estilos, contiene la definición de los métodos utilizados para aplicar a todos los controles de un formulario su estilo correspondiente, definido en la *hoja de estilos* activa (seleccionada previamente).
- ✓ *cEstiloDefecto*. Clase que interactúa con la base de datos, accediendo únicamente a la tabla de estilos, *tblStyles*, para consultarla.

***Figura 5-16**. Conjunto de clases que componen el proyecto StyleLoader.*

5.2.3. Generador de estilos.

El generador de estilos es una herramienta de la plataforma de administrador que permite crear nuevos estilos, así como eliminar o modificar los ya existentes. Genera *hojas de estilos*, que se guardan en un directorio específico de la plataforma.

- ✓ *frmStyleGen*. Formulario principal del generador de estilos, contiene los métodos que permiten cargar un estilo existente en el editor para modificarlo, crear nuevos estilos, y seleccionar el estilo activo de la plataforma.
- ✓ *frmNewStyle*. Pequeño formulario que se muestra al usuario al disponerse a crear un nuevo estilo, para que introduzca los datos que identificarán al mismo.

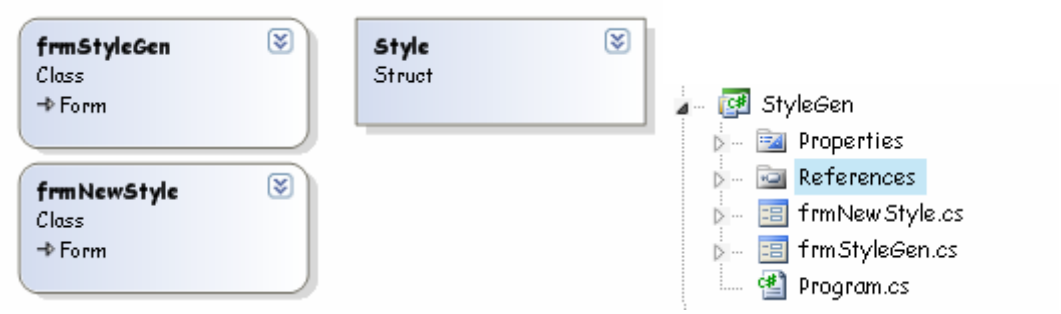


Figura 5-17. *Conjunto de formularios que componen el proyecto StyleGen.*

El proyecto *NegStyleGen* implementa la capa de negocio del generador de estilos. Está compuesta por dos clases destinadas a realizar operaciones contra la base de datos.

- ✓ *cNegStyleGen*. Definición de los métodos necesarios para acceder la tabla *tblStyles*, en la que se guarda toda la información referente a los estilos definidos en la plataforma. Realiza no tan sólo consultas, sino también altas bajas y modificaciones de los registros de dicha tabla.
- ✓ *clsOperaciones*. Clase que tiene implementadas numerosas funciones genéricas para ejecutar las operaciones más comunes contra la base de datos, entre ellas la conexión y desconexión con la misma.

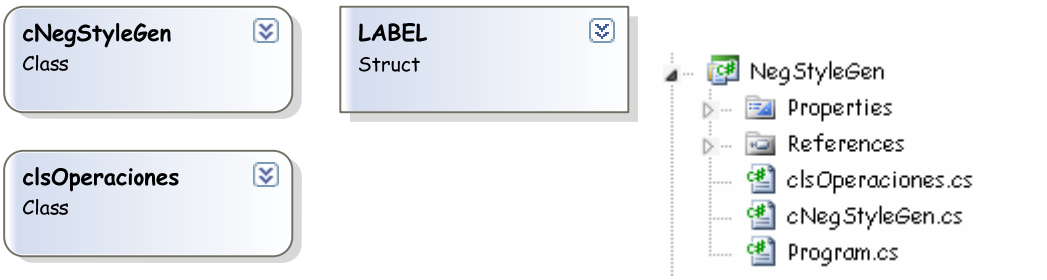


Figura 5-18. *Conjunto de clases que componen el proyecto NegStyleGen.*

5.2.4. Generador de recursos de imágenes.

Esta es otra de las herramientas de la plataforma de administrador, que permite al usuario crear y modificar los paquetes de iconos/imágenes del sistema. Mediante el generador de recursos de imágenes y el generador de estilos, es posible personalizar por completo el aspecto visual de la plataforma, de forma rápida y sencilla, a gusto de cada usuario.

- ✓ *frmIRMMain.* Formulario principal del generador de recursos de imágenes. Tiene implementadas las funciones que permiten renombrar los archivos de recursos de imágenes, cargar su contenido para añadirles o quitarles imágenes, crear nuevos paquetes de iconos, etc.
- ✓ *frmRenameResx*. Pequeño formulario que se muestra al usuario al disponerse a crear o renombrar un archivo de recursos de imágenes, para que introduzca los datos que identificarán al mismo.

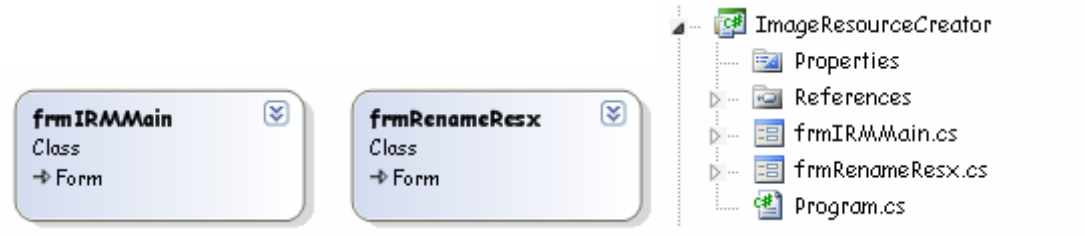


Figura 5-19. *Conjunto de formularios que componen el proyecto ImageResourceCreator.*

El proyecto *NegImgResources* implementa la capa de negocio del generador de recursos de imágenes.

- ✓ *cTblImgResources.* Definición de los métodos que acceden a la tabla *tblImgResources* para realizar consultas, altas, bajas y modificaciones.

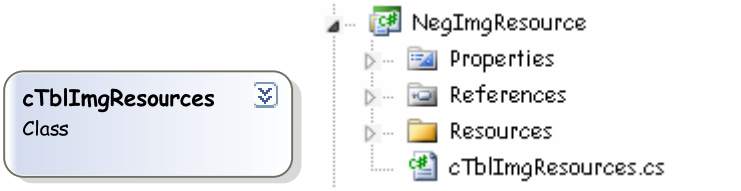


Figura 5-20. *Conjunto de clases que componen el proyecto NegImgResource.*

5.2.5. Gestión de menús.

Este proyecto está específicamente dedicado

- ✓ *cTblMenuMen.* Esta clase implementa la capa de negocio del proyecto *GestionMenus*; en ella están definidos los métodos que permiten acceder a las tablas *tblFileMen* y *tblMenuMen* para realizar consultas, inserciones y modificaciones de los datos que contienen.

- ✓ *cCargaMenus.* Definición de los métodos que se utilizan para cargar los menús modulares almacenados en la tabla *tblMenuMen*. Podemos ver un ejemplo de funcionamiento de estos métodos en el árbol explorador del formulario principal de la plataforma de cliente.
- ✓ *cGuardaMenus.* Definición de los métodos destinados a realizar inserciones o actualizaciones en la tabla *tblMenuMen*. Esta clase se utiliza en el editor de menús para crear menús modulares nuevos o modificar los ya existentes.

Figura 5-21. *Conjunto de clases que componen el proyecto GestionMenus.*

5.2.6. Plataforma de administrador.

La plataforma de administrador está compuesta por un conjunto de herramientas destinadas exclusivamente a los programadores de la empresa AxialSoft, S.L. Entre los formularios principales de este proyecto se encuentran:

- ✓ *FrmMain*. Formulario principal de la plataforma de administrador, es de tipo *MdiContainer*.
- ✓ *FrmExplorador*. Primer formulario *MdiChildren* del formulario principal. El árbol explorador que muestra las herramientas disponibles en la plataforma de cliente se carga estáticamente por código.
- ✓ *FrmEditorModulos*. Permite editar el icono asociado y la descripción de los módulos de la plataforma de cliente.
- ✓ *FrmPerfiles*. Editor de los perfiles de acceso de la plataforma de cliente.
- ✓ *FrmNuevoPerfil*. Permite definir un nuevo perfil de acceso; se ejecuta desde el formulario *FrmPerfiles*.
- ✓ *FrmDefMenus*. Editor de menús modulares de la plataforma de cliente. Utiliza el proyecto *GestionMenus* como capa de negocio.

Figura 5-22. *Formularios principales del proyecto AppAdministrador.*

5.2.7. Axiwin – Plataforma de cliente.

La plataforma de cliente es, en sí misma, la aplicación que la empresa AxialSoft, S.L. va a comercializar. A pesar de que el núcleo del E.R.P. está compuesto por muchos proyectos, cada uno con sus respectivos formularios y clases, los principales formularios de la plataforma cliente son los siguientes:

- ✓ *FrmMainCliente*. Formulario principal de la plataforma de cliente, es de tipo *MdiContainer*.
- ✓ *FrmExploradorCliente***.** Primer formulario *MdiChildren* del formulario principal. El árbol explorador que muestra las herramientas disponibles en la plataforma de cliente se carga estáticamente por código, exceptuando los módulos, que se cargan de acuerdo a los menús modulares asignados a cada usuario que se autentica en la plataforma.
- ✓ *FrmPermisosCliente***.** Formulario para gestionar los usuarios del sistema, permitiendo asignarles los menús modulares y definir sus permisos específicos por aplicaciones y campos. Interactúa con las tablas de la base de datos *tblAcceso* y *tblRestriccion*.
- ✓ *FrmNewUserWizard.* Pequeño formulario que permite dar de alta a un usuario en el sistema; se ejecuta desde el formulario *FrmPermisosClientes*.

- ✓ *FrmConfigBotonera.* Permite al usuario configurar la distribución de los botones en la botonera principal del formulario *FrmMainCliente*. Interactúa con la base de datos, utilizando la clase *clsBotoneras* como capa de negocio.
- ✓ *FrmProgFunciones.* Permite al usuario definir las acciones asociadas a las teclas de función F1 hasta F12 y a las combinaciones de dichas teclas con *Shift*, así como de los 10 botones numerados de la botonera.
- ✓ *FrmGestionAuditoria.* Permite obtener la información relativa a la auditoria de todas las operaciones de consulta, alta, baja y modificación de datos llevadas a cabo en la plataforma de cliente. Utiliza ficheros XML para almacenar dicha información.

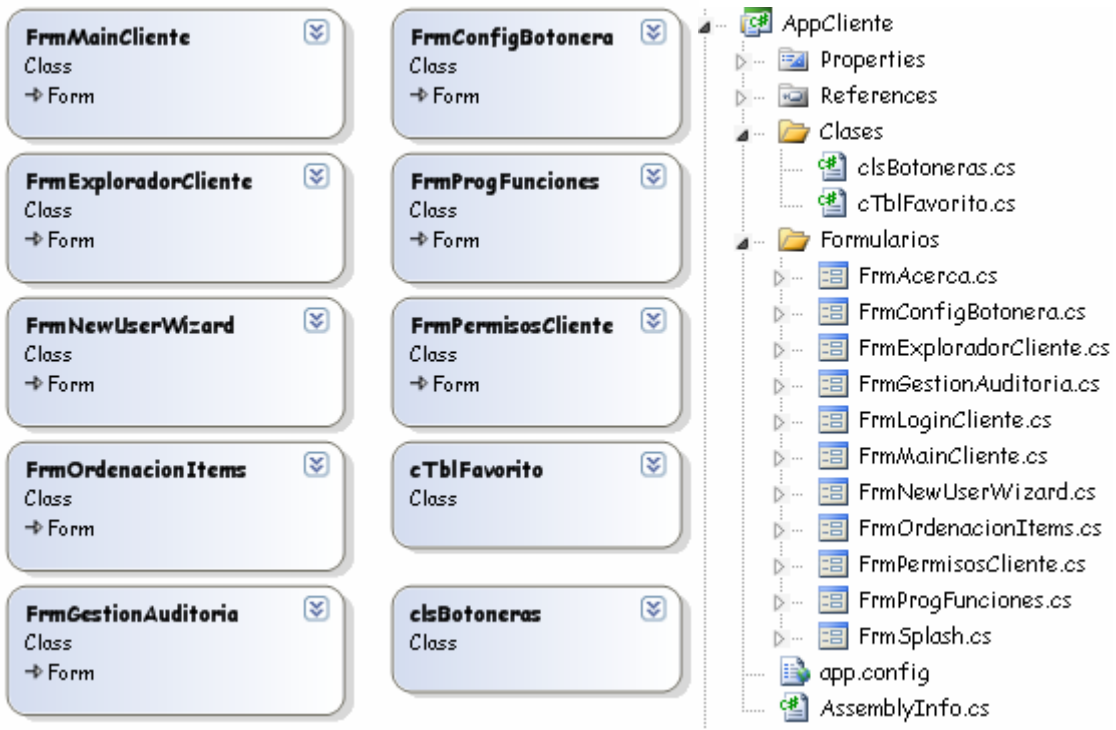


Figura 5-23. *Formularios principales y clases del proyecto AppCliente o Axiwin.*

5.3. Librerías.

Los siguientes proyectos están referenciados en la plataforma *Axiwin* como librerías dinámicas (DLL's). Todos ellos se han desarrollado como parte de la solución de Visual Studio .NET que genera los ejecutables de las plataformas de administrador y cliente, pero no forman parte de este proyecto.

- ✓ AxiClient.
- ✓ Axilis_Acciones.
- ✓ Axilis_Definir.
- ✓ Axilis_Obtener.
- ✓ Axilis_Plantillas.
- ✓ AxiMessage.
- ✓ AxiMigrador.
- ✓ AxinFont.
- ✓ AxiPack.
- ✓ AxiScript.
- ✓ AxiUpdate.
- ✓ EstructurasCodigo.
- ✓ ConexionBaseDatos.
- ✓ Interpretador.
- ✓ LangLoader.
- ✓ NegAxiCUP.
- ✓ NegAxiGen.
- ✓ NegNucleo.
- ✓ PreAxigen.
- ✓ PresCargaPantallas.
- ✓ PreCompilador.
- ✓ RptConnect.

A éstas hay que añadir las DLL's que generan los controles implementados específicamente para la nueva plataforma *Axiwin*.

Por otra parte, en la fase de migración fue necesario y fundamental utilizar la OCX (OLE control extensión) *agigen*, proporcionada e implementada por la empresa AxialSoft, S.L. en Visual C++ 5, para poder extraer la información de los ficheros de campos .mas.

Capítulo 6. Resultados.

Capítulo 6. Resultados.

Antes de centrarnos en la evaluación cualitativa de los resultados obtenidos, es importante recalcar que se han cumplido satisfactoriamente los principales requisitos del cliente respecto al desarrollo del nuevo E.R.P. *Axiwin.* Se ha creado una plataforma*:*

- ✓ Totalmente compatible con su antecesora, mediante a la implementación de unas complejas y elaboradas herramientas de migración de datos.
- ✓ Completamente dinámica y muy rápida, gracias a la intensiva utilización de un sistema gestor de base de datos, un elemento fundamental e indispensable del nuevo E.R.P.
- ✓ Altamente personalizable, con un aspecto moderno, atractivo, y fácilmente adaptable al paso del tiempo, lo que le garantiza un largo ciclo de vida.

6.1. Consideraciones sobre el proyecto de migración.

- ✓ El proceso de migración tenía un consumo de memoria muy elevado, derivado de una mala gestión de los recursos y una programación algo descuidada. Este hecho era tan crítico que, al migrar grandes volúmenes de datos, la aplicación se quedaba colgada, siendo incluso necesario reiniciar el equipo para solucionar el problema.

  Solución: Se ha sobrecargado el método *dispose()* en la mayoría de las clases que participan en el proceso de migración, forzando al *garbage collector* a ejecutarse más a menudo para liberar recursos.

- ✓ Los resultados obtenidos con la migración de pantallas no son todo lo satisfactorios que deberían ser. Pese a traducir perfectamente la distribución de las etiquetas y las cajas de texto en las pantallas, la carga dinámica de las mismas muestra irregularidades. Detectamos que el origen de dichas irregularidades se hallaba en el tamaño de las fuentes, calculado utilizando píxels en vez de caracteres como unidades de medida. Además no se ha

conseguido migrar todos los detalles (marcos y delimitadores) de las pantallas originales.

Solución: Para poder arreglar/ajustar manualmente las pantallas que presentan alguna anomalía en la carga dinámica de controles, se ha creado el editor PreAxigen. Esta herramienta permite no tan sólo editar las propiedades de cada control individualmente, sino también añadir líneas y controles de tipo *Groupbox* a las pantallas, mejorando su aspecto enormemente.

6.2. Consideraciones sobre la nueva plataforma.

- ✓ La carga de pantallas es un proceso excesivamente lento, lo cual es inaceptable en una plataforma de uso comercial de la envergadura del E.R.P. *Axiwin*, especialmente cuando se utiliza una base de datos en sustitución del sistema de ficheros original. Esta ralentización en la carga dinámica de pantallas se debe a la creación, en tiempo de ejecución, de una enorme cantidad de controles (entre *labels*, *textbox*, y demás) en base a la información almacenada en distintas tablas de la base de datos, a lo que hay que añadir la posterior carga de estilos e idiomas.

  Solución: Se está estudiando la forma de optimizar y acelerar el proceso de carga, utilizando la misma filosofía de sobrecargar el método *dispose()* y obligar al *garbage collector* a liberar recursos con mucha más frecuencia de la habitual.

- ✓ La nueva plataforma *Axiwin* no está acabada en su totalidad, todavía quedan por implementar algunas de las funcionalidades esenciales para poder comercializarla. A mediados de octubre de 2006 se entregará al cliente una primera versión, no definitiva, a fin de presentar el E.R.P. en la Fira Expolimp Hostelco de Barcelona, con carácter demostrativo. La plataforma tendrá que estar finalizada a principios de noviembre, cuando se entregará a la empresa AxialSoft, S.L. para que realice la presentación oficial del E.R.P. en el SIMO de Madrid.

Capítulo 7.
Análisis económico.

Capítulo 7. Análisis económico.

7.1. Distribución del coste temporal en fases.

El coste temporal de este Trabajo Final de Carrera está dividido en 4 grandes fases principales, las cuales a su vez están divididas en subfases.

- ✓ ***Fase 1. Estudio teórico y ampliación de la plataforma Axiwin original.***

Esta fase comprende el estudio y documentación del E.R.P. *Axiwin*, propiedad de la empresa AxialSoft, S.L., y las posteriores fases de corrección de problemas de funcionamiento e incorporación de novedades en el código fuente original. Esta fase representa casi un 40% del total de horas dedicadas al desarrollo de este proyecto.

- ✓ ***Fase 2. Migración y adaptación de pantallas y menús.***

Implementación de las aplicaciones necesarias para llevar a cabo la migración del sistema de ficheros de la plataforma original a base de datos. Este proyecto se centra en la traducción de los ficheros de pantallas y menús. Junto con la fase de estudio teórico, esta es la que ha tenido mayor coste temporal, un 34%.

- ✓ ***Fase 3. Análisis funcional y orgánico.***

Diseño de los formularios principales de las plataformas de administrador y cliente del nuevo E.R.P. *Axiwin*, y de las tablas de la base de datos. Esta fase representa un 11% del coste temporal total del proyecto.

- ✓ ***Fase 4. Implementación de la nueva plataforma.***

Implementación del código asociado a los formularios diseñados en la fase anterior. Implementación de la interacción con la base de datos (capa de negocio de la plataforma) y desarrollo, entre otras, de las herramientas *Generador de estilos*, *Generador de recursos de imágenes*, *Editor de menús*, etc. Esta fase representa un 17% del coste temporal total del proyecto.

7.2. Coste total del Trabajo Final de Carrera.

A continuación se muestran dos gráficos que ilustran la distribución del coste temporal en las fases comentadas en el apartado anterior.

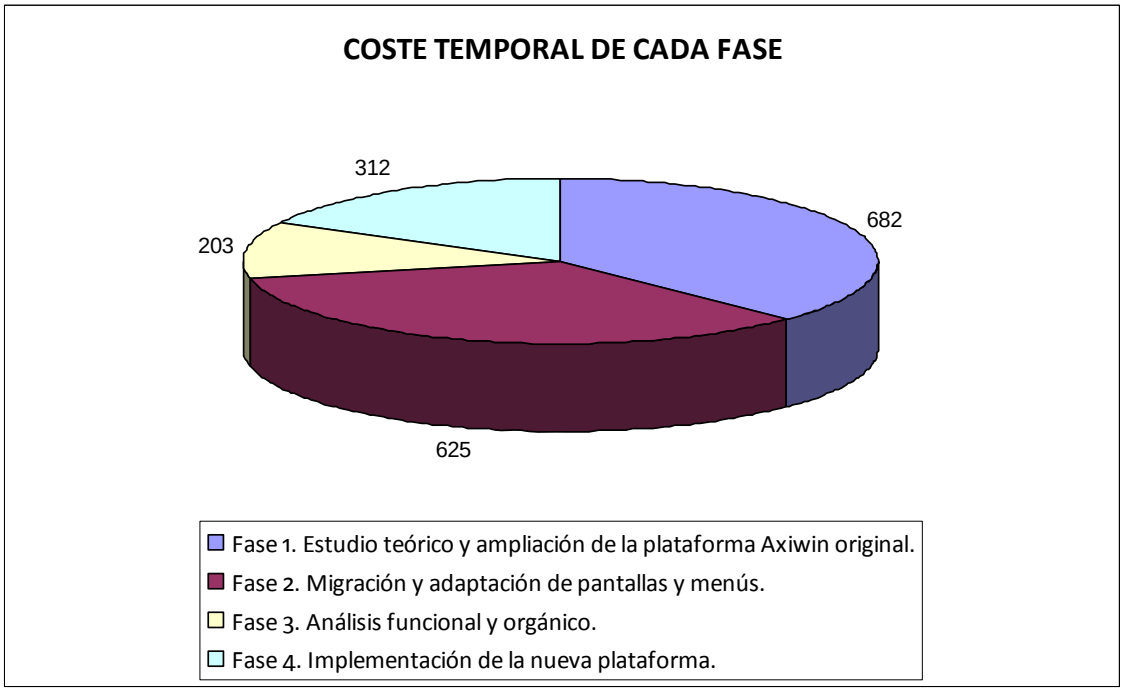


***Figura 7-1**. Coste temporal de cada fase.*

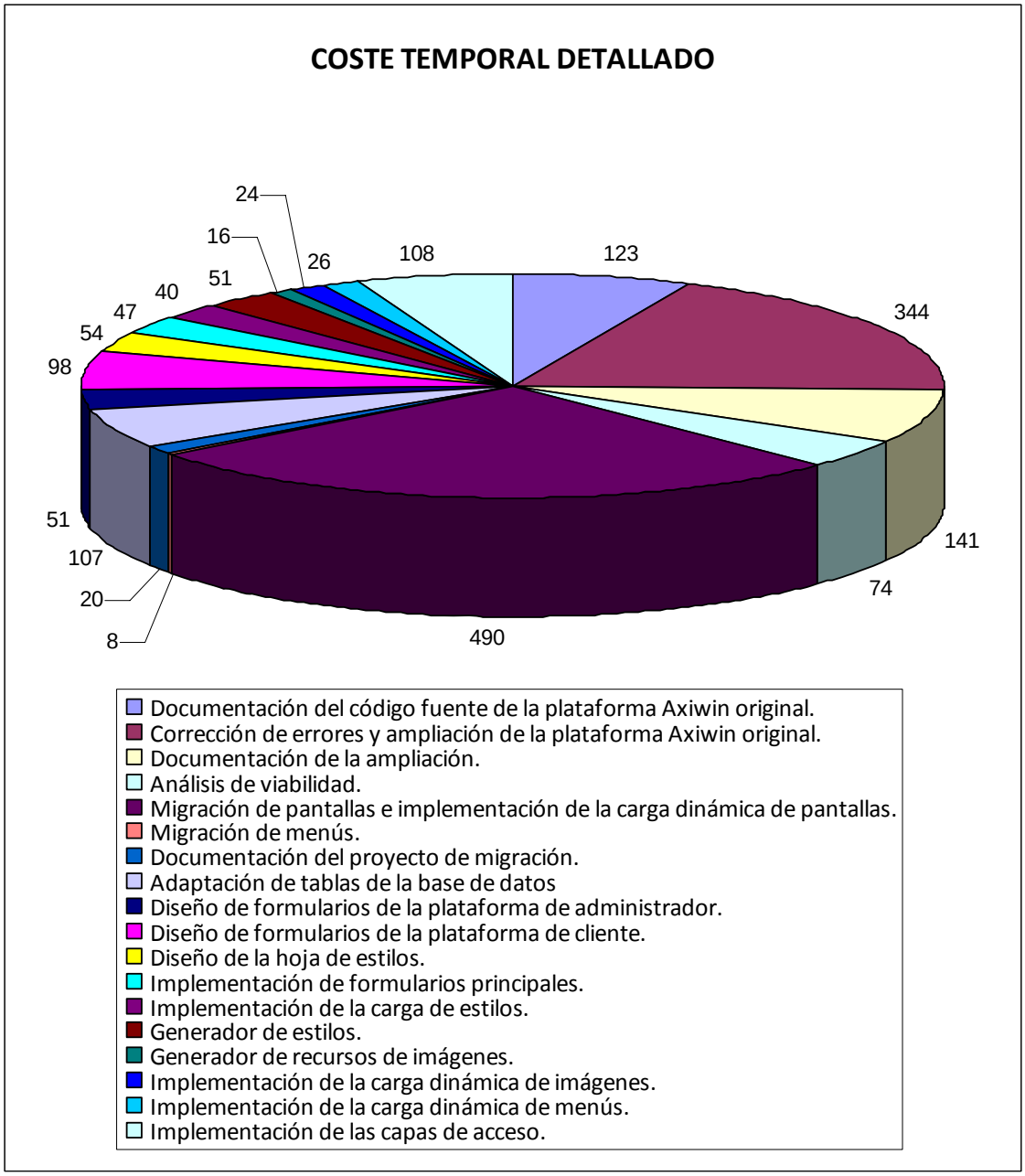


***Figura 7-2**. Coste temporal detallado de cada fase del proyecto.*

Capítulo 8.
Conclusiones y líneas de futuro.

Capítulo 8. Conclusiones y líneas de futuro.

8.1. Conclusiones.

Este Trabajo Final de Carrera ha batido muchos récords personales; entre otros, ha representado:

- ✓ La aplicación más grande implementada hasta el momento, en cualquier lenguaje de programación.
- ✓ La primera aplicación Windows, desarrollada en Visual Studio .NET y utilizando el lenguaje C#.
- ✓ El primer proyecto software llevado a cabo junto a un equipo de trabajo y bajo la dirección de un jefe de proyecto.
- ✓ La primera aplicación desarrollada para uso comercial, destinada al mercado de las pequeñas y medianas empresas.

Se ha tenido la oportunidad de ejercer de técnico-analista antes de obtener el título de Ingeniería en Informática de Sistemas, lo que ha representado a la vez una experiencia laboral real muy provechosa, y un reto muy grande que ha requerido de mucha responsabilidad y capacidad para tomar decisiones importantes.

En las numerosas fases que han compuesto el desarrollo de este proyecto se ha aprendido sobre muchas y muy diferentes disciplinas, todas ellas relacionadas con la implementación de una plataforma software, y se ha trabajado con gran cantidad de aplicaciones distintas, desconocidas hasta el momento; entre ellas:

- ✓ Microsoft Visual C++ 5 y 6, y Microsoft Visual Studio .NET 2003 y 2005, los entornos de desarrollo utilizados, con los que se ha aprendido a crear controles, utilizar DLL's (librerías dinámicas) y OCX's (OLE control extension), tratar con ficheros XML, etc.
- ✓ Microsoft SQL Server 2000 y Microsoft SQLExpress 2005, el sistema gestor de base de datos sobre el que se apoya firmemente el E.R.P. *Axiwin*, gracias al cual se ha practicado intensivamente la utilización del potentísimo lenguaje relacional SQL.

- ✓ Crystal Reports, para crear plantillas y generar listados.
- ✓ CVS, un sistema de control de versiones que facilita el trabajo en equipo.
- ✓ Aplicaciones de edición de iconos.

El primer contacto con Visual Studio .NET y C# ha sido totalmente satisfactorio, descubriendo en el primero la plataforma de desarrollo más versátil y potente del mercado hoy en día, y en el segundo un lenguaje flexible, sencillo e increíblemente completo, que ofrece al programador recursos casi ilimitados para implementar, prácticamente, cualquier funcionalidad imaginable.

Ha sido especialmente gratificante y muy instructivo el hecho de formar parte de un completo equipo de trabajo, que ha sacado el proyecto Axialsoft adelante a base de constancia y mucha voluntad, desarrollando paralelamente, pero de forma independiente, las diferentes partes en que está dividido el E.R.P. *Axiwin*. Es interesante mencionar que, en los inicios del proyecto, el equipo estaba formado únicamente por 2 técnicos y el jefe de proyecto y, entre bajas e incorporaciones, la plantilla ha llegado a contar con 9 técnicos trabajando a la vez.

Por otra parte, trabajar con un equipo de gente tan grande requiere de mucha organización y comunicación interna, lo que en algunas ocasiones ha originado confusión y caos ya que, por muy separadas y diferenciadas que estén las tareas de cada miembro del equipo, es inevitable que éstas converjan en algún momento. La herramienta de control de versiones utilizada para gestionar todo el código de la plataforma *Axiwin* ha sido, a pesar de su fiabilidad, uno de los principales motivos de conflicto al arruinar, en más de una ocasión, preciosas horas de trabajo con una fusión incorrecta de dos versiones distintas de un mismo archivo.

En lo que a la consecución de metas se refiere, pese a las dificultades encontradas en cada fase del desarrollo de este proyecto, el resultado final es muy satisfactorio, habiendo cumplido los principales requisitos del cliente de crear una plataforma dinámica, personalizable, visualmente atractiva y totalmente compatible con su antecesora.

Cabe destacar, finalmente, que este Trabajo Final de Carrera ha significado más de dos años de entrega y compromiso. En todo este tiempo el proyecto Axialsoft ha transcurrido por etapas muy distintas, comprendiendo desde análisis y documentación a diseño e implementación de código, tomando forma poco a poco, pasando de ser un conjunto de documentos de especificaciones y análisis de viabilidad a convertirse en una realidad. Pese a eso, persiste la sensación de haber dedicado dos años a realizar, día tras día, la misma tarea. Y, aunque los momentos en los que se ha disfrutado desarrollando este proyecto, junto con todo el conocimiento adquirido, compensan con creces los momentos más duros, la monotonía y el aburrimiento han llevado a una situación insostenible de agotamiento mental. Lo que lleva a la conclusión de que es recomendable que los primeros pasos que se den en el mundo laboral sean con proyectos pequeños que abarquen distintas disciplinas y que permitan la rotación y oxigenación del equipo de trabajo para no desgastar a sus componentes.

8.2. Líneas de futuro.

- ✓ ***Portabilidad***.

Actualmente el E.R.P. *Axiwin* sólo puede ejecutarse en sistema operativos Windows, y más concretamente en versiones superiores a Windows 95. La empresa AxialSoft, S.L. tiene intención de adaptar la plataforma para su utilización en sistemas operativos UNIX/Linux. Esta adaptación de haría, muy probablemente, con la ayuda de *Mono Project*.

Mono Project proporciona el software necesario para desarrollar y ejecutar aplicaciones .NET de tipo cliente y servidor en máquinas Linux, Solaris, Mac OSX, Windows y Unix. Mono también es compatible con aplicaciones desarrolladas en Java, Python y otros lenguajes de programación. Además es un software gratuito y de código abierto (*open source*).

- ✓ ***Aplicación web y aplicación para dispositivos móviles.***

La empresa AxialSoft, S.L. se plantea la creación de una versión *web* del E.R.P. *Axiwin*, lo que a la vez serviría para solucionar el problema de compatibilidad con UNIX/Linux, dado que los formularios *web* se podrían acceder igualmente desde cualquier sistema operativo. Es necesario hacer un estudio de viabilidad para determinar la factibilidad de llevar a cabo este proyecto, que requeriría volver a crear gran parte de los formularios de la plataforma *Axiwin*.

Se ha planteado también la posibilidad de crear una versión para PocketPC o PDA del E.R.P. *Axiwin*. La adaptación sería relativamente sencilla, dado que Windows CE, el sistema operativo para dispositivos móviles de Microsoft, es compatible con Visual Studio .NET.

- ✓ ***Nuevo Axigen.***

Actualmente el generador de aplicaciones, integrado por el editor de código, el compilador y el editor de pantallas, está incompleto. El editor de código no permite crear nuevas aplicaciones, ni editar el código de las ya existentes; el compilador carece

del analizador semántico, y el editor de pantallas sólo permite la modificación de las pantallas existentes en la base de datos, pero no la creación de pantallas nuevas. Además, el proyecto *Axigen* tiene planificada la inclusión de un debugador (*debugger*) de código. Está planificado llevar a cabo todas estas mejoras próximamente.

- ✓ ***AxiAyuda.***

Implementación de la herramienta de ayuda del E.R.P. *Axiwin*. Esta herramienta se ejecutará de forma dinámica, mostrando al usuario la información pertinente según la aplicación en la que éste se encuentre al solicitar la ayuda.

- ✓ ***AxiDocumentador.***

Implementación de una herramienta que genere automáticamente la documentación asociada a las aplicaciones desarrolladas por AxialSoft, S.L. en su propio lenguaje de programación, mediante la utilización de *tag's* predefinidos en el código fuente. Esta herramienta estaría vinculada al generador de aplicaciones *Axigen* y a la *AxiAyuda*, comentada en el punto anterior.

Capítulo 9. Bibliografía.

Capítulo 9. Bibliografía.

KERNIGHAN, B.W.; RITCHIE, D.M. (1991). *El lenguaje de programación C: [con base en ANSI C]*. 2ª ed. Editorial Prentice Hall. ISBN 978-968-880-205-2.

GUNDERLOY, M.; JORDEN, J.L. (2000). *Mastering SQL Server 2000*. 1ª ed. Editorial Sybex. ISBN 0-7821-2627-8.

STANSFIELD, S. (1996). *Cómo se hace con Visual C++ 4*. 1ª ed. Editorial Inforbook's, S.L. ISBN 84-89700-34-6.

SLUSSER, J. "Getting a 'Handle' on the MDI Client".
http://www.codeproject.com/cs/miscctrl/mdiclientcontroller.asp

THROWER, E. "Combobox over listview cell?".
http://www.dotnet247.com/247reference/msgs/14/71810.aspx

TSKHVARADZE, V. "How to create a DLL library in C and then use it with C#".
http://www.codeproject.com/csharp/UseCDLLlibinCS.asp

SHEPHERD, G. "How do I color an individual cell depending upon its value or some external method".
http://www.syncfusion.com/faq/windowsforms/search/745.aspx

VALDEZATE SAYALERO, C. "Controles planos y sensibles al tacto en C#. Efecto bidimensional y reacción al contacto redibujando el control con GDI+".
http://www.elguille.info/colabora/puntoNET/ControlesFlacos.htm

"HOW TO: Trap Keystrokes in .NET Controls by Using Visual C# .NET".
http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B320584

MICROSOFT DEVELOPER'S NETWORK (*Biblioteca de ayuda online de Microsoft*).
http://msdn.microsoft.com

WIKIPEDIA
http://www.wikipedia.org

GOOGLE.
http://www.google.com

Anexo I.

Anexo I. Hoja de estilos.

Se añade a continuación una de las *hojas de estilos* que utiliza la nueva plataforma *Axiwin* para cargar el aspecto visual de sus formularios. La *hoja de estilos* no es más que un fichero XML que está, por lo tanto, estructurado en forma de tablas de datos. Ésta *hoja de estilos* en concreto contiene la definición del estilo que muestra la plataforma *Axiwin* por defecto, llamado *axialStyle*.

```
<?xml version="1.0" standalone="yes"?>
<Style>
  <ControlAttributes>
    <Forms>
      <BackColor>#8d989d</BackColor>
      <BackImg>None</BackImg>
      <BackImgLayout>None</BackImgLayout>
      <FormIcon>icono_axial.ico</FormIcon>
      <ForeColor>White</ForeColor>
      <FontName>Microsoft Sans Serif</FontName>
      <FontSize>9</FontSize>
      <FontStyle>Regular</FontStyle>
    </Forms>
    <Panel>
      <BackColor>#8d989d</BackColor>
      <BorderColor>White</BorderColor>
      <BorderWidth>1</BorderWidth>
    </Panel>
    <Buttons>
      <BackImg>boton_Normal.png</BackImg>
      <BackImgInactive>boton_deshabilitado.png</BackImgInactive>
      <BackImgMouseOver>boton_sobre.png</BackImgMouseOver>
      <BackImgMouseDown>boton_abajo.png</BackImgMouseDown>
      <BackImgLayout>Stretch</BackImgLayout>
      <ForeColor>#000000</ForeColor>
      <FontName>Microsoft Sans Serif</FontName>
      <FontSize>9</FontSize>
      <FontStyle>Regular</FontStyle>
    </Buttons>
    <Textbox>
      <BackColor>White</BackColor>
      <BackColorInactive>#8d989d</BackColorInactive>
      <BorderColor>#506470</BorderColor>
      <BorderWidth>1</BorderWidth>
      <ForeColor>#000000</ForeColor>
      <FontName>Microsoft Sans Serif</FontName>
      <FontSize>9</FontSize>
      <FontStyle>Regular</FontStyle>
    </Textbox>
    <Listview>
      <BackColor>#E0E0E0</BackColor>
      <BackImg>None</BackImg>
      <BackImgTiled>False</BackImgTiled>
      <ForeColor>Black</ForeColor>
```

```
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Listview>
  <Labels>
    <BackColor>#8d989d</BackColor>
    <BorderColor>#506470</BorderColor>
    <BorderWidth>1</BorderWidth>
    <ForeColor>White</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Labels>
  <Combobox>
    <BackColor>White</BackColor>
    <BackColorInactive>#8d989d</BackColorInactive>
    <BorderColor>#506470</BorderColor>
    <BorderWidth>0</BorderWidth>
    <ForeColor>#000000</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Combobox>
  <Groupbox>
    <BackColor>Transparent</BackColor>
    <BorderColor>#506470</BorderColor>
    <BorderWidth>1</BorderWidth>
    <ForeColor>#ffffff</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Groupbox>
  <Menu>
    <BackColor>#8d989d</BackColor>
    <ForeColor>White</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Menu>
  <Toolbar>
    <BackColor>#8d989d</BackColor>
    <ForeColor>#000000</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Toolbar>
  <Tabcontrol>
    <BackColor>#8d989d</BackColor>
    <BackImg>None</BackImg>
    <BackImgLayout>None</BackImgLayout>
    <TabColor>#8d989d</TabColor>
    <TabColorInactive>#506470</TabColorInactive>
    <BorderColor>#506470</BorderColor>
    <ForeColor>#000000</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Tabcontrol>
  <Treeview>
```

```
    <BackColor>White</BackColor>
    <LineColor>#506470</LineColor>
    <ForeColor>#000000</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Treeview>
  <Datagrid>
    <BackColor>White</BackColor>
    <AltRowBackColor>White</AltRowBackColor>
    <GridColor>#506470</GridColor>
    <ForeColor>#000000</ForeColor>
    <CaptionBackColor>#000000</CaptionBackColor>
    <CaptionForeColor>#FFFFFF</CaptionForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
  </Datagrid>
</ControlAttributes>
<ButtonImages>
  <Image>
    <NameActive>aceptar.png</NameActive>
    <NameInactive>aceptar_des.png</NameInactive>
    <Id>1</Id>
    <Alignment>MiddleLeft</Alignment>
  </Image>
  <Image>
    <NameActive>cancelar.png</NameActive>
    <NameInactive>cancelar_des.png</NameInactive>
    <Id>2</Id>
    <Alignment>MiddleLeft</Alignment>
  </Image>
  <Image>
    <NameActive>izquierda.png</NameActive>
    <NameInactive>izquierda_des.png</NameInactive>
    <Id>3</Id>
    <Alignment>MiddleLeft</Alignment>
  </Image>
  <Image>
    <NameActive>derecha.png</NameActive>
    <NameInactive>derecha_des.png</NameInactive>
    <Id>4</Id>
    <Alignment>MiddleRight</Alignment>
  </Image>
  <Image>
    <NameActive>impresora.png</NameActive>
    <NameInactive>impresora_des.png</NameInactive>
    <Id>5</Id>
    <Alignment>MiddleLeft</Alignment>
  </Image>
  <Image>
    <NameActive>ayuda.png</NameActive>
    <NameInactive>ayuda_des.png</NameInactive>
    <Id>6</Id>
    <Alignment>MiddleLeft</Alignment>
  </Image>
</ButtonImages>
<MdiContainer>
  <BackImage>agitapiz-gris.png</BackImage>
```

```
    <BackImageLayout>Stretch</BackImageLayout>
  </MdiContainer>
  <PanelInfo>
    <BackColor>#ffdca6</BackColor>
    <BackImage>None</BackImage>
    <BackImageLayout>Stretch</BackImageLayout>
    <BorderColor>#fe9901</BorderColor>
    <BorderWidth>1</BorderWidth>
    <LabelBackColor>Transparent</LabelBackColor>
    <LabelBorderColor>Transparent</LabelBorderColor>
    <ForeColor>#000000</ForeColor>
    <FontName>Microsoft Sans Serif</FontName>
    <FontSize>9</FontSize>
    <FontStyle>Regular</FontStyle>
    <Cenefa>logo_cenefa.png</Cenefa>
  </PanelInfo>
  <PanelCenefa>
    <BackColor>Transparent</BackColor>
    <BackImage>axicenefa.bmp</BackImage>
    <BackImageLayout>Stretch</BackImageLayout>
  </PanelCenefa>
</Style>
```

Anexo II.

Anexo II. Sobrescribiendo el método WndProc.

El siguiente código muestra cómo se ha sobrescrito el método de pintado *WndProc()* propio del control *Textbox*, creando el nuevo control *FlatTextbox* tras incorporarle al primero dos propiedades que permiten definir el color y el grosor del marco de la caja de texto. Esta función es igual para los controles *FlatLabel*, *FlatPanel* y *FlatCombobox*.

```
protected override void WndProc(ref Message m)
{
```

La siguiente línea llama a la función original de pintado del control.

```
        base.WndProc (ref m);
        Graphics g = this.CreateGraphics();
```

A continuación creamos un rectángulo cuyas medidas coinciden con los límites del control. Vamos a utilizar el rectángulo para pintar el marco del control.

```
        Rectangle Cliente = this.ClientRectangle;
```

Establecemos el color del marco según el valor del atributo *BorderColor*.

```
        _ActiveBorderColor = _BorderColor;
```

Si el valor del atributo *BorderWidth* (grosor del marco) es mayor que o, pintamos el marco, sino no.

```
        if (this.BorderStyle != BorderStyle.None || this.BorderWidth > 0)
        {
```

Finalmente, pintamos el rectángulo vacío sobre el control, haciendo coincidir sus vértices y su tamaño de modo que el rectángulo simule el marco del control. La sintaxis de la función *g.DrawRectangle* es la siguiente:

```
void DrawRectangle(  Pen pen,        //lápiz (determina color y ancho del marco)
                     float x,        //coordenada X
                     float y,        //coordenada Y
                     float width,    //ancho del rectángulo a dibujar
                     flota height)   //altura del rectángulo a dibujar
```

```
            g.DrawRectangle(new Pen(_ActiveBorderColor, _BorderWidth), 0, 0,
                            Cliente.Width - 1, Cliente.Height - 1);
        }
        g.Dispose();
}
```