Universitat Autònoma
de Barcelona

GESTOR COMERCIAL

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Sistemes

realitzat per

Xavier Royo Melero

i dirigit per

Marc Talló Sendra

Escola d'Enginyeria

Sabadell, Setembre de 2010

El sotasignat, ***Marc Talló Sendra***

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per en ***Xavier Royo Melero***

I per a que consti firma la present. Sabadell, ***Setembre*** de ***2010***

Signat: ***Marc Talló Sendra***

Dedicado a todas aquellas personas que, pese a todo, han creído en mí

Resumen

A lo largo de esta memoria se explicarán todos los aspectos fundamentales para la creación del proyecto que se presenta.

Este se basa en una aplicación de escritorio que permita a un vendedor tomar nota de los pedidos que resulten de sus visitas comerciales y su posterior envío. Para ello se han desarrollado una serie de herramientas y opciones que hagan esta tarea sea mucho más rápida y fiable.

Pese a haberse podido enfocar esta solución de maneras muy diferentes se ha tratado de construir una estructura sobre la que se puedan ir desarrollando nuevas mejoras en caso de necesidad.

Es por ello que se tratará de explicar en este documento todos estos aspectos así como otros factores considerados de mención especial.

ÍNDICE

1. Introducción

El proyecto de final de carrera supone la culminación en forma de práctica de los conocimientos adquiridos durante los años de formación académica y laboral.

Con el objetivo de reflejar significativamente una parte de tales conocimientos se ha decidido utilizar una aplicación de gestión comercial como escenario de fondo, la cual nos permitirá desarrollar diferentes aspectos como la programación orientada a objetos, la gestión de bases de datos, comunicación con servidores de datos y la interactuación con ellos.

A medida que se iba realizando el proyecto surgían nuevas ideas en cada uno de estos apartados, pero que pese no haberse llevado a cabo para no extender excesivamente el marco de trabajo establecido por el director de proyecto, se mencionarán brevemente.

Las impresiones al término de la realización del proyecto han sido muy satisfactorias a la vez que han despertado la motivación para profundizar, en un futuro no muy lejano, en aspectos relacionados con el proyecto.

1.1 Presentación

El proyecto pretende ofrecer una herramienta completa para la gestión de los pedidos de un representante comercial.

Esta gestión abarcará diferentes aspectos, como la toma de pedidos, almacenamiento de los mismos, posibles modificaciones posteriores y su envío a la empresa de destino. También se proveerá al usuario de herramientas de análisis estadísticos para la explotación de la información.

En el ejemplo del proyecto se presupondrá un agente de ventas multicartera, es decir, un vendedor que pueda representar a más de una empresa, hasta un máximo de tres. Esta característica del programa condiciona el hecho de que la estructura de la información debe ser la misma para las empresas de la aplicación. Paralelamente se aplicarán reglas internas de tratamiento de pedidos comunes a todas las empresas. Dichas reglas se detallarán más adelante.

Otra característica importante de la aplicación es que es también multiusuario, lo cual permite que varios usuarios puedan utilizar el mismo software para las mismas empresas. De todos modos es significativo que los beneficios de esta opción se reducen a casos muy concretos.

Los datos de la aplicación tales como la tarifa, clientes, etc., podrán ser actualizados utilizando una base de datos secundaria. Esta base de datos deberá ser modificada por la empresa principal y enviada al representante comercial mediante cualquier solución existente (servidor de ficheros, servidor web, correo electrónico…).

Para el desarrollo de la aplicación se ha utilizado Delphi 2010, de Embarcadero Technologies y se ataca a una base de datos de Ms Access 2007. Sobre el entorno de desarrollo de Delphi se han instalado paquetes de componentes que no se distribuyen con el estándar y que dotarán a nuestra aplicación de potencia, versatilidad y velocidad. Los más importantes son los componentes *VCL* del fabricante DevExpress, los componentes de exportación de datos de FlExCel y los componentes de distribución gratuita RXLib 2.7.5 entre otros.

1.2 Objetivos

El objetivo principal del proyecto, en lo que a funcionalidad se refiere, es el de crear una aplicación para gestionar pedidos basada en una estructura de datos que maximice la flexibilidad y escalabilidad.

Esta estructura se diseñará para poder desarrollar unos procesos básicos bien determinados:

- Selección y consulta de clientes
- Selección y consulta de artículos
- Creación, modificación, eliminación y consulta de pedidos
- Protocolos de comunicación para el envío y recepción de información
- Herramientas de configuración de la propia aplicación
- Sistemas de explotación de datos para análisis estadísticos

A nivel de seguridad será la propia aplicación la que regule las diferentes transacciones sobre la base de datos. El acceso a la base de datos será transparente al usuario y estará dotada de medidas de seguridad como la encriptación de datos. La aplicación está diseñada para que el usuario no pueda acceder ni modificar aquello para lo cual no está autorizado.

Las diferentes vistas de la aplicación deben ser amigables e intuitivas para facilitar un uso cómodo y rápido del software. Diferentes elementos de ayuda estarán disponibles así como completos sistemas de control de errores.

1.3 Estado del arte

La idea del proyecto presentado es una solución totalmente a medida, sin aquellas funcionalidades no necesarias propias de un software de gestión de ventas. Pese a todo se ha rastreado el mercado en busca de soluciones alternativas existentes.

De entre ellas ha habido algunas que se han descartado directamente por sus carencias a nivel funcional, como el hecho de que sólo funcionen para PDA's o aquellas que para la recogida del pedido era necesaria una conexión permanente a internet.

Entre las aplicaciones seleccionadas para su estudio encontramos las siguientes:

Consoltic – Fuerza de ventas

Sus funcionalidades son muy similares a las del proyecto presentado. Es una solución móvil de gestión de pedidos para comerciales que integra comunicación con la central permitiendo así una sincronización on-line. Ofrece herramientas propias de oficina móvil (información de artículos, pedidos, facturación), permite programar rutas a clientes, promoción de artículos, listados de ventas, etc. El precio de la aplicación es según número de licencias adquiridas.

Doscontrol – Movilidad CF Sales

Según las especificaciones del fabricante, "DosControl – Movilidad CF Sales" es un producto de venta comercial estándar y configurable, de rápida implementación y puesta en marcha, con capacidad de gestionar grandes volúmenes de datos y comunicación on-line.

Esta solución, aunque funcionalmente ha resultado ser una opción interesante, ha sido descartada por el hecho de estar únicamente integrada con aplicaciones del mismo fabricante como Sage Logic Class i Sage X3 ERP

OptimaSuite – Módulo de pedidos

Aplicación para tomar nota de pedidos en visita comercial, integrable en PDA, *Smartphone* o PC. Entre sus funcionalidades destacan la posibilidad de poder consultar tarifas, condiciones especiales, histórico de compras y existencias en tiempo real. El pedido se puede enviar por fax o correo electrónico. Puede integrarse con cualquier *ERP* o software *CRM*. Facturación del producto según número de licencias adquiridas.

A pesar de que las especificaciones se ajustan bastante al software que se pretende desarrollar, la interfaz no es muy amigable, y debería ser más intuitiva para poder realizar los pedidos de una manera más rápida.

Tras analizar las diferentes opciones estudiadas, y aunque alguno de los programas, como el de Consoltic, se acerca bastante a lo que se pretende ofrecer, se desarrollará una aplicación a medida implementando todas las particularidades necesarias.

1.3.1 Contexto de aplicación del software

Este software va dirigido a aquellas empresas que emplean procesos manuales en la toma de pedidos en una visita comercial. Un caso típico es el de aquella empresa cuyos representantes comerciales toman nota del pedido en papel y luego lo envían por fax o lo reescriben en un archivo de texto que luego envían por correo electrónico. La problemática radica en el hecho de que antes de enviarse el pedido a la empresa el comercial debe revisar el pedido escrito manualmente. Esto es debido a que, en muchos casos, dada la velocidad con la que se toma el pedido, se tiende a cometer ciertos errores en la codificación de los artículos pedidos o en la redacción de sus descripciones. Asimismo surgen a menudo problemas de lectura del pedido por parte de aquellos clientes que han solicitado copia del mismo.

Una vez llegan los pedidos a la empresa los pedidos deben ser reintroducidos de nuevo en el sistema informático de forma manual, lo cual, aparte de ser un proceso lento, favorece la aparición de nuevos errores.

Por otro lado, toda la información que la empresa envía al comercial (recibos, listados, informes, etc.) se hace en papel y mediante sistemas de mensajería convencional. No hace falta decir que este sistema es lento y costoso.

1.4 Motivaciones

Este proyecto nace de la idea de desarrollar ciertos conocimientos adquiridos en un entorno laboral en combinación con el aprendizaje obtenido durante los años de formación académica.

Hace ya bastantes años realicé un cursillo de Delphi en la EUIS el cual me despertó mucho interés. Con el transcurso de los años me dediqué, a nivel personal y laboral a desarrollar pequeñas aplicaciones que me permitieran enriquecer mis conocimientos de este lenguaje que, hoy por hoy sigue siendo injustamente discriminado en algunos ámbitos del mundo del desarrollo de software.

Ahora, años más tarde, me produce una gran satisfacción poder usar este gran lenguaje de programación para la creación del proyecto de fin de carrera, y con el que pretendo demostrar que las aplicaciones de escritorio aún siguen vivas en una era donde todo parece ser controlado desde un navegador.

1.5 Estructura de la memoria

Si se ha seguido el orden de lectura de esta memoria se habrá visto que la documentación, tras la sección de agradecimientos, se inicia con una introducción al planteamiento genérico del proyecto que se presenta y los objetivos que se persiguen con su desarrollo.

Tras dar a conocer el proyecto a presentar se hará un breve estudio de la viabilidad del mismo, analizándolo tanto funcional como económicamente. De igual forma se detallará la planificación que se debe seguir para su realización, la valoración económica y de los riesgos que implica y finalmente se explicarán las conclusiones a las que se han llegado tras dicho estudio.

Una vez concluido el análisis de la viabilidad del proyecto se explicarán, en el capítulo 3, los fundamentos teóricos que forman la base de conocimiento del proyecto, que servirán para comprender mejor el profundo análisis de las especificaciones que se desarrollará en el capítulo siguiente.

Para finalizar se hará una breve mención del diseño del proyecto, qué pautas se han seguido para su implementación, las pruebas que se han aplicado para comprobar su fiabilidad y qué conclusiones se han extraído de su realización.

Como anexo 1 de la memoria se presenta un pequeño manual de la aplicación y un pequeño diccionario de aquellos términos que sean susceptibles de ser definidos o ampliados. Tales términos serán marcados con la letra *cursiva*.

Durante el desarrollo de la memoria puede que se encuentren sugerencias o ideas de posibles ampliaciones sobre aquello que se está explicando.

En tal caso se indicarán con el símbolo de una bombilla  y encuadradas en un marco de color rojo.

De igual modo se indicarán con el símbolo de un libro  aquellas referencias dignas de mención por su relación, directa o indirecta, con el tema que se está tratando. En esta ocasión el marco será de color azul.

Finalmente, los fragmentos de código de la aplicación que se comenten se mostrarán en un cuadro naranja.

2. Estudio de viabilidad

En esta parte del documento se detallan los diferentes aspectos de la viabilidad del proyecto.

2.1 Introducción

El marco de este estudio abarca desde la fase inicial de requerimientos del proyecto hasta su puesta en marcha, analizando en cada etapa sus posibles implicaciones en la resolución de su viabilidad a nivel económico, técnico, legal y operativo.

2.2 Descripción y objetivos

Tal y como se adelantó en la introducción, se plantea el proyecto como una solución para la mejora en la toma de pedidos de una visita comercial. Se dotará al vendedor de un sistema informático con toda la información necesaria para registrar un pedido, esto es, clientes, artículos, ofertas, existencias, etc.

Una vez finalizada la visita, el comercial podrá enviar el pedido a la empresa a través de internet.

El proyecto tiene como finalidad principal sustituir el lento proceso de tomar nota de pedidos de forma manual y eliminar todos los errores derivados al mismo tiempo que se agiliza el proceso. El hecho de dotar al vendedor de información actualizada y la posibilidad de enviar el pedido in-situ mejorará la calidad del servicio y reducirá el tiempo de entrega del material pedido con el objetivo principal de aumentar la productividad de la fuerza comercial.

Los objetivos principales son:

A) Eliminación de posibles errores en la anotación de pedidos y su envío a la empresa madre.

B) Disponer de la información necesaria para la venta actualizada en tiempo real.

C) Envío de pedidos al momento para minimizar el tiempo de servicio

D) Eliminación del tiempo empleado para introducir los pedidos en el sistema informático de la empresa dado que la información será recibida con un formato específico que permitirá su automatización.


Para introducir los pedidos en el sistema informático de la empresa principal es necesario un software que integre la información generada por el programa presentado en el proyecto.

E) Información de la situación económica del cliente (impagos, riesgo, etc.)

F) Mejora de la oferta comercial. Se proveen estadísticas para poder ofrecer al cliente determinados artículos en función de su perfil de consumo.

G) Posibilidad de ofrecer información telemática al cliente: copias de pedido, informes, etc.

H) Mostrar imágenes de los productos.

I) Venta multicartera.

J) Cumplimiento de sistemas de seguridad y legalidad de datos personales.

K) El proceso de anotación de pedidos debe ser lo más rápido posible.

L) Velocidad de la puesta en marcha del software.


En el caso de tener que reinstalar el software se pueden utilizar herramientas de terceros para crear imágenes de los sistemas completos o archivos para el respaldo del programa.

Clasificación de los objetivos

Tipo de clasificación	Objetivos
Críticos	A, C, D, E, M, L
Prioritarios	B, F, J
Secundarios	G,H, I

2.3 ESTADO DEL ARTE

Una de las principales características del proyecto que la diferencia del resto de competidores que se encuentran en el mercado es su estructura funcional y de datos: una parte importante de las opciones y funcionalidades del programa son configurables, ya sea por el usuario o por un administrador con permisos de acceso a la base de datos.

Se ha partido de la idea que crear un pequeño diccionario de datos. Es decir, información sobre la información. El diseño de los menús y los *scripts* de *transacciones* sobre la información se almacenan en la misma base de datos, permitiendo que, con pequeñas modificaciones, la aplicación funcione sobre cualquier base de datos.

Esta manera de enfocar el diseño de la aplicación ha sido motivada por la idea de que el proyecto presentado debería ofrecer un alto grado de personalización para poder así ajustarse al mayor número posible de clientes potenciales.

Asimismo, en su diseño estándar se ha tratado de hacer énfasis en la facilidad de uso y velocidad de acceso a los diferentes elementos que la forman, intentando evitar problemas que, a juicio del autor, plantean otras soluciones.

2.4 Especificaciones

En este apartado se explicará con sumo detalle aquellas especificaciones que cumple el proyecto. Se dividirán en tres apartados: especificaciones funcionales, especificaciones no funcionales y especificaciones técnicas. Como apreciará el lector, en esta sección encontrará numerosas sugerencias y referencias.

2.4.1 Especificaciones funcionales

El programa deberá permitir la selección rápida del cliente, implementando para la búsqueda por código o por descripción parcial del nombre.

 Podría ser de utilidad realizar la búsqueda por teléfono o e-mail.

En esta pantalla se mostrarán en un *grid* todos los datos disponibles del cliente. Estos datos no se podrán modificar ya que será la empresa que los proporciona quien envíe un

fichero para poder actualizarlos. Si algún dato del cliente debe ser cambiado, se notificará a la empresa utilizando las opciones del programa disponibles para tal fin.

Se dispone de un código especial para aquellos clientes nuevos. Si seleccionamos un el código para el cliente nuevo se hará visible un botón donde podremos informar de todos los datos disponibles del cliente. Estos datos se enviarán en un fichero adjunto junto al pedido que se envía. La información, puede ser cifrada, en caso de ser necesario.

Una vez seleccionado el cliente se mostrará una pantalla con todos los datos necesarios para poder tomar nota del pedido. Para ello podremos buscar el artículo por código, descripción o simplemente haciendo clic en el grid sobre el artículo deseado. Para las búsquedas por descripción se podrán filtrar los resultados del grid de artículos para que sólo muestre aquellos artículos que contengan el texto buscado en su descripción.

Una vez seleccionado el artículo se podrá cambiar la cantidad a pedir, que por defecto será la unidad de caja. También se podrá cambiar la descripción y aplicar descuentos en caso que el artículo lo permita y hasta un máximo permitido. El cambio de descripción no se registrará en el listado de artículos y sólo afectará a la línea del pedido en concreto.

Para cada una de las líneas de pedido se podrá especificar el tipo de venta: sin cargo, abono, obsequio, etc., sin posibilidad de crear líneas del mismo tipo repetidas.

Respecto a la cabecera del pedido, y una vez seleccionado el cliente de nuestra visita, únicamente podremos cambiar el número de pedido (en caso de no haber seleccionado ningún artículo) y la forma de pago, que por defecto será la forma predeterminada de nuestro cliente informada en el sistema informático de la empresa. La fecha del pedido se obtendrá del sistema y no podrá ser modificada.

Al ser un programa multiempresa, se podrá seleccionar, en una misma visita, la empresa de la cual queremos hacer una venta. Los datos de los artículos serán propios de cada empresa, a diferencia de los datos de clientes, que serán comunes a las empresas configuradas. Para una misma visita, podremos vender artículos de cualquier empresa simultáneamente.

También tendremos la posibilidad de realizar la venta de un artículo que no se tenga en tarifa mediante un código destinado a tal efecto. Deberemos introducir la descripción del mismo y la cantidad solicitada. El precio no se podrá modificar ya que será en la empresa donde se marcará dicho precio al ser un artículo fuera de tarifa.

 Como opción del programa se podría permitir introducir un precio a los artículos que no se encuentran en tarifa.

Los artículos que se encuentren en oferta serán mostrados en rojo. El comercial tendrá la posibilidad de aplicar o no la oferta.

Las líneas del pedido en curso se mostrarán en un grid visible en un apartado destinado a tal efecto. En dicho grid se podrán modificar las cantidades de los artículos pedidos o eliminar las líneas de pedido que se desee. Existirá una opción para poder filtrar las líneas de artículos según la empresa.

Como tratamiento especial, se ha diseñado la aplicación para que, según estén informados en la base de datos, sea posible que cada vez que seleccionemos un artículo se añada automáticamente otro en concreto.

 Esto es útil en casos en los que, por ejemplo, un artículo comporta la inclusión de otro artículo accesorio obligatorio o un cargo adicional codificado en forma de artículo.

También tendremos la opción de eliminar o editar pedidos ya grabados en caso de ser necesario. Una vez el pedido esté listo procederemos a su envío telemático. La aplicación nos ofrecerá diversas maneras de enviar el pedido: usando nuestro cliente de correo por defecto, por correo electrónico mediante un cliente de correo interno o mediante protocolo *FTP*.

 El formato del fichero a enviar con la información del pedido podría ser parametrizado según las necesidades de cada empresa. En el anexo 3 se informan de los formatos utilizados en el proyecto.

Junto al pedido se puede adjuntar un report de las visitas que hemos hecho a los clientes. Este report se podrá generar simplemente indicando una empresa, la fecha de inicio de nuestras visitas y la fecha final.

En el proceso de envío se generará un archivo de texto por cada pedido y se enviará a su empresa de destino por separado.

Como ayudas para la venta dispondremos de la posibilidad de mostrar imágenes de los artículos que dispongan de ellas, o realizar consultas estadísticas sobre los datos de ventas de clientes o artículos. Estos datos podrán ser visualizados en forma de gráfico o tabla y exportados a fichero.

Por lo que respecta a la recepción de información de la empresa madre se dispone de un apartado para visualizar las facturas mensuales. Los archivos de facturas de clientes podrán ser visualizados siempre que el sistema operativo lo interprete.

Para la actualización de datos de clientes o tarifas se dispondrá de una base de datos secundaria vinculada a la principal, que recibiremos de la empresa por cualquier sistema de envío de ficheros. Una vez guardado en nuestro equipo, el programa detectará si existen cambios y actualizará los datos en caso necesario.

2.4.2 Especificaciones no funcionales

Las especificaciones no funcionales sobre las que se ha hecho un primer esfuerzo han sido aquellas que afectan al control de errores del flujo de datos de la aplicación. Se ha tratado de minimizar, mediante un diseño coherente, las posibilidades de error de datos.

Dado que la aplicación se implementará en ordenadores de mano, posiblemente poco potentes, el uso de recursos debe estar optimizado para intentar alcanzar el máximo rendimiento posible. Una de las soluciones que se ha implementado ha sido la creación dinámica de los formularios (pantallas) de la aplicación según se necesite, ahorrando memoria en caso de no ser usadas. Asimismo, el diseño de los formularios debe estar enfocado a crear un entorno amigable, rápido y fácil de usar.

También de dispone de herramientas para asegurar la integridad de la base de datos mediante su compactación o reparación.

Como se ha comentado con anterioridad, el software presentado debe ser escalable y susceptible de ser modificado o ampliado de una manera rápida ya que la escalabilidad de la aplicación es un factor a tener muy en cuenta si queremos que nuestra aplicación dure mucho tiempo.

Para poder optimizar los recursos de tráfico de datos a través de internet se ha tratado de minimizar el tamaño de los archivos que se envían. Para poder cumplir con la normativa establecida en la *LOPD* se tendrá la posibilidad, con pequeñas modificaciones en el software, de cifrar los datos que afecten a clientes.

Con el mismo fin los archivos de bases de datos están protegidos mediante contraseña, pese a que existen en la red numerosas utilidades para recuperar contraseñas de bases de datos olvidadas.

 Para recuperar contraseñas de bases de datos Ms Access http://www.xaviware.cl/pwdmdb_es.aspx

2.4.3 Especificaciones técnicas

En lo que al hardware se refiere, la aplicación está pensada para funcionar en cualquier ordenador PC compatible que opere bajo Ms Windows y que pueda ofrecer una resolución mínima de 800x600.

Entrando en detalle, la configuración más básica en la que se ha probado el software ha sido la siguiente:

Procesador Intel Celeron 500 MHz
Memoria RAM de 256 MB.
Sistema Operativo Microsoft Windows 2000 Profesional
Pantalla de 8.4 ‘’, resolución 800x600, 8 bit de profundidad de color

De cualquier manera ya es un poco difícil encontrar en el mercado alguna máquina con prestaciones tan reducidas. Lo recomendado es una máquina con Ms Windows XP o Windows Vista o Windows 7 y un mínimo de 1 GB de RAM. Respecto al procesador, con un Atom® o similar, ya es suficiente para hacer que el programa funcione con holgura.

Aunque se ha tratado de adaptar la aplicación a sistemas con pantalla táctil, la aplicación funciona perfectamente con teclado, en caso de ser necesario.

En el proyecto se puede ver un ejemplo del comportamiento del manejador de gestos en el apartado que muestra las imágenes de los artículos. Con un gesto hacia la izquierda o hacia la derecha las imágenes pasan a la anterior o a la siguiente. Con un movimiento vertical se amplía la imagen al tamaño de pantalla disponible o vuelve a su tamaño original

 Sería recomendable adaptar el software y sacar aún más partido a las nuevas prestaciones que ofrece Windows 7 en lo que a pantalla táctil se refiere mediante su tecnología Windows Touch.

Para el sistema de transferencia de datos es recomendable, aunque no es imprescindible, el contar con el acceso a un servidor de ficheros (servidor FTP), provisto por la empresa y al cual se tenga acceso para almacenar los pedidos diarios o cualquier otra información generada por la aplicación.

Respecto al software los requisitos son muy asequibles. Aparte de la compatibilidad con el sistema operativo antes mencionada, sería también recomendable el tener configurado un cliente de correo electrónico, a poder ser Ms Outlook Express, Ms Outlook, o Microsoft Mail. Una licencia de Ms Excel también es necesaria para poder realizar la exportación de algunos datos y la presentación de informes mediante plantillas.

Para la parte del desarrollo las necesidades del hardware y software son un poco mayores. El software utilizado es Delphi 2010 Architect , de Embarcadero Technologies, aunque también hubiese sido factible realizar el proyecto con las versiones Professional o Enterprise. Se necesitarán también tener instaladas las siguientes colecciones de componentes: Dev Express VCL, Jedi Visual Component Library, RxLib y FlexCel.

Igualmente es necesario disponer de cliente de correo electrónico de Microsoft, una licencia de Ms Excel y el acceso a un servidor FTP para realizar todas las pruebas necesarias.

Respecto a la base de datos, el programa funciona sobre una de Ms Access, aunque con pequeñas modificaciones, y como ya se comentó con anterioridad, se podría utilizar cualquier otra de las que se encuentran disponibles en el mercado.

En la creación de esta memoria se han utilizado el editor de textos Ms Word 2007 y el planificador de proyectos Ms Project.

De la potencia de nuestro entorno hardware de desarrollo dependerá la velocidad del mismo. Delphi 2010 es un buen consumidor de recursos de memoria y procesador. La velocidad de arranque del *IDE* y los tiempos de compilación vendrán marcados principalmente por estos dos recursos, por tanto, cuanto mejores sean mayor será nuestra fluidez de trabajo.

La máquina utilizada en este proyecto ha sido un ordenador portátil Dell Vostro 1220 con procesador T9550 y 4GB de RAM con pantalla de 12’’, aunque una pantalla más grande es recomendable para poder visualizar simultáneamente más paneles de información del desarrollo en Delphi.

2.5 PLANIFICACIÓN

En el diagrama de Gantt que se muestra en la figura 1 se detalla la planificación temporal de las diferentes fases que componen el proyecto. Durante su transcurso y por necesidades obvias inherentes al desarrollo del mismo se han visto alteradas algunas fases del proyecto.

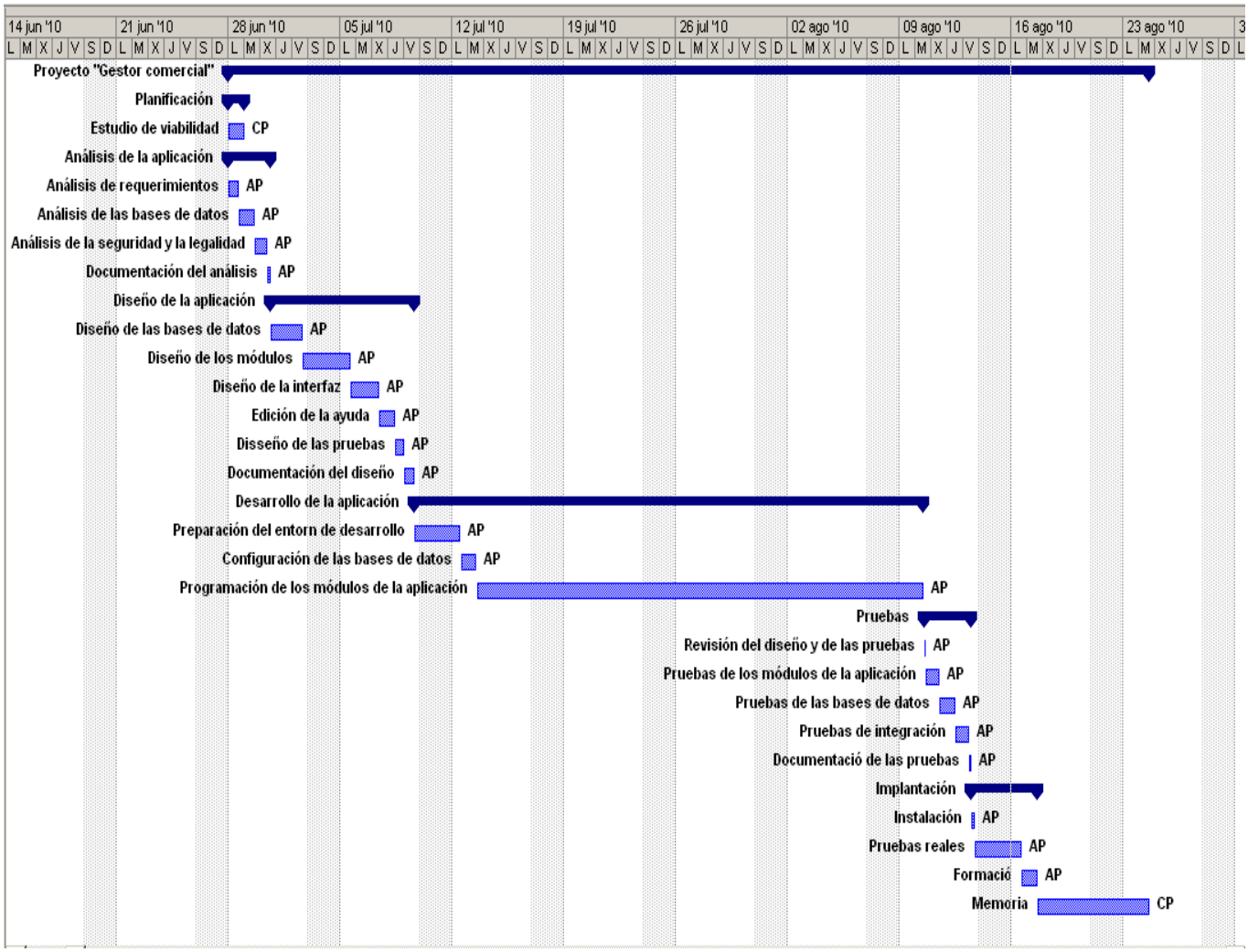


Fig1. Diagrama de Gantt de la Planificación temporal del proyecto.

2.6 VALORACION

Este proyecto no representa una solución comercial real, pero para poder ofrecer una estimación económica aproximada se han tenido en cuenta las siguientes valoraciones sobre los recursos utilizados en un escenario simulado:

Los recursos humanos estarán formados por el coordinador de proyecto y un analista programador. El primero será el encargado de supervisar la evolución del desarrollo del proyecto y la aprobación de la memoria, mientras que el segundo realizará las tareas propias del análisis, programación y pruebas del proyecto.

Los recursos propios de software, tales como las licencias del entorno de desarrollo, licencias de software destinado a pruebas del proyecto y hardware utilizado.

Valoración de recursos			
Recursos humanos	Horas	Precio/hora	Total
Coordinador de proyectos	48	100	4800
Analista programador	293	50	14650
			19450
Recursos propios	Coste	Uso/amortización (meses)	Total
Software de desarrollo	2000	3/36	167
Software de pruebas	780	3/36	65
Hardware	850	3/36	71
			303
		Total	**19753**

Tabla1. Valoración de los recursos del proyecto

En la siguiente figura se detalla el costo económico y en horas de las tareas asignadas al coordinador de proyectos y al analista programador. El tiempo dedicado por el coordinador de proyecto representa aproximadamente un 14% del total, teniendo en cuenta que se ha considerado, según su rol, como autor de la memoria. El analista programador, en cambio, se hace cargo del 86% del proyecto, en lo que a horas se refiere.

	Nombre del recurso	Costo	Trabajo
1	⊟ Coordinador de proyecto	4,800.00 €	48 horas
	Estudio de viabilidad	*800.00 €*	*8 horas*
	Memoria	*4,000.00 €*	*40 horas*
2	⊟ Analista programador	14,650.00 €	293 horas
	Análisis de requerimientos	*300.00 €*	*6 horas*
	Análisis de las bases de datos	*400.00 €*	*8 horas*
	Análisis de la seguridad y la legalidad	*200.00 €*	*4 horas*
	Documentación del análisis	*200.00 €*	*4 horas*
	Diseño de las bases de datos	*800.00 €*	*16 horas*
	Diseño de los módulos	*400.00 €*	*8 horas*
	Diseño de la interfaz	*600.00 €*	*12 horas*
	Edición de la ayuda	*400.00 €*	*8 horas*
	Diseño de las pruebas	*300.00 €*	*6 horas*
	Documentación del diseño	*300.00 €*	*6 horas*
	Preparación del entorno de desarrollo	*300.00 €*	*6 horas*
	Configuración de las bases de datos	*400.00 €*	*8 horas*
	Programación de los módulos de la aplicación	*8,000.00 €*	*160 horas*
	Revisión del diseño y de las pruebas	*50.00 €*	*1 hora*
	Pruebas de los módulos de la aplicación	*300.00 €*	*6 horas*
	Pruebas de las bases de datos	*400.00 €*	*8 horas*
	Pruebas de integración	*300.00 €*	*6 horas*
	Documentación de las pruebas	*100.00 €*	*2 horas*
	Instalación	*200.00 €*	*4 horas*
	Pruebas reales	*300.00 €*	*6 horas*
	Formación	*400.00 €*	*8 horas*

Fig2. Uso y costo de los recursos humanos según etapa del proyecto

2.7 EVALUACIÓN DE LOS RIESGOS

Tras analizar los riesgos, estos se clasifican según su impacto. En cada caso se propondrá un plan de contingencia.

Riesgos con impacto crítico

- Determinación de los requisitos funcionales y no funcionales: Debido a que el principal motivo de la elección del proyecto es la voluntad de tener una herramienta totalmente a medida es de vital importancia que el proyecto se ajuste a las

necesidades especificadas. De ser necesario se planificarán reestructuraciones del programa según las nuevas funcionalidades deseadas.

- Correcta elección de los elementos de desarrollo. Se estudiarán con especial atención las herramientas que se utilizarán en la programación de la aplicación, teniendo en cuenta su compatibilidad y futura continuidad por parte del fabricante. Se centrarán esfuerzos para regirse en la medida de lo posible a los estándares.

- Seguridad y legalidad: Se aplicarán las reglas legales vigentes y se pondrá énfasis en los controles de seguridad. En caso necesario se buscará asesoría al respecto.

- Recursos humanos: Se utilizarán los recursos necesarios para poder ajustarse al máximo al número de horas establecido y evitar así el impacto económico negativo que supone el retraso del desarrollo del proyecto.

Riesgos con impacto marginal

- Aunque el amplio margen en la fecha de entrega a priori no representa un grave problema, se respetará en la medida de lo posible la planificación hecha en el estudio de viabilidad teniendo en cuenta que el cumplimiento de los puntos de impacto crítico pueden afectar a la evolución temporal del proyecto.

Riesgos con impacto catastrófico

- Abandono del proyecto: No se contempla esta posibilidad.

2.8 CONCLUSIONES

Después de analizar las diferentes etapas del estudio de viabilidad se llega a la conclusión de que se puede ofrecer un proyecto con garantías de éxito.

Los beneficios que comporta el uso del software presentado, como la reducción de errores y la eliminación de gastos de los procesos manuales hacen que la informatización del proceso de anotación de pedidos sea económicamente muy viable.

Cabe decir que, pese a no presentar una extrema complejidad técnica se ha hecho un gran esfuerzo en el diseño y planificación del proyecto para poder ofrecer un producto altamente configurable y que garantice la total viabilidad operativa y la plena satisfacción del usuario.

3. FUNDAMENTOS TEÓRICOS

En el presente capítulo se presentan ciertos conocimientos técnicos que han sido considerados necesarios para la realización del Proyecto de Fin de Carrera. No se pretende en ningún modo realizar un desarrollo exhaustivo de todos estos conocimientos, sino ofrecer una introducción necesaria para la comprensión de las herramientas y estructura utilizadas en el proyecto.

3.1 ¿Por qué Delphi 2010?

Como se explicará en el presente capítulo Delphi constituye una completa herramienta para el desarrollo rápido de aplicaciones ideal, entre otros, para proyectos de escritorio.

Tras iniciarme en su programación en la EUIS y usarlo de manera profesional durante años descubrí que con Delphi podía realizar sobradamente aquellos proyectos que me iban surgiendo. Además, existen muchos componentes comunes con otros lenguajes de programación como ASP .NET o C++ que permiten, que sea más sencilla una posible migración de arquitecturas en caso de que se requiera.

Gracias a esto y a la potente conectividad de Delphi con cualquier base de datos se ha considerado que Delphi 2010 y Ms Access 2000 podían ser la respuesta para el desarrollo del proyecto que se ha planteado.

Pese a las grandes ventajas que ofrece Delphi 2010 podemos encontrar en el mercado (pago de licencia de desarrollo) o de forma gratuita en Internet otras herramientas para poder realizar un proyecto de este estilo como es el caso de JAVA , Visual Basic, C# o C++.

3.2 Delphi 2010

Embarcadero ® Delphi ® 2010 se autodefine como uno de los entornos RAD más respetados y ampliamente utilizados hoy en día.

Se estima que hay más de 1.7 millones de programadores que han elegido Delphi sobre otras herramientas debido a que Delphi acelera radicalmente el desarrollo de aplicaciones de escritorio, estación de trabajo, sistemas táctiles, terminales punto de venta, servicios del sistema operativo, kiosk y Web sin sacrificar un ápice de potencia o control sobre la programación.

3.2.1 Introducción

Delphi nació de la mano de Borland en la primavera de 1995. Basado en Object-Pascal, daba soporte a formularios, estaba provisto de un compilador nativo muy rápido y soportaba acceso a numerosas bases de datos.

Con el tiempo se fueron distribuyendo nuevas versiones de Delphi mejoradas: Delphi versión 2, 3, 4, 5, 6, 7, 8, 2005 y 2006, propiedad de Borland (Inprise durante algunos años), 2007 propiedad de Codegear y las versiones 2009 y 2010 de Embarcadero Technologies. A fecha de este escrito, la versión 2010 es la última distribución disponible.

De las versiones con más éxito destacan la versión 7, por su gran robustez y estabilidad y la versión 2010, que según los entendidos es la mejor versión de Delphi que se ha distribuido hasta el momento desde Delphi 7.

En 2001 se creó Kylix, una versión de Delphi para entornos Linux, aunque la falta de programadores que lo utilizaran provocó que tras la versión 3 se dejara de dar soporte.

Existe también una versión gratuita de una herramienta alternativa a Delphi con compilador Free-Pascal, se trata de Lazarus.

 Podrá encontrar más información www.lazarus.freepascal.org

Delphi 2010 cuenta con tres versiones: Architect, Enterprise y Professional. Dependiendo de nuestros requisitos de desarrollo de proyecto deberemos estudiar bien qué versión necesitamos, ya que , mientras la versión Architect ofrece todas las funcionalidades disponibles, las versiones Enterprise y Professional las limitan principalmente en aspectos como el modelado *UML*, conectividad con ciertas bases de datos o DataSnap.

 Las diferencias de precios entre versiones son más que notables. Pudiendo ir desde 395€ la versión de licencia de actualización de Delphi 2010 Professional, hasta los 4495€ de la versión de licencia nueva Delphi 2010 Architect + Mantenimiento.

Las principales características de la versión de Delphi 2010 son las siguientes:

- IDE RAD con diseño rápido de arrastrar y soltar
- Más de 250 controles VCL
- Soporte integrado táctil y de gestos
- dbExpress con soporte para las 9 principales bases de datos
- Middleware DataSnap para n-capas con JSON, REST, HTTP, COM, y XML
- Soporte para depurar aplicaciones con múltiples hilos
- UML / métricas de código y auditorías
- Despliegue en Windows 2000, XP, Vista, y Windows 7 con el mismo código

3.2.2 Componentes

Se ha llegado a decir de Delphi que es un lenguaje orientado a componente. Un componente es un objeto, visual o no, que tiene una serie de propiedades, métodos y eventos, como puede ser un botón, una casilla de validación o un grid de datos.

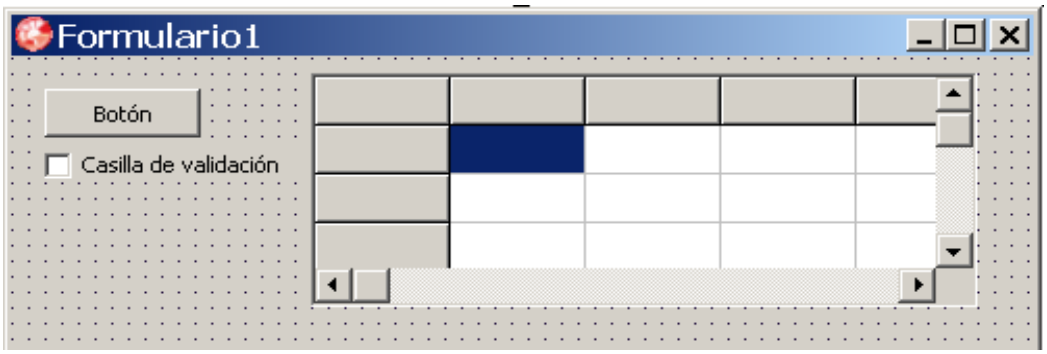


Imagen 3.2.2.1 Ejemplo de componentes visuales sobre un formulario.

La colección de componentes existentes para Delphi es amplísima. Abarca desde componentes estándar básicos hasta complejos componentes orientados a bases de datos, modelado 3D o proyectos de sistemas biométricos.

Posiblemente de la buena elección y conocimiento de nuestros componentes dependerá el éxito del resultado de nuestra aplicación.

En el proyecto, se han instalado varias colecciones de componentes que se explicarán brevemente a continuación:

RxLib Component Library. Es una colección de componentes gratuita que surgió inicialmente para la versión 3 de Delphi. Actualmente está disponible la versión RxLib 2.7.5. Contiene componentes visuales genéricos y algunos orientados a bases de datos. Para el proyecto se ha utilizado un grid que permite dar color a los registros que queramos o insertar otros componentes visuales en celdas, entre otras funciones.

JVCL, Jedi Visual Component Library. Colección de 500 componentes aproximadamente, de distribución al público bajo los términos de la MPL (Mozilla Public License) y mantenida por la Jedi-Community , que se encarga de su desarrollo y adaptación para la compatibilidad de las diferentes versiones de Delphi.

 El sitio web oficial es http://jvcl.delphi-jedi.org/

En Junio de 2002 RxLib se une a JVCL en lo que supuso una de sus más importantes distribuciones.

TMS FlexCel Studio. Es una suite de componentes para las versiones de Delphi 6 hasta Delphi 2010 y Lazarus (también soportado en C++ Builder). Permite crear ficheros nativos en formato Ms Excel (97-2003-XP) sin la necesidad de librerías o *DLL's*. No es gratuito, el coste de una licencia para un único desarrollador es de 75 euros.

TMS FlexCel Studio for VCL/LCL

 Como alternativa se encontraba disponible para versiones anteriores a la 2010 unos componentes realmente buenos del fabricante AfalinaSoft (www.afalinasoft.com). Inicialmente fueron de pago, pero por lo visto ya no se da soporte y en su web anuncian que se ofrecerán como "Open Source" en breve.

Dev Express VCL. Developers Express Inc. es un prestigioso fabricante de todo tipo de componentes, plug-ins, librerías y controles para múltiples plataformas de desarrollo como SilverLight, Visual Studio 2010, Delphi 2010/ C++ Builder o ASP .NET, entre otros.

Por lo que respecta a los componentes creados para Delphi destaca su colección llamada VCL Subscription, que es la más completa. Ofrece componentes para usos tan variados como paneles de navegación, grids y editores de datos, barras de herramientas y menús, control OLAP Data Mining, inspector de objetos, gestor de distribución de componentes, skins, visores Master-Detail, tree-views o distribución de flujo. Destacan entre ellos el componente Quantum Grid, un grid con el que se puede hacer prácticamente de todo y Pivot Grid, con el que podremos combinar y operar con datos de una consulta simplemente con soltar y arrastrar.

La colección Dev Express VCL Subscription ofrece paquetes de componentes VCL y controles ASP .NET. Su precio ronda los $1500

Aunque su producto más premiado es The Quantum Pack, incluido en el VCL Subscription. Ha obtenido diferentes galardones al mejor set de componentes (2002,2003 y 2004) y mejores librerías (2003 y 2004) designado por Delphi Informant Magazine.

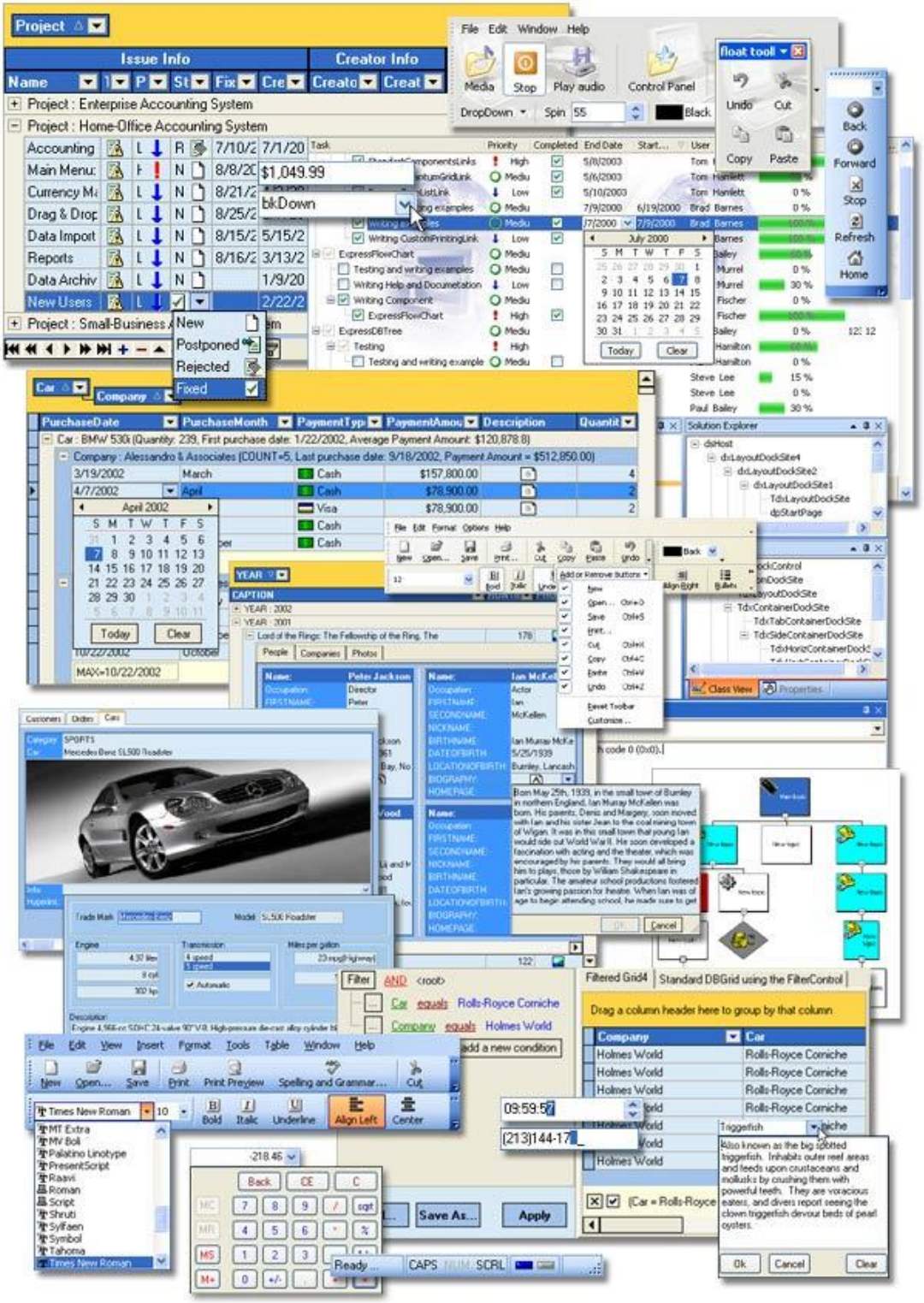


Imagen 3.2.2.2. The QuantumPack™ para Delphi y C++Builder

Aparte de los mencionados existen muchísimos otros de terceros no menos importantes, como ProEssentials (aplicaciones financieras), Altova MapForce 2010 (integración visual de datos) o SyncFusion Essential Studio (bases de datos y reporting).

De entre aquellos que ya se incluyen con la distribución de Delphi 2010 cabe destacar Rave Reports (generación de informes), Ribbon Controls (aspecto Office 2007) o Indy (componentes para internet, FTP, mail, Gopher, Whois, *TCP*, *DNS* …)

 En la web www.componentsource.com encontrará información y venta de todos los componentes comentados y muchos más disponibles para todos los entornos de desarrollo.

Como se habrá podido deducir, el dominio de una plataforma de desarrollo incluye mucho más que aquello que es meramente el lenguaje, estructuras de datos o conocimiento de funciones. Muchos de estos componentes son realmente complejos y requieren una alta formación por parte del programador. Hasta el punto que no es extraño encontrar ofertas de trabajo donde su dominio sea un requisito deseado.

3.2.3 Units, BPL's y DLL's

Las Units son librerías con el código fuente que se compila al crear el ejecutable.

En una aplicación típica de Delphi en una Unit es donde se desarrolla el código asociado a los controles que se hallan en el formulario que los contiene. Pero, de la misma manera que el lenguaje C permite añadir código de funciones con su cláusula "include", Delphi permite usar ("uses") Units con funciones específicas, y que al añadirlas a nuestro proyecto se compilarán con el resto del programa. Eso nos permite crear librerías con funciones propias y reutilizar el código nuestro, o de terceros, extendiendo así diferentes aplicaciones. A este tipo de enlace de código se le conoce como enlace estático.

Se pueden encontrar en Internet multitud de Units (librerías) específicas que nos provean de funciones que necesitamos. El problema es que, como se ha comentado, la Unit se compila con el ejecutable y, según su tamaño, puede llegar a provocar que nuestro ejecutable sea más grande de lo deseado. También podemos encontrarnos con el caso en el que la función o funciones que necesitamos de la librería se usen en momentos muy puntuales y no sea muy eficiente hacer que residan en el ejecutable.

Como alternativa encontramos el uso de las DLL's (Dynamic-Link Library) o BPL's (Borland Package Libraries).

Una biblioteca de enlace dinámico, o DLL , es un archivo con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo Windows. En otros sistemas también existe este concepto, pero no se denomina DLL.

Las ventajas de las DLL's son diversas:

- Reducen el tamaño de los archivos ejecutables: Gran parte del código puede estar almacenado en bibliotecas y no en el propio ejecutable lo que redunda en una mejor modularización.

- Pueden estar compartidas entre varias aplicaciones: Si el código es suficientemente genérico, puede resultar de utilidad para múltiples aplicaciones (por ejemplo, la *MFC* es una biblioteca dinámica con clases genéricas que recubren la API gráfica de Windows y que usan gran parte de las aplicaciones).

- Facilitan la gestión y aprovechamiento de la memoria del sistema: La carga dinámica permite al sistema operativo aplicar algoritmos que mejoren el rendimiento del sistema cuando se carguen estas bibliotecas. Además, al estar compartidas, basta con mantener una copia en memoria para todos los programas que la utilicen.

- Brindan mayor flexibilidad frente a cambios: Es posible mejorar el rendimiento o solucionar pequeños errores distribuyendo únicamente una nueva versión de la biblioteca dinámica. Nuevamente, está corrección o mejora será aprovechada por todas las aplicaciones que compartan la biblioteca.

Pese a las ventajas que suponen las DLL's existen inconvenientes como los problemas de compatibilidad según versiones. Dada la evolución de los sistemas de software, en numerosas ocasiones se desarrollan las DLL's modificándolas de manera que dejan de ser compatibles para ciertos programas que las utilizan, lo que comporta que dichos programas dejen de funcionar o lo hagan incorrectamente. También es posible que al desinstalar un programa se desinstale una DLL que también usan otras aplicaciones. Estos problemas son conocidos como "el infierno de las DLL" (DLL Hell).

Como solución Microsoft ha planteado, entre otras medidas, los scripts de instalación MSI, que contienen información sobre qué librerías son utilizadas por determinadas aplicaciones y qué versión, o la tecnología Side-by-side. En esta última, que constituye un estándar para los ficheros ejecutables en la versión de Windows XP y posteriores, Windows

almacena en la carpeta Windows\WinSxS múltiples versiones de las DLL's y las carga bajo demanda, reduciendo así los conocidos problemas de dependencia de estas librerías.

Las BPL's son también bibliotecas de enlace dinámico, pero diseñadas exclusivamente para su uso con aplicaciones generadas con Delphi. Igual que con las DLL's, el uso de BPL's nos permite colocar porciones de nuestra aplicación en módulos separados que pueden ser compartidos entre varias aplicaciones.

Otra diferencia fundamental es que las BPL's pueden contener Units con código, componentes y formularios e incluso Clases, lo que nos permite escribir código orientado a objeto. Lo que las BPL's permiten, a diferencia de las DLL's, es guardar en ellas todos los componentes que necesita nuestra aplicación.

Tanto las DLL's como las BPL's desempeñan un papel importante en la reducción de código. Como se ha comentado, la principal razón para el uso de estas librerías es la reducción del tamaño de nuestras aplicaciones. Pese a ello debemos tener cuidado con ello, ya que podemos caer en el error de necesitar grandes librerías para nuestros programas. Por ejemplo el principal paquete de componentes visuales, vcl140.bpl, ya ocupa más de 2MB. Sin embargo, si se distribuyen aplicaciones que tengan código en común, puede ser recomendable guardar este código en paquetes. De este modo, en caso de necesitar pequeñas modificaciones, sólo deberán hacerse sobre los módulos afectados.

Pero también existen problemas con las versiones en el uso de BPL's. Si se actualiza un paquete y se compila con una versión diferente de Delphi, también deberemos modificar el ejecutable que usa estos paquetes.

 Cualquier paquete suministrado para su uso por una aplicación debe ser compilado usando la misma versión de Delphi usada para compilar el ejecutable.

 Se recomienda incluir el nombre de la versión del compilador de Delphi en el nombre de la BPL cuando se distribuya (vcl70.bpl, vcl80.bpl, vcl140.bpl…)

En el proyecto no se han utilizado BPL's ni DLL's. En su lugar se han compilado con el ejecutable diferentes Units que contienen funciones de encriptación, transferencia de ficheros, consultas a bases de datos, etc.

3.3 Ms Access

Ms Access 2000 ha sido el motor de bases de datos utilizado para albergar la información que gestiona el proyecto presentado. Pese a sus detractores, ha cumplido con creces su cometido.

3.3.1 Introducción

Ms Access, que vio la luz en 1992, es el gestor de base de datos de Microsoft comúnmente utilizado para pequeños proyectos que no necesitan las prestaciones que puedan requerir otro software como ERP's, CRM's, etc.

Con anterioridad, el mercado de los gestores de bases de datos estaba dominado por Paradox y dBase (Borland) y FoxPro (Fox Technologies). Este último fue adquirido por Microsoft en 1992.

Hasta el lanzamiento del Service Pack 8 para el Jet Database Engine (que es como propiamente se denomina al motor sobre el cual se construye Ms Access y otros programas de Microsoft), este gestor de bases de datos no gozaba de buena reputación entre los programadores y usuarios debido a los eventuales problemas de corrupción de datos y pérdida de información. Pero a partir del punto en el que Ms Access ha ganado en fiabilidad lo ha hecho paralelamente en tamaño de datos y aplicaciones que lo usan, hasta el punto en que es una base de datos utilizada en pequeñas aplicaciones en servidores web.

Ms Access 2000 , como gestor de bases de datos relacional, nos permite realizar todas las operaciones propias de estas, sobradamente conocidas: crear tablas con múltiples tipos de datos, crear índices de búsqueda, relaciones, consultas, modificaciones de tablas y datos…, es decir, todas aquellas sentencias DML (Data Manipulation Language), DDL (Data Definition Language) propias de cualquier *SGBD.* Pero también nos permite crear formularios e informes, utilizar asistentes para consultas, importar y exportar datos, programar macros o vincular bases de datos externas.

Con la llegada de la versión 2010 de MsAccess, se ha potenciado el uso de plantillas para realizar cualquier tarea de una manera prácticamente profesional. Se ha facilitado la creación de macros , la creación de expresiones (gracias a Intellisense) y se ofrecen nuevas herramientas para la publicación nuestros datos en Internet como la integración con los servicios de SharePoint.

3.3.2 Características

Pese a que ya se dispone en el mercado de versiones más modernas de Ms Access, la versión 2000 ofrece todo lo imprescindible para el buen funcionamiento del proyecto presentado. Pero aparte de las funcionalidades utilizadas dispone de otras mejoras respecto a versiones anteriores que se comentan a continuación:

- Integración con la Web: Ms Access le permitirá generar páginas *HTML* con la información que necesite. También podrá integrar elementos Web de Microsoft como componentes de hoja de cálculo, componentes de gráficos dinámicos o componentes de tablas dinámicas.

- Interoperabilidad con SQL Server: Ms Access es capaz de conectarse a Ms SQL Server mediante el estándar OLE DB, para poder así utilizar la aplicación como interfaz para la base de datos SQL Server. Incluso permite realizar ciertas tareas administrativas como replicaciones, copias de seguridad o restauración de datos.

- Mejoras de uso: De entre el gran surtido de mejoras destacan las que afectan a su manejo. Se dispone de una rediseñada ventana de datos y nuevas capacidades de poder arrastrar objetos para exportar datos o crear accesos directos, autocorregir los nombres de objetos de una manera automática, imprimir diagramas de relaciones mediante asistentes, reagrupar los controles para poder manipularlos simultáneamente o asistentes para la creación de tablas y consultas *SQL*. También podrá convertir versiones de bases de datos anteriores.

- Capacidades de programación: Se dispone de soporte Unicode para la internacionalización de aplicaciones. Además incluye una potente versión de Ms Visual Basic para el desarrollo de las mismas.

3.3.3 Acceso a datos. ADO.

Pero de todas las características de la base de datos que se han enumerado en el punto anterior nos centraremos en el acceso a tablas y consultas SQL.

Delphi dispone de diversas herramientas para la conexión a bases de datos. En el proyecto se ha establecido la conexión con la base de datos mediante objetos ADO (ActiveX Data Objects).

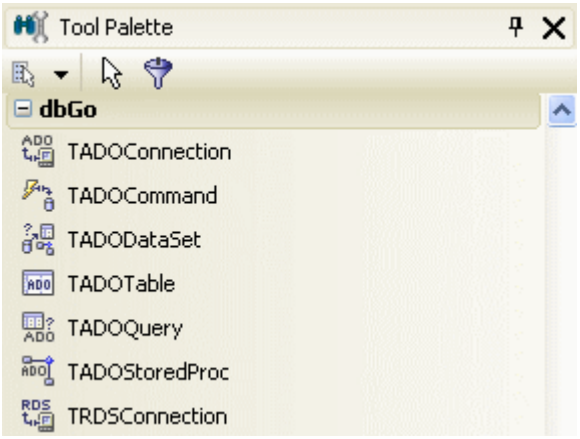


Imagen 3. Componentes ADO de la paleta de herramientas de Delphi 2010.

ADO es una tecnología de Microsoft que crea una interfaz de alto nivel para objetos de datos que, en muchas aplicaciones, ha ocupado el lugar de DAO (Data Access Objects) o RDO (Remote Data Objects). A diferencia de estas últimas, que estaban concebidas para el acceso únicamente a bases de datos relacionales, ADO permite además acceder a múltiples fuentes de datos como páginas web, hojas de cálculo o simplemente ficheros de texto.

Junto con *OLE DB* y *ODBC*, ADO es uno de los componentes principales de la especificación UDA (Universal Data Access) de Microsoft, la cual está diseñada para ofrecer un acceso a datos consistente independientemente de cuál sea su estructura de datos.

En cualquier proyecto Delphi se pueden crear, tanto en tiempo de diseño, como en tiempo de ejecución, todos aquellos objetos ADO que necesitemos. Por ejemplo, desde una misma aplicación podemos tener diferentes objetos ADOConnection que permitan la conexión con diferentes orígenes de datos.

Los componentes ADO, encontrados en la pestaña dbGo de Delphi 2010, tienen las siguientes particularidades:

- ADOConnection: Nos permite establecer la conexión con cualquier fuente de datos. Podemos utilizar los orígenes de datos definidos en el sistema operativo o definir nuestra propia cadena de conexión.

- ADOCommand: Utilizado para el envío de sentencias SQL que no devuelven resultados: Delete, Insert, Alter, Create, etc.

- ADODataset: Con él podremos acceder a resultados de datos obtenidos mediante sentencias Select. Normalmente estos datos se suelen mostrar en un grid mediante el uso de un objeto intermedio de la clase TDatasource.

- ADOTable: Nos da acceso directo a los registros de una tabla de la base de datos.

- ADOQuery: Al igual que los objetos de la clase TADODataset retornan los registros resultantes de consultas al origen de datos. A grandes rasgos su funcionamiento es igual, aunque la clase TADOQuery fue creada para facilitar la migración de acceso a ADO desde BDE (Borland Database Engine).

- ADOStoredProc: Permite acceder y ejecutar a los procedimientos almacenados de la base de datos a la que se halle conectado.

- RDSConnection: Diseñado para utilizar un recordset ADO remotamente a través de protocolos *HTTP*, *HTTPS* o *DCOM*.

Para poder utilizar el resto de componentes tales como ADOdataset, ADOCommand, ADOTable, ADOQuery y ADOStoredProc es necesario indicarle qué conexión (ADOConnection) está usando o, en su defecto, indicarle qué cadena de conexión deberá utilizar.


Se puede consultar www.connectionstrings.com si ha olvidado la nomenclatura básica de cadena de conexión a alguna base de datos.

Dependiendo de la calidad de la conexión a la base de datos o de la complejidad de las operaciones a realizar, como la ejecución de complejas consultas que tarden un tiempo considerable en completarse, debemos tener en cuenta los tiempos máximo permitidos hasta considerar la operación errónea.

Por defecto Delphi establece un tiempo de 30 segundos hasta provocar una excepción en caso de que la tarea no acabe en ese tiempo. Si la máquina sobre la que trabajamos es muy lenta y la operación es grande, es importante no olvidar aumentar ese tiempo, para permitir que se complete satisfactoriamente.

4. Análisis

En el análisis del proyecto se ha estudiado detalladamente la idea global del programa y se ha ido seccionando en módulos menores independientes con una funcionalidad bien determinada.

El análisis está centrado en la idea de la visita comercial y todas las acciones que pueden requerir. Las más importantes son:

- Iniciar nueva visita
- Modificar el pedido en curso
- Abrir un pedido ya grabado
- Eliminar pedidos seleccionados
- Enviar pedidos

En cada una de ellas deberemos tener en cuenta en qué situación estamos y actuar en consecuencia. Por ejemplo, en el caso de abrir un pedido ya grabado de una empresa debemos comprobar si el usuario tiene un pedido en curso y si es así, debemos preguntar si desea grabarlo o no antes de abrir el pedido seleccionado.

Dentro de cada una de ellas continuaremos con la división de tareas para facilitar la modularización del software de manera que al final obtengamos una colección de funciones que realicen tareas bien concretas. Por ejemplo, para poder grabar un pedido necesitamos obtener un nuevo número de pedido y , a su vez, comprobar que ese número de pedido no existe en la base de datos, si es así deberemos obtener uno libre. Toda esta casuística debe quedar bien clara en el momento del análisis, ya que, cuanto más detallados estén los casos de uso más fácil se nos presentará el desarrollo de la aplicación.

Cierto es que, dependiendo de la práctica del desarrollador, existen algunas funcionalidades cuya realización se decide según las necesidades que vayan apareciendo una vez se ha iniciado la programación. Esto nos servirá como experiencia para afrontar nuevos proyectos futuros.

También se deben tener en cuenta otros requisitos durante el análisis. Estos pueden afectar a la aplicación de diferentes tipos de seguridad, protocolos, usuarios, formatos, etc.

4.1 Usuarios y seguridad

La seguridad ha sido un factor muy importante en el diseño del proyecto. Cierto es que el usuario, como mucho, puede acceder a sus propios datos y no a información destinada a terceros. Pese a ello se ha intentado dotar a la aplicación de una estructura de seguridad que garantice mínimamente que el usuario no va a poder alterar la información de la base de datos que no corresponda.

En este proyecto se diferencian dos tipos de usuarios: los usuarios de la aplicación y los de la base de datos.

Los usuarios de la base de datos son dos: usuario estándar y administrador.

El administrador de la base de datos tiene los privilegios totales para poder manipular la base de datos por completo. Normalmente este administrador, en caso de de no ser él mismo, consultará al programador de la aplicación las reglas que debe respetar para poder modificar la estructura de datos o su configuración.

El usuario estándar de la base de datos tiene únicamente los permisos necesarios para poder hacer funcionar la aplicación. Es decir, se han restringido sus capacidades de modificación o creación sobre aquellos elementos de la base de datos (tablas, consultas..) a los cuales no debe tener acceso.

De todas maneras el programa ya hace de primer filtro, ya que no permite al usuario estándar realizar ninguna transacción que no se le haya permitido.

Ms Access provee varios tipos de seguridad para sus bases de datos como la codificación de la base de datos, el acceso por clave o las restricciones de seguridad a nivel de cuentas de usuario o grupo.

Imagen 4.1. Ventana de configuración de cuentas de usuario y de grupo en Ms Access.

Los permisos de los diferentes usuarios o grupos creados se asignan a los diferentes tipos de objetos. Como ejemplo mostrado en la imagen 4.2, los permisos sobre las tablas afectan a la lectura y modificación de su diseño, la lectura, actualización, inserción o eliminación de sus datos y a su administración.

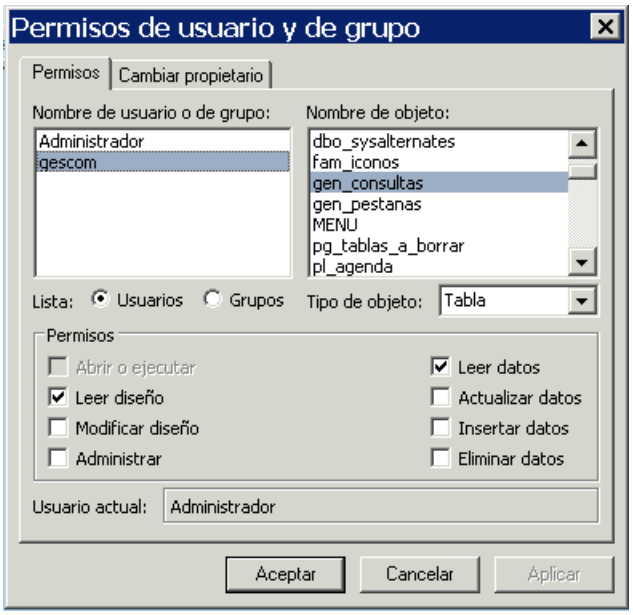


Imagen 4.2. ventana de permisos de usuario y de grupo

La información sobre las cuentas de seguridad debe acompañar a la base de datos se almacena en un archivo de tipo “.MDW”

Respecto a los usuarios de la aplicación, en caso de haber varios, son controlados por el mismo programa y el motivo de su existencia es el de poder facilitar el uso del software a diferentes vendedores y poder diferenciar los pedidos que realizan. De todos modos, es una característica del programa que se presupone de rara utilización y, por tanto, se ha configurado el programa para que la ventana en la que se solicita el usuario de la aplicación sea mostrada opcionalmente según la decisión del administrador. El usuario por defecto que se utiliza una vez se ha arrancado la aplicación es el 01.

Cualquier usuario de la aplicación hace uso de los permisos del usuario estándar de base de datos exceptuando los casos de actualización de datos. En los casos de actualización de datos el usuario de la aplicación.

En los casos en los que se debe actualizar cierta información de la base de datos el programa inicia una sesión en la base de datos secundaria con permisos de administrador. Este proceso se hace de manera totalmente transparente al usuario de la aplicación, es decir, el vendedor. La contraseña de acceso a la base de datos se encuentra encriptada en el fichero de configuración “.ini” que acompaña al ejecutable. En caso de que sea

necesario, el administrador podrá cambiar la clave a la base de datos que contiene las actualizaciones, pero del mismo modo deberá hacer el cambio respectivo en el fichero de configuración.

4.2 LOPD

Otro factor a tener en cuenta es que debemos asegurar que la gestión de los datos personales cumpla con la legalidad vigente, en este caso con la ya famosa LOPD (Ley Orgánica de Protección de Datos).

Según esta ley los datos personales que se utilizan deben ser protegidos mediante una serie de medidas de carácter técnico y organizativo que garanticen su seguridad. En el caso del los datos manejados por el proyecto se deben aplicar medidas de seguridad de nivel básico ya que es una información facilitada por el mismo interesado para legitimar las relaciones comerciales, como el nombre, dirección, teléfono, etc.

Estas medidas de seguridad básicas afectan a la protección de datos mediante claves. Tal y como se detalló en la lista de requerimientos, la base de datos deberá ser protegida mediante una contraseña. Será deber del administrador facilitar los mecanismos necesarios para poder establecer unas reglas de modificación de claves. La estructura de la aplicación permitirá que esto sea posible.

 Puede consultar información adicional en www.agpd.es

4.3 Estructura de la interfaz

La aplicación está estructurada conforme a una aplicación *MDI* (Multiple Document Interface) típica. En ella se construye una ventana (o formulario) principal que albergará aquellas ventanas que se vayan creando en tiempo de diseño según los casos de uso del programa.

Como muestra la imagen 4.3 cuando creamos un formulario podemos seleccionar su tipo, siendo en este caso el formulario principal e tipo fsMDIForm, mientras que aquellos formularios que se muestran dentro de este serán de tipo fsMDIChild, que se crearán en tiempo de ejecución.

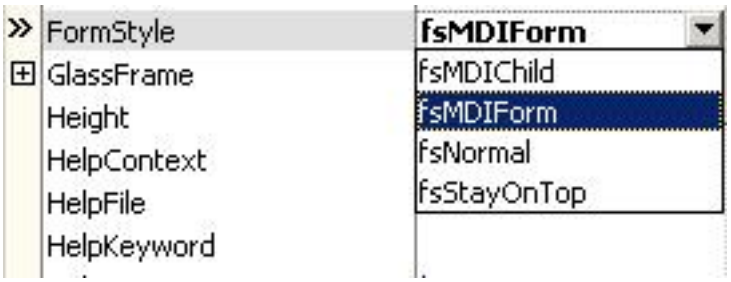


Imagen 4.3.1 Estilos de formularios Delphi.

El formulario principal, que es siempre visible, está formado por 3 partes bien diferenciadas: menú, formulario hijo y barra de estado.

En el menú (ver imagen 4.3.1) se muestran los diferentes apartados de la aplicación en pestañas. Estas pestañas, que son configurables desde la base de datos, muestran los diferentes iconos de las opciones que contienen.

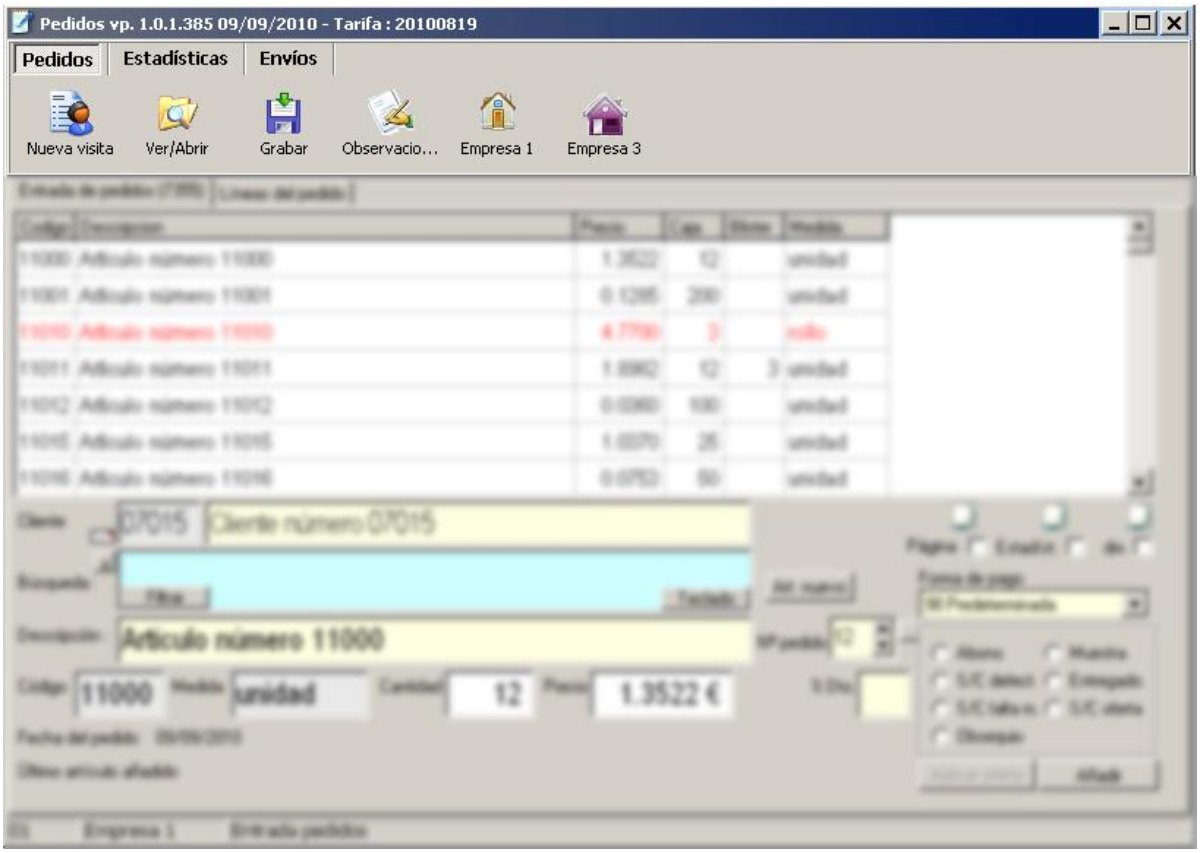


Imagen 4.3.2 Vista del menú de la aplicación

En la parte del título de la ventana se mostrará el nombre del programa (también configurable), su versión y fecha de compilación y la versión de la tarifa de artículos que usa. Este último dato también es opcional y configurable desde base de datos.

El formulario hijo (MDI child) (ver imagen 4.3.2) ocupa el resto de la ventana a excepción de la parte inferior dedicada a la barra de estado. El tamaño del formulario hijo vendrá condicionado por la resolución con la que estemos trabajando. La resolución recomendada es de 800x600.

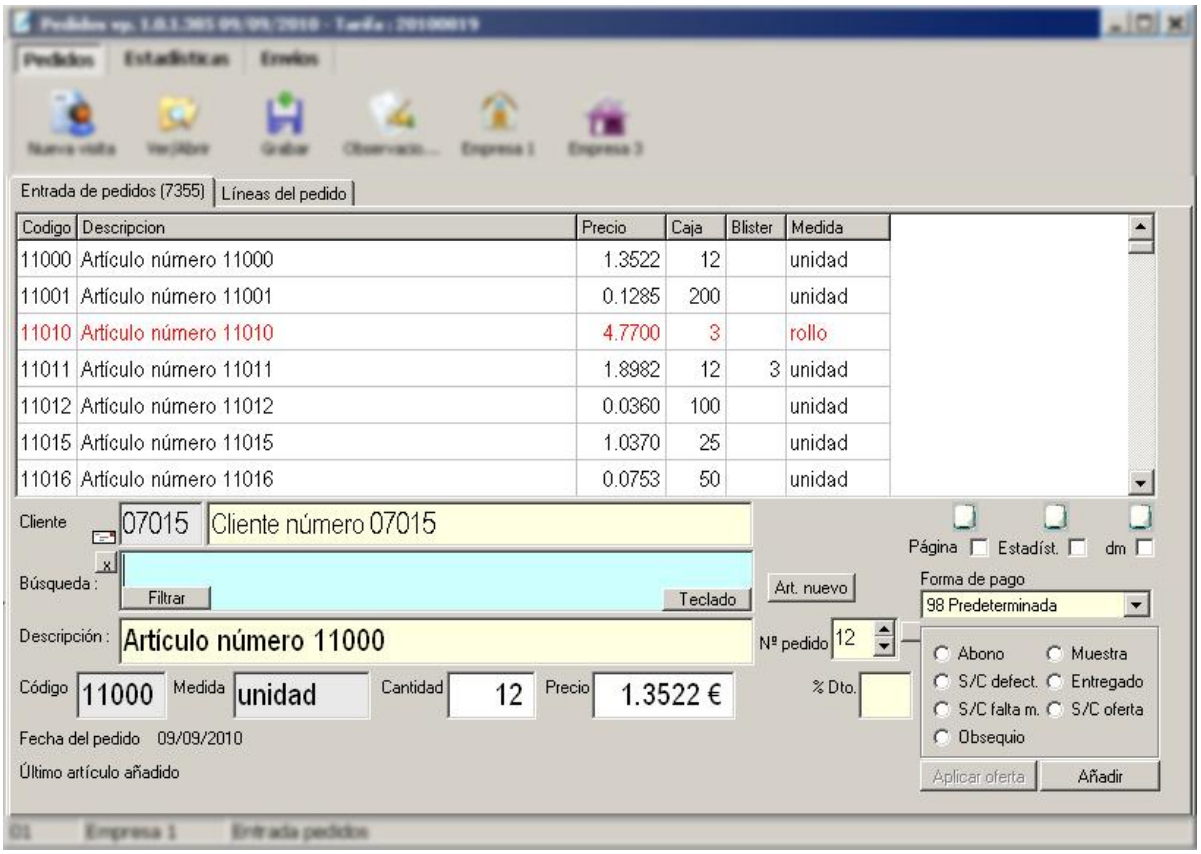


Imagen 4.3.3. Vista del formulario hijo.

La estructura del formulario hijo no sigue ninguna norma en especial y está diseñado según la funcionalidad que ofrece. En el anexo 1 se muestran todos los formularios que aparecen en la aplicación.

Por último la barra de estado se encuentra dividida en tres partes (ver imagen 4.3.3). En la primera se mostrará el código del usuario de la aplicación (normalmente 01), en la segunda nos informará de la empresa con la que estamos trabajando y en la última se muestra el título del formulario con el que estamos trabajando.

Imagen 4.3.4. Vista de la barra de estado.

5. Diseño e implementación

En este tema se documentan todas las decisiones técnicas que se han llevado a cabo para el diseño del proyecto y su implementación con las herramientas comentadas.

Podemos realizar su clasificación en dos apartados principales: las bases de datos utilizadas y los módulos de la aplicación. Para facilitar su explicación se comentarán algunos fragmentos del código utilizado.

5.1 Bases de datos

El proyecto utiliza dos bases de datos. Una de ellas está diseñada específicamente para almacenar toda la información que podamos necesitar para el correcto funcionamiento del programa como pueden ser los datos de pedidos, clientes o artículos y además aquella información del propio software, como configuración de menús, consultas, etc. La segunda base de datos contiene una estructura muy similar a la principal, pero su cometido es completamente diferente. En la segunda base de datos disponemos de la información necesaria para poder actualizar los datos que se encuentran en la primera base de datos.

¿Cómo se realiza esta actualización? La idea es sencilla. En la segunda base de datos se encuentran vinculadas las tablas de la base de datos principal susceptibles de ser modificadas, tanto en su estructura como su información. Es decir, con una conexión a la base de datos secundaria podemos acceder también a los datos de la base de datos principal. De esta manera, cuando tengamos que actualizar, por ejemplo, los datos de artículos o de clientes, el programa se conectará a la segunda base de datos, y ejecutará las transacciones pertinentes para llevar a cabo los cambios requeridos. Esta base de datos, que en realidad es un único fichero, será recibida periódicamente por el usuario con los cambios que su empresa haya realizado.

 Se ha considerado mejor vincular las tablas de la primera base de datos en la segunda por la facilidad que supone realizar cambios en la propia vinculación de tablas en la segunda base de datos.

Las tablas que nos encontramos en las bases de datos son las siguientes:

Base de datos principal (AGDB1 .MDB)

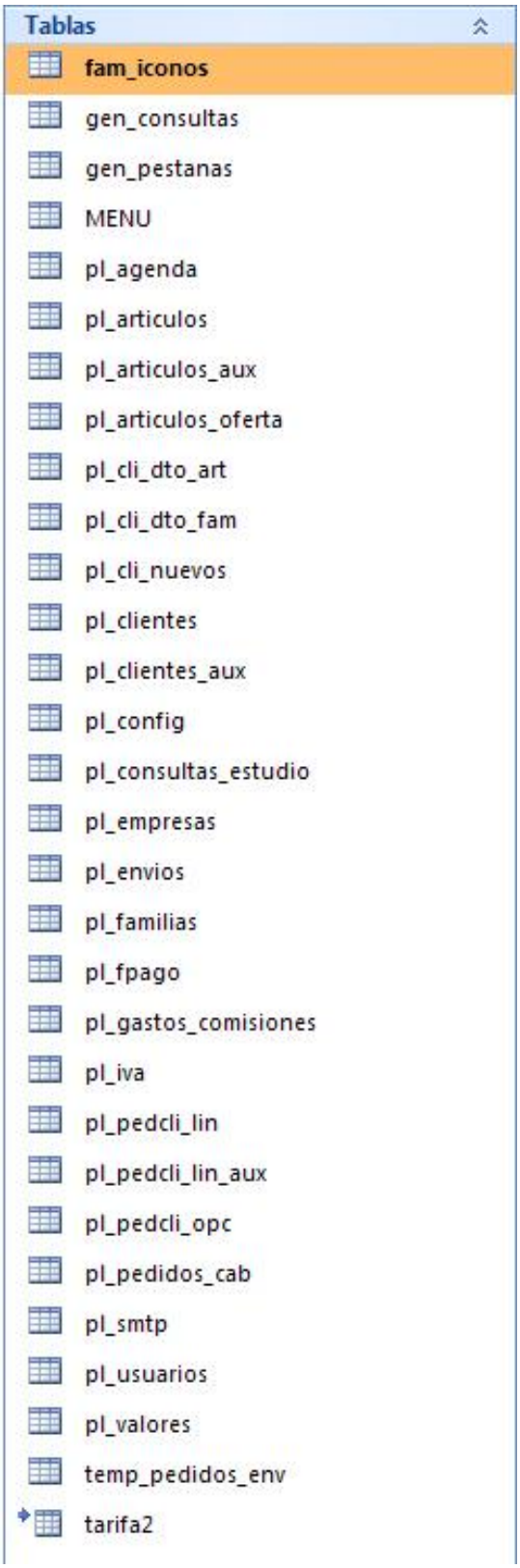


Imagen 5.1. Visión de las tablas de la base de datos AGDB1.MDB

Base de datos secundaria (AGDB2.MDB)

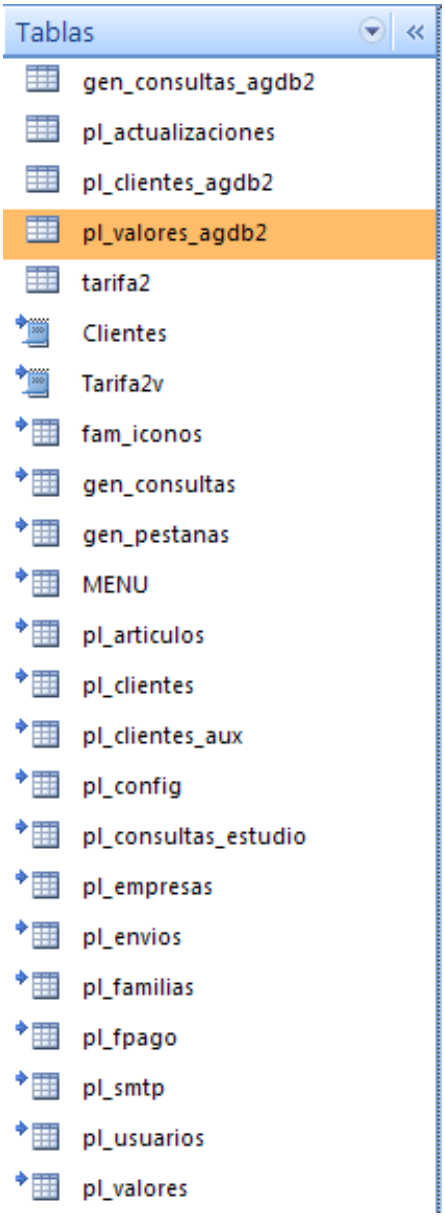


Imagen 5.2. Estructura de tablas de la base de datos de actualizaciones.

Tal y como se observa en la imagen, en la base de datos secundaria tenemos vinculadas las tablas de la base de datos AGDB1.MDB. A su vez podemos tener vinculaciones a fuentes de datos externas como otras bases de datos o archivos de texto, Excel, etc. Al ser la base de datos AGDB2 un archivo que el programa no modifica, podemos realizar sobre ella tantos cambios como el administrador crea conveniente.

Las tablas de AGDB1 están normalizadas para evitar así posibles errores de integridad y de coherencia de datos. En el anexo 2 de esta memoria se puede encontrar un diagrama que muestra las tablas de la base de datos y sus relaciones.

Otra parte importante del desarrollo de la aplicación es el conjunto de consultas SQL que se utilizan. Salvo alguna pequeña excepción, todas ellas se encuentran también almacenadas en la base de datos AGDB1, en la tabla "Gen_Consultas". Esta tabla es de vital importancia porque cualquier pequeño error en cualquiera de las consultas nos proporcionará una información errónea o impedirá que el programa funcione.

Las consultas se encuentran numeradas y desde la aplicación se irán ejecutando según el proceso que se esté realizando. Muchas de ellas están diseñadas para recibir parámetros y dotar de mayor flexibilidad al sistema de transacciones. Para su uso se ha implementado un vector donde estas consultas se irán guardando a medida que se vayan necesitando. De este modo si una consulta se ejecuta más de una vez, se comprueba primero que esté en memoria antes de volver a realizar nuevo un acceso a la base de datos.

5.2 Módulos de la aplicación

La estructura de los módulos de la aplicación estará condicionada por el modo de implementar la gestión de la información a tratar. Para poder ofrecer una visión global de los módulos implementados se enumerarán algunos de los más significativos y se mostrará una pequeña parte del código fuente que tienen asociado.

Muchos de estos módulos están implementados para los diferentes eventos que pueden producirse con el uso de un objeto. Por ejemplo, si sobre un control hacemos un clic, un doble clic o un clic con el botón derecho, etc.

Los módulos más destacados en los que se estructurará el proyecto son:

- Acceso a las bases de datos: A partir de los componentes ADO estableceremos la conexión a las bases de datos y gestionaremos su información.

```
     conn_str:='Provider=Microsoft.Jet.OLEDB.4.0;Password=gescom;User
ID=gescom;Data  Source=c:\gescom\agdb1.mdb;Mode=Share  Deny  None;Extended
Properties="";

     conn_str:=conn_str+';Persist  Security  Info=True;Jet  OLEDB:System
database=c:\gescom\seguridad1.mdw;Jet      OLEDB:Registry      Path="";Jet
OLEDB:Database  Password="";Jet  OLEDB:Engine  Type=5;Jet  OLEDB:Database
Locking Mode=1;';

adoconnection1.ConnectionString:=conn_str;

Ejemplo de conexión a la base de datos principal
```

- Generación dinámica de los formularios del proyecto.

```
procedure TMainForm.estadisticasExecute(Sender: TObject);
var Child: TFrm_estadisticas;
    f:tform;
begin
     f:=Application.findcomponent('frm_estadisticas') as Tform;
     if assigned(f) then
     muestraform(sender,f)
     else
     begin
          Child := TFrm_estadisticas.Create(Application);
          statusbar.panels[2].text:=child.Caption;
     end;
end;
Ejemplo de creación de un formulario en tiempo de ejecución
```

- Estructura de la clase TPedido, que nos permitirá saber con qué pedido estamos trabajando y en qué estado se encuentra

```
type
  TPedidos = class (Tobject)
     pempresa:string[2];
     pempresa_desc:string[25];
     pempresa_num:integer;
     pnumdoc:integer;
     pfecha:tdate;
     pusuario:string[2];
     pestado:integer;
     pcliente:string[5];
     pfpago:integer;
     pimporte:real;
     plineas:integer;
     pobs1:string[80];
     pobs2:string[80];
     pobs3:string[80];
     pobs4:string[80];
end
```

```
type tpedido_activo= class
  pedido:tpedidos;
  public
        procedure cambiaicono(sender:tobject);
        procedure cambiaetiqueta;
  end;
```

- Funciones de extracción de información de la base de datos. Debido a la multitud de sentencias SQL utilizadas y la diversidad de su parametrización se han implementado sobrecarga de funciones para tener en cuenta todos los casos posibles.

```
Function ejecuta_proc (sender:tobject; nombre_proc:string;
parametros:tstrings):boolean;

Function ejecuta_comando (sender:tobject; num_consulta:integer;
parametros:tstrings) :boolean; overload; sobrecarga

function ejecuta_comando (sender:tobject; num_consulta:integer):boolean;
overload; sobrecarga

function obtener_valor_bd(sender:TObject; num_consulta:integer;
parametros:tstrings):variant; overload; sobrecarga

function obtener_valor_bd(sender:TObject; num_consulta:integer;
valor:variant):variant; overload; sobrecarga

function obtener_valor_bd(sender:TObject; num_consulta:integer;
parametros:tstrings; nombre_campo:string; borra_parametros:boolean)
:variant; overload;

sobrecarga
```

- Funciones de envío de datos. Mediante FTP o correo.

```
function comprueba_actualizaciones(sender:tobject):boolean;

function envia_pedidos_ftp(sender:tobject; ftp1:Tidftp; archivos:Tstrings)
:boolean;

function conecta_ftp (sender:tobject;ftp1:Tidftp):boolean;

function get_fecha_fichero (sender:tobject; ftp1:Tidftp; nombrefi:string)
:tdatetime;

function descarga_fichero(sender:tobject; ftp1:Tidftp;  nombrefi:string
):boolean;
function enviaEmail (servidor : string; usuarioy : string; pwdy : string;
asunto:string; mensaje:tstrings; conAutenticacion:boolean; email_origen :
string; nombre_origen : string; email_destino : string; cclist : string;
attachlist:tstrings) : boolean;
```

- Formato de cadenas de texto e información a exportar para adecuarse a las especificaciones requeridas.

```
…
vpl_medida:=obtener_valor_bd(sender,34,parametros,'xmedida',true);
vpl_articulo:=dset_lineas_pedido.fieldbyname('xdescripcion').AsString;

vpl_precio:=formatfloat('0.0000',dset_lineas_pedido.fieldbyname('xprec_ven
ta').Asfloat);
if vpl_precio='0.0000' then vpl_precio:='';
vpl_dto:=dset_lineas_pedido.fieldbyname('xdto_lin').AsString;

vpl_subtotal:=formatfloat('0.0000',dset_lineas_pedido.fieldbyname('ximport
e').asfloat);
```

```
if vpl_subtotal='0.0000' then vpl_subtotal:='';
vpl_tipolin:=dset_lineas_pedido.fieldbyname('xtipo_lin').asstring;
writeln(f,vpl_tipolin+' '+vpl_codigo+' '+format('%5s',[vpl_cantidad])+'
'+format('%-9s',[vpl_medida])+' '+format('%-40s',[vpl_articulo])+'
'+format('%10s',[vpl_precio])+' '+format('%2s',[vpl_dto])+'
'+format('%10s',[vpl_subtotal]));
writeln(f,'');
dset_lineas_pedido.Next;
```

Ejemplo de código usado en la exportación del pedido a fichero de texto

- Mantenimiento de la base de datos. No hay que olvidar que al trabajar con una base de datos se debe asegurar su integridad, por ejemplo con funciones para su compactación y reparación. En el caso de MS Access debemos usar las librerías especiales como ADO_TLB, ADODB_TLB, JRO_TLB y ADO_DB.

```
function Compactayrepara(bdorigen:string; bdfin:string) : Boolean;
var
       oJetEng   : JetEngine;
       soldmdb:string;
       snewmdb:string;
begin
       sOldMDB := bdorigen;
       sNewMDB := bdfin;

       try
          oJetEng := CoJetEngine.Create;
          oJetEng.CompactDatabase(sOldMDB, sNewMDB);
          oJetEng := Nil;
          Result  := True;
       except
          oJetEng := Nil;
          Result  := False;
       end;
end;
```

Pese al tiempo destinado al análisis, es habitual que en la fase del diseño, y sobre todo en la fase de implementación, nos encontremos en situaciones que no habíamos previsto. En estos casos se debe volver a analizar el proceso o rutina tantas veces como sea necesario para que su implementación sea lo más eficiente posible.

Es importante conocer también las fuentes de información de las que disponemos a la hora de afrontarnos a un problema. Muchas de las situaciones conflictivas que nos aparecen durante el proceso de análisis e implementación seguramente tienen una base documentada por otros programadores y podremos encontrar solución en algún rincón de Internet.

Por esto último, hay que ser conscientes de nuestros conocimientos y de las habilidades sobre las herramientas de las que disponemos. Puede que durante el análisis enfoquemos una solución que, sobre el papel nos parece fácil de implementar, pero que a la hora de la verdad nos conlleva mucho más tiempo del que habíamos planificado. Para situaciones conflictivas es recomendable siempre disponer de un plan alternativo.

5.3 Control de datos

Para poder asegurar una coherencia de toda la información con la que se trabaja se han implementado diversos sistemas de control de datos en función del componente o situación donde se esté tratando.

Ciertos componentes permiten la parametrización de la información según máscaras lo cual nos permite ya en tiempo de diseño decidir cuáles son los tipos de datos que van a ser aceptados o no.

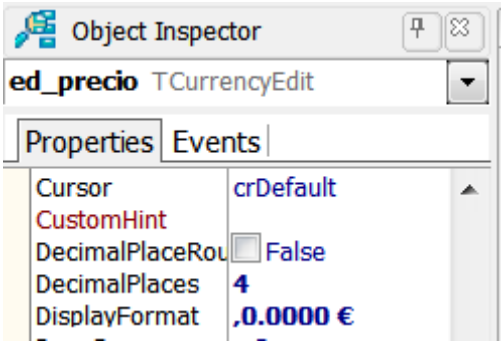


Imagen 5.3. Ejemplo de formateo de los datos de entrada en las propiedades de un control TCurrencyEdit.

Al igual que el ejemplo anterior existen multitud de criterios de control sobre los datos, como la longitud o valor máximo o mínimo, valores por defecto, etc.

Los formatos también podrán controlarse en tiempo de ejecución, con lo que dotamos de mayor flexibilidad al programa por si queremos realizar algún cambio guardando los formatos en la base de datos.

```
TFloatField(frm_entrada_pedidos.rxgrid_tarifa.DataSource.DataSet.fieldByNa
me('precio')).DisplayFormat := '0.0000';
```

A nivel de flujo de datos interno se controlan con qué tipo de datos, e incluso componentes estamos operando

```
if frm_detalle_cli_nuevo.Components[a] is tedit then
(frm_detalle_cli_nuevo.Components[a] as tedit).Clear;
```

Las conversiones entre tipos de datos son habituales para asegurar que no existirán problemas con la información que tratemos.

Formatos de fecha:

```
pact.pedido.pfecha:=strtodate(fecha);
```

Conversión de cadenas a numéricos:

```
if strtoint( trim( copy( frm_entrada_pedidos.cb_fpago.items[a],1, 2))) =
pact.pedido.pfpago then…
```

También se comprueba que el uso de los datos sea lógico

```
if dtp_fec_fin.date<dtp_fec_ini.date then showmessage('Error en la
selección de fechas')
```

En caso de que la entrada de datos no sea la adecuada se impide continuar hasta que el dato introducido es incorrecto.

```
if length(ed_cliente_buscado.Text)=5 then
begin
 dbgrid1.DataSource.DataSet.Locate('xcliente_id', ed_cliente_buscado.Text,
[]);
 if ( ed_cliente_buscado.text <> dset_busca_cli.FieldByName( 'xcliente_id'
).asstring ) then
           begin
lb_cliente_no_encontrado.Visible:=true;
             bbtn_aceptar.Enabled:=false;
           end
           else
           begin
       lb_cliente_no_encontrado.Visible:=false;
bbtn_aceptar.Enabled:=true;
           end;
end;
```

En este caso se inhabilita el botón de “Aceptar” que da paso a la siguiente pantalla.

6. Pruebas

Durante el desarrollo de la aplicación se ha ido verificando el buen funcionamiento de sus funciones a medida que se iba desarrollando.

6.1 Control de errores

A nivel de programación de código se han implementado los siguientes puntos.

- Funciones lógicas que retornan el éxito o no de la operación y tratan el control de flujo.
- Uso de instrucciones que capturan y gestionan las excepciones de error p.e. try... except... end, try… finally… end.
- Uso de máscaras de datos para evitar entrada de información errónea.
- Diseño relacional de la base de datos con restricciones de clave primaria y clave externa.
- Uso ocasional de tipos de datos *variant* para variables de múltiples tipos.
- Mensajes de error para informar de una operativa incorrecta.

Todas las opciones de la aplicación han sido testadas numerosas ocasiones y el flujo de datos ha sido correcto en cada una de ellas.

6.2 Capacidad de la base de datos

Para comprobar la capacidad de gestión de cantidades importantes de información se ha desarrollado un pequeño software que permite la creación aleatoria de pedidos según unos parámetros de entrada.

En este programa, tal y como indica la Imagen 6.2.1, aparecen los datos de clientes y artículos disponibles en una de las 3 empresas posibles.

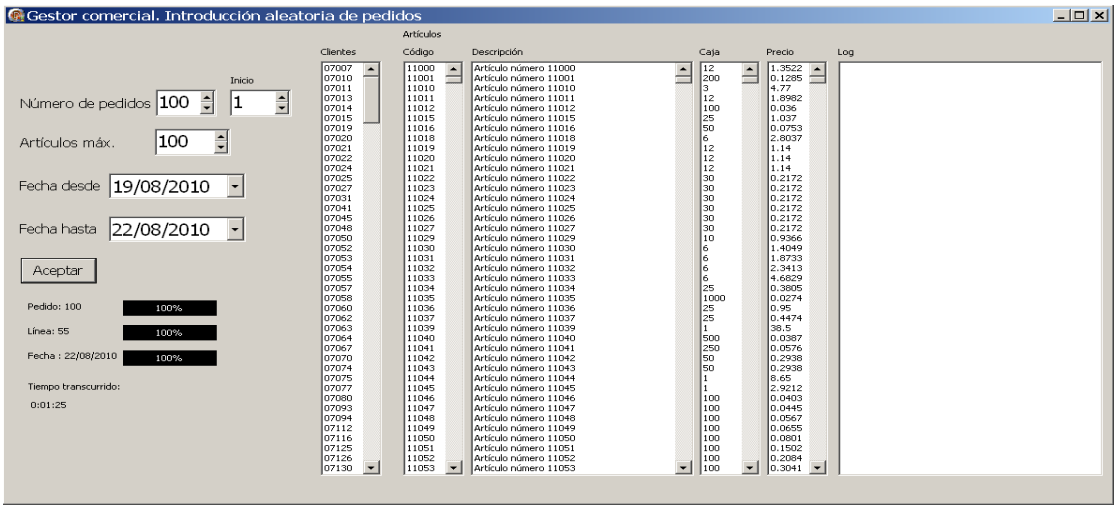


Imagen 6.2.1 Test de introducción automática de pedidos.

Se solicita que se introduzca un número de pedidos diario, como máximo 500, a partir de un número de pedido inicial.

Se debe informar también del número de artículos máximo que contendrá nuestro pedido. Este se generará aleatoriamente entre 1 y el máximo seleccionado. El cliente del pedido también se seleccionará al azar. Como mucho se permitirán 500 líneas por pedido. En la simulación se supondrá que la cantidad que se ha solicitado de cada artículo es igual a la unidad de la caja (ver imagen 4)

Finalmente el programa generará con estos datos tantos pedidos como se haya indicado para cada uno de los días entre “Fecha desde” y “Fecha hasta”.

Durante los tests no se ha producido ningún problema. En una de las pruebas se han llegado a insertar más de medio millón de líneas y la base de datos ha respondido perfectamente a semejante cantidad de información introducida.

Las pruebas se han realizado en dos ordenadores diferentes. Uno de ellos es un Asus EeePc T101Mt con procesador Atom 450 y 1GB de RAM, el otro es un PC de sobremesa con procesador Intel Quad Core Q6600 con 4GB de RAM. En el primero el test ha durado casi 6 minutos, mientras que en el segundo no ha alcanzado los 3 minutos. Esto nos hace reflexionar sobre la posibilidad de que la incorporación masiva de de datos externos sea un problema en máquinas lentas si no se optimiza su programación.

Una vez realizados los tests ya disponemos de información aleatoria para poder continuar con las pruebas de envío de pedidos, generación de informes, generación de gráficos y cálculo de acumulados.

6.3 Envío de datos

Teniendo ya un gran volumen de información ya podemos realizar las pruebas de envío de datos.

Para esta fase de las pruebas se enviarán por el gestor de correo interno a la aplicación y el correo externo un mensaje con los 500 ficheros adjuntos.

Las pruebas se han hecho con dos clientes de correo Ms Outlook 2007 y Mozilla Thunderbird. Mientras en Ms Outlook no había problema para adjuntar los 500 archivos de texto generados, al intentar crear el mismo mail usando Mozilla se producía un error MAPI. Haciendo más pruebas se ha detectado que el número máximo de ficheros adjuntos que ha aceptado Mozilla sin generar error ha sido de 100. Sin embargo, si se crea con Mozilla ThunderBird un mail manualmente y se adjuntan más de 100 ficheros no hay problema.

Respecto al envío de mensajes por FTP no se han encontrado problemas en ninguna de las pruebas realizadas.

6.4 Explotación de datos

Por último se realizan las pruebas de generación de tablas dinámicas y generación de gráficos.

Se observa que la cantidad de datos a manejar es excesiva en consultas que devuelven cientos de miles de registros. Los cálculos se hacen pesados y lentos, pero funcionan. En las pruebas realizadas con decenas de miles de registros se aligera considerablemente el rendimiento y se puede ofrecer una información más rápida y cómoda de consultar. Como resultado de esta prueba se decide que lo ideal sería programar más consultas que devuelvan paquetes más pequeños.

También se modifica la generación de gráficos y se implementa un sistema de reconocimiento de parámetros y dotarlo así de mayor flexibilidad de uso. De esta manera será el mismo usuario quien personalice la consulta según sus necesidades

7 Mejoras y extensiones de la aplicación

Una vez finalizado un proyecto, posiblemente se puede quedar uno con la sensación de que, con lo que ha aprendido, retomaría el proyecto introduciendo ciertos cambios. Eso es exactamente lo que me ha sucedido a mí.

Los cambios que realizaría y los que se me ocurren como alternativa válida a lo que he hecho son:

- Implementación de librerías más optimizadas según su uso. Uso de BPL's en vez de Units. Almacenamiento de los formularios en BPL's para el ahorro de memoria.
- Uso de la programación mediante hilos para procesos que pueden ejecutarse en background y aumentar así la velocidad del programa.
- Mayor búsqueda de pequeñas aplicaciones externas que pudieran ser llamadas desde la aplicación.
- Mayor configuración del programa almacenada en la base de datos.
- Conexión con bases de datos remotas para el envío y recepción de información.
- Incorporación del tratamiento de ficheros con formato XML.
- Posibilidad de tener una aplicación multiidioma
- Potenciar la generación de informes.
- Realización de pruebas para la exportación de datos a software del paquete Open Office.

8 Conclusiones

Tras terminar el Proyecto de Fin de Carrera y pensar en las horas que ha necesitado para llevarse a cabo, uno se plantea lo importantes que son cada una de las fases que lo componen: desde la primera reunión con el tutor del proyecto hasta la preparación para presentarlo ante el tribunal.

Normalmente, si se ha trabajado en algún sector de la informática, no se suele afrontar un proyecto en todas sus fases tal y como se ha hecho. Aquí radica, a mi parecer, el principal beneficio del Proyecto de Fin de Carrera.

En mi caso, proponía un proyecto del cual conocía el lenguaje a usar y el planteamiento principal para empezar a desarrollarlo, pudiéndome extender en detalles de mayor complejidad, que no había tenido la ocasión de conocer con anterioridad.

Sin duda alguna puedo decir que, al término de este trabajo, me siento orgulloso y a la vez contento de todo lo que el proyecto, con el que he pasado algunos momentos duros, me ha permitido aprender.

8.1 Desviaciones

Durante el transcurso del proyecto, han habido fases donde el desarrollo del mismo ha sufrido ciertas variaciones. Pese a haber realizado una planificación de manera relativamente minuciosa, las desviaciones más importantes han surgido a la hora de diseñar el proyecto y su posterior programación.

Esto ha sido debido a que durante la programación se han valorado pequeños cambios sobre el diseño del software que aportaran ciertas mejoras y que no se habían tenido en cuenta durante la fase de planificación.

También durante la fase de pruebas se ha decidido crear una pequeña aplicación para poder dar de alta pedidos en el programa, de manera aleatoria.

Esto ha facilitado el poder disponer de información que se ha utilizado para los tests que se han ido aplicando al proyecto, aunque el tiempo dedicado a las pruebas se ha incrementado notablemente.

En la tabla que se muestra a continuación se detallan las alteraciones que ha sufrido el proyecto en su desarrollo respecto a la planificación que se había propuesto inicialmente.

Desviaciones temporales de las tareas del proyecto.

Tareas de la aplicación	Horas planificadas	Horas reales
Planificación	8	16
Análisis	22	22
Diseño	56	64
Desarrollo	174	174
Pruebas	23	31
Implantación	18	18
Documentación	40	40
Total	341	367

9 Bibliografía

Han sido numerosas las páginas web visitadas para obtener información durante la realización de este proyecto. Por suerte o por desgracia debo decir que no he consultado ni libros ni revistas especializadas, todas las fuentes consultadas se han encontrado en Internet.

Las más importantes han sido:

www.clubdelphi.com

www.delphibasics.co.uk

www.marcocantu.com

delphi.about.com

www.torry.net

www.devexpress.com

es.wikipedia.org

www.experts-exchange.com

ANEXO 1 – MANUAL DEL USUARIO

Este manual está estructurado según el orden lógico, o más probable, que se pueda seguir durante la elaboración del pedido. En cada apartado se explicarán las diferentes opciones que existen y qué funcionamiento tienen.

- Selección de usuario

En el caso de tener varios usuarios para la aplicación se mostrará una pantalla para poder elegir qué usuario va a usar el programa. El usuario del programa no es el mismo que el usuario de la base de datos.

Un Identificador comercial puede tener uno o varios usuarios. Lo habitual sería tener un usuario por identificador.

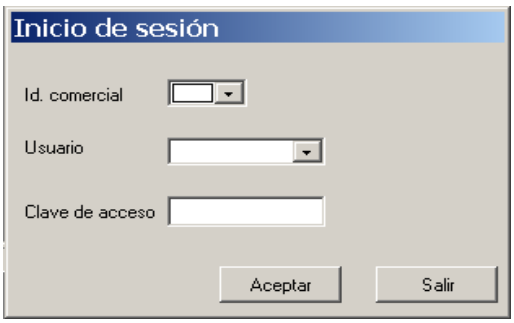


Imagen A1. Inicio de sesión el la aplicación

- Selección del cliente

Una vez nos hayamos identificado se nos mostrará la lista de nuestros clientes. En ella podremos buscarlos o por código o descripción. Para facilitar su escritura en pantallas táctiles se ha incluido un sencillo teclado en pantalla.

En caso de visitar a un cliente nuevo se utilizará el código 99999. Posteriormente se podrán informar los datos sobre el cliente.

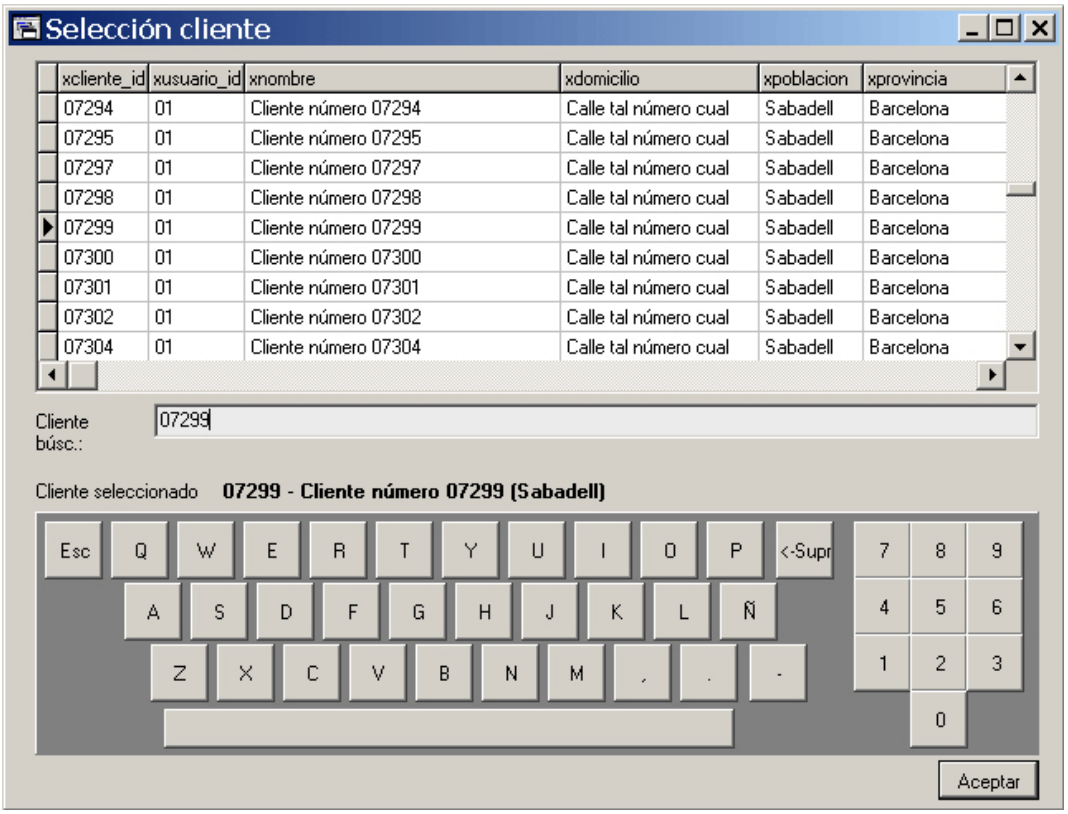


Imagen A2. Selección del cliente

Tras haber seleccionado el cliente y dada la importancia que tiene la comunicación con él, se podrá configurar el programa para que solicite el mail del cliente en caso de que no esté nformado. Esta información utilizará una línea de observaciones del pedido que se realice.

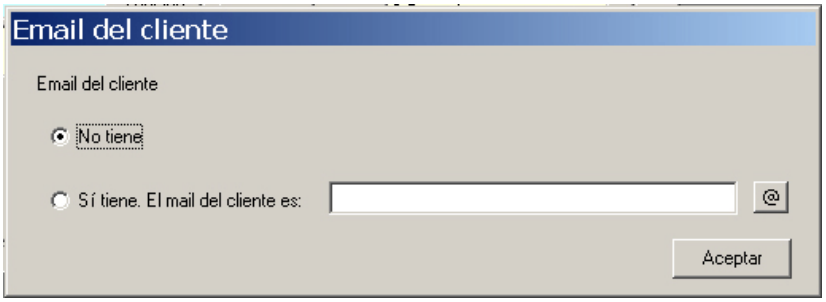


Imagen A3. Ventana de información del mail del cliente.

- Entrada de pedidos

La ventana de la entrada de pedidos es donde se muestra toda la información de los artículos y las opciones necesarias para realizar un pedido.

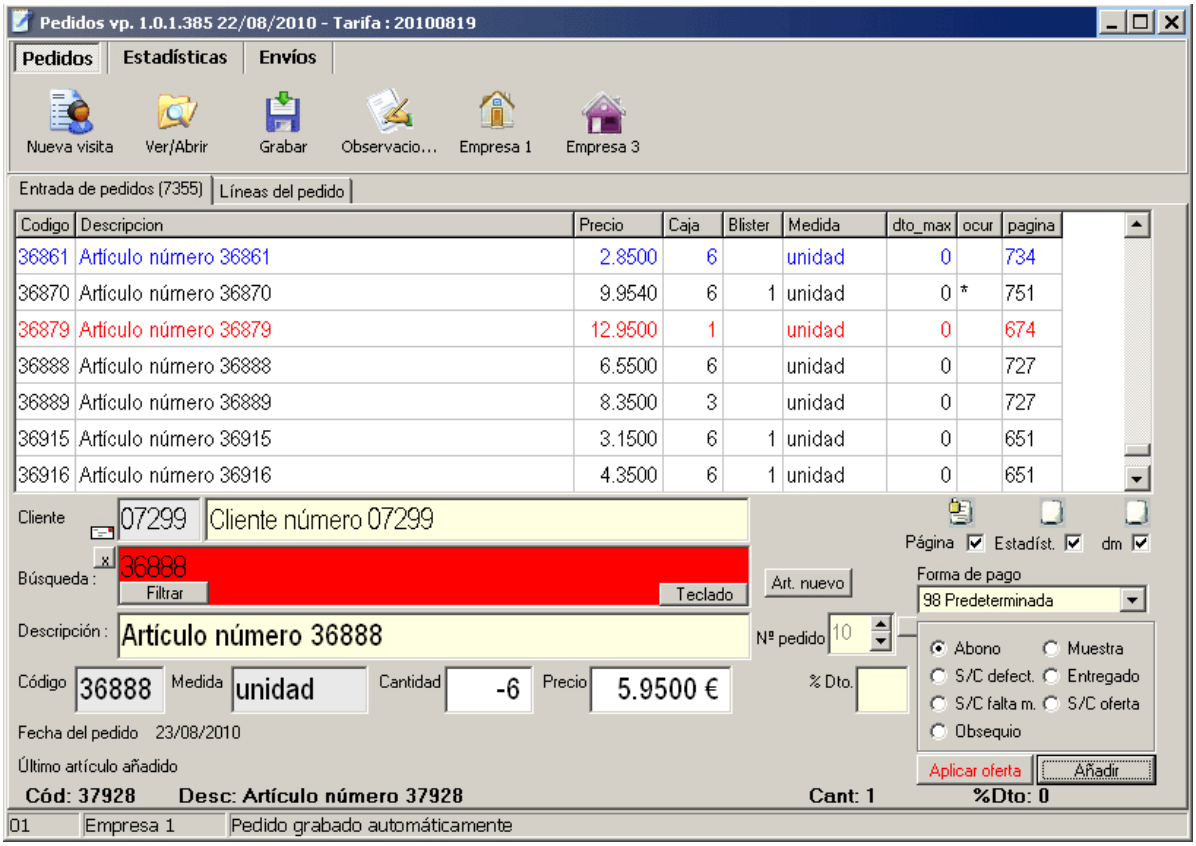


Imagen A4. Formulario principal de entrada de pedidos.

En la parte superior de la pantalla, se pueden ver los diferentes apartados de los que consta la aplicación: la gestión de pedidos, estadísticas y envíos. Justo debajo aparecen aquellas opciones de las que dispone.

El formulario contiene dos pestañas fijas: una para la confección del pedido (Entrada de pedidos) y otra donde se pueden ver las líneas del pedido que vamos creando. Existe una tercera que se activa en caso de que el artículo disponga de alguna imagen. El número que hay en la pestaña de entrada de pedidos indica el número de referencias de nuestra tarifa.

En la primera, se muestra el grid donde aparecerán todos los artículos disponibles para su venta. Los artículos pueden ser de cuatro colores diferentes: negro, azul, rojo o verde. Cada uno tendrá el significado que le queramos dar.

Podemos ordenar los artículos por cualquiera de los campos que se ven en el grid, simplemente haciendo clic en en nombre de la columna.

Existen tres columnas cuya visibilidad es opcional: Página, estadísticas, y descuento máximo. La primera indica el número de página del catálogo donde aparece el artículo, la segunda columna muestra un '*' en aquellos artículos que han sido comprados con

anterioridad por el cliente del pedido en curso, y la tercera muestra la información sobre el valor del descuento máximo que se le permite hacer al artículo.

También podremos ver el código de cliente seleccionado y su nombre. Justo a su izquierda hay un icono de un sobre sobre el que, haciendo clic, nos informará del mail del cliente. Si el cliente es nuevo se visualiza un botón "Detalle" que da acceso al formulario donde introduciremos los datos del cliente nuevo. Esta información se guardará y se adjuntará automáticamente al pedido enviado como un fichero de texto.

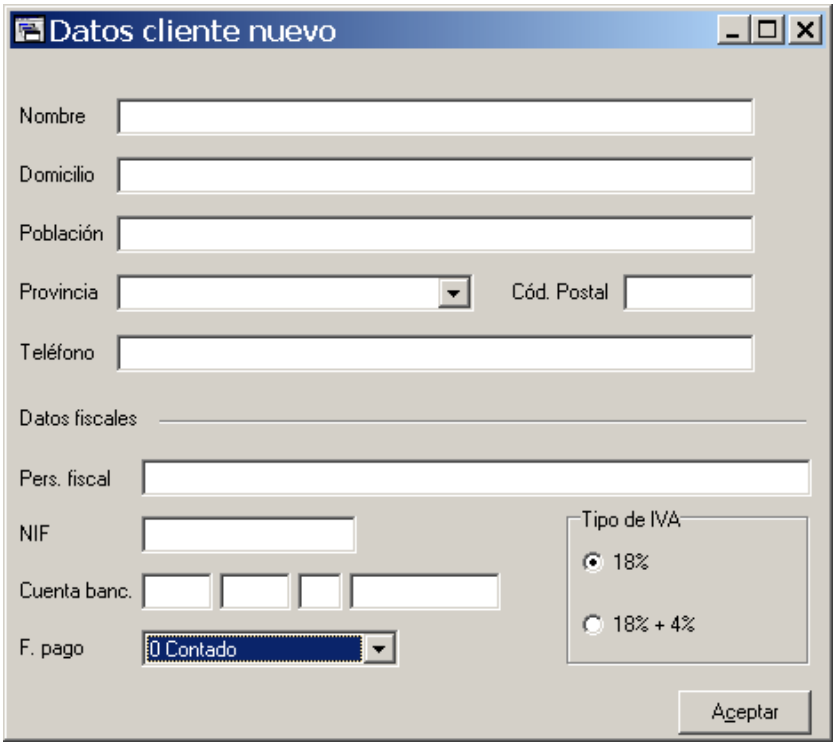


Imagen A5. Formulario de introducción de datos del cliente nuevo.

Respecto a la cabecera, se puede cambiar la forma de pago y el número de pedido. Este último siempre y cuando no se haya añadido ya alguna línea.

La búsqueda de artículos es muy sencilla. Se puede buscar por código, descripción o filtrar según una cadena de texto. También se ofrece la posibilidad de mostrar un pqueño teclado en pantalla.

Por cada una de las líneas que insertemos se podrá elegir su tipo, si es un abono, una línea sin cargo, etc. No se permiten insertar dos líneas del mismo artículo y tipo.

Según las empresas configuradas en el sistema se dispone de un icono por cada una de ellas que nos indicará si el pedido está en alguno de estos tres estados: nuevo, modificado, grabado. (ver imagen A6)

Fig A6. Imágenes de los estados del pedido

Los artículos que se encuentren en oferta se indicarán iluminando el cuadro de "Búsqueda" de color rojo. Para aplicar la oferta existe un botón que lo permite.

El precio del artículo no se puede modificar aunque sí la descripción, cantidad y descuento (siempre que el descuento máxmimo lo permita).

Por último, cada vez que añadamos un artículo se indicará al pie del formulario como recordatorio.

En la pestaña de líneas de pedido se muestran las líneas que se van añadiendo.

En este apartado también se puede cambiar la cantidad de la línea, visualizar el importe de lo que llevamos y o filtrar según la empresa que queramos.

Usando el botón derecho sobre el grid de artículos, accederemos al histórico. En él se muestra una pantalla con el histórico de las ventas del artículo marcado, tanto del cliente del pedido en curso como de todos los clientes.

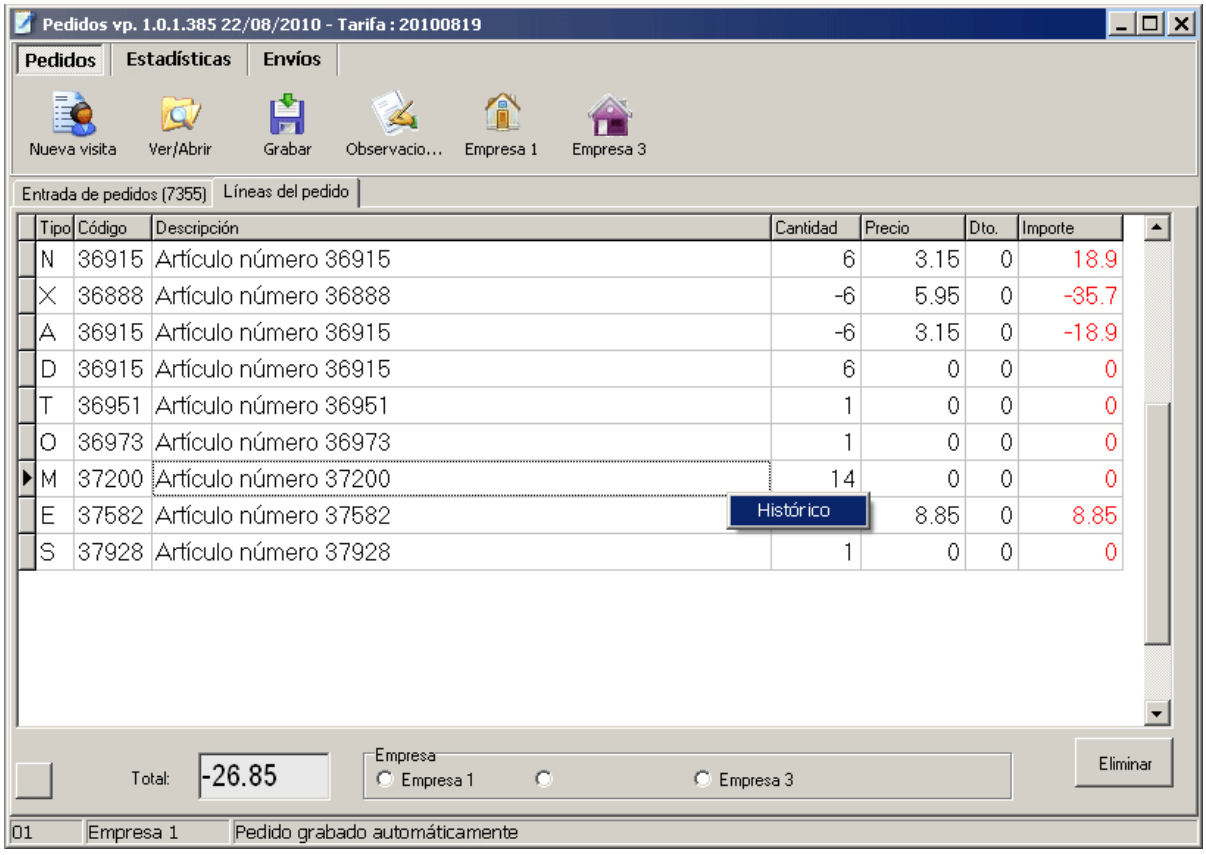
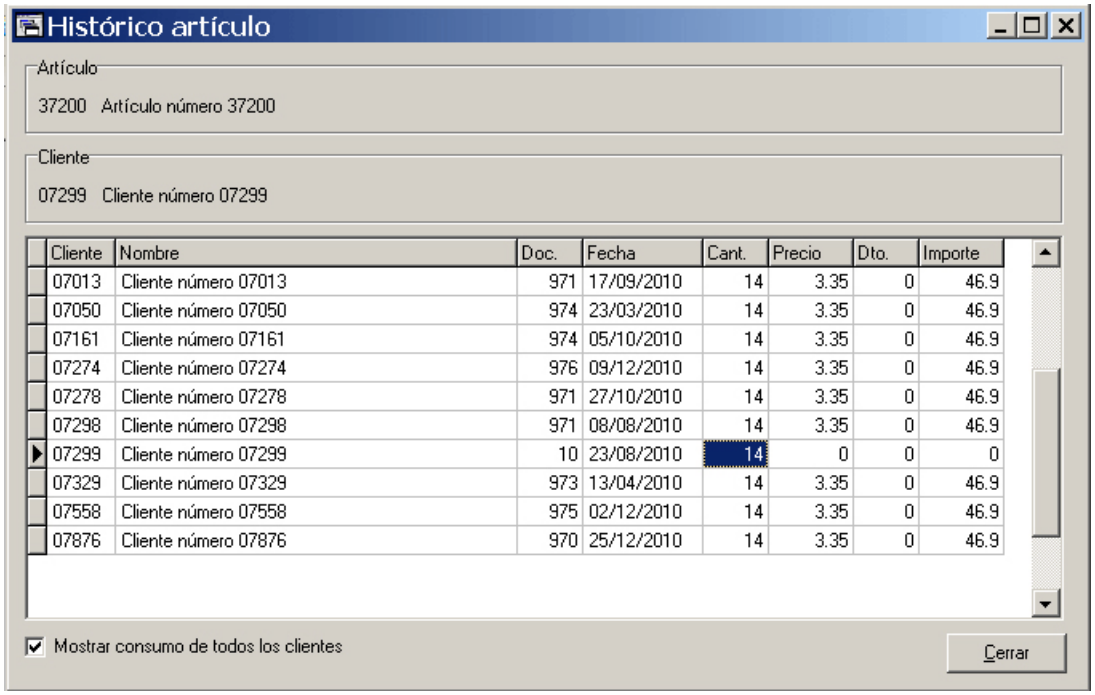


Imagen A7. Líneas del pedido

La opción de acceder al histórico también está disponible en el grid de selección de artículo en la pestaña “Entrada de pedidos”. Si la opción “Histórico” no aparece es posible que se encuentre dehabilitada. Es función del Administrador activar o desactivar esta ventana emergente.



Cliente	Nombre	Doc.	Fecha	Cant.	Precio	Dto.	Importe
07013	Cliente número 07013	971	17/09/2010	14	3.35	0	46.9
07050	Cliente número 07050	974	23/03/2010	14	3.35	0	46.9
07161	Cliente número 07161	974	05/10/2010	14	3.35	0	46.9
07274	Cliente número 07274	976	09/12/2010	14	3.35	0	46.9
07278	Cliente número 07278	971	27/10/2010	14	3.35	0	46.9
07298	Cliente número 07298	971	08/08/2010	14	3.35	0	46.9
07299	Cliente número 07299	10	23/08/2010	14	0	0	0
07329	Cliente número 07329	973	13/04/2010	14	3.35	0	46.9
07558	Cliente número 07558	975	02/12/2010	14	3.35	0	46.9
07876	Cliente número 07876	970	25/12/2010	14	3.35	0	46.9

Imagen A8. Histórico de ventas del artículo

La pestaña “Imagen” sólo se mostrará en caso de que el artículo tenga asociada alguna imagen. Si existen varias imágenes podremos desplazarnos por ellas con las flechas que se muestran o, con un movimiento del clic hacia la derecha o hacia la izquierda

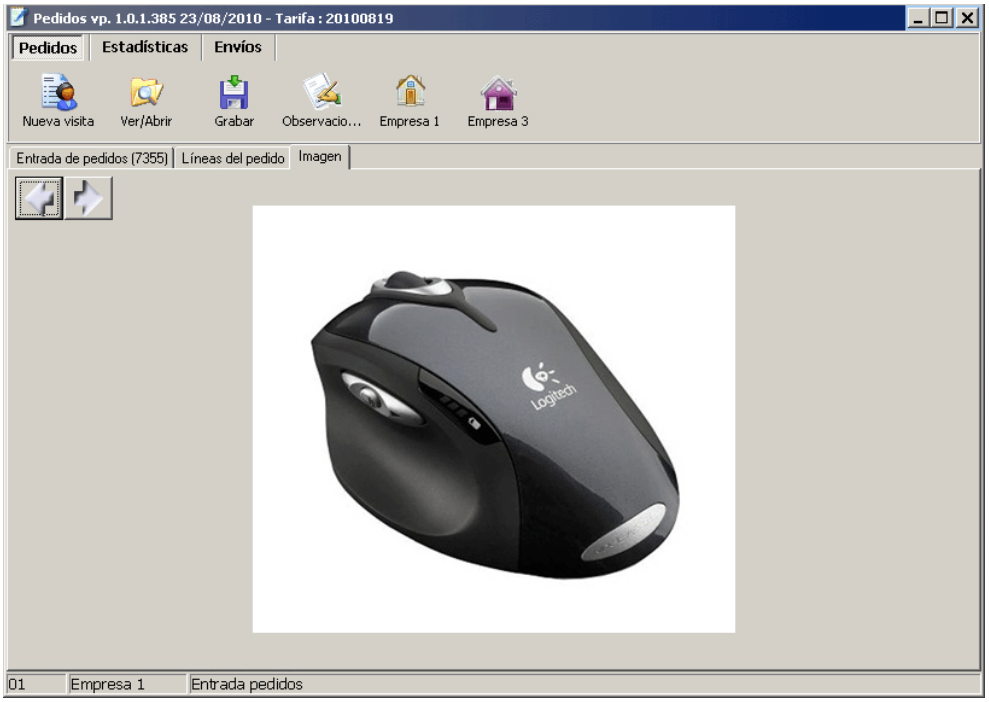


Imagen A8. Imagen del artículo

Si el sentido del “gesto“ táctil que se haga sobre la imagen es vertical, se alternará entre mostrar la imagen en su tamaño real o al máximo que de el formulario que la contiene. Esta opción se ha implementado para conocer un poco el funcionamiento del sistema Touch para pantallas táctiles.

En las opciones de la parte superior veremos los botones que permiten grabar el pedido, introducir observaciones, ver o abrir pedidos, cambiar de empresa o realziar una nueva visita.

Por defecto cada tres líneas añadidas el pedido se guarda automáticamente y si salimos del pedido, aunque no hayamos hecho clic en “Grabar” el pedido quedará grabado. Si no lo deseamos, deberemos borrarlo.

Para las observaciones disponemos de tres líneas de 80 caracteres cada una y una línea reservada para el programa dónde se incluirá la información sobre el mail. Asimismo se ofrece de nuevo un teclado y opciones de borrado rápido de líneas.

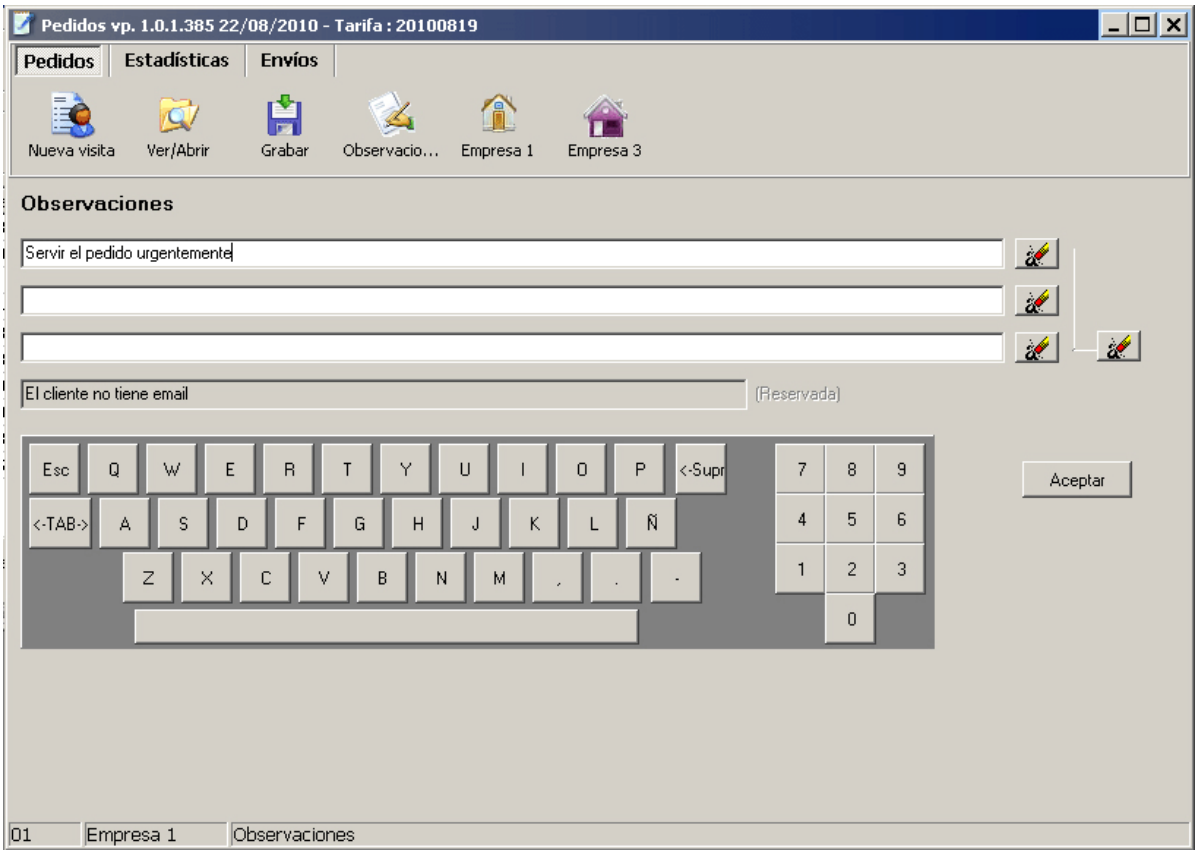


Imagen A9. Observaciones del pedido

Por último, en el formulario de “Ver/Abrir” , aparte de poder abrir un pedido ya grabado para poder editarlo, se pueden realizar las tareas de visualizarlo, eliminarlo o imprimirlo.

Podemos realizar la búsqueda de los pedidos según empresa, fecha y cliente.

En esta misma ventana se pueden visualizar los documentos de facturas mensuales que se reciben de nuestra empresa. Sólo deberemos seleccionar el mes y el año y hacer clic en “Ver facturas de”.

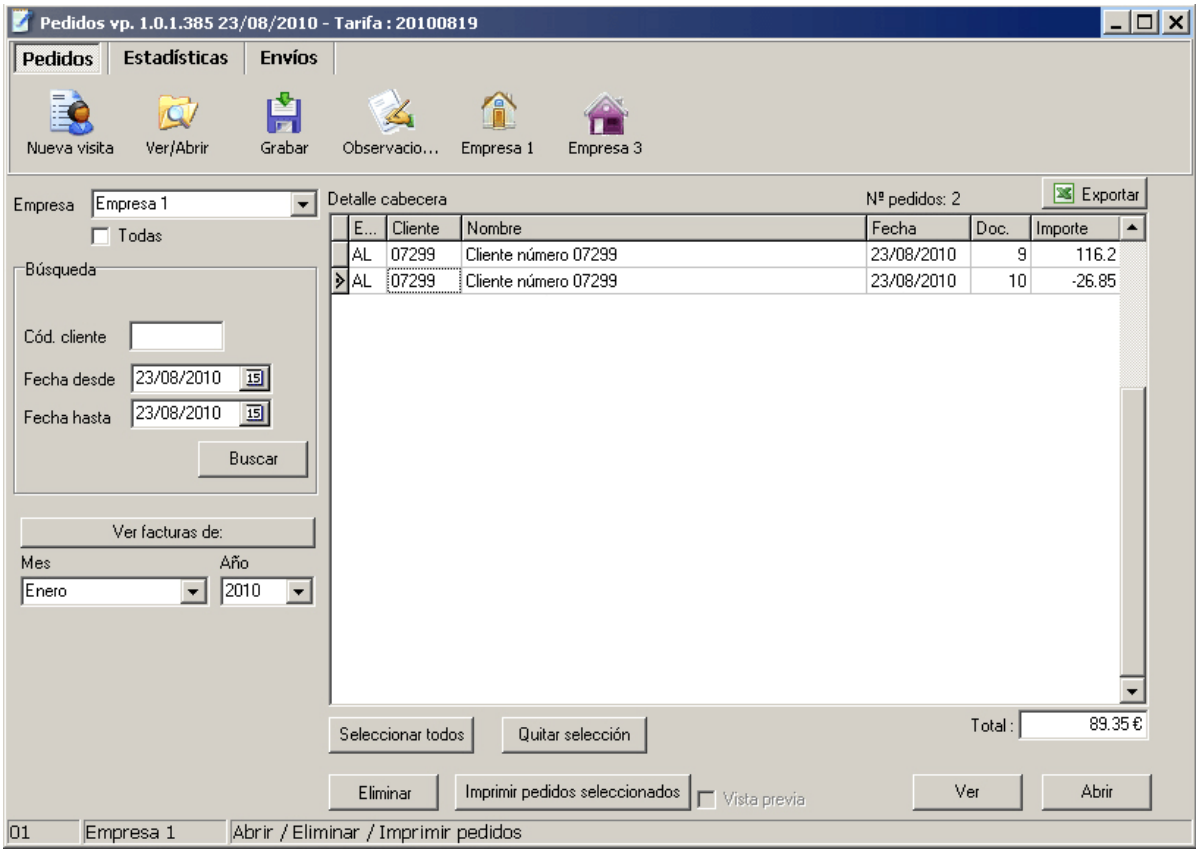


Imagen A10. Opciones de la ventana “Ver/abrir”

- Estadísticas

La ventana de estadísticas se divide en dos partes: una de tablas dinámicas y otra de gráficos.

En ambas opciones se parte de una consulta, pero se opera con su resultado de diferente manera.

Con alas tablas dinámicas podemos “pivotar” con los datos según nos interese mostrar las prioridades de la agrupación. A su vez estos campos pueden filtrarse y ordenarse como se desee.

En el ejemplo de la imagen A11, el campo empresa, que se encuentra fuera de selección, se va a situar delante del campo cliente.

Así conseguiremos primero agrupar por empresa luego por “Cliente”, luego por “Fecha” y como acumulado usaremos el campo “Importe”.

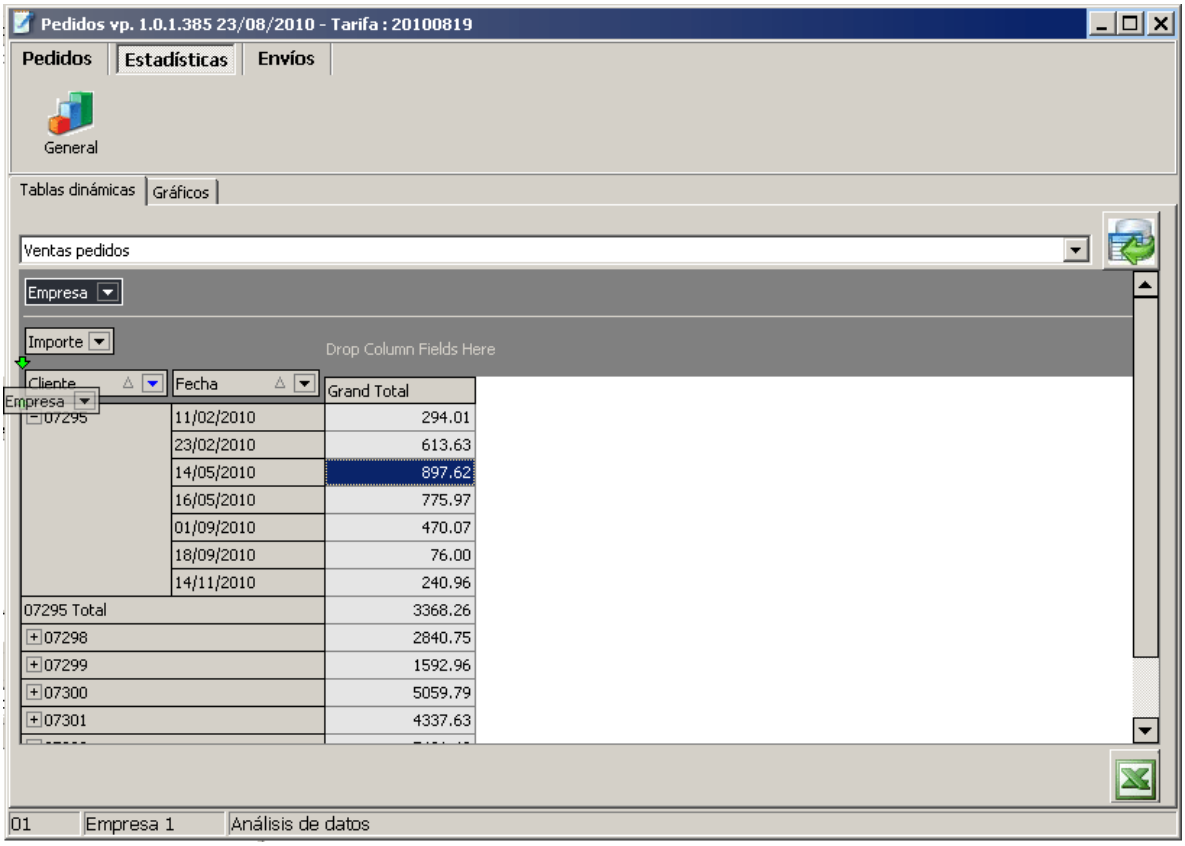


Imagen A11. Tablas dinámicas.

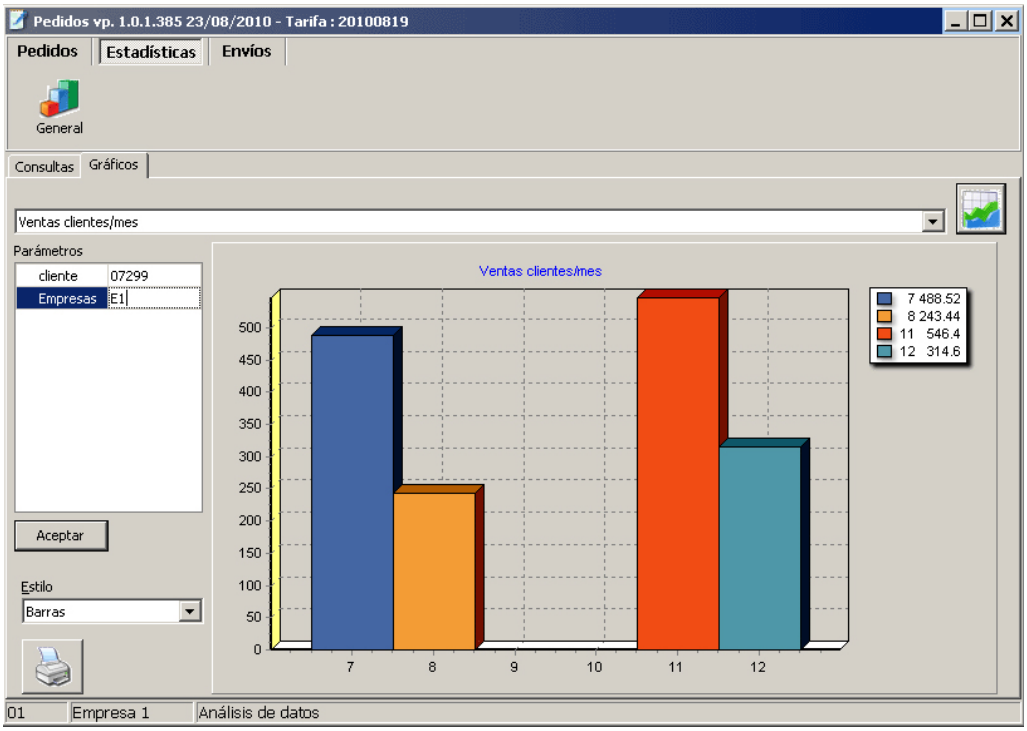
Las tablas pueden ser exportadas a Excel mediante el botón que lleva su símbolo.

Respecto a los gráficos la particularidad es que las consultas que se ofrecen pueden aceptar parámetros.

En el ejemplo se selecciona la consulta “Ventas clientes/mes”, pero antes de poder ver algún resultado debemos rellenar los campos “Cliente” y “Empresas” y luego hacer clic en el botón Aceptar.

Una vez mostrado el gráfico podemos cambiar su estilo a barras, líneas, puntos, pastel, etc.

El gráfico puede ser impreso directamente mediante el botón del icono de la impresora.



A12. Gráficos estadísticos.

- Envíos

Si deseamos enviar los pedidos debemos ir a la parte de “Envíos”. En este módulo, tal y como ilustra la Imagen A13, podemos enviar cualquier pedido que queramos.

Para ello disponemos de un calendario, para seleccionar la fecha de los pedidos. Por defecto no se permite enviar pedidos de diferentes días.

En el grid correspondiente aparece la información requerida sobre los pedidos a enviar. También podremos saber si el pedido ya ha sido enviado o no.

En la parte baja de la pantalla nos encontramos con los controles habituales para enviar un mail. Los datos de estos controles se rellenarán automáticamente.

Estos son:

El origen (“De”), es decir, el usuario. Aparece el nombre del usuario. Se puede cambiar, pero los cambios no se quedan registrados.

El destinatario (“Para”). Según la Empresa que seleccionemos figurará un mail u otro. También se puede cambiar pero sin la posibilidad de guardar los cambios de manera permanente.

El asunto. Siempre será el mismo : “Enviados X pedido/s”. Deberemos indicar el número de pedidos a enviar. Este número deberá coincidir con los pedidos seleccionados en el grid de la parte superior. Si no coinciden Mostrará un mensaje de error.

Disponemos de algunas opciones para el envío de mensajes con los pedidos por email.

La más importante es que podemos usar o no el cliente de correo saliente (SMTP) que incorpora la aplicación.

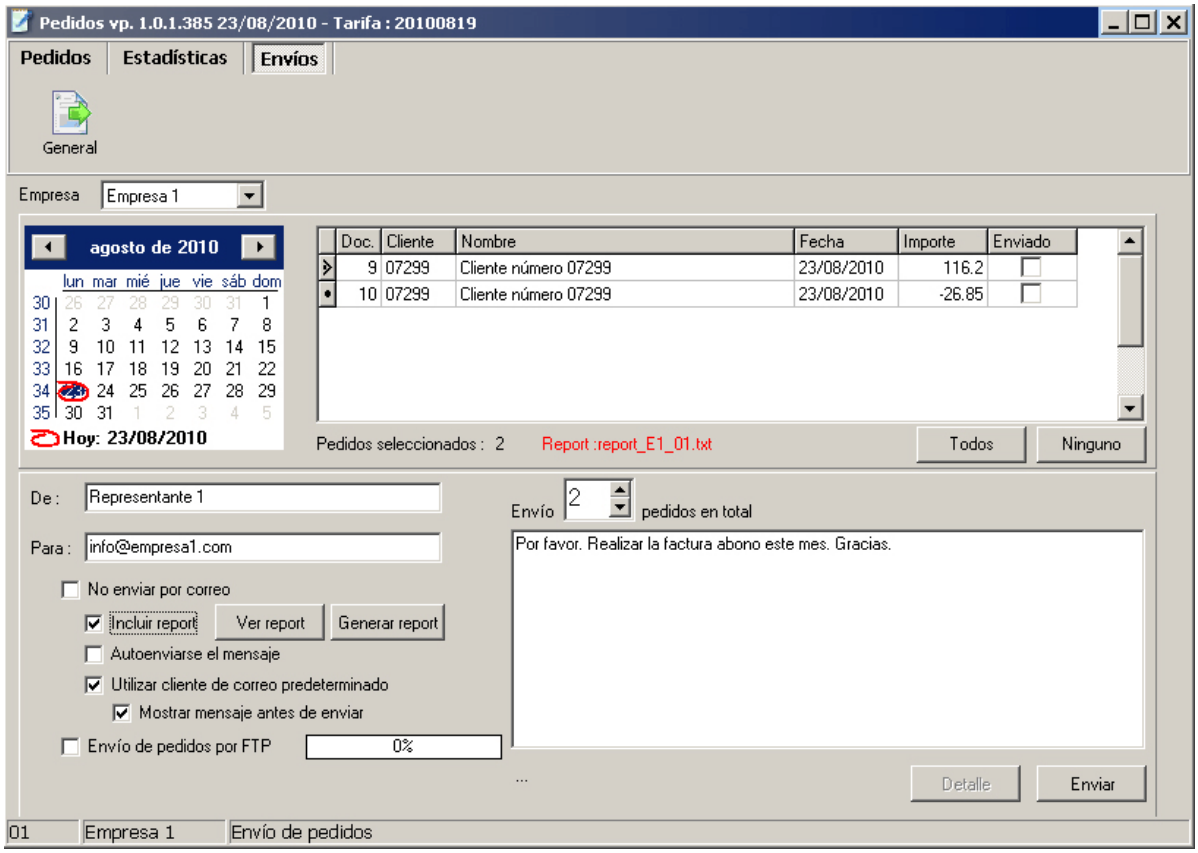
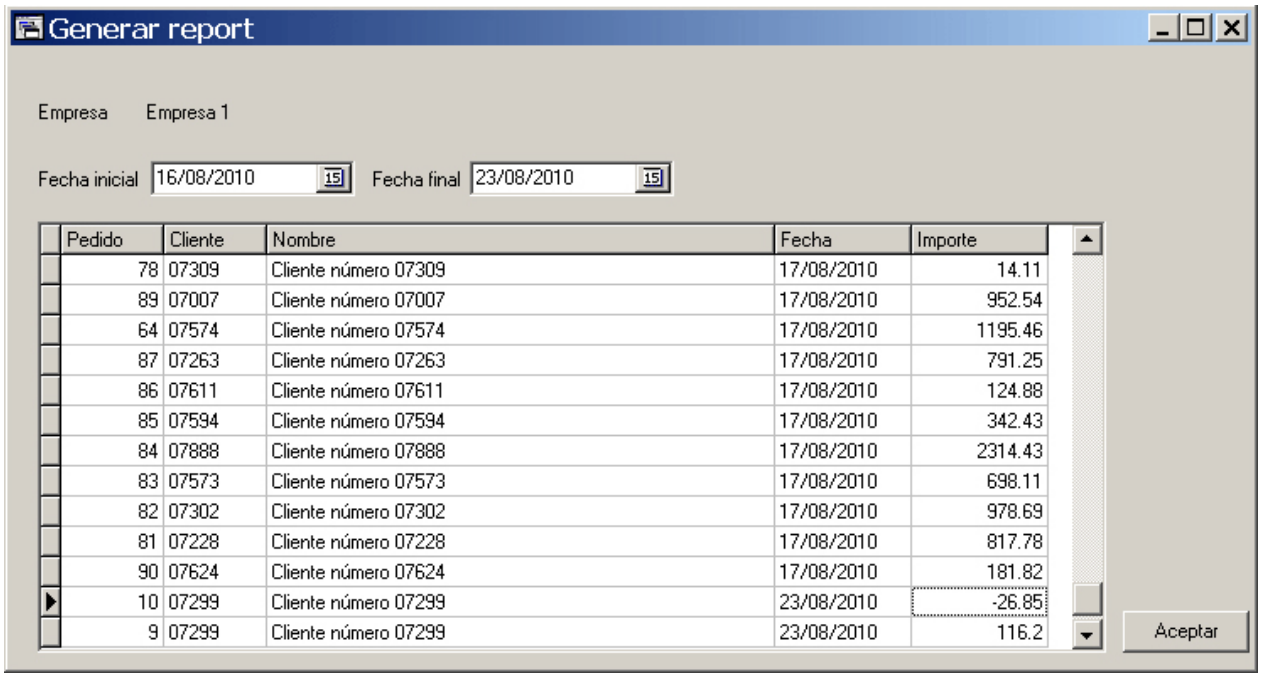


Imagen A13. Módulo de envío de pedidos.

Es la opción estándar y por ello, si nos preferimos utilizar nuestro cliente de correo (Mozilla, Outlook…) deberemos informarlo. Si quieremos que se abra el pedido antes de ser enviado también hay que marcar la casilla correspondiente.

Junto al pedido se puede enviar el informe de visitas (pedidos) de la semana. Es muy sencillo de generar, ya que simplemente hay que hacer clic en el botón “Generar report”, seleccionar las fechas de inicio y fin y hacer clic en Aceptar.



Pedido	Cliente	Nombre	Fecha	Importe
78	07309	Cliente número 07309	17/08/2010	14.11
89	07007	Cliente número 07007	17/08/2010	952.54
64	07574	Cliente número 07574	17/08/2010	1195.46
87	07263	Cliente número 07263	17/08/2010	791.25
86	07611	Cliente número 07611	17/08/2010	124.88
85	07594	Cliente número 07594	17/08/2010	342.43
84	07888	Cliente número 07888	17/08/2010	2314.43
83	07573	Cliente número 07573	17/08/2010	698.11
82	07302	Cliente número 07302	17/08/2010	978.69
81	07228	Cliente número 07228	17/08/2010	817.78
90	07624	Cliente número 07624	17/08/2010	181.82
10	07299	Cliente número 07299	23/08/2010	-26.85
9	07299	Cliente número 07299	23/08/2010	116.2

Imagen A14. Generación de informes de visita.

Una vez tenemos todas las opciones marcadas como nos interesa enviaremos el pedido y se creará un mail en nuestro cliente de correo habitual. En caso de haberlo marcado en las opciones, este mail se visualizará antes de ser puesto en la bandeja de salida.

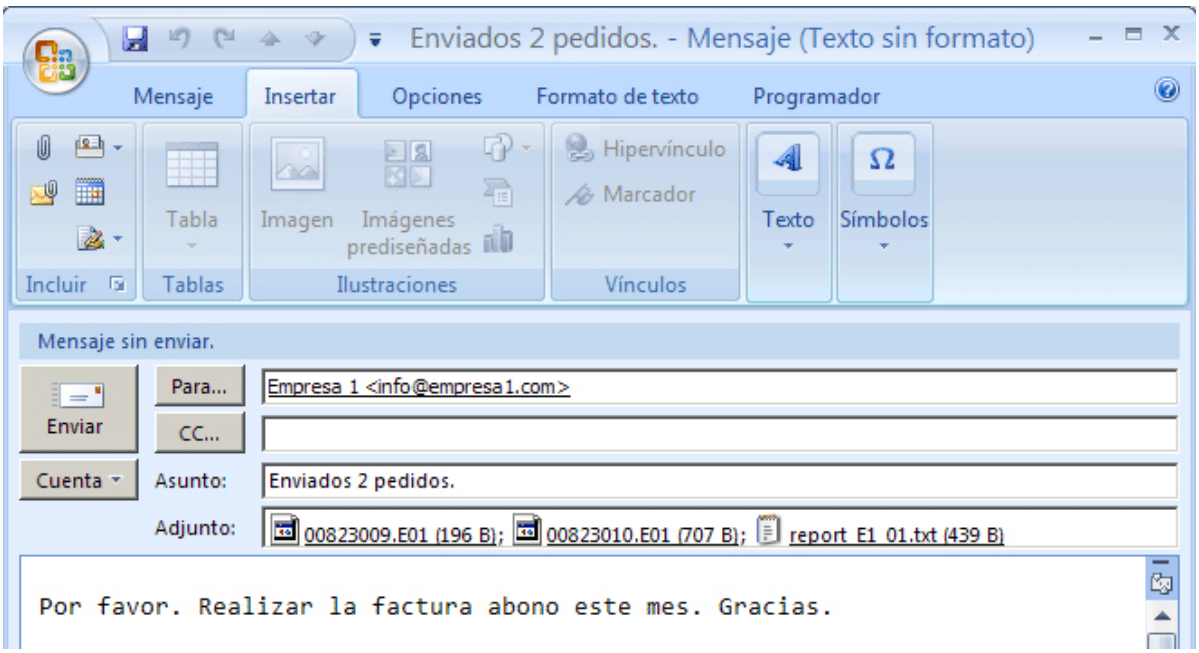


Imagen A15. Ejemplo de mail generado por el programa.

Los pedidos pueden ser enviados también por el servicio de transferencia de ficheros FTP. Incluso se pueden combinar los dos sistemas de envío.

En este último se mostrará una barra de progreso del porcentaje de envío de pedido realizado.

Anexo 3- Formatos

Principalmente en el proyecto, aparte de las estructruras internas de los datos, se han diseñado determinados formatos para la información genererada.

Los más importantes son el formato de fichero de texto que genera el programa, el report de visitas, el fichero con la información del cliente nuevo y el formato de lectura de archivos de facturas.

3.1 Fichero de pedido.

El fichero que genera el pedido constará de tres tipos de línea: uno para la cabecera, otro para las líneas del pedido y la última para las observaciones.

El nombre del fichero tendrá el formato siguiente:

AMMDDPPP.ERR

Donde,

A: Última cifra del año. 0 para el caso de 2010

MM: Mes

DD: Día

PPP: Número de pedido (001-999)

E: Inicial de la empresa

RR: Número de representante

Los campos son de ancho fijo y utilizan todo el espacio reservado.

La línea de cabecera comenzará por ‘01’ y tendrá el siguiente formato.

Línea_id: 2 caracteres para el identificador de línea de cabecera ‘01’

Cliente: 5 caracteres para el código de cliente.

Forma de pago: 3 caracteres para la forma de pago. El primero siempre es 0.

Tarifa: 8 caracteres para el identificador de tarifa utilizada en el pedido.

Las líneas de pedido empezarán por 02

Línea_id: 2 caracteres para el identificador de línea de artículo '02'

Artículo: 5 caracteres para el código del artículo

Cantidad: 8 caracteres para la cantidad. Acepta negativos.

Descripción: 40 caracteres para la descripción del artículo.

Precio: 8 caracteres para la parte entera del precio, 1 posición que indica el punto decimal y 4 caracteres más para los decimales.

Descuento: 2 caracteres para el descuento

Tipo de línea: 1 carácter para el tipo de línea.

Por último, las líneas de observaciones empezarán por 03

Línea_id: 2 caracteres para el formato identificador de línea de observaciones.

Descripción: 80 caracteres por cada línea de observación.

3.2 Formato de archivos de facturas

Para poder visualizar los archivos de facturas que se anvíen al comercial, estos deben seguir la siguiente norma en su nombre:

FAMMRR.EXT

F: Letra 'F'. Indica que es factura.

A: Última cifra del año. 0 para el caso de 2010

MM: Mes de la factura

RR: Identificador del representante comercial.

EXT: Extensión propia del archivo de factura (.PDF, .TXT, .DOC, ect.)

3.3 Información del cliente nuevo.

Cuando se realiza una visita a un cliente que no está registrado se permite guardar la información que el cliente nos pueda facilitar.

El archivo tendrá el mismo nombre que el fichero de pedido al cual vaya asociado y precedido de 'dc_'.

Un ejemplo de la información que alberga se muestra a continuación.

```
Detalle cliente nuevo

---------------------

Representante: 01-Xavier Royo Melero

Empresa: AL

Fecha: 01/09/2010

Documento: 10

---------------------

Nombre: Cliente nuevo 1

Domicilio: Calle Barcelona, 1

Poblacion: Sabadell

Provincia: 08- Barcelona

Cód. Postal: 08000

Teléfono: 626262626

Pers. Fiscal: Informática Sabadell SL

NIF: 4343434343X

Domic. banc.: 0000-0000-00-0000000000

Forma pago: 0 - 0 Contado

Tipo IVA: 18%
```

Diccionario de términos

CRM: Customer Relationship Management. Software diseñado para gestionar todos los aspectos necesarios en la relación comercial con el cliente.

DCOM: Distributed Component Object Model. Sistema de componentes de software de Microsoft para comunicarse con ordenadores en línea.

DLL: Dynamic-Link Library. Librería de enlace dinámico con código ejecutable y que puede ser llamado desde una aplicación principal.

DNS: Domain Name System. Sistema de nombres de dominio que establece la asociación entre direcciones IP y el dominio al cual está vinculado.

ERP: Enterprise Resource Planning. Software para empresas que integra la informatización de todos los procesos que intervienen.

FTP: File Transfer Protocol. Protocolo para la transferencia de ficheros.

Grid: Cuadrícula orddenada en filas y columnas donde se disponen los datos a mostrar.

HTML: HyperText Markup Language. Lenguage básico básado en etiquetas utilizado para mostrar páginas Web.

HTTP: HyperText Transfer Protocol. Protocolo usado en cada transacción de la WWW.

HTTPS: Hypertext Transfer Protocol Secure. HTTP con sistemas de seguridad.

IDE: Integrated Development Environment. Entorno de desarrollo integrado que ofrecen determinados lenguages de programación.

LOPD: Ley Orgánica de Protección de Datos.

MFC: Microsoft Foundation Classes. Librerías del API de Windows.

MDI: Multiple Document Interface. Característica de aplicaciones donde sus ventanas residen bajo una ventana padre.

PDA: Personal Digital Assistant. Pequeño ordenador de mano originalmente diseñado como agenda electrónica.

RAD: Rapid Application Development. Desarrollo rápido de aplicaciones.

Script. Archivo de órdenes de procesamiento por lotes.

SmartPhone: Teléfono Inteligente. Dispositivo que funciona como un teléfono móvil que incorpora características propias de un ordenador personal.

SQL: Structured Query Language. Lenguaje de consultas sobre bases de datos.

TCP: Transmission Control Protocol. Protocolo básico de las comunicaciones en Internet.

Transacción: Proceso en el cual se trata la información de los sistemas computacionales de manera completa o definida.

UML: Unified Modeling Language. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema

VCL: Visual Component Library. Librería de componentes visuales.

Xavier Royo Melero