ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO EN INFORMÁTICA

Título del proyecto:

ANÁLISIS DE TEXTOS EN PDF

Alumna: Leticia Osácar Landa

Tutor: Jesús Villadangos Alonso

Pamplona, Abril de 2013

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

1.1 Introducción

El objetivo principal de este proyecto es el de la implementación de una aplicación para ordenadores que realice un análisis de textos en archivos PDF y que parte de los resultados se vean reflejados en una interfaz para el fácil alcance y comprensión del usuario.

Lo que queremos llevar a cabo es la búsqueda una palabra especifica en un conjunto de archivos PDF y agregar una anotación por cada palabra y página en el archivo correspondiente cuando esta palabra aparezca en ellos. Podremos realizar esta búsqueda de manera individual o haciendo una búsqueda masiva y automática dentro de un directorio. Poblaremos una base de datos con los archivos que estudiemos para poder tener información directa de todas las palabras que existen en los archivos con la lectura de un archivo PDF.
Finalmente, mediante el uso de la interfaz, podremos ver los archivos PDF que deseemos, así como las anotaciones e información básica de los mismos.

¿Qué es un archivo PDF?
PDF (Formato de documento portátil) es un formato de almacenamiento de documentos digitales independiente de plataformas de software o hardware. Este formato es de tipo compuesto, lo cual quiere decir que lo componen los siguientes elementos: imagen vectorial, mapa de bits y texto.
Las características principales de este formato son las siguientes:

- Es multiplataforma. Es decir, puede ser utilizado en los principales sistemas operativos (Windows, Unix/Linux o Mac) sin que los archivos sufran modificación alguna. También está presente para dispositivos móviles como Android e iOS.
- Pueden estar compuestos por cualquier elemento de texto, multimedia, hipertextos (vínculos) etc...
- Es uno de los formatos más utilizados a la hora de realizar intercambios de documentos en Internet.

- Puede cifrarse para proteger su contenido e incluso se pueden firmar digitalmente.
- Nos ofrece la oportunidad de poder borrar permanentemente información confidencial de los PDF censurando texto visible e imágenes o eliminando la información oculta.
- Se permiten búsquedas cuando se quiere encontrar una palabra o expresión concreta en el documento PDF así como los metadatos propios de cada documento.

Todos los ficheros PDF comparten la misma estructura interna que está compuesta por cuatro partes:

- Cabecera: información sobre la especificación del estándar PDF que se ha utilizado, como por ejemplo, la versión.
- Cuerpo: descripción de los elementos utilizados en las páginas del fichero.
- Tabla de referencias cruzadas: información de los elementos usados en las páginas del fichero.
- Coda: indica donde encontrar la tabla de referencias cruzadas.[1]

Todo ello esta estandarizado por el estándar ISO 32000. De la misma manera, este estándar es también la base para los estándares PDF para fines específicos tales como: PDF/a para archivado, PDF/E para ingeniería, PDF/X o PDF/VT para impresión, PDF para sanidad y PDF/UA para accesibilidad.

Cuando cogemos cualquier documento y lo convertimos al formato PDF, lo que obtenemos es un nuevo archivo cuyo aspecto es el mismo que si se hubiese impreso en papel. Lo que ocurre es que, a diferencia de los documentos impresos, los archivos PDF pueden contener material multimedia adicional como botones, campos de formulario, videos etc… para que el usuario pueda interactuar con él. Para poder utilizar este tipo de archivos solamente es necesario descargarse la herramienta software gratuita Adobe Reader. Es interesante saber que se ha introducido la tecnología OCR para el reconocimiento de textos digitalizados en PDF, lo cual amplia y mejora las opciones de trabajo con los mismos. [2]

Uno de los elementos clave de los documentos PDF en este proyecto son las etiquetas/anotaciones. Se utilizan para añadir comentarios en los documentos acerca del

tema que queramos dentro del documento mismo. Existen muchos tipos de anotaciones: notas adhesivas, resaltar texto (high-lighting), utilización de líneas y formas, sellos, etc. pero nosotros en este proyecto hemos trabajado solamente con las notas adhesivas.

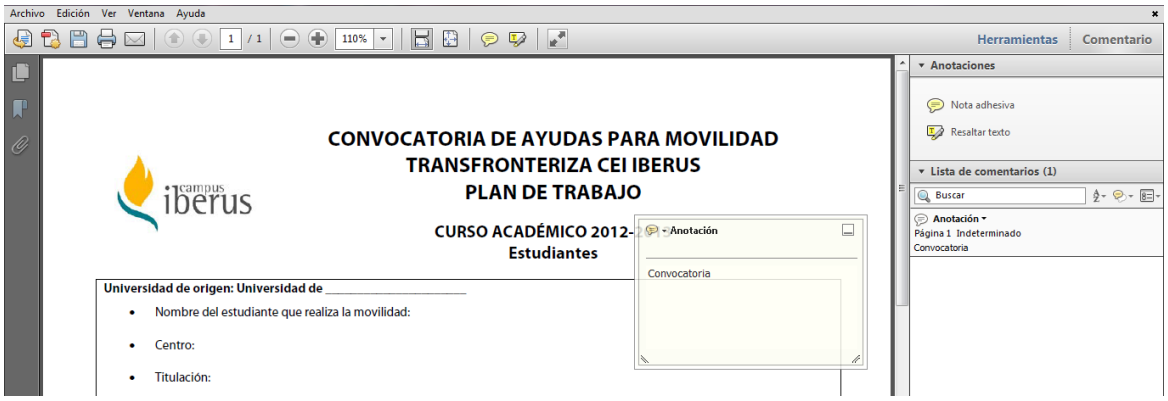


Figura 1 Visualización de archivo PDF con anotación

La manera tradicional, o de usuario, de añadirlas es muy sencilla. El usuario pulsaría sobre la palabra "comentario" de la parte superior derecha de la imagen y automáticamente aparecería la columna que tenemos en la parte derecha de la imagen. No tendríamos más que pulsar sobre la palabra "Nota Adhesiva" y pinchar en la parte del texto del archivo en la que queramos que este la anotación. Una vez habido colocado la anotación en su correcto lugar, rellenaremos el espacio en blanco que hay en la anotación con el contenido que nosotros queramos. Después de esto, podemos ver cómo queda un pequeño rectángulo amarillo (el color puede variar según en usuario) que sirve para advertir al lector del archivo que ahí hay una anotación adherida.

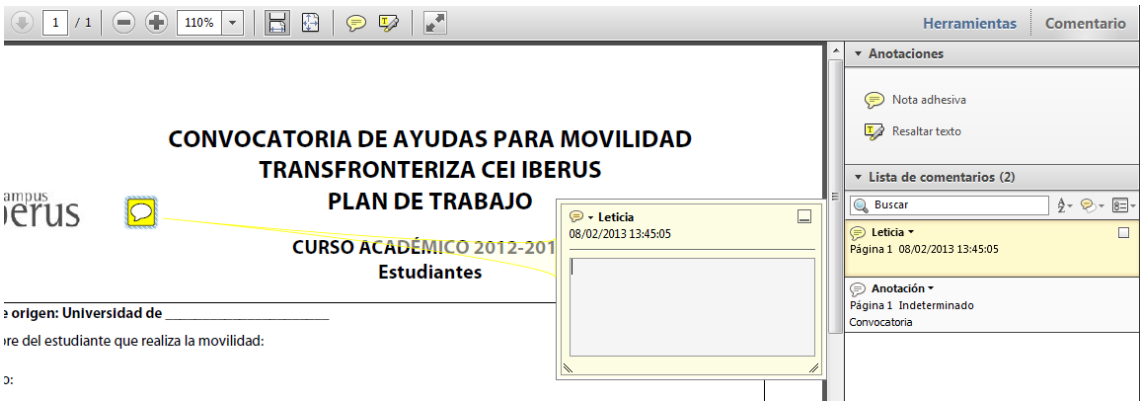


Figura 2 Visualización de archivo PDF con anotacion seleccionada

Pero para hacerlo con el lenguaje de programación que estamos utilizando, C#, y la ayuda de las librerías necesarias el código que corresponde a la agregación de una anotación en es el siguiente:

```
//LECTURA
PdfReader reader = new PdfReader(ruta_del_archivo);

//DONDE QUEREMOS ESCRIBIR EL PDF CON LA/S ANOTACIONES
PdfStamper stamper = new PdfStamper(reader, new
FileStream("C:/Users/Leticia/Documents/Visual Studio
2010/Projects/version1/version1/resultados/" + titulo_pdf + "_anotado_" + contenido +
".pdf", FileMode.Create, FileAccess.ReadWrite, FileShare.None));

//Rectangulo se puede modificar como queramos, tipo, color etc...
PdfAnnotation anot_nueva;
PdfWriter writer = stamper.Writer;
PdfString titulo = new PdfString("Anotación");

foreach (KeyValuePair<int, string> anot in lista_paginas)
{
        contenido = lista_paginas.ElementAt(i).Value;
        nnotation.CreateText(stamper.Writer, new iTextSharp.text.Rectangle(40, 500, 360,
530), "" + titulo + "", "" + contenido + "", true, "help");
        stamper.AddAnnotation(anot_nueva, anot.Key);
        i++;
}
stamper.Close();
```

Figura 3 Código para crear y agregar una anotación

Mediante el código escrito aquí arriba podemos definir en qué página y en qué parte de la página queremos poner la anotación, su contenido, el título y el aspecto que tendrá la anotación, es decir, color, forma, tamaño,…

A pesar de que todos, o la gran mayoría, de lectores de archivos PDF tienen implementada la función de búsqueda de palabras en el texto y la función de añadir anotaciones, no tenemos constancia de que exista una aplicación que englobe todas las funciones que han sido implementadas en esta, ya que dentro de la misma aplicación podemos buscar palabras y anotaciones, crearlas, modificarlas e informar de toda la información que queramos sobre los archivos PDF que hemos estudiado.

1.2 Herramientas de programación

Para la programación de la aplicación hemos utilizado la herramienta Microsoft Visual Studio 2010 Professional para C# sobre el sistema Windows 7.

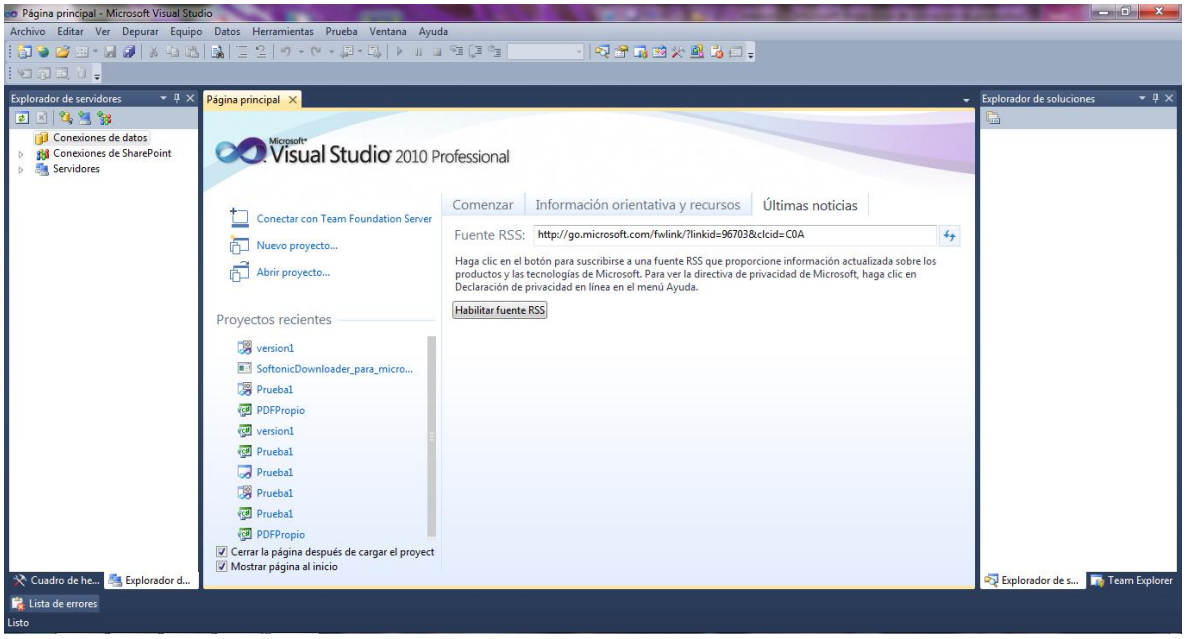


Figura 4 Interfaz principal de inicio de Visual Studio 2010

El lenguaje que hemos utilizado para la aplicación ha sido C#, que es un lenguaje orientado a objetos que fue desarrollado y estandarizado por Microsoft y forma parte de la plataforma .NET. Tiene una gran similitud a Java pero incluye muchas mejoras que derivan de otros lenguajes, tales como: C++, Java, Visual Basic o Delphi. [3]

La utilización de este programa, y la mejora desde otras versiones anteriores a la misma, ha ayudado mucho en el proceso de implementación ya que aparecen constantes ayudas y avisos cuando existen fallos en la escritura o estructura del programa que estamos llevando a cabo. Además, el "debugger" que tiene es clave a la hora de encontrar y arreglar fallos y errores en la implementación y a la hora de ejecutar el programa, haciendo ver posibles errores que no se han descubierto antes en pruebas realizadas.

Para saber un poco más a acerca de los programas en C#, a continuación veremos la estructura general que siguen a la hora de ser implementados:

```
using System;
namespace YourNamespace
{
    class YourClass
    {
    }

    struct YourStruct
    {
    }

    interface IYourInterface
    {
    }

    delegate int YourDelegate();

    enum YourEnum
    {
    }

    namespace YourNestedNamespace
    {
        struct YourStruct
        {
        }
    }

    class YourMainClass
    {
        static void Main(string[] args)
        {
            //Your program starts here...
        }
    }
}
```

Figura 5 Estructura de un programa en C#

Los programas en C# pueden constar de uno o varios archivos. Así mismo, cada archivo puede contener cero o varios espacios de nombres. Un espacio de nombres puede contener tipos como clases, structs, interfaces, enumeraciones y delegados, además de otros espacios de nombres diferentes.

A continuación explicaremos brevemente que son cada uno de estos componentes:

- Clases: son un tipo de datos eficaces que definen los datos mismos y su comportamiento. Son tipos de referencia. Las clases admiten herencia a diferencia de las estructuras.

- Estructuras: son tipos de valor que pueden crear constructores. No aceptan herencia.
- Espacio de nombres: se utilizan de dos maneras. .NET Framework los utiliza para organizar las múltiples clases y también para declarar un espacio de nombres propio y poder controlar el ámbito de clase y nombres de método en proyectos grandes.
- Interfaces: describen un grupo de comportamientos que pueden pertenecer a cualquier clase o estructura. Sus miembros con automáticamente públicos.
- Delegados: son unos tipos que hacen referencia a un método. Cuando se asigna un método a un delegado, este se comporta exactamente como el método. [4]

La aplicación que se ha creado es una aplicación de tipo escritorio. Se utilizará en ordenadores donde el usuario cargará la aplicación para el uso de la misma. Este tipo de aplicación ayuda a los usuarios a realizar la utilización de una herramienta de manera sencilla para resolver una necesidad o tarea que tenían o para simplificarla.

CAPÍTULO 2: METODOLOGÍA

Equipo de trabajo:

Jefe de proyecto: Leticia Osácar.

Cliente: Jesús Villadangos.

Revisiones:

Hemos realizado revisiones de cómo se iban cumpliendo los requisitos del proyecto prácticamente todas las semanas. A medida que se han ido cumpliendo los requisitos también han ido apareciendo nuevos, los cuales se han solventado de la misma manera que los anteriores.

Una vez terminado el proceso de cumplimentación de los requisitos, se empezaron a hacer pruebas para comprobar la consistencia de la aplicación frente a las pruebas que a las cuales la hemos sometido.

Finalmente, nos centramos en el diseño de la interfaz que utilizamos para el uso del cliente y en ver de qué manera se podía maximizar su eficiencia.

Para la realización de esta aplicación hemos utilizado un desarrollo iterativo, el cual ha sido estructurado en las siguientes 3 fases o iteraciones:

FASE 1: las herramientas que hemos utilizado para programar la aplicación eran conocidas por haberlas usado para la realización de otros proyectos. Por tanto, enfatizamos en la lectura y práctica de tutoriales relacionados con la extensión PDF de los archivos y el lenguaje C#.

FASE 2: durante esta etapa implementación de los métodos y funciones que se han necesitado para la creación del programa. Creación de la base de datos a utilizar en el proyecto y las tablas necesarias.

FASE 3: esta última etapa se ha centrado en la definición y en el diseño de la interfaz. Hemos refinado el código de la implementación y hemos completado pequeños detalles relacionados con la utilización del programa de cara al usuario final.

CAPÍTULO 3: REQUISITOS

3.1 Requisitos generales

Lo que vamos a tratar de realizar mediante este proyecto será el análisis de textos en archivos PDF y el vuelco de esta información en una base de datos de tal manera que el usuario pueda consultar los datos que han sido analizados fácilmente y pueda tenerlos accesibles desde la aplicación.

Podemos dividir las funciones de la aplicación en dos partes.

Primero nos dedicaremos al almacenamiento de la información de los archivos PDF y de escribir en ella las etiquetas/anotaciones necesarias. En esta interfaz tenemos la parte en la que introducimos o elegimos la palabra/s que queremos buscar en los PDF, pero antes el usuario habrá seleccionado que tipo de búsqueda quiere llevar a cabo, es decir, si quiere realizar una búsqueda individual (uno a uno) de la palabra en los archivos o si quiere que esta búsqueda se haga automáticamente dentro de un directorio. Si las palabras se encuentran en el documento añadiremos una anotación por cada palabra y página en las que se encuentre, sino desecharemos el archivo. Durante este proceso se añadirán a la base de datos los datos necesarios para luego poder obtener la información de los documentos y sus anotaciones pulsando el botón “PDF Informativo”. Durante el proceso de anotación aparecerá una barra de progreso que ayudará al usuario a saber la duración que tendrá la acción y saber cuánto tiempo restante le queda.

La segunda parte se basa en la visualización de los documentos PDF con los que hemos trabajado y los que hemos conseguido con las anotaciones. Podremos elegir qué carpeta queremos analizar, que documento queremos visualizar y este se podrá ver en la interfaz de usuario. Debajo de la lista de archivos del directorio seleccionado, podremos encontrar la información básica del documento en pantalla.

Todo ello se llevara a cabo mediante dos interfaces, aunque anteriormente el usuario podrá elegir entre varias opciones para realizar la búsqueda (automática/individual).

La navegación básica se realizará entres 4 ventanas fácilmente manejables por el usuario.

No habrá más de un tipo de usuario al que se dirija esta aplicación, será un usuario estándar.

3.2 Requisitos del sistema y hardware

Para la creación de la aplicación hemos utilizado el programa Microsoft Visual Studio 2010 Profesional sobre el sistema operativo Windows 7. También se pueden instalar en los siguientes sistemas operativos: Windows 7; Windows Server 2003 R2 (32-Bit x86); Windows Server 2003 R2 x64 editions; Windows Server 2003 Service Pack 2; Windows Server 2008 R2; Windows Server 2008 Service Pack 2; Windows Vista Service Pack 2; Windows XP Service Pack 3.

De la misma manera, también se requieren los siguientes requisitos mínimos del sistemas: CPU a 1.6GHz o más rápido, 1024MB de RAM, 3GB de espacio libre en el disco duro, tarjeta compatible con DirectX9 con una resolución de pantalla de 1024x768 o mayor y una unidad de DVD-ROM.

3.2.1 Librerías necesarias

Tenemos que tener en cuenta que para la creación y el correcto funcionamiento de la aplicación utilizamos muchos métodos durante su implementación, por tanto, tenemos que añadir en el programa VS2010 las referencias a múltiples librerías o referencias externas según vaya apareciendo su necesidad durante la creación de la aplicación, ya sea por medio del aviso del programa o por nuestro conocimiento. Las librerías y referencias que aparecen a continuación han sido añadidas a las que vienen por defecto en todo proyecto que se inicia en VS2010.

- **AcroPDFLib y AxAcroPDFLIb:** se trata de dos espacios de nombres que contienen varios conjuntos de métodos que permiten acceder a los controles del navegador de PDF. Nos permiten la carga de un archivo PDF, la navegación por varias páginas dentro del archivo PDF y la visualización de diversas opciones de impresión del mismo.

- **Itextsharp:** es una librería que nos permite generar y manipular archivos PDF dentro de C# en la plataforma .NET. Es una de las librerías clave en este proyecto, ya que nos aporta gran parte de los métodos necesarios para el manejo de archivos PDF.

- **PDFBox-0.7.3:** es una librería Open Source en Java que nos permite trabajar con documentos PDF. Nos permite crear nuevos documentos PDF, manipular los existentes y poder extraer un amplio contenido de los documentos.

- **System.Data.SqlServerCe:** este espacio de nombres es una colección de clases que se utiliza en el entorno de desarrollo administrado para proporcionar acceso a los dispositivos inteligentes. Se pueden crear y administrar bases de datos de SQL Server Mobile en un dispositivo inteligente, así como establecer conexiones a las bases de datos SQL Server.

- **System.Text.RegularExpressions:** este espacio de nombres proporciona el acceso al motor de expresiones regulares en .NET Framework.

- **System.IO:** este espacio de nombres contiene tipos necesarios para poder leer y escribir en los archivos y secuencias de datos así como los tipos que proporcionan compatibilidad básica con los archivos y directorios.

- **System.Diagnostics:** este espacio de nombres nos proporciona clases que permiten interactuar con los procesos del sistema, registros de eventos y contadores de rendimiento.

- **System.Collections:** este espacio de nombres contiene interfaces y clases que definen varias colecciones de objetos, como listas, colas, matrices de bits, tablas hash y diccionarios.

- **System.Data.SqlClient:** este espacio de nombres nos provee de los datos necesarios para conectar .NET Framework y SQL Server.

3.3 Librería iTextSharp

Para el desarrollo de esta aplicación hemos tenido que adquirir de forma externa diferentes herramientas para poder implementar todo lo relacionado con el manejo de los archivos en PDF. Para todo ello, hemos utilizado la librería iTextSharp junto con otras.

iTextSharp es una libre Open Source que se utiliza para poder manipular archivos PDF, RTF y HTML que .Ha sido desarrollada desde la librería iText para Java, pero esta es una versión específica para el lenguaje C#. Esta API fue creada para ayudar y permitir a los desarrolladores realizar las siguientes tareas:

- Generar documentos e informes basado en datos procedentes de archivos XML y bases de datos.
- Crear mapas y libros, aprovechando numerosas características interactivas disponibles en PDF.
- Añadir numerosas características como: números de páginas, marcas de agua,… existentes en documentos PDF.
- Concatenar y dividir páginas de archivos PDF que ya existen.
- Rellenar formularios interactivos.
- Dar servicio a documentos generados o manipulados dinámicamente en navegadores web.

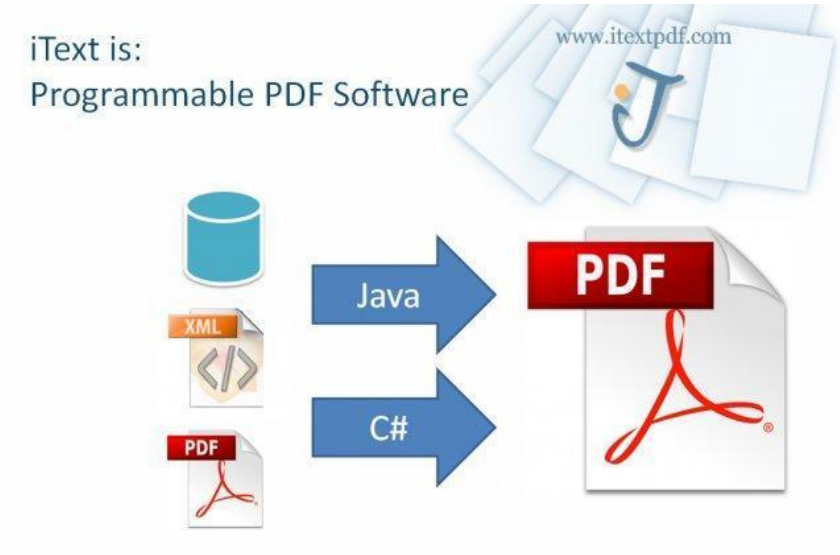


Figura 6 Esquema iTextSharp

Para crear documentos PDF tendremos que utilizar básicamente dos clases: la clase Document y la clase PdfWriter, tal y como veremos en la implementación de la aplicación. Podremos crearlos partiendo desde cero desde una base de datos, un archivo XML o desde cualquier otra fuente. Tenemos tres formas para hacerlo:

- Utilizando objetos de alto nivel como Chunk, Phrase, Paragraph, List,…
- Utilizando funcionalidades de bajo nivel como PdfContentByte. Esta clase contiene una serie de métodos que esquematizan todos los operandos y operadores disponibles en el modelo de imagen de Adobe. Esta clase también tiene numerosos métodos que permiten dibujar arcos, círculos, rectángulos y texto en posiciones absolutas del documento.
- Utilizando PdfGraphics2D, que es la implementación de iTextSharp de la clase Grapics2D de Java. Esta no ha sido utilizada en este proyecto.

iTexSharp puede convertir archivos XML o XHTML/CSS a PDF mediante XML Worker, pero iTextSharp no convierte archivos en Word a PDF.

Si queremos realizar la actualización de archivos PDF realizar una instancia de PdfReader para poder acceder a cualquier archivo existente, se puede combinar con la clase PdfStamper para agregar contenido extra al mismo. También utilizaremos esta clase para rellenar formularios interactivos. Si deseamos dividir o combinar documentos PDF utilizaremos las clases PdfCopy, PdfSmartCopy o PdfCopyFields y también utilizando objetos de PdfImportedPage con PdfWriter o PdfStamper.

iTextSharp además también nos permite firmar archivos PDF existentes así como encriptarlos.

Para la lectura y acceso a los objetos de los archivos PDF utilizaremos la clase PdfReader. Si el contenido no está rasterizado se puede convertir la página a texto plano. Cabe destacar que iTextSharp no realiza OCR. La limitación respecto a este tema nos hará buscar alguna solución puesto que es necesaria una correcta lectura de los elementos en el archivo PDF para poder obtener resultados consecuentes.

Todos estos puntos quedaran aclarados durante la implementación de la aplicación y veremos diversos ejemplos en los que se utilizan y se ve su funcionalidad.

3.4 Diagrama de casos de uso

3.4.1 Caso de uso 1

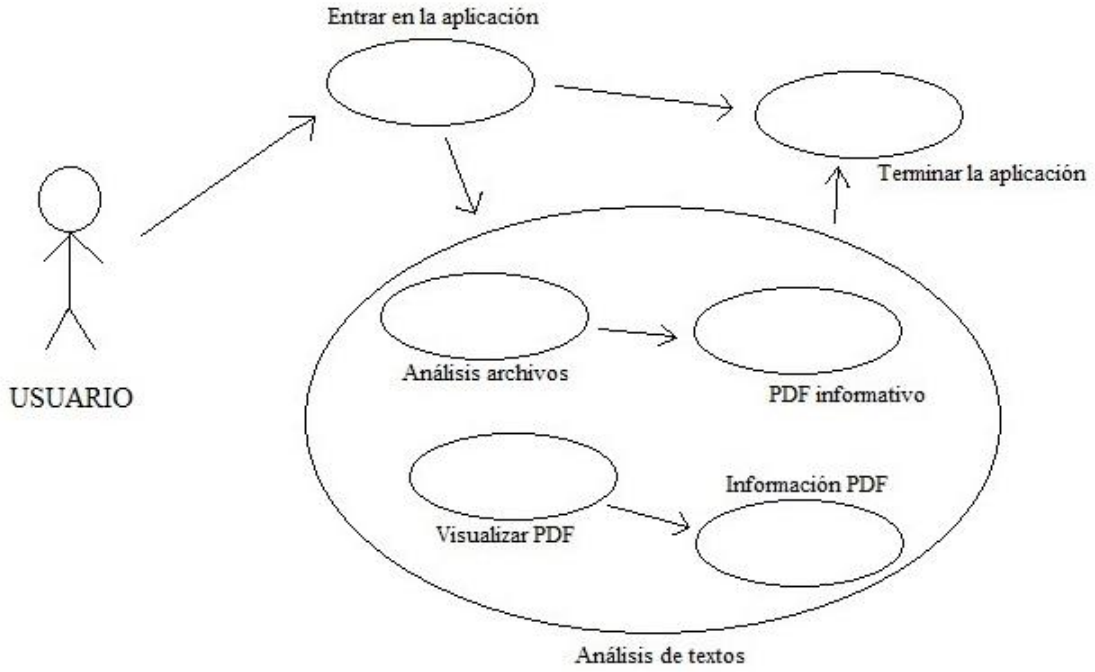


Figura 7 Caso de uso 1

- Actor: nuestro actor será el usuario de la aplicación.
- Descripción: el usuario elige que palabra y que documento quiere analizar. Después de este proceso puede visualizar tanto la información almacenada en la base de datos como los documentos tratados y su información básica.
    - Entrar a la aplicación: el usuario accede a la aplicación.
    - Terminar la aplicación: el usuario finaliza la ejecución de la aplicación.
    - Análisis de textos: este caso de uso se compone de varios diferentes.
        - Análisis de archivos: elegimos que palabra queremos buscar en los documentos que tenemos. Realizamos la búsqueda y si es positiva guardamos los datos en la base de datos y realizamos la anotación de la palabra encontrada en una copia del documento. Si no se encuentra, informamos de la búsqueda negativa al usuario.

- PDF informativo: Nos conectamos a la base de datos y obtenemos los nombres y anotaciones de los archivos que hemos analizado en un PDF informativo.
- Visualizar PDF: elegimos que archivo PDF queremos ver en la interfaz.
- Información PDF: después de elegir el PDF que queremos ver, obtenemos la información básica (título, autor y anotaciones) del fichero.

- Precondiciones: deberemos tener establecida una conexión a la base de datos y tener documentos en formato PDF para su análisis.
- Poscondiciones: el análisis de textos de los documentos ha sido completado y los documentos han sido anotados.
- Flujo alternativo: pueden darse varias situaciones que hagan que la aplicación no funcione: errores en la base de datos, errores a la hora de poblar las tablas de la base de datos, errores en los documentos PDF.

CAPÍTULO 4: ANÁLISIS Y DISEÑO

4.1 Diagramas de secuencia

A continuación se presentan los diferentes diagramas de secuencia que se originan durante el funcionamiento de la aplicación.

4.1.1 Diagrama de secuencia 1

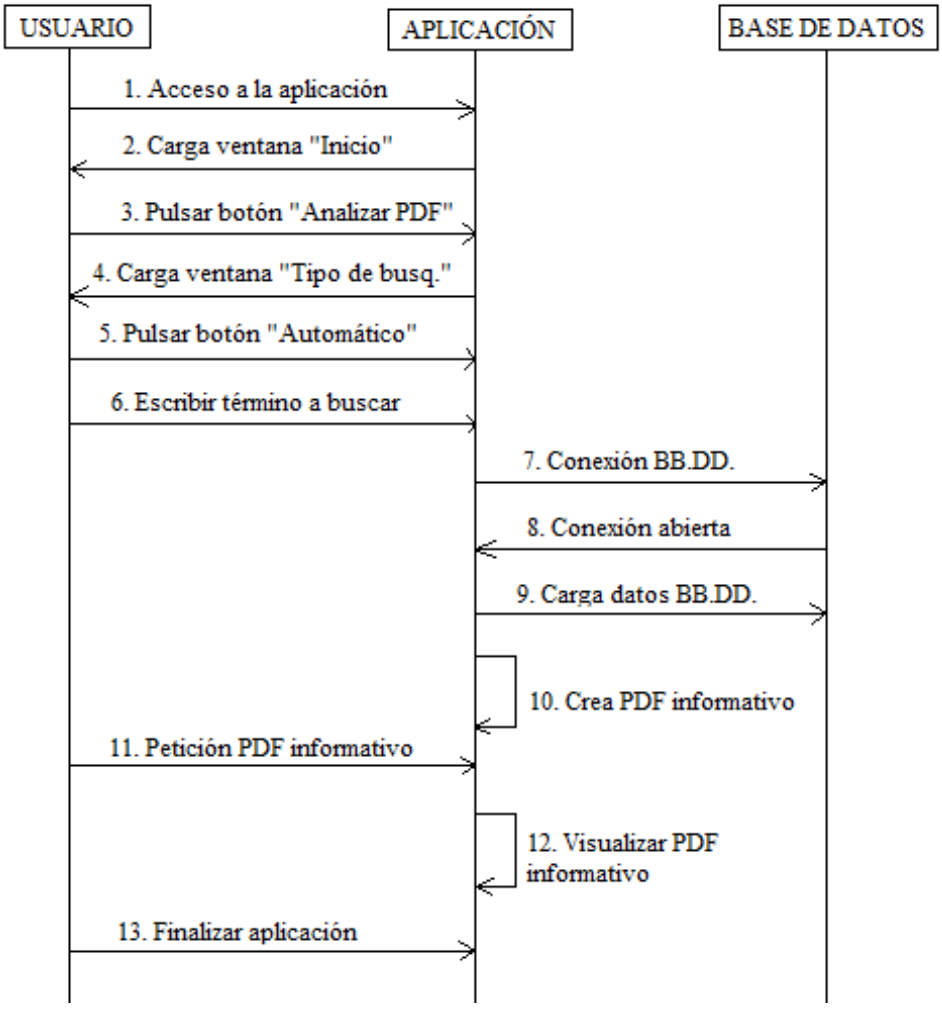


Figura 8 Diagrama de secuencia 1

En este primer diagrama de secuencia vemos que clase de interacción existe entre el usuario, la aplicación y la base de datos.

En primer lugar el usuario accederá a la aplicación y esta le cargará la ventana de "Inicio". A continuación el usuario pulsará el botón "Analizar PDF" y la aplicación cargará la ventana "Tipo de búsqueda", donde el usuario podrá elegir si quiere realizar una búsqueda automática o individual. En este caso, elegirá la búsqueda automática pulsando el botón correspondiente y el usuario en la siguiente ventana ya podrá introducir el término que quiere buscar. Se abrirá una conexión a la base de datos, donde se cargarán los datos del archivo PDF que se está analizando. Si el usuario quisiese realizar la petición del archivo PDF informativo acerca de los archivos analizados hasta ahora, la aplicación ya la tendría preparada y solamente pulsando el botón de "Pdf Informativo" se desplegaría en la pantalla para que pudiera visualizarlo.

4.1.2 Diagrama de secuencia 2

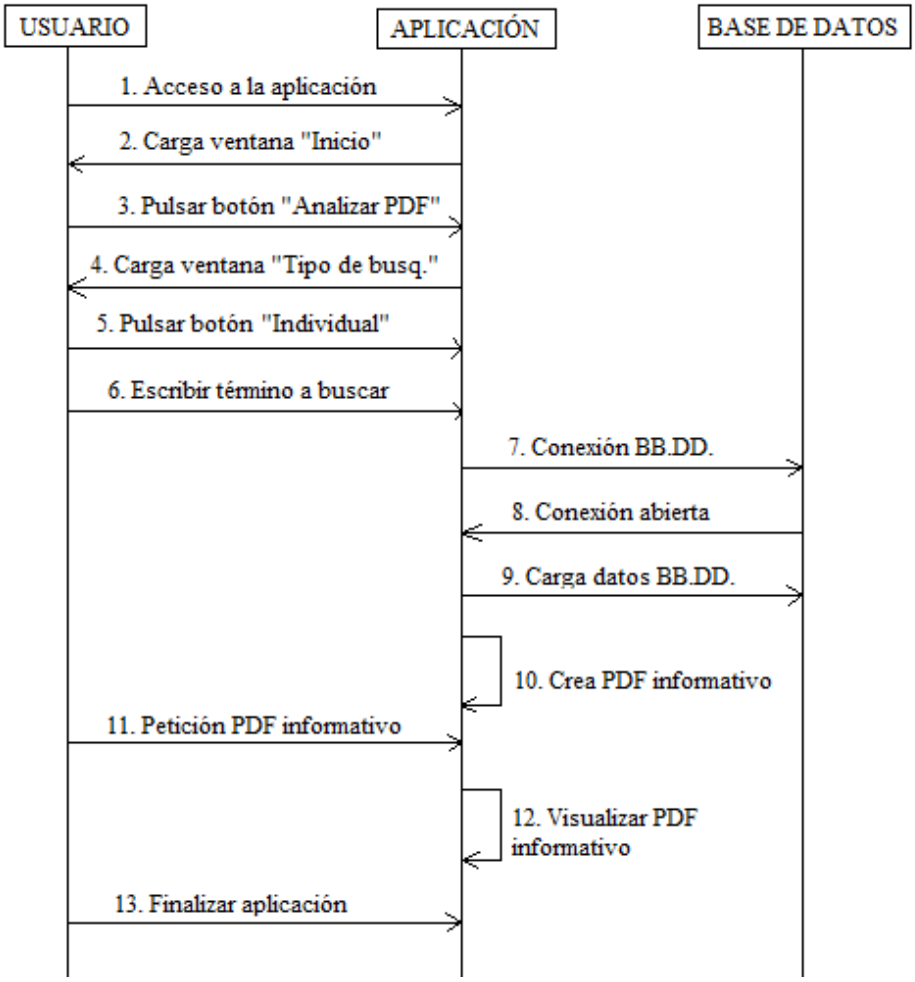


Figura 9 Diagrama de secuencia 2

Este segundo diagrama representa la misma interacción del usuario, la aplicación y la base de datos, la única diferencia radica en el tipo de búsqueda que elije realizar. En este caso se realiza una búsqueda individual de las palabras en los diferentes archivos. Las demás interacciones serán las mismas.

4.1.3 Diagrama de secuencia 3

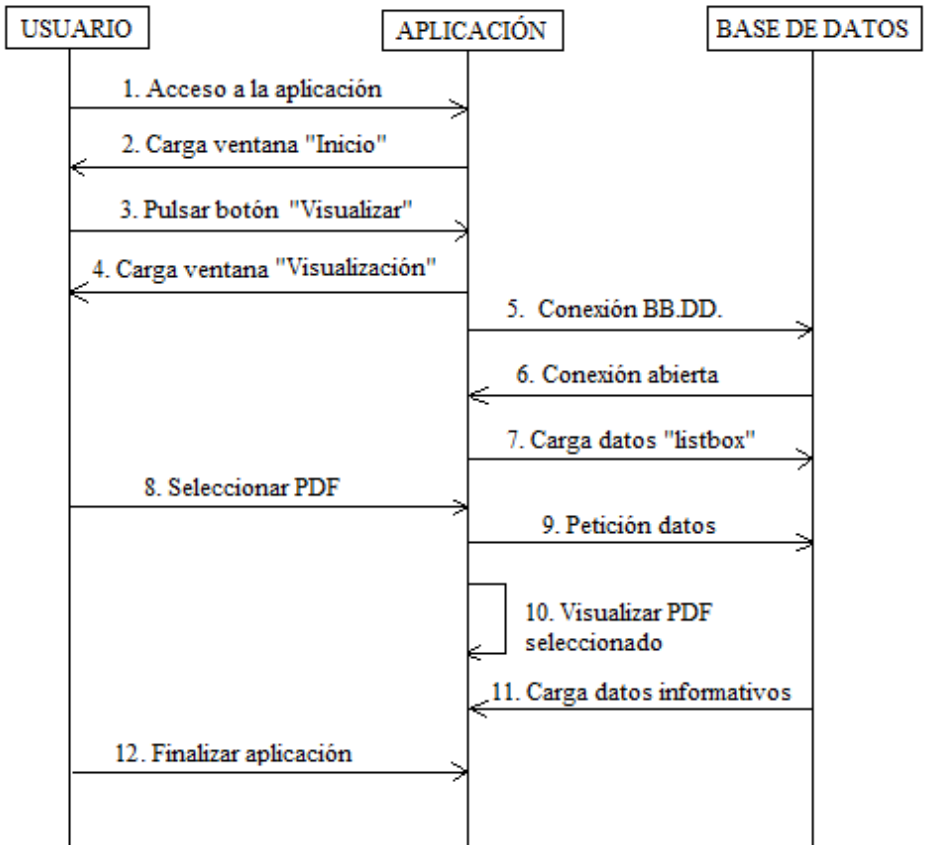


Figura 10 Diagrama de secuencia 3

Este último diagrama de secuencia el usuario en vez de analizar los archivos PDF pide una visualización de los mismos. Por tanto, al cargarse la ventana de "Visualización" se abrirá una conexión con la base de datos, de la cual se obtendrá la lista de archivos para cargar en el listbox, el usuario elegirá de esta lista los archivos PDF que quiera visualizar en esta interfaz y también se cargará desde la base de datos la información pertinente del archivo que haya elegido. El usuario podrá finalizar cuando quiera la ejecución del programa y finalizar la aplicación.

4.2 Diagrama de clases y código de la aplicación

4.2.1 Diagrama de clases

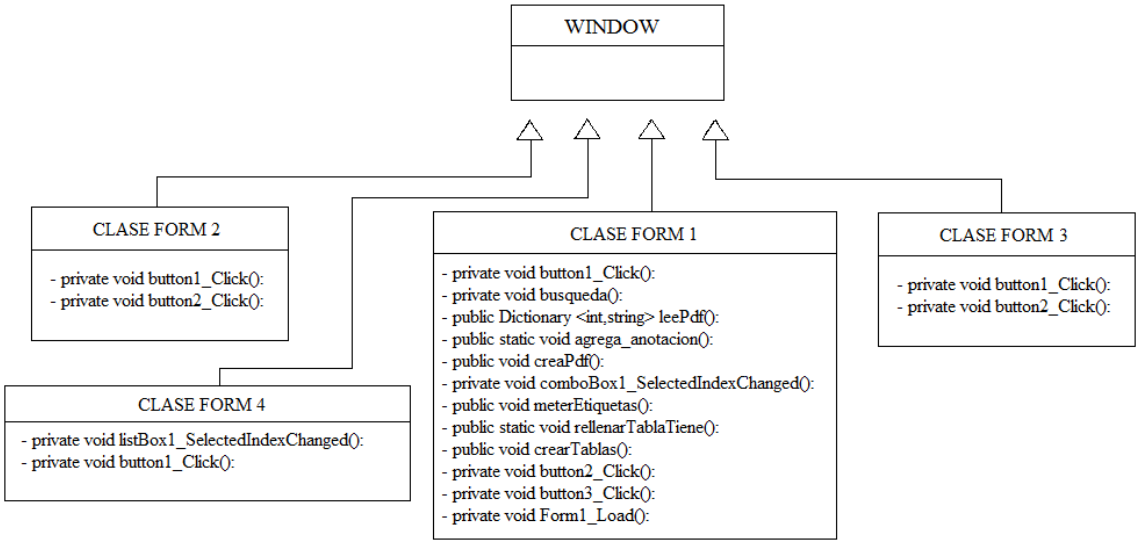


Figura 11 Diagrama de clases

Como podemos comprobar el diagrama de clases superior, todas las clases creadas en la aplicación heredan de una forma simple de la clase general "Window", de la que obtienen todas las funciones para llevar a cabo la implementación de la aplicación.
Después, se han implementado las funciones precisas en cada clase para poder cumplir con los requisitos planteados desde un principio.

4.2.2 Clases y métodos de la aplicación

En este apartado explicaremos las clases y métodos que formar la aplicación que hemos creado.

Contamos con cuatro clases, Form1, Form2, Form3 y Form4, en toda la aplicación. Dentro de ellas se llevan a cabo todos los métodos necesarios para el funcionamiento de la aplicación.

- Clase "Form1":
  - *public void búsqueda():*

Esta función se encarga de realizar la búsqueda de la palabra elegida en los PDF de la ruta de archivos que elijamos. Si la palabra no se encuentra en ninguno de los archivos de la ruta que hallamos seleccionado nos aparecerá un mensaje informativo advirtiéndonos de ello, sino se ejecutaran todas las funciones del programa.

  - *public Dictionary<int,string> leePdf():*

En esta función conseguimos realizar la lectura del archivo PDF que hemos elegido. En caso de que existe algún tipo de error en el archivo a leer, nos saltará una ventana de error que nos avisará del fallo.

Puesto que puede darse el caso de que se puedan escribir palabras cualesquiera para la búsqueda, tenemos que tener en cuenta que pueden haber problemas con las minúsculas y mayúsculas, ya que se diferencian en la forma que se escriben pero no en el significado. Para ello utilizamos método con expresiones regulares que nos tendrá en cuenta las diferentes variables en la forma de la palabra.

```
//Expresion regular para encontrar la palabra tal cual viene en el texto
else if (System.Text.RegularExpressions.Regex.IsMatch(currentText,
elegido,System.Text.RegularExpressions.RegexOptions.IgnoreCase))
{
        paginas_etiquetas.Add(cPage, elegido)
}
```

Figura 12 Código para expresiones regulares

  - *public string almacenar_pdf():*

Mediante esta función almacenamos los datos de interés (identificación, título y fecha de creación) del archivo PDF que hemos elegido con ayuda de la interfaz en la base de datos.

  - *public static void agrega_anotacion():*

Con esta función añadimos una anotación por cada palabra y página a buscar que encontremos. Podemos definir libremente qué título y contenido que va a llevar la anotación así como su posición en la página cuando la encuentre.

Dentro de esta función realizamos la función de rellenar una de las tablas de la base de datos, la tabla "tiene" más concretamente.

Un aspecto a tener en cuenta en este paso es el de la protección que contienen algunos archivos PDF. Algunos archivos vienen protegidos por contraseñas que no pueden ser adivinadas de ninguna manera y no permiten que los archivos puedan ser copiados o modificados, pero sí que permiten que puedan ser leídos. Por tanto, tendremos acceso al contenido de los archivos pero no podremos realizar una copia y añadir una o varias anotaciones a este archivo que creamos. Esto quiere decir que si nos encontramos frente a un archivo protegido solamente podremos utilizarlo para ver si existe la palabra o las palabras que se quieren buscar en él, contabilizarla a continuación pero no tendremos una copia con las anotaciones en el directorio creado para ello.

- *public void creaPdf():*

En esta función crearemos un archivo PDF informativo que nos dará información de los archivos que están almacenados en la base de datos en un momento concreto. Obtendremos un documento con los títulos y etiquetas/anotaciones que contienen cada uno de ellos realizando la lectura de la tabla "tiene" de la base de datos y plasmando los datos que contenga en un archivo PDF nuevo.

```
Document doc = new Document(PageSize.A4, 10, 10, 10, 10);
string filename = "C:/Users/Leticia/Documents/Visual Studio
2010/Projects/version1/version1/resultados/info_pdfs.pdf";
FileStream file = new FileStream(filename, FileMode.OpenOrCreate);
PdfWriter.GetInstance(doc, file);
doc.Open();
```

Figura 13 Código para crear un archivo PDF

```
//Escribimos en el nuevo documento
char[] carac = { '[',']' };
for (int k = 0; k < lista2.Count; k++)
{
        string[] titulos = lista1[k].ToString().Split(carac);
        string[] etis = lista2[k].ToString().Split(carac);

        doc.Add(new Paragraph("- Titulo Pdf: "+titulos[0].ToString()));
        doc.Add(new Paragraph("\t \t - Etiquetas: "+etis[0].ToString()));
        doc.Add(new Paragraph("\n"));
}
doc.Close();
```

Figura 14 Código para añadir párrafos al PDF

- *public void meterEtiquetas():*

Realizamos la operación de rellenar la tabla "Etiquetas" en la base de datos nada más comenzar la aplicación. Obtendremos las etiquetas que existen en ese momento que almacenar de la lectura de los ítems del combobox y cada vez que el usuario escriba una palabra nueva a buscar en el textbox.

- *public static void rellenarTablaTiene():*

Rellenamos la tabla "tiene" en la base de datos. Esta tabla se refiere a las etiquetas que contiene cada archivo PDF que analizamos. Insertaremos el identificador de cada etiqueta y el identificador de cada archivo PDF para relacionarlos entre sí.

- *public void crearTablas():*

Con esta función creamos al inicio de la ejecución del programa las tablas necesarias en la base de datos. Las tablas son las siguientes: "etiquetas", "pdfs" y "tiene".

- *private void button2_Click():*

Creamos el PDF informativo llamando a la función creaPdf().

- *private void button3_Click():*

Llamamos a la función busqueda() con el atributo que cogemos desde el combobox o bien desde textbox en el que escribimos la palabra exacta que queremos buscar. En caso de que los dos estén vacíos o escritos nos aparecerá un mensaje avisando al usuario de ello.

- *private void Form1_Load():*

Queremos que cuando se ejecute por primera vez la cargar de la ventana al usuario se realicen una serie de funciones, tales como: crear las tablas en la base de datos, rellenar las opciones que se van a dar en el combobox al usuario, rellenar la tabla "etiquetas" de la base de datos y crear el directorio en el que se guardarán los archivos PDF que se anoten. Esta función sólo se realizará una vez por ejecución de programa.

Cabe resaltar que utilizamos el traspaso de variable entre ventana y ventana dentro de esta aplicación. Puesto que necesitamos saber cuál de los dos botones (búsqueda automática o individual) ha sido pulsado por el usuario, utilizaremos una variable que cambiará de valor según el botón que haya pulsado el usuario. Esta variable vendrá de la ventana anterior. Solamente tenemos que cargar la variable a la hora de invocar la función de creación de la ventana que queremos.

```
auto = 1;
Form1 frm = new Form1(auto);
frm.Show();
```

Figure 15 Código para el traspaso de variables 1

```
public Form1(int auto)
{
        InitializeComponent();
        this.auto = auto;
}
```

Figure 16 Código para el traspaso de variables 2

- Clase “Form 2”:
  - *private void button1_Click():*

    Con esta función lanzamos el formulario “Form 4” a pantalla.

  - *private void button2_Click():*

    Con esta función lanzamos el formulario “Form 3” a ejecución.

- Clase “Form 3”:
  - *public Form3():*

    Utilizamos esta función precreada para rellenar el listbox con los archivos PDF anotados por defecto. También hacemos invisibles los labels que utilizaremos para plasmar la información que se obtenga de los archivos PDF en un primer momento.

  - *private void button1_Click():*

    Al pulsar este botón daremos al usuario la opción de poder cambiar la ruta de los archivos que quiere que se le muestren en el listbox principal. El usuario sabrá en todo momento la carpeta de los archivos que se están mostrando puesto que aparecerá su nombre encima del listbox que contiendo los nombres de los archivos PDF.

```
System.Text.StringBuilder nom_pdf_final = new System.Text.StringBuilder();
nom_pdf_final = nom_pdf_final.Append(nom_pdf[0]);
for (int k = 1; k < nom_pdf.Length - 3; k++)
{
        nom_pdf_final.Append('_');
        nom_pdf_final.Append(nom_pdf[k]);
 }
 string nom_pdf_f = nom_pdf_final.ToString();
```

Figure 17 Código para crear la cadena de texto del nombre de archivo

  - *private void listBox1_SelectedIndexChanged():*

    Activamos y cargamos de la información que le ha sido asignada a los labels que vamos a utilizar para mostrar la información del archivo PDF. Puesto que no se

puede extraer la información acerca del autor de todos los documentos PDF hemos implementado unas líneas de código que nos ayudan a saber si existe esa información en el documento que estamos analizando.

```
label6.Text = "-";
Dictionary<string, string> metadatosPDF = R.Info;
foreach (KeyValuePair <string,string> clave in metadatosPDF)
{
        if (clave.Key == "Author")
        {
                label6.Text = clave.Value;
        }
}
```

Figura 18 Código para la búsqueda del autor

Realizamos una búsqueda en la base de datos de las etiquetas que corresponden al archivo PDF que ha sido seleccionado por el usuario y las mostramos en el segundo listbox de la interfaz.

- Clase "Form4":
  - *private void button1_Click():*

Cargamos el formulario "Form1" y pasamos la variable auto con valor 1. De esta manera sabremos si se trata de una búsqueda automática o manual.

  - *private void button2_Click():*

Cargamos el formulario "Form1" y pasamos la variable auto con valor 0.

CAPÍTULO 5: BASE DE DATOS Y DISEÑO DE LA INTERFAZ

5.1 Base de datos

La aplicación que hemos creado se conecta a una base de datos para poder realizar el recuentro y control de las palabras que aparecen en los diferentes documentos que vamos a analizar. Para ello hemos creado un esquema de entidad relación para el correcto diseño de la base de datos.

5.1.1 Modelo entidad – relación

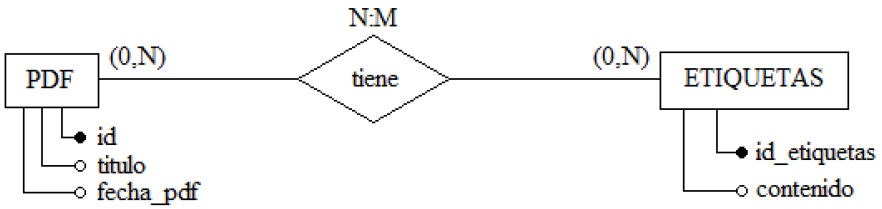


Figura 19 Modelo Entidad - Relación

5.1.2 Análisis de las entidades y relaciones

PDFS	ETIQUETAS
ID	ID_ETIQUETAS
TÍTULO	CONTENIDO
FECHA_PDF	

Figura 20 Tabla de entidades y atributos

- **Análisis de las entidades:**

PDFS: esta entidad nos representa el archivo PDF del que vamos a obtener el análisis textual. Estos archivos pueden contener o no etiquetas/anotaciones.

- Atributos:
    - Id: se trata de identificar de manera única cada PDF que analicemos.
    - Título: guardamos el título de cada documento estudiado.

- Fecha_pdf: nos referimos a la fecha de creación de cada archivo PDF.

ETIQUETAS: con esta entidad identificamos las palabras que el usuario quiere buscar y se encuentran en los archivos analizados. Funcionan como si fuesen anotaciones que se hacen en el documento estudiado.

- Atributos:
  - Id_etiquetas: otorgamos un identificativo único a cada nueva palabra convertida en etiqueta para tenerlas todas distinguidas.
  - Contenido: con este atributo guardaremos el contenido de las etiquetas/anotaciones que hagamos en el documento.

- **Análisis de las relaciones**

tiene (N:M): un PDF puede contener ninguna, una o varias etiquetas dentro del mismo y una etiqueta puede encontrarse en ninguno, uno o varios documentos. Con esta relación conseguimos unir cada archivo con las etiquetas/anotaciones que contiene.

- Atributos:
  - Id_pdf: cogeríamos cada identificación de los documentos estudiados.
  - Id_etiquetas: obtenemos la identificación de cada etiqueta.

- **Paso a tablas:**

PK: Clave primaria. FK: Clave foránea.

PDFS:

ID	TITULO	FECHA_PDF
PK		

Figura 21 Paso a tablas 1

ETIQUETA:

ID_ETIQUETAS	CONTENIDO
PK	

Figura 22 Paso a tablas 2

TIENE:

ID_PDF	ID_ETIQUETAS
FK	FK

Figura 23 Paso a tablas 3

La composición de esta tabla se hace con dos claves foráneas, que son las claves primarias de las tablas "PDFS" (id_pdf) y "ETIQUETAS" (id_etiqueta).

5.2 Diseño de la interfaz

La aplicación está estructurada en 4 sencillas interfaces. Mediante la navegación entre las mismas el usuario conseguirá llevar a cabo todas las acciones que han sido implementadas en esta aplicación.

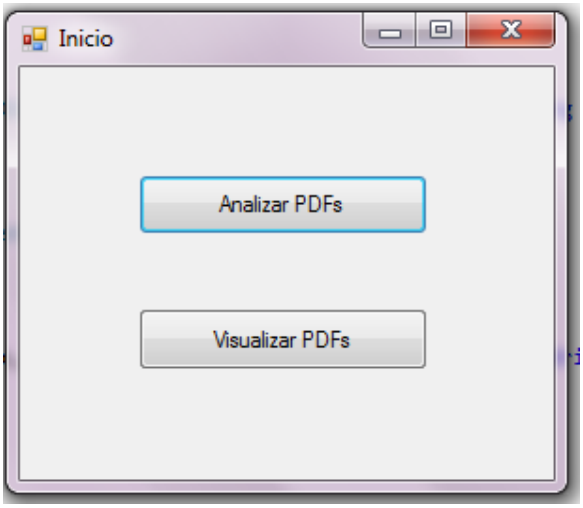


Figure 24 Interfaz "Inicio"

La primera interfaz es la de "Inicio". En ella el usuario encuentra dos botones que puede pulsar, uno el de "Analizar PDFs" y el otro el de "Visualizar PDFs". Si el usuario pulsa el primero le llevará a la interfaz representada en la figura 25 y si pulsa el segundo irá a la interfaz de la figura 30.

Si el usuario quiere terminar la ejecución en este momento sólo tiene que pulsar el botón de cierre de la esquina superior derecha y la aplicación terminará de ejecutarse.

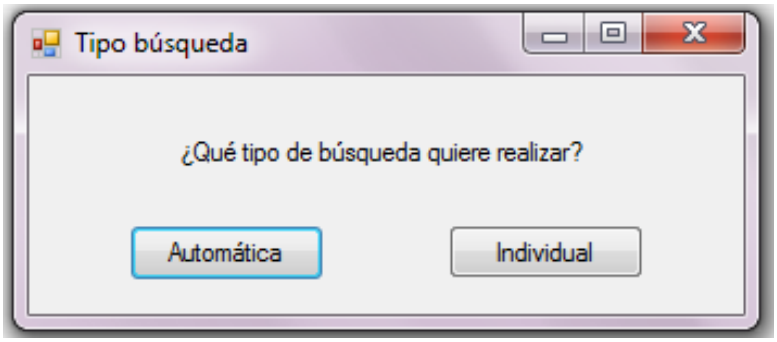


Figura 25 Interfaz "Tipo de búsqueda"

Esta es la interfaz al que usuario accederá si pulsa el botón "Analizar PDFs". Aquí se le presenta la opción de poder elegir el tipo de búsqueda que quiere realizar mediante dos botones, el de "Automática" y el de "Individual". Cuando el usuario pulse cualquiera de estos dos botones tendrá acceso a la interfaz de la figura 26.

Esta interfaz recoge la información del tipo de búsqueda que el usuario quiere realizar, la traspasa a la interfaz de la figura 25 y en función de la misma realizará de manera diferente la búsqueda de los términos que quiere.

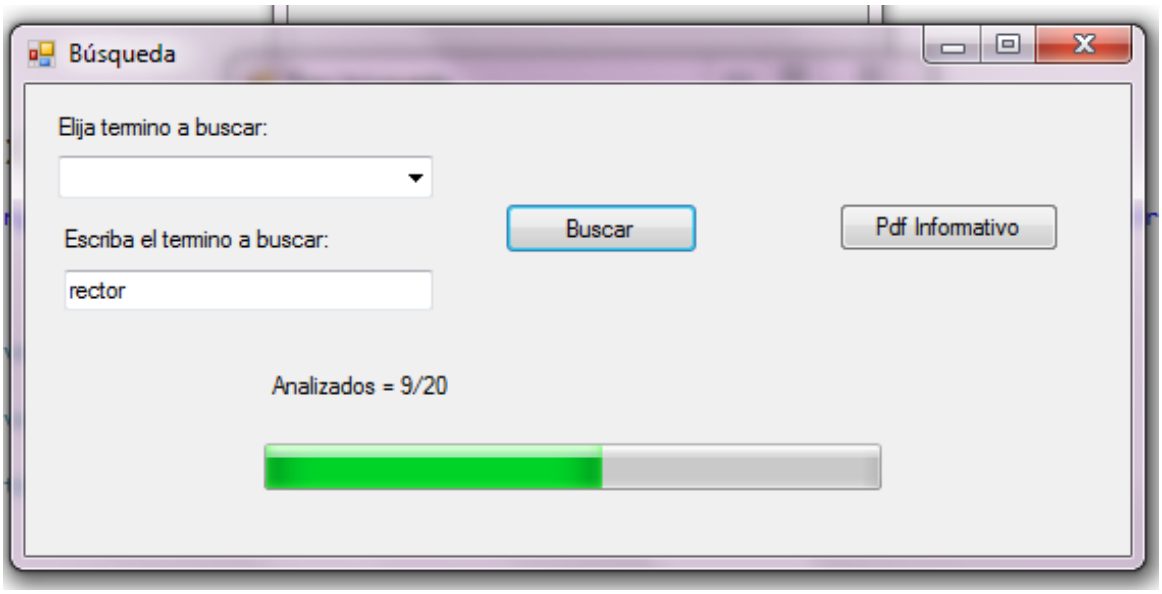


Figura 26 Interfaz "Búsqueda"

Esta es la interfaz de "Búsqueda". Una vez elegido el tipo de búsqueda que se quiere realizar, el usuario ya podrá elegir que término se quiere buscar en los documentos PDF. En esta interfaz contamos con dos botones, un combobox y un textbox, en los cuales se elegirán e insertarán, respectivamente, los términos a buscar. Si el usuario ha elegido la

opción de la búsqueda automática debido a que puede que la búsqueda se demore bastante, se ha implementado una barra de progreso para el usuario sepa en todo momento cuantos archivos se van a analizar y cuantos lleva analizados hasta el momento. Cuando termine el análisis, se desplegará automáticamente un archivo PDF que recogerá todos los hallazgos de la búsqueda para que el usuario pueda saber el resultado de la misma.

Si el usuario quiere elegir una palabra ya establecida en el combobox, lo seleccionará y a continuación pulsará el botón "Buscar", sino el mismo insertará la palabra que quiere buscar en el textbox habilitado debajo para ello. Dependiendo del tipo de búsqueda que haya elegido antes aparecerán dos cuadros de diálogos diferentes, es decir, si el usuario ha elegido una búsqueda automática podrá elegir la carpeta en la que quiere realizar la búsqueda de todas las que haya en el sistema, tal y como se puede ver en la figura 27.

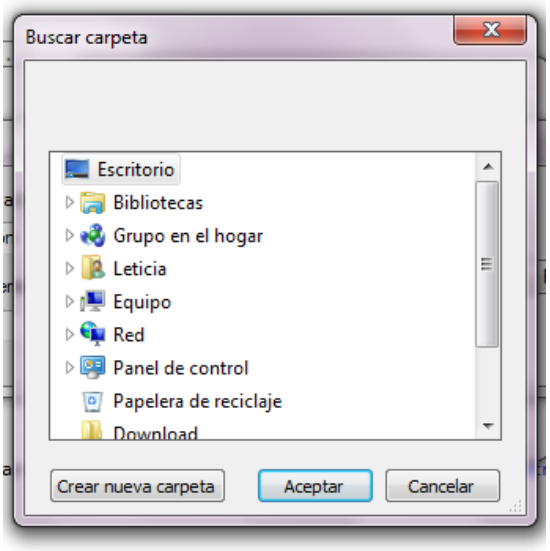


Figura 27 Cuadro de dialogo 1

Podemos observar que el usuario puede elegir la carpeta que quiera para realizar la búsqueda. En cambio, si ha elegido una búsqueda individual se le desplegará un cuadro de dialogo diferente.

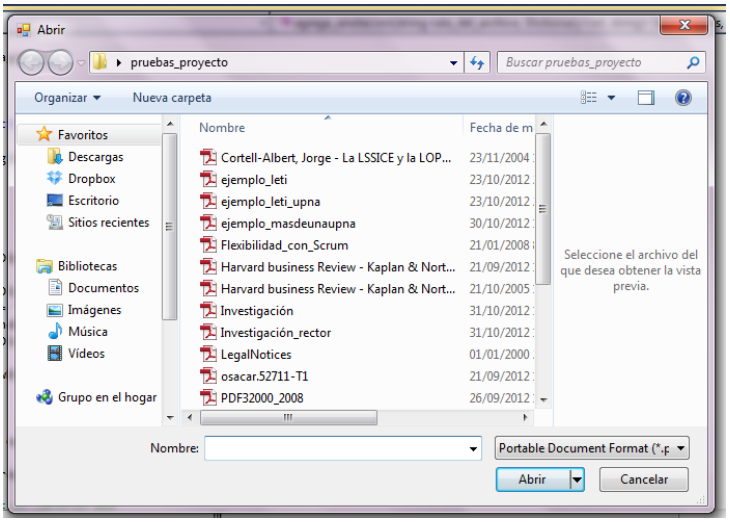


Figura 28 Cuadro de dialogo 2

Este cuadro de dialogo permite al usuario elegir el archivo único sobre el cual quiere realizar la búsqueda dentro de los distintos directorios de su sistema.

Y por último, si el usuario pulsa el tercer botón, es decir, el de "Pdf Informativo" se desplegará una interfaz en la que aparecerá el archivo PDF que contiene el estado de la base de datos de los archivos con sus correspondientes etiquetas.

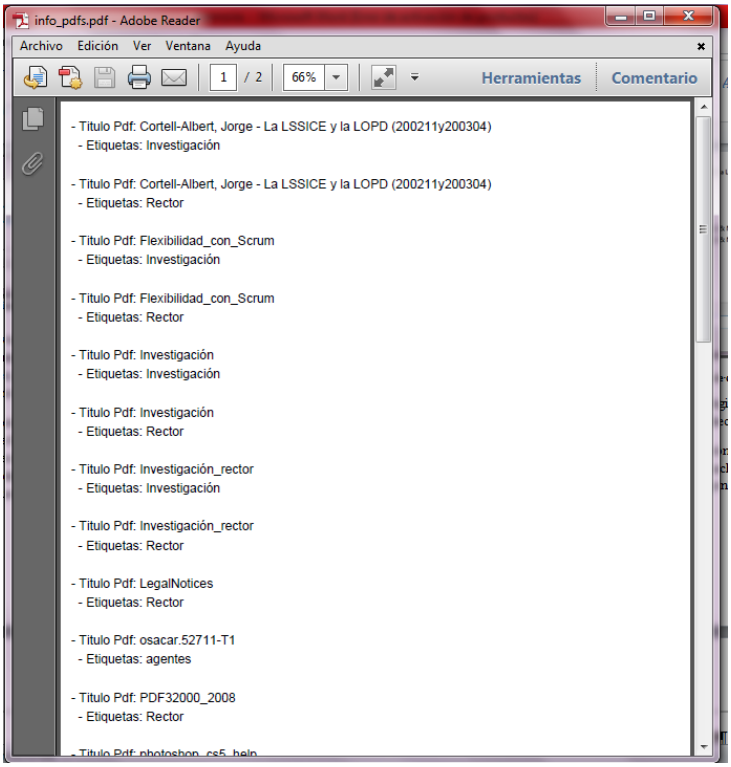


Figura 29 Interfaz despliegue PDF

Podemos comprobar que aparece el listado de archivos PDF analizados con su título y con cada una de las etiquetas que se han encontrado en ellas.

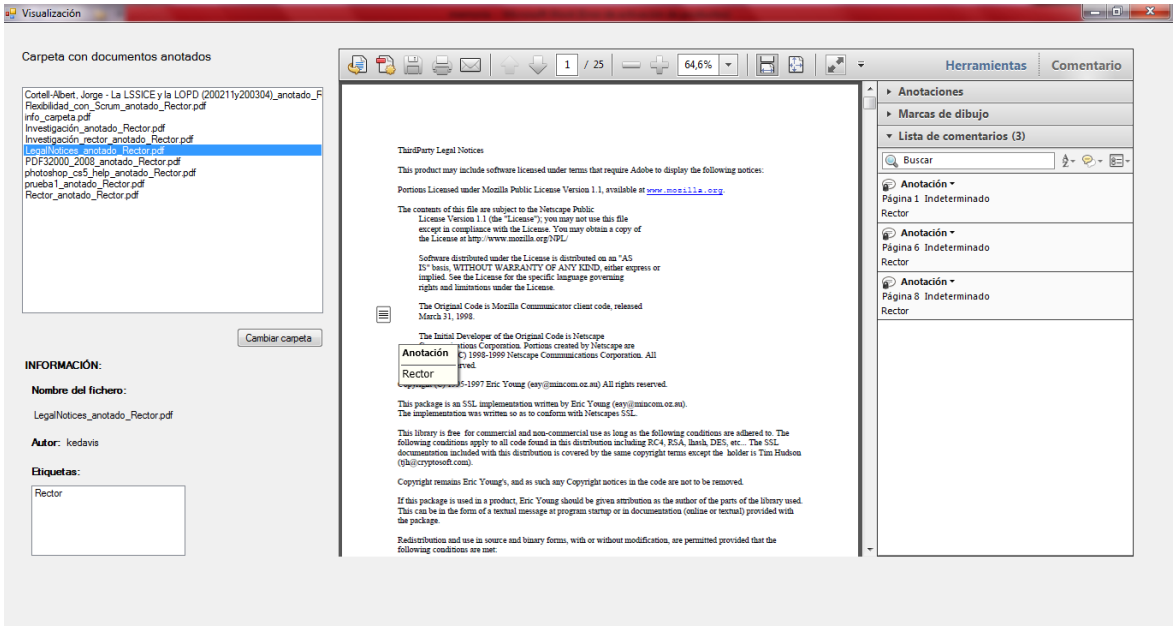


Figura 30 Interfaz "Visualización"

Esta es la última interfaz. Es la interfaz en la que podemos visualizar los archivos PDF con sus etiquetas e información básica.

Como podemos observar tenemos un listbox que contiene todos los archivos que están en la carpeta actual. La carpeta que viene por defecto es la carpeta en la que archivan los documentos PDF anotados, pero el usuario puede cambiar esta carpeta en el momento que quiera pulsando el botón que se encuentra debajo del listbox. Debajo del listbox se encuentra la información del archivo que este seleccionado, la información está expuesta mediante labels y otro listbox más que sirve para acumular las etiquetas de cada archivo. El resto de la interfaz es un visualizador de archivos PDF integrado en la interfaz. El archivo que se visualice será el seleccionado desde el listbox.

En cualquier punto de la ejecución de la aplicación el usuario podrá finalizar la misma pulsa sobre el botón rojo de la esquina superior derecha de la interfaz.

CAPÍTULO 6: PRUEBAS DE RENDIMIENTO

Hemos realizado un estudio acerca de la relación entre el tiempo en el que se tarda en realizar el trabajo de análisis y anotación de los archivos en función del número de palabras, caracteres y tamaño que tienen los documentos a estudiar.

A continuación se mostrarán las gráficas conseguidas después de realizar el estudio.

6.1 Gráficos

tamFichero	num KCarac	num KPal	tiempo (s)
86 KB	0,14	0,02	38,03
89 KB	2,27	0,37	53,00
207 KB	4,27	0,80	46,31
357 KB	30,08	5,27	103,41
1402 KB	51,13	9,31	99,00
136 KB	67,43	12,11	54,06
5559 KB	180,63	34,25	682,91
879 KB	359,95	63,76	151,03
38727 KB	1509,11	307,13	10955,78
8785 KB	1673,69	335,38	7987,09

Figura 31 Tabla de datos 1

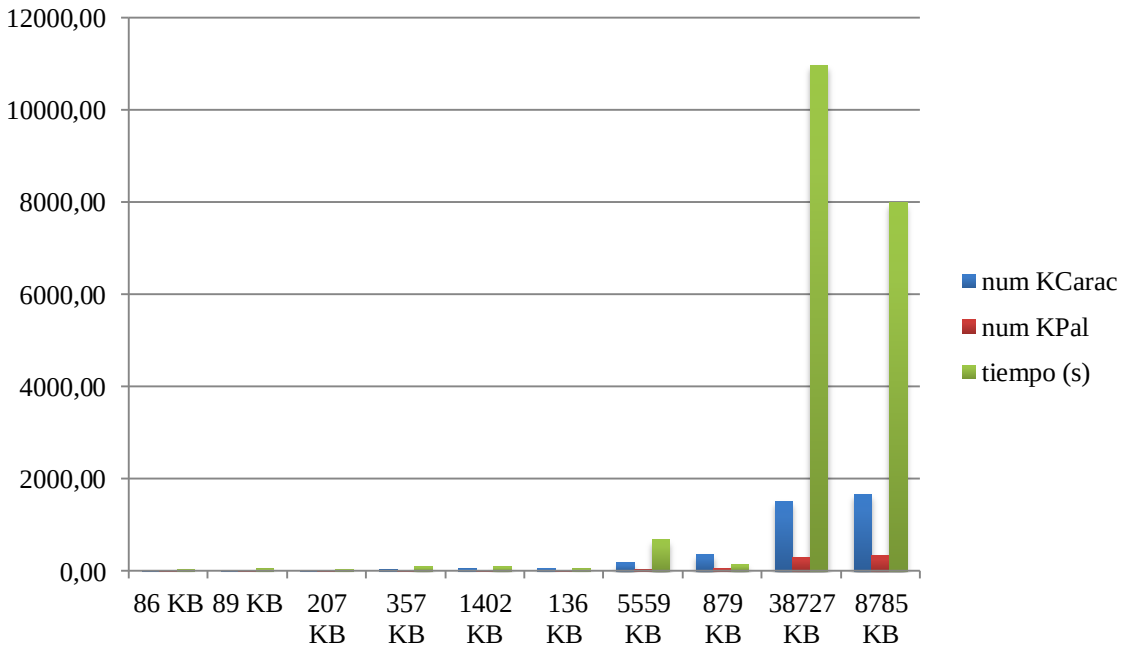


Figura 32 Gráfica 1

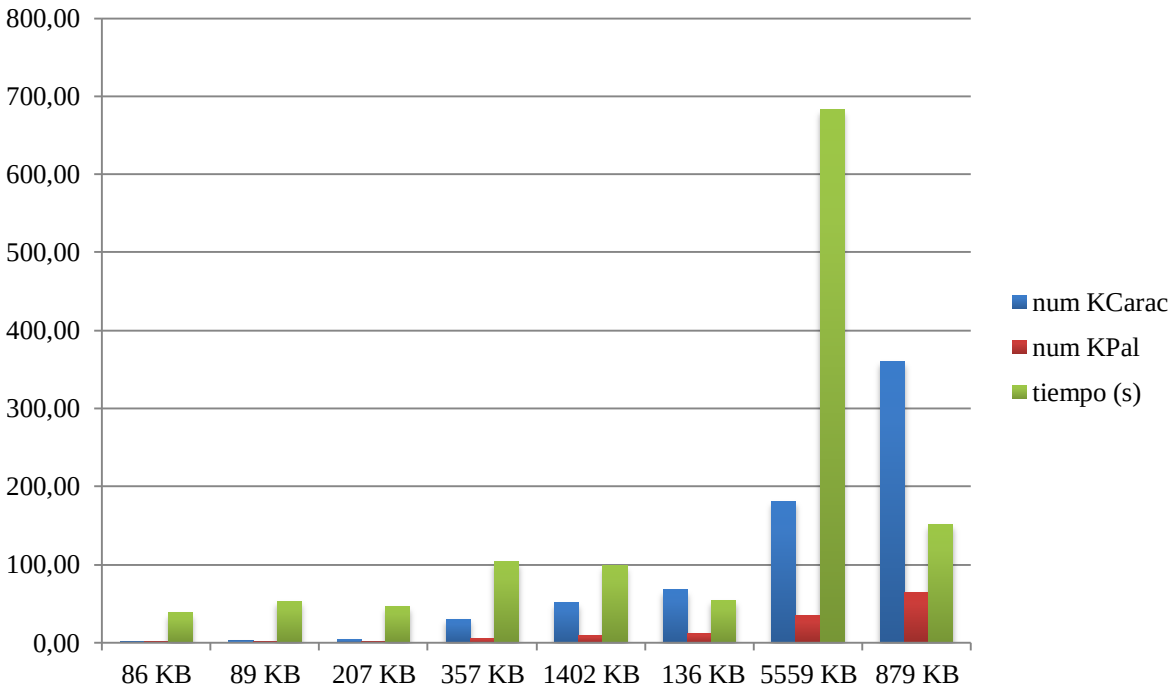


Figura 33 Gráfica 2

Se han realizado estas dos gráficas para representar la tabla que recoge los datos de número de caracteres, número de palabras y tiempo de ejecución de cada archivo.

Como podemos comprobar, en la primera gráfica los archivos con un número mayor de caracteres requieren de un tiempo de ejecución mayor pese a que el número de palabras sea parecido, con lo cual tenemos que tener en cuenta que a mayor número de caracteres que contengan los archivos, su tiempo de ejecución crece considerablemente.

El tamaño del archivo también parece que sea importante a la hora de medir el tiempo pero puede ocurrir que el tamaño aumento por elementos que hagan que pese más, como imágenes, tablas, etc… y no tengan nada que ver en los requisitos que se están estudiando. Pese a ello, al tener que trabajar con un archivo más pesado estos afecta al tiempo de ejecución.

La segunda gráfica nos da la misma información que la primera, pero se ha acotado en función al tiempo puesto que existen algunos valores que disparan la escala de la gráfica y no podríamos ver claramente los resultados.

Podemos comprobar que existe un valor que difiere del resto respecto al tiempo de ejecución. Como hemos comentado antes esto puede ser por la carga de imágenes, gráficos u otros elementos que ocupan espacio pero no sirven a la hora de realizar el recuento ni lectura de las palabras pero si influyen a la hora de "mover" el archivo.

También hemos realizado un estudio obteniendo las medias, mínimos, máximos y desviaciones típicas de varios archivos. Los datos han sido obtenidos después de realizar 64 ejecuciones de cada archivo.

	archivo FlexiScrum	archivo LegalNotice	archivo programa3	archivo solitari	archivo Hardvar
Tiempo medio análisis	682,91	54,06	296,25	53,00	103,41
Tiempo mínimo	615,00	49,00	277,00	47,00	95,00
Tiempo máximo	1002,00	69,00	377,00	94,00	199,00
Desviación estándar	82,05	4,70	15,73	6,86	12,43
Numero kCarar	180627,00	67426,00	31204,00	2268,00	30081,00

Figura 34 Tabla datos de 2

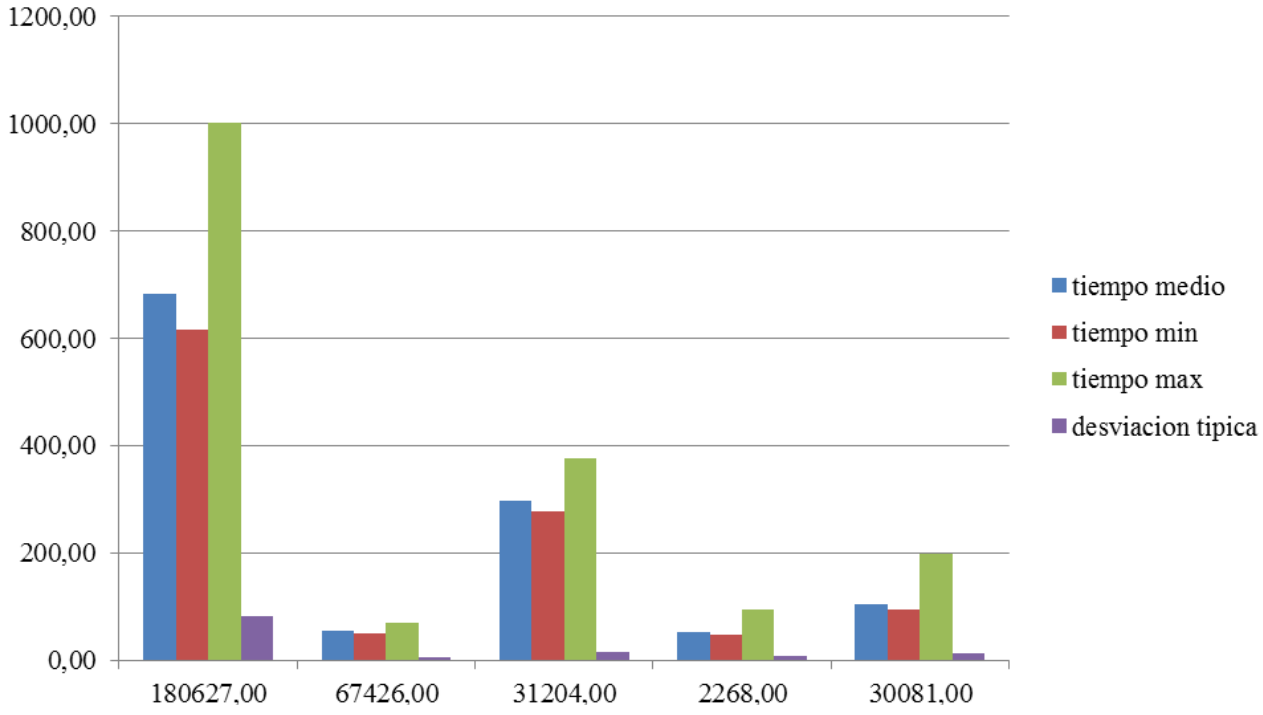


Figura 35 Gráfico 3

Como podemos observar tanto en la tabla de datos y en el gráfico existe una relación directa entre lo que tarda en ejecutarse pero sí que existen algunos valores que aunque

tengan un numero notablemente diferente de caracteres obtienen prácticamente el mismo tiempo de ejecución. Como hemos comentado anteriormente esto puede ser porque a aparte de lo que vendrían siendo las palabras existen otros elementos que cargan de peso el archivo pero no forman parte de las características que estudiamos a la hora de realizar el estudio.

También cabe notar que cuanto mayor es el número de caracteres mayor es la desviación típica de los mismos, esto quiere decir en este tipo de archivos es cuando mayor diferencia de tiempo de ejecución encontraremos.

Claramente podemos decir que existe una relación directa entre el tamaño del archivo que se va a estudiar con el tiempo que tardan en ejecutarse las tareas implementadas pero también tenemos que tener en cuenta que a veces el número de caracteres que contiene y tiempo de ejecución no tienen relación puesto que el tamaño de un archivo viene dado también por los múltiples elementos multimedia que puede contener dentro de sí.

CAPÍTULO 7: CONCLUSIONES

El objetivo principal de este proyecto era el de realizar una aplicación que pudiese analizar textos en PDF y plasmar la información principal de cada uno a los usuarios. El objetivo se ha visto cumplido y puesto que se ha realizado una aplicación para ordenadores que pudiese llevar a cabo estas acciones.

Una vez más, se ha visto la importancia de tener claros los requisitos principales de la aplicación, que pequeños cambios en los mismos pueden acarrear importantes trastornos a lo planteado inicialmente y que la estructuración de las etapas de desarrollo juega un papel importante en ello. Se ha aprendido a administrar bien los recursos de los que se dispone y a compaginarlos con los que ya se tienen anteriormente.

Se han visto ampliados los conocimientos acerca del lenguaje de programación que hemos utilizado, C#, y más en concreto todo lo relacionado con los archivos en PDF, dando un importante énfasis al apartado relacionado con el manejo de las anotaciones y etiquetas puesto que ha sido una de las partes más difíciles de llevar a cabo.

Respecto al entorno de desarrollo utilizado, Visual Studio 2010, no ha habido nuevos hallazgos respecto con anteriores versiones pero lo que sabíamos anteriormente se ha visto reafirmado. Es una buena herramienta de trabajo para este tipo de aplicaciones desarrolladas con C#.

CAPÍTULO 8: LÍNEAS FUTURAS

Teniendo en cuenta que esta aplicación está dirigida al usuario y el mismo interactúa mediante las interfaces con la misma, todas las mejores relacionadas con la mejora de las interfaces entrarán dentro de las líneas futuras. Por ejemplo, se podrían utilizar técnicas y aplicaciones que mejoren la apariencia de botones, texto, etc… para que sean más amigables para el usuario.

Otro punto a tener en cuenta es el que engloba lo relacionado con el OCR. La librería iTextSharp que hemos utilizado está limitada en uso de este recurso de reconocimiento óptico de objetos. Hemos intentado buscar librerías libres para Visual Studio que contuviesen esta herramienta pero nos ha sido imposible encontrarlo. Hemos encontrado que el uso de estas librerías esta también relacionado con aspectos de la instalación del paquete Office en el ordenador pero aun cambiando diversos puntos en el proceso de instalación no se ha conseguido obtener ningún resultado positivo.

Existen programa de servicios Web de pago que proveen de diversas herramientas para poder llevar a cabo el uso de OCR, asique este tipo de solución puede ser tenida en cuenta para futuras mejoras en la aplicación.

Puesto que cuando se comienza a utilizar y se da un uso continuado de las cosas es cuando más ideas de mejora aparecen, cualquiera de estas ideas que el usuario tenga debe ser escuchada y tenida en cuenta para que el uso de la aplicación sea de lo más satisfactoria para él.

BIBLIOGRAFÍA

[1] http://es.wikipedia.org/wiki/PDF

[2] http://www.adobe.com/es/products/acrobat/adobepdf.html

[3] http://es.wikipedia.org/wiki/C_Sharp

[4] http://msdn.microsoft.com/es-es/library/w2a9a9s3(v=vs.80).aspx

- http://what-when-how.com/itext-5/creating-annotations-itext-5/
- http://itextpdf.com/book/digitalsignatures20120920.pdf
- http://msdn.microsoft.com/es-es/library/ms228504(v=vs.80).aspx
- http://www.itextpdf.com/itext.php
- http://sourceforge.net/projects/itext/
- http://get.adobe.com/es/reader/

ANÁLISIS DE TEXTOS EN PDF

Leticia Osácar Landa

Ingeniería Informática

Abril 2013, Pamplona

Introducción

- OBJETIVO DEL PROYECTO:
  - Desarrollar aplicación de escritorio para el análisis de textos en archivos PDF.
  - Ver resultados en una interfaz.

- ETAPAS:
  - Fase 1: Tutoriales relacionados con archivos PDF.
  - Fase 2: Implementación funciones. Base de datos.
  - Fase 3: Definición y diseño de interfaz.

Introducción

- ¿Qué es un archivo .PDF?
  - Formato de almacenamiento de documentos digitales multiplataforma.
  - Elementos de texto, multimedia, vínculos…
  - Formato más utilizado para el intercambio de documentos en Internet.
  - Cifrado para proteger contenido.
  - Elemento clave en el proyecto:

    Permite búsquedas de palabras o expresiones.

Introducción

- Elementos clave en el proyecto: ETIQUETAS / ANOTACIONES.
- Usuario añade comentarios.
- Notas adhesivas o resaltar texto.
- Nuestro caso:

  palabra encontrada -> añado comentario

Herramientas de programación

- Visual Studio 2010 Professional.
- Lenguaje C#.
- Sentencias SQL.
- Librerías PDF: iTextSharp.

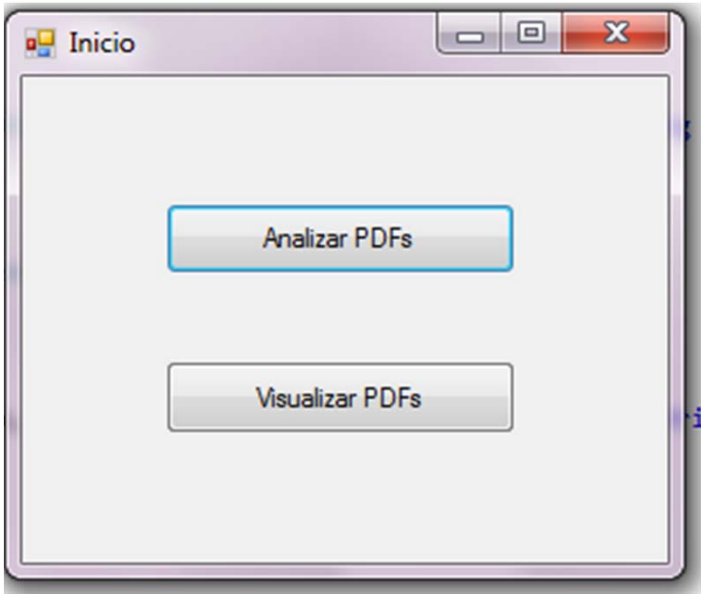
Librería iTextSharp

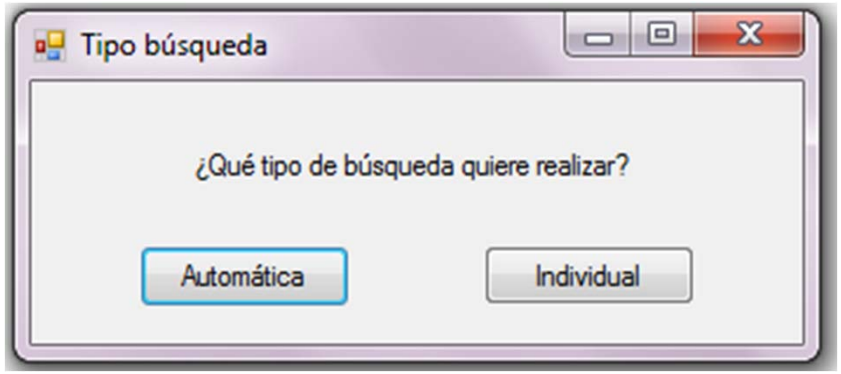
- Librería Open Source.
- Basado en lenguaje Java (iText).
- API para permitir:
  - Documentos e informes basados en XML y bases de datos.
  - Rellenar formularios interactivos.
  - Añadir características a documentos PDF existentes.
  - Concatenar y dividir páginas en archivos PDF.

Librería iTextSharp

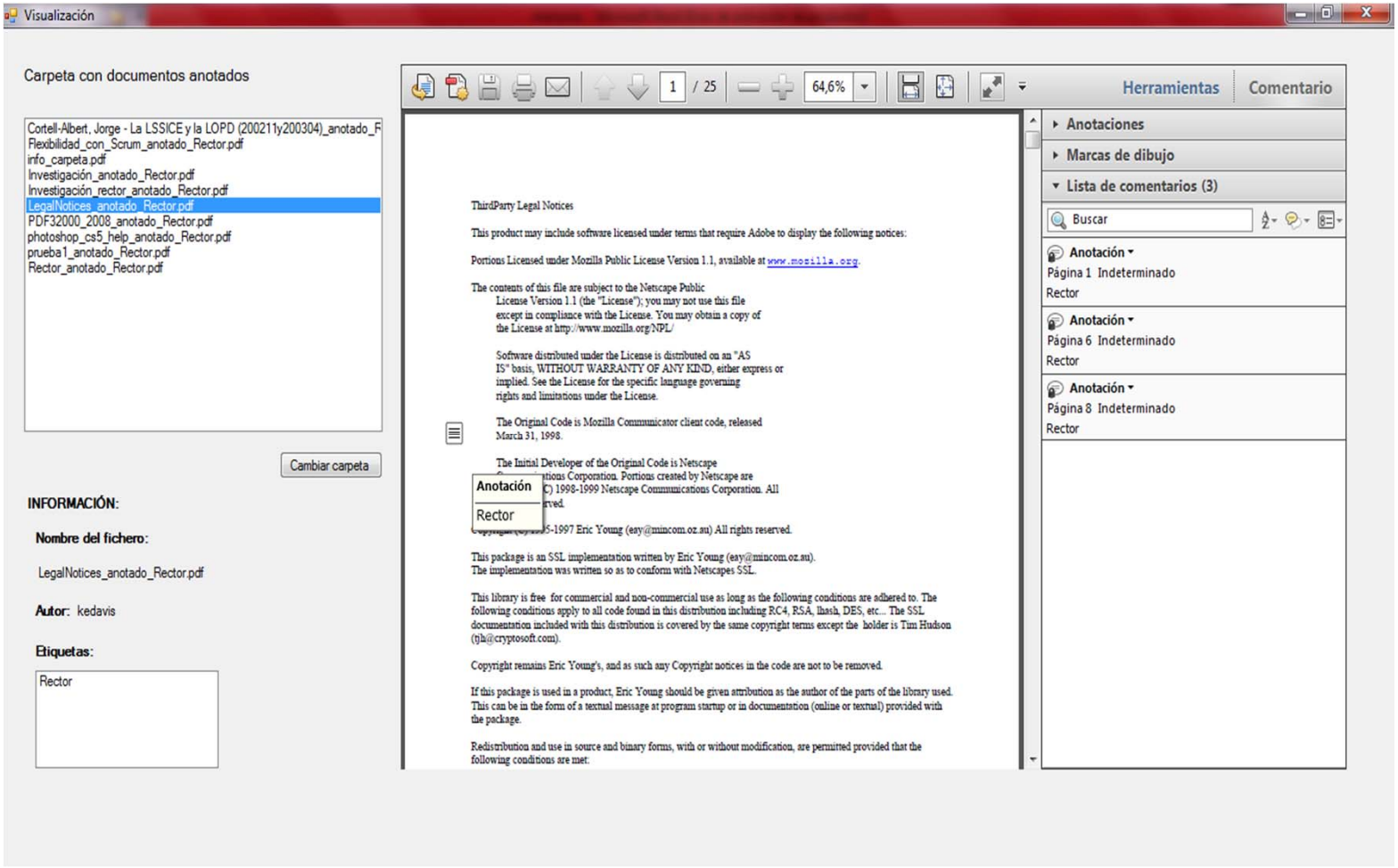
- Lectura, creación y actualización.
  - PdfReader, Document, PdfWriter, PdfStamper…
- Firmar y encriptar.
- iTextSharp no realiza OCR.

Funcionalidades del sistema





Funcionalidades del sistema



Requisitos del sistema y hardware

- Uso del lenguaje C# para la implementación.
- Windows 7.
- Visual Studio 2010 Professional.
- Requerimientos técnicos:
  - Ordenador de mesa.

Diagrama de casos de uso

- Actor: Usuario.
- Descripción:
  - Entrar aplicación.
  - Terminar aplicación.
  - Análisis de textos:
    - Análisis de archivos.
    - PDF informativo.
    - Visualizar PDF.
    - Información PDF.

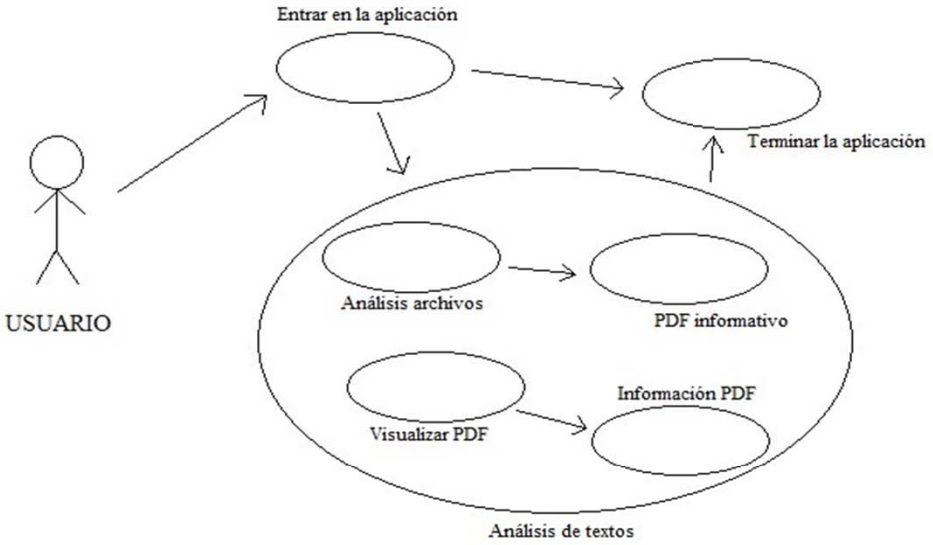
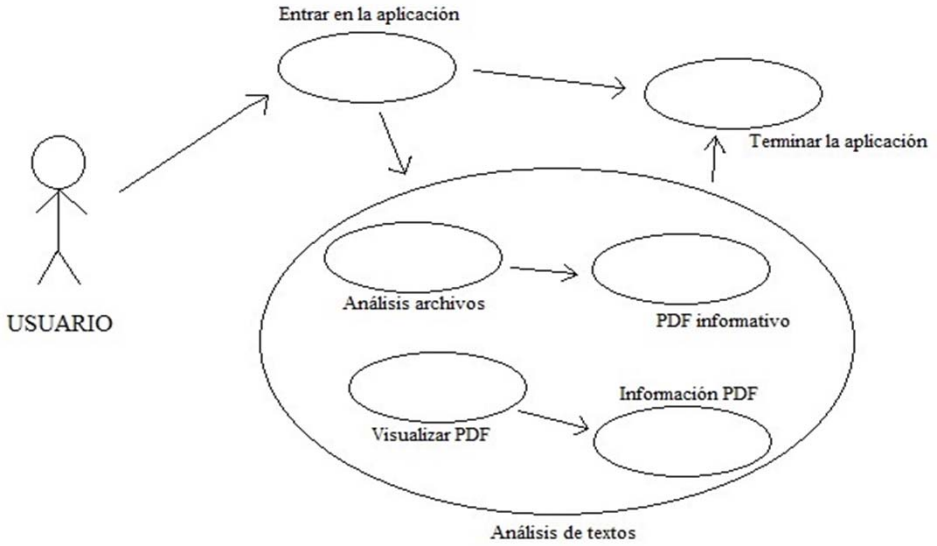


Diagrama de casos de uso

- Precondiciones:
  - Conexión base de datos.
  - Tener docs en PDF.
- Poscondiciones:
  - Análisis de textos completado.
  - Documentos anotadc
- Flujos alternativos:
  - Errores bases de datos.
  - Errores en docs PDF.



Esquema Entidad - Relación

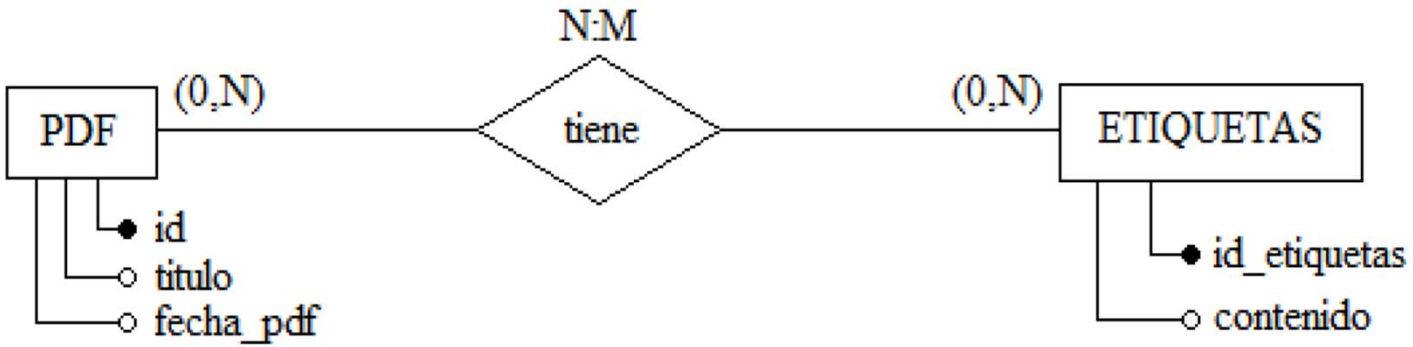


Diagrama de secuencia
Analizar PDFs automáticamente

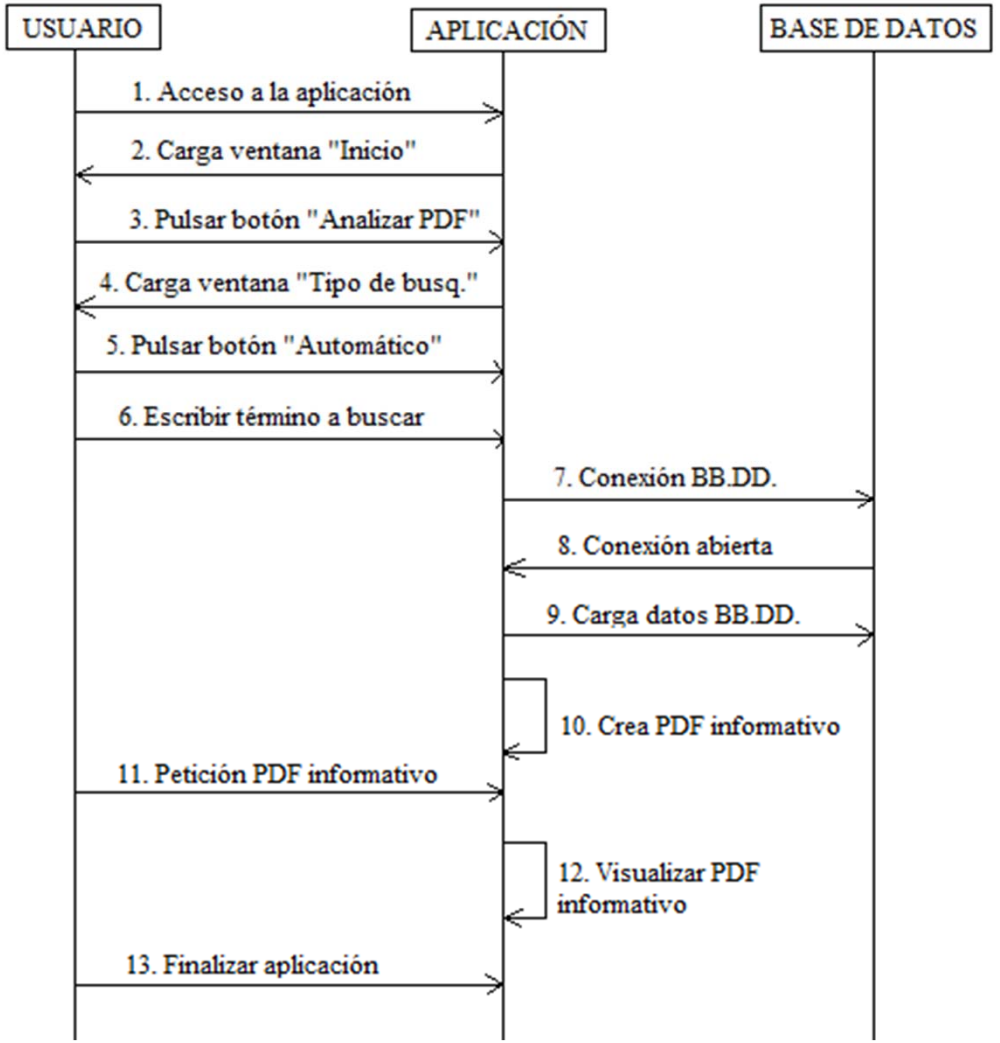


Diagrama de secuencia Analizar PDFs individualmente

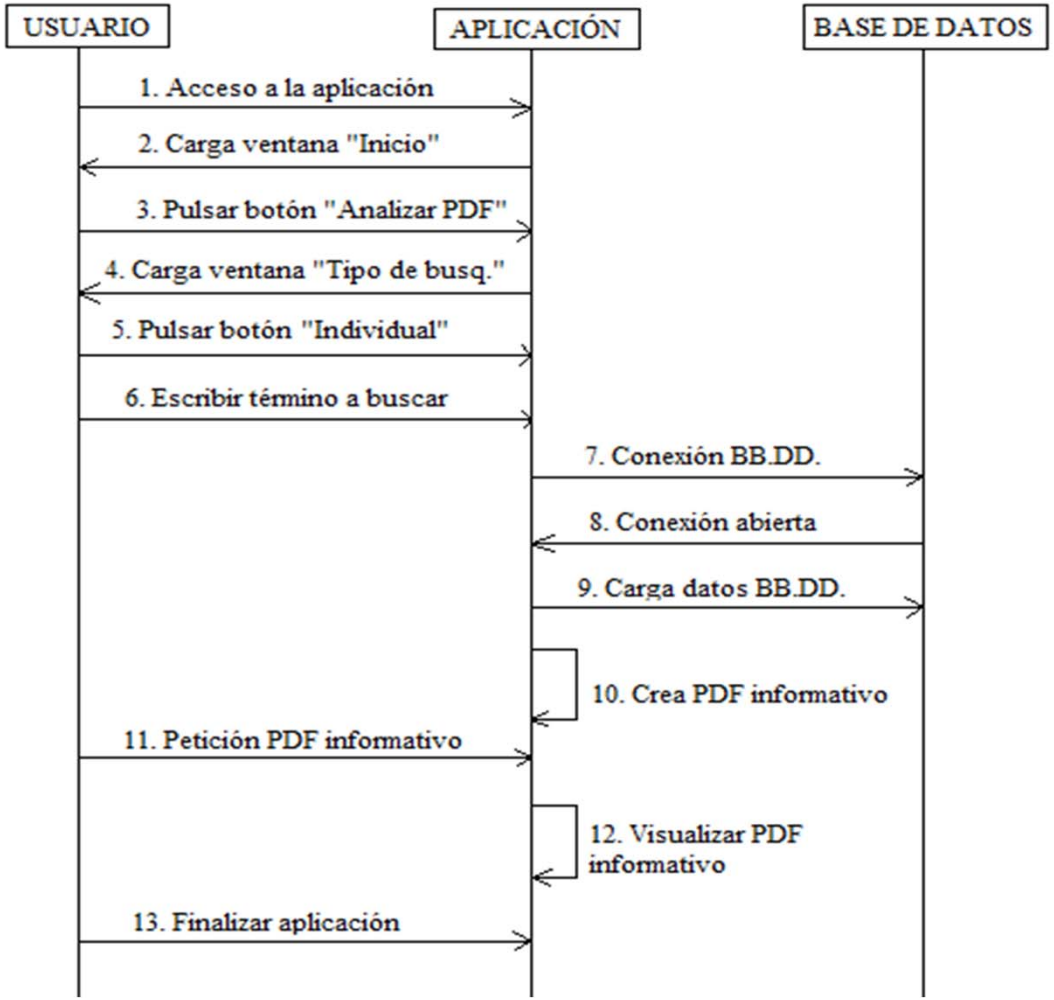


Diagrama de secuencia Visualización de PDFs

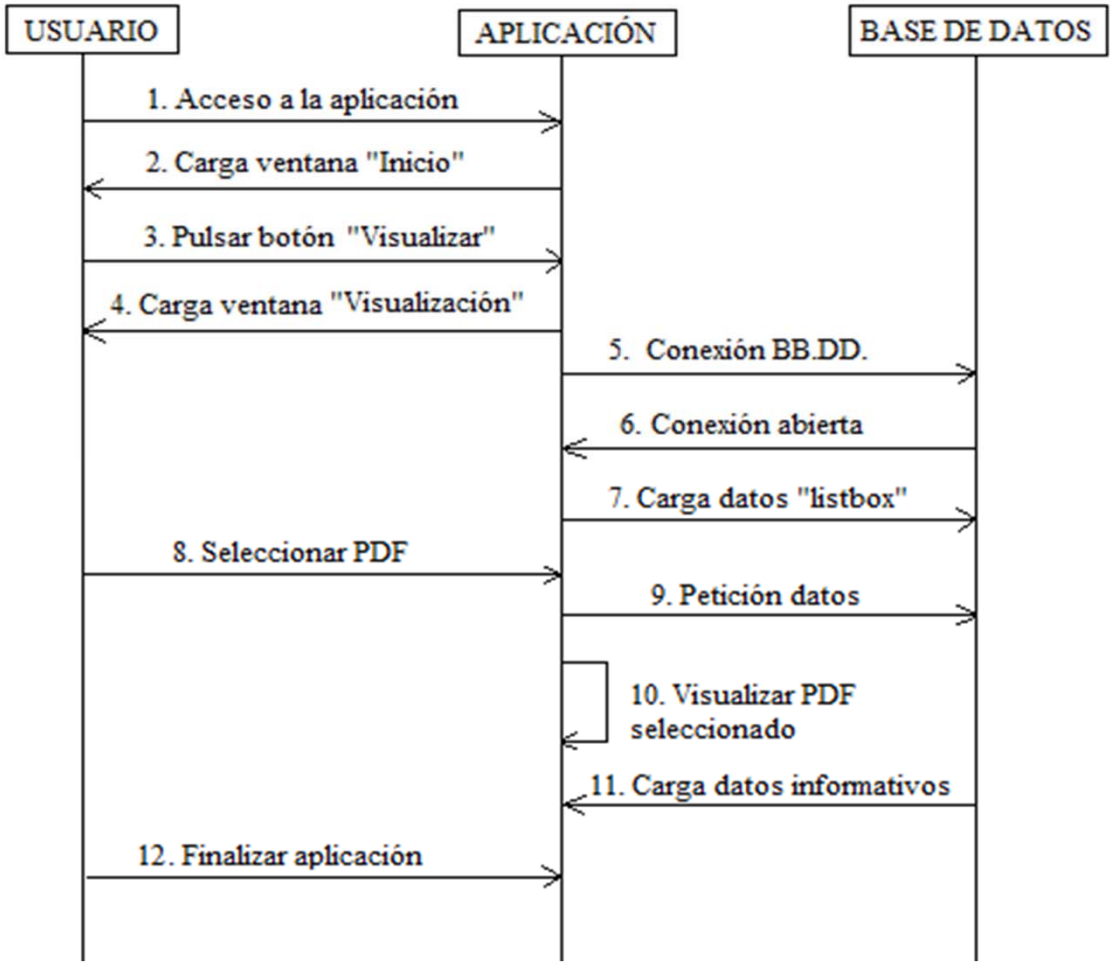
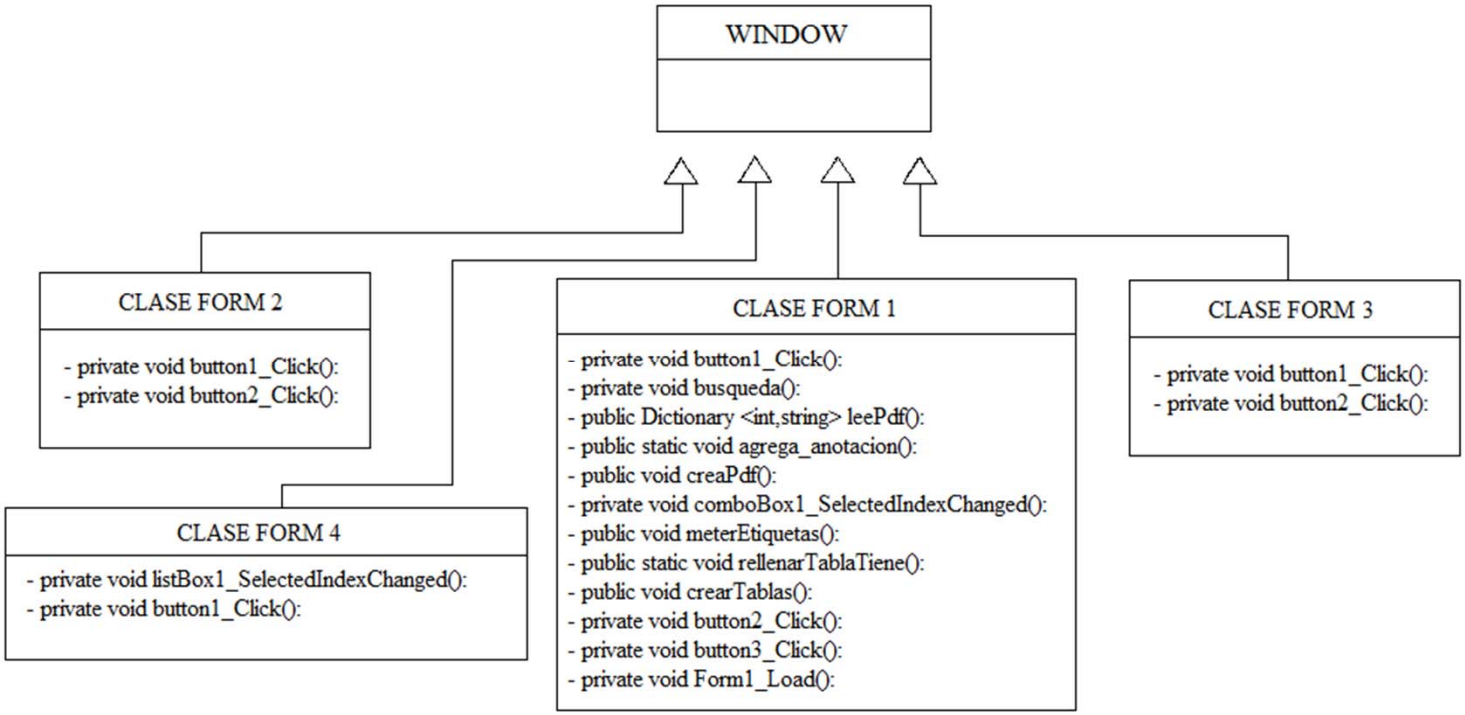
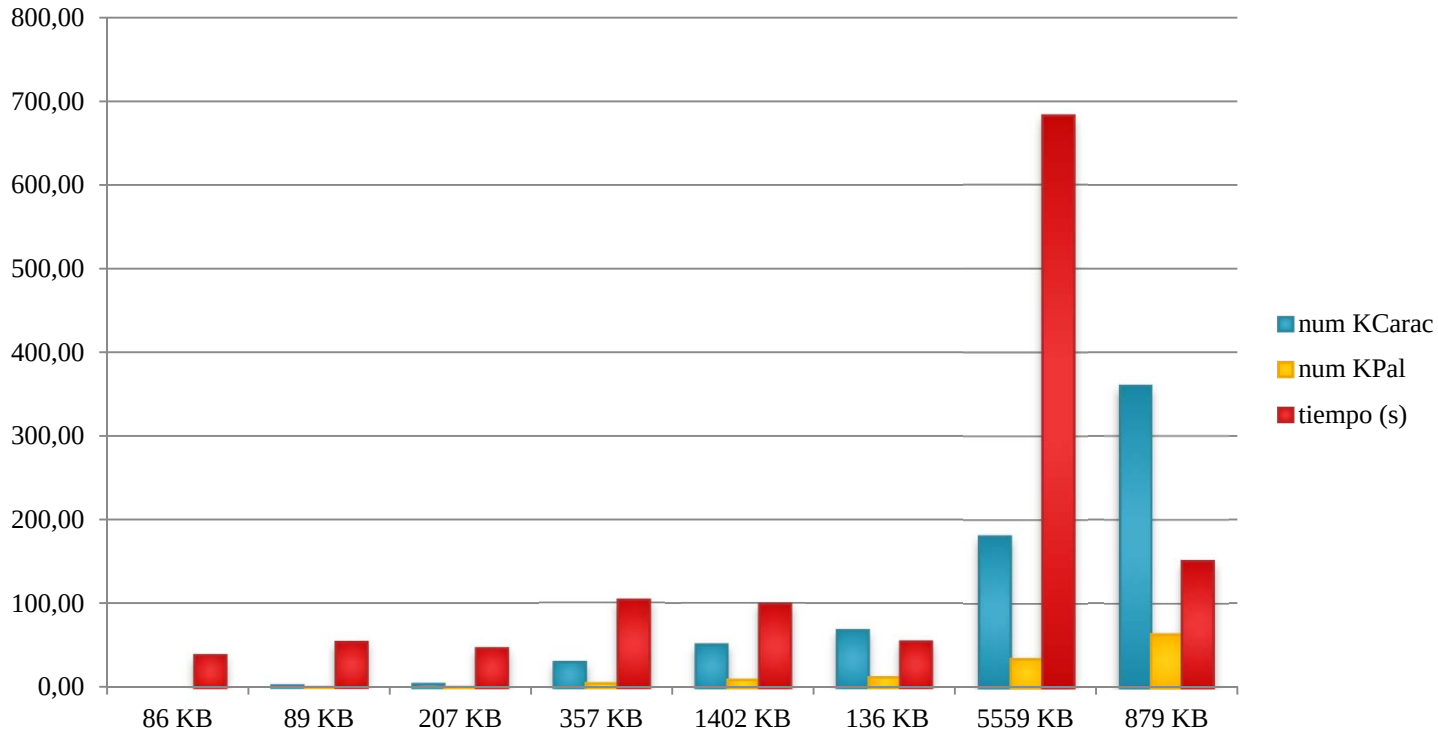


Diagrama de clases



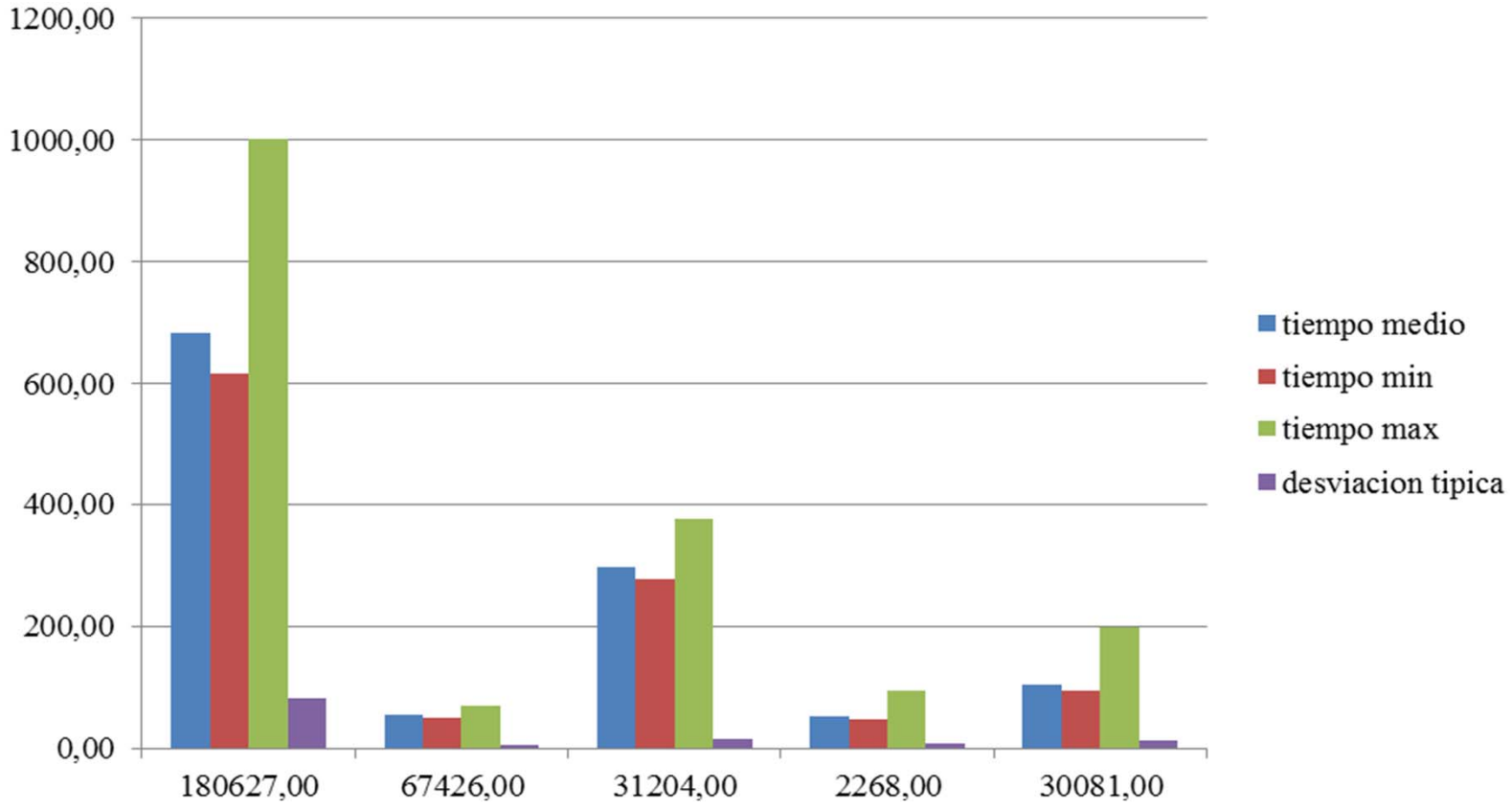
Pruebas de rendimiento

- Tiempo de ejecución en función de el tamaño del archivo, número de caracteres y palabras.



Pruebas de rendimiento

- Tiempo medio, mínimo, máximo y desviación estándar en función del número de caracteres.



Conclusiones

- Análisis de textos y visualización de la información obtenida.
- Ampliación de lenguaje C#.
- Herramienta Visual Studio 2010.
- Organización del tiempo de trabajo.

Líneas futuras

- Mejora apariencia de la interfaz.
- Inclusión de elemento OCR.
  - Webs de pago.