Desarrollo y puesta en marcha de un software empresarial de gestión en una PYME

MEMORIA PRESENTADA POR:

Lluís Bernabeu Bravo

GRADO DE INGENIERÍA INFORMÁTICA

Convocatoria de defensa: *Septiembre 2019*

RESUMEN

Actualmente, los sistemas informáticos se han convertido en un elemento imprescindible a la hora de gestionar una empresa. La informática facilita el trabajo de administración en las empresas, así como la facturación y gestión de clientes…

El software que se ha programado y se va a exponer en este TFG, facilita el trabajo de gestión en una PYME, en concreto en una de empresa dedicada a la fabricación de persianas y cortinas. Para ello se tuvo que hacer una investigación de las necesidades y hacer una programación a medida.

Todo el software está desarrollado en Java por la facilidad de programación que tiene su librería visual, además se ha utilizado Firebird como motor de base de datos, por la facilidad de instalación y la facilidad de transporte de la base de datos.

PALABRAS CLAVE

JAVA – Software – firebird – base de datos - PYME - Jaspersoft – Netbeans – Swing

ABSTRACT

Currently, computer systems have become an essential element when managing a company. Computer science facilitates the work of administration in companies, as well as billing and customer management ...

The software that has been programmed and will be exhibited in this TFG, facilitates the management work in an SME, specifically in a company dedicated to the manufacture of blinds and curtains. To do this, an investigation of the needs had to be done and a customized program was made.

All the software is developed in Java because of the ease of programming of its visual library, Firebird has also been used as a database engine, due to the ease of installation and ease of transport of the database.

KEY WORDS

JAVA – Software – firebird – database – SME - Jaspersoft – Netbeans - Swing

AGRADECIMIENTOS

Primeramente, quiero agradecer a todos los profesores que he tenido durante mi vida de estudiante, en especial a los que me han ayudado y motivado a llegar donde he llegado.

A mi director de TFG, Pedro José Ramiro Zafra, por su inestimable ayuda tanto en los estudios como en el ámbito laboral.

A mis padres les quiero agradecer la educación que me han dado, además de darme la posibilidad de estudiar una carrera, cosa que ellos no pudieron hacer.

A mi pareja Laura, por apoyarme siempre, soportándome en épocas malas, celebrando los éxitos y dándome consuelo en los fracasos.

Y por último, agradecer a los que no están, en especial a mi abuela Cornelia, quien me cuidó y educó cuando mis padres no pudieron.

Índice

Capítulo 1: Introducción y Objetivos 11
1.1 Introducción 12
1.2 Motivación 12
1.3 Objetivos 13
Capítulo 2: Estado del arte, análisis de requerimientos y selección de los componentes 15
2.1 Estado del arte 16
2.2 Análisis de requerimientos 18
2.3 Selección de los diversos elementos 19
2.4 Fase de desarrollo 20
Capítulo 3: BBDD 23
3.1 Definición de la BBDD 24
3.1.1 TB_Articulos 25
3.1.2 TB_Clientes 25
3.1.3 TB_Empresa 26
3.1.4 TB_Documentos 27
3.1.5 TB_Lin_Documentos 27
3.1.6 TB_Abonos y TB_Lin_Abonos 28
3.1.7 Tb_Informe_cliente y Tb_trimestre 28
3.1.8 Otras tablas 28
3.2 Triggers y procedures 28
Capítulo 4: Programación 30
4.1 Mantenimiento general 32
4.1.1 Botón Buscar 34
4.1.2 Botón “nuevo” 35
4.1.3 Doble clic tabla 35
4.1.4 Botón “guardar” 36
4.1.5 Botón cancelar 37
4.1.6 Botones navegación 37
4.1.7 Botón “eliminar” 39
4.2 Mantenimiento simple 40
4.2.1 Gestiones impresoras 40
4.2.2 Datos de la empresa 42
4.2.3 Opciones 43
4.3 Etiquetas 46
4.3.1 Botón añadir artículos 47

4.3.2 Añadir cliente ... 49
4.3.3 Borrar cliente y borrar artículo ... 51
4.3.3.1 Borrar cliente ... 51
4.3.3.2 Borrar artículo ... 51
4.3.4 Botón imprimir ... 52
4.4 Documentos de Venta ... 54
4.4.1 Cabecera ... 56
4.4.1.1 Borrar artículo ... 57
4.4.1.2 Transformar ... 58
4.4.2 Cuerpo ... 59
4.4.2.1 Unidades ... 60
4.4.2.2 Lineal ... 60
4.4.2.3 Cuadrado ... 61
4.4.2.4 Refactorizar ... 62
4.4.2.5 Calcular totales ... 63
4.4.3 Pie ... 64
4.4.3.1 Imprimir ... 64
4.4.3.2 Etiquetas ... 65
4.4.3.3 Guardar ... 66
4.4.3.4 Eliminar ... 68
4.5 Informes ... 70
4.6 Importador/Exportador ... 72
Capítulo 5: Informes ... 73
5.1 Documentos de venta ... 75
5.2 Etiquetas ... 78
5.3 Informes ... 79
Capítulo 6: Anexo ... 81
Capítulo 7: Bibliografía ... 89
Capítulo 8: Conclusiones y líneas futuras ... 91
8.1 Líneas futuras ... 92
8.2 Opinión personal ... 93

Índice de ilustraciones

Ilustración 1. Porcentaje utilización IDE https://www.dunebook.com/intellij-vs-eclipse/ 20
Ilustración 2. Diagrama de Gantt del proyecto de desarrollo 22
Ilustración 3. Esquema Relacional BBDD 24
Ilustración 4. TB_ARTICULOS 25
Ilustración 5 - TB_CLIENTES 25
Ilustración 6. TB_EMPRESA 26
Ilustración 7. TB_DOCUMENTOS 27
Ilustración 8. TB_LIN_DOCUMENTOS 27
Ilustración 9. Trigger antes de insertar 28
Ilustración 10. Trigger después del borrado 29
Ilustración 11. Procedure después del borrado 29
Ilustración 12. Pantalla inicio 31
Ilustración 13. Inicio nueva pantalla 32
Ilustración 14. Mantenimiento General - Lista 32
Ilustración 15. Mantenimiento General - Ficha 33
Ilustración 16. Mantenimiento general - Artículos 33
Ilustración 17. Método buscar 34
Ilustración 18. Sentencia SQL para buscar 34
Ilustración 19. Rellenar Tabla 34
Ilustración 20. Botón nuevo 35
Ilustración 21. Evento Doble clic 35
Ilustración 22. Método guardar 36
Ilustración 23. Insertar elemento 36
Ilustración 24. Guardar elemento 36
Ilustración 25. Botón cancelar 37
Ilustración 26. Botones navegación 37
Ilustración 27. Rellenar ficha 38
Ilustración 28. Botón eliminar 39
Ilustración 29. Eliminar elemento 39
Ilustración 30. Gestiones impresoras 40
Ilustración 31. Impresoras instaladas en el sistema 40
Ilustración 32. Rellenar desplegables 41
Ilustración 33. Guardar impresoras 42
Ilustración 34. Datos de la empresa 42
Ilustración 35. Borrado masivo de datos 43
Ilustración 36. Copia de seguridad 44
Ilustración 37. Cierre de año 45
Ilustración 38. Cambio de I.V.A. por defecto 45
Ilustración 39. Gestión etiquetas 46
Ilustración 40. Añadir artículo a etiqueta 47
Ilustración 41. Botón añadir artículo 47
Ilustración 42. Bucle para añadir los artículos 48
Ilustración 43. Consulta SQL para añadir el artículo 48
Ilustración 44. Bucle añadir artículo a la tabla 48
Ilustración 45. Añadir cliente a etiquetas 49
Ilustración 46. Botón añadir cliente 49

Ilustración 47. Llamada dependiendo de la pantalla 50
Ilustración 48. Método AñadirAux 50
Ilustración 49. Limpiar cliente 51
Ilustración 50. Método borrar artículo 51
Ilustración 51. Bucle for para recorrer la tabla 52
Ilustración 52. Recalcular código de barras 52
Ilustración 53. Parámetros Jasper 53
Ilustración 54. Compilar e imprimir Jasper 53
Ilustración 55. Etiqueta pequeña 54
Ilustración 56. Documentos de venta - Lista 54
Ilustración 57. Documentos de venta - Ficha 55
Ilustración 58. Cabecera doc. venta 56
Ilustración 59. Focus lost - Tipo de doc. venta 56
Ilustración 60. Eliminar artículo 57
Ilustración 61. Eliminar artículo de la tabla 57
Ilustración 62. Transformar documento de venta 58
Ilustración 63. Actualizar número de factura 58
Ilustración 64. Insertar las líneas 59
Ilustración 65. Focus gain - Tabla artículos 59
Ilustración 66. Método elegir 60
Ilustración 67. Método unidades 60
Ilustración 68. Método lineal 60
Ilustración 69. Método cuadrado 61
Ilustración 70. Método refactorizar 62
Ilustración 71. Seleccionar I.V.A. 63
Ilustración 72. Calcular totales 63
Ilustración 73. Asignar valores 63
Ilustración 74. Pie - Documento de ventas 64
Ilustración 75. Documento de venta impreso 64
Ilustración 76. Gestión etiquetas doc. ventas 65
Ilustración 77. Etiqueta con remitente 65
Ilustración 78. Etiqueta sin remitente 65
Ilustración 79. Buscar forma de pago 66
Ilustración 80. Insertar cabecera doc. venta 67
Ilustración 81. Insertar líneas doc. venta 68
Ilustración 82. Insertar en abonos 68
Ilustración 83. Actualizar stock 69
Ilustración 84. Marcar como abonado 69
Ilustración 85. Pantalla informe de ventas 70
Ilustración 86. Sentencia SQL para informe 71
Ilustración 87. Informe impreso 71
Ilustración 88. Importador 72
Ilustración 89. Exportador 72
Ilustración 90. Factura – Jasper 75
Ilustración 91. Sentencia SQL para cargar cabecera 76
Ilustración 92. Sentencia SQL para cargar las líneas 76
Ilustración 93. Sentencia SQL para cargar totales 77
Ilustración 94. Ficheros JRXML - Doc. Ventas 77
Ilustración 95. Etiqueta grande 78
Ilustración 96. Etiqueta pequeña 78
Ilustración 97. Ficheros JRXML - Etiquetas 79
Ilustración 98. Informe 79

Ilustración 99. Sentencia SQL - informe 80
Ilustración 100. Ficheros JRXML – Informes 80
Ilustración 101. Librerías utilizadas 82
Ilustración 102. SQL Ventas Diario 82
Ilustración 103. SQL Ventas Semana 83
Ilustración 104. SQL Ventas Mes 83
Ilustración 105. SQL Ventas Familia 84
Ilustración 106. SQL Ventas Cliente 84

Capítulo 1: Introducción y Objetivos

1.1 Introducción

Hoy en día los sistemas de información tienen un papel decisivo en las empresas, este factor condiciona en muchos casos el éxito o fracaso de un proyecto empresarial. Por lo que el correcto desarrollo e implantación de un software en una empresa, puede ser la clave para el buen funcionamiento y alcance de los objetivos estratégicos de esta. Un buen diseño de software puede ayudar a lograr ventajas competitivas, aligerando el trabajo y dando información necesaria para conseguir objetivos de manera más rápida y sencilla.

1.2 Motivación

La principal motivación de este proyecto fue la oportunidad de desarrollar un software a medida para una empresa que tenía unos problemas concretos y unas necesidades específicas no cubiertas por el que en esa época era su principal software de gestión. Esta empresa necesitaba, por su forma de trabajar, un software que se adaptase a su forma de vender y tuviese unos informes específicos. Tras analizar el mercado se observó que no había ningún software que cumpliese con los requisitos o los que cumplían los requisitos técnicos tenían un coste, económicamente hablando altísimo, y a parte no se adaptaban del todo a la forma de trabajar, con lo que se concluyó que se necesitaba una programación a medida. Esta situación, los llevó a buscar a una persona que les hiciese un software de gestión empresarial que cumpliese todas sus necesidades.

Por otro lado, la empresa tenía un software del año 1999 por el cual tenía que pagar licencias de actualización y de mantenimiento. En el nuevo software se quería dejar estos gastos de lado y que fuesen todo licencias de software libre.

El software que tenían era muy complejo de utilizar, ya que en la época en la que se programó no existían muchas de las herramientas que hay hoy en día, con lo que las operativas eran muy difíciles de realizar, teniendo limites en algunos aspectos que ahora ya no están.

Hoy en día, una empresa que no tenga un buen software de gestión empresarial que contribuya a su gestión empresarial, está condenada al fracaso, ya que, además de no contribuir a alcanzar los objetivos de la empresa, va a perder mucho tiempo y dinero en hacer los informes, las ventas, en cuadrar los trimestres, etc. Hacer un software sencillo, rápido y útil era la principal motivación de cara a afrontar este nuevo reto.

Es importante recalcar que por razones de confidencialidad y a petición expresa de los responsables de la empresa, se han omitido ciertos datos, imágenes y logos que puedan identificar la empresa objeto de este TFG.

1.3 Objetivos

Hay dos objetivos generales que engloban a toda la aplicación, y dentro ellos se irán desglosando en objetivos específicos de cada apartado.

El primer objetivo era seleccionar las tecnologías a utilizar, base de datos, lenguaje de programación, sistema de informes, etc. Todo ello debía ser bajo el licenciamiento de software libre ya que la empresa no quería pagar licencias.

El segundo objetivo era implementar una aplicación sencilla y con mucho potencial que contribuyese a la gestión de la empresa y aligerase el trabajo de oficina que era uno de los grandes problemas a los que se enfrentaba la empresa.

Capítulo 2: Estado del arte, análisis de requerimientos y selección de los componentes

Para el buen desarrollo del software empresarial a medida para la Pyme en cuestión, además de realizar un análisis de requerimientos, es necesario estudiar los lenguajes de programación, bases de datos y entornos de desarrollo para poder seleccionar los que mejor se adapten a las necesidades de la empresa.

En este capítulo se va a estudiar el estado del arte y se van a describir las necesidades específicas de la empresa y como se planificó todo el desarrollo del software, así como la elección de cada uno de los elementos que componen el desarrollo final.

2.1 Estado del arte

En este apartado se van a tratar las principales diferencias entre los diferentes lenguajes de programación, bases de datos y entornos de desarrollo que se han barajado entre las opciones posibles.
Para ello se van a utilizar diferentes tablas marcadas con un “Si”, si el objeto de estudio tiene la característica descrita, o con un “No” en caso contrario.

Lenguajes de programación			
Característica	Java	C++	Python
Multiplataforma	Si	Si	Si
Compilado	No	Si	No
Interpretado	Si	No	Si
Gratis	Si	Si*	Si
Comunidad	Si	Si	Si
Interfaz gráfica	Si	Si	Si
Librerías de terceros	Si	Si	Si
Orientado a objetos	Si	Si	Si
Base de datos	Si	Si	Si
Fácil de aprender	Si	Si	Si

**El compilador utilizado puede ser de pago; algunos de ellos ofrecen muchas ventajas frente a los que son gratis.*

A la vista de la tabla anterior, la única diferencia entre los tres lenguajes es si son interpretados o compilados. La principal diferencia entre un lenguaje interpretado y compilado es que el compilado requiere una acción previa a la ejecución que es la propia compilación que traduce el código escrito por el programador a código máquina. En cambio, los lenguajes interpretados traducen este código a código máquina en tiempo de ejecución, lo que puede hacer que en algunas instrucciones se ralenticen.

El caso de Java es un caso particular puesto que es un lenguaje compilado, pero no a código máquina sino a un lenguaje que hace de intermediario bautizado como Bytecode. Éste será interpretado posteriormente por la Java Virtual Machine, lo que hace que Java sea un lenguaje multiplataforma y no se haya de compilar por cada Sistema Operativo en el que se desee instalar. Esta característica hace que sea una ventaja frente a sus competidores.

BBDD			
Característica	Firebird	MySQL	PostgresSql
Licencia	Open Source	Open Source	Open Source
Triggers	Si	Si	Si
Concurrencia	Si	Si	Si
Procedures	Si	Si	Si
Vistas	Si	Si	Si
AutoIncremental	No	Si	Si
Generators	Si	Si	Si
JDBC	Si	Si	Si
Base de datos física	Si	No	No

Un requisito indispensable para la empresa era que la base de datos fuese un archivo físico que ellos pudieran mover y hacer copias fácilmente sin dependencia de herramientas externas o conocimientos informáticos avanzados. Este aspecto, es de las pocas diferencias que existen entre estas tres bases de datos relacionales.

Entornos de desarrollo			
Característica	Netbeans	Intelllj	Eclipse
Diseño Swing	Si	Si	Si**
Generación JAR*	Si	Si	Si**
Debug	Si	Si	Si
Gratis	Si	Si***	Si
Compilación Incremental	Si****	No	SI
Plugins	Si	Si	Si
Autocompletar	Si	Si	SI

**El archivo de tipo JAR, es el ejecutable de Java. La generación automática de este elemento facilita el trabajo de programación.*

***En eclipse si se quiere utilizar una herramienta para diseñar el aspecto visual de las pantallas, se tiene que instalar mediante plugin. Además, la generación de JAR no es tan potente como en sus competidores, teniendo que utilizar herramientas externas.*

****La herramienta de desarrollo de Java es gratuita, aunque tiene una versión de pago que incorpora otros lenguajes de programación.*

*****La compilación incremental no es una de sus herramientas, pero se puede instalar mediante un plugin.*

A la vista de los resultados, se puede observar que los tres están a la par en cuanto a características, aunque en este caso destaca Netbeans por incorporar un diseño Swing más potente que la competencia y además poder incorporar la compilación incremental.

2.2 Análisis de requerimientos

La empresa en la que se ha desarrollado este software a medida es una empresa unifamiliar, que cuenta con pocos empleados con lo que lo que agilizar el trabajo de oficina es la prioridad marcada como objetivo.
La actividad principal de la empresa es la construcción de cortinas y persianas a medidas al por mayor. Al ser una empresa familiar, el número de empleados es limitado, siendo variable dependiendo de si es temporada baja o temporada alta. En temporada alta pueden llegar a ser seis personas. Al ser tan pocos trabajadores, la función que desarrollan cada uno de ellos es similar, puesto que no tienen un rol diferente asignado. Se pueden diferenciar tres perfiles diferentes:
1.Montador.
2.Montador-Oficinista.
3.Montador-Transportista.

Actualmente, la empresa cuenta con una infraestructura informática simple, cuentan con un ordenador de sobremesa donde está el programa actual de gestión y el correo, tres impresoras, dos de ellas de etiquetas y una para los informes/documentos de venta. Además, cuentan con un servidor externo contratado para el alojamiento de la página web.

El software de gestión que tenían databa del 1999 y sin actualizar desde entonces, con los problemas inherentes de seguridad y operatividad que esto conlleva. Esta situación les obligaba a tener una versión antigua del sistema operativo para que pudiese funcionar.
Para poder actualizar este software, la empresa propietaria les obligaba a adquirir un sistema de licencias y pagos mensuales, lo cual después del mal resultado que estaba dando, no querían hacer. Por estas razones, les surgió la idea de que les hiciesen un software a medida, personalizado y que cumpliese todas sus necesidades y les facilitara el trabajo del día a día.

Por su forma de trabajar, ellos necesitaban una gestión íntegra de los artículos donde pudiesen tener tres formas de medida, unitaria, lineal y por metros cuadrados y que la pantalla de ventas pudiese calcular estos valores sin tener que ellos hacer un esfuerzo extra en cada venta. Además, también necesitaban tener una gestión de los clientes más exhaustiva que la que se tenía actualmente, para poder añadir los clientes a las etiquetas, a las facturas o a los albaranes.

Para poder especificar estas necesidades y muchas otras, se mantuvieron reuniones durante una semana con todos los componentes de la empresa, donde se estudiaron sus necesidades y donde se analizó el anterior software, viendo sus debilidades y sus virtudes y con ello poder hacer una planificación de todas las herramientas que debía tener el programa de cara a facilitar el trabajo y satisfacer sus necesidades.

Una vez que las necesidades estaban claras, se pasó a diseñar el entorno visual con el que pudieran familiarizarse y así hacerles el trabajo más fácil. Para ello se mantuvieron reuniones para diseñar el aspecto visual de cada uno de los elementos que componían el software.

Por último, se analizaron los documentos de venta del programa anterior y se mejoraron. Además, se añadieron informes tanto impresos como por pantalla que en el anterior no disponían.

Tras estos análisis, se volvió a sondear el mercado en busca de un software comercial que pudiera satisfacer las necesidades concretas y el precio que estaba dispuesto a pagar la empresa. Se llegó a la conclusión de que no existía actualmente un software que se adaptase a todas las necesidades que tenía este cliente en específico, por su forma de vender, por sus etiquetas, por las operativas que tienen que hacer, con lo que no quisieron ni intentar adaptar un software comercial, ya que el coste económico hubiese sido incalculable, por lo que escogieron esta solución de software a medida.

Para llevar a cabo todo lo descrito anteriormente, lo primero que se tuvo que hacer es elegir las herramientas que se iban a utilizar.

2.3 Selección de los diversos elementos

Partiendo de las necesidades descritas en el apartado anterior y los análisis de los lenguajes, BBDD y entornos de programación, lo primero que se escogió fue el lenguaje de programación, que tras los análisis pertinentes (que a continuación se resumen) fue Java el finalmente escogido.

Java es un lenguaje de programación orientado a objetos, fue creado en 1991 y surgió de la idea de tener que programar el código tan solo una vez y que la Java Virtual Machine se encargará de “traducir” ese código al código necesario en cada arquitectura y plataforma.

Se escogió Java por las siguientes razones:

- Facilidad de exportación del código.
- Gran cantidad de librerías nativas y de terceros.
- Buena integración con Jaspersoft, software para hacer informes; en este caso facturas, albaranes…
- Curva de aprendizaje rápida.
- Librería visual Swing.
- Facilidad de integración con las bases de datos.
- Es gratis.
- Es Multiplataforma.

Con la BBDD lo más lógico habiendo elegido Java como lenguaje de programación sería elegir MySQL ya que es de las más utilizadas con Java, pero en este caso se escogió Firebird. Ambas bases de datos soportan las mismas operaciones, aun así, lo que hizo elegir el motor de base de datos de Firebird en vez de MySQL es la facilidad de instalación y de exportación, ya que al ser un archivo físico se puede mover fácilmente haciendo “copiar y pegar” y cabe recordar que esta característica era requisito indispensable de la empresa.

En el caso de la empresa en cuestión, y tal y como ya se ha mencionado, el poder exportar la BBDD fácilmente era un requisito y una ventaja ya que no querían depender de nadie para poder exportar la BBDD de un ordenador a otro o hacer copias de seguridad.

Para otras empresas, esto será una desventaja ya que si no se protege debidamente cualquiera podría llevarse el archivo, aunque para acceder a ella se necesitaría un usuario y contraseña, como en cualquier otra BBDD. Aun así, esta forma de operar es muy útil para empresas pequeñas y con poca infraestructura, ya que facilita el trabajo

de copias de seguridad y la instalación en otros ordenadores. Además, como Java, es multiplataforma y se podría utilizar en cualquier sistema operativo.

Y, por último, la elección del IDE. Hoy en día hay tres grandes IDE para desarrollo en Java que son: Netbeans, IntelliJ y Eclipse. La elección para este proyecto fue Netbeans.

Como se puede ver en el gráfico de la *"Ilustración 1"*, Netbeans es el menos utilizado de los tres. En este caso, la elección se debió a un tema meramente personal, ya que IntelliJ no lo conocía y en cuanto a Eclipse, nunca me terminó de gustar la forma en la que se trataba las herramientas y la librería Swing, además de que había muchos problemas con los Workspaces.

El principal motivo de utilizar Netbeans en vez de Eclipse fue la manera que tenía Netbeans de tratar la librería Swing, ya que se puede hacer el diseño visual, simplemente arrastrando y dejando caer los elementos en la pantalla, siendo la visualización del resultado final inmediata. Otro motivo que me llevo a elegir Netbeans fue la generación de Jar desde un simple botón. Esta herramienta te genera el ejecutable y te exporta las librerías que necesita para funcionar en un simple clic, siendo en Eclipse más difícil de hacer.

Si volviera a tener que realizar la elección actualmente, y dada mi experiencia actual en Eclipse, tal vez el entorno escogido hubiese sido Eclipse.

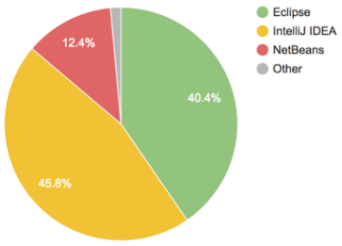


Ilustración 1. Porcentaje utilización IDE https://www.dunebook.com/intellij-vs-eclipse/

2.4 Fase de desarrollo

En este apartado se recogerán los aspectos más importantes de la planificación y desarrollo del proyecto, además se realizará una estimación del tiempo dedicado a realizar el programa, testearlo e implantarlo en la empresa.

Las fases:

- **Definición del proyecto.**
  Se va a realizar un análisis de requerimientos y decidir qué herramientas informáticas utilizar para el desarrollo y posterior implantación del software.

- **Estudio del software actual y análisis de requisitos.**
  Hacer un breve estudio del software que tienen actualmente funcionando para ver cómo mejorarlo y analizar los requisitos específicos de la empresa.

- **Diseño de la base de datos**
  Creación de los elementos de la base de datos necesarios para que los datos sean guardados y tengan consistencia. Tablas, "triggers", procedimientos…

- **Diseño y programación de artículos, familias, grupos familias, unidades de venta y clientes**
  Diseño y programación de las ventanas que dan configuración a los elementos anteriormente descritos.

- **Importador de artículos y clientes**
  Programación de un importador de artículos y clientes de la antigua base de datos a la nueva en formato de XLSX.

- **Diseño de la pantalla de ventas con albaranes, facturas y abonos**
  Programar la pantalla de ventas para poder realizar albaranes, facturas y poder abonarlos. También diseñar el aspecto visual para poder imprimirlas con Jaspersoft.

- **Pantalla de etiquetas e informes**
  Realizar los informes requeridos y la pantalla para la impresión de etiquetas en dos tamaños (pequeño y mediano). Ambos elementos se imprimirán mediante reports de Jaspersoft diseñados específicamente.

- **Otros elementos de configuración**
  Pantallas para configurar los datos de la empresa, el I.V.A. aplicado (en este caso 21%), las zonas de impresión, exportador, borrador masivo…

- **Cierre de año y copia de seguridad**
  Programar herramienta para poder realizar un cierre de año y copias de seguridad periódicas cada vez que se cierra el programa.

- **Testeo, creación de manual y puesta en marcha**
  Pruebas unitarias de todos los módulos generados, creación de un manual para el uso de cualquier persona y posterior instalación en la empresa.

Una vez planificadas todas las fases se hará una estimación del tiempo que se va a invertir en llevar este software y documentación a cabo.

Estimación de tiempo		
Orden		Tiempo(horas)
1	Definición del proyecto	10
2	Estudio del software actual y análisis de requisitos	10
3	Diseño de la base de datos	40
4	Diseño y programación de artículos, familias, grupos familias, unidades de venta y clientes	100
5	Importador de artículos y clientes	5
6	Diseño de la pantalla de ventas con albaranes, facturas y abonos	150
7	Pantalla de etiquetas e informes	40
8	Otros elementos de configuración	40
9	Cierre de año y copia de seguridad	10
10	Testeo, creación de manuales y documentación y puesta en marcha	20

Total horas	425

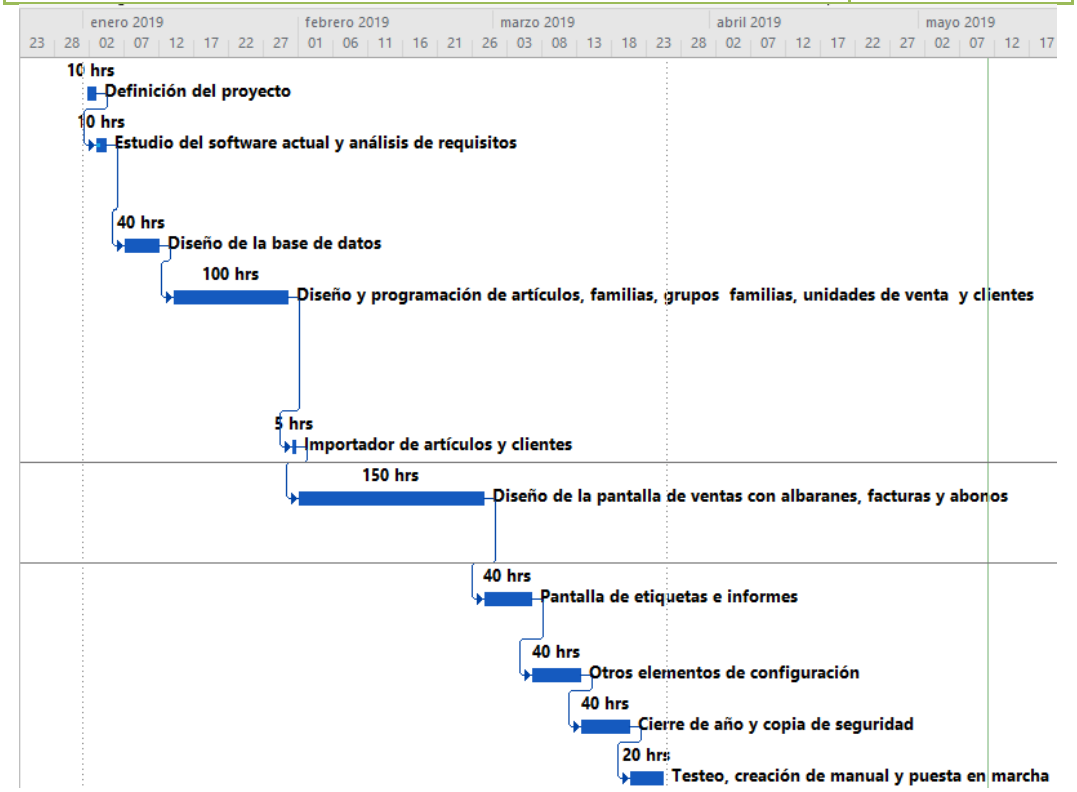


Ilustración 2. Diagrama de Gantt del proyecto de desarrollo

A raíz de la tabla anterior y del diagrama de Gantt se puede observar que más de la mitad de las horas invertidas en desarrollar este software están en el diseño y programación de los elementos esenciales (artículos, familias, grupos…) y en implementar la pantalla que se encarga de las ventas.

Capítulo 3: BBDD

En este capítulo se va a exponer la BBDD que se generó a raíz de todas las conversaciones y análisis con el cliente, las tablas que la componen y todas las operaciones que realiza de forma automática para el buen funcionamiento del software una vez implementado.

Cabe destacar que el esqueleto de la BBDD se generó antes de empezar el software y algunas de las operaciones auxiliares fueron incorporándose a medida que surgió la necesidad.

3.1 Definición de la BBDD

La base de datos consta de 17 tablas para toda la configuración del software empresarial. En la siguiente imagen se pueden ver las relaciones entre las tablas. Con el objeto de simplificar su comprensión, hay campos que no se muestran en el diagrama relacional.

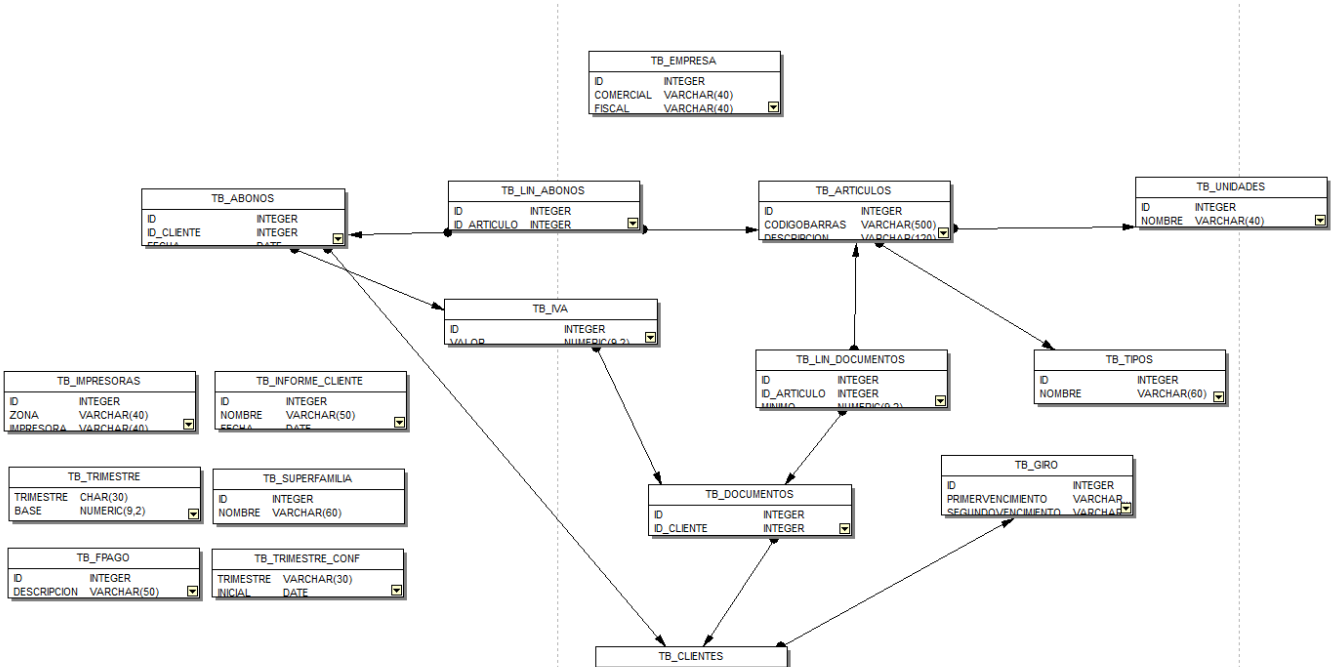


Ilustración 3. Esquema Relacional BBDD

Además, la base de datos cuenta con 13 “procedures”, 26 “triggers” y 13 “generators”.

En firebird no existe el concepto de “autoincremental” como si lo puede haber en MySql. En este motor de base de datos se hace de forma diferente, ya que se tiene que crear un “trigger” que se ejecute cuando se inserte una nueva fila en la tabla y de ahí llamar a un procedimiento que a su vez llamará al generador que será el encargado de guardarse el último número generado. Esto nos da mucho más control sobre los ID que se van a generar, pudiendo ser el orden requerido y teniendo siempre el último accesible fácilmente para las tablas relacionadas.

3.1.1 TB_Articulos

La tabla de los artículos contiene dos claves foráneas a las tablas TB_TIPOS y TB_UNIDADES. Básicamente esto da la funcionalidad de saber si el artículo en la pantalla de ventas se va a gestionar como lineal, metro cuadrado o unitario, además de saber a qué "familia" pertenece (en este caso TIPO).
Con este diseño, se cumpliría el requisito del cliente de poder facturar por esos tres tipos de unidades. Además, el campo "multiplofac" es un valor interno que utiliza la empresa para recalcular el precio en los metros lineales, otro requisito indispensable para el software.

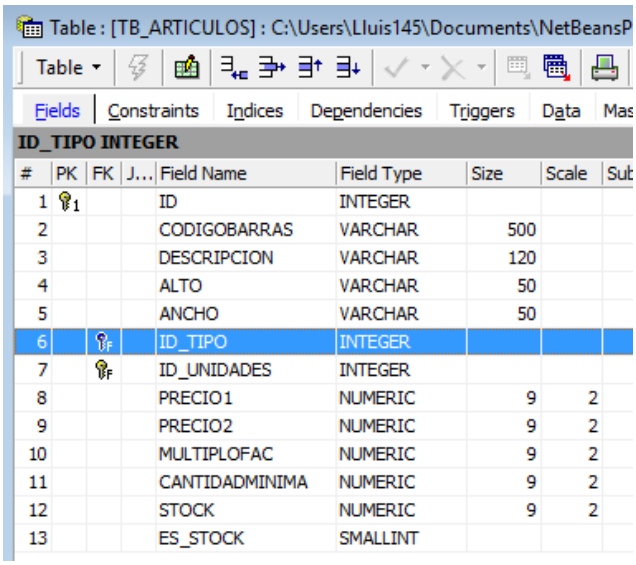


Table : [TB_ARTICULOS] : C:\Users\Lluis145\Documents\NetBeansP

Table | Fields | Constraints | Indices | Dependencies | Triggers | Data | Mas

ID_TIPO INTEGER

#	PK	FK	J...	Field Name	Field Type	Size	Scale	Sub
1	1			ID	INTEGER			
2				CODIGOBARRAS	VARCHAR	500		
3				DESCRIPCION	VARCHAR	120		
4				ALTO	VARCHAR	50		
5				ANCHO	VARCHAR	50		
6		F		ID_TIPO	INTEGER			
7		F		ID_UNIDADES	INTEGER			
8				PRECIO1	NUMERIC	9	2	
9				PRECIO2	NUMERIC	9	2	
10				MULTIPLOFAC	NUMERIC	9	2	
11				CANTIDADMINIMA	NUMERIC	9	2	
12				STOCK	NUMERIC	9	2	
13				ES_STOCK	SMALLINT			

Ilustración 4. TB_ARTICULOS

3.1.2 TB_Clientes

En la tabla siguiente se puede observar que lo único que tiene de especial es que tiene una clave foránea a la tabla TB_GIRO. El poder girar pagos (transferencia bancaria que se hace efectiva tras cierto tiempo pactado entre vendedor y comprador) a los clientes era algo que tenían en el software anterior, con lo que querían poder disponer de esa herramienta en el nuevo software. Más tarde se comprobó, que esta herramienta no se utilizó, a pesar de haberla incluido el cliente como requisito.

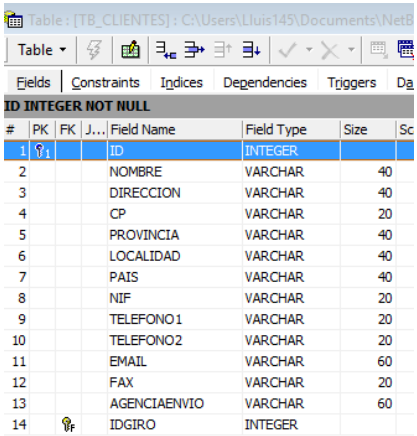


Table : [TB_CLIENTES] : C:\Users\Lluis145\Documents\NetB

Table | Fields | Constraints | Indices | Dependencies | Triggers | Da

ID INTEGER NOT NULL

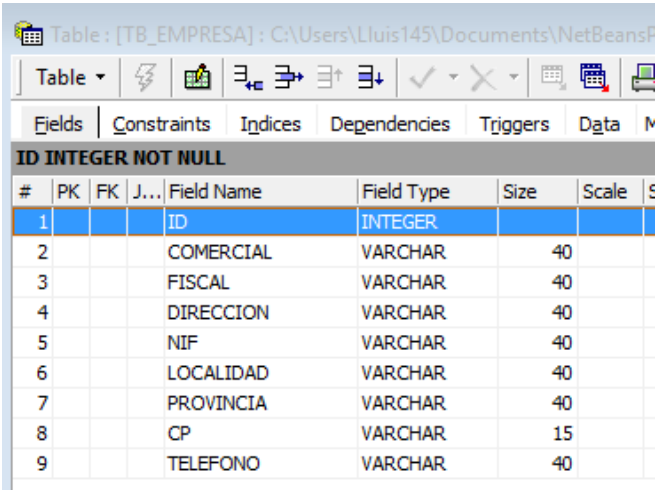
#	PK	FK	J...	Field Name	Field Type	Size	Sc
1	1			ID	INTEGER		
2				NOMBRE	VARCHAR	40	
3				DIRECCION	VARCHAR	40	
4				CP	VARCHAR	20	
5				PROVINCIA	VARCHAR	40	
6				LOCALIDAD	VARCHAR	40	
7				PAIS	VARCHAR	40	
8				NIF	VARCHAR	20	
9				TELEFONO1	VARCHAR	20	
10				TELEFONO2	VARCHAR	20	
11				EMAIL	VARCHAR	60	
12				FAX	VARCHAR	20	
13				AGENCIAENVIO	VARCHAR	60	
14		F		IDGIRO	INTEGER		

Ilustración 5 - TB_CLIENTES

3.1.3 TB_Empresa

Otro requisito indispensable para que el software satisficiese sus necesidades era que los datos comerciales que aparecían en las etiquetas y documentos de venta se pudieran cambiar con facilidad. Siendo tan sencillo como cambiar en una casilla el dato requerido y haciéndose efectivo al instante y no tener que depender de una empresa o persona ajena para que hiciese los cambios.

Esta tabla tiene la característica de que no tiene ninguna función tipo “Trigger” o “Procedure” asociada, su cambio se hará vía una operación SQL de “UPDATE”.



#	PK	FK	J...	Field Name	Field Type	Size	Scale
1				ID	INTEGER		
2				COMERCIAL	VARCHAR	40	
3				FISCAL	VARCHAR	40	
4				DIRECCION	VARCHAR	40	
5				NIF	VARCHAR	40	
6				LOCALIDAD	VARCHAR	40	
7				PROVINCIA	VARCHAR	40	
8				CP	VARCHAR	15	
9				TELEFONO	VARCHAR	40	

Ilustración 6. TB_EMPRESA

3.1.4 TB_Documentos

Este podría ser el elemento más importante de toda la BBDD ya que contiene todas las cabeceras de los documentos de venta, sean albaranes o facturas. Además, indica, si el documento ha sido abonado, el cliente al que hace referencia, el % de I.V.A. (que coge de la tabla de I.V.A.), si tiene recargo de equivalencia o no y otros datos.

Aunque existen algunas referencias a otras tablas que no están creadas explícitamente, se manejarán desde código, como podría ser el ID_ABONO, ID_IVA, ID_FPAGO, etc.

Table : [TB_DOCUMENTOS] : C:\Users\Lluis145\Documents\NetBe

Table | Fields | Constraints | Indices | Dependencies | Triggers | Data | M

TOTALDTO NUMERIC(9,2)

#	PK	FK	J...	Field Name	Field Type	Size	Scale	S
1	1			ID	INTEGER			
2				ID_CLIENTE	INTEGER			
3				FECHA	DATE			
4				CODIGO_DOC	SMALLINT			
5				REMITENTE	SMALLINT			
6				ID_IVA	INTEGER			
7				TOTAL	NUMERIC	9	2	
8				BASEIMPONIBLE	NUMERIC	9	2	
9				IMPORTE_IVA	NUMERIC	9	2	
10				RECARGO	NUMERIC	9	2	
11				IMPORTE_RECARGO	NUMERIC	9	2	
12				ID_FPAGO	INTEGER			
13				NUMALBARAN	INTEGER			
14				NUMFACTURA	INTEGER			
15				NUMPEDIDO	INTEGER			
16				ID_ABONO	INTEGER			
17				TOTALDTO	NUMERIC	9	2	

Ilustración 7. TB_DOCUMENTOS

3.1.5 TB_Lin_Documentos

En esta tabla se guardan los productos (líneas) agregados al documento; una línea por cada producto que se añada. Aquí se almacenan los totales por producto tanto en descuento, como en cantidades. Junto a la tabla de TB_DOCUMENTOS, es una de las más importantes.

Table : [TB_LIN_DOCUMENTOS] : C:\Users\Lluis145\Documents\

Table | Fields | Constraints | Indices | Dependencies | Triggers | Data

ID INTEGER NOT NULL

#	PK	FK	J...	Field Name	Field Type	Size	Scale
1	1			ID	INTEGER		
2		F		ID_ARTICULO	INTEGER		
3				MINIMO	NUMERIC	9	2
4				CANTIDAD	NUMERIC	9	2
5				PRECIO	NUMERIC	9	2
6				DTO	NUMERIC	9	2
7				IMPORTE	NUMERIC	9	2
8		F		ID_FACTURA	INTEGER		
9				METROS	NUMERIC	9	2
10				ANCHO	NUMERIC	9	2
11				ALTO	NUMERIC	9	2

Ilustración 8. TB_LIN_DOCUMENTOS

3.1.6 TB_Abonos y TB_Lin_Abonos

Estas dos tablas son prácticamente idénticas a las explicadas en los dos apartados anteriores. La única diferencia es que tanto el precio como las cantidades estarán en negativo ya que serán documentos abonados por un cliente final que ya no tendrán validez.

3.1.7 Tb_Informe_cliente y Tb_trimestre

Estas dos tablas son especiales porque son tablas “temporales”, se utilizarán, como se verá en capítulos posteriores, para rellenar los informes impresos que el cliente quiera tener en papel. Para ello el software insertará los valores del informe calculado en pantalla en esta tabla y posteriormente el “Report” (informe) lo recogerá de ahí.
Cada vez que se calcule un informe, se borrarán las filas de la tabla y se insertarán de nuevo.
En este punto, en el caso de trabajo concurrente en varias máquinas podría haber un problema; como en esta empresa, únicamente trabajan en un ordenador no pasará nada. La solución en el caso de querer implementarlo para varias máquinas sería crear una tabla más con usuarios y contraseñas y estas tablas crearlas en “caliente” con el nombre de la tabla concatenado con el id de usuario. De momento no es un requisito.

3.1.8 Otras tablas

Las tablas que restan son tablas que no tienen ninguna relevancia que requiera de explicación, ya que son tablas con datos simples de entrada, sin ninguna operativa especial que las defina, y, por lo tanto, no se van a explicar explícitamente.

3.2 Triggers y procedures

Todas las tablas tienen dos “triggers” como mínimo para controlar el ID autonumérico, uno se hace antes de insertar y el otro después de borrar. Básicamente cambian el valor del “Generator” sumándole o restándole uno, para así siempre tener los campos actualizados. Desde código se controla que no haya problemas con el borrado, ya que no se deja borrar un valor de una tabla si depende de otra.

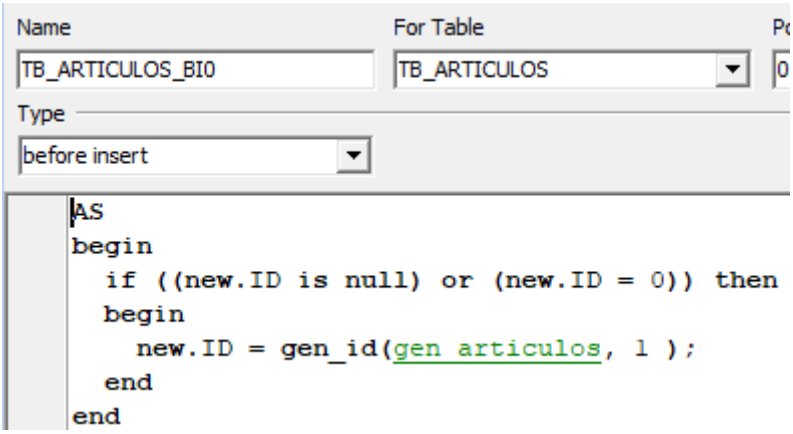


Ilustración 9. Trigger antes de insertar

Y después del borrado.

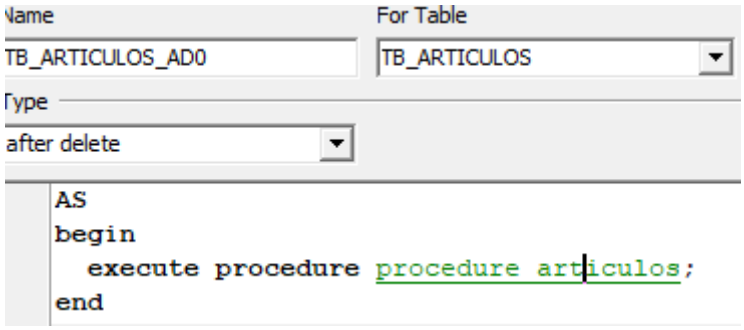


Ilustración 10. Trigger después del borrado

```
begin
 execute statement 'select max(id) from tb_articulos' into :max_id;
  execute statement 'select count(id) from tb_articulos' into :aux;
  if(aux=0) then
  max_id=0;
  execute statement 'set generator gen_articulos to ' || :max_id;
  suspend;
end
```

Ilustración 11. Procedure después del borrado

Capítulo 4: Programación

En este capítulo se mostrará como es el programa visualmente (solo las pantallas principales) y se describirán los principales algoritmos utilizados para nutrir al software de datos.
Cabe destacar que el código completo de esta aplicación de gestión empresarial consta de unas 5000 líneas de código.

Tal y como ya se ha explicado, para definir el aspecto visual del software se realizó un análisis de requerimientos a través de entrevistas a los usuarios que utilizaban el software anterior, para así adaptarlo a sus necesidades, facilitándoles el trabajo desde un principio y mejorando los aspectos clave que su anterior software no cumplía.

La primera de las tareas en cuanto a la programación era generar una pantalla principal sencilla pero elegante, para que, si alguna persona ajena a la empresa viese el programa, viese un software profesional con los colores y logo que representan a la marca.

Para ello, cuenta con una pantalla principal compuesta por una imagen de fondo de pantalla (que se puede cambiar dinámicamente) y una serie de menús que dan acceso a todo el software objeto de este TFG.

Recuerde que por razones de confidencialidad se han omitido datos e imágenes y logos que puedan identificar la empresa objeto de este TFG.

Ilustración 12. Pantalla inicio

Está pantalla es la que contiene el “Main” de Java (el “main” de java es el programa principal a partir del que se cargan todos los demás) y por lo tanto la que hace los llamamientos a las otras pantallas de la siguiente forma:

```
private void jMenuItem11ActionPerformed(java.awt.event.ActionEvent evt) {
    gestionArticulos ven = new gestionArticulos();
    ven.setVisible(true);
}
```

Ilustración 13. Inicio nueva pantalla

La gran ventaja de utilizar Netbeans, es que los eventos visuales los genera el propio entorno de desarrollo. Esto facilita en gran medida la tarea de programar nuevas pantallas y ayuda a ver visualmente el resultado. Una vez se navega a través de la pantalla principal se puede observar que en este software hay seis tipos de pantallas:

1. Mantenimiento general (artículos, clientes, familias…)
2. Mantenimiento simple
3. Etiquetas
4. Documentos de venta
5. Informes
6. Importador/Exportador

4.1 Mantenimiento general

Como se acordó en las reuniones, las pantallas de mantenimiento general siempre iban a tener el mismo diseño divididas en dos pestañas.
La primera pestaña, “LISTA”, contiene como elemento principal una tabla con todos los datos de la base de datos. Para poder filtrar esta tabla, existe un campo de búsqueda en el cual se puede buscar por cualquier valor que aparezca en la pantalla.
Para acceder a la segunda pestaña, “FICHA”, se puede acceder al pulsar el botón “NUEVO” o pulsando dos veces sobre cualquier elemento de la tabla.
Una particularidad del mantenimiento general es que, dependiendo de la pantalla, se abrirá en modo pantalla completa o en modo ventana en el centro de la pantalla.

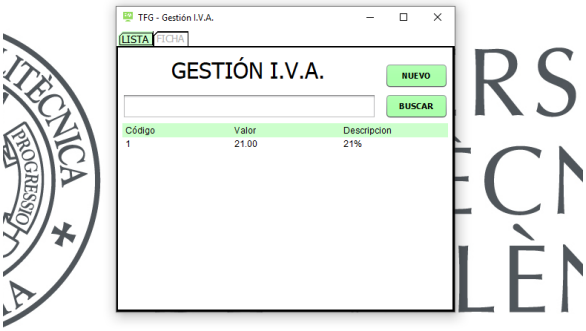


Ilustración 14. Mantenimiento General - Lista

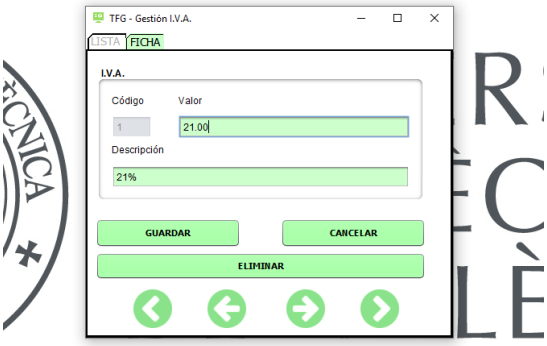


Ilustración 15. Mantenimiento General - Ficha

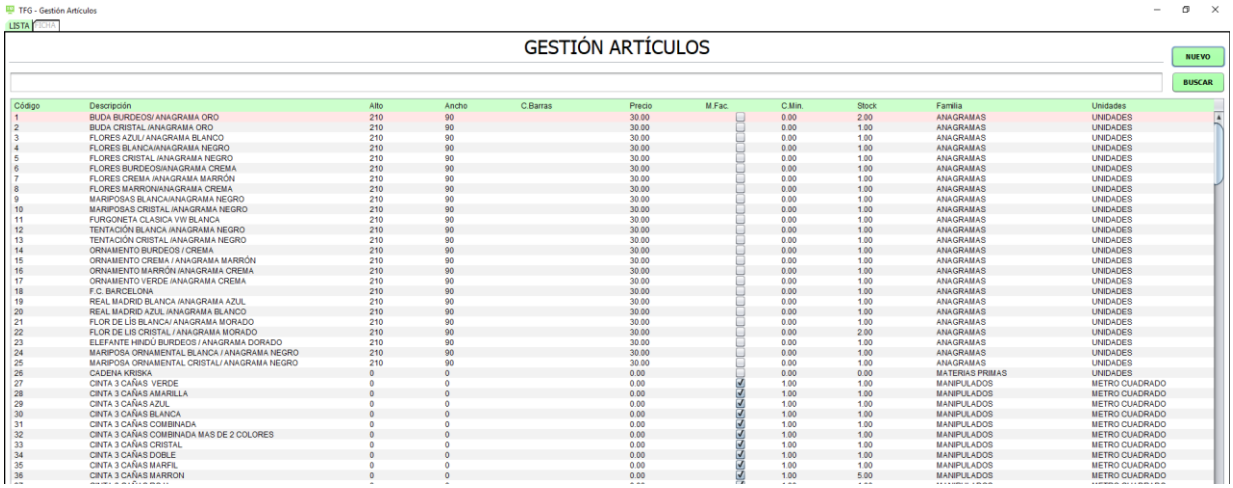


Ilustración 16. Mantenimiento general - Artículos

La pestaña “FICHA” siempre cuenta con tres botones para “GUARDAR”, “CANCELAR” o “ELIMINAR”, además de los botones de navegación de primero, anterior, siguiente y último.

4.1.1 Botón Buscar

El botón “Buscar” tiene la siguiente programación (también hay un evento para el botón “Enter” del teclado):

```
try {
    vaciarTabla();
    aux = listaBuscar();
    rellenarTabla(aux);
    buscar = true;

} catch (SQLException ex) {
    Logger.getLogger(gestionIva.class.getName()).log(Level.SEVERE, null, ex);
}
```

Ilustración 17. Método buscar

El funcionamiento general de este botón es el siguiente: lo primero que hace es vaciar la tabla; una vez vaciada, llama a la función “listaBuscar()” que hace una operación de tipo “Select” a la base de datos.
En el caso particular de la pantalla de I.V.A. simplemente buscará por los campos valor y descripción y en la pantalla de gestión de artículos buscará por todos los elementos que se pueden ver en la tabla.
Para ello en la “Select” se utiliza la cláusula “OR” para concatenar los distintos filtros que se quieren hacer; siempre se ordena por “ID ASC”. La empresa decidió que así debía ser este el funcionamiento:

```
String seleccion = "SELECT * FROM tb_iva where valor CONTAINING  '" + ent + "'"
        + "or descripcion CONTAINING  '" + ent + "' order by id asc";
```

Ilustración 18. Sentencia SQL para buscar

El método “rellenarTabla”, como su nombre indica, tiene como funcionalidad añadir el resultado del “ResultSet” generado por la base de datos al “Jtable”, para ello se crea un Objecto “Object” y se le añaden todos los datos del “ResultSet”:

```
private void rellenarTabla(ResultSet rs) {
    try {
        while (rs.next()) {
            Object[] fila = new Object[3];//Creamos
            // de la consulta
            fila[0] = rs.getInt("id"); //Lo que hay
            fila[1] = rs.getString("valor");
            fila[2] = rs.getString("descripcion");
            modelo.addRow(fila); // Añade una fila a
        }
        tblClientes.updateUI();//Actualiza la tabla
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Ilustración 19. Rellenar Tabla

Durante las distintas funciones y métodos se va tratando la excepción de SQL para saber si en algún momento la conexión se ha perdido y por lo tanto el software no puede continuar funcionando.

4.1.2 Botón “nuevo”

El botón “nuevo” simplemente desactiva la pestaña “LISTA”, activa la pestaña “FICHA” y redirige al usuario a la segunda pestaña:

```
btnFicha.setSelectedIndex(1);
btnFicha.setEnabledAt(0, false);
btnFicha.setEnabledAt(1, true);
txtCodigo.setText("0");
```

Ilustración 20. Botón nuevo

El parámetro que se le asigna al “txtCodigo” es simplemente para saber que el programa está en modo “añadir” y no en modo “modificar”, como se controlan todos los códigos a través de “Procedures” en Firebird, no hay problema ya que si es un registro “modificar” siempre tendrá valor.

4.1.3 Doble clic tabla

Si el usuario hace doble clic sobre un elemento de la tabla, el programa hará la misma funcionalidad que si se pulsa el botón “nuevo”, en cambio ahora cargará los elementos que hay en la tabla en los campos.

```
tblIva.addMouseListener(
        new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2) {
            btnFicha.setSelectedIndex(1);
            btnFicha.setEnabledAt(0, false);
            btnFicha.setEnabledAt(1, true);
            int row = tblIva.getSelectedRow();
            txtCodigo.setText(tblIva.getValueAt(row, 0) + "");
            txtValor.setText(tblIva.getValueAt(row, 1) + "");
            txtDescripcion.setText(tblIva.getValueAt(row, 2) + "");
            comprobarNull();
```

Ilustración 21. Evento Doble clic

El método “comprobarNull” simplemente cambiará el texto “null” por “ ” en el caso de que alguno de los campos de la tabla no tenga valor, para que la pantalla sea más amigable con el usuario.

4.1.4 Botón “guardar”

Primero comprobará si estamos en modo “modificar” o “añadir” (para ello mirará si el código es cero o no), inmediatamente llamará al método que corresponda para finalmente bloquear la pestaña “FICHA”, iniciará la tabla y vaciará los campos de la “FICHA”:

```
if (txtCodigo.getText().equalsIgnoreCase("0")) {
    insertarIva();
} else {
    guardarIva();
}
btnFicha.setEnabledAt(1, false);
iniciarTabla();
vaciarCampos();
```

Ilustración 22. Método guardar

Evidentemente, el método “insertarIva” hará un “INSERT” en la BBDD y el “guardarIva” un “UPDATE” a través del campo “ID”.

```
private void insertarIva() {
    try {
        String seleccion;
        seleccion = "insert into tb_iva (valor, descripcion)\n"
                + "  values ('" + txtValor.getText() + "', '" + txtDescripcion.getText() + "')";
        conn.prepareStatement(seleccion).executeUpdate();
    } catch (SQLException ex) {
        Logger.getLogger(gestionIva.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Ilustración 23. Insertar elemento

```
private void guardarIva() {
    String seleccion;
    try {
        seleccion = "update tb_iva set valor=?, descripcion=? where id=?";
        PreparedStatement update = conn.prepareStatement(seleccion);
        update.setString(1, txtValor.getText());
        update.setString(2, txtDescripcion.getText());
        update.setInt(3, Integer.parseInt(txtCodigo.getText()));
        update.executeUpdate();
    } catch (SQLException ex) {
        Logger.getLogger(gestionIva.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Ilustración 24. Guardar elemento

En todas las pantallas de mantenimiento general son los mismos métodos, pero con distintos SQL para nutrir a la ventana de datos.

4.1.5 Botón cancelar

El botón cancelar es el más simple de todos ya que solo hará el cambio entre pestañas y vaciará los campos.

```
btnFicha.setSelectedIndex(0);
btnFicha.setEnabledAt(0, true);
btnFicha.setEnabledAt(1, false);
vaciarCampos();
```

Ilustración 25. Botón cancelar

4.1.6 Botones navegación

Cabe destacar que los cuatro botones de navegación tienen un comportamiento similar. Estos botones lo primero que hacen es mover el "ResultSet" (elemento que contiene una matriz con los datos de la base de datos), para ello comprueban si la tabla inicial está filtrada o no y dependiendo de ello devuelven un "ResultSet" ya movido a la posición necesaria de lo filtrado o de la lista inicial.

```
private ResultSet moverRS() {
    ResultSet rs = null;
    try {
        boolean encontrado = false;
        if (buscar == true) {
            rs = listaBuscar();
        } else {
            rs = listaInicial();
        }
        while (rs.next() && !encontrado) {
            if (rs.getString("id").equalsIgnoreCase(txtCodigo.getText())) {
                encontrado = true;
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger(gestionIva.class.getName()).log(Level.SEVERE, null, ex);
    }
    return rs;
}
```

Ilustración 26. Botones navegación

Después dependiendo del botón que se haya pulsado se utilizará una función del "ResultSet" u otra, siendo estas: primero → "first()", anterior → "previous()", siguiente → "next()" o último → "last()".

Una vez movido el “ResultSet” a la posición requerida se llamará al método “rellenarFicha” pasándole como argumento el “ResultSet” movido previamente.

```
private void rellenarFicha(ResultSet rs) {

    try {
        txtCodigo.setText(rs.getString("id"));
        txtValor.setText(rs.getString("valor"));
        txtDescripcion.setText(rs.getString("descripcion"));
    } catch (SQLException ex) {
        Logger.getLogger(gestionIva.class.getName()).log(Leve
    }
```

Ilustración 27. Rellenar ficha

Como se observa, simplemente se rellenan los datos de la pestaña “FICHA” con los datos del "ResultSet” movido.

4.1.7 Botón “eliminar”

Este botón lo primero que hará es comprobar si estamos en modo “añadir” o “modificar”. Solo funcionará si estamos en el segundo modo, “modificar”.
En el caso del I.V.A. no se puede borrar el primer registro ya que si no se podría tener facturas sin I.V.A. y esto conllevaría a un error, por lo que se comprueba si es el valor 1 o no. Si no es el caso, se pregunta si quiere eliminar el valor, si la respuesta es positiva, se llama al método “eliminarIva()”.

```
if (!txtCodigo.getText().equalsIgnoreCase("0")) {
    if (txtCodigo.getText().equalsIgnoreCase("1")) {
        JOptionPane.showMessageDialog(null, "No se puede borrar el I.V.A.
    } else if (JOptionPane.showConfirmDialog(rootPane, "¿Desea realmente
            "Eliminar I.V.A.", JOptionPane.YES_NO_OPTION) == JOptionPane.
        btnFicha.setEnabledAt(1, false);
        eliminarIva();
    }

}
```

Ilustración 28. Botón eliminar

Este método simplemente hace una petición SQL “DELETE” a la base de datos pasándole como valor el valor del campo “ID”.

```
private void eliminarIva() {
    try {
        String seleccion = "delete from tb_iva where id=?";
        PreparedStatement delete = conn.prepareStatement(seleccion);
        delete.setInt(1, Integer.parseInt(txtCodigo.getText()));
        delete.executeUpdate();

        JOptionPane.showMessageDialog(null, "Eliminado correctamente");
    } catch (SQLException ex) {
        Logger.getLogger(gestionIva.class.getName()).log(Level.SEVERE, null, ex);
    }
    iniciarTabla();
    vaciarCampos();
}
```

Ilustración 29. Eliminar elemento

4.2 Mantenimiento simple

Se denomina a este tipo de mantenimiento como “mantenimiento simple” porque no tienen ni barra de búsqueda ni dos pestañas, tan solo un título, campos para insertar o modificar datos y un botón de guardar.
El ejemplo más claro podría ser “gestión de impresoras” o “gestión de empresa” donde se actualizan las impresoras que va a utilizar el programa para cada elemento o los datos que va a mostrar en los “Reports” (informes) de JasperSoft.

4.2.1 Gestiones impresoras

La pantalla de gestión de impresoras es una simple pantalla con tres desplegables del tipo “JComboBox” que se rellenan dinámicamente cada vez que se accede a la pantalla, seleccionando en la lista el valor que haya guardado en la BBDD.

Ilustración 30. Gestiones impresoras

Para saber las impresoras instaladas en el sistema se utiliza la siguiente llamada que devuelve una matriz de “PrintService” a la que se le puede hacer un “getName()” para recoger el nombre de cada impresora.

```
services = PrintServiceLookup.lookupPrintServices(null, null);
```

Ilustración 31. Impresoras instaladas en el sistema

Después se llama al método “rellenarCombos()”:

```
private void llenarCombos() {
    try {
        String etiquetas, docs, informes;
        String seleccion = "Select * from tb_impresoras";

        ResultSet rs = conn.prepareStatement(seleccion).executeQuery();
        rs.next();
        etiquetas = rs.getString("impresora");
        rs.next();
        docs = rs.getString("impresora");
        rs.next();
        informes = rs.getString("impresora");
        for (int i = 0; i < services.length; i++) {
            cbDocVentas.addItem(services[i].getName());
            cbEtiquetas.addItem(services[i].getName());
            cbInformes.addItem(services[i].getName());
            if (docs.equalsIgnoreCase(services[i].getName())) {
                cbDocVentas.setSelectedIndex(i);
            }
            if (etiquetas.equalsIgnoreCase(services[i].getName())) {
                cbEtiquetas.setSelectedIndex(i);
            }
            if (informes.equalsIgnoreCase(services[i].getName())) {
                cbInformes.setSelectedIndex(i);
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger(gestionImpresoras.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Ilustración 32. Rellenar desplegables

En “rellenarCombos()”, primero se recogen los tres nombres de las impresoras guardadas en tres variables de tipo “String”. A continuación, mediante un bucle de tipo “for”, se recorre todo el array de las impresoras instaladas en el sistema y añadirá a los “combobox” el nombre de la impresora de sistema. Posteriormente si el nombre coincide, se seleccionará ese elemento en la matriz, sino no hará nada, con lo que si no está guardada o la que está guardada no coincide con la lista que devuelve el Sistema Operativo, se seleccionará la primera por defecto.

Además, está clase también contiene otro método para guardarlas impresoras, el cual únicamente (previamente cargas en la BBDD) actualizará los registros de la base de datos.

```
private void guardarImpresoras() {
    try {
        String seleccion = "update tb_impresoras set \n"
                + " impresora= '" + cbEtiquetas.getSelectedItem().toString() + "' where id=1";

        conn.prepareStatement(seleccion).executeUpdate();

        seleccion = "update tb_impresoras set \n"
                + " impresora= '" + cbDocVentas.getSelectedItem().toString() + "' where id=2";

        conn.prepareStatement(seleccion).executeUpdate();
        seleccion = "update tb_impresoras set \n"
                + " impresora= '" + cbInformes.getSelectedItem().toString() + "' where id=3";

        conn.prepareStatement(seleccion).executeUpdate();
        JOptionPane.showMessageDialog(null, "Guardado con éxito!");
        this.setVisible(false);
    } catch (SQLException ex) {
        Logger.getLogger(gestionImpresoras.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Ilustración 33. Guardar impresoras

4.2.2 Datos de la empresa

Esta pantalla contiene los datos que la empresa puede gestionarse sin tener que cambiar ningún informe o documento de venta, ya que se guardarán en la BBDD que es de donde se nutren la totalidad de los documentos impresos.

En el caso de que el software se hubiese querido comercializar y no fuese para una empresa en concreto, estos datos se hubiesen tenido que añadir a un archivo de licencia encriptado y montar un sistema de licencias para así poder restringir el uso de personas no autorizadas, pero como no se dio el caso, fue la manera más simple de darles autocontrol sobre el programa.

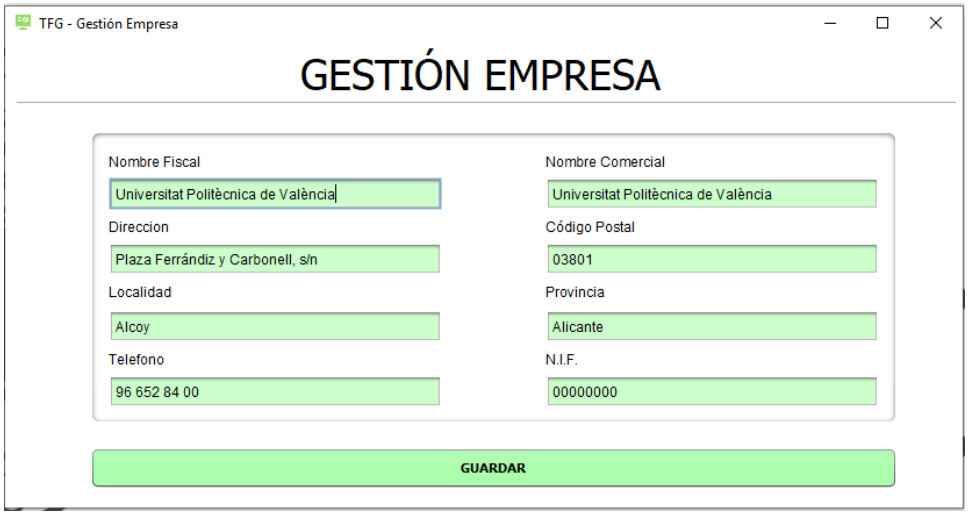


Ilustración 34. Datos de la empresa

4.2.3 Opciones

La pantalla de opciones contiene cuatro pestañas:

- “borrados”,
- “copia seguridad”,
- “cierre de año”
- “I.V.A. Defecto”.

En este tipo de mantenimiento no hay inserción de datos, simplemente utilidades de cara a que el usuario tenga mayor versatilidad utilizando el programa.

La pestaña de “borrados” le permite al usuario borrar los albaranes entre dos fechas (que no hayan sido facturados), por ejemplo, por si hay algún error o por si se quiere hacer una limpieza de datos e inicializar la base de datos.
Además, permite borrar todos los clientes y artículos, como el texto indica, únicamente antes de haber utilizado el programa o en el caso de que esté limpio de documentos de venta.

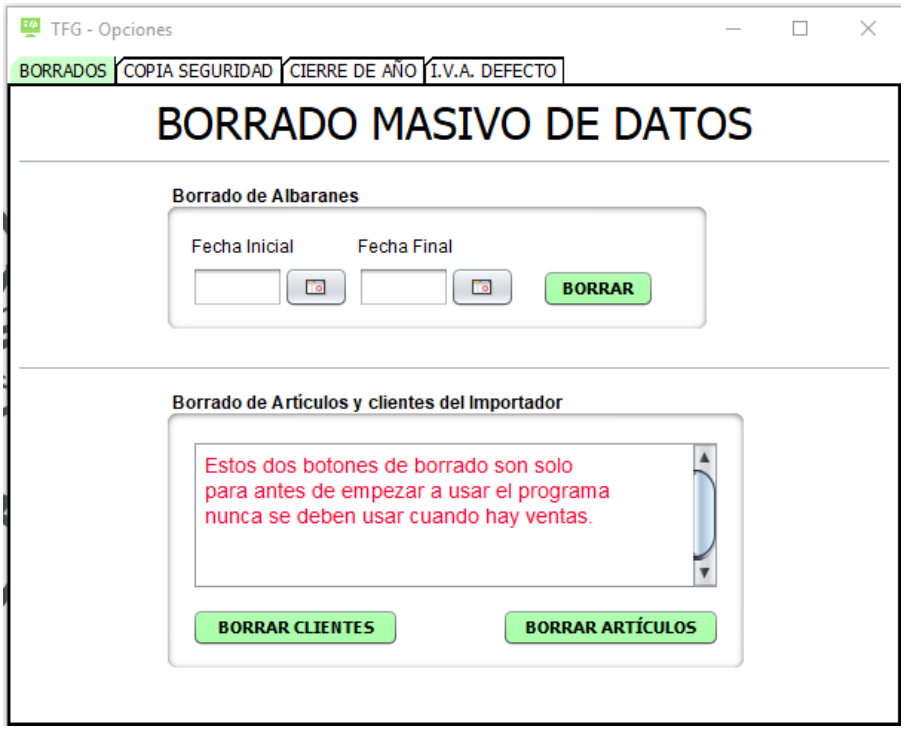


Ilustración 35. Borrado masivo de datos

La pestaña de “copia de seguridad” permite configurar la ruta donde se requiere que se haga la copia de seguridad. Puede ser cualquier ruta accesible en el sistema, pudiendo ser una carpeta compartida, una carpeta de Dropbox, Google Drive o cualquier otro servicio en la nube. Al hacer clic sobre el recuadro verde, se abrirá un explorador de archivos donde seleccionar la ruta.

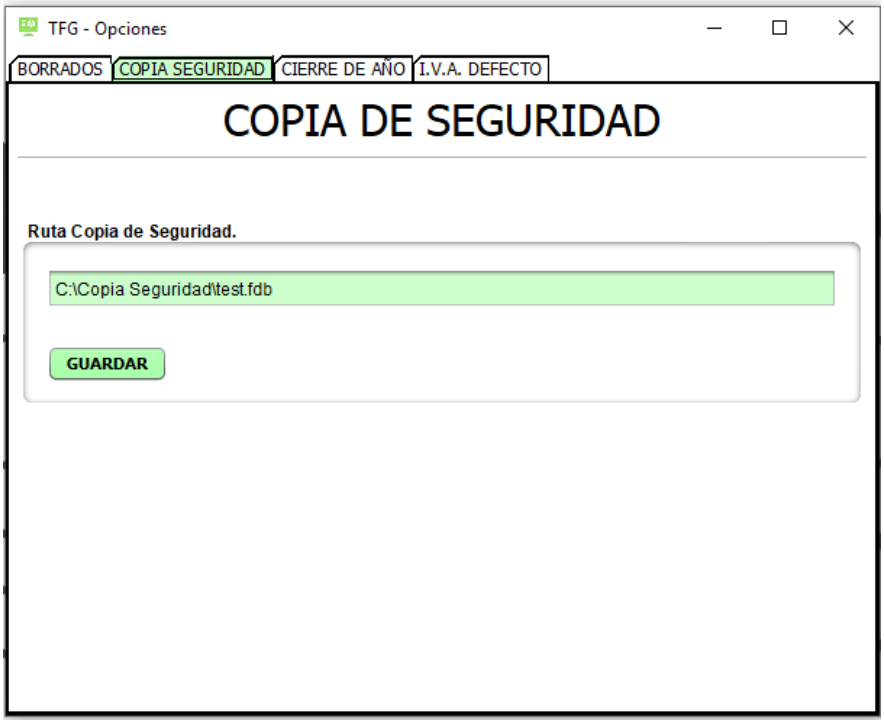


Ilustración 36. Copia de seguridad

El “cierre de año” permitirá hacer un cierre anualmente. La operativa que hará será copiar la base de datos actual a un directorio dentro de la raíz del programa y posteriormente un borrado de todos los documentos de venta, quedando el año siguiente limpio de documentos y pudiendo empezar de 0.

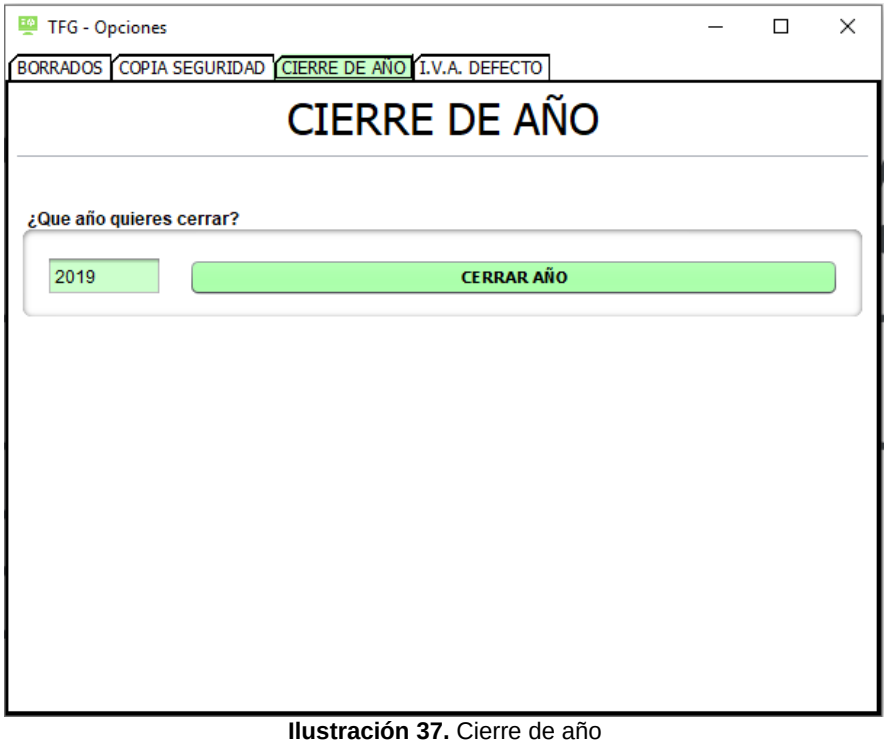


Ilustración 37. Cierre de año

La última pestaña permitirá seleccionar cual es el I.V.A. por defecto que se requiere en las facturas pudiendo cambiarlo por los otros valores de la ventana explicada en apartados anteriores.

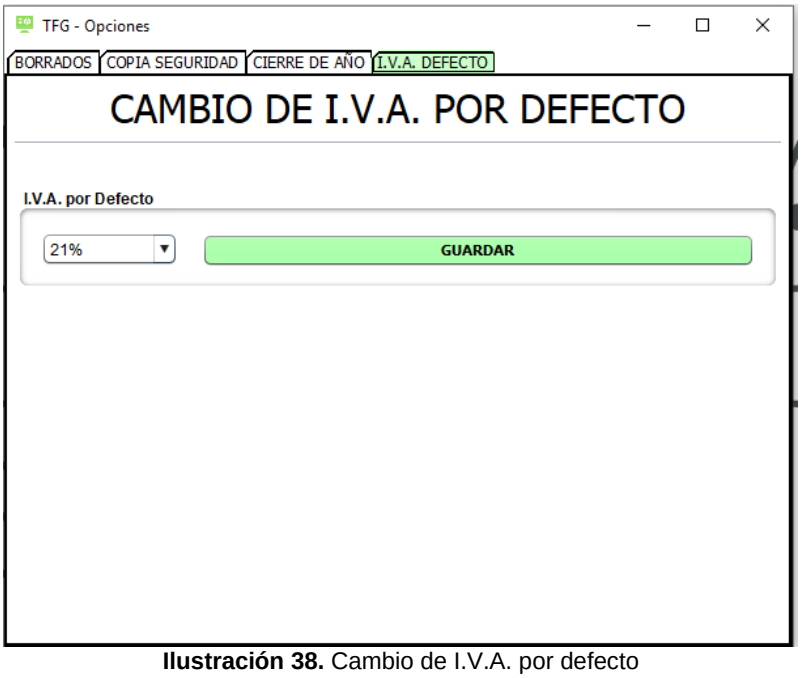


Ilustración 38. Cambio de I.V.A. por defecto

4.3 Etiquetas

La empresa por su forma de trabajar tiene dos etiquetas, unas que contiene un código de barras, para etiquetar individualmente cada producto y otras que son más grandes, que sirven para etiquetar el producto una vez se envía a sus clientes.

La primera etiqueta, se gestiona de manera individual, desde un menú al que se puede acceder desde “Facturación” → “Etiquetas Cod. Barras”. La otra etiqueta se gestiona dentro de cada “Documento de Venta”.

En las reuniones que se mantuvieron con el cliente, se decidió que la pantalla de gestión de “etiqueta pequeña” (como la llaman ellos), debería contener cinco botones.

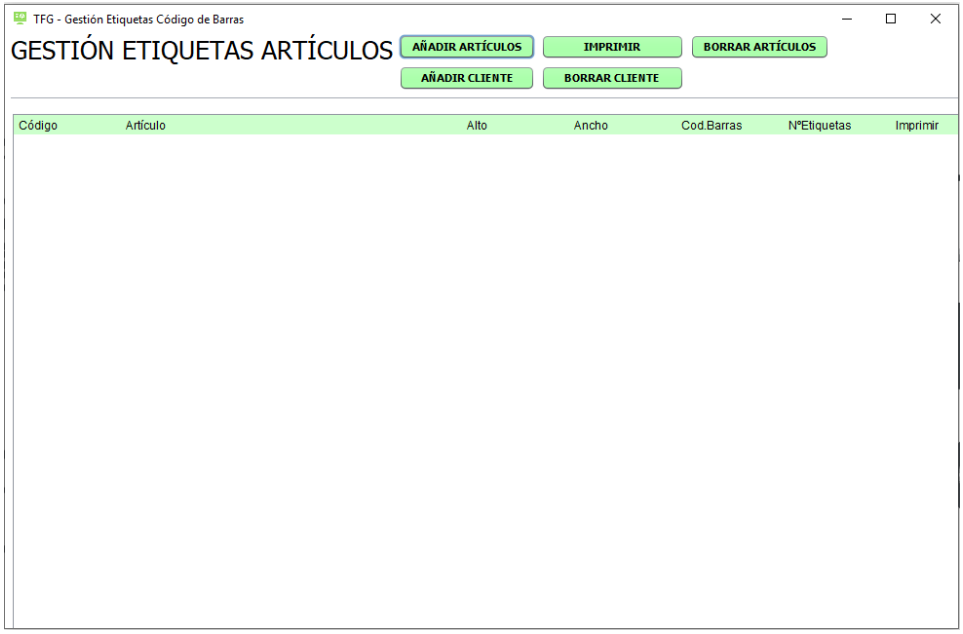


Ilustración 39. Gestión etiquetas

4.3.1 Botón añadir artículos

Al pulsar el botón añadir artículos se abrirá un “mantenimiento general”, pero solo en modo “LISTA”, donde el usuario puede seleccionar varios artículos y añadirlos a la gestión de etiquetas. También tiene un botón Buscar para poder filtrar los resultados.

Por motivos de protección de datos se han ocultado los datos de los artículos.

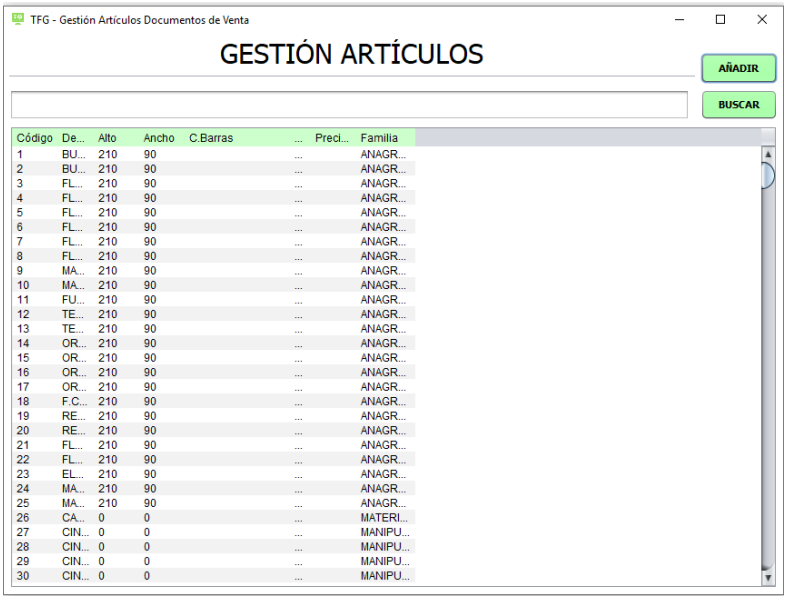


Ilustración 40. Añadir artículo a etiqueta

En el aspecto de programación al pulsar el botón añadir artículos se hará lo siguiente:

```
private void btnAñadirArticulosActionPerformed(java.awt.event.ActionEvent evt) {
    gestionArticulosExtra ven = new gestionArticulosExtra();
    ven.ra(tblLineas, true);
    ven.setVisible(true);
}
```

Ilustración 41. Botón añadir artículo

Se le pasará al método “ra” la tabla sobre la que tiene que añadir las líneas de los artículos seleccionados y un valor booleano para indicarle que es la pantalla etiquetas. Esto se ha realizado así porque también se utilizará esta misma pantalla para añadir artículos en la pantalla de documentos de venta.

Puesto que los mantenimientos generales están explicados anteriormente, en este apartado no se va a explicar cómo se listan los artículos, simplemente se explicará que es lo que el botón añadir hace en este caso.

```
int row = 0;
for (int i = 0; i < tblClientes.getSelectedRows().length; i++) {
    row = tblClientes.getSelectedRows()[i];
    if (!etiqueta) {
        añadirArticulos(Integer.parseInt(tblClientes.getValueAt(row, 0) + ""));
    } else {
        añadirEtiqueta(Integer.parseInt(tblClientes.getValueAt(row, 0) + ""));
    }
}
this.setVisible(false);
```

Ilustración 42. Bucle para añadir los artículos

Como puede haber varias líneas seleccionadas, el código realizará un bucle de tipo “for” de toda la matriz que ha seleccionado. En este caso como el booleano está a True entrará en el método “añadirEtiqueta”, pasándole como valor el primer valor de la fila.

```
public void añadirEtiqueta(int id) {
    DefaultTableModel defaultTableModel = (DefaultTableModel) articulosExtra.getModel();
    try {
        String seleccion = "select * \n"
                + "from tb_articulos where id=" + id;

        PreparedStatement ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

        ResultSet rs = ps.executeQuery();
```

Ilustración 43. Consulta SQL para añadir el artículo

Posteriormente, se hará una consulta a la base de datos sobre el campo “ID”, para recoger todos los valores necesarios.

```
        ResultSet rs = ps.executeQuery();
        while (rs.next()) {
            Object[] fila = new Object[13];//Creamos un Objeto con tantos parámetros como dato
            fila[0] = rs.getString("id");
            fila[1] = rs.getString("descripcion");
            fila[2] = rs.getString("alto");
            fila[3] = rs.getString("ancho");
            fila[4] = rs.getString("codigobarras");
            fila[5] = "1.0";
            fila[6] = true;
            defaultTableModel.addRow(fila); // Añade una fila al final del modelo de la tabla
        }
        articulosExtra.setModel(defaultTableModel);
    } catch (SQLException ex) {
        Logger.getLogger(gestionFacturas.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Ilustración 44. Bucle añadir artículo a la tabla

Y por último se añaden a la tabla pasada por el método de llamada. El añadir etiqueta se podría haber solucionado de otra forma, ahorrándose esa llamada a la BBDD y recogiendo los datos desde el “Jtable”, pero se optó por hacerlo de esta forma para asegurarnos de que los datos eran correctos, ya que, aunque actualmente solo funciona

en un equipo, se podría instalar en múltiples equipos y de esta forma se asegurarían de que los datos siempre son correctos.

4.3.2 Añadir cliente

El botón añadir cliente funcionará de forma parecida al explicado en el apartado anterior, pero con algunas diferencias. Este botón simplemente añadirá un texto a todas las etiquetas. En las reuniones que se mantuvieron con el cliente, se acordó que no tenía que aparecer en la pantalla una vez añadido, pero sí que apareciese en las etiquetas. Este fue el motivo por el que se utilizó un "JLabel" con la visibilidad oculta. Otra de las diferencias es que la tabla del mantenimiento de Clientes solo dejará seleccionar uno y no múltiples como en los artículos.

Por motivos de protección de datos y confidencialidad se han ocultado los datos de los clientes en la siguiente pantalla.

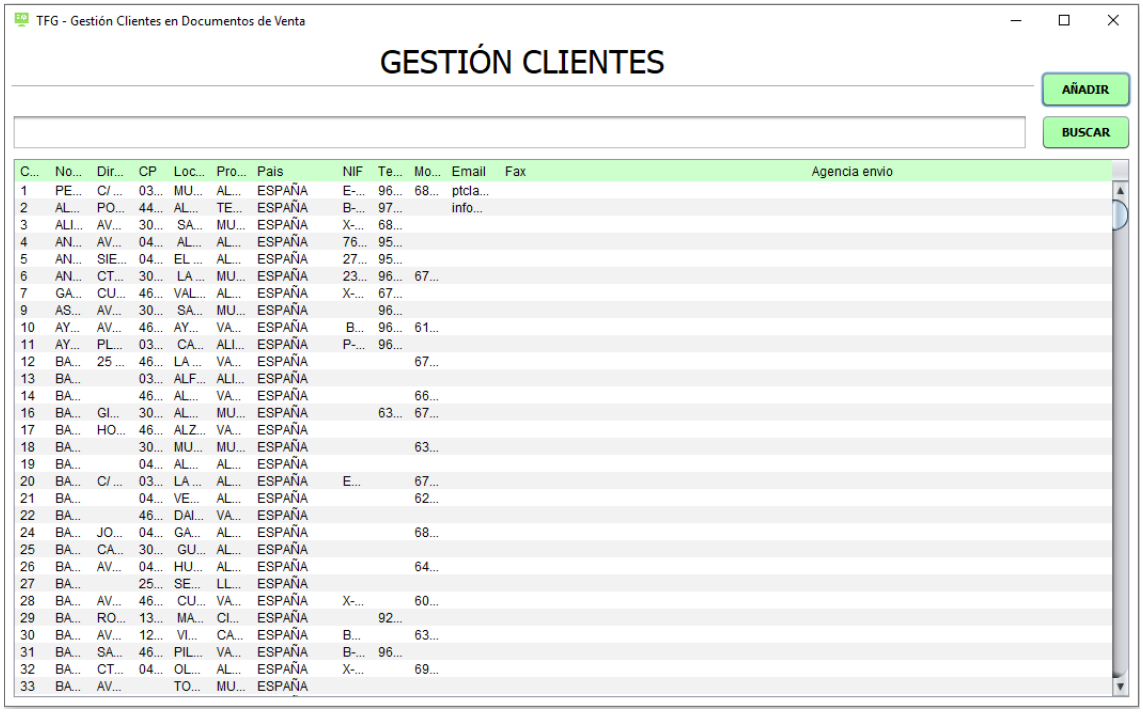


Ilustración 45. Añadir cliente a etiquetas

En el aspecto de la programación la llamada al método de la siguiente pantalla es casi idéntico al de los artículos, simplemente aquí se le pasará un "JLabel" en vez de un "JTable" y ningún booleano.

```
private void btnAñadirClienteActionPerformed(java.awt.event.ActionEvent evt) {
    gestionClientesExtra ven = new gestionClientesExtra();
    ven.cliente(txtCliente);
    ven.setVisible(true);

}
```

Ilustración 46. Botón añadir cliente

Como esta misma pantalla se utilizará también en los documentos de venta, aquí también hay un “If/else” a partir de un booleano que determinará desde que pantalla se está haciendo la llamada

```
if (pantalla == true) {
    añadir();
} else {
    añadirAux();
}
```

Ilustración 47. Llamada dependiendo de la pantalla

En este caso será el método “añadirAux()”, el que añadirá el texto al “Jlabel” pasado por parámetro anteriormente

```
private void añadirAux() {
    String nombre = tblClientes.getValueAt(tblClientes.getSelectedRow(), 1).toString();
    String localidad = tblClientes.getValueAt(tblClientes.getSelectedRow(), 4).toString();
    String provincia = tblClientes.getValueAt(tblClientes.getSelectedRow(), 5).toString();
    String email = tblClientes.getValueAt(tblClientes.getSelectedRow(), 10).toString();

    cliente.setText(nombre.toUpperCase() + " " + email.toLowerCase() + " "
            + localidad.toUpperCase() + "(" + provincia.toUpperCase() + ") ");
    this.setVisible(false);
}
```

Ilustración 48. Método AñadirAux

Como se ha comentado en el apartado anterior, había dos formas de programar el “añadi”r en la pantalla de artículos. Esta hubiese sido la otra forma de hacerlo, ya que cuando se programó este método no se pensó en las repercusiones que podría tener de cara a uso en múltiples dispositivos ya que el cliente acotó que los cambios en los clientes se hacían una vez cada mucho tiempo. Aun así, está anotado como mejora para la siguiente versión.

4.3.3 Borrar cliente y borrar artículo

4.3.3.1 Borrar cliente

Este botón no tiene nada de especial, ya que la única función que tiene es poner una cadena de caracteres vacía en el “Jlabel” del cliente

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
  txtCliente.setText("");
}
```

Ilustración 49. Limpiar cliente

4.3.3.2 Borrar artículo

El “Jbutton” lo primero que hará será preguntarle al usuario si desea realmente eliminar el producto que ha seleccionado en la “jtable”, si la respuesta es positiva, eliminará la fila de la tabla y la actualizará.

Seguidamente y por temas de que la cabecera del “Jtable” no se actualizaba el color al hacer esta operativa, se tenía que volver a renderizar.

```
if (JOptionPane.showConfirmDialog(rootPane, "¿Desea realmente eliminar este producto?\n "
        + "Una vez eliminado no habrá vuelta atrás.",
        "Salir del sistema", JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
    lineas.removeRow(tblLineas.getSelectedRow());
    lineas.fireTableStructureChanged();
    resizeColumnWidth(tblLineas);
}
for (int i = 0; i < tblLineas.getColumnCount(); i++) {
    tblLineas.getColumnModel().getColumn(i).setHeaderRenderer(new render());
}
```

Ilustración 50. Método borrar artículo

4.3.4 Botón imprimir

El método asociado a este botón es el que más trabajo ha llevado de hacer y el más largo de explicar, es por eso por lo que se ha dejado para el final del apartado.

Una vez se pulsa el botón imprimir la primera acción que realiza es recorrer todas las líneas de la tabla mediante un bucle “for”; únicamente entrará a realizar la impresión si el “checkbox” de imprimir está marcado. Al recorrer todas las líneas, imprimirá tantas veces como líneas tenga. Además, el programa permite imprimir varias etiquetas del mismo código de barras, por lo tanto, habrá otro bucle que hará la operativa tantas veces como ponga en el número de la celda.

```
    for (int i = 0; i < tblLineas.getRowCount(); i++) {
        if (tblLineas.getValueAt(i, 6).toString().equalsIgnoreCase("true")) {

            for (int s = 0; s < Double.parseDouble(tblLineas.getValueAt(i, 5).toString()); s++) {
                try {
```

Ilustración 51. Bucle for para recorrer la tabla

En una de las reuniones, surgió el problema de que algunos de los códigos de barras que tenían en la empresa, no eran correctos, ya que tenían menos valores de los que el código de barras necesitaba (ean-13). El fallo era que el código de validación se lo calculaba el programa que tenían y no lo hacía correctamente, y al intentar imprimir y no tener ese valor daba error el código de barras, así que se decidió a hacer un método que lo calculase a la hora de la impresión, si no estaba.

```
        String codigo = tblLineas.getValueAt(i, 4).toString();
        if (tblLineas.getValueAt(i, 4).toString().length() < 12) {
            int r = 12 - tblLineas.getValueAt(i, 4).toString().length();
            for (int j = 0; j < r; j++) {
                codigo = "0" + codigo;
            }
            int par = 0;
            int impar = 0;
            int todo = 0;
            for (int z = 0; z < codigo.length(); z++) {
                if (z % 2 == 0) { //si la posición actual es par
                    par = par + Integer.parseInt(codigo.charAt(z) + "");
                } else {
                    impar = impar + Integer.parseInt(codigo.charAt(z) +
                }
            }
            impar = impar * 3;
            todo = impar + par;
            int z = 0;
            if (todo % 10 != 0) {
                z = (int) (todo % 10 / 1);
            }
            todo = 10 - z;
            codigo = codigo + "" + todo;
        }
```

Ilustración 52. Recalcular código de barras

Una vez el código está calculado, lo siguiente que habrá que hacer será imprimir. Dependiendo de si ha seleccionado el cliente o no, imprimirá una etiqueta con los datos del cliente o una etiqueta con los datos de la empresa. Además, al "report" se le pasan unos parámetros vía "HashMap" (mapa de valores, tipo Clave-Valor) para así rellenarlo.

```
Map parameters = new HashMap();
parameters.put("idart", tblLineas.getValueAt(i, 0));
parameters.put("ANCHO", tblLineas.getValueAt(i, 3));
parameters.put("ALTO", tblLineas.getValueAt(i, 2));
parameters.put("ARTICULO", tblLineas.getValueAt(i, 1));
parameters.put("CODIGOBARRAS", codigo);

String documento;
if (txtCliente.getText().equalsIgnoreCase("")) {
    documento = "etiquetaPeque";
} else {
    documento = "etiquetaPequePer";
    parameters.put("CLIENTE", txtCliente.getText());
}
```

Ilustración 53. Parámetros Jasper

A continuación, se selecciona la impresora de la tabla impresoras con el nombre "ETIQUETAS", se compila el informe en tiempo real, se le asigna una impresora y se exporta.

```
String seleccion = "SELECT IMPRESORA FROM TB_impresoras where zona='ETIQUETAS'";

PreparedStatement ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

ResultSet rs = ps.executeQuery();
rs.next();
String printerName = rs.getString("IMPRESORA");

JasperReport report = JasperCompileManager.compileReport(
        file.getAbsoluteFile() + "\\src\\documentos\\" + documento + ".jrxml");
//se procesa el archivo jasper
JasperPrint myJRprintReportObject = JasperFillManager.fillReport(report, parameters, conn);
JRSaver.saveObject(myJRprintReportObject, file.getAbsoluteFile() + "\\src\\documentos\\MyFa
PrintRequestAttributeSet printRequestAttributeSet = new HashPrintRequestAttributeSet();
//printRequestAttributeSet.add(MediaSizeName.ISO_A6);

PrintServiceAttributeSet printServiceAttributeSet = new HashPrintServiceAttributeSet();
printServiceAttributeSet.add(new PrinterName(printerName, null));

JRPrintServiceExporter exporter = new JRPrintServiceExporter();

exporter.setExporterInput(new SimpleExporterInput(file.getAbsoluteFile() + "\\src\\document
SimplePrintServiceExporterConfiguration configuration = new SimplePrintServiceExporterConfi
configuration.setPrintRequestAttributeSet(printRequestAttributeSet);
configuration.setPrintServiceAttributeSet(printServiceAttributeSet);
configuration.setDisplayPageDialog(false);
configuration.setDisplayPrintDialog(false);
exporter.setConfiguration(configuration);
exporter.exportReport();
```

Ilustración 54. Compilar e imprimir Jasper

No se hace ninguna comprobación en la impresora porque se supone que existe, si no existiese daría un error y no imprimiría nada.

El resultado final sería como la imagen de a continuación dependiendo de si se ha elegido un cliente o no.

Ilustración 55. Etiqueta pequeña

4.4 Documentos de Venta

La pantalla documentos de venta es la más importante de todas, donde reside toda la facturación de la empresa. Es a la que más horas se le ha dedicado para que su funcionamiento sea óptimo de cara facilitar el trabajo de gestión y de oficina.

Como la BBDD es la del cliente final, algunos de los datos se han ocultado para proteger su privacidad.

Como la mayoría de las pantallas, esta tiene dos pestañas, “LISTA” y “FICHA”, las cuales tienen el funcionamiento de un mantenimiento general. Si se pulsa sobre un documento de venta se abrirá para su “edición” y si se pulsa sobre el botón “nuevo” se abrirá en modo “nuevo”.

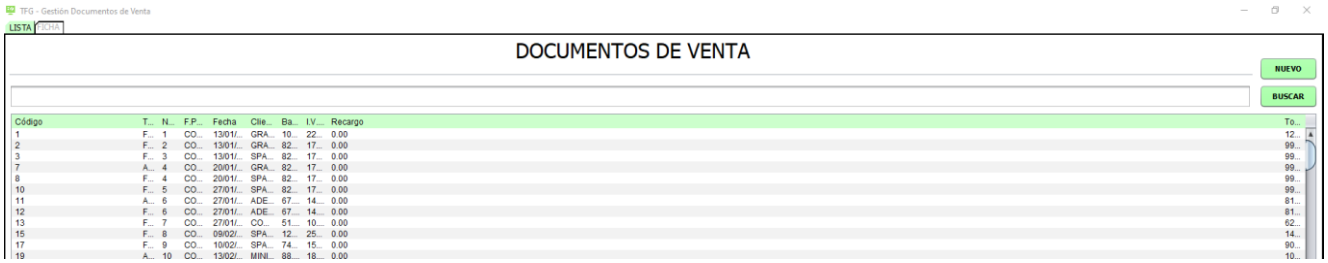


Ilustración 56. Documentos de venta - Lista

Al pulsar sobre el botón nuevo, se cambiará a la pestaña de “FICHA” y aparecerá lo siguiente:

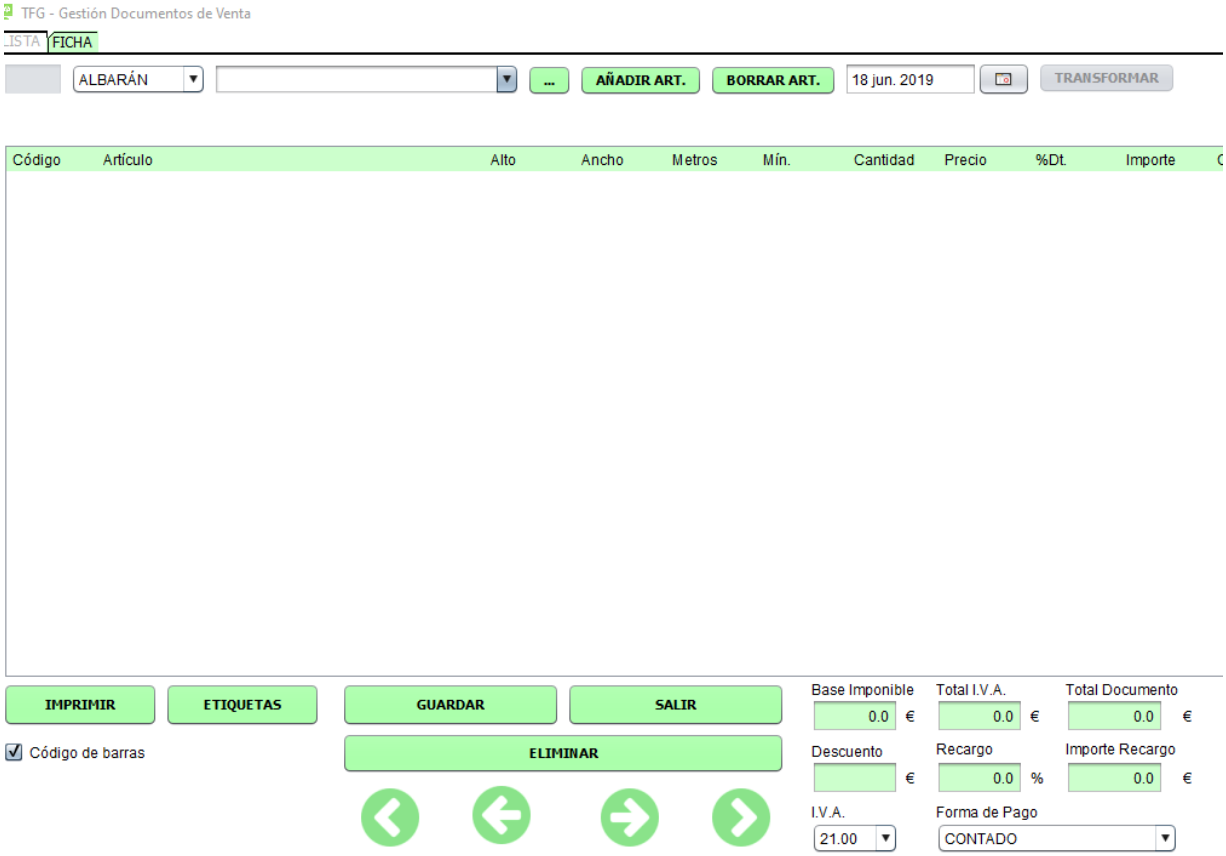


Ilustración 57. Documentos de venta - Ficha

Como se puede observar el diseño de la pantalla cuenta con tres secciones:

1. Cabecera: Contiene el tipo de documento, el cliente, los botones para añadir/borrar artículos, la fecha del documento de venta y un botón transformar para pasar un albarán a factura.
2. Cuerpo: Contiene todas las líneas del documento de venta, los artículos y sus peculiaridades.
3. Pie: Contiene los botones de acción para finalizar un Documento de venta, para eliminarlo (solo albaranes no transformados) o abonarlo, imprimirlo, cambiar la forma de pago, el I.V.A. que se aplica o los totales.

4.4.1 Cabecera

Ilustración 58. Cabecera doc. venta

Los botones de "…" y "Añadir art." tienen el mismo funcionamiento que los botones que se han visto en el apartado anterior sobre la pantalla etiquetas con código de barras.

El botón de la fecha desplegará un calendario donde se podrá seleccionar cualquier fecha siempre y cuando sea un albarán, si es una factura solo se podrá seleccionar la fecha del día de hoy, si se selecciona otra, automáticamente el programa pondrá la fecha de hoy. Para ello hay varios elementos de seguridad, que hacen que esto no pueda suceder, por ejemplo, hay un evento del tipo "focusLost" en el desplegable del tipo de documento, que al cambiar de elemento pone el texto abonar en vez de eliminar en el botón de eliminar, desactiva la fecha y desactiva el transformar.

```
if (cbTipo.getSelectedItem().toString().equalsIgnoreCase("factura")) {
    try {
        Calendar ahoraCal = Calendar.getInstance();

        SimpleDateFormat formatoDelTexto = new SimpleDateFormat("dd-MM-yyyy");
        String data = ahoraCal.get(Calendar.DATE) + "-" + (ahoraCal.get(Calendar.MONT

        cbFecha.setDate(formatoDelTexto.parse(data));
        cbFecha.setEnabled(false);

    } catch (ParseException ex) {
        Logger.getLogger(gestionFacturas.class
                .getName()).log(Level.SEVERE, null, ex);
    }
    btnEliminar.setText("ABONAR");
    abono = true;
    btnTransformar.setEnabled(false);
} else {
    cbFecha.setEnabled(true);
    btnTransformar.setEnabled(false);
    btnEliminar.setText("ELIMINAR");
    abono = false;
}
```

Ilustración 59. Focus lost - Tipo de doc. venta

4.4.1.1 Borrar artículo

El botón eliminar artículo requiere que primero haya un elemento de la tabla seleccionado y posteriormente pregunta al usuario si realmente desea eliminar este artículo de la tabla o del albarán.

Para saber si una línea esta agregada a la base de datos o no, hay que mirar la primera columna de la tabla. Ei ésta no contiene nada significa que aún no se ha añadido a la BBDD, si contiene un número significa que ya está en la BBDD y por lo tanto no solo se eliminaría de forma visual, sino que habría que eliminarla de la BBDD y recalcular los totales.

```
int row = tblLineas.getSelectedRow();
if (!tblLineas.getValueAt(row, 0).equals("")) {

    String seleccion = "select cantidad,id_articulo from tb_lin_documentos where id=" + tb
    PreparedStatement ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSIT
            ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

    ResultSet rs = ps.executeQuery();
    rs.next();
    Double cantidad = rs.getDouble("cantidad");
    String idart = rs.getString("id_articulo");

    seleccion = "select stock from tb_articulos where id=" + idart;
    ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

    rs = ps.executeQuery();
    rs.next();
    Double stockActual = rs.getDouble("stock") + cantidad;
    seleccion = "update tb_articulos set stock='" + stockActual + "' where id=" + idart;

    conn.prepareStatement(seleccion).executeUpdate();

    seleccion = "delete from tb_lin_documentos where id=" + tblLineas.getValueAt(row, 0);

    conn.prepareStatement(seleccion).executeUpdate();
}
```

Ilustración 60. Eliminar artículo

Primero se sumará la cantidad de la línea al stock del artículo y posteriormente se eliminará la línea de la base de datos.

```
}
lineas.removeRow(tblLineas.getSelectedRow());
lineas.fireTableStructureChanged();
calcularTotales();
resizeColumnWidth(tblLineas);
for (int i = 0; i < tblLineas.getColumnCount(); i++) {
    tblLineas.getColumnModel().getColumn(i).setHeaderRenderer(new render());
}
```

Ilustración 61. Eliminar artículo de la tabla

Por último, se eliminará la fila del modelo, se actualizará para que no se muestren y se recalcularán los totales.

4.4.1.2 Transformar

Este botón solo está activo si el documento está guardado y es un albarán, si se cumplen los requisitos y se clica. Llamará a un método que primero buscará la cabecera en la tabla "tb_documentos" y el número de factura máximo que hay en esta tabla, para posteriormente insertar todos los datos con la numeración calculada.

```
String seleccion = "select * from tb_documentos where id=" + txtCodigo.getText();
PreparedStatement ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

ResultSet rs = ps.executeQuery();
rs.next();
Calendar ahoraCal = Calendar.getInstance();
String data = (ahoraCal.get(Calendar.MONTH) + 1) + "-" + ahoraCal.get(Calendar.DATE) + "-" + ahoraCal.get(Calendar.YEAR);

seleccion = "select max(numfactura) from tb_documentos ";
ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

ResultSet rz = ps.executeQuery();
rz.next();
int id_fac = rz.getInt("max") + 1;

seleccion = "insert into tb_documentos (baseimponible,importe_iva,total,recargo,importe_recargo,id_fpago,id_cliente,id_iva
        + " values ('" + rs.getString("baseimponible") + "','" + rs.getString("importe_iva") + "',"
        + "'" + rs.getString("total") + "','" + rs.getString("recargo") + "','" + rs.getString("importe_recargo") + "'"
        + ",'" + rs.getString("id_fpago") + "','" + rs.getString("id_cliente") + "','" + rs.getString("id_iva") + "','" +
        + ",'" + id_fac + "','" + rs.getDouble("totaldto") + "',1,'" + rs.getString("numalbaran") + "')";

conn.prepareStatement(seleccion).executeUpdate();
```

Ilustración 62. Transformar documento de venta

Una vez insertados estos datos, recogerá el número de factura y se lo insertará al albarán anterior para saber que ha sido transformado y ya no se puede eliminar.

```
seleccion = "select max(numfactura) from tb_documentos ";
ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

rz = ps.executeQuery();
rz.next();
id_fac = rz.getInt("max");
seleccion = "update tb_documentos set numfactura=" + id_fac + " where id=" +

conn.prepareStatement(seleccion).executeUpdate();
```

Ilustración 63. Actualizar número de factura

Por último, se repetirá el mismo proceso con las líneas del documento anterior.

```
seleccion = "select max(id) from tb_documentos ";
ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

rs = ps.executeQuery();
rs.next();
int id = rs.getInt("max");
seleccion = "select * from tb_lin_documentos where id_factura=" + txtCodigo.getText();
ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

rs = ps.executeQuery();
while (rs.next()) {

    seleccion = "insert into tb_lin_documentos (id_articulo, alto, ancho, metros, minim
            + "  values ('" + rs.getString("id_articulo") + "', '" + rs.getString("alto
            + " '" + rs.getString("metros") + "', '" + rs.getString("minimo") + "','" +
            + "'" + rs.getString("precio") + "','" + rs.getString("dto") + "','" + rs.g

    conn.prepareStatement(seleccion).executeUpdate();
```

Ilustración 64. Insertar las líneas

Si todo ha ido correctamente, se mostrará un mensaje diciendo que ha sido transformado correctamente.

4.4.2 Cuerpo

La tabla contiene un evento del tipo foco ganado para calcular el importe y los totales. Cada vez que se hace clic sobre una celda, recalcula el valor de la línea.

Primero llamará a un método llamado “elegir” que será el encargado de elegir qué tipo de artículo es, siendo tres los tipos que tiene esta empresa: unidades, metro lineal y metro cuadrado. A este método se le pasará el resultado de una función llamada buscar la cual devuelve el tipo de producto que es, además del múltiplo de facturación que se utiliza para calcular los totales por línea en algunos tipos (este no se devolverá, sino que será asignado en una variable global).

```
elegir(buscar(tblLineas.getValueAt(tblLineas.getSelectedRow(), 1) + ""), tblLineas.getSelectedRow());
calcularTotales();
```

Ilustración 65. Focus gain - Tabla artículos

El método “elegir” llamará a otros métodos dependiendo del tipo que se haya pasado por parámetro.

```
private void elegir(int elegir, int linea) {

    if (elegir == 1) {
        unidades(linea);
    } else if (elegir == 2) {
        lineal(linea);
    } else if (elegir == 3) {
        cuadrado(linea);
    }
```

Ilustración 66. Método elegir

4.4.2.1 Unidades

Las unidades se calculan a través de la ecuación:

$$(cantidad\ x\ precio) - (descuento\ x\ cantidad\ x\ precio)$$

```
private void unidades(int linea) {
    Double cantidad = Double.parseDouble(tblLineas.getValueAt(linea, 6) + "");
    Double precio = Double.parseDouble(tblLineas.getValueAt(linea, 7) + "");
    Double descuento = Double.parseDouble(tblLineas.getValueAt(linea, 8) + "") / 100;
    Double importe;
    importe = (cantidad * precio) - (descuento * cantidad * precio);
    importe = Math.round(importe * 100.0) / 100.0;
    tblLineas.setValueAt(importe, linea, 9);
    tblLineas.setValueAt(1.0, linea, 4);
```

Ilustración 67. Método unidades

4.4.2.2 Lineal

Este método es un poco más complicado que el anterior. Por sus características particulares de operar, la empresa quería que el mínimo de ancho fuese 1 por lo que, si se ponía un valor por debajo, se actualizaría el valor a 1.

Si el múltiplo de facturación no es 0, significa que el ancho no puede ser un valor cualquiera, tiene que ir de 5 en 5, por lo que se refactoriza el valor recalculándolo en base a esta regla.
Por último, a la ecuación del apartado anterior se le añade el ancho en ambas partes:

$$(cantidad\ x\ precio\ x\ ancho) - (descuento\ x\ cantidad\ x\ precio\ x\ ancho)$$

```
private void lineal(int linea) {
    Double cantidad = Double.parseDouble(tblLineas.getValueAt(linea, 6) + "");
    Double precio = Double.parseDouble(tblLineas.getValueAt(linea, 7) + "");
    Double ancho = Double.parseDouble(tblLineas.getValueAt(linea, 3) + "");
    Double descuento = Double.parseDouble(tblLineas.getValueAt(linea, 8) + "") / 100;
    Double importe = null;
    if (ancho < 1.0) {
        ancho = 1.0;
    }
    if (multiplo != 0.0) {
        ancho = Double.parseDouble(refactorizar(ancho + "") + "");
    }
    ancho = ancho / 100.0;
    importe = (cantidad * precio * ancho) - (descuento * cantidad * precio * ancho);
    importe = Math.round(importe * 100.0) / 100.0;
    tblLineas.setValueAt(ancho, linea, 4);
    tblLineas.setValueAt(importe, linea, 9);
```

Ilustración 68. Método lineal

4.4.2.3 Cuadrado

En este tipo de artículo, se calcula a través de los metros cuadrados que tiene (ancho x alto). En este caso en vez de refactorizar únicamente el ancho, también se hará con la altura y posteriormente se multiplicarán entre ellas, siendo el mínimo de metros 1.

La ecuación en este caso quedará así:

$$(cantidad\ x\ precio\ x\ metros) - (descuento\ x\ cantidad\ x\ precio\ x\ metros)$$

```
private void cuadrado(int linea) {
    Double cantidad = Double.parseDouble(tblLineas.getValueAt(linea, 6) + "");
    Double precio = Double.parseDouble(tblLineas.getValueAt(linea, 7) + "");
    Double altura = Double.parseDouble(tblLineas.getValueAt(linea, 2) + "");
    Double ancho = Double.parseDouble(tblLineas.getValueAt(linea, 3) + "");
    Double descuento = Double.parseDouble(tblLineas.getValueAt(linea, 8) + "") / 100;
    Double importe = null;
    Double metros = null;
    if (multiplo != 0.0) {
        altura = Double.parseDouble(refactorizar(altura + "") + "");
        ancho = Double.parseDouble(refactorizar(ancho + "") + "");
    }
    ancho = ancho / 100.0;
    altura = altura / 100.0;
    metros = ancho * altura;
    metros = Math.round(metros * 100.0) / 100.0;
    if (metros < 1.0) {
        metros = 1.0;
    }
    importe = (cantidad * precio * metros) - (descuento * cantidad * precio * metros);
    importe = Math.round(importe * 100.0) / 100.0;
    tblLineas.setValueAt(importe, linea, 9);
    tblLineas.setValueAt(metros, linea, 4);
```

Ilustración 69. Método cuadrado

4.4.2.4 Refactorizar

```
private int refactorizar(String ref) {
    StringTokenizer st = new StringTokenizer(ref + "", ".");

    int todo = Integer.parseInt(st.nextToken());
    int s = 0;
    if (todo % 10 != 0) {
        s = (int) (todo % 10 / 1);
    }
    if (s != 0 && s != 5) {
        if (s < 5) {
            s = 5 - s;
            todo = todo + s;
        } else {
            s = 10 - s;
            todo = todo + s;
        }
    }
    return todo;
}
```

Ilustración 70. Método refactorizar

4.4.2.5 Calcular totales

Una vez realizado el método “elegir” y sus métodos internos, se ejecutará el método “calcularTotales” el cual actualizará todos los valores de la sección “pie”.

Primero se recogerá el valor de IVA seleccionado en el combo del “pie”.

```
try {
    String seleccion = "select \n"
            + "    valor\n"
            + "from tb_iva\n"
            + "where id=" + (cbIVA.getSelectedIndex() + 1);

    PreparedStatement ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

    ResultSet rs = ps.executeQuery();
    rs.next();
```

Ilustración 71. Seleccionar I.V.A.

Posteriormente, se calcularán los valores de descuento, total y el tanto por ciento del descuento sobre el total del documento de venta.

```
Double recargo = Double.parseDouble(txtRecargo.getText()) / 100;
Double iva = rs.getDouble("valor") / 100.0;
Double sumaTotal = 0.0;
Double descuento = 0.0;
Double tantoporcientodescuento = 0.0;
for (int i = 0; i < tblLineas.getRowCount(); i++) {
    tantoporcientodescuento = 1 - (Double.parseDouble(tblLineas.getValueAt(i, 8).toStrin
    sumaTotal = sumaTotal + Double.parseDouble(tblLineas.getValueAt(i, 9).toString());
    descuento = descuento + ((Double.parseDouble(tblLineas.getValueAt(i, 9).toString())
}
```

Ilustración 72. Calcular totales

Y, por último, se asignarán todos los valores a sus respectivos textos.

```
sumaTotal = Math.round(sumaTotal * 100.0) / 100.0;
descuento = Math.round(descuento * 100.0) / 100.0;
iva = Math.round(sumaTotal * iva * 100.0) / 100.0;
recargo = Math.round(sumaTotal * recargo * 100.0) / 100.0;
txtBase.setText(sumaTotal + "");
txtDescuento.setText("-" + descuento);
txtIva.setText(iva + "");
Double suma = Math.round((iva + sumaTotal + recargo) * 100.0) / 100.0;
txtTotal.setText(suma + "");
txtImporteRecargo.setText(recargo + "");
tblLineas.clearSelection();
```

Ilustración 73. Asignar valores

4.4.3 Pie

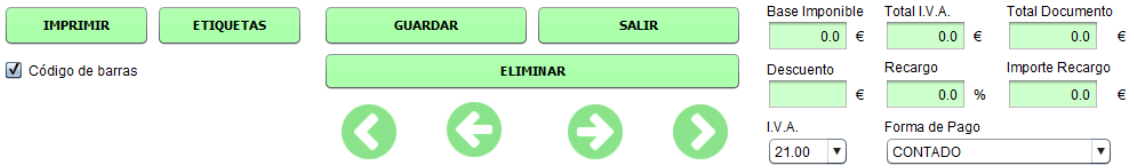


Ilustración 74. Pie - Documento de ventas

Esta sección contiene el panel de botones encargado de gestionar todo el documento de venta, además de los campos que almacenan el total del documento.

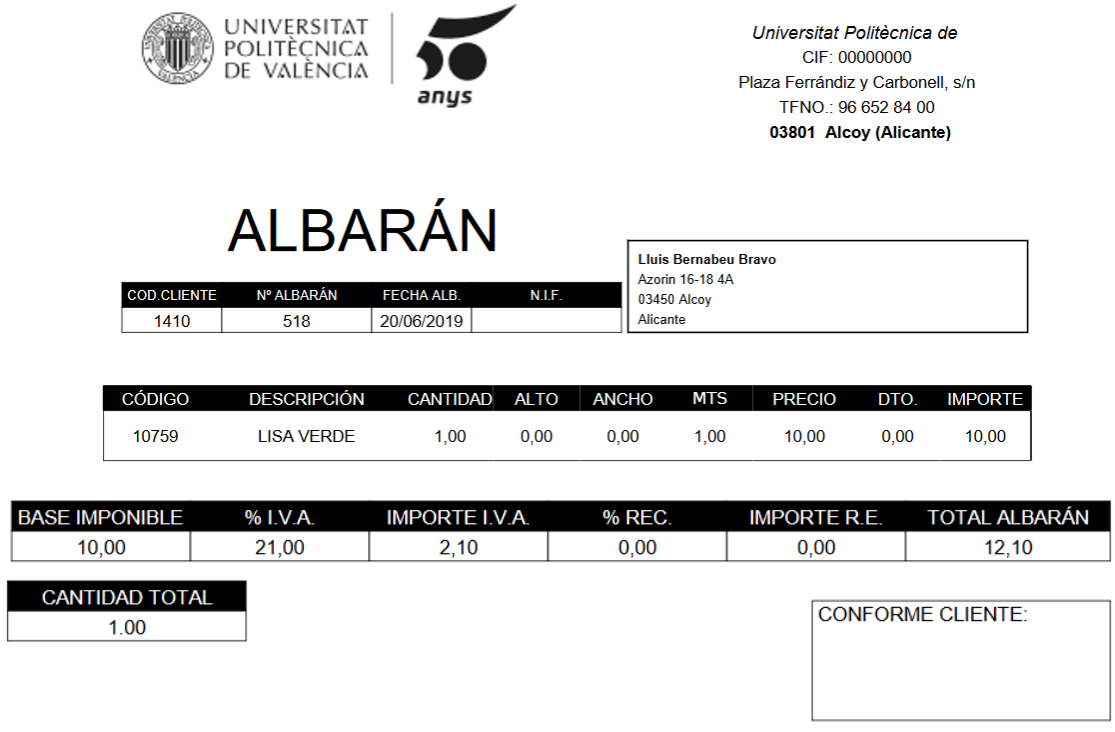
Los botones de navegación entre documentos de venta no van a ser explicados puesto que tienen el comportamiento de cualquier mantenimiento simple explicado ya de forma extensa y detallada en apartados anteriores.

4.4.3.1 Imprimir

El botón imprimir tiene una singularidad y es que el documento de venta tiene que estar guardado, ya que si no está guardado el programa lo detectará y mostrará un error como que no se puede imprimir ya que no ha sido guardado.

Si ya estaba guardado el documento, lo vuelve a guardar por si se ha hecho algún cambio antes de pulsar el botón imprimir, y posteriormente se lanza el método imprimir que es igual (excepto el documento que se compila) que el explicado en capítulos anteriores.

Una vez impresa, el documento de venta quedaría así, dependiendo del tipo que sea se mostrarán unos datos u otros.


UNIVERSITAT POLITÈCNICA DE VALÈNCIA

50 anys

Universitat Politècnica de
CIF: 00000000
Plaza Ferrándiz y Carbonell, s/n
TFNO.: 96 652 84 00
03801 Alcoy (Alicante)

ALBARÁN

COD.CLIENTE	Nº ALBARÁN	FECHA ALB.	N.I.F.
1410	518	20/06/2019	

Lluis Bernabeu Bravo
Azorin 16-18 4A
03450 Alcoy
Alicante

CÓDIGO	DESCRIPCIÓN	CANTIDAD	ALTO	ANCHO	MTS	PRECIO	DTO.	IMPORTE
10759	LISA VERDE	1,00	0,00	0,00	1,00	10,00	0,00	10,00

BASE IMPONIBLE	% I.V.A.	IMPORTE I.V.A.	% REC.	IMPORTE R.E.	TOTAL ALBARÁN
10,00	21,00	2,10	0,00	0,00	12,10

CANTIDAD TOTAL
1.00

CONFORME CLIENTE:

Ilustración 75. Documento de venta impreso

4.4.3.2 Etiquetas

El botón etiquetas abrirá una pantalla donde estarán todos los elementos del documento de venta, esta pantalla es similar a la vista en etiquetas en apartados anteriores, la diferencia es que esta no tiene botones para añadir, simplemente imprimir.

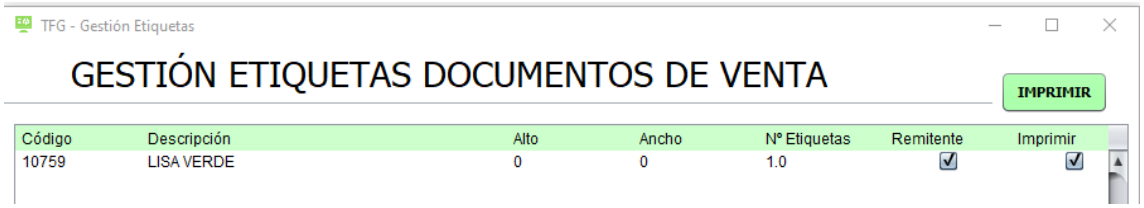


Ilustración 76. Gestión etiquetas doc. ventas

Al imprimir, dependiendo de si está el "checkbox" de "remitente" marcado o no, mostrará una etiqueta u otra, siendo la única diferencia entre ambas, que en el marcado mostrará los datos de la empresa y en la otra no.

Ilustración 77. Etiqueta con remitente

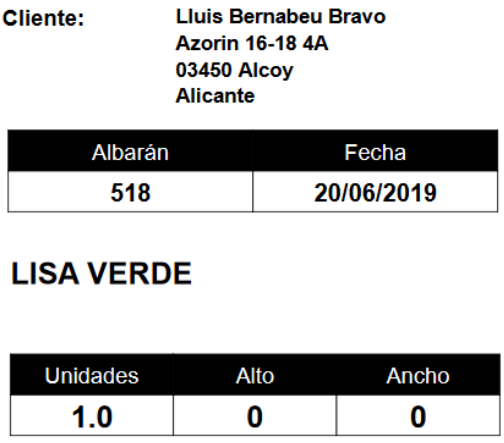


Ilustración 78. Etiqueta sin remitente

4.4.3.3 Guardar

Este es otro de los botones principales de esta pantalla y de todo el programa, ya que es el encargado de insertar o modificar los datos de los documentos de venta.

Si el documento de venta no tiene un código, significa que aún no se ha insertado en la base de datos y por lo tanto al método que hay que llamar es al de insertar.

Este método, primero buscará la forma de pago y el cliente en la base de datos a partir de los datos seleccionados en pantalla.

```
try {
    String seleccion = "";
    int id = 0;
    seleccion = "select id from tb_fpago where descripcion='" + cbFormaPago.getSelectedItem().toString() + "'";
    PreparedStatement ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

    ResultSet rs = ps.executeQuery();
    rs.next();
    id = rs.getInt("id");
    int id_cliente = 0;
    if (!cbClientes.getSelectedItem().toString().equalsIgnoreCase(" ")) {
        seleccion = "select id from tb_clientes where nombre='" + cbClientes.getSelectedItem().toString() + "'";
        ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);
```

Ilustración 79. Buscar forma de pago

Posteriormente, insertará la cabecera del documento de venta y se guardará el código asociado a esta, dependiendo de si es un albarán o una factura.

```
if (cbTipo.getSelectedIndex() == 1) {
    seleccion = "select max(numfactura) from tb_documentos ";
    ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

    rs = ps.executeQuery();
    rs.next();
    int id_fac = rs.getInt("max") + 1;
    idr = id_fac;
    seleccion = "insert into tb_documentos (baseimponible,importe_iva,total,re
            + " values ('" + txtBase.getText() + "','" + txtIva.getText() + "'
            + ",'" + id + "','" + id_cliente + "','" + (cbIVA.getSelectedIndex
else if (cbTipo.getSelectedIndex() == 0) {

  seleccion = "select max(numalbaran) from tb_documentos ";
  ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
          ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

  rs = ps.executeQuery();
  rs.next();
  int id_alb = rs.getInt("max") + 1;
  idr = id_alb;
  seleccion = "insert into tb_documentos (baseimponible,importe_iva,total,re
          + " values ('" + txtBase.getText() + "','" + txtIva.getText() + "'
          + ",'" + id + "','" + id_cliente + "','" + (cbIVA.getSelectedIndex

  conn.prepareStatement(seleccion).executeUpdate();

  seleccion = "select  max(id) from tb_documentos";
  ps = conn.prepareStatement(seleccion, ResultSet.TYPE_SCROLL_INSENSITIVE,
          ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);
```

Ilustración 80. Insertar cabecera doc. venta

Seguidamente, actualizará el stock de los productos añadidos siempre y cuando el producto tenga marcado que es de stock.

Por último, insertará todas las líneas del documento de venta en su tabla correspondiente con el código previamente calculado.

```
// System.out.println(tblLineas.getValueAt(i, 2) + "'
seleccion = "insert into tb_lin_documentos (id_articu
        + "  values ('" + id_art + "', '" + tblLineas
        + " '" + tblLineas.getValueAt(i, 4) + "', '"
        + "'" + tblLineas.getValueAt(i, 7) + "','" +

conn.prepareStatement(seleccion).executeUpdate();
//txtNumeroFac.setText(id+"");
seleccion = "select max(id) from tb_lin_documentos";
PreparedStatement pss = conn.prepareStatement(selecci
        ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CU

ResultSet rz = pss.executeQuery();
rz.next();
tblLineas.setValueAt(rz.getString("max"), i, 0);
```

Ilustración 81. Insertar líneas doc. venta

Una vez insertado el documento de venta en la base de datos y puesta una numeración, la siguiente vez que se pulse el botón guardar, llamará al método guardar que en vez de insertar actualizará el registro de la base de datos. Este método al ser similar al explicado anteriormente ya no se va a explicar. Además, hay algunos pasos intermedios que se han obviado en la explicación para hacerla más simple.

4.4.3.4 Eliminar

Como se explicó anteriormente, el botón eliminar tiene dos funcionamientos dependiendo de si es un albarán o una factura. Si es un albarán simplemente eliminará el albarán y sus líneas (siempre y cuando no esté transformado a factura). Por otro lado, si es una factura, por temas legales no se puede eliminar, y lo que se hace es hacer un abono que se queda registrado en la base de datos con los valores en negativo.

Para abonar una factura, se insertará en la tabla de los abonos su cabecera y sus líneas.

```
seleccion = "insert into tb_abonos (baseimponible,importe_iva,total,rec
        + " values ('-" + rs.getString("baseimponible") + "','-" + rs.g
        + "'-" + rs.getString("total") + "','-" + rs.getString("recargo
        + ",'" + rs.getString("id_fpago") + "','" + rs.getString("id_cl
        + ",'" + rs.getString("numfactura") + "','" + (rs.getDouble("to
conn.prepareStatement(seleccion).executeUpdate();
```

Ilustración 82. Insertar en abonos

Se actualizará el stock a como estaba antes de la factura.

```
rs = ps.executeQuery();
rs.next();
Double stockActual = rs.getDouble("stock") + cantidad;
seleccion = "update tb_articulos set stock='" + stockActual + "' where id=" + idart;

conn.prepareStatement(seleccion).executeUpdate();
```

Ilustración 83. Actualizar stock

Y se marcará como que el documento ha sido abonado correctamente.

```
seleccion = "update tb_documentos set id_abono=" + id +

conn.prepareStatement(seleccion).executeUpdate();
```

Ilustración 84. Marcar como abonado

Puesto que la pantalla de abonos es un simple visor, prácticamente igual que la pantalla de ventas, pero sin su funcionalidad, no se va a explicar.

4.5 Informes

EL cliente hizo mucho hincapié en que la pantalla de informes debía tener mucho potencial, con un listado de informes de cada elemento que tuviese el software, por lo que se le dedicaron una gran cantidad de horas y de reuniones para dejarlo a su gusto.

Como se observa, la pantalla de los informes cuenta con dos fechas y un listado de informes que se generan en pantalla y posteriormente se pueden imprimir.

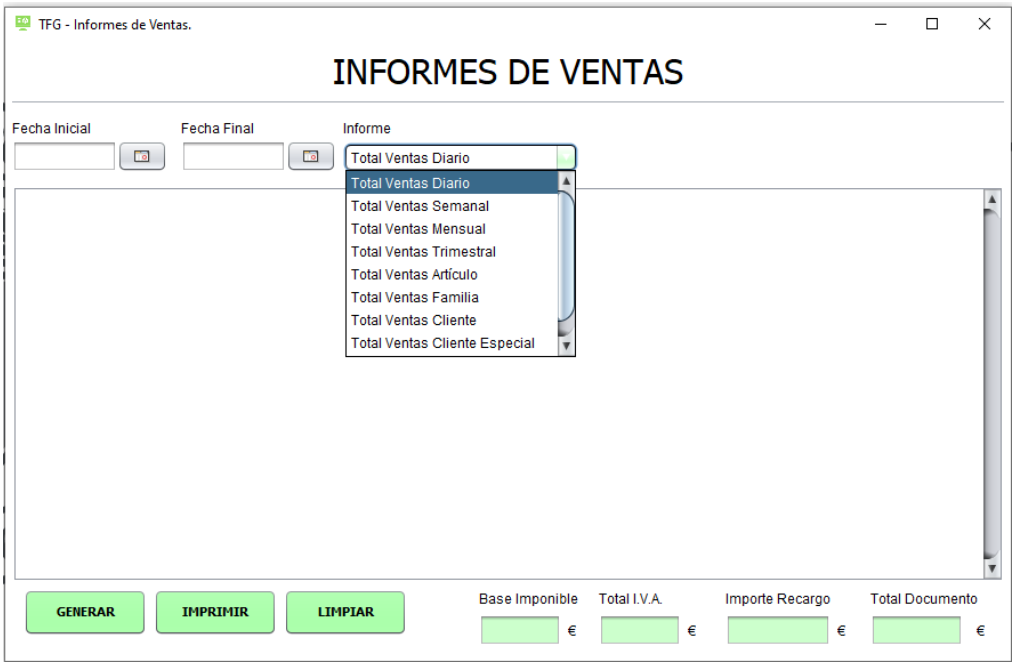


Ilustración 85. Pantalla informe de ventas

En este caso, el botón generar tiene asociado un condicional tipo “switch” que, dependiendo del valor seleccionado en el desplegable, lanzará un método u otro.

No se van a explicar los métodos utilizados para cada uno de los informes puesto que simplemente son sentencias SQL de tipo “Select” con sus cláusulas dependiendo del informe que sea. Un ejemplo sería la siguiente sentencia SQL para el informe de ventas por artículo.

```
String seleccion = "select \n"
        + "    tb_articulos.descripcion,\n"
        + "    sum( cantidad) cantidad,\n"
        + "    tb_lin_documentos.precio,\n"
        + "    sum( dto) dto,\n"
        + "    sum( importe) importe,\n"
        + "    sum( metros) metros\n"
        + "from tb_articulos\n"
        + "   inner join tb_lin_documentos on (tb_articulos.id = tb_lin_documentos.id_articulo)\n"
        + "   inner join tb_documentos on (tb_lin_documentos.id_factura = tb_documentos.id)\n"
        + "where \n"
        + "   (\n"
        + "      (tb_documentos.codigo_doc = 1)\n"
        + "      and \n"
        + "      (fecha between '" + datainicial + "' and '" + datafinal + "')\n"
        + "and (id_abono is null )"
        + "   )\n"
        + "group by tb_articulos.descripcion, tb_lin_documentos.precio\n"
        + "order by tb_articulos.descripcion";
```

Ilustración 86. Sentencia SQL para informe

Una vez impreso el informe de ventas aparecería así, teniendo en cuenta que dependiendo del informe seleccionado aparecerá agrupado por unos valores o por otros.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

50 anys

Universitat Politècnica de
CIF: 00000000
Plaza Ferrándiz y Carbonell, s/n
TFNO.: 96 652 84 00
03801 Alcoy (Alicante)

INFORME VENTAS DIARIO

De 18/06/2019 a -

FECHA	BASE IMPONIBLE	I.V.A.	RECARGO	TOTAL
20/06/2019	16.00	3,36	0,00	19,36

BASE IMPONIBLE	IMPORTE I.V.A.	IMPORTE R.E.	TOTAL FACTURAS
16.00	3.36	0.00	19.36

Ilustración 87. Informe impreso

4.6 Importador/Exportador

Al tener otro programa con todos los artículos y clientes ya guardados, a la empresa le surgió la necesidad de tener un importador de estos dos elementos. Se realizó una pantalla para poder importar estos datos la primera vez a la BBDD y así no tener que introducirlos de forma manual uno a uno.

La pantalla, dependiendo del botón que se pulse, irá a buscar un archivo “CSV” a la carpeta de instalación u otro, cogerá los datos almacenados en él y los importará a la base de datos. Es importante hacer esto siempre con la base de datos vacía, sino puede ocasionar errores y problemas.

Ilustración 88. Importador

Cabe destacar que, además, tuvieron muchos problemas para exportar los datos en el otro programa, con lo que se decidió que era buena idea tener un “exportador” de todos los datos a formato Excel, por si en un futuro les hacía falta en alguna migración

Ilustración 89. Exportador

Capítulo 5: Informes

En este capítulo se van a exponer algunos de los “Reports” que se diseñaron con y para el cliente. Para la realización de los informes se utilizó la herramienta Jaspersoft, la cual genera archivos de tipo “jrxml” que pueden ser compilados con el mismo programa o posteriormente desde Java. Además, se puede conectar con la BBDD, se le pueden pasar parámetros vía “HashMap” y también se le pueden programar funciones en Java dentro de los “Reports”.

Cada elemento del programa que contiene un botón imprimir va asociado (como se ha visto en capítulos anteriores) a un archivo “jrxml” que posteriormente se compila transformándolo a “Jasper” y se imprime.

En total hay 38 archivos tipo “jrxml”, aunque puede haber varios “jxrml” por elemento. Por ejemplo, en las etiquetas se vio que podía ser con o sin remitente; para separar esta característica se han generado dos archivos tipo “jrxml” cada uno con un nombre y mediante programación se elige uno u otro. Esta funcionalidad tiene una ventaja y un inconveniente:

- La ventaja es que se puede diferenciar a simple vista que “Report” es cada cual.
- El inconveniente es que en caso de hacer algún cambio se debería hacer en ambos, duplicándose el trabajo de mantenimiento, aunque esto rara vez pasa.

5.1 Documentos de venta

En la ilustración número 90 se puede observar el diseño de un documento de venta; este diseño es el mismo (con algunos cambios) en los albaranes y en los abonos.
En él se puede observar las variables “$F{}” que se cargan directamente de la base de datos mediante una conexión y una sentencia SQL. En Jaspersoft existen también otras llamadas “$P{}” que son pasadas desde código mediante un “HashMap”.

Por otro lado, la imagen que aparece en todos los informes (salvo etiquetas) es dinámica y está en la carpeta de la instalación, pudiéndose cambiar de manera sencilla con un simple “copiar-pegar” y sustituyendo la anterior. Se carga con el siguiente código en Jasper:

new File("").getAbsolutePath()+"/logo.jpg"

Cada una de las tablas tiene asociada una sentencia SQL, además de la principal para los datos de la cabecera.

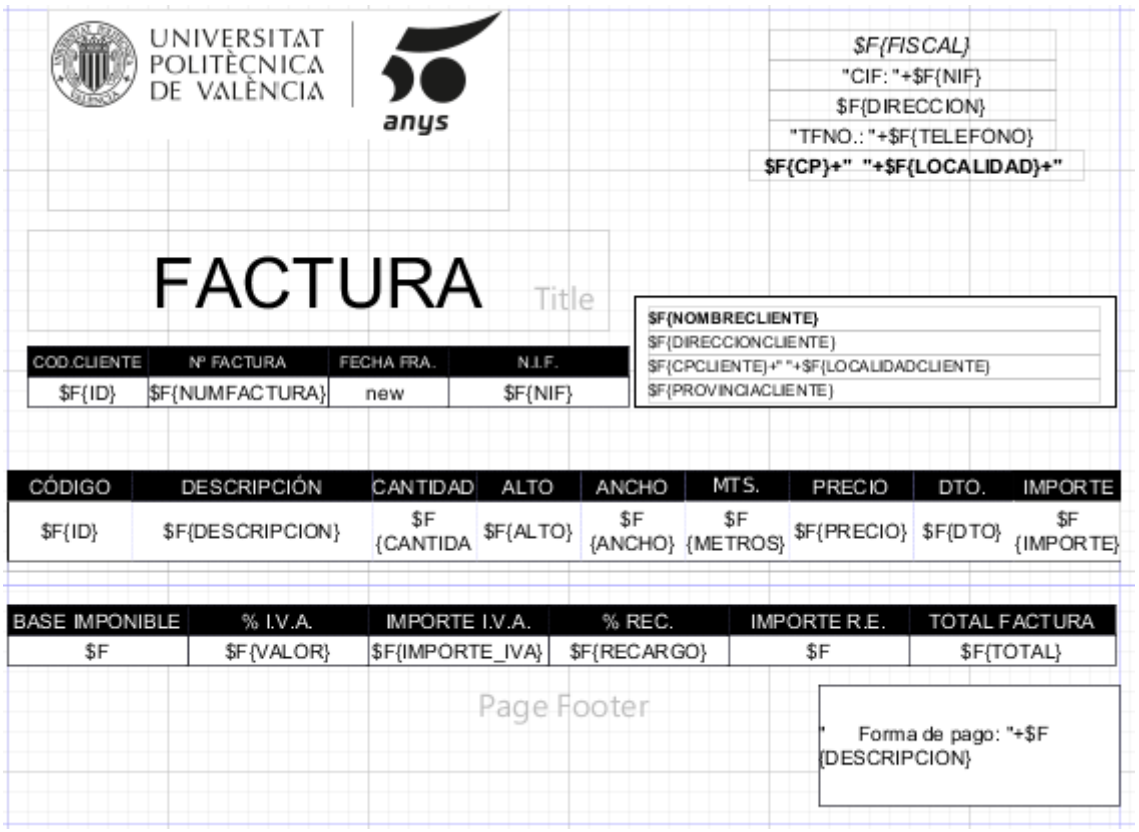


Ilustración 90. Factura – Jasper

De esta sentencia SQL se extraen los datos de la cabecera del documento, así como los datos del cliente y de la empresa.

```
SELECT "FISCAL",
        "TB_EMPRESA"."DIRECCION",
                "TB_EMPRESA"."NIF",
                "TB_EMPRESA"."LOCALIDAD",
                "TB_EMPRESA"."PROVINCIA",
                "TB_EMPRESA"."CP",
                "TB_EMPRESA"."TELEFONO",
        "PRECIO1",
        "PRECIO2",
        "ANCHO",
        "ALTO",
        "DESCRIPCION",
        "CODIGOBARRAS",
        "TB_CLIENTES"."NOMBRE" AS "NOMBRECLIENTE",
        "TB_CLIENTES"."DIRECCION" AS "DIRECCIONCLIENTE",
        "TB_CLIENTES"."CP" AS "CPCLIENTE",
        "TB_CLIENTES"."PROVINCIA" AS "PROVINCIACLIENTE",
        "TB_CLIENTES"."LOCALIDAD" AS "LOCALIDADCLIENTE"
FROM "TB_CLIENTES","TB_ARTICULOS","TB_EMPRESA"
WHERE "TB_CLIENTES"."ID"=(SELECT ID_CLIENTE FROM TB_DOCUMENTOS WHERE ID=$P{numfactura})
```

Ilustración 91. Sentencia SQL para cargar cabecera

De esta sentencia SQL se extraen los datos de la tabla de líneas de documento; esta sentencia devolverá tantos registros como líneas tenga el documento; esto hará que el Jaspersoft imprima tantas veces como elementos haya en el resultado.

CÓDIGO	DESCRIPCIÓN	CANTIDAD	ALTO	ANCHO	MTS.	PRECIO	DTO.	IMPORTE
$F{ID}	$F{DESCRIPCION}	$F{CANTIDA	$F{ALTO}	$F{ANCHO}	$F{METROS}	$F{PRECIO}	$F{DTO}	$F{IMPORTE}

```
SELECT "TB_LIN_DOCUMENTOS"."ID",
    "TB_ARTICULOS"."DESCRIPCION",
    "TB_LIN_DOCUMENTOS"."ALTO",
    "TB_LIN_DOCUMENTOS"."ANCHO",
    "TB_LIN_DOCUMENTOS"."METROS",
    "TB_LIN_DOCUMENTOS"."PRECIO",
    "TB_LIN_DOCUMENTOS"."CANTIDAD",
    "TB_LIN_DOCUMENTOS"."DTO",
    "TB_LIN_DOCUMENTOS"."IMPORTE"
FROM "TB_LIN_DOCUMENTOS"
    INNER JOIN "TB_ARTICULOS" ON
     "TB_LIN_DOCUMENTOS"."ID_ARTICULO" = "TB_ARTICULOS"."ID"
     where id_factura= $P{numfactura}
```

Ilustración 92. Sentencia SQL para cargar las líneas

Por último, de esta sentencia SQL se extraen los totales del documento de venta.

BASE IMPONIBLE	% I.V.A.	IMPORTE I.V.A.	% REC.	IMPORTE R.E.	TOTAL FACTURA
$F	$F{VALOR}	$F{IMPORTE_IVA}	$F{RECARGO}	$F	$F{TOTAL}

```
SELECT "TB_IVA"."VALOR",
  "TB_DOCUMENTOS"."BASEIMPONIBLE",
  "TB_DOCUMENTOS"."IMPORTE_IVA",
  "TB_DOCUMENTOS"."IMPORTE_RECARGO",
  "TB_DOCUMENTOS"."RECARGO",
  "TB_DOCUMENTOS"."TOTAL"
FROM "TB_IVA"
  INNER JOIN "TB_DOCUMENTOS" ON
   "TB_DOCUMENTOS"."ID_IVA" = "TB_IVA"."ID"
   where  "TB_DOCUMENTOS"."ID"=$P{numfactura}
```

Ilustración 93. Sentencia SQL para cargar totales

Existen varios tipos de ficheros “JRXML” dependiendo de si el documento de venta no tiene cliente, si tiene código de barras, etc. Además, si es de tipo albarán y ha sido transformado a factura (o si es factura transformada de un albarán) aparecerá el número desde el que ha sido transformado.
Después de todos estos filtros, nos encontramos con que hay 18 ficheros para los documentos de venta. Esta situación facilita el trabajo de diseño, pero dificulta el trabajo de mantenimiento. Hasta el momento no se ha modificado ninguno.

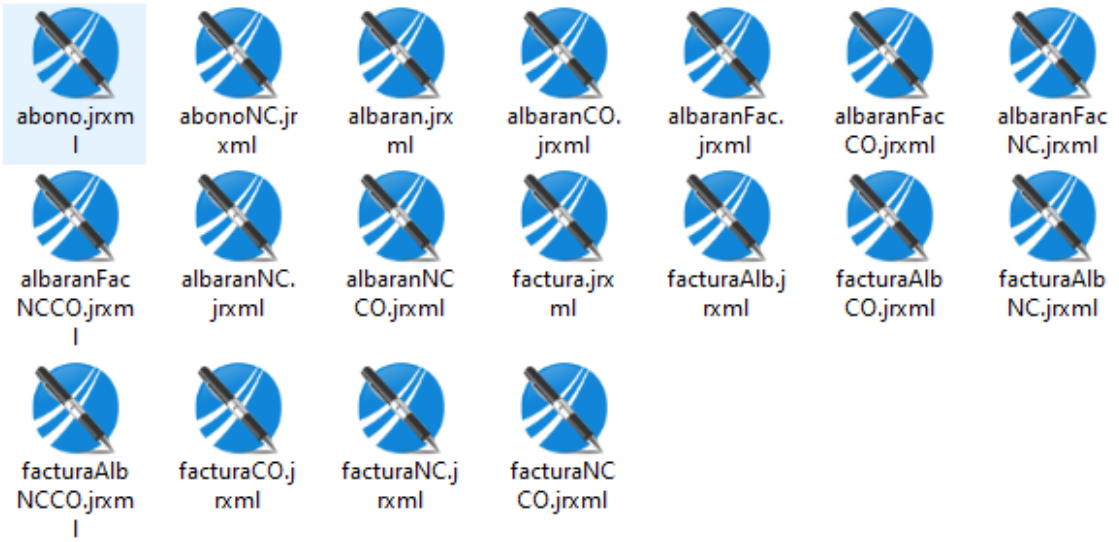


Ilustración 94. Ficheros JRXML - Doc. Ventas

5.2 Etiquetas

En las etiquetas tenemos el mismo diseño a nivel de sentencias SQL que en el apartado anterior; una cabecera que se carga en la sentencia principal y una sentencia por cada tabla. Aunque en esta hay una característica especial y es que la tabla de unidades, alto y ancho se carga desde variables pasadas por código.

Ilustración 95. Etiqueta grande

En el diseño de la etiqueta pequeña hay una diferencia clara con la etiqueta grande, y es que todos los datos son pasados por parámetro. Esto es así porque existe un problema en Jaspersoft que no calcula bien el código de barras si es pasado por sentencia SQL y da problemas, así que se decidió solucionarlo pasando todos los parámetros por código.

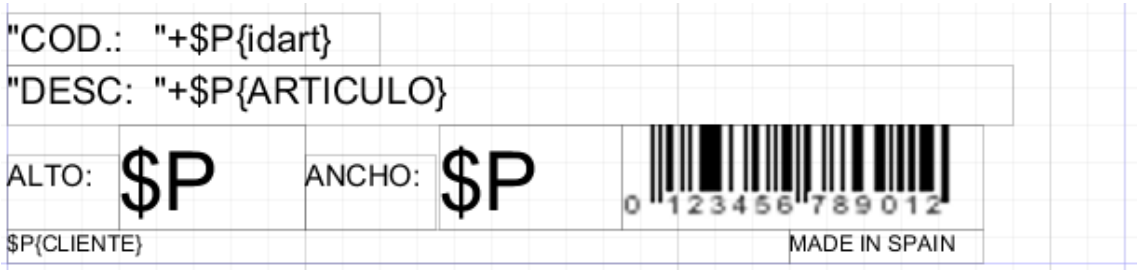


Ilustración 96. Etiqueta pequeña

Además, tal y como pasaba con los documentos de venta, aquí también existen varios diseños dependiendo de unas características u otras.

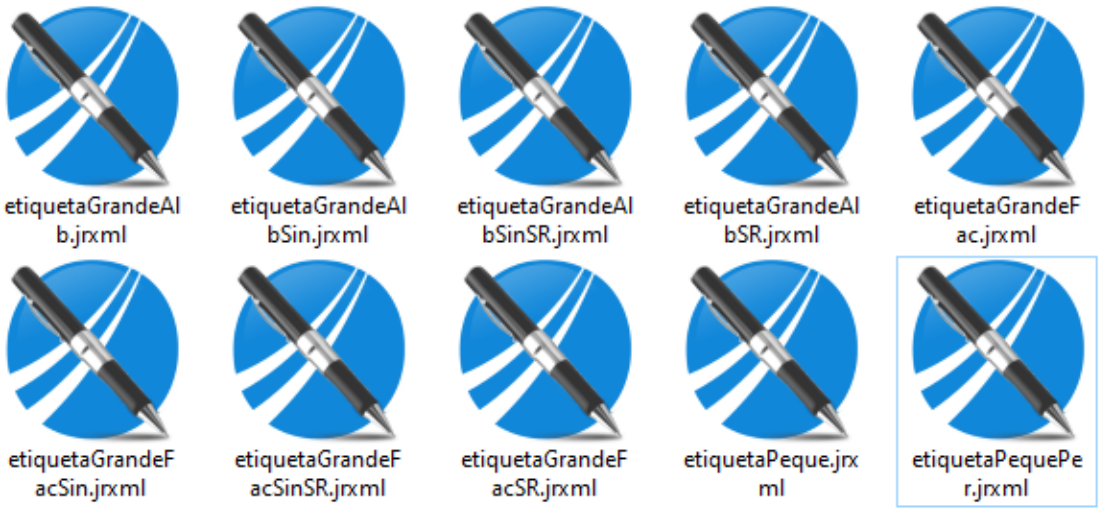


Ilustración 97. Ficheros JRXML - Etiquetas

5.3 Informes

Para los informes, existe un diseño para cada uno de los informes existentes explicados en los capítulos anteriores. Por simplificar solo se va a explicar uno de ellos, puesto que los demás son iguales exceptuando la sentencia SQL que carga las líneas de la tabla.

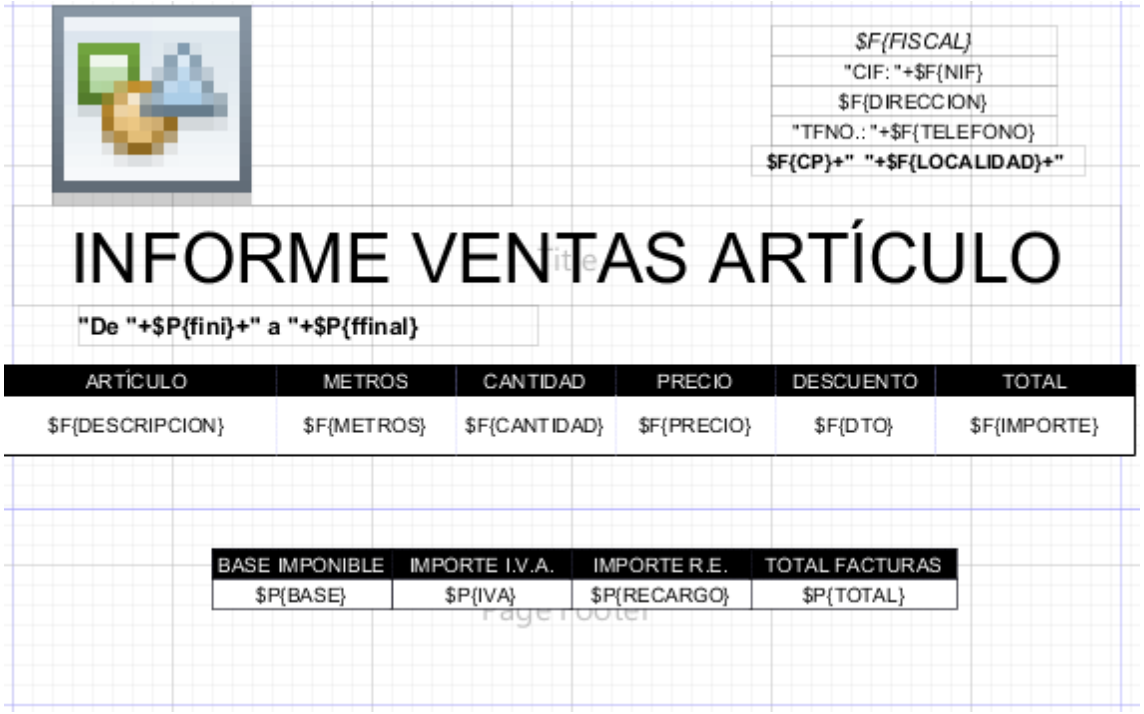


Ilustración 98. Informe

En el caso de este informe, la sentencia SQL que carga los datos de las líneas sería la de la ilustración número 98, pasándole como parámetros las fechas inicial y final. En esta sentencia SQL está la diferencia entre un informe y otro.

ARTÍCULO	METROS	CANTIDAD	PRECIO	DESCUENTO	TOTAL
$F{DESCRIPCION}	$F{METROS}	$F{CANTIDAD}	$F{PRECIO}	$F{DTO}	$F{IMPORTE}

```
select tb_articulos.descripcion,
                sum( cantidad) cantidad,
              tb_lin_documentos.precio,
               sum( dto) dto,
              sum( importe) importe,
               sum( tb_lin_documentos .metros) metros
             from tb_articulos
              inner join tb_lin_documentos on (tb_articulos.id = tb_lin_documentos.id_articulo)
             inner join tb_documentos on (tb_lin_documentos.id_factura = tb_documentos.id)
              where
                (
                 (tb_documentos.codigo_doc = 1)
                and
                  (fecha between $P!{inicial} and  $P!{final} )
                  and   (remitente=0)
               and (id_abono is null )
                 )
               group by tb_articulos.descripcion, tb_lin_documentos.precio
               order by tb_articulos.descripcion
```

Ilustración 99. Sentencia SQL - informe

Para los informes existen nueve tipos de “Report”, uno por cada uno de ellos.

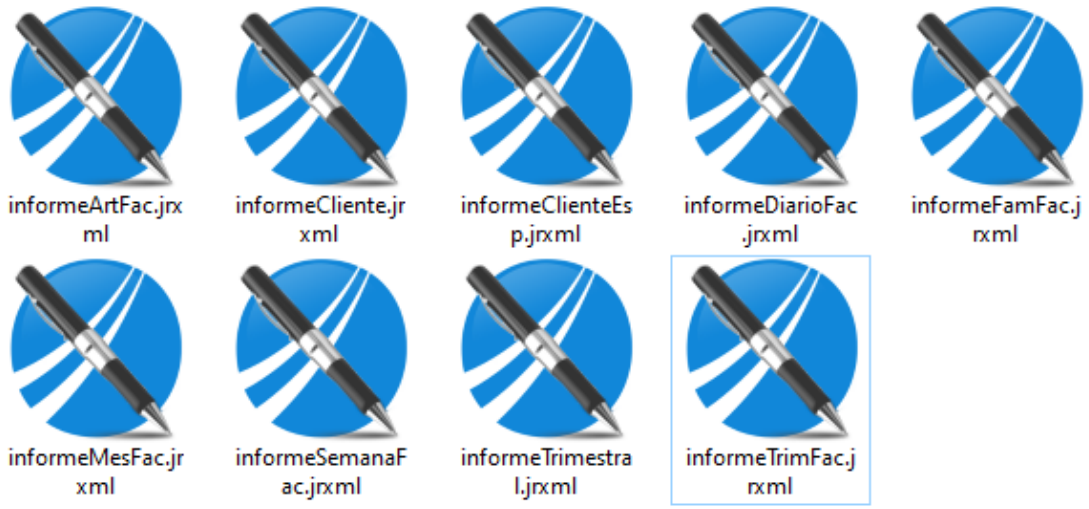


Ilustración 100. Ficheros JRXML – Informes

Capítulo 6: Anexo

Librerías Utilizadas:

batik-anim-1.8.jar
batik-awt-util-1.8.jar
batik-bridge-1.8.jar
batik-css-1.8.jar
batik-dom-1.8.jar
batik-ext-1.8.jar
batik-gvt-1.8.jar
batik-parser-1.8.jar
batik-script-1.8.jar
batik-svg-dom-1.8.jar
batik-util-1.8.jar
batik-xml-1.8.jar
castor-core-1.3.3.jar
castor-xml-1.3.3.jar
commons-beanutils-1.7.jar
commons-collections-3.2.2.jar
commons-digester-1.7.jar
commons-io-2.5.jar
commons-logging-1.0.2.jar
DateChooser.jar
eclipselink.jar
itext-2.1.7.jar
itextpdf-5.5.9.jar
jasperreports-6.2.2-20160505.150232-15.jar
jasperreports-6.3.0.jar
jasperreports-chart-themes-6.2.2-20160505.150...
jasperreports-fonts-6.1.1.jar
jasperreports-functions-6.2.2-20160505.150541...
javax.persistence_2.1.0.v201304241213.jar
jaybird-full-2.2.10.jar
jcalendar-1.4.jar
jdt-compiler-3.1.1.jar
joda-time-2.0.jar
org.eclipse.persistence.jpa.jpql_2.5.2.v20140319...
org.krysalis.barcode4j_2.1.0.jar
org.springframework.asm-3.0.1.release.jar
org.springframework.beans-3.1.2.release.jar
org.springframework.core-3.1.0.release.jar
poi-3.14-20160307.jar
swingx-all-1.6.4.jar
xml-apis-2.10.0.jar
xml-apis-ext.jar
xmlgraphics-commons-2.0.jar

Ilustración 101. Librerías utilizadas

```
String seleccion = "select \n"
        + "    tb_documentos.fecha,\n"
        + "    sum( total) total,\n"
        + "    sum( baseimponible) base,\n"
        + "    sum( importe_iva) iva,\n"
        + "    sum( importe_recargo) recargo,\n"
        + "    tb_documentos.id_abono\n"
        + "from tb_documentos\n"
        + "where \n"
        + "   (\n"
        + "      (id_abono is null )\n"
        + "   and \n"
        + "      (codigo_doc = 1)\n"
        + "   and \n"
        + "      (fecha between '" + datainicial + "' and '" + datafinal + "'))\n"
        + "group by tb_documentos.fecha, tb_documentos.id_abono \n"
        + "order by fecha";
```

Ilustración 102. SQL Ventas Diario

```
String seleccion = "select \n"
        + "    EXTRACT (WEEK FROM fecha) semana,\n"
        + "    EXTRACT (year FROM fecha) anyo,\n"
        + "    sum( total) total,\n"
        + "    sum( baseimponible) base,\n"
        + "    sum( importe_iva) iva,\n"
        + "    sum( importe_recargo) recargo,\n"
        + "    tb_documentos.id_abono\n"
        + "from tb_documentos\n"
        + "where \n"
        + "   (\n"
        + "      (id_abono is null )\n"
        + "   and \n"
        + "      (codigo_doc = 1)\n"
        + "   and \n"
        + "      (fecha between '" + datainicial + "' and '" + datafinal + "'))\n"
        + "group by semana, tb_documentos.id_abono,anyo\n"
        + "order by anyo,semana";
```

Ilustración 103. SQL Ventas Semana

```
String seleccion = "select \n"
        + "    EXTRACT (month FROM fecha) mes,\n"
        + "    EXTRACT (year FROM fecha) anyo,\n"
        + "    sum( total) total,\n"
        + "    sum( baseimponible) base,\n"
        + "    sum( importe_iva) iva,\n"
        + "    sum( importe_recargo) recargo,\n"
        + "    tb_documentos.id_abono\n"
        + "from tb_documentos\n"
        + "where \n"
        + "   (\n"
        + "      (id_abono is null )\n"
        + "   and \n"
        + "      (codigo_doc = 1)\n"
        + "   and \n"
        + "      (fecha between '" + datainicial + "' and '" + datafinal + "'))\n"
        + "group by mes, tb_documentos.id_abono,anyo\n"
        + "order by anyo,mes";
```

Ilustración 104. SQL Ventas Mes

```
String seleccion = "select \n"
        + "    sum( cantidad ) cantidad,\n"
        + "    sum( dto ) dto,\n"
        + "    sum( importe ) importe,\n"
        + "    sum( metros ) metros,\n"
        + "    tb_tipos.nombre\n"
        + "from tb_documentos\n"
        + "   inner join tb_lin_documentos on (tb_documentos.id = tb_lin_documentos.id_factura)\n"
        + "   inner join tb_articulos on (tb_lin_documentos.id_articulo = tb_articulos.id)\n"
        + "   inner join tb_tipos on (tb_articulos.id_tipo = tb_tipos.id)\n"
        + "where \n"
        + "   (\n"
        + "      (tb_documentos.codigo_doc = 1)\n"
        + "      and (id_abono is null )\n"
        + "      and \n"
        + "      (fecha between '" + datainicial + "' and '" + datafinal + "')\n"
        + "   )\n"
        + "group by tb_tipos.nombre\n"
        + "order by nombre";
```

Ilustración 105. SQL Ventas Familia

```
String seleccion = "select sum( total) total,\n"
        + "sum( baseimponible) base,\n"
        + "sum( importe_iva) iva,\n"
        + "sum( importe_recargo) recargo,\n"
        + "sum(totaldto) dto,\n"
        + "tb_clientes.nombre\n"
        + "from tb_documentos\n"
        + "inner join tb_clientes on (tb_documentos.id_cliente = tb_clientes.id)\n"
        +"where \n"
        + "(\n"
        + "(tb_documentos.codigo_doc = 1)\n"
        + "and (id_abono is null )\n"
        + "and \n"
        + "(fecha between '" + datainicial + "' and '" + datafinal + "')\n"
        + ")\n"
        + " group by tb_clientes.nombre\n"
        + "order by nombre";
```

Ilustración 106. SQL Ventas Cliente

Método calcular ventas por cliente especial

```
private void ventasClienteEspecial() {
   try {
    String año = txtAño.getText();
    String dataInicial = año + "/01/01", dataFinal = año + "/12/31";
    String seleccion = "delete from tb_informe_cliente";
    conn.prepareStatement(seleccion).executeUpdate();

    seleccion = "select sum(total) as totalt,id_cliente from
tb_documentos\n" +
     "where \n" +
     "(\n" +
     "(tb_documentos.codigo_doc = 1)\n" +
     " and (id_abono is null)\n" +
     " and \n" +
     " (fecha between '01/01/" + año + "' and '12/31/" + año + "')\n"
+
     "   )\n" +
     " group by id_cliente";
    PreparedStatement ps;
    ps = conn.prepareStatement(seleccion,
ResultSet.TYPE_SCROLL_INSENSITIVE,
     ResultSet.CONCUR_READ_ONLY, ResultSet.HOLD_CURSORS_OVER_COMMIT);

    ResultSet rs = ps.executeQuery();
    String ini1 = año + "/01/01", fin1 = año + "/03/31";
    String ini2 = año + "/04/01", fin2 = año + "/06/30";
    String ini3 = año + "/07/01", fin3 = año + "/09/30";
    String ini4 = año + "/10/01", fin4 = año + "/12/31";

    while (rs.next()) {
     if (rs.getDouble("totalt") >= 3000) {
      if (rs.getInt("id_cliente") != 0) {
       seleccion = "select nombre from tb_clientes\n" +
        "where id=" + rs.getInt("id_cliente");
       ps = conn.prepareStatement(seleccion,
ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY,
ResultSet.HOLD_CURSORS_OVER_COMMIT);

       ResultSet cogernombre = ps.executeQuery();
       cogernombre.next();
       String client = cogernombre.getString("nombre");
       if (cbCliente.getSelectedIndex() == 0 ||
cbCliente.getSelectedItem().toString().equalsIgnoreCase(client)) {
        String inicial = "", fin = "";
```

```
String trimestre = "";
int r = 4;
if (cbTrimestre.getSelectedIndex() == 1) {
 inicial = ini1;
 fin = fin1;
 trimestre = "Primer Trimestre";
 r = 1;
 dataInicial = inicial;
 dataFinal = fin;
} else if (cbTrimestre.getSelectedIndex() == 2) {
 inicial = ini2;
 fin = fin2;
 trimestre = "Segundo Trimestre";
 r = 1;
 dataInicial = inicial;
 dataFinal = fin;
} else if (cbTrimestre.getSelectedIndex() == 3) {
 inicial = ini3;
 fin = fin3;
 trimestre = "Tercer Trimestre";
 r = 1;
 dataInicial = inicial;
 dataFinal = fin;
} else if (cbTrimestre.getSelectedIndex() == 4) {
 inicial = ini4;
 fin = fin4;
 trimestre = "Cuarto Trimestre";
 r = 1;
 dataInicial = inicial;
 dataFinal = fin;
}
for (int i = 0; i < r; i++) {
 if (cbTrimestre.getSelectedIndex() == 0) {
  switch (i) {
   case 0:
    inicial = ini1;
    fin = fin1;
    trimestre = "Primer Trimestre";
    break;
   case 1:
    inicial = ini2;
    fin = fin2;
    trimestre = "Segundo Trimestre";
    break;
   case 2:
    inicial = ini3;
    fin = fin3;
    trimestre = "Tercer Trimestre";
```

```
            break;
           case 3:
            inicial = ini4;
            fin = fin4;
            trimestre = "Cuarto Trimestre";
            break;
          }
         }
         seleccion = "select * from tb_documentos\n" +
          "where( (tb_documentos.codigo_doc = 1)\n" +
          " and (id_abono is null )\n" +
          "    and \n" +
          "(fecha between '" + inicial + "' and '" + fin + "')\n" +
          "and id_cliente=" + rs.getInt("id_cliente") + "\n" +
          " )";
         ps = conn.prepareStatement(seleccion,
ResultSet.TYPE_SCROLL_INSENSITIVE,
          ResultSet.CONCUR_READ_ONLY,
ResultSet.HOLD_CURSORS_OVER_COMMIT);

         ResultSet dentro = ps.executeQuery();

         while (dentro.next()) {
          int cliente = dentro.getInt("id_cliente");

          seleccion = "select nombre from tb_clientes\n" +
           "where id=" + cliente;
          ps = conn.prepareStatement(seleccion,
ResultSet.TYPE_SCROLL_INSENSITIVE,
           ResultSet.CONCUR_READ_ONLY,
ResultSet.HOLD_CURSORS_OVER_COMMIT);

          ResultSet rz = ps.executeQuery();
          rz.next();
          String nombre = rz.getString("nombre");

          seleccion = "insert into
tb_informe_cliente(nombre,fecha,factura,base,iva,recargo,dto,total,tri
mestre) \n" +
           " values ('" + nombre + "','" + dentro.getDate("fecha") +
"'," + dentro.getInt("numfactura") + "," +
dentro.getDouble("baseimponible") + "," +
dentro.getDouble("importe_iva") + "," +
           "" + dentro.getDouble("importe_recargo") + "" +
           "," + dentro.getDouble("totaldto") + "," +
dentro.getDouble("total") + ",'" + trimestre + "')";
          conn.prepareStatement(seleccion).executeUpdate();
         }}}}}}
```

Capítulo 7: Bibliografía

Web oficial de Oracle: https://www.oracle.com/es/java/

Web oficial de Firebird: https://firebirdsql.org/

Web oficial de Netbeans: https://netbeans.org/

Web oficial de Jaspersoft: https://www.jaspersoft.com/es

Creando informes JasperReports: http://codigoxules.org/creando-informes-java-jasperreports-desde-jaspersoft-studio/

Manual firebird y java: https://firebirdsql.org/file/Jaybird_2_1_JDBC_driver_manual.pdf

Agencia Española de Protección de Datos: https://www.aepd.es/

Autoincremental Firebird: https://firebird21.wordpress.com/2013/10/02/el-generador-autoincremental/

Software a medida vs comercial: https://webprogramacion.com/366/blog-informatica-tecnologia/comparativa-software-a-medida-vs-software-comercial.aspx

Página oficial Odoo: https://www.odoo.com/es_ES/

Capítulo 8: Conclusiones y líneas futuras

Tras la realización y posterior implantación del software en la empresa, toca analizar si se han cumplido los objetivos marcados al principio del proyecto.

El primer objetivo era seleccionar las tecnologías a utilizar, base de datos, lenguaje de programación, sistema de informes, etc. Todo ello debía ser bajo el licenciamiento de software libre ya que la empresa no quería pagar licencias. Por ello se eligieron herramientas de software libre y libres de licencias.

El segundo objetivo era implementar una aplicación sencilla y con mucho potencial que contribuya a la gestión de la empresa y aligerara el trabajo de oficina que era uno de los grandes problemas a los que se enfrentaba la empresa.

Una vez implantado el software en la empresa, la gestión general ha mejorado y el trabajo de oficina se ha reducido y también mejorado considerablemente, facilitándoles el trabajo y haciendo que pasen menos tiempo en la oficina, por lo que la producción también ha mejorado.

La mejora en cuanto a calidad de trabajo descrita por los trabajadores después de meses de trabajo es infinita. El poder controlar todo lo que hacen de manera sencilla, y sin tener que perder horas para poder hacer cualquier trámite.

Esto lleva a la conclusión de que, tras la finalización del proyecto, se puede considerar que los objetivos marcados al principio de este han sido correctamente alcanzados y que tener una buena infraestructura informática para la gestión de una empresa lleva a un ahorro en tiempo y dinero. Por el contrario, si no se tiene una buena infraestructura o la que se tiene no se mantiene y actualiza (como era el caso), la pérdida de tiempo y dinero puede ser mayor que la posible inversión en mantenerla al día.

8.1 Líneas futuras

En este proyecto se han conseguido todos los objetivos propuestos, pero en un programa que está funcionando en una empresa día tras día, las necesidades van a ir incrementándose con lo que el desarrollo no se puede dar por finalizado.

A continuación, se nombran algunas de las líneas futuras que se van a implementar, algunas propuestas por el cliente y otras como mejoras del propio programa.

- Mejorar el sistema de licencias.
- Optimizar ciertas partes del código.
- Revisar el código para que pueda funcionar en varios terminales a la vez.
- Implantación de nuevos informes.
- Estudiar posible implantación de la contabilidad.
- Integración con Prestashop.
- Integración con sistema de agencia de envíos.
- Integración de transferencias bancarias.
- Etc.

8.2 Opinión personal

Este Trabajo Final de Grado es la culminación de un periodo de seis años (FPS y Grado) de mi vida, donde he adquirido y aprendido competencias a nivel personal y profesional. Tras este periodo he podido aprender una profesión y darme cuenta de que hay que seguir formándose constantemente para estar siempre al día en este mundo tecnológico.

Cuando se me presentó la oportunidad de realizar este proyecto a la empresa, pensé que no conseguiría sacarlo adelante puesto que la complejidad que se planteaba era grande, aun así, el investigar y trabajar cada día supuso ir logrando pequeñas metas y tras mucho trabajo, conseguir finalizarlo.

Actualmente la empresa está trabajando con el software a diario y según las opiniones que me han ido trasladando, el software cumple las funciones que querían con creces, siendo lo errores hasta el día de hoy de cero.

Para concluir, gracias a este proyecto he conseguido enriquecerme profesionalmente adquiriendo más experiencia en este ámbito; la que me ha ayudado a acceder a ofertas laborales que antes de realizarlo no hubiese podido alcanzar.