UNIVERSITAT OBERTA DE CATALUNYA

eFactura

Aplicación de gestión de facturas electrónicas basado en la tecnología .NET

Francisco Javier Cañabate Bernete
Enginyeria Tècnica de Informàtica de Sistemes.
Consultor: Jordi CeballosVillach

Junio de 2008

Resumen.

En la presente memoria se presenta el trabajo realizado en el proyecto de desarrollo de la aplicación eFactura que pretende realizar la gestión de las facturas de forma electrónica, sin el uso de papel.

El proyecto se ha realizado utilizando el Framework .NET de Microsoft. La aparición de este framework supuso un importante avance pues en una misma plataforma, se integran diversas tecnologías de gran aceptación en el mundo tecnológico, como pueden ser ADO.NET, ASP.NET, AJAX, LINQ o VSTO. Esta última tecnología, Visual Studio Tools for Office, ha sido la utilizada en este proyecto y permite adjuntar código escrito en diversos lenguajes de alto nivel a las aplicaciones del entorno ofimático Office, con lo que se le añade la capacidad de hacer casi cualquier cosa con esta suite. En concreto, en este proyecto, desde un libre de Excel, se accede a una base de datos realizada en Access, se muestra su información, se añaden y modifican registros, se realizan consultas, con los datos obtenidos, se crean documentos de Word y de Adobe Acrobat, se firman digitalmente, se comprimen en formato Zip y se envían por correo electrónico a sus destinatarios, también se puede obtener un informe sobre los datos necesarios para la presentación de la declaración del IRPF, así como un gráfico por cliente de los ingresos obtenidos junto con otro gráfico de los ingresos totales. Todo esto de una forma sencilla, sin apenas movimientos de ratón ni demasiados desplazamientos. Se ha procurado buscar el mayor nivel de usabilidad, procurando reducir al máximo el número de acciones por parte del usuario para obtener el resultado buscado.

La tecnología .NET permite usar, como ya se ha comentado, varios lenguajes de programación como Visual Basic.NET o C#, incluso mezclar código de diversos lenguajes. Este proyecto se ha realizado usando el lenguaje C#, un lenguaje orientado a objetos, de gran potencia y altamente integrado en la plataforma .NET. Como entorno de programación se ha utilizado, como no podía ser de otra forma, Visual Studio en su versión 2008.

Contenido

Índice de ilustraciones.

1. Introducción.

1.1. Justificación y contexto del proyecto.

El presente proyecto se enmarca dentro de la tecnología .NET de Microsoft. Esta tecnología ofrece múltiples y potentes herramientas para el desarrollo de una gran variedad de tipos de aplicaciones, desde tratamientos de datos, creación de servicios web, creación de sitios webs dinámicos etc. En concreto, este proyecto se centra en el uso de una parte de .NET que permite asociar código en un lenguaje de programación de alto nivel, en este caso se ha usado C#, a una aplicación o un documento de la suite ofimática de Microsoft, Office. Para conseguirlo, se debe usar el complemento "Visual Studio Tools For Office", dentro del entorno de desarrollo "Visual Studio 2008". En el diseño, se ha usado la versión 2007 profesional de esta suite.

La motivación que me ha llevado a la elección de este proyecto es la gran implantación que tiene Office en el entorno empresarial y aun domestico y lo atractivo de poder añadir potencia a estas aplicaciones, evitando el uso de aplicaciones externas que interrumpan el flujo habitual de trabajo. Creo que es un campo con un futuro interesante.

1.2. Descripción del proyecto.

El presente proyecto, "Factura electrónica", tiene como objetivo el implantar la facturación electrónica, así como facilitar su completa gestión, utilizando las herramientas de Microsoft Office, complementadas cuando sea necesario, con funcionalidades programadas incrustadas en una plantilla de Excel. Con ello se busca adecuarse a las últimas tendencias en facturación, facilitar la gestión liberando de parte de la carga que los métodos anteriores suponían, y mejorar la imagen de cara a los clientes.

En este proyecto se precisa de un único modulo software en el que se diferencian cuatro grupos funcionales muy interrelacionados. Primero está la gestión de facturas, segundo, la gestión de datos de clientes, tercero, la gestión del envió de correo electrónico y cuarto y último, la presentación de informes.

Físicamente todos los grupos funcionales se ejecutan en un solo PC y con una única interfaz que es una página de Excel, si bien se usan ficheros en formato Word y Pdf para la generación de las facturas, Outlook para el envío de correo electrónico y Access para almacenar los datos persistentes como clientes y facturas. Todos estos elementos software están instalados en el PC del usuario, no necesitando más componentes que una conexión a internet para el envío de correo electrónico.

En esta tecnología, existen dos posibles caminos de para adjuntar código a Office.

Una es personalización de nivel de documento y la otra personalización de nivel de aplicación.

La diferencia entre los dos sistemas es que en el primero, el código está vinculado a un documento concreto, mientras que en la personalización a nivel de aplicación, el código está disponible para una aplicación y para todos sus documentos.

En este caso se ha optado por una personalización a nivel de documento puesto que la funcionalidad que se ha añadido no es de tipo genérico y no sería de utilidad tenerla disponible para todos los documentos.

La arquitectura de una personalización de nivel de documento, viene ilustrada en la siguiente imagen extraída de la dirección de internet http://msdn2.microsoft.com/es-es/library/zcfbd2sk.aspx.

En la imagen se puede ver como desde una aplicación de Microsoft Office, el usuario abre el documento. Esta acción dispara la propiedad _AssemblyLocation provocando que inicie el proceso que derivará en la preparación del código agregado al documento.

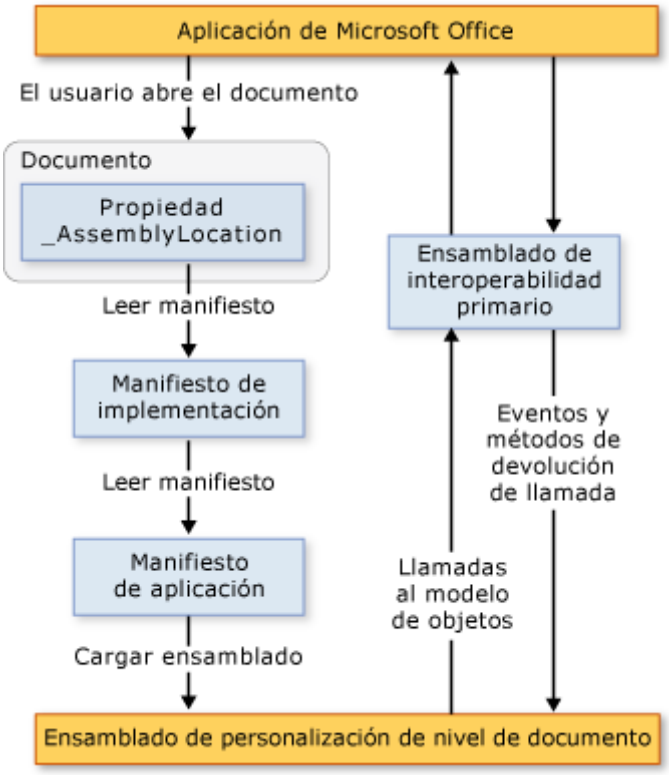


Ilustración 1. Arquitectura.

1.3. Objetivos.

El objetivo de este proyecto es integrar las aplicaciones de Microsoft office mediante .NET y la creación de los documentos necesarios, para crear un entorno que permita gestionar de manera automatizada las facturas que se han de enviar a los clientes. La aplicación debe permitir la gestión de una base de datos de los clientes, creación de informes y gráficas y generación y envío automático de facturas firmadas con certificado digital de forma que se cumplan los requisitos de la normativa de Hacienda para su total legalidad.

A nivel personal y académico, los objetivos son comprobar y demostrar la asimilación de los conocimientos impartidos en las diferentes asignaturas cursadas durante los estudios, como son, Enginyeria del software, las asignaturas del área de programación, base de datos, etc..

Otro de los objetivos es el de demostrar la capacitación básica que todo ingeniero debe tener, en buscar y asimilar información necesaria para la resolución de un problema usando las herramientas a su alcance. De entre estas posibles herramientas, destaca actualmente Internet y siendo la UOC (Universitat Oberta de Catalunya) una universidad no presencial con un uso intenso de la red Internet, parece lógico que esta búsqueda sea de forma prioritaria en Internet. Este medio presenta un vehículo de difusión inmediato para cualquier nueva tecnología que se lance al mercado, sin los retardos que supone el tiempo que un escritor tarda en redactar un libro, imprimirlo, distribuirlo y publicitarlo.

Un último objetivo es el de tener un primer contacto con la tecnología .NET y la herramienta profesional de desarrollo Visual Studio, así como su extensión Visual Studio Tools for Office.

1.4. Planificación.

1.4.1. Ciclo de vida.

En este proyecto se ha utilizado el método de desarrollo de ingeniería del software denominado ciclo de vida clásico o en cascada con prototipaje y diseño orientado a objetos.

Este método se basa en etapas sucesivas. Cada etapa debe ser completada antes de iniciar la siguiente pues el producto final de una etapa se utiliza como base de inicio de la siguiente. Las diferentes etapas de la que consta, serían:

- Análisis previo. Esta fase pretende un primer acercamiento al problema a resolver, así como una temporización previa del proyecto.
- Análisis de requisitos. Se describen los casos de uso, es decir las actividades que el sistema debe realizar y las clases principales que se creen necesarias sin restringirse a un lenguaje de programación concreto.
- Diseño. Esta fase define las clases y relaciones de la base de datos, en el lenguaje a utilizar en el desarrollo del software.
- Implementación y pruebas. La fase de implementación comporta en desarrollo del software y su prueba.
- Final. Se incluye en esta fase la instalación y posterior mantenimiento del producto obtenido en la fase de implementación.

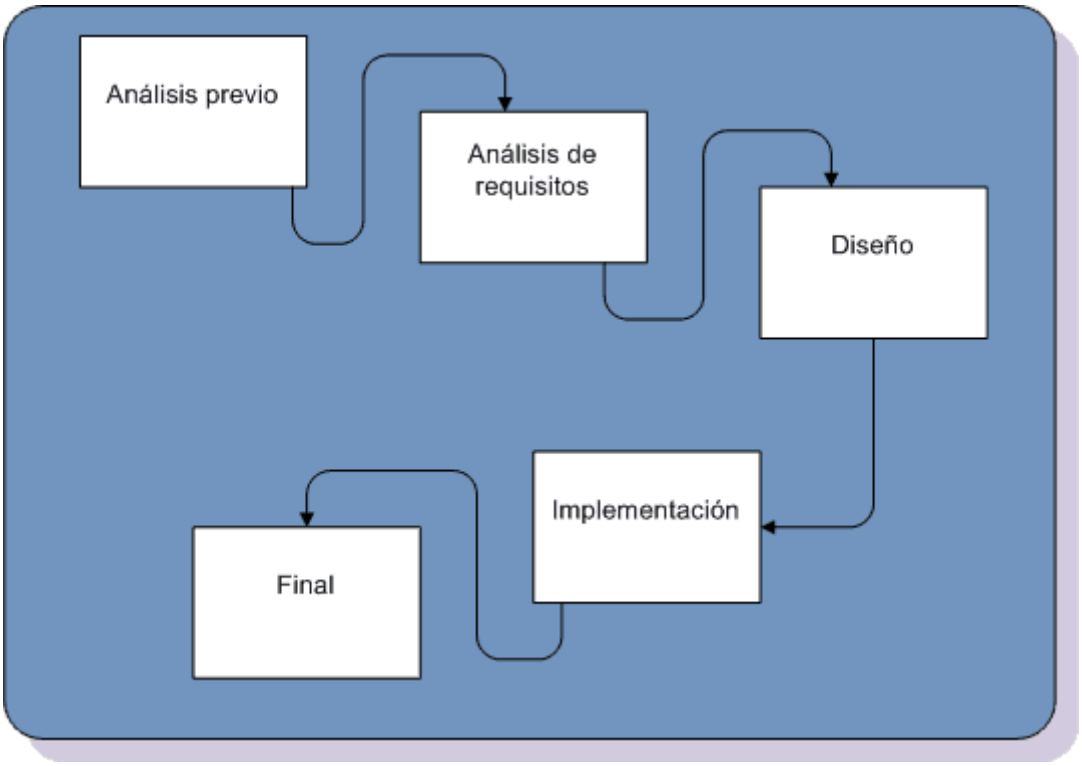


Ilustración 2. Ciclo de vida en cascada.

1.4.2. Temporización.

En este apartado se expondrán los detalles relacionados con la temporización y contenido de cada etapa.

1.4.2.1. Detalle de actividades de cada etapa.

- Análisis previo/planificación.
  - Elección del proyecto.
  - Definición.
  - Planificación.
  - Creación y entrega de documento plan de proyecto.

- Análisis de requisitos.
  - Especificación de requisitos funcionales.
  - Especificación de requisitos no funcionales.
  - Creación de documento de análisis.

- Diseño.
  - Diseño de la arquitectura.
  - Diseño de clases.
  - Diseño de persistencia.
  - Diseño del prototipo.

- Creación del documento de diseño.
- Entrega de documentación y prototipo.

- Implementación.
  - Implementación de la persistencia.
  - Implementación de la interfaz principal de usuario.
  - Implementación de la presentación de datos en la interfaz de usuario.
  - Implementación de Acceso a datos para su modificación.
  - Implementación de la creación de los documentos de las facturas.
  - Implementación de la firma digital de los documentos.
  - Implementación del envío por correo de las facturas.
  - Implementación de la compresión de las facturas.
  - Publicación e instalación de la implementación.
  - Creación del documento de instalación y de uso de la implementación.

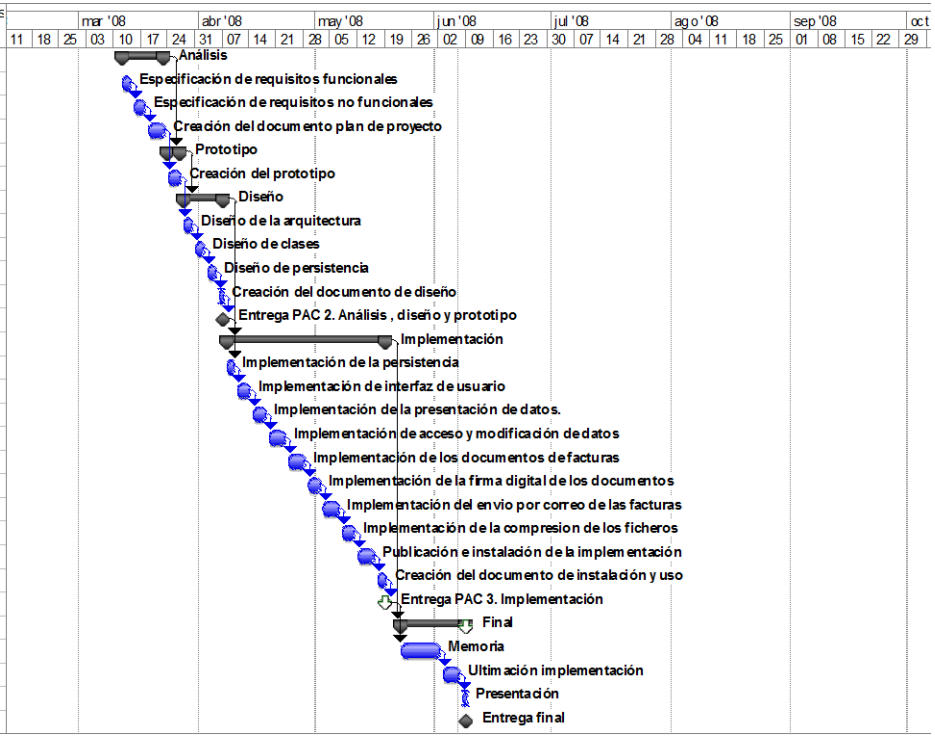
- Final.
  - Ultimación de la implementación.
  - Creación de la memoria.
  - Grabación de la presentación.
  - Entrega final.

1.4.2.2. Diagrama de Gantt.

A continuación se incluye el diagrama de Gantt del proyecto en el que se puede observar de forma gráfica la distribución en el tiempo de las tareas mencionadas en el punto anterior.

Ilustración 3. Diagrama de Gantt.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predeces
7	**Análisis**	**10 días**	**mar 11/03/08**	**sáb 22/03/08**	**1**
8	Especificación de requisitos funcionales	2 días	mar 11/03/08	jue 13/03/08	6
9	Especificación de requisitos no funcionales	3 días	vie 14/03/08	lun 17/03/08	8
10	Creación del documento plan de proyecto	3 días	mar 18/03/08	sáb 22/03/08	9
11	**Prototipo**	**3 días**	**dom 23/03/08**	**mié 26/03/08**	**7**
12	Creación del prototipo	3 días	dom 23/03/08	mié 26/03/08	10
13	**Diseño**	**11 días**	**jue 27/03/08**	**dom 06/04/08**	**11**
14	Diseño de la arquitectura	2 días	jue 27/03/08	sáb 29/03/08	12
15	Diseño de clases	2 días	dom 30/03/08	mar 01/04/08	14
16	Diseño de persistencia	2 días	mié 02/04/08	vie 04/04/08	15
17	Creación del documento de diseño	2 días	sáb 05/04/08	dom 06/04/08	16
18	Entrega PAC 2. Análisis , diseño y prototipo	0 días?	dom 06/04/08	dom 06/04/08	17
19	**Implementación**	**41 días**	**lun 07/04/08**	**dom 18/05/08**	**13**
20	Implementación de la persistencia	3 días	lun 07/04/08	mié 09/04/08	18
21	Implementación de interfaz de usuario	4 días	jue 10/04/08	dom 13/04/08	20
22	Implementación de la presentación de datos.	4 días	lun 14/04/08	jue 17/04/08	21
23	Implementación de acceso y modificación de datos	5 días	vie 18/04/08	mar 22/04/08	22
24	Implementación de los documentos de facturas	5 días	mié 23/04/08	dom 27/04/08	23
25	Implementación de la firma digital de los documentos	4 días	lun 28/04/08	jue 01/05/08	24
26	Implementación del envio por correo de las facturas	5 días	vie 02/05/08	mar 06/05/08	25
27	Implementación de la compresion de los ficheros	4 días	mié 07/05/08	sáb 10/05/08	26
28	Publicación e instalación de la implementación	4 días	dom 11/05/08	jue 15/05/08	27
29	Creación del documento de instalación y uso	2 días	vie 16/05/08	dom 18/05/08	28
30	Entrega PAC 3. Implementación	0 días	dom 18/05/08	dom 18/05/08	29
31	**Final**	**18 días**	**jue 22/05/08**	**dom 08/06/08**	**19**
32	Memoria	11 días	jue 22/05/08	dom 01/06/08	30
33	Ultimación implementación	5 días	lun 02/06/08	vie 06/06/08	32
34	Presentación	2 días	sáb 07/06/08	dom 08/06/08	33
35	Entrega final	0 días	dom 08/06/08	dom 08/06/08	



1.4.3. Herramientas utilizadas.

Las herramientas utilizadas en este proyecto son las siguientes:

- Microsoft Office 2007 Professional. Suite ofimática. Excel Word, Access, Outlook, Project y Visio.
- Visual Studio 2008. Entorno de desarrollo.
- Visual Studio Tools for Office. Complemento de Visual Studio para Office.
- Itextsharp. Librería para la confección y manipulación de ficheros Pdf.
- Ionic.Utils.Zip. librería para la compresión de archivos en formato zip.
- ArgoUML. Creación de gráficos de casos de uso.
- Camtasia Studio 5. Presentación virtual del proyecto.

1.4.4. Productos obtenidos.

- Plan de proyecto.
- Prototipo.
- Manual de uso del prototipo.
- Documento de análisis.
- Documento de diseño.
- Manual de instalación.
- Manual de usuario.
- Aplicación.
- Memoria de proyecto.
- Presentación.

2. Requisitos iniciales.

A continuación se describe el escenario inicial del proyecto.

2.1. Escenario de partida.

Un cliente que realiza sitios web para pequeñas empresas desea comenzar a utilizar la factura en formato electrónico así como facilitar sus labores de creación y gestión de facturas. Desea también que se le facilite la recopilación de la información necesaria para presentar a hacienda en la declaración de la renta. El cliente ha utilizado hasta ahora el paquete ofimático Microsoft Office y desea continuar usándolo ya que se siente cómodo y seguro con él. Por todo esto, se plantea de utilizar las tecnologías ofrecidos por la plataforma Framework .NET, concretamente el paquete Visual Studio Tools for Office, a través del entorno de programación Visual Studio 2008. Como lenguaje de programación se utilizará C#, debido a su gran similitud con el lenguaje

Java que es el más usado por el equipo de programación que se encargara del proyecto.

2.2. Usuarios a considerar.

En este proyecto se ha considerado un único usuario ya que debido a su relativa simplicidad y a que no existen distintos roles excluyentes, no es necesario ningún otro usuario.

2.3. Requisitos funcionales.

A continuación paso a detallar los requisitos funcionales, entendiendo como tal las funcionalidades que el usuario precisa obtener del software a desarrollar.

2.3.1. Descripción del guion.

2.3.1.1. Crear cliente.

El sistema debe presentar una tabla con los datos de los clientes ya introducidos anteriormente y posibilitar al usuario la introducción de nuevos clientes. Tras finalizar la introducción de datos y mediante la pulsación del botón “Guardar cambios”, el usuario indica al sistema que los datos introducidos deben ser grabados en la base de datos. A continuación el sistema procede a acceder a la base de datos y actualizarla con los nuevos datos.

2.3.1.2. Crear factura.

El sistema debe presentar una tabla con las facturas ya introducidas anteriormente y posibilitar al usuario la introducción de nuevas facturas. Tras finalizar la introducción de datos y mediante la pulsación del botón “Guardar cambios”, el usuario indica al sistema que los datos introducidos deben ser grabados en la base de datos. A continuación el sistema procede a acceder a la base de datos y actualizarla con los nuevos datos.

2.3.1.3. Modificar cliente.

Como precondición el usuario ha seleccionado un cliente en la tabla de clientes, y ha modificado alguno/s de los datos del cliente. Tras la modificación pertinente, el usuario mediante la pulsación del botón “Guardar cambios”, indica al sistema que la modificación de datos a concluido y a continuación el sistema procede a acceder a la base de datos y actualizar los datos del cliente.

2.3.1.4. Modificar factura.

Como precondición el usuario ha seleccionado un cliente en la tabla de clientes, una factura en la tabla de facturas del cliente y ha modificado alguno/s de los datos de la factura. Tras la modificación pertinente, el usuario mediante la pulsación del botón “Guardar cambios”,

indica al sistema que la modificación de datos a concluido y a continuación el sistema procede a acceder a la base de datos y actualizar los datos de la factura.

Una de las modificaciones posibles será mantener actualizado el estado de la factura (pendiente de envío, enviada, cobrada, rechazada…).

2.3.1.5. Borrar cliente.

Como precondición el usuario ha seleccionado un cliente en la tabla de clientes. La pulsación del botón "Borrar cliente" provoca la eliminación del cliente y de todas sus facturas de las tablas. El usuario mediante la pulsación del botón "Guardar cambios", indica al sistema que proceda con la eliminación permanente del cliente y de todas sus facturas de la base de datos.

2.3.1.6. Borrar factura.

Como precondición el usuario ha seleccionado un cliente en la tabla de clientes y una factura en la tabla de facturas del. La pulsación del botón "Borrar factura" provoca la eliminación de la factura de la tabla. El usuario mediante la pulsación del botón "Guardar cambios", indica al sistema que proceda con la eliminación permanente de la factura de la base de datos.

2.3.1.7. Enviar factura.

Como precondición el usuario debe haber seleccionado un cliente en la tabla de clientes y una factura en la tabla de facturas del cliente. Tras la pulsación por parte del usuario del botón de "Enviar factura", el sistema creará el documento de la factura en el formato elegido por el cliente (PDF o DOC). El fichero resultante se certificará, se comprimirá y se enviará por correo electrónico usando Outlook. Una vez enviado el fichero, se modificará el estado de la factura a enviada.

2.3.1.8. Crear informe IRPF.

Tras la pulsación por parte del cliente del botón "Informe IRPF", el sistema presentará una lista con la información de importe bruto, IVA correspondiente e importe sin IVA en dos listas. En una se verán las cantidades de forma trimestral, y otra en la que se verán las cantidades anuales.

2.3.1.9. Crear informe gráfico.

El usuario selecciona un cliente en la tabla de clientes. Tras la pulsación por parte del cliente del botón "Informe gráfico", el sistema presentará un gráfico de barras con los importes de las facturas mensuales del cliente seleccionado en la tabla de clientes. La información se presentará del año en curso y de los dos años anteriores.

2.4. Requisitos no funcionales.

2.4.1. Requisitos de interfaz.

No se estima necesaria la creación de un interfaz propio de la aplicación ya que lo que se busca es la integración de esta en el entorno de Office, por lo que, la interfaz, será la propia interfaz del programa Microsoft Excel, a través de la cual, mediante incrustación de un panel de acciones que contendrá los botones necesarios se adjuntará la funcionalidad añadida en este proyecto a la propia del sistema Office.

Las pantallas de entrada de datos serán tablas contenidas en hojas de Excel.

2.4.2. Requisitos de seguridad.

No se ha previsto añadir sistemas de seguridad adicionales a los usados hasta ahora por el cliente.

2.4.3. Requisitos de información.

La información de cada cliente será:

CIF de la empresa o DNI del cliente, nombre de la empresa o del cliente, dirección social, teléfono de contacto, teléfono móvil, fax, dirección de correo electrónico, preferencia de formato de las facturas.

La información de cada factura será:

Código de la factura, datos del cliente, descripción del motivo de la factura, importe de la factura, tipo de IVA aplicado, importe del IVA aplicado, estado de la factura.

Así mismo y con el objeto de poder certificar los ficheros contenedores de las facturas, se precisará de un certificado con clave privada proporcionado por un centro de certificación oficial.

3. Análisis del sistema.

En este apartado se expone el resultado de la etapa de análisis en forma de casos de uso.

3.1. Diagramas de casos de uso.

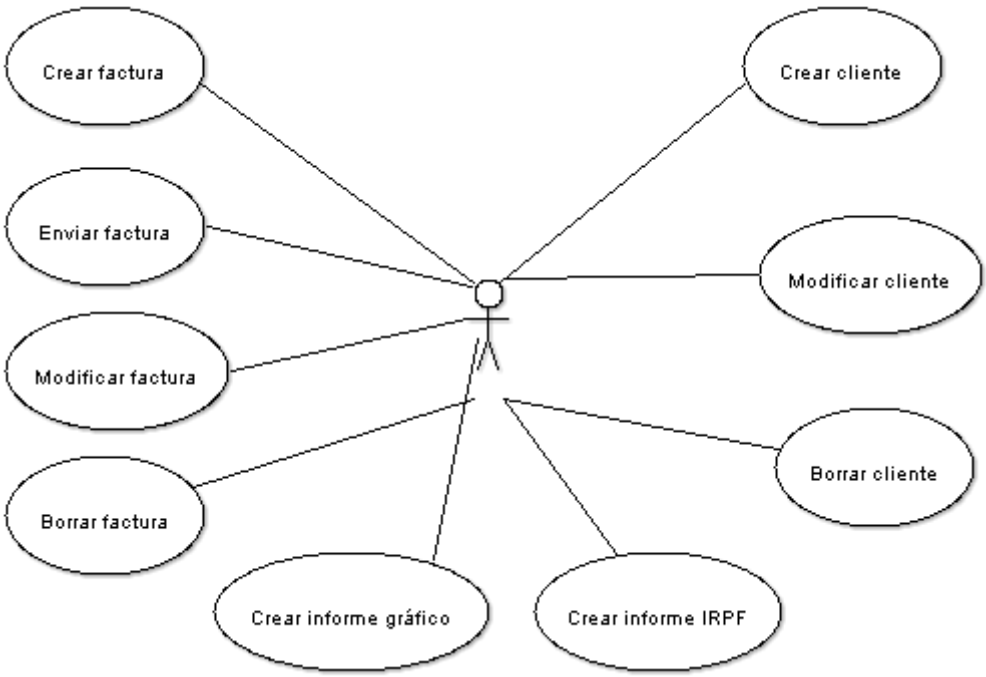


Ilustración 4. Diagrama de casos de uso.

Este es el diagrama de casos de uso que refleja las acciones que puede emprender el usuario, o como se denomina en el argot de los casos de uso, el actor. En el podemos observar que solo aparece un actor, el actor principal, y seis casos de uso diferentes. A continuación pasamos a detallar los casos de uso.

3.2. Descripción textual de los casos de uso.

En este apartado se realiza una descripción textual e individualizada de los casos de uso para determinar más claramente su contenido.

3.2.1. CU01-Crear cliente.

Identificador	**CU01**
Nombre	**Crear cliente**
Autor	**Francisco Javier Cañabate**
Resumen	El sistema debe presentar una tabla con los datos de los clientes ya introducidos anteriormente y posibilitar al actor la introducción de nuevos clientes. Tras finalizar la introducción de datos y mediante la pulsación del botón "Guardar cambios", el actor indica al sistema que los datos introducidos deben ser grabados en la base de datos. A continuación el sistema procede a acceder a la base de datos y actualizarla con los nuevos datos.
Actor	**Usuario**
Precondiciones	**El cliente no existe en la base de datos**
Postcondiciones	**El cliente ha sido añadido a la base de datos**
Flujo normal	**1. El sistema presenta una tabla con los clientes contenidos en la base de datos.** **2. El actor rellena la última fila de la tabla, que estará vacía y marcada con un asterisco, con los datos del nuevo cliente.** **3. El actor confirma los datos introducidos pulsando el botón "Guardar cambios".** **4. El sistema accede a la base de datos y se almacenan los datos del cliente.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.2. CU02-Crear factura.

Identificador	**CU02**
Nombre	**Crear factura**
Autor	**Francisco Javier Cañabate**
Resumen	El sistema debe presentar una tabla con las facturas ya introducidos anteriormente y posibilitar al actor la introducción de nuevas facturas. Tras finalizar la introducción de datos y mediante la pulsación del botón "Guardar cambios", el actor indica al sistema que los datos introducidos deben ser grabados en la base de datos. A continuación el sistema procede a acceder a la base de datos y actualizarla con los nuevos datos.
Actor	**Usuario**
Precondiciones	**El cliente existe en la base de datos. La factura no existe en la base de datos.**
Postcondiciones	**La factura ha sido incorporada a la base de datos.**
Flujo normal	**1. El actor selecciona al cliente en la tabla de clientes.** **2. El sistema El sistema actualiza la tabla de facturas con las pertenecientes al cliente seleccionado.** **3. El actor rellena la última fila de la tabla, que estará vacía y marcada con un asterisco, con los datos del a nueva factura.** **4. El actor confirma los datos introducidos pulsando el botón "Guardar cambios".** **5. El sistema accede a la base de datos y se almacenan los datos de la factura.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.3. CU03-Modificar cliente.

Identificador	**CU03**
Nombre	**Modificar cliente**
Autor	**Francisco Javier Cañabate**
Resumen	Como precondición el actor ha seleccionado un cliente en la tabla de clientes y ha modificado alguno/s de los datos del cliente. Tras la modificación pertinente, el actor mediante la pulsación del botón “Guardar cambios”, indica al sistema que la modificación de datos a concluido y a continuación el sistema procede a acceder a la base de datos y actualizar los datos del cliente.
Actor	**Usuario**
Precondiciones	**El cliente existe en la base de datos.**
Postcondiciones	**El cliente ha sido modificado en la base de datos.**
Flujo normal	**1. El actor selecciona el cliente en la tabla de clientes.** **2. El actor modifica los datos y confirma la modificación con la pulsación del botón ”Guardar cambios”.** **5. El sistema almacena los datos del cliente modificado en la base de datos.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.4. CU04-Modificar factura.

Identificador	**CU04**
Nombre	**Modificar factura**
Autor	**Francisco Javier Cañabate**
Resumen	Como precondición el actor ha seleccionado un cliente en la tabla de clientes, una factura en la tabla de facturas del cliente y ha modificado alguno/s de los datos de la factura. Tras la modificación pertinente, el actor mediante la pulsación del botón "Guardar cambios", indica al sistema que la modificación de datos a concluido y a continuación el sistema procede a acceder a la base de datos y actualizar los datos de la factura. Una de las modificaciones posibles será mantener actualizado el estado de la factura (pendiente de envío, enviada, cobrada, rechazada…).
Actor	**Usuario**
Precondiciones	**El cliente existe en la base de datos.** **La factura existe en la base de datos.**
Postcondiciones	**La factura ha sido modificada en la base de datos.**
Flujo normal	**1. El actor selecciona el cliente en la tabla de clientes.** **2. El sistema actualiza la tabla facturas con las pertenecientes al cliente seleccionado.** **3. El actor modifica los datos y confirma la modificación con la pulsación del botón "Guardar cambios".** **5. El sistema almacena los datos de la factura modificada en la base de datos.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.5. CU05-Borrar cliente.

Identificador	**CU05**
Nombre	**Borrar cliente**
Autor	**Francisco Javier Cañabate**
Resumen	Como precondición el actor ha seleccionado un cliente en la tabla de clientes. El actor pulsa el botón "Borrar cliente" y el sistema elimina de la tabla "Clientes" al cliente seleccionado así como todas sus facturas de la tabla "Facturas". El actor mediante la pulsación del botón "Guardar cambios", indica al sistema que al sistema que proceda a acceder a la base de datos y elimine los datos de forma permanente.
Actor	**Usuario**
Precondiciones	**El cliente existe en la base de datos.**
Postcondiciones	**El cliente ha sido eliminado de la base de datos.**
Flujo normal	**1. El actor selecciona el cliente en la tabla de clientes.** **2. El actor pulsa el botón "Borrar cliente".** **3. El actor confirma la eliminación con la pulsación del botón "Guardar cambios".** **5. El sistema accede a la base de datos y elimina de forma permanente el cliente y sus facturas.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.6. CU06-Borrar factura.

Identificador	**CU06**
Nombre	**Borrar factura**
Autor	**Francisco Javier Cañabate**
Resumen	Como precondición el actor ha seleccionado un cliente en la tabla de clientes y una factura en la tabla de facturas del cliente. El actor mediante la pulsación del botón "Borrar factura", indica al sistema que elimine la factura seleccionada, a continuación el actor pulsa el botón "Guardar cambios. El sistema procede a acceder a la base de datos y eliminar los datos de la factura.
Actor	**Usuario**
Precondiciones	**El cliente existe en la base de datos.** **La factura existe en la base de datos.**
Postcondiciones	**La factura ha sido eliminada de la base de datos.**
Flujo normal	**1. El actor selecciona el cliente en la tabla de clientes.** **2. El sistema actualiza la tabla facturas con las pertenecientes al cliente seleccionado.** **3. El actor selecciona una factura y pulsa el botón "Borrar factura.** **4. El sistema elimina la factura de la tabla "Facturas".** **5. El cliente pulsa el botón "Guardar cambios".** **6. El sistema elimina los datos de la factura de la base de datos.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.7. CU07-Enviar factura.

Identificador	**CU07**
Nombre	**Enviar factura**
Autor	**Francisco Javier Cañabate**
Resumen	Como precondición el actor debe haber seleccionado un cliente en la tabla de clientes y una factura en la tabla de facturas del cliente. Tras la pulsación por parte del actor del botón de "Enviar factura", el sistema creará el documento de la factura en el formato elegido por el cliente (PDF o DOC). El fichero resultante se certificará, se comprimirá y se enviará por correo electrónico usando Outlook. Una vez enviado el fichero, se modificará el estado de la factura a enviada.
Actor	**Usuario**
Precondiciones	**La factura existe en la base de datos.**
Postcondiciones	**La factura ha sido enviada por correo electrónico al cliente y se ha registrado este envío en la base de datos.**
Flujo normal	**1. El actor selecciona el cliente en la tabla de clientes.** **2. El sistema actualiza la tabla facturas con las pertenecientes al cliente seleccionado.** **3. El actor selecciona la factura en la tabla de facturas.** **4. El actor pulsa el botón de "Enviar factura".** **5. El sistema crea el documento de la factura con los datos contenidos en las tablas de clientes y facturas.** **6. El sistema firma el documento lo comprime y lo envía por correo electrónico.** **6. El sistema actualiza el estado de la factura a enviada en la base de datos.** **7. Se informa al actor del resultado de la.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.8. CU08-Crear informe IRPF.

Identificador	**CU08**
Nombre	**Crear informe IRPF**
Autor	**Francisco Javier Cañabate**
Resumen	Tras la pulsación por parte del actor del botón “Informe IRPF”, el sistema presentará una lista con la información de importe bruto, IVA correspondiente e importe sin IVA en dos listas. En una se verán las cantidades de forma trimestral, y otra en la que se verán las cantidades anuales.
Actor	**Usuario**
Precondiciones	**Los datos solicitados existen en la base de datos.**
Postcondiciones	**Se ha generado en informe solicitado.**
Flujo normal	**1. El actor pulsa el botón de “Informe IRPF”.** **2. El sistema presenta un gráfico con el contenido de la consulta.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

3.2.9. CU09-Crear informe gráfico.

Identificador	**CU09**
Nombre	**Crear informe gráfico**
Autor	**Francisco Javier Cañabate**
Resumen	El actor selecciona un cliente en la tabla de clientes. Tras la pulsación por parte del actor del botón "Informe gráfico", el sistema presentará un gráfico de barras con los importes de las facturas mensuales del cliente seleccionado en la tabla de clientes. La información se presentará del año en curso y de los dos años anteriores.
Actor	**Usuario**
Precondiciones	**Los datos solicitados existen en la base de datos.**
Postcondiciones	**Se ha generado en informe solicitado.**
Flujo normal	**1. El actor selecciona el cliente en la tabla de clientes.** **2. El actor pulsa el botón de "Informe gráfico".** **3. El sistema accede a la información necesaria en la base de datos y presenta el informe al.**
Flujos alternativos	**Ninguno.**
Inclusiones	**Ninguna.**
Extensiones	**Ninguna.**

4. Diseño.

El diseño se ha realizado siguiendo el patrón MVC o modelo-vista-controlador. Según este patrón, el diseño de una aplicación informática se debe realizar creando una separación del código según su función principal. Por un lado tenemos el modelo que hace referencia a la persistencia de los datos y su gestión. Por otro lado tenemos la vista que se encarga de la interfaz de usuario. Por último tenemos el controlador que es el encargado de dar respuesta a los eventos que ocurran.

En nuestro caso, el control de eventos se encuentra en el panel de acciones cuya clase es ActionsPaneControl.cs. Esta clase contiene los controladores de los botones del panel de acciones que desencadenan la ejecución de las funcionalidades de la aplicación. La pulsación de uno de sus botones desencadena una serie de llamadas a los servicios proporcionados por las clases auxiliares que son las que proporcionan las funcionalidades concretas. Estas clases son:

CrearPdf.cs. Crea la factura en formato Pdf.

CrearWord.cs. Crea la factura en formato Doc.

Compresion.cs. Comprime en formato Zip el documento de la factura.

Correo.cs. Envía la factura comprimida por correo electrónico usando Outlook.

CrearGrafico.cs. Crea el informe gráfico.

El modelo es gestionado de forma automática por el sistema con alguna pequeña excepción. Estas excepciones están contenidas en la clase Datos.cs que se encarga de actualizar el estado de las facturas tras su envío por correo, de salvar los cambios realizados en las tablas de datos y de recuperar los datos necesarios para la realización del informe gráfico.

Por último queda la vista que es gestionada por el sistema de forma autónoma.

4.1. Decisiones tecnológicas.

4.1.1. Lenguaje de desarrollo.

EL lenguaje de desarrollo utilizado ha sido c#. La razón principal de esta elección es su gran similitud con el lenguaje Java que es el que más he usado hasta la fecha. La elección de otro de los posibles lenguajes habría supuesto un esfuerzo mayor en su aprendizaje y por lo tanto habría consumido un tiempo del que no disponía, debido a lo acotado del plazo de las entregas.

4.1.2. Base de datos.

Para la persistencia de los datos se ha utilizado Access. Las necesidades de almacenamiento y gestión de información requeridas por este proyecto son cubiertas por esta aplicación de forma satisfactoria. Además, no requiere de una instalación compleja, de hecho, se instala incluida en Office, por lo que una vez instalada la suite, ya está disponible para su uso.

4.1.3. Acceso a datos.

El acceso a datos se ha realizado mediante dos objetos DataGridView insertados en la hoja 1 del libro Excel que conforma la aplicación. La ventaja de estos objetos es que automatizan las operaciones de lectura y escritura, facilitando enormemente la gestión de los datos. Como aspecto negativo presentan limitaciones en lo que a flexibilidad se refiere, aunque es solventable mediante programación. Un DataGridView contiene la tabla de Clientes y el otro la tabla de facturas. Se han configurado en forma principal-detalle, con lo que basta seleccionar una entrada en la tabla principal, que en nuestro caso es Clientes, para obtener los registros relacionados con dicha entrada en la segunda tabla.

4.1.4. Diagrama estático de diseño.

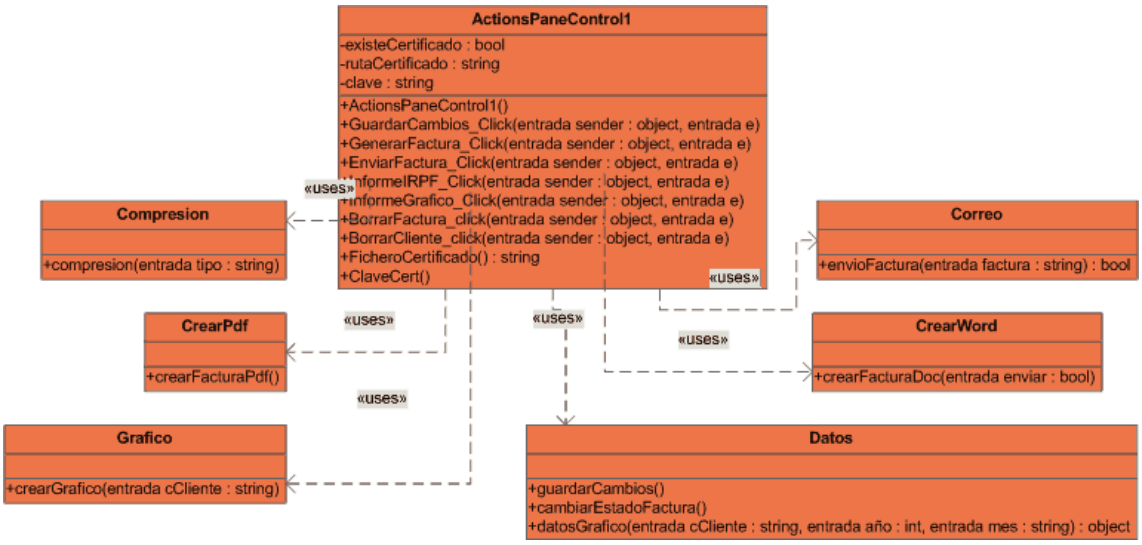


Ilustración 5. Diagrama estático de diseño.

En el anterior gráfico se puede observar como toda la funcionalidad ofrecida, gira en torno a la clase ActionsPaneControl1. El resto de clases proporcionan métodos estáticos que suministran las utilidades concretas.

4.1.5. Diseño de la persistencia.

A continuación se muestra el diseño conceptual empleando el modelo entidad-relación. Como se puede observar, el diseño es bastante sencillo. Consta únicamente de dos entidades, CLIENTE y FACTURA, con conectividad uno a muchos (1: N), es decir que un cliente puede tener varias facturas pero una factura solo puede tener un cliente. Además se comprueba que en la relación, el cliente es obligatorio pero la factura no. El sentido de esto último es que se puede registrar un cliente antes de realizarle la factura. Esto tiene sentido, sin embargo no tiene sentido registrar una factura sin que exista un cliente que responda ante ella.

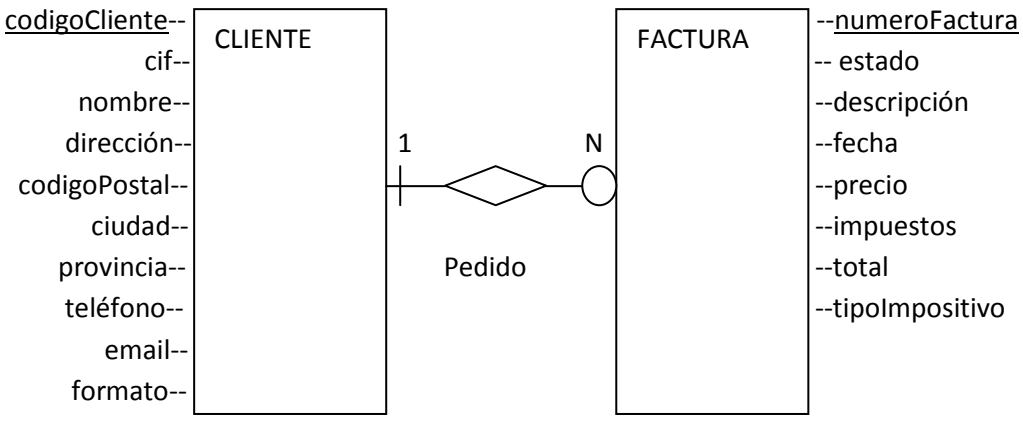


Ilustración 6. Modelo Entidad-Relación.

4.1.6. Diseño lógico.

En el diseño lógico, partimos del diseño conceptual y del modelo entidad-relación, para confeccionar el modelo relacional que es en el que nos basaremos puesto que es el utilizado por la base de datos Access, elegida para desarrollar este proyecto.

El sentido de esta etapa es que se ha resuelto en la etapa anterior el problema de la estructuración de los datos en el modelo conceptual y se procederá a adaptarlo concentrándose en las cuestiones tecnológicas relacionadas con el modelo de datos a utilizar que, como ya se comento, es el modelo relacional.

4.1.6.1. *Entidades.*

En la entidad FACTURA se ha añadido el atributo codigoCliente. Esto es consecuencia de aplicar la transformación del modelo conceptual al lógico cuando nos encontramos con una conectividad 1: N. Este nuevo atributo, establece la relación lógica entre ambas entidades.

CLIENTE (codigoCliente, cif, nombre, dirección, codigoPostal, ciudad, provincia, teléfono, email, formato)

FACTURA (numeroFactura, codigoCliente, descripción, fecha, precio, tipoImpositivo, impuestos, total, estado)

Donde {codigoCliente} referencia a CLIENTE

Además de estas dos entidades, se han derivado de ellas tres vistas que combinan la información de forma adecuada a las necesidades de información del proyecto.

Vista **Facturas consulta** (codigoCliente, Fecha por mes, Fecha por año, Suma de total)

Vista **IRPF anual** (Año, Precio, Impuestos, Total)

Vista **IRPF trimestral** (Año, trimestre, Precio, Impuestos, Total)

4.1.7. Tablas.

4.1.7.1. Tabla cliente.

La tabla cliente contiene los datos necesarios para la identificación del cliente y para la gestión y envío de la factura electrónica.

Tabla CLIENTE				
Nombre columna	**Tipo de dato**	**Clave**	**Nulo permitido**	**Descripción**
codigoCliente	int	Clave principal	No	Código único adjudicado al cliente
cif	Varchar(9)	Clave alternativa	No	Código de identificación fiscal del cliente
nombre	Varchar(50)	No	No	Nombre del cliente
dirección	Varchar(50)	No	No	Dirección del cliente
codigoPostal	Varchar(5)	No	No	Código postal del cliente
ciudad	Varchar(20)	No	No	Ciudad de residencia del cliente
provincia	Varchar(20)	No	No	Provincia de residencia del cliente
teléfono	Varchar(9)	No	No	Teléfono del cliente
email	Varchar(50)	No	No	Dirección de correo electrónico del cliente
formato	Varchar(3)	no	no	Formato preferido para el envío de las facturas

4.1.7.2. Tabla FACTURA.

La tabla factura contiene los datos necesarios para emitir la factura así como el estado de esta (enviada, cobrada…).

Tabla FACTURA				
Nombre columna	**Tipo de dato**	**Clave**	**Nulo permitido**	**Descripción**
numeroFactura	int	Clave principal	No	Código único adjudicado al cliente
codigoCliente	int	Clave foránea a CLIENTE	No	Código de identificación fiscal del cliente
descripción	Varchar(200)	No	No	Describe el trabajo motivo de la factura
fecha	datetime	No	No	Fecha de emisión de la factura
precio	int	No	No	Precio del trabajo realizado
impuestos	int	No	No	Impuestos aplicados
total	int	No	No	Precio final
estado	Varchar(20)	No	No	Estado de la factura.
tipoImpositivo	Int	No	No	Tipo de IVA aplicado

4.1.7.3. Vista Facturas consulta.

Esta vista es utilizada en la confección del informe gráfico. Proporciona los ingresos totales de un cliente por año y mes.

Vista Facturas consulta				
Nombre columna	**Tipo de dato**	**Clave**	**Nulo permitido**	**Descripción**
codigoCliente	int	Clave principal	No	Código de identificación fiscal del cliente
Fecha por mes	datetime	No	No	Facturas correspondientes a un mes
Fecha por año	datetime	No	No	Facturas correspondientes a un año
Suma de total	int	No	No	Suma de Precio final de un mes de un año.

4.1.7.4. Vista IRPF anual.

Esta vista es utilizada en la confección del informe IRPF. Proporciona los datos de ingresos netos, IVA pagado e ingresos brutos de un año.

Vista IRPF anual				
Nombre columna	**Tipo de dato**	**Clave**	**Nulo permitido**	**Descripción**
Año	datetime	Clave principal	No	Determina el año del informe
Precio	int	No	No	Precio cobrado en el año
Impuestos	int	No	No	IVA pagado en el año
Total	int	No	No	Precio final cobrado en el año

4.1.7.5. Vista IRPF trimestral.

Esta vista es utilizada en la confección del informe IRPF. Proporciona los datos de ingresos netos, IVA pagado e ingresos brutos de un trimestre.

Vista **IRPF trimestral** (Año, trimestre, Precio, Impuestos, Total)

Vista IRPF trimestral				
Nombre columna	**Tipo de dato**	**Clave**	**Nulo permitido**	**Descripción**
Año	datetime	Clave principal	No	Determina el año del informe
Trimestre	datetime	Clave principal	No	Determina el trimestre del informe
Precio	int	No	No	Precio cobrado en el año
Impuestos	int	No	No	IVA pagado en el año
Total	int	No	No	Precio final cobrado en el año

4.1.8. Prototipo de la interfaz de usuario.

A continuación se incluyen unas imágenes del prototipo realizado en la etapa de diseño a modo ilustrativo.

4.1.8.1. Pantalla principal.

La pantalla principal es una hoja de Excel en la que se observa en la parte superior, una serie de botones que dan acceso a las distintas funcionalidades del sistema. Más abajo se observa una tabla con los datos del cliente y la factura.

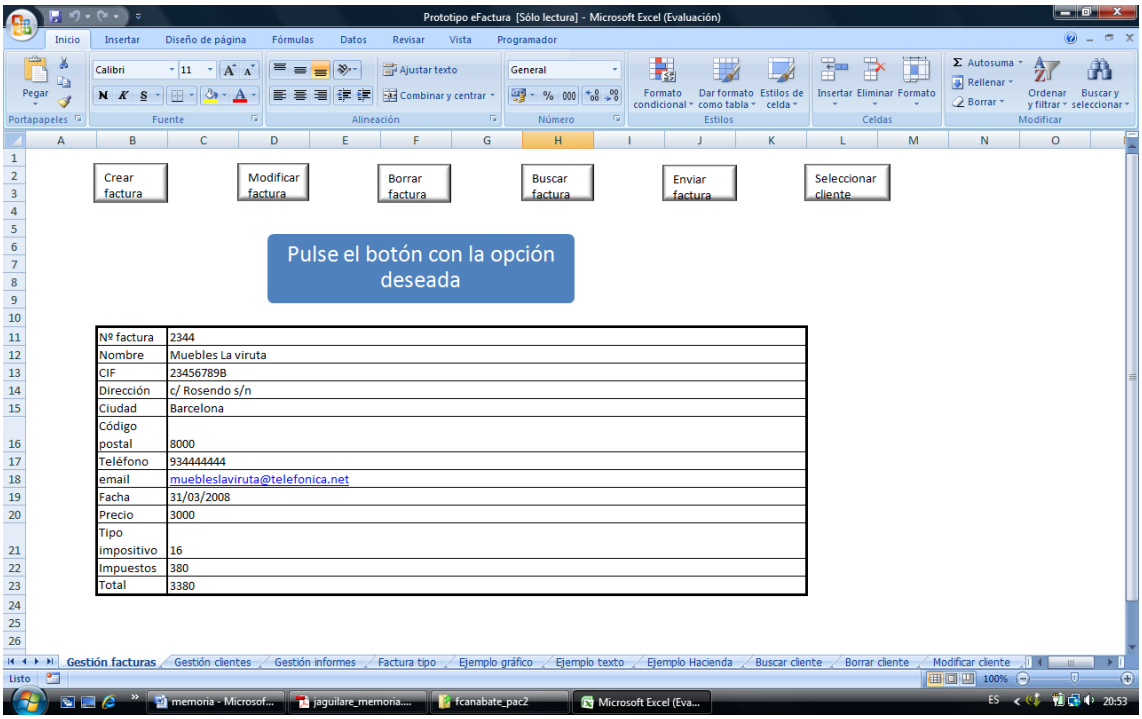


Ilustración 7.Prototipo. Pantalla principal.

4.1.8.2. Pantalla crear factura.

Al pulsar el botón Crear factura de la pantalla principal, se obtiene acceso a la siguiente pantalla, en la que se pueden introducir los datos de una nueva factura y confirmar o cancelar la operación.

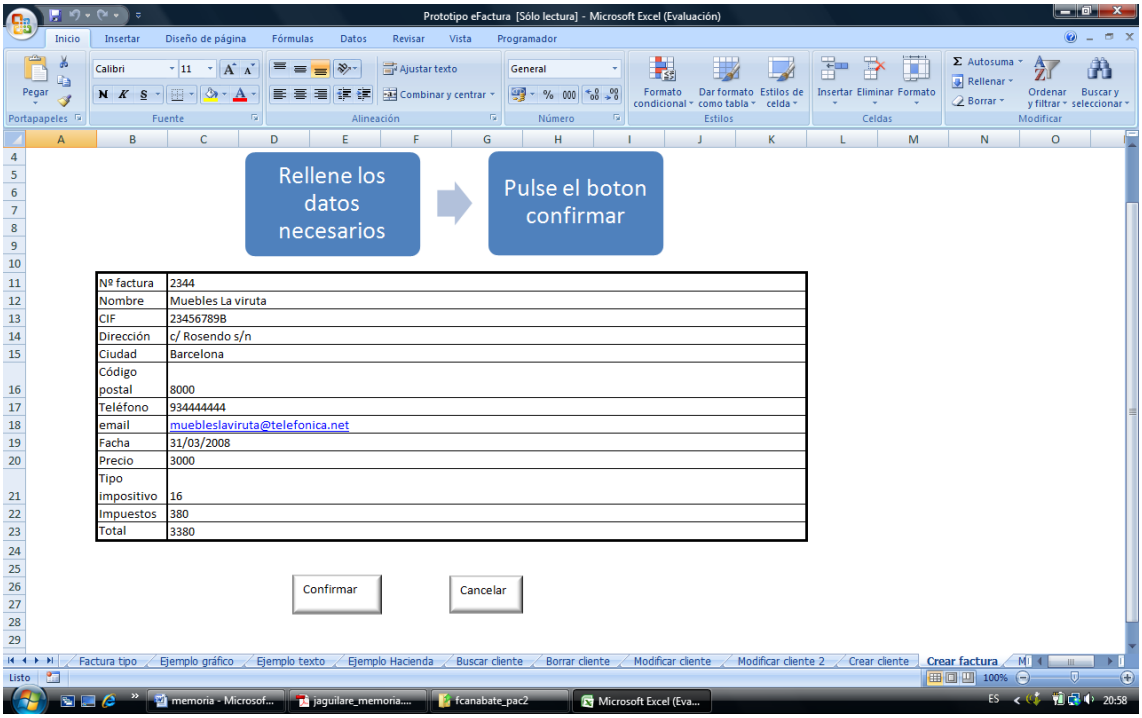


Ilustración 8. Prototipo. Pantalla Crear factura.

4.1.8.3. Pantalla informe grafico.

En la siguiente pantalla se obtiene un gráfico de los ingresos obtenidos de un cliente.

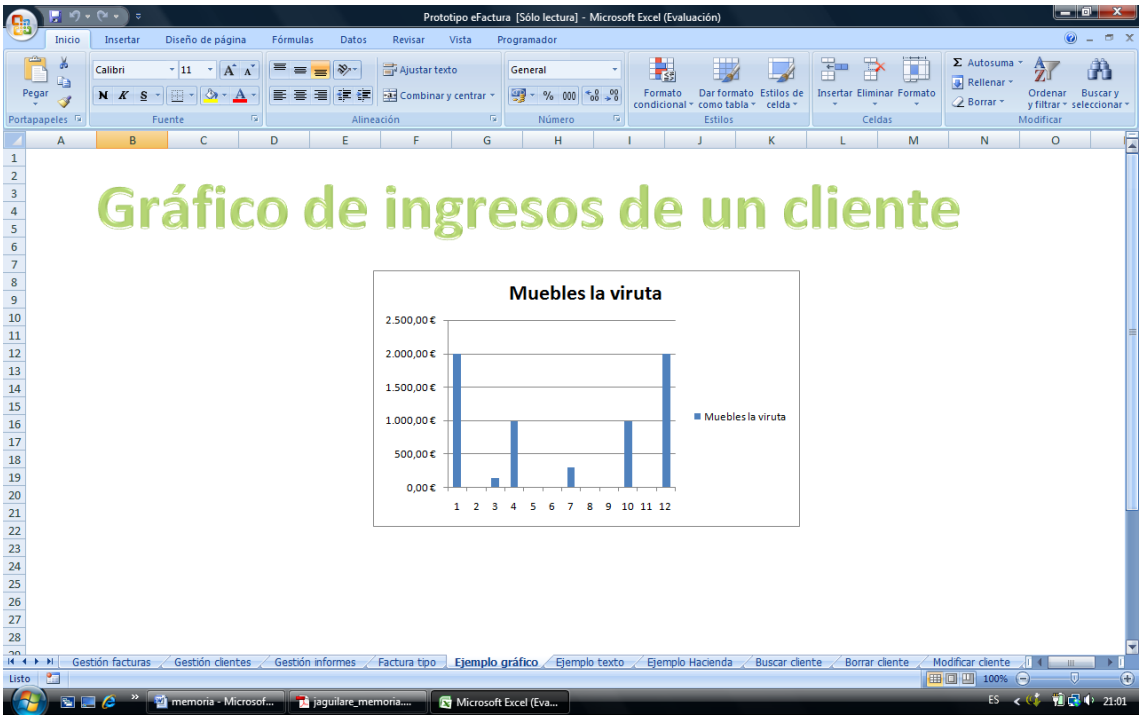


Ilustración 9. Prototipo. Pantalla informe gráfico.

5. Implementación.

En este apartado se expondrán las principales características del código utilizado en la implementación.

5.1. Explicación del modelo de programación usado.

La implementación de este proyecto, como se explico en apartados anteriores, se ha realizado utilizando Visual Studio Tools for Office. La aplicación base de la implementación es un libro de Excel para el que se ha desarrollado una personalización a nivel de documento. Dicho esto, se realizará un breve repaso sobre la estructura que conforma esta personalización.

En el nivel básico, una personalización a nivel de documento de un libro Excel consta de una clase denominada ThisWorkBook en la que se incluye el código general para todo el libro, y una clase por cada hoja del libro que se denomina Hojax donde x corresponde con el mismo número de la hoja del libro. En cada una de las clases de hoja, se incluye el código necesario para la funcionalidad particular de esta hoja. Dentro de cada una de estas clases, tanto la de ThisWorkBook como las de las distintas hojas, se encuentran varios métodos por defecto. El método Startup se ejecuta una sola vez al iniciar el documento. En él se incluyen la inicialización de los delegados correspondientes a los eventos atendidos por la clase y cualquier otra inicialización necesaria. Por su parte, el método Shutdown se ejecuta al cerrar el libro y se utiliza para liberar recursos.

Un detalle importante sobre el diseño empleado es que la totalidad de los métodos utilizados en las clases añadidas, es decir todas menos las clases ThisWorkBook, y las relativas a las hojas, se han diseñado como estáticos. El motivo es que de esta manera se pueden utilizar sin necesidad de instanciar un objeto de la clase. Esto es posible ya que los métodos representan servicios que se ofrecen a las clases principales y por lo tanto solo se necesita que se ejecuten una vez con cada llamada, después su utilidad desaparece.

5.1.1. Clase ThisWorkBook .

Aparte de todo esto, se pueden añadir más clases siempre que sean necesarias.

En este proyecto, se ha utilizado el método Startup de la clase ThisWorkBook para inicializar un elemento denominado ActionsPaneControl. Al estar definido en esta clase que como se dijo, es de aplicación para todo el libro, estará disponible en todas las hojas del libro Excel. Este elemento actúa como contenedor de otros controles. En este caso, contiene la mayoría de los botones, en concreto, todos los que dan acceso a las distintas funcionalidades de la aplicación.

El código incluido en el método Startup para inicializar el ActionsPaneControl es el siguiente:

```
ActionsPaneControl1 actions = new ActionsPaneControl1();
            this.ActionsPane.Controls.Add(actions);
```

También se utiliza para ocultar las dos hojas auxiliares de que consta el libro.

La decisión de usar este control viene dada por su utilidad para implementar el patrón modelo-vista-controlador. En concreto, y debido a que al crear este control, automáticamente se añade una clase en la que se definen los controles contenidos en el ActionsPaneControl, de forma directa, se implementa el control de eventos, con lo que prácticamente tenemos el patrón implementado, ya que como se explicará más adelante el modelo también se define al utilizar otro elemento, el DataGridView, y la vista, prácticamente es controlada por el sistema. También ofrece un aspecto más estético que la inclusión de los botones en la misma hoja. De peso fue también la característica de tener disponibles los controles en todas las hojas definiéndolos en un solo sitio.

5.1.2. Clase Hoja1.

Como se anticipaba en el párrafo anterior, otra de las decisiones de diseño importantes ha sido la utilización del elemento DataGridView. Este elemento se asocia con una tabla y con solo incluir unas pocas líneas de código, se consigue que aparezca en la posición que se le indique, una tabla enlazada con el contenido que se desee de la relación de la base de datos elegida. Esta tabla se rellena automáticamente al iniciar la hoja, apareciendo todos los datos requeridos.

El motivo principal, es que solo con incluir este elemento y con muy poca configuración, obtenemos una funcionalidad muy importante, que de no usarlo requeriría de bastante código, para acceder directamente a la base de datos, ordenar los datos, modificarlos, añadir nuevos datos… También permite especificar un campo para que se rellene automáticamente al introducir una nueva línea e incluso que se autoincremente, con lo que obtenemos un posible índice para la tabla.

Además, permite definir la zona de pantalla que debe ocupar, apareciendo unas barras de desplazamiento caso de ser necesarias lo que permite una buena organización del espacio de trabajo, permitiendo eliminar la necesidad de realizar desplazamientos de la página, contribuyendo a la usabilidad de la aplicación.

Por todo esto se obtó por incluir este elemento. Concretamente se incluyen dos DataGridView, uno para la tabla Clientes y otro para la tabla Facturas. Al definir el enlace a datos se establece una configuración principal-detalle entre las tablas Clientes-Facturas respectivamente, con lo que se consigue que al seleccionar un elemento de la tabla principal, o tabla Clientes, la tabla detalle se actualice con las facturas del cliente seleccionado. Todo esto sin escribir nada de código.

En cuanto a la Hoja1 que conforma la pantalla principal de la aplicación, el único código que se ha introducido en el método Startup ha sido el siguiente,

```
if (this.NeedsFill("proyectoDataSet"))
            {

this.facturasTableAdapter.Fill(this.proyectoDataSet.Facturas);
            }
```

Con estas líneas conseguimos que automáticamente se rellene en pantalla, la tabla Facturas. Otra estructura como la anterior conseguirá lo mismo para la tabla Clientes.

5.1.3. Clase Hoja2.

En la Hoja2, que se encarga de mostrar el informe para recuperar los datos referentes a ingresos e IVA, se incluyen unas líneas similares a las anteriores, que en este caso, atacan a dos de las vistas creadas en la base de datos.

La decisión de utilizar vistas en lugar de recuperar y transformar los datos por programación, se basa en el hecho de que esta es una de las funciones principales de Access y por lo tanto, su código debe ser muy eficiente y robusto en estas operaciones, además, una de las premisas de este proyecto era la de usar Office y por lo tanto parecía adecuado aprovechar sus posibilidades.

5.1.4. Clase ActionsPaneControl1.

La clase ActionsPaneControl1, realiza la atención a eventos mediante el control de los botones incluidos en su diseñador. La atención de los eventos se realiza en dos partes, con el siguiente código incluido en el constructor de la clase, se escucha la pulsación del botón GuardarCambios,

```
this.GuardarCambios.Click += new
System.EventHandler(this.GuardarCambios_Click);
```

Y una vez producido el evento de la pulsación del botón, se ejecuta el siguiente método,

```
private void GuardarCambios_Click(object sender, EventArgs e)
        {
            //salvar los cambios realizados a la base de datos
            Datos.GuardarCambios();
        }
```

Este método inicia las acciones esperadas al producirse el evento. El resto de botones se atienden de forma similar.

Otra decisión de diseño fue la de crear los documentos de las facturas en lugar de rellenar un formulario ya creado. El motivo de esta elección es por un lado que de esta forma se evitan ficheros externos que pueden dar problemas de diversa índole como que se muevan de sitio, se corrompan, etc... También se valoró el hecho de que de esta forma se reducía la configuración del sistema y por último, se utilizaba más a fondo la herramienta Visual Studio Tools for Office, utilizándola también en documentos Word.

Para crear el documento en formato Pdf se utilizó la librería ItextSharp de libre uso, pues permitía la creación del documento de forma relativamente sencilla así como realizar la firma del mismo, con lo que cumplía con los requisitos del proyecto.

5.1.5. Clase Datos.

La clase Datos se crea para incluir en ella las manipulaciones de la base de datos, como guardar los cambios introducidos en las tablas. Por lo tanto, constituye el apartado Modelo junto con la base de datos, del patrón Modelo-Vista-Controlador o MVC.

También se incluye un acceso de solo lectura, "DataReader", para obtener los datos necesarios para la realización del informe gráfico. El fin de esta inclusión, es el de realizar un acceso manual completo a la base de datos, creando la cadena de conexión,

```
"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=|DataDirectory|\proyecto.accdb"
```

activando la conexión,

```
OleDbConnection conexion = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=|DataDirectory|\proyecto.accdb");
```

realizando la orden SQL,

```
String orden = "SELECT [Suma De Total] FROM [Facturas
Consulta] WHERE [Código cliente] = " + cCliente + " AND
[Fecha Por año] = " + año + " AND [Fecha Por mes] ='" + mes +
"'";
```

y lanzándola contra la base de datos,

```
OleDbCommand ordenSql = new
OleDbCommand(orden, conexion);
```

así como realizar la lectura del resultado obtenido.

```
conexion.Open();
OleDbDataReader mydt =
ordenSql.ExecuteReader();
while (mydt.Read())
aux = mydt.GetValue(0);
mydt.Close();
conexion.Close();
```

En otras palabras, si bien la automatización de los anteriores accesos ahorró tiempo y esfuerzo, se quería demostrar la posibilidad de realizar estos accesos de forma manual.

5.1.6. Clase Grafico.

La clase Grafico, se encarga de recopilar los datos necesarios para la construcción del gráfico, utilizando los servicios de la clase Datos. Se piden uno a uno a medida que se va rellenando el rango de celdas con el que se formará el gráfico.

```
for (int columna = 2; columna < 5; columna++)
            {
                Globals.Hoja3.Cells[51, columna] = año;
                for (int fila = 52; fila < 64; fila++)
                {
                    Globals.Hoja3.Cells[fila, 1] = mesAux.ToString();

                    Globals.Hoja3.Cells[fila, columna] =
Datos.DatosGrafico(cCliente, año, mesAux.ToString());

                    mesAux += 1;
                }
                mesAux = mes.enero;
                año += 1;
            }
```

Como se puede observar tenemos dos construcciones for anidados que se encargan de rellenar las celdas con los datos apropiados. El bucle for exterior va recorriendo los años, mientras que el bucle for interior recorre los meses.

Se ha utilizado un enumerado para facilitar el recorrido por los meses.

```
enum mes { enero, febrero, marzo, abril, mayo, junio, julio,
agosto, septiembre, octubre, noviembre, diciembre };
```

Los enumerados permiten utilizar textos en el código mientras que internamente, el sistema utiliza números, con lo que es muy sencillo realizar recorridos, comparaciones y ordenarlos.

Por último en esta clase se crea el gráfico.

```
Excel.ChartObjects ChartObjects1 =
(Excel.ChartObjects)Globals.Hoja3.ChartObjects(missing);
Excel.ChartObject chartObject1 = ChartObjects1.Add(100, 20, 400,
300);
chartObject1.Chart.ChartWizard(objRange,
Excel.XlChartType.xl3DColumn, missing, missing, missing,
missing, missing, "Ingreso anual del cliente seleccionado",
missing, missing, missing);
chartObject1.Name = "myChartObject";
chartObject1.Activate();
```

5.1.7. Clase CrearWord.

La clase CrearWord abre la aplicación Word, crea un nuevo documento, y lo rellena con los datos de la factura y cliente seleccionado. Tiene un parámetro de entrada de tipo booleano que sirve para diferenciar cuando se desea visualizar el documento o se desea enviarlo por correo. El motivo de esta diferenciación es que se desea enviar el correo con la aplicación abierta se produce un fallo. Se ha solucionado mediante este parámetro diferenciador que en caso de tener valor cierto, cierra la aplicación antes de retornar del envío del fichero y en caso de que se desee ver la factura, valdrá falso, no se cierra la aplicación, permitiendo que el documento se visualice con total tranquilidad.

```
if (enviar)
{
object doNotSaveChanges =
Word.WdSaveOptions.wdDoNotSaveChanges;
document.Close(ref doNotSaveChanges, ref missing, ref
missing);
application.Quit(ref doNotSaveChanges, ref missing, ref
missing);
}
```

También se encarga de iniciar el proceso de firmado digital con certificado de la siguiente forma,

```
document.Signatures.Add();
```

Esta sentencia provoca que se acceda al almacén de certificados digitales de Internet Explorer, se pueda elegir un certificado y se realice la firma digital del documento.

5.1.8. Clase CrearPdf.

La clase CrearPdf es muy similar a la clase CrearWord, con la salvedad de que no utiliza Visual Studio Tools for Office, sino la librería ItextSharp para crear y firmar el documento de la factura en formato Pdf. Una diferencia importante es que en lugar de llamar al almacén de certificados de Internet Explorer, esta clase precisa de un fichero con el certificado y la clave privada para firmar digitalmente el documento. Este ha sido el único camino encontrado para realizar la firma digital utilizando esta biblioteca.

Desaparece también el parámetro de entrada ya que no existe la posibilidad de abrir el documento Pdf desde la aplicación.

5.1.9. Clase Correo.

La clase Correo utiliza el programa Outlook para el envío de los mensajes de correo. Aunque es posible y sencillo utilizar C# para enviar el mensaje utilizando librerías propias de .NET, el

hecho de utilizar Outlook hace que el uso de Office sea más intenso, y como se ha comentado anteriormente, esta era una de las premisas del proyecto.

Surgió un problema al enviar los ficheros por correo. En concreto, se corrompía la firma de los documentos, con el resultado de que el fichero Word se abría pero sin la firma y el documento Pdf no se podía abrir. Este problema surgía al enviar los ficheros con gestores de correo Smtp. Usando gestores Web como el utilizado por la Uoc, los ficheros se recibían con la firma intacta. Esto fallo se comprobó desde tres ordenadores diferentes, con Windows Vista y con Windows Xp, con ficheros creados desde la aplicación y firmados directamente desde Word, incluso con un Pdf firmado desde el programa de Adobe. Siempre se producía el mismo error. La solución a este problema fue el enviar los documentos comprimidos en formato ZIP, tras lo cual, tras recibir por correo el documento y descomprimirlo, se observaba la firma correctamente.

5.1.10. Clase Compresion.

Para realizar la compresión en formato Zip de los documentos se ha utilizado la librería `Ionic.Utils.Zip`. Esta librería no se incluye con las librerías de .Net. Se ha utilizado esta librería por su facilidad de uso y la simplicidad del código necesario como se puede ver a continuación. Las librerías incluidas en .Net no son de aplicación directa sobre un fichero y por lo tanto la opción utilizada era la más cómoda y sencilla.

```
ZipFile zip = new ZipFile(@"c:\tmp\factura.zip");
                {
                    if (tipo.Equals("PDF"))
                    {
                        zip.AddFile("factura.pdf");
                    }

                    else if (tipo.Equals("DOC"))
                    {
                        zip.AddFile("factura.doc");
                    }
                    zip.Save();
                }
```

5.2. Capturas de pantalla.

A continuación se incluyen capturas de la aplicación.

5.2.1. Pantalla principal.

En la pantalla principal se observa, en la parte superior, el panel de acciones con los botones que proporcionan acceso a las distintas funcionalidades de la aplicación. A continuación se puede observar la tabla de clientes y la tabla de facturas.

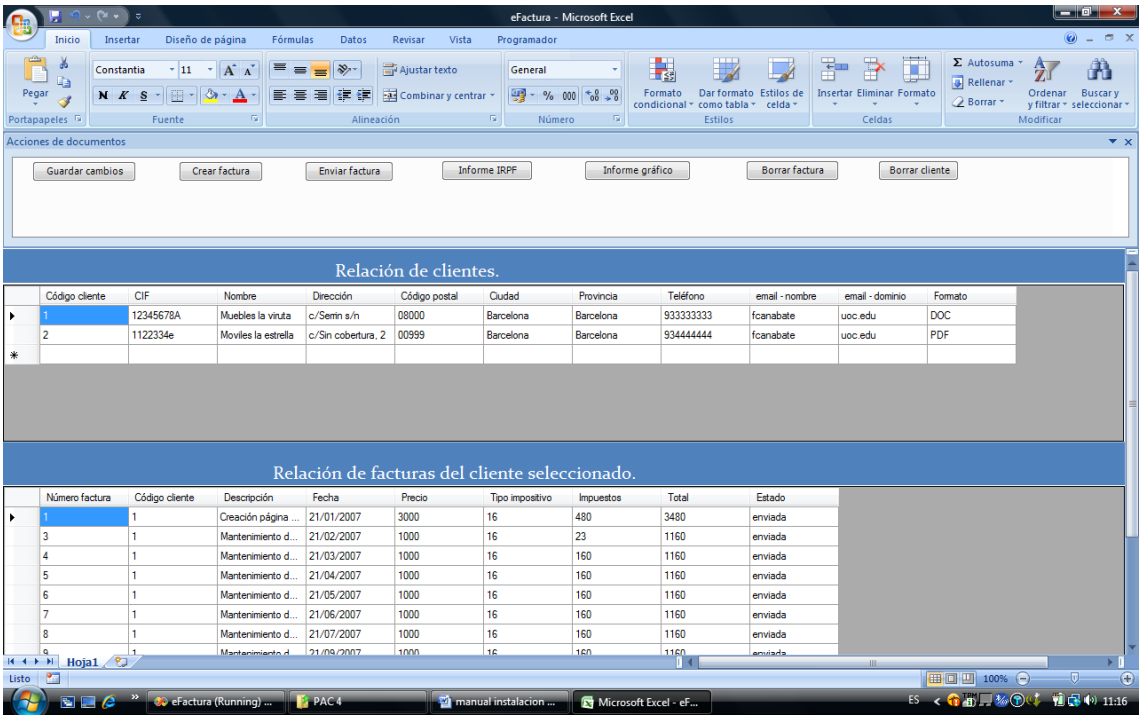


Ilustración 10. Implementación. Pantalla principal.

5.2.2. Imagen de la factura en formato doc.

Este es el resultado de pulsar el botón “Crear factura” cuando el formato solicitado es doc.

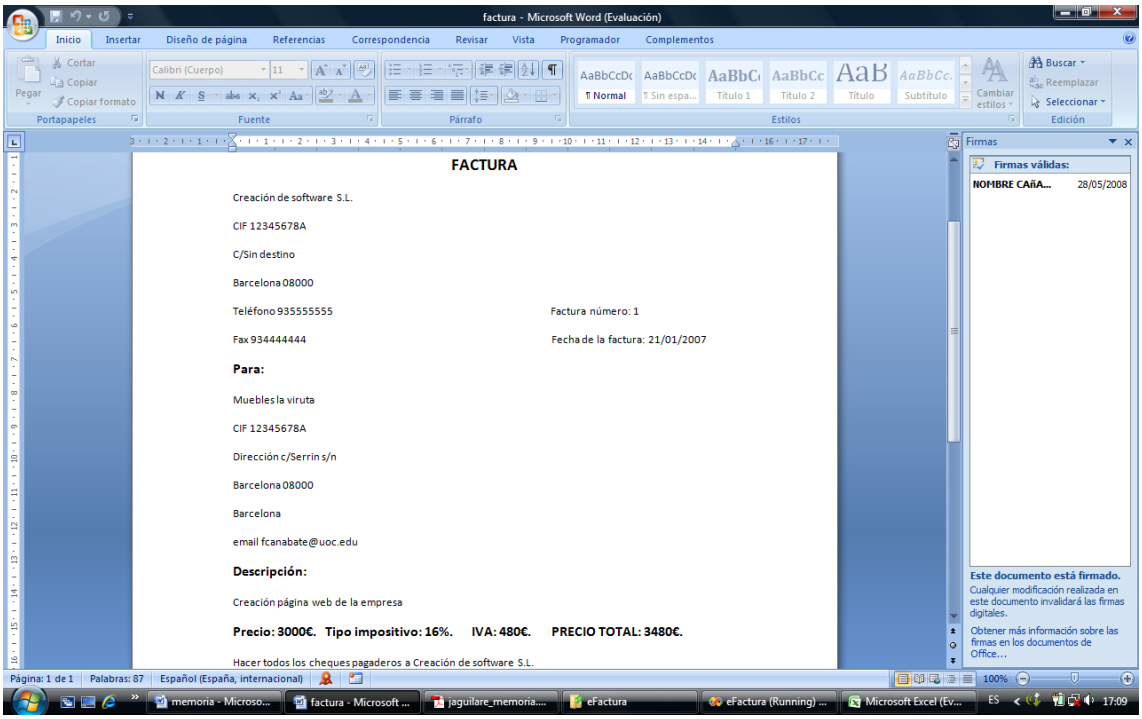


Ilustración 11. Implementación. Pantalla crear factura.

5.2.3. Imagen de un informe IRPF.

En esta imagen se observa a la izquierda el resumen anual, y a la derecha el resumen trimestral.

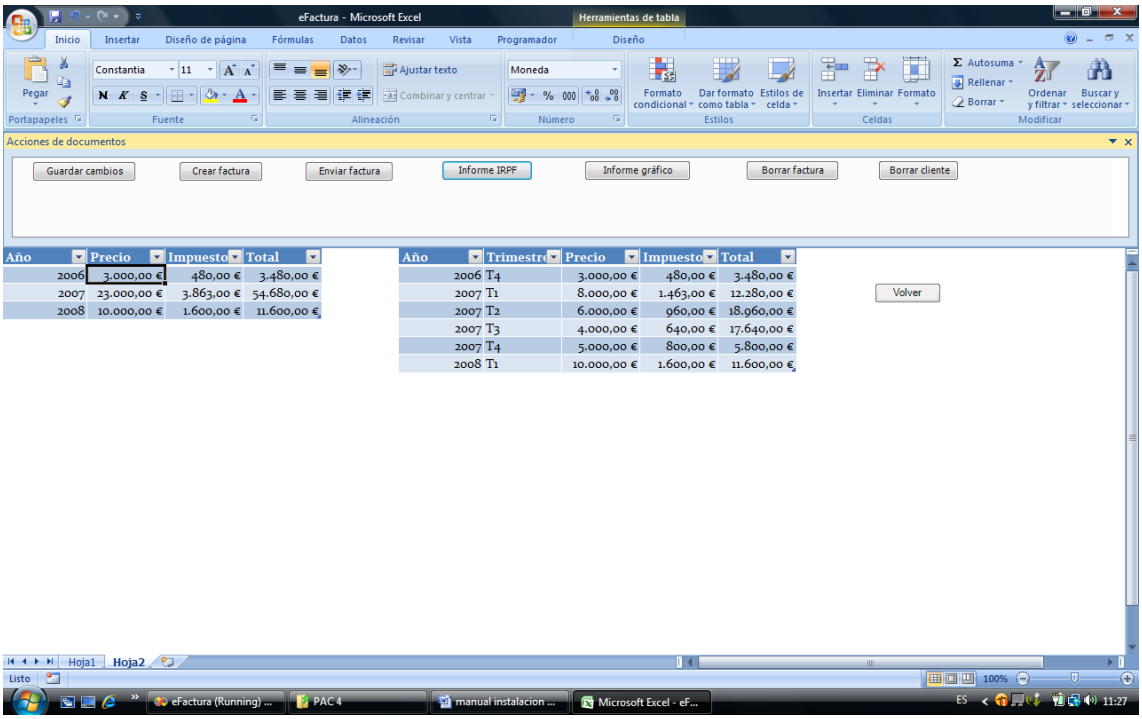


Ilustración 12. Implementación. Pantalla informe IRPF.

5.2.4. Imagen de un informe gráfico.

Esta imagen muestra el resultado de pulsar el botón de “Informe gráfico”.

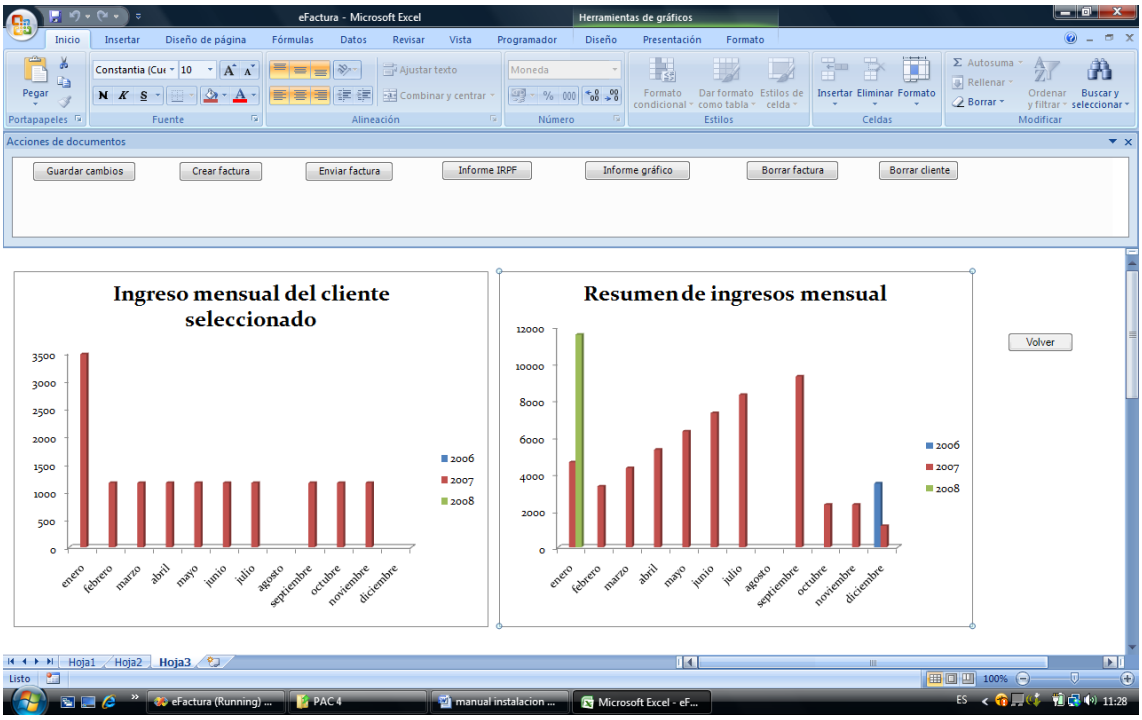


Ilustración 13. Implementación. Pantalla informe gráfico.

6. Conclusiones.

Durante el desarrollo del proyecto, han surgido multitud de problemas, que en un principio parecían insalvables. La constancia y la ilusión han permitido ir solventándolos poco a poco junto con la aplicación de los conocimientos adquiridos durante el desarrollo de los estudios de Enginyeria tècnica en Informàtica de sistemes, quizás sobre todo los de Enginyeria del Software, se han mostrado decisivos en la consecución de los objetivos marcados al inicio del proyecto. Un problema que parecía irresoluble en el primer contacto, se ha ido volviendo un poco más asequible a medida que se iban completando las distintas etapas del proceso demostrando que la planificación y el estudio realizado de forma previa a la programación sí que son importantes.

La utilización de Visual Studio Tools for Office para adjuntar código a las aplicaciones de Office ofrece la ventaja de que el usuario no cambia de entorno, puede continuar con sus herramientas ofimáticas habituales, minimizando el aprendizaje, puesto que se supone que ya domina el entorno de Office y mejorando la usabilidad del producto.

En cuanto a la experiencia, ha sido realmente enriquecedora. Ha supuesto un primer contacto con muchas nuevas herramientas, como Visual Studio, que ha sido el primer entorno de desarrollo profesional con el que he trabajado, también ha sido la primera aplicación en un entorno gráfico de ventanas. El uso de Visual Studio Tools for Office ha sido muy interesante por lo prometedor de su futuro. Como negativo destacaría que debido a lo ajustado del tiempo

disponible, solo ha sido posible arañar la superficie de todo lo que pueden aportar estas tecnologías y herramientas. Sin duda la tecnología .Net ha sido un gran descubrimiento que me ha dejado con ganas de más.

7. Glosario.

En el desarrollo de este documento se intentado evitar la utilización de abreviaturas y términos complejos para facilitar su lectura y comprensión. No obstante, en un documento de las características del presente es inevitable la utilización de algunos términos técnicos. A continuación se ofrece una lista de los términos y abreviaturas utilizados junto con una breve descripción.

Término	Descripción
.Net	Conjunto de tecnologías para el desarrollo de aplicaciones de Microsoft.
ActionsPaneControl	Elemento de Visual Studio que actúa como contenedor de otros controles o etiquetas.
C#	Lenguaje de programación orientada a objetos de Microsoft.
Certificado digital	Certificado emitido por una central de acreditación que pretende garantizar que la identidad de la persona que lo utiliza es la que dice ser.
DataGridView	Elemento de Visual Studio que se utiliza para presentar datos de un origen de datos al que se enlaza.
DataReader	Elemento de Visual Studio que proporciona una conexión de solo lectura con un origen de datos.
FrameWork .NET	Ver .NET
MVC	Modelo-Vista-Controlador. Patrón de desarrollo de aplicaciones en el que se separa el Modelo de datos, la interfaz de usuario y el control de los eventos.
Pdf	Formato portable de documentos de Adobe.
Visual Studio	Entorno de programación de Microsoft
Visual Studio Tools for Office	Extensión de Visual Studio para añadir código a las aplicaciones de Office
Zip	Formato de compresión de archivos.

8. Bibliografía.

Ceballos, F. J. (2007). *Visual C# 2ª edicíon.* Paracuellos de Jarama, Madrid: Ra-Ma.

Codeplex. (s.f.). Recuperado el 2008, de http://www.codeplex.com/DotNetZip

ItextSharp. (s.f.). Recuperado el 2008, de http://itextsharp.sourceforge.net/

Microsoft Developer Network. (s.f.). Recuperado el 2008, de http://msdn.microsoft.com/es-es/default.aspx

Nabble - iTextSharp forum. (s.f.). Recuperado el 2008, de http://www.nabble.com/iTextSharp-f4188.html