Universidad Rafael Landívar

Facultad de Ingeniera

Ingeniería en informática y sistemas

Manejo e implementación de archivos

Inge: Melissa Gonzales

Lector de metadatos avanzado para archivos PDF

Ángel Santiago Urbina Lopez - 1546122

Oscar Alejandro Luna Ordoñez - 1530622

Quetzaltenango, Guatemala

24/10/2023

Introduccion

La información contenida en documentos PDF es una fuente invaluable de conocimiento y datos en la actualidad. Sin embargo, la capacidad de extraer y recopilar información de estos archivos de manera eficiente puede ser un desafío. En este contexto, nos complace presentar nuestro proyecto, una aplicación Java diseñada específicamente para abordar la necesidad de recopilar datos desde archivos PDF de manera sistemática y efectiva.

Nuestra aplicación se centra en el procesamiento y extracción de datos desde archivos PDF, ofreciendo una solución ágil y altamente personalizable. Utilizando el poderoso lenguaje de programación Java, hemos creado una herramienta que permite a los usuarios definir reglas de extracción, realizar búsquedas avanzadas y automatizar la recopilación de datos de manera precisa.

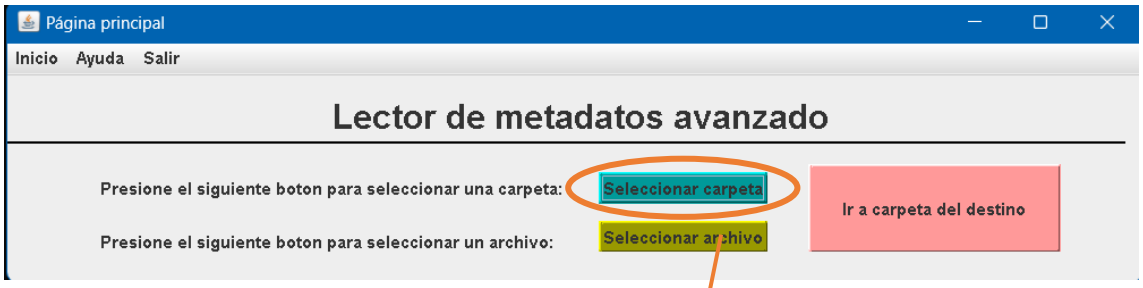
A lo largo de este proyecto, exploraremos en detalle las características de nuestra aplicación, así como los aspectos técnicos que hacen posible la extracción de datos desde archivos PDF. Presentaremos ejemplos prácticos que ilustrarán cómo esta herramienta puede ser empleada en una variedad de escenarios, desde la extracción de datos en informes y documentos académicos hasta la automatización de tareas de procesamiento de documentos.

En un mundo donde los datos son esenciales para la toma de decisiones y la generación de conocimiento, nuestra aplicación se erige como una solución valiosa para aquellos que desean aprovechar la información contenida en archivos PDF de manera eficiente. Esperamos que este proyecto pueda satisfacer las necesidades de investigadores, profesionales y entusiastas de la información que buscan simplificar y optimizar el proceso de recopilación de datos desde archivos PDF, brindando una experiencia más fluida y productiva.

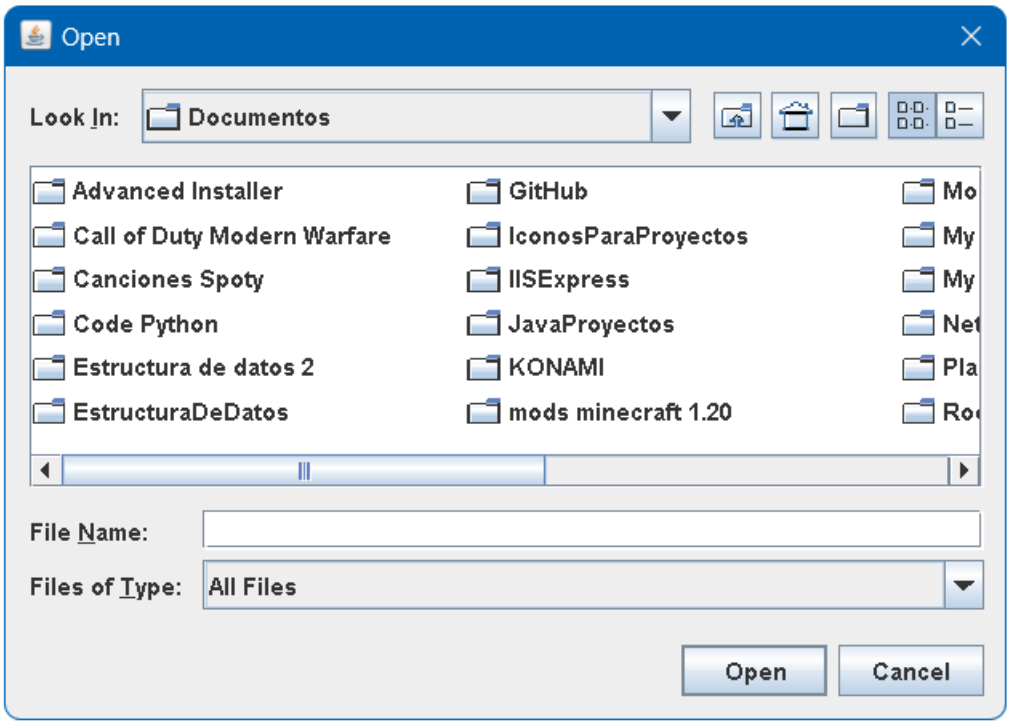
Manual de uso de la aplicación

A primera vista, cuando iniciamos la aplicación podremos ver un menú muy simple y sencillo, esto se hizo con la intención que sea fácil de entender la función de este programa. Como podemos observar hay dos opciones, la primera para seleccionar una carpeta y la segunda solo para seleccionar un archivo en específico. La primera es para leer los datos de todos los archivos PDFS que se encuentren en una carpeta y la segunda para leer los datos de un archivo PDF en específico.

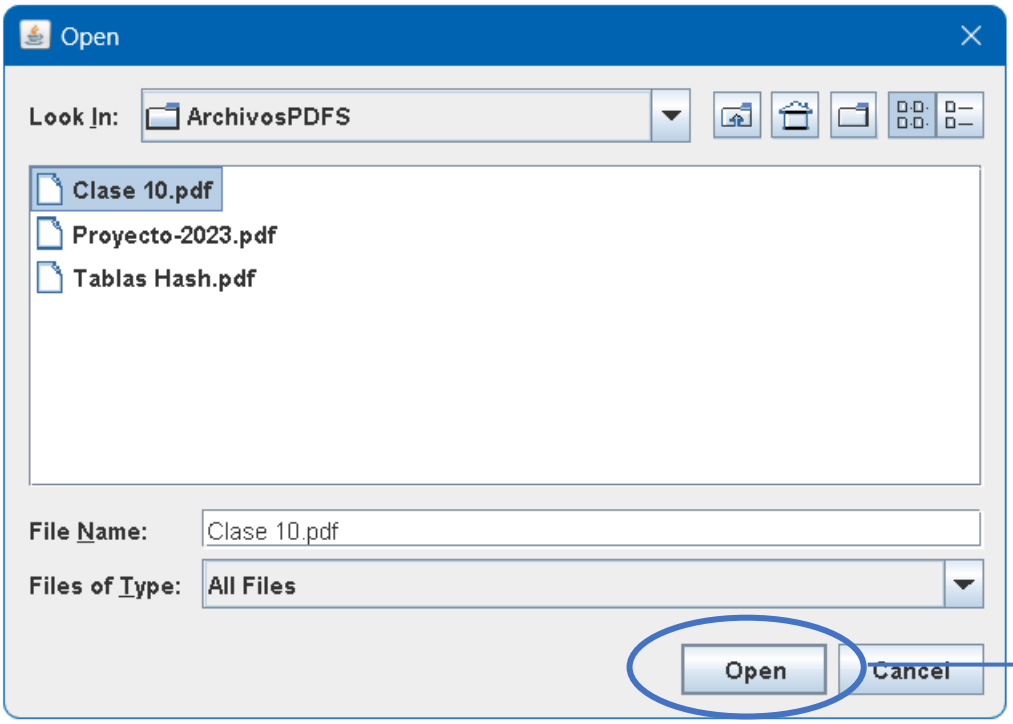
Lector de archivos para una carpeta:



Al momento de presionar este botón se nos abrirá el siguiente explorador de archivos



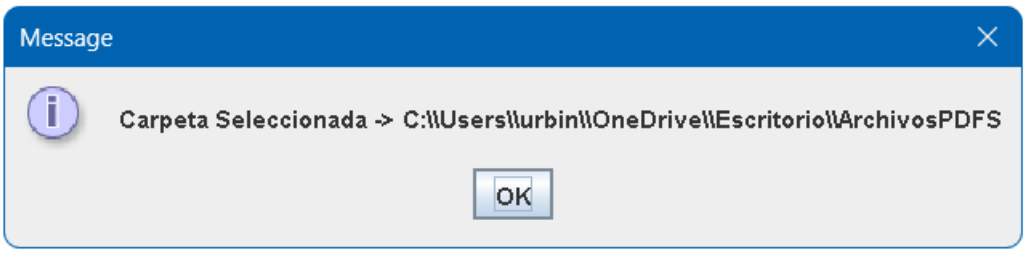
En este explorador lo único que debemos de realizar es hacer la búsqueda de la carpeta que nosotros queramos leer.



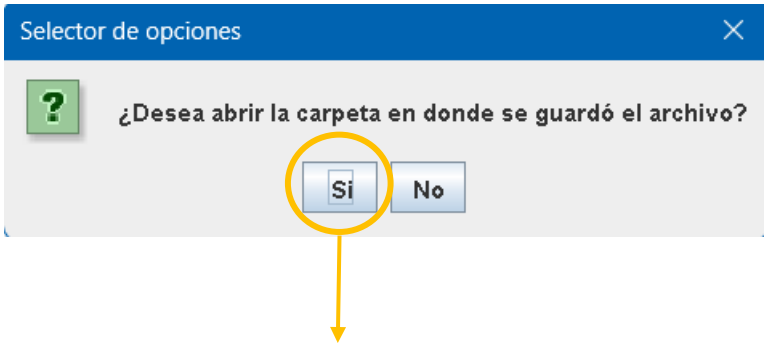
Cuando tengamos la carpeta seleccionada (Como en el caso de la imagen de arriba) lo único que tendremos que hacer es apretar el botón “Open” y ya el resto lo hará la aplicación.

Botón “Open” para seleccionar la carpeta y que esta lea los datos de los PDFS.

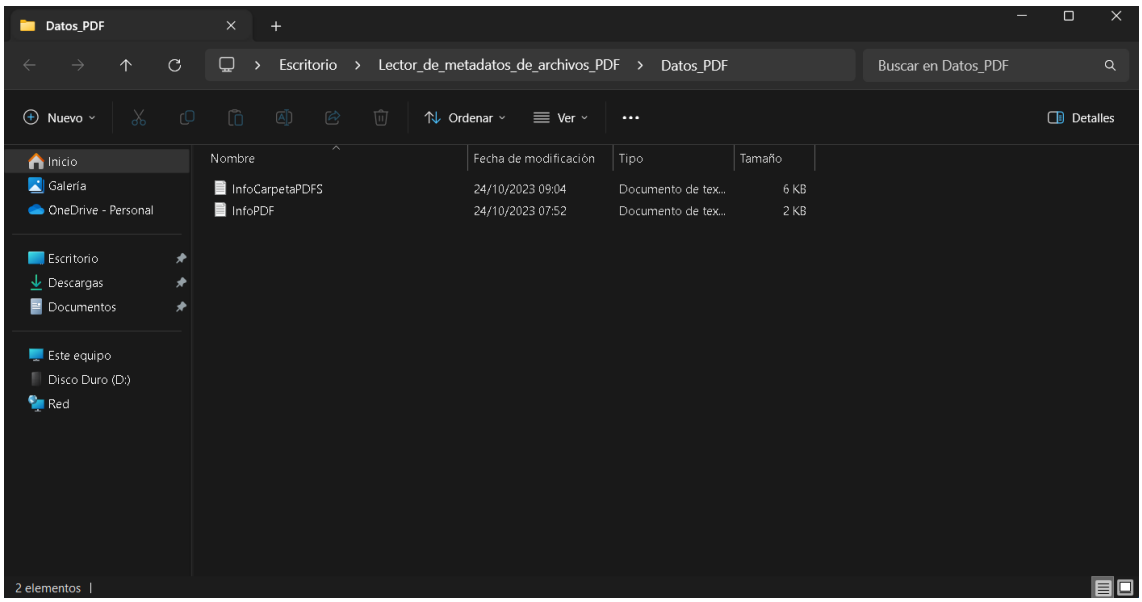
Cuando le demos click al botón de “Open” nos aparecerá el siguiente mensaje

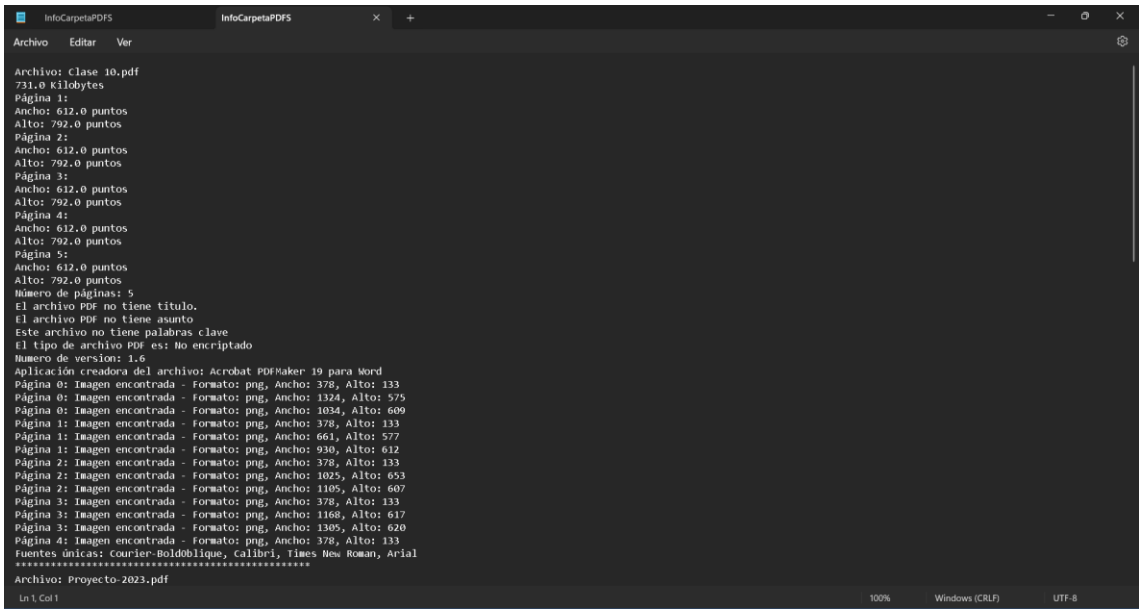


Este mensaje nos dirá la carpeta que seleccionamos, si es la correcta apachamos “Ok” y nos generara nuestro archivo de texto con la información de todos los PDFS. También nos mostrara el siguiente mensaje



Si apretamos el siguiente botón pasara lo siguiente, nos direcciona directamente hacia donde esta la carpeta con los archivos .txt con la información de los PDF.

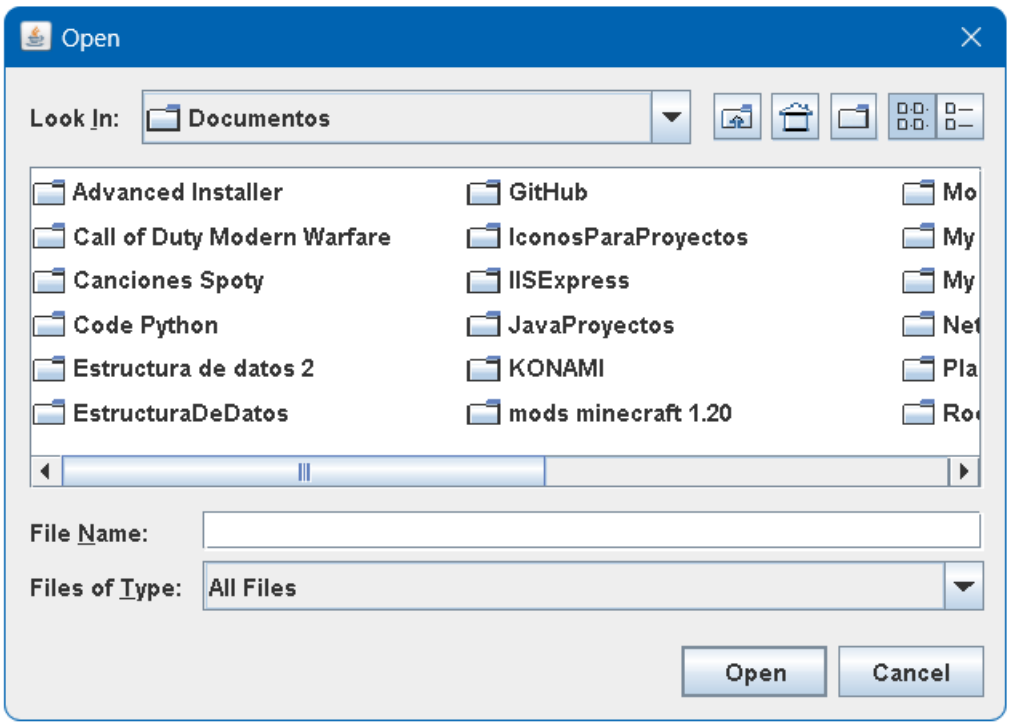




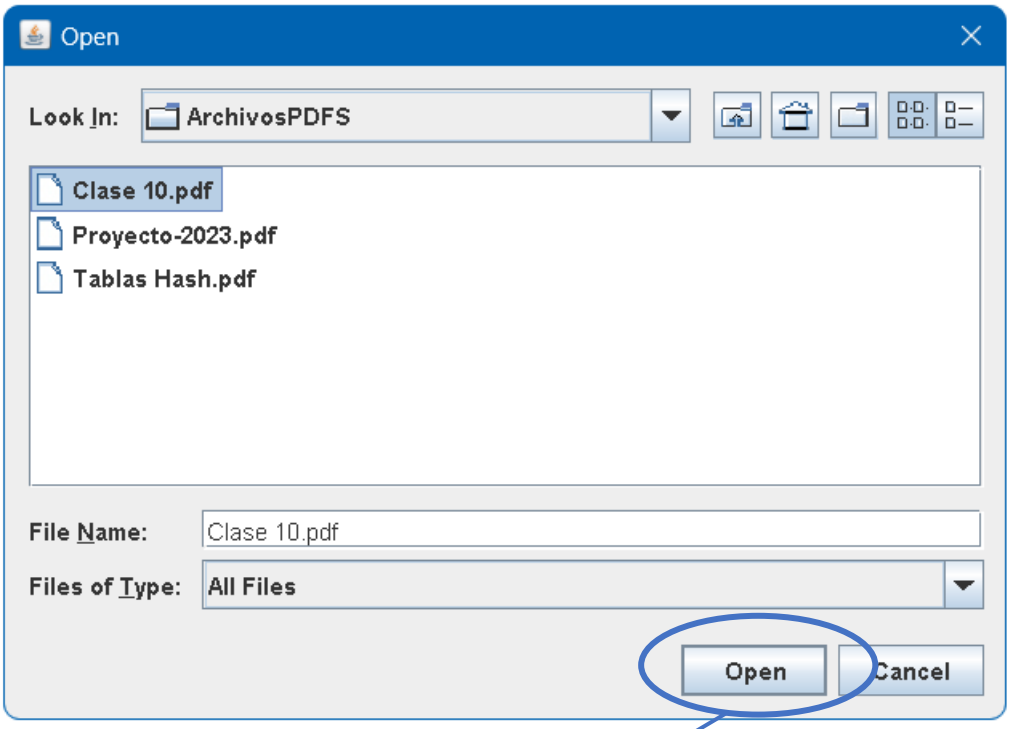
Ya por ultimo seleccionamos el archivo “InfoCarpetasPDF” y como podemos ver en la imagen, se crea el archivo de texto con toda la información solicitada.

Lector de archivos para un archivo en específico:

Al momento de presionar este botón se nos abrirá el siguiente explorador de archivos



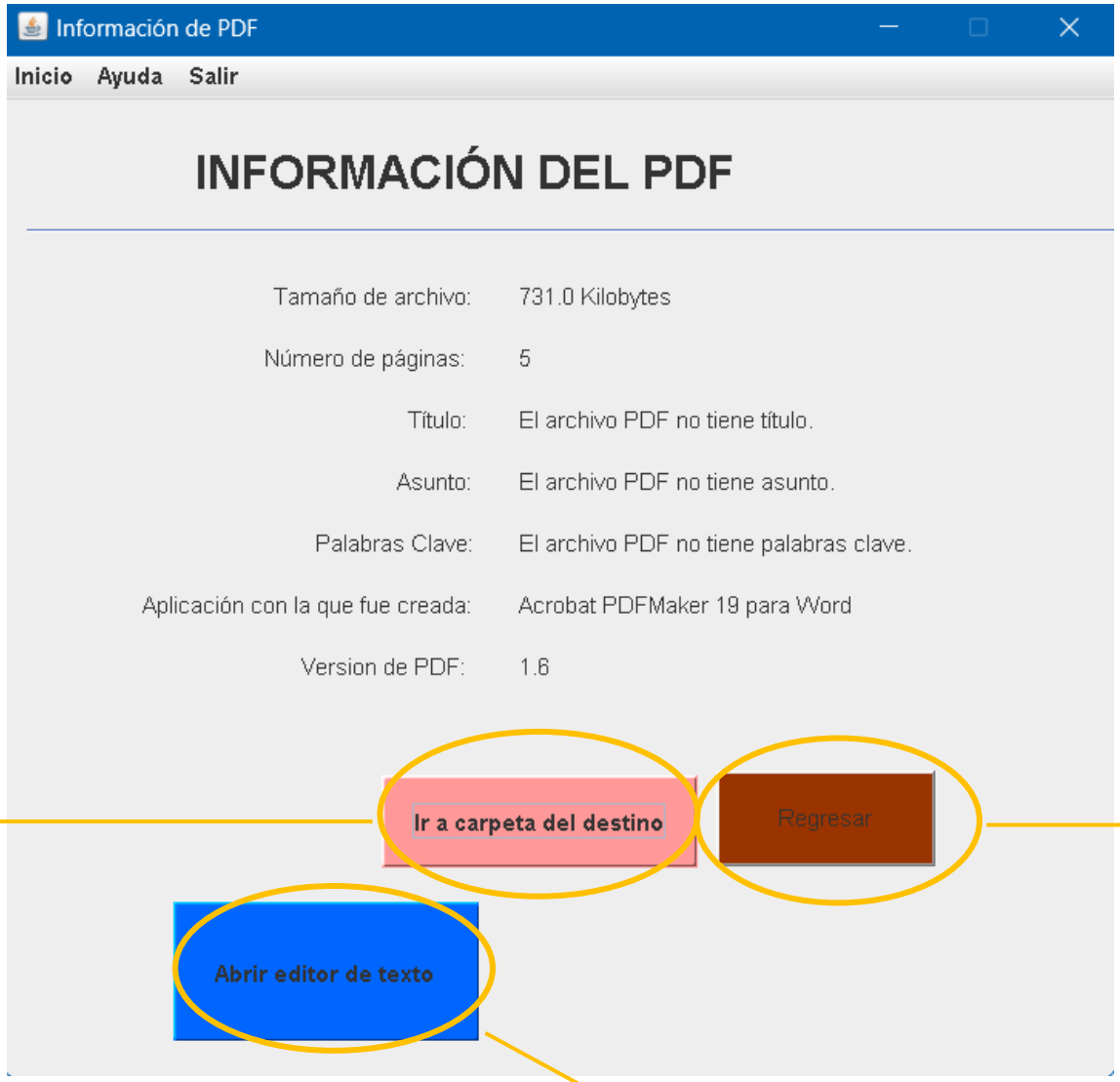
En este explorador lo único que debemos de realizar es hacer la búsqueda del archivo que nosotros queramos leer.



Botón “Open” para seleccionar el archivo y que este lea los datos de los PDFS.

Cuando tengamos el archivo PDF seleccionado (Como en el caso de la imagen de arriba) lo único que tendremos que hacer es apretar el botón “Open” y ya el resto lo hará la aplicación.

Cuando le demos click al botón open se nos abrirá la siguiente ventana

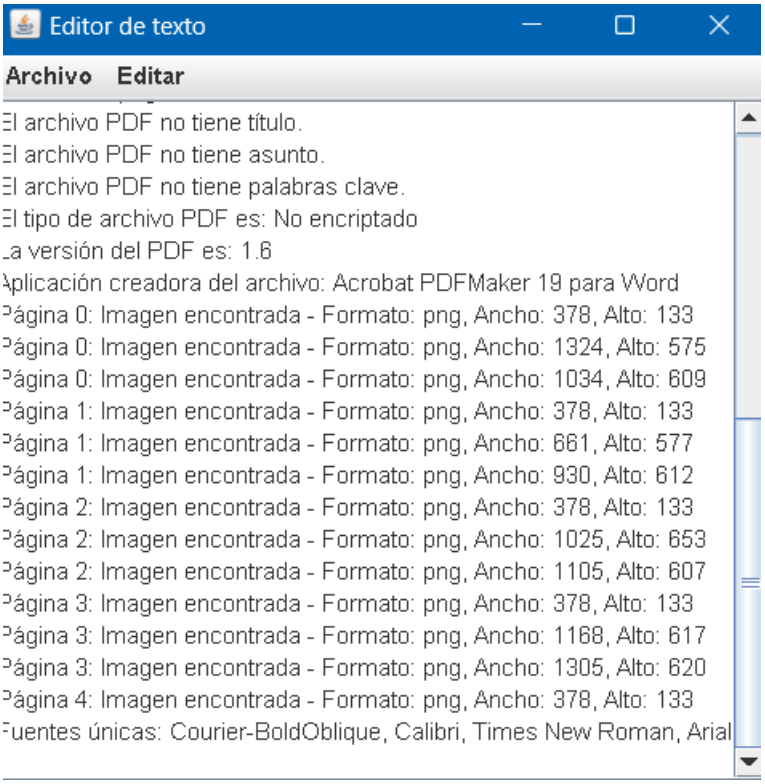


Boton “Abrir editor de texto” este nos sirve para abrir en donde podemos editar.

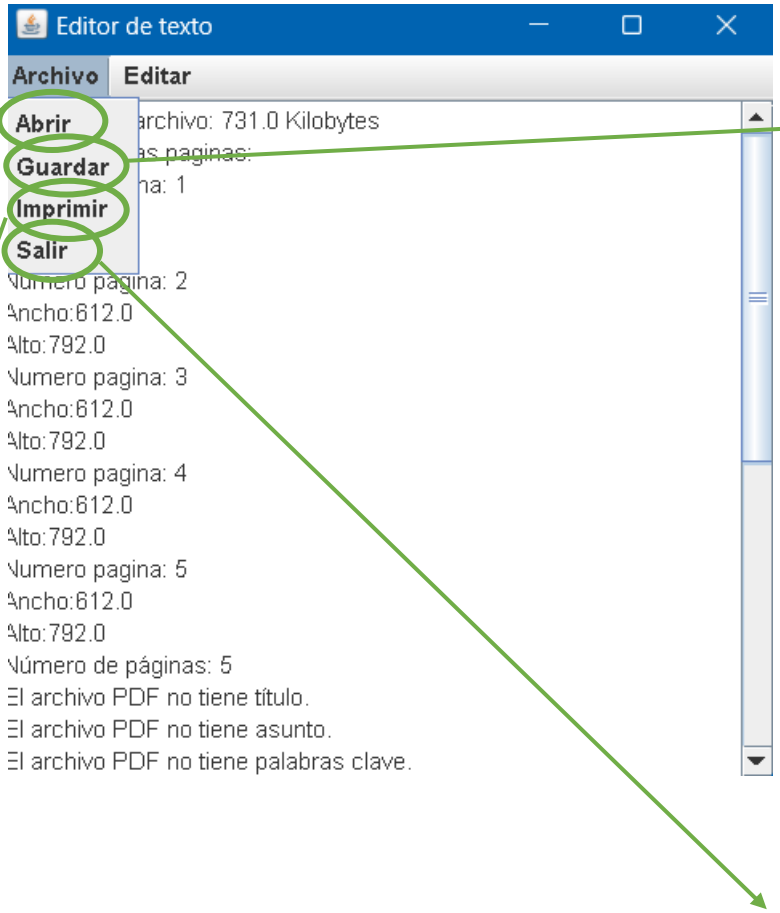
Botón “Regresar”, este nos regresara al menú principal al mismo tiempo que crea el archivo de texto con la información del PDF.

Botón “Ir a la carpeta del destino”, este nos abrirá nuestro explorador de archivos y nos abrirá la carpeta en donde están los archivos .txt para poder abrirlos por si deseamos.

Si apretamos el botón “Abrir editor de texto” nos abrirá la siguiente ventana



Nos abrirá un ventana nueva en donde esta toda la información del archivo .txt, para editarla simplemente tenemos que escribir lo que queramos agregar o borrar lo que deseamos quitar.



Podemos abrir otro archivo .txt

Podemos imprimir el archivo .txt

Apretamos este botón cuando terminemos de editar lo que deseamos para así poder guardarlo

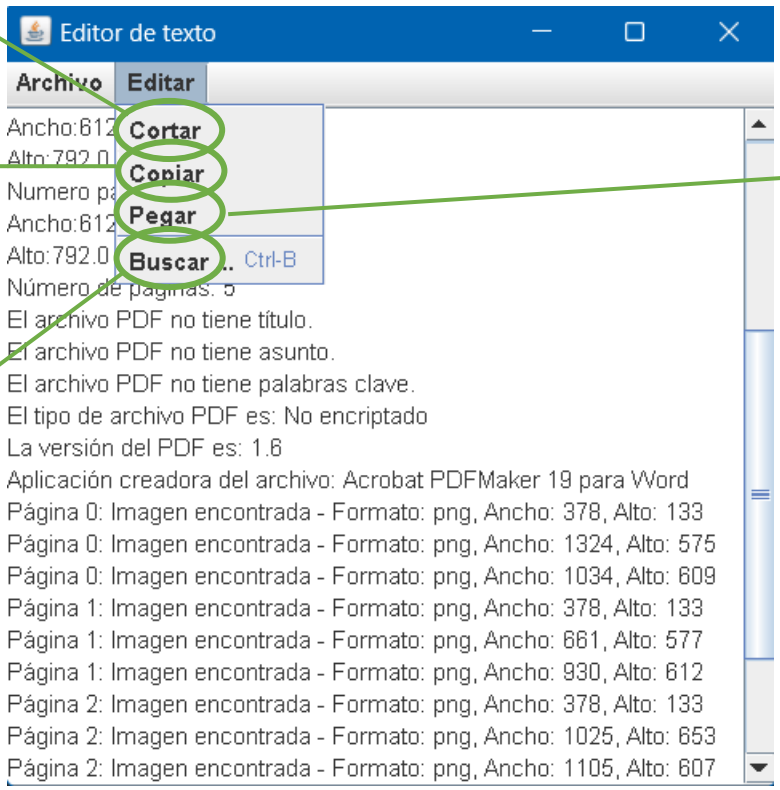
Salir de el editor de texto

Cortar texto seleccionado

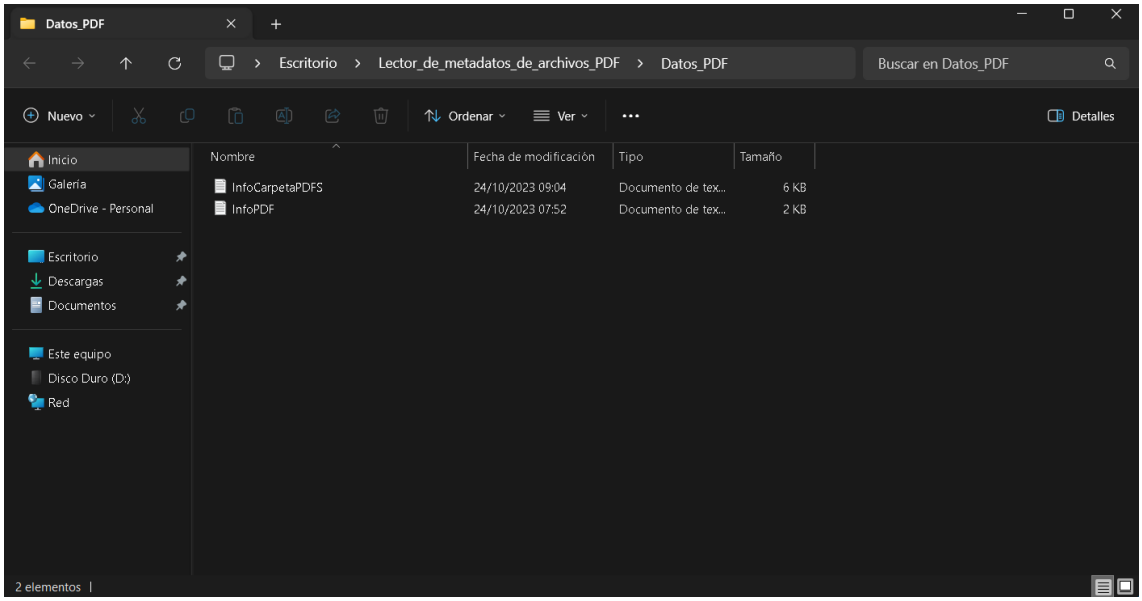
Copiar texto seleccionado

Opción para buscar por palabras clave

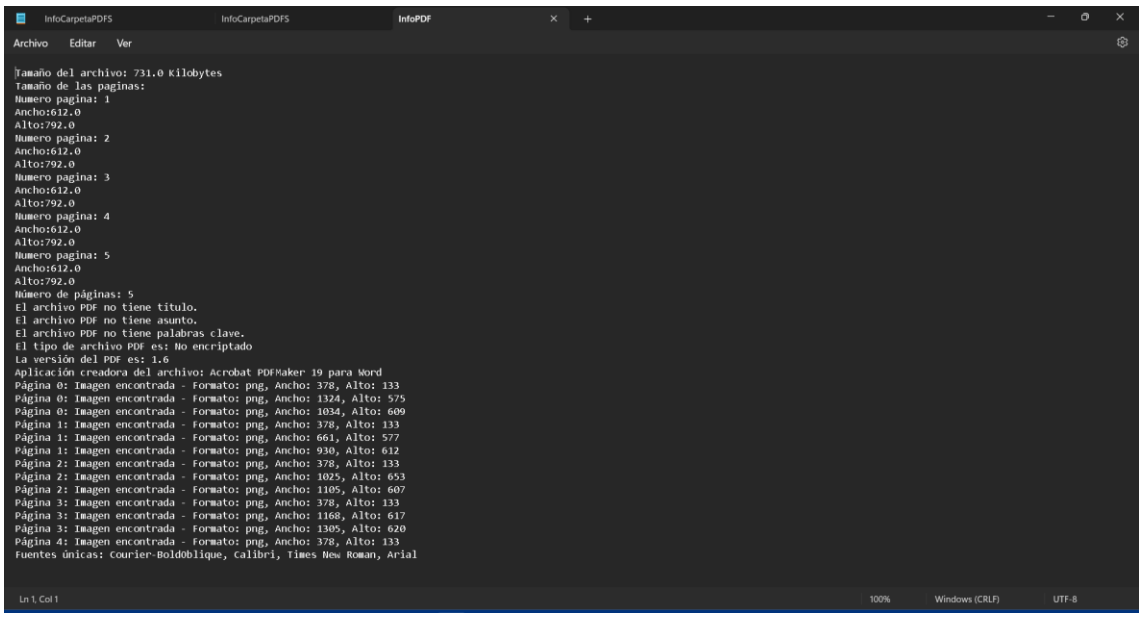
Pegar lo que tengamos en el portapapeles

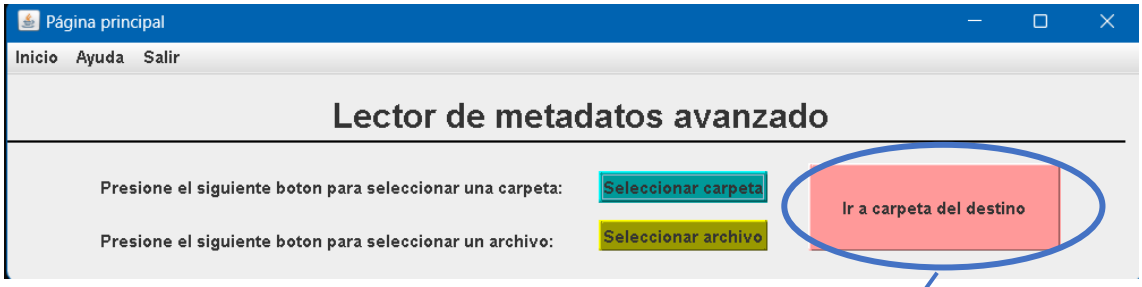


después de saber como funciona el editor de texto, si apretamos el botón “Ir a la carpeta destino” nos enviara a la carpeta en donde tenemos los archivos .txt, como se muestra en la siguiente imagen.



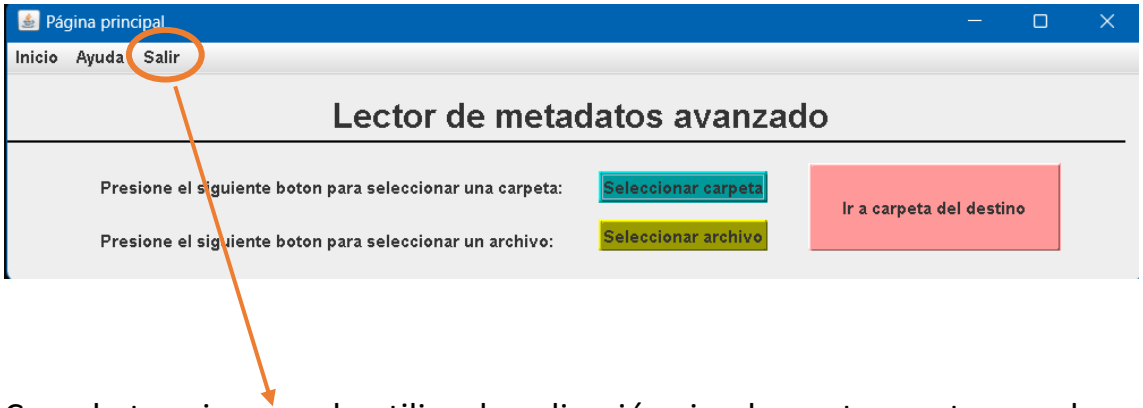
Cuando terminemos de editar o de simplemente leer los datos y de abrir la carpeta donde tenemos el archivo .txt, simplemente abrimos el archivo “Info PDF” y podremos ver ahí los datos del archivo PDF, con los datos editados hechos por nosotros, si es que hicimos algún cambio.





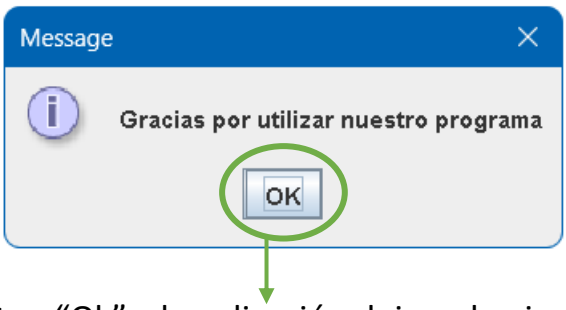
Este botón simplemente lo que hace es abrir en el explorador de archivos la carpeta en la que guardaremos los archivos .txt de los datos de los PDFS.

Si tenemos alguna duda sobre como utilizar el programa podemos apretar este botón y nos saldrá este archivo PDF (si, el archivo que estas leyendo).



Cuando terminemos de utilizar la aplicación simplemente apretamos el botón “Salir”

Nos saldra en pantalla el siguiente mensaje.

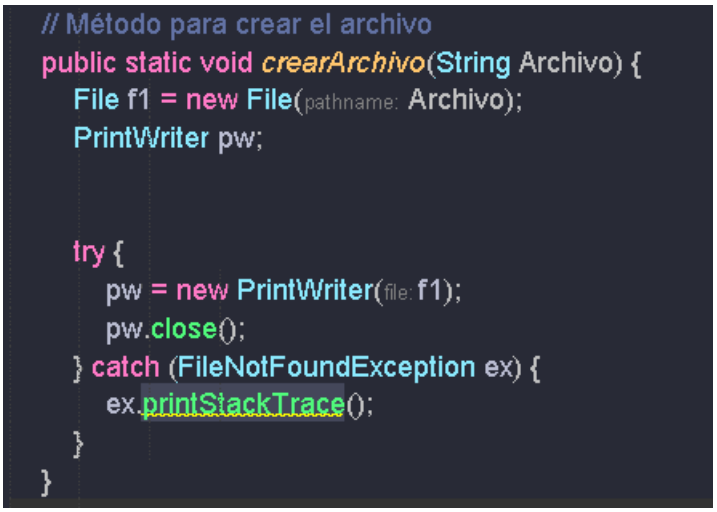


Apretamos el boton “Ok” y la aplicación dejara de ejecutarse.

Estructura de almacenamiento

Para poder almacenar la información que obtenemos del PDF tenemos múltiples opciones, nosotros optamos por escribir dicha información en un archivo de texto (.txt), lo que hacemos para que esto funcione es utilizar ciertos métodos para poder manejar la información a nuestro antojo en los archivos de texto. Los métodos son los siguientes:

- Crear archivo: Este método toma un nombre de archivo como entrada (dirección en donde esta el archivo), el nombre del archivo es de tipo String, crea un archivo con ese nombre (o sobrescribe el archivo si ya existe) y luego lo cierra. Este código es útil cuando se necesita crear un archivo vacío o sobrescribir un archivo existente en el sistema de archivos.



```
// Método para crear el archivo
public static void crearArchivo(String Archivo) {
    File f1 = new File(pathname: Archivo);
    PrintWriter pw;


    try {
        pw = new PrintWriter(file: f1);
        pw.close();
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    }
}
```

- Escribir archivo: Este método toma un nombre de archivo (dirección de donde se encuentra), este es tipo String, y un texto como entrada, también es tipo String, y escribe el texto en el archivo especificado. Si el archivo no existe, se creará. Si el archivo ya existe, el texto se agregará al final. Este código es útil cuando se necesita agregar contenido a un archivo o crear un archivo con contenido

nuevo.

```java
// Método para escribir en el archivo
public static void escribirArchivo(String Archivo, String texto) {
    File f1 = new File(pathname: Archivo);

    // Capturar errores
    try {
        PrintWriter pw = new PrintWriter(new FileWriter(file: f1, append: true));
        pw.println(x: texto);
        pw.close();

        // Mensaje para indicar que se creó el archivo con éxito
    } catch (FileNotFoundException ex) {
        ex.printStackTrace(s: System.out);
    } catch (IOException ex) {
        ex.printStackTrace(s: System.out);
    }
}
```

- Leer archivo: Cuando llames a este método y pases un nombre de archivo (dirección de donde se encuentra), tipo String, como argumento, el método devolverá el contenido del archivo en forma de una cadena, que puedes utilizar en tu programa en lugar de solo imprimirlo en la consola.

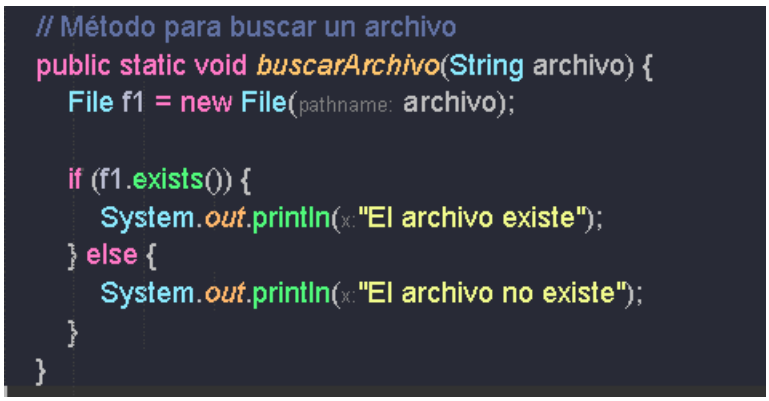
```java
// Método para leer el archivo
public static String leerArchivo(String Archivo) {
    File f1 = new File(pathname: Archivo);

    // Capturar errores
    try {
        BufferedReader br = new BufferedReader(new FileReader(fileName: Archivo));
        String leer = br.readLine();
        while (leer != null) {
            System.out.println(x: leer);
            leer = br.readLine();
        }
        br.close();
    } catch (FileNotFoundException ex) {
        ex.printStackTrace(s: System.out);
    } catch (IOException ex) {
        ex.printStackTrace(s: System.out);
    }

    return null;
}
```

- Archivos en la carpeta: Este método proporciona una forma sencilla de listar y mostrar los nombres de los archivos y subdirectorios dentro de una carpeta especificada. Puedes llamar a este método pasando la ruta de una carpeta, ósea una variable tipo String, como argumento, y te mostrará los nombres de los elementos contenidos en esa carpeta en la consola. Es una función útil para obtener una vista general de los contenidos de una carpeta en el sistema de archivos.

- Buscar archivos: Este método proporciona una forma de verificar la existencia de un archivo en el sistema de archivos. Puedes llamar a este método pasando la ruta de un archivo, variable tipo String, como argumento, y te indicará si el archivo existe o no en la ubicación especificada. Es útil para realizar comprobaciones antes de realizar operaciones de lectura o escritura en un archivo.



- Buscar una palabra: Este método permite al usuario ingresar una palabra o cadena y busca esa palabra en un archivo de texto línea por línea. Si se encuentra la palabra, muestra un mensaje que indica en qué línea se encontró. Este método es útil para realizar

búsquedas de palabras específicas en archivos de texto. Ten en cuenta que este código solo muestra la primera ocurrencia de la palabra en cada línea. Si deseas encontrar todas las ocurrencias, deberás adaptar el código para que lo haga.

```java
// Método para Buscar una palabra
public static void BuscarPalabra(String archivo) throws IOException {
    String palabra = JOptionPane.showInputDialog(parentComponent: null, message: "Ingrese un cadena: ");

    try {
        BufferedReader br = new BufferedReader(new FileReader(fileName: archivo));
        String leer = "";
        int i = 0;
        while ((leer = br.readLine()) != null) {
            i++;
            if (leer.contains(s: palabra)) {
                JOptionPane.showMessageDialog(parentComponent: null, "La palabra " + palabra + " fue encontrada en la línea " + i);
            }
        }
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    }
}
```

- Leer CSV e imprimir linea: Estos métodos permiten la lectura de un archivo CSV, división de cada línea en valores, y la impresión de esos valores en la consola. Cada valor en una línea se imprime seguido de

una coma, lo que es típico en la representación de datos CSV.

```
static String línea;
static String[] lineaLeida = null;

// Método para leer archivos CSV
public static void leerCSV(String archivo) throws IOException {
    try {
        BufferedReader br = new BufferedReader(new FileReader(fileName: archivo));
        while ((línea = br.readLine()) != null) {
            lineaLeida = línea.split(regex: ",");
            imprimirLinea();
            System.out.println();
        }
        br.close();
        línea = null;
        lineaLeida = null;
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

// Método para imprimir la línea
public static void imprimirLinea() {
    for (int i = 0; i < lineaLeida.length; i++) {
        System.out.print(lineaLeida[i] + ",");
    }
}
```

Para poder agregar, eliminar o editar datos del archivo de texto, se opto por crear una ventana nueva y que esta abra el archivo de texto y que nosotros podamos hacer lo que deseemos con la información de este, para lograr esto utilizamos los siguientes métodos:

- Correr la ventana: Este código se utiliza para abrir y leer un archivo de texto, y luego mostrar su contenido en un área de texto, para que

de esta forma podamos editar algo que deseemos.

```
//============================================================ main
public void correr() {
    String filePath = "C:\\Users\\urbin\\OneDrive\\Escritorio\\Lector_de_metadatos_de_archivos_PDF\\Datos_PDF\\InfoPDF.txt";
    File file = new File(pathname: filePath);
    area.setText(t: "");

    BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(file));
    } catch (FileNotFoundException ex) {
        Logger.getLogger(name: AbrirArchivo.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
    String linea = null;
    try {
        linea = reader.readLine();
    } catch (IOException ex) {
        Logger.getLogger(name: AbrirArchivo.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
    while (linea != null) {
        // Aquí lo que tengamos que hacer con la línea puede ser esto
        area.append(str: linea);
        area.append(str: System.getProperty(key: "line.separator"));
        try {
            linea = reader.readLine();
        } catch (IOException ex) {
            Logger.getLogger(name: AbrirArchivo.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
    }

}
```

- Guardar archivo: Este código se utiliza para permitir al usuario seleccionar una ubicación y un nombre de archivo y guardar el contenido del área de texto en ese archivo. Es una implementación común en aplicaciones que requieren la funcionalidad de guardar archivos, como editores de texto o aplicaciones de procesamiento de documentos.

```
public class GuardarArchivo implements ActionListener {

    public void actionPerformed(ActionEvent ae) {

        int retval = fileChooser.showSaveDialog(parent: frmEditor.this);
        if (retval == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();

            PrintWriter writer = null;
            try {
                writer = new PrintWriter(file);
            } catch (FileNotFoundException ex) {
                Logger.getLogger(name: GuardarArchivo.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
            }
            writer.print(s: area.getText());
            writer.close();

        }
    }
}
```

Algoritmos utilizados

En el proyecto utilizamos distintos algoritmos para poder obtener la información que deseamos del PDF seleccionado, es importante marcar que ninguno de estos algoritmos es complejo, son sencillos y buscamos siempre la forma de simplificarlos al máximo, también es importante destacar el uso de la librería "PDFBox" ya que esta nos facilitó mucho la extracción de datos, todos o en la mayoría de los algoritmos se utiliza esta librería para obtener los datos. Los algoritmos son los siguientes:

- Calcular tamaño de los archivos: Esta función calcula el tamaño de un archivo PDF y lo almacena en kilobytes o megabytes, dependiendo de su tamaño. No utiliza algoritmos complejos, solo realiza una conversión de bytes a kilobytes o megabytes.

```
// Definiar el tamaño de los archivos pdfs
public static void CalcularTamañoArchivos(File archivo, String archivoGuardar) {
    Métodos met = new Métodos();

    long tamanoBytes = archivo.length();
    float tamaño = tamanoBytes / 1000;

    if (tamaño > 1000) {
        met.escribirArchivo(Archivo:  archivoGuardar, tamaño / 1000 + " Megabytes");
    } else {
        met.escribirArchivo(Archivo:  archivoGuardar, tamaño + " Kilobytes");
    }
}
```

- Calcular tamaño de las páginas: Esta función calcula el tamaño de cada página en un archivo PDF, obteniendo el ancho y alto de cada página. No utiliza un algoritmo en el sentido tradicional, sino que accede a las propiedades del archivo PDF y extrae información sobre

el tamaño de las páginas.

```
// Definir el tamaño de las paginas de los archivos pdfs
public static void CalcularTamañoPaginas(File archivo, String archivoGuardar) {
    Métodos met = new Métodos();

    try (PDDocument document = PDDocument.load(file: archivo)) {
        int numeroDePaginas = document.getNumberOfPages();

        for (int pageNum = 0; pageNum < numeroDePaginas; pageNum++) {
            PDPage page = document.getPage(pageIndex: pageNum);

            float ancho = page.getMediaBox().getWidth();
            float alto = page.getMediaBox().getHeight();

            met.escribirArchivo(Archivo: archivoGuardar, "Página " + (pageNum + 1) + ":");
            met.escribirArchivo(Archivo: archivoGuardar, "Ancho: " + ancho + " puntos");
            met.escribirArchivo(Archivo: archivoGuardar, "Alto: " + alto + " puntos");
        }
        System.out.println(x:"----------");
    } catch (IOException e) {
    }
}
```

- Calcular el número de páginas: Esta función calcula el número de páginas en un archivo PDF. No utiliza un algoritmo complejo, simplemente obtiene la información directamente del objeto PDDocument que representa el archivo PDF.

```
// Calcular el numero de paginas de los archivos pdfs
public static void CalcularNumeroPaginas(PDDocument pdf, String archivoGuardar) {
    Métodos met = new Métodos();
    int numPaginas = pdf.getNumberOfPages();

    met.escribirArchivo(Archivo: archivoGuardar, "Número de páginas: " + numPaginas);
}
```

- Obtener título: Esta función extrae y muestra el título del archivo PDF. Nuevamente, no implica un algoritmo complejo, sino que

accede a los metadatos del archivo PDF y recupera el título.

```
// Definir el titulo de los archivos pdfs
public static void ObtenerTitulo(PDDocumentInformation info, String archivoGuardar) {
    Métodos met = new Métodos();
    String titulo = info.getTitle();

    if (titulo != null && !titulo.isEmpty()) {
        met.escribirArchivo(Archivo: archivoGuardar, "Titulo: " + titulo);
    } else {
        met.escribirArchivo(Archivo: archivoGuardar, texto:"El archivo PDF no tiene título.");
    }
}
```

- Obtener asunto PDF: Similar a la función anterior, esta obtiene y muestra el asunto del archivo PDF.

```
// Determinar el asuntos de los archivos pdfs
public static void ObtenerAsuntoPDF(PDDocumentInformation info, String archivoGuardar) {
    Métodos met = new Métodos();
    String asuntoPDF = info.getSubject();

    if (asuntoPDF != null && !asuntoPDF.isEmpty()) {
        met.escribirArchivo(Archivo: archivoGuardar, "Asunto: " + asuntoPDF);
    } else {
        met.escribirArchivo(Archivo: archivoGuardar, texto:"El archivo PDF no tiene asunto");
    }
}
```

- Obtener palabras clave: Esta función obtiene y muestra las palabras clave (keywords) asociadas con el archivo PDF.

```
// Determinar las palabras claves del pdf
public static void ObtenerPalabrasClaves(PDDocumentInformation info, String archivoGuardar) {
    Métodos met = new Métodos();
    String palabrasClave = info.getKeywords();

    if (palabrasClave != null && !palabrasClave.isEmpty()) {
        met.escribirArchivo(Archivo: archivoGuardar, "Palabras clave: " + palabrasClave);
    } else {
        met.escribirArchivo(Archivo: archivoGuardar, texto:"Este archivo no tiene palabras clave");
    }
}
```

- Obtener tipo PDF: Determina si el archivo PDF está encriptado o no. No utiliza un algoritmo en sí, sino que verifica una propiedad del

objeto PDDocument.

```
// Determinar el tipo de pdf que es
public static void ObtenerTipoPDF(PDDocument pdf, String archivoGuardar) {
    Métodos met = new Métodos();
    if (pdf.isEncrypted()) {
        met.escribirArchivo(Archivo: archivoGuardar, texto:"El tipo de archivo PDF es: Encriptado");
    } else {
        met.escribirArchivo(Archivo: archivoGuardar, texto:"El tipo de archivo PDF es: No encriptado");
    }
}
```

- Obtener versión PDF: Obtiene y muestra la versión del archivo PDF.

```
// Determinar la version del pdf
public static void ObtenerVersionPDF(PDDocument pdf, String archivoGuardar) {
    Métodos met = new Métodos();
    float version = pdf.getVersion();

    met.escribirArchivo(Archivo: archivoGuardar, "Numero de version: " + version);
}
```

- Obtener aplicación creadora: Extrae y muestra la aplicación que se utilizó para crear el archivo PDF.

```
// Determinar la aplicacion con la que fue creada el pdf
public static void ObtenerAplicacionCreador(PDDocumentInformation info, String archivoGuardar) {
    Métodos met = new Métodos();
    String aplicacionCreadora = info.getCreator();

    if (aplicacionCreadora != null && !aplicacionCreadora.isEmpty()) {
        met.escribirArchivo(Archivo: archivoGuardar, "Aplicación creadora del archivo: " + aplicacionCreadora);
    } else {
        met.escribirArchivo(Archivo: archivoGuardar, texto:"El archivo PDF no tiene la aplicación creadora.");
    }
}
```

- Obtener imágenes: Esta función busca y muestra información sobre las imágenes dentro del archivo PDF, incluyendo su formato, ancho y alto. Utiliza un bucle para recorrer las páginas y buscar imágenes.

```
// Determinar las imagenes del pdf
public static void ObtenerImagenes(PDDocument pdf, String archivoGuardar) throws IOException {
    Métodos met = new Métodos();
    int pageNum = 0;
    for (PDPage page : pdf.getPages()) {
        PDResources resources = page.getResources();
        for (COSName xObjectName : resources.getXObjectNames()) {
            PDXObject xObject = resources.getXObject(name: xObjectName);
            if (xObject instanceof PDImageXObject) {
                PDImageXObject image = (PDImageXObject) xObject;
                met.escribirArchivo(Archivo: archivoGuardar, "Página " + pageNum + ": Imagen encontrada - Formato: " + image.getSuffix() + ", Ancho: " + image.getWidth() + ", Alto: " + image.getHeight());
            } else if (xObject instanceof PDFormXObject) {
                PDFormXObject form = (PDFormXObject) xObject;
                for (COSName subXObjectName : form.getResources().getXObjectNames()) {
                    PDXObject subXObject = form.getResources().getXObject(name: subXObjectName);
                    if (subXObject instanceof PDImageXObject) {
                        PDImageXObject subImage = (PDImageXObject) subXObject;
                        met.escribirArchivo(Archivo: archivoGuardar, "Página " + pageNum + ": Imagen encontrada - Formato: " + subImage.getSuffix() + ", Ancho: " + subImage.getWidth() + ", Alto: " + subImage.getHeight());
                    }
                }
            }
        }

        pageNum++;
    }
}
```

- Obtener fuentes: Esta función analiza las fuentes utilizadas en el archivo PDF y muestra las fuentes únicas presentes en el documento. Utiliza un mapeo de nombres de fuentes y un conjunto para mantener un registro de las fuentes únicas.

```
// Determinar las fuentes del pdf
public static void ObtenerFuentes(PDDocument pdf, String archivoGuardar) {
    Métodos met = new Métodos();
    Map<String, String> fontMapping = new HashMap<>();
    fontMapping.put(key: "TT0", value:"Times New Roman");
    fontMapping.put(key: "TT1", value:"Calibri");
    fontMapping.put(key: "TT2", value:"Arial");
    fontMapping.put(key: "TT3", value:"Courier-BoldOblique");

    Set<String> uniqueFonts = new HashSet<>();

    for (PDPage page : pdf.getPages()) {
        PDResources resources = page.getResources();
        COSDictionary fonts = (COSDictionary) resources.getCOSObject().getDictionaryObject(key: COSName.FONT);
        if (fonts != null) {
            for (COSName fontName : fonts.keySet()) {
                String fontNameString = fontName.getName();
                String mappedFontName = fontMapping.get(key: fontNameString);
                if (mappedFontName != null) {
                    uniqueFonts.add(e:mappedFontName);
                }
            }
        }
    }

    met.escribirArchivo(Archivo: archivoGuardar, "Fuentes únicas: " + String.join(delimiter:", ", elements: uniqueFonts));
}
```

- Código principal: Este código se utiliza para recopilar información de archivos PDF en una carpeta seleccionada por el usuario y almacenar esa información en un archivo de texto. Luego, muestra la información en la consola y ofrece la opción de abrir la carpeta en la que se guardó el archivo de texto. Este proceso implica la manipulación de archivos, lectura y escritura de información, y la interacción con el usuario a través de cuadros de diálogo.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String archivo_guardar = "C:\\Users\\urbin\\OneDrive\\Escritorio\\Lector_de_metadatos_de_archivos_PDF\\Datos_PDF\\InfoCarpetaPDFS.txt";
    Métodos met = new Métodos();
    met.crearArchivo(Archivo:  archivo_guardar);

    JFileChooser jfc = new JFileChooser();

    jfc.showOpenDialog(parent: jfc);
    File archivoSeleccionado = jfc.getSelectedFile();

    String parent = archivoSeleccionado.getParent();
    String nuevoParent = parent.replaceAll(regex:  "\\\\", replacement: "\\\\\\\\");
    JOptionPane.showMessageDialog(parentComponent: null, "Carpeta Seleccionada -> " + nuevoParent);

    String rutaCarpeta = nuevoParent;

    File carpeta = new File(pathname: rutaCarpeta);

    if (carpeta.exists() && carpeta.isDirectory()) {
        File[] archivos = carpeta.listFiles();

        if (archivos != null) {
            int seleccion = JOptionPane.showOptionDialog(parentComponent: null, message:  "¿Desea abrir la carpeta en donde se guardó el archivo?",
                    title:  "Selector de opciones", optionType:  JOptionPane.YES_NO_CANCEL_OPTION,
                    messageType: JOptionPane.QUESTION_MESSAGE, icon: null,// null para icono por defecto.
                    new Object[]{"Si", "No"}, initialValue:  "opcion 1");
            System.out.println(x:seleccion);
            if (seleccion == 0) {
                String folderPath = "C:\\Users\\urbin\\OneDrive\\Escritorio\\Lector_de_metadatos_de_archivos_PDF\\Datos_PDF";

                // Abre la carpeta en el sistema de archivos
                try {
                    Desktop.getDesktop().open(new File(pathname: folderPath));
```

```
                try {
                    Desktop.getDesktop().open(new File(pathname: folderPath));
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            } else {
                System.out.println(x:"");
            }
            for (File archivo : archivos) {
                if (archivo.isFile() && archivo.getName().toLowerCase().endsWith(suffix:  ".pdf")) {
                    try {
                        PDDocument pdf = PDDocument.load(file: archivo);
                        PDDocumentInformation info = pdf.getDocumentInformation();

                        met.escribirArchivo(Archivo:  archivo_guardar, "Archivo: " + archivo.getName());
                        CalcularTamañoArchivos(archivo, archivoGuardar:  archivo_guardar);
                        CalcularTamañoPaginas(archivo, archivoGuardar:  archivo_guardar);
                        CalcularNumeroPaginas(pdf, archivoGuardar:  archivo_guardar);
                        ObtenerTitulo(info, archivoGuardar:  archivo_guardar);
                        ObtenerAsuntoPDF(info, archivoGuardar:  archivo_guardar);
                        ObtenerPalabrasClaves(info, archivoGuardar:  archivo_guardar);
                        ObtenerTipoPDF(pdf, archivoGuardar:  archivo_guardar);
                        ObtenerVersionPDF(pdf, archivoGuardar:  archivo_guardar);
                        ObtenerAplicacionCreador(info, archivoGuardar:  archivo_guardar);
                        ObtenerImagenes(pdf, archivoGuardar:  archivo_guardar);
                        ObtenerFuentes(pdf, archivoGuardar:  archivo_guardar);
                        met.escribirArchivo(Archivo:  archivo_guardar, texto:"**************************************************");
                        pdf.close();
                    } catch (IOException e) {
                        System.err.println("Error al leer el archivo PDF: " + archivo.getName());
                    }
                }
            }
            try {
```

```
        try {
            // Especifica la ruta del archivo de texto que deseas abrir
            String rutaArchivo = "C:\\Users\\urbin\\OneDrive\\Escritorio\\Lector_de_metadatos_de_archivos_PDF\\Datos_PDF\\InfoCarpetaPDFS.txt";

            // Crea un objeto BufferedReader para leer el archivo
            BufferedReader br = new BufferedReader(new FileReader(fileName: rutaArchivo));

            String linea;
            while ((linea = br.readLine()) != null) {
                System.out.println(x: linea);
            }

            // Cierra el BufferedReader
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        System.err.println(x: "La carpeta no existe o no es una carpeta válida.");
    }
}
```

Conclusiones

- Nuestra aplicación permite a los usuarios extraer datos de archivos PDF de manera simple y eficiente. Pueden definir reglas de extracción personalizadas y buscar términos específicos, lo que facilita el proceso de obtención de información relevante.
- Ofrecemos a los usuarios la flexibilidad de personalizar el proceso de extracción de datos de acuerdo con sus necesidades. Esto incluye la capacidad de especificar la ubicación de los datos, aplicar patrones de búsqueda y seleccionar qué información recopilar.
- Nuestra herramienta es capaz de automatizar la recopilación de datos, lo que ahorra tiempo y reduce la carga de trabajo manual. Esto resulta especialmente útil en entornos donde se requiere el procesamiento de grandes volúmenes de documentos PDF.
- El proyecto tiene aplicaciones en diversas áreas, desde la investigación y análisis de datos hasta la automatización de tareas relacionadas con documentos. Puede ser útil en entornos empresariales, académicos y de investigación, así como en la generación de informes personalizados.
- Al desarrollar esta aplicación en Java, hemos aprovechado la versatilidad y robustez de este lenguaje de programación. Java es ampliamente compatible y escalable, lo que garantiza que esta solución sea accesible para una amplia gama de usuarios.