Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Sistema para el reconocimiento óptico de caracteres en la digitalización de documentos.

Autores: Ronald Domínguez Marrero

David Caballero Savón

Tutores: Ing. Mayleidis López Fernández

Ing. Adrián Rivera Correa

La Habana, junio del 2015

"Año 56 de la Revolución"

Declaración de Autoría

Declaramos ser los autores del presente Trabajo de Diploma y le otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo con carácter exclusivo.

Para que así conste firmamos la presente declaración jurada de autoría en Ciudad de La Habana a los _____ días del mes de _____ del año _____.

Ronald Domínguez Marrero

_____

Firma del autor(a)

David Caballero Savón

_____

Firma del autor(a)

Ing. Mayleidis López Fernández

_____

Firma del Tutor(a)

Ing. Adrián Rivera Correa

_____

Firma de Tutor(a)

Dedicatoria

David:

Dedico este trabajo a mi madre Arelis por ayudarme, apoyarme incondicionalmente en todas las decisiones que he tomado durante estos 5 años.

A mi hermana Daineris por su ayuda, comprensión e inspiración.

A mi familia Savón que tanto apoyo recibí de su parte incondicionalmente, en especial a mi abuelo Omar por ser un ejemplo a seguir por su perseverancia e ímpetu de trabajo, a mis tías Yamile, Karelia, Katerine que me ayudaron muchísimo en el transcurso de la carrera, a mi tío Ernesto que a pesar de la distancia me apoyó y me guio por el buen camino.

A mi familia Massuín, especialmente a mi abuela Sara, a mis tíos Adrián y Abram, que siempre me dieron ánimo y me apoyaron muchísimo a pesar de las difíciles situaciones por la pasamos, a mi tía Teresita que la considero como mi segunda madre.

A mi familia Caballero, a mis tíos y tías Angela, Idalmis, Eida, Félix, Reina y Neivis, en especial a mi prima Marysledis de la cual recibí su apoyo incondicional.

A mi novia Aleyda que tanto luchó conmigo en los dos años que llevamos.

A mis compañeros de grupo, a mi amigo Adrián que a pesar de la distancia siempre fue y será un ejemplo a seguir.

Por último quiero dedicarle este trabajo a mi padre que siempre me dio fuerzas cuando me faltaban para seguir adelante, gracias a las cualidades que me inculcó en vida.

Ronald

Dedico este trabajo a mi familia por apoyarme siempre en todo y a mi novia por estar siempre ahí para mí.

Agradecimientos

David:

A mi tutor Adrián Rivera que sin él no hubiera sido posible este trabajo.

A mi compañero de tesis que se ha comportado como un amigo, gracias por su apoyo incondicional durante el transcurso de la Universidad.

A mi novia Aleyda por su estar al lado mío en estos 2 últimos años, y brindarme su ayuda de amiga y compañera.

A mi familia que tanto me ha apoyado en el transcurso de la carrera. A mis compañeros que tanto compartimos, a Ronald, Alexander, Ernán, Dennis, Dariel, Carlos Osvaldo, que tanto me apoyaron incondicionalmente en la Universidad.

A mi madre por haber luchado tanto para poder graduarme a mi hermana por apoyarme, a mi abuelo Omar, a mis tías y tíos: Teresa, Yamile, Karelia, Katerine, Adrián, Abram, Ernesto.

En especial a mi padre por darme tanto espíritu de lucha y fuerzas para poder continuar en los momentos difíciles a pesar de no estar físicamente a mi lado.

Ronald

Quiero agradecer primeramente a los tutores, especialmente a Adrián por todo su apoyo y ayuda incondicional.
A mi familia por ser un motivo para seguir a adelante.
A mi novia por todo su apoyo.
A mis compañeros de grupo por todos los momentos compartidos.
A todas las personas que de una manera u otra aportaron su granito de arena para que este trabajo pudiera realizarse.

Resumen

A causa de la necesidad de almacenar y compartir la información de manera fácil, surge la digitalización de los documentos. Para facilitar el acceso y el uso de la información digitalizada se crean los sistemas para el reconocimiento óptico de caracteres (OCR, por sus siglas en inglés). Estos sistemas están diseñados para reconocer los caracteres que se encuentran dentro de las imágenes resultantes del proceso de digitalización de documentos y de esta manera crear un documento digital con el texto contenido en estas imágenes, el cual le permita al usuario interactuar con dicha información de manera más sencilla. La presente investigación se realiza a causa de la necesidad de la biblioteca de encontrar un OCR desarrollado para plataformas libres para seguir realizando el proceso de digitalización de los documentos que está llevando a cabo, ya que el software utilizado actualmente es propietario y la institución debe migrar hacia tecnologías libres por políticas de la universidad y del país. Con la realización de este trabajo se pretende desarrollar un sistema que reduzca los errores existentes en el reconocimiento de caracteres automático durante el proceso de digitalización de documentos en la biblioteca de la Universidad de las Ciencias Informáticas utilizando tecnologías libres. Este sistema debe ser capaz de detectar y reconocer correctamente los caracteres contenidos dentro de imágenes digitales. Para ello se realiza un análisis a los algoritmos existentes que realizan este proceso, se seleccionan las herramientas necesarias para el desarrollo del sistema, se diseña, se implementa y se realizan pruebas para comprobar la efectividad y el correcto funcionamiento del mismo.

Con la implementación del sistema para el reconocimiento óptico de caracteres se pretende que la biblioteca de la Universidad de las Ciencias Informáticas pueda concluir con su migración a software libre y a su vez pueda seguir realizando el proceso de digitalización de los documentos con buena calidad.

Palabras claves: digitalización, reconocimiento óptico de caracteres.

Índice

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

Introducción

El Reconocimiento Óptico de Caracteres (OCR, por sus siglas en inglés) es un proceso mediante el cual se identifican automáticamente imágenes, símbolos o caracteres que pertenecen a un determinado alfabeto captados por un dispositivo electrónico, para luego almacenarlos en forma de datos y así poder interactuar con ellos. (1)

Estos sistemas tienen múltiples aplicaciones, entre las más conocidas se encuentran: el reconocimiento de matrículas de autos, la indexación de imágenes en bases de datos, el reconocimiento de datos en documentos estructurados con OCR Zonal y la digitalización de documentos.

Los sistemas para el reconocimiento óptico de caracteres son muy utilizados en el proceso de digitalización de archivos, ya que una de sus aplicaciones es precisamente el reconocimiento del texto contenido en imágenes digitales.

Los archivos se utilizan para gestionar, atesorar, conservar y difundir el patrimonio documental. Estos pueden almacenar documentos históricos recibidos por donación, depósito, transferencia y adquisición. Con la digitalización de los mismos se facilita el manejo de la información, así como su preservación, almacenamiento y transportación. Mediante la digitalización se reduce la manipulación de los documentos originales y permite conservarlos para retrasar su deterioro. Además, disminuye el espacio físico y el riesgo de robo o destrucción de los documentos. (2)

Cada vez es más imprescindible la digitalización de los archivos a causa del crecimiento de la información y la necesidad de recuperación de los documentos. Los archivos demandan cambios inmediatos en el registro archivístico, transformaciones en la forma de catalogar y recuperar la información: requieren proveer mejores servicios que la antigua infraestructura no puede soportar, como es la digitalización de documentos, de forma que los usuarios puedan acceder al documento original. Dentro de este contexto, se abren los trabajos de evaluación de las nuevas estrategias del servicio, orientado a las necesidades de los usuarios y el uso de nuevas tecnologías de información. (2)

Un documento digital puede ser el resultado de haber procesado con un escáner un documento originalmente impreso. Primeramente se obtiene una imagen (fotografía digital) del documento

escaneado. La imagen que se obtiene puede ser guardada en un medio electrónico, pero no tiene las capacidades de hipertexto de un documento de texto digital. Posteriormente esta imagen puede ser procesada por un sistema de reconocimiento óptico de caracteres para así convertirla en un documento digital con todas las capacidades de hipertexto.

La biblioteca de la Universidad de las Ciencias Informáticas (UCI) preserva gran volumen de documentos como libros de textos, revistas y otros documentos así como trabajos de diploma resultado de la conclusión de estudios de pre y postgrado, en su mayoría referidos a temáticas relacionadas a aspectos de la informática. A este lugar acuden diariamente estudiantes y profesores solicitando acceso a estos documentos para consultar información de interés con el objetivo de realizar trabajos docentes, proyectos investigativos y de desarrollo. Este procedimiento de búsqueda de información se realiza examinando los temas de interés para el usuario de acuerdo a los títulos de los documentos, lo que hace lento e ineficaz el proceso, debido a que se puede omitir información valiosa que solo se encuentre explícita en el contenido del documento, para realizar una búsqueda más profunda habría que abrir cada documento existente hasta que se encuentre el que contenga la información que busca el usuario. Para facilitar el acceso a la información, la biblioteca realiza un proceso de digitalización de los documentos y para ello utiliza la herramienta Adobe Acrobat Profesional.

En estos momentos la universidad lleva a cabo el plan de migración hacia tecnologías libres. La biblioteca, como las demás instituciones dentro de la universidad, lleva a cabo este proceso de migración, pero no ha podido completarlo porque para la digitalización de los documentos utiliza la herramienta antes mencionada que es propietaria. Esta herramienta por su condición de software privado no permite realizar modificaciones en su código fuente ya que no se tiene acceso al mismo, lo que imposibilita poder adaptarlo a ciertos entornos de trabajo. Por este motivo se decidió iniciar la búsqueda de un software desarrollado para plataformas libres que permita seguir realizando este proceso con calidad.

Los sistemas para el reconocimiento óptico de caracteres encontrados presentan errores en el reconocimiento de los caracteres, ya que en ocasiones confunden números con letras, como es el caso de la “I” con el “1”, la “O” con el “0” y la “S” con el “5”. Muchos no realizan un correcto procesamiento a los documentos que contienen imágenes, algunos no son capaces de diferenciar figuras y textos, existen diferentes tipos de letras que varios de estos software no reconocen como por ejemplo la cursiva, si las imágenes tienen inclinación o el texto que contienen está inclinado muchos no son capaces de

procesarlas correctamente y a las imágenes que contienen texto a color o el fondo a color algunos de estos sistemas no les realizan un correcto procesamiento. Por estos motivos la institución ha decidido no utilizar ninguna de estas aplicaciones y se ha propuesto desarrollar su propio sistema para el reconocimiento óptico de caracteres.

La situación antes planteada lleva a enunciar el siguiente **problema de investigación**:
¿Cómo reducir los errores en el reconocimiento de caracteres automático durante el proceso de digitalización de documentos utilizando tecnologías libres?

Definiéndose como **objeto de estudio** el Reconocimiento Óptico de Caracteres (OCR).

Para dar solución al problema antes expuesto se plantea como **objetivo general:** Desarrollar un sistema que reduzca los errores existentes en el reconocimiento de caracteres automático durante el proceso de digitalización de documentos en la biblioteca de la Universidad de las Ciencias Informáticas utilizando tecnologías libres. Desglosándose en los siguientes **objetivos específicos:**

- Elaborar el marco teórico conceptual referente al objeto de estudio.
- Diseñar el sistema para el reconocimiento óptico de caracteres.
- Implementar el sistema para el reconocimiento óptico de caracteres.
- Realizar pruebas al sistema de reconocimiento óptico de caracteres para comprobar su efectividad.

Enmarcándose como **campo de acción** el reconocimiento óptico de caracteres en la digitalización de documentos.

Para darle cumplimiento al objetivo planteado se proponen las siguientes **tareas de la investigación**:

- Análisis de los principales conceptos asociados al reconocimiento óptico de caracteres para obtener la base teórica necesaria. [Responsable: Ronald Domínguez Marrero]
- Revisión bibliográfica acerca de las soluciones existentes que permiten el reconocimiento óptico de caracteres, para determinar fortalezas e insuficiencias en las mismas y valorar la viabilidad de su uso. [Responsable: David Caballero Savón]
- Análisis para definir las herramientas, métodos y metodologías de desarrollo de software para la realización del sistema. [Responsable: Ronald Domínguez Marrero y David Caballero Savón]

- Especificación de los requisitos del sistema de reconocimiento óptico de caracteres, para determinar las funcionalidades a implementar. [Responsable: Ronald Domínguez Marrero, David Caballero Savón]
- Diseño de la solución del sistema de reconocimiento óptico de caracteres. [Responsable: Ronald Domínguez Marrero, David Caballero Savón]
- Implementación de la fase de pre-procesamiento. [Responsable: David Caballero Savón]
- Implementación de la fase de segmentación. [Responsable: David Caballero Savón]
- Implementación de la fase de extracción de características. [Responsable: David Caballero Savón]
- Implementación de la fase de reconocimiento. [Responsable: David Caballero Savón]
- Implementación de la fase de conformación del documento de salida. [Responsable: David Caballero Savón]
- Aplicación de las pruebas de software al sistema de reconocimiento óptico de caracteres. [Responsables: David Caballero y Ronald Domínguez].

El **resultado esperado** es obtener una primera versión del sistema para el reconocimiento óptico de caracteres. Se pretende que el sistema implementado reconozca gran cantidad de caracteres con diferentes fuentes de letras, que sea capaz de diferenciar entre textos e imágenes y que la salida del proceso sea un documento que esté en un formato de texto editable.

Los **Métodos Teóricos** utilizados para dar cumplimiento a las tareas son los siguientes:

Analítico-Sintético: Para la realización del presente trabajo se estudiaron diferentes bibliografías referentes al desarrollo de sistemas OCR, permitiendo extraer de estas los elementos más importantes que se relacionan con el tema. Se analizaron también varios algoritmos, lo que permitió la selección de las técnicas a emplear para el desarrollo del sistema.

Análisis Histórico-Lógico: En el presente trabajo se realiza un estudio de la evolución y desarrollo de algunos sistemas OCR desarrollados hasta el momento y sus principales aportes, lo que sirvió de guía para desarrollar el sistema.

Modelación: Se utilizó la modelación para representar por medio de diagramas los conceptos asociados a las fases y procesos de los sistemas OCR para de esta manera hacerlos más comprensibles a la hora del desarrollo.

Los **métodos empíricos** utilizados fueron:

Entrevista: Para la realización del trabajo se realizó una entrevista a la persona encargada de realizar la digitalización de documentos en la biblioteca, con el objetivo de obtener información acerca de este proceso y de los principales problemas a los que se enfrenta para la realización del mismo.

Experimento: En el presente trabajo se utilizó este método para la conformación del sistema. Se cuentan con varias propuestas de algoritmos para cada fase y por esta razón se realizó un experimento para comprobar la efectividad de los mismos. Los resultados fueron comparados para escoger los algoritmos con menor cantidad de errores detectados.

Justificación de la investigación

La presente investigación se realiza debido a que la biblioteca de la UCI utiliza para la digitalización de los documentos un sistema OCR propietario, dicho sistema no es adaptable a determinados entornos de trabajos de acuerdo a la necesidad del cliente, ya que su código fuente no es visible y en igual forma no permite modificaciones. Por ende es necesario desarrollar un sistema OCR implementado con tecnologías libres el cual resulte de gran impacto económico y social a nuestra universidad y para el país.

Estructura del trabajo:

Capítulo 1. “Fundamentación teórica”: En este capítulo se realiza un análisis del estado del arte de los sistemas OCR. Se mencionan los principales conceptos asociados al tema y se definen la metodología, herramientas y lenguajes a utilizar.

Capítulo 2. “Análisis y diseño”: En este capítulo se realiza el modelado de los artefactos generados en los flujos de trabajo Análisis y Diseño, los cuales guían la implementación del sistema a partir de la arquitectura definida. Además se define el algoritmo que da solución al problema.

Capítulo 3. “Implementación y pruebas”: En este capítulo se realiza una breve explicación del funcionamiento de los componentes del sistema y se realizan las pruebas necesarias para comprobar la efectividad de la aplicación desarrollada, mostrándose los resultados obtenidos.

Capítulo 1 “Fundamentación teórica”

1.1 Introducción

En este capítulo se hace alusión a los principales términos a tratar en el desarrollo del trabajo. Se realiza un análisis del estado del arte de los sistemas de Reconocimiento Óptico de Caracteres. También se hace un estudio para seleccionar la metodología, herramienta para el modelado, herramientas de desarrollo y el lenguaje de programación que se utilizan en la realización del presente trabajo.

1.2 Descripción de la problemática

En la biblioteca de la UCI se está llevando a cabo el proceso de digitalización de los documentos mediante la herramienta Adobe Acrobat Profesional. Este proceso se realiza con el objetivo de facilitar el acceso a la información contenida en las diferentes bibliografías existentes con el fin de brindar un mejor servicio a las personas que acuden diariamente a este lugar. En estos momentos la UCI está aplicando el plan de migración hacia tecnologías libres. Por lo que la biblioteca ha decidido cambiar la herramienta utilizada en la digitalización de los documentos, ya que la misma es propietaria. Por este motivo y por políticas de la universidad, la dirección de la biblioteca ha decidido utilizar un software desarrollado con tecnologías libres. La dirección de la biblioteca asignó al departamento de informática la tarea de encontrar un software libre que cumpla con estos requerimientos para seguir realizando la digitalización de los documentos. Los encargados realizaron la búsqueda de varios sistemas que realicen este proceso y llegaron a la conclusión que la mayoría de los sistemas encontrados para el proceso de digitalización de documentos están desarrollados con tecnologías propietarias; también se detectó que al existir muy pocos sistemas para el reconocimiento óptico de caracteres desarrollados con tecnologías libres, no abunda la documentación referente a ellos. Después de un análisis realizado a las soluciones existentes, se detectaron diferentes problemas a la hora de realizar el proceso de digitalización, entre ellos se encuentran: incorrecto procesamiento a los documentos que contienen figuras, ya que en ocasiones no las diferencian del texto; no reconocen algunos tipos de letras, como por ejemplo la cursiva; problemas para procesar imágenes con el texto inclinado e incorrecto procesamiento de las imágenes que contienen el texto o el fondo a color. Después de detectar estas deficiencias, la dirección de la biblioteca tomó la decisión de no utilizar ninguno de los sistemas encontrados e implementar su propio sistema para el reconocimiento óptico de caracteres. Para la realización del trabajo se enfocó la investigación en los sistemas desarrollados con tecnologías libres, no descartando la posibilidad de tomar algunas de las

ventajas de los sistemas propietarios existentes. La dirección de la biblioteca planteó que el software debe ser capaz de procesar textos en varios tipos de fuentes de letras, principalmente las más usadas, debe diferenciar en el documento lo que es texto e imagen y el documento generado debe estar en un formato reconocible por los editores de texto para poder realizar sobre éste las modificaciones que se crean necesarias.

1.4 Conceptos asociados al dominio del problema

Los sistemas OCR utilizados para la digitalización de documentos realizan las siguientes fases: **Pre-procesamiento**, fase en la que se mejora y se prepara la imagen para las fases posteriores; **Segmentación,** fase en la que se separan los diferentes elementos contenidos en la imagen de una página de un documento de texto tales como textos y figuras; **Extracción de características,** fase en la que se extraen las características de los caracteres detectados; **Reconocimiento,** fase en la que se reconocen los caracteres contenidos en la imagen de entrada; **Conformación de documento,** fase en la que se conforma y exporta el texto obtenido como resultado del proceso al formato de documento deseado. En la ilustración 1 se muestra el flujo de estas fases.

Ilustración 1 "Fases de un sistema OCR"

A continuación se profundiza en los conceptos asociados a las fases de un OCR para una mayor comprensión.

En la fase de pre-procesamiento se realiza la *binarización* de la imagen original y con ella se obtiene una imagen en blanco y negro que contiene las propiedades y características esenciales de la imagen. Como resultado de esta operación se reduce la presencia de datos que no son importantes en la imagen, que no muestran información valiosa y de esta manera facilita el procesamiento que realizan las otras fases del sistema, como la segmentación. La *segmentación* o *fragmentación* permite la descomposición de los elementos de una determinada región en la imagen. El resultado de esta operación facilita la detección de

las zonas de texto y las zonas de figuras. Como parte del proceso de *extracción de características* se encuentran varios métodos, entre los más utilizados está el *adelgazamiento* de los elementos de la imagen, que consiste en borrar los puntos del borde de las regiones detectadas de manera que esta conserve su topología, como resultado de este proceso se obtiene el esqueleto de los elementos para facilitarle al sistema el reconocimiento de estos elementos, el reconocimiento se puede realizar mediante la comparación con patrones definidos para cada una de las letras y caracteres existentes. En esta etapa se deberán comparar los caracteres obtenidos con los caracteres teóricos almacenados en una base de datos. Este paso es muy importante, ya que el buen funcionamiento del sistema OCR se debe en gran medida a una buena definición de esta etapa. Por último se realiza la *composición del texto* de salida que debe ser un documento digital en formato reconocible por varios editores de texto.

1.3 Análisis de sistemas homólogos

1.3.1 OmniPage

OmniPage Ultimate es una aplicación para la conversión y digitalización de documentos. Permite trabajar con grandes volúmenes de documentos desde varios dispositivos, archivar y convertir documentos en sitios de almacenamiento en la nube. Con OmniPage se pueden convertir documentos de papel y de formato PDF, formularios e imágenes de cámaras digitales en archivos electrónicos que puede modificar, compartir y escuchar con una voz clara y natural en dispositivos móviles. Los documentos convertidos tienen la misma apariencia que el original, incluso las columnas, las tablas, las viñetas y los gráficos, y son fáciles de modificar. OmniPage incluye el reconocimiento de idiomas basado en los alfabetos cirílico, griego y latino así como también el de los idiomas chino, japonés y coreano. (3)

1.3.2 ABBYY FineReader

ABBYY FineReader 11 es una aplicación profesional idónea para aumentar la productividad en pequeñas y medianas empresas o departamentos, así como en instituciones gubernamentales y educativas, ya sea para extraer texto de documentos en papel o para introducir documentos en archivos que permitan búsquedas. ABBYY FineReader hace que trabajar con documentos sea fácil y eficiente, y fomenta un mayor rendimiento de sus procesos empresariales. Proporciona resultados rápidos en 189 idiomas, en cualquier combinación. El uso intuitivo y la automatización con un solo clic le permiten hacer más con menos pasos. Admite una amplia gama de formatos de salida. (4)

1.3.3 NSOCR

Nicomsoft OCR es un kit de desarrollo de software de OCR. Nicomsoft es una compañía de software de propiedad privada fundada en 1999, que se especializa en el desarrollo de las bibliotecas eficientes para las empresas y aplicaciones de utilidad para los usuarios finales. Garantiza el reconocimiento fiable de cualquier fuente, incluso en imágenes de baja calidad. Tiene un excelente algoritmo de análisis para imágenes con mala calidad y distorsionada. Utiliza diccionarios para el reconocimiento de varios idiomas como Inglés, alemán, francés, español y el ruso. (5)

1.3.4 Top OCR

TopOCR está diseñado para ser simple y fácil de usar para la digitalización de documentos. Es un software de procesamiento de imágenes avanzado que puede arreglar los problemas causados por el bajo contraste y condiciones de poca luz. Tampoco hay necesidad de preocuparse si sus documentos están boca abajo o de lado. TopOCR puede hacerse cargo de esto con su corrección de la orientación automática de texto que transforma las imágenes a su orientación. (6)

OCR rápido y preciso, puede leer 11 idiomas. El modo de impresión pequeño mejora la precisión del OCR en las letras pequeñas o en las imágenes capturadas por las cámaras de baja resolución. Puede eliminar objetos de fondo y el ruido de las imágenes. El contraste de ecualización mejora las imágenes de bajo contraste. (6)

1.3.5 Tesseract

Tesseract es un motor OCR libre. Fue desarrollado originalmente por Hewlett-Packard como software propietario entre 1985 y 1995. Tras diez años sin ningún desarrollo, fue liberado como código abierto en el año 2005 por Hewlett-Packard y la Universidad de Nevada, Las Vegas. Tesseract es desarrollado actualmente por Google y distribuido bajo la licencia Apache, versión 2.0. Está considerado como uno de los motores OCR libres con mayor precisión disponible actualmente. (7)

Es un **motor** de Reconocimiento Óptico de Caracteres (OCR) que está disponible para múltiples sistemas operativos. En éste motor de OCR se basan múltiples programas, que son los que realmente proporcionarán una interfaz al usuario para sacar partido a Tesseract-OCR. Por tanto conviene tener claro que si sólo se instalara el motor Tesseract-OCR sólo se podría interactuar con él a base de instrucciones que se deberían escribir desde la línea de comandos de una consola. (8)

1.3.6 GOCR

GOCR es un programa libre para el reconocimiento óptico de caracteres que funciona en Windows, Linux y MacOS. Es un programa que permite reconocer caracteres contenidos dentro de una imagen. Permite seleccionar para la salida el formato que se desee. Soporta la mayoría de los formatos de imagen. (9)

1.3.7 Tabla de comparación de los sistemas analizados

Luego de realizar un análisis a los sistemas descritos anteriormente se confeccionó la siguiente tabla de comparación (Tabla 1), con el objetivo de enmarcar y centrar el estudio en las principales características de estos sistemas. Para realizar esta tabla se tuvo en cuenta el tipo de software (libre o propietario), las ventajas y desventajas de su uso y algunos de los formatos que admiten como entrada y salida.

	Software Libre		**Características más destacadas**		**Formatos de entrada y salida**
	Si	**No**	**Ventajas**	**Desventajas**	
Omnipage v18		X	• Reconocimiento de caracteres fiable. • Procesos bien definidos y personalizables. • Conversión a numerosos formatos. • Integración con programas de terceros. (3)	• Ejecución lenta. • Sistema Obsoleto. (3)	**Entrada**: TIF, PCX, DCX, BMP, JPG, JP2, HDP, GIF, PNG, XIF, MAX, PDF, XPS. **Salida**: WAV, DOC, OPF, HTM, PDF, XPS, XSN, XLSX, XLS, PPTX, PPT, RTF, LIT, DOC, XML, DOCX, TXT, CSV, WPD. (3)
ABBY FineReader v11		X	• Motor OCR de alta calidad. • Interfaz limpia e intuitiva. • Soporte para 186 idiomas. • Corrector ortográfico. (4)	• Problemas de compatibilidad con algunos escáneres concretos. (4)	**Entrada**: BMP, JPG, JP2, HDP, GIF, PNG, XIF, PDF. **Salida:** DOC, DOCX, PDF, RTF, TXT, XLS, XSLX, HTML, CSV, PPT. (4)
NSOCR		X	• Interfaz multilingüe. • Reconocimiento de caracteres especiales. • Gran número de formatos soportados. (5)	• Problemas con algunos caracteres específicos como la i. (5)	**Entrada:** PDF, BMP, JPEG, PNG, TIFF, GIF. **Salida:** DOC, DOCX, PDF, RTF, TXT, HTML, PDF (5)

TopOCR	X		• Resultado inmediato. • Dos paneles para comparar imagen y texto. • Opción de escoger el idioma del texto a importar. (6)	• No reconoce bien textos con fondos de colores. • No reconoce bien textos de color. (6)	**Entrada**: BMP, GIF, JPG, TIFF. **Salida:** DOC, PDF. (6)
Tesseract	X		• Excelente precisión de reconocimiento. • Buen tiempo de respuesta. • Multiplataforma. • Puede convertir los textos en más de 40 idiomas. (8)	• No posee interfaz gráfica. • Procesa solo una imagen a la vez. • El fichero de salida no permite la corrección ortográfica. • La imagen de entrada debe tener buena calidad. • El idioma por defecto es inglés, para procesar documentos en otros idiomas, es necesario descargarlos e instalárselos. (8)	**Entrada**: JPG, PDF, PNG, JPEG. **Salida:** DOC, DOCX, PDF. (8)
GOCR	X		• Buena precisión en el reconocimiento. • Se puede utilizar con diferentes *front-end*[1]. • Multiplataforma. (9)	• Presenta dificultades con letras cursivas. • Presenta dificultades con los textos manuscritos. • Las imágenes con inclinación no las procesa correctamente. • Tiempo de respuesta lento. (9)	**Entrada**: JPG, PDF, PNG, JPEG, TIFF, GIF, PNM. **Salida**: DOC, DOCX, PDF, RTF, TXT, XLS, XSLX, HTML, CSV. (9)

Tabla 1 "Comparación de sistemas homólogos"

[1] Front-end: parte del software con la que el usuario interactúa.

1.3.8 Conclusiones del análisis de las soluciones existentes

Con el análisis realizado a los sistemas homólogos se obtuvieron las siguientes conclusiones:

- El tipo de software para el reconocimiento óptico de caracteres más utilizado es el propietario.
- Son pocas las empresas que invierten y desarrollan sistemas para el reconocimiento óptico de caracteres, por lo que no existe suficiente información sobre ellos.
- La mayoría de los sistemas para el reconocimiento óptico de caracteres implementados con tecnologías libres son multiplataforma, no ocurriendo lo mismo con la mayoría de los sistemas privativos.
- Los sistemas analizados desarrollados con tecnologías libres no son confiables porque presentan errores en el reconocimiento de caracteres y a excepción de Tesseract la mayoría no cuenta con respaldo.

Con el análisis realizado anteriormente se determinó que la mayoría de los sistemas existentes implementados con tecnologías libres no garantizan un correcto reconocimiento óptico de caracteres. Además no se tiene acceso al código fuente de los sistemas propietarios por lo que no se les puede realizar modificaciones para adaptarlos a las tecnologías libres, y de los software libres que se analizaron no se tienen los conocimientos necesarios para aplicar modificaciones sobre los mismos, ya que la mayoría de estos no cuentan con una documentación suficiente que permita modificar su código fuente. Por lo expuesto anteriormente el equipo de trabajo decidió implementar su propio sistema para el reconocimiento óptico de caracteres.

1.5 Aplicaciones de los sistemas OCR.

1.5.1 Reconocimiento de texto manuscrito.

En el caso de reconocimiento de escritura manuscrita a la hora de corrección de exámenes, existe la posibilidad, añadiendo un listado de léxico (nombres y apellidos), de acercarse al 100% de acierto. A través de las casillas de respuesta se pueden reconocer palabras, como nombres de países, nombres de regiones, marcas comerciales, en resumen, todo aquello que pueda ser integrado en una lista de palabras.

Por otro lado, se puede llegar a comprender una frase cuando se ha terminado de leer. Esto implica una operación de niveles morfológicos, léxico y sintáctico que se consigue mediante el reconocimiento del

habla continua. Para llevar a cabo esa metodología, se utilizan algoritmos robustos que utilizan una segmentación previa. (10)

1.5.2 Reconocimiento de matrículas

Una de sus aplicaciones son los radares. Estos deben ser capaces de localizar una matrícula de un vehículo con condiciones de iluminación, perspectiva y entorno variables. En la etapa de segmentación, se buscan texturas similares a la de una matrícula y se aísla el área rectangular que forma la matrícula. Finalmente, se aplica un proceso de clasificación múltiple sobre el conjunto de píxeles pertenecientes a la matrícula, proporcionando una cadena de caracteres que se tienen que ajustar a un modelo conocido: el formato de una matrícula. Si aparece algún error, es corregido. (10)

1.5.3 Indexación en bases de datos

Con el aumento de información publicada que ha tenido lugar en los últimos años, cada vez son más los métodos que se utilizan para organizar todo este material almacenado en bases de datos. Uno de estos contenidos son las imágenes. Una de las formas más corrientes de buscar imágenes es a partir de metadato introducido manualmente por los usuarios. Actualmente han aparecido buscadores que proporcionan la posibilidad de buscar imágenes mediante el texto que aparecen en ellas, como el buscador DIRS(*Document Image Retrieval System*) que mediante un algoritmo de Reconocimiento Óptico de Caracteres, extrae el texto que aparece en la imagen y lo utiliza como metadato que podrá ser utilizado en las búsquedas. Esta tecnología proporciona una posibilidad en la búsqueda de imágenes y demuestra que el OCR aún puede dar mucho de sí. (10)

1.5.4 Reconocimiento de datos estructurados con OCR Zonal

Se usa para digitalizar de forma masiva grandes cantidades de documentos estructurados o semi-estructurados (facturas, nóminas, albaranes, pólizas, justificantes bancarios, etc.), catalogando automáticamente los documentos con los metadatos obtenidos y archivándolos en formato digital de forma indexada para facilitar su posterior búsqueda. Tiene el inconveniente de que es necesario diseñar previamente las plantillas, pero con una buena configuración se ahorra mucho tiempo en el proceso de digitalización. (10)

1.6 Metodología de desarrollo de software

En el presente trabajo se realizó un estudio para seleccionar la metodología de desarrollo de software a utilizar en la implementación del sistema, quedando como resultado de este estudio lo siguiente:

Piattini define a la metodología de desarrollo de software como “un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. (11)

Metodologías ágiles	Metodologías tradicionales
Preparadas para cambios en el proyecto	Cierta resistencia a los cambios
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños(menos de 10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Muchos artefactos
Pocos roles	Muchos roles

Tabla 2 "Comparación de los tipos de metodologías de desarrollo de software"

Para la selección de la metodología de desarrollo se tuvo en cuenta que:

- Los requisitos pueden sufrir cambios durante el proceso de implementación.
- El cliente, en este caso la biblioteca, está en constante comunicación con el equipo de desarrollo.
- Se cuenta con un equipo de desarrollo pequeño, debido a que está integrado por 2 estudiantes.
- Se conoce que la primera versión funcional del sistema se puede realizar en pocos meses.

Analizando lo expuesto anteriormente se decide que la metodología a utilizar debe ser ágil, ya que es la que más se ajusta a las características y propósito del trabajo.

Definición de la metodología de desarrollo de software

Para definir la metodología de desarrollo a utilizar se decide realizar un estudio a dos de ellas: Programación extrema (XP, por sus siglas en inglés) y Scrum.

Scrum

Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos años. Está especialmente indicada para proyectos con rápidos cambios de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (12). Véase en la Ilustración 2 el ciclo de vida de Scrum.

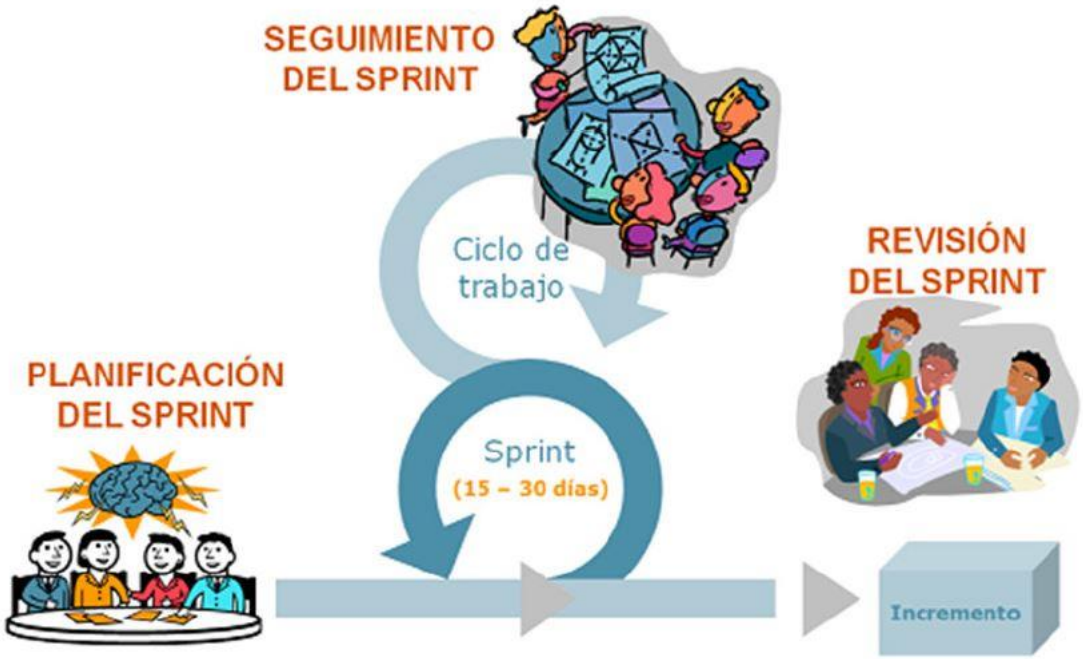

Ilustración 2 "Ciclo de vida de Scrum"(13)

Programación extrema (XP)

Es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (12). Véase en la Ilustración 3 el ciclo de vida de XP.

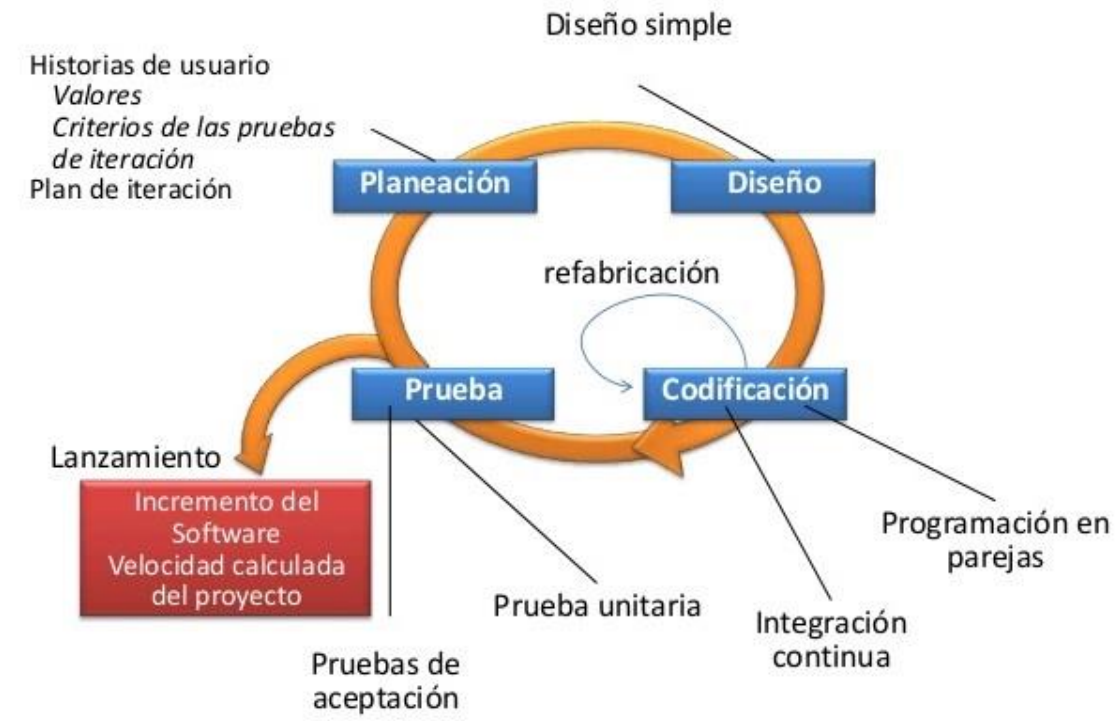

Ilustración 3 "Ciclo de vida de XP". (14)

De lo expuesto anteriormente resultó la siguiente tabla de comparación:

Criterio de comparación	XP	Scrum
Tamaño de los proyectos	Pequeños y medianos	Pequeños, medianos y grandes
Tamaño del equipo	Menores de 10	Múltiples equipos menores de 10
Estilo de desarrollo	Iterativo y rápido	Iterativo y rápido
Mecanismos de abstracción	Orientado a objetos	Orientados a objetos

Proceso de desarrollo	• Diseño simple. • Pruebas. • Refactorización • Programación en pares. • Integración continua. • Cliente en el equipo de desarrollo.	• Planes de lanzamientos • Distribución, revisión y ajuste de los estándares de producto • Sprint • Revisión del Sprint • Cierre

Tabla 3 "Comparación entre XP y Scrum"

Luego de realizar un análisis de las características particulares de cada una de las metodologías de desarrollo de software mencionadas anteriormente y en concordancia con la solución que se propone como resultado de esta investigación se decide utilizar XP como metodología de desarrollo de software. XP permite simplicidad, la comunicación entre los usuarios finales y los desarrolladores, permite además la retroalimentación o reutilización de código, está definida para equipos pequeños de desarrollo, tiene prioridad de satisfacer al cliente mediante un desarrollo iterativo y rápido y está abierta a cambios en los requerimientos.

Una de las características esenciales de XP es la utilización de historias de usuario. La historia de usuario es la técnica utilizada por esta metodología para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. (12)

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento estas pueden romperse y reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (12)

El ciclo de vida ideal de XP consiste de seis fases (12)

Fase I: Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. (12)

Fase II: Planificación de la Entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. (12)

Fase III: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible, ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. (12)

Fase IV: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento). (12)

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la

puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura. (12)

Fase VI: Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo. (12)

1.7 Lenguajes

1.7.1 Lenguaje de modelado

El lenguaje de modelado utilizado será UML, el cual es un lenguaje diseñado para visualizar, especificar, construir y documentar software orientados a objetos.

1.7.2 Lenguaje de programación

Un **lenguaje de programación** es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos, reglas sintácticas y reglas semánticas que definen su estructura, el significado de sus elementos y expresiones. (13)

El lenguaje de programación que se utilizará para el desarrollo del sistema es Java, este es un lenguaje muy potente y está diseñado para la implementación de sistemas libres. Además se decidió usar este lenguaje porque los algoritmos que realizan el proceso de clasificación que se van a utilizar en el software están implementados en la herramienta Weka y se encuentran desarrollados en Java.

Java es un lenguaje orientado a objeto, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portable, de alto desempeño, multihilos y dinámico. (14)

Java es orientado a objetos porque da buen soporte a las técnicas de desarrollo de la programación orientada a objetos (POO por sus siglas) y a la reutilización de componentes de software. Es distribuido

porque se ha diseñado para trabajar en ambientes de redes y contienen una gran biblioteca de clases. Mediante el código Java se pueden manipular recursos URL con la misma facilidad que C y C++ utilizan recursos locales (archivos). Es un lenguaje interpretado**,** ya que el compilador Java traduce cada fichero fuente de clases a código de bytes (*Bytecode*), que puede ser interpretado por todas las máquinas que den soporte a un visualizador que funcione con Java. Este *Bytecode* no es específico de una máquina determinada, por lo que no se compila y enlaza como en el ciclo clásico, sino que se interpreta. Es robusto porque el código Java no se quiebra fácilmente ante errores de programación. Así la flexibilidad que existe en la declaración y manejo de tipos en C y C++ se torna en restricciones en Java, donde no es posible la conversión forzada (*cast*) de enteros en punteros y no ofrece soporte a los punteros que permitan saltarse reglas de manejo de tipos. Así en Java no es posible escribir en áreas arbitrarias de memoria ni realizar operaciones que corrompan el código. En resumen se eliminan muchas de las posibilidades de "trucos" que ofrecían C y C++. Es seguro porque las mismas características antes descritas que evitan la corrupción de código evitan su manipulación. Es dinámico porque al contrario de C++ que exige que se compile de nuevo la aplicación al cambiar una clase madre, Java utiliza un sistema de interfaces que permite aligerar esta dependencia. Como resultado, los programas Java pueden permitir nuevos métodos y variables en un objeto de biblioteca sin afectar a los objetos dependientes. (14)

1.8 Herramientas utilizadas para el desarrollo del sistema

1.8.1 Herramienta de modelado

Las herramientas de modelado de sistemas informáticos, son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Permiten crear un "simulacro" del sistema, a bajo costo y riesgo mínimo. (15)

Las buenas herramientas de modelado cumplen con las siguientes características:

- Permiten una visión descendente del sistema.
- Poseen componentes gráficos con algo de apoyo textual.
- El modelo resultante debe ser transparente (fácil de comprender).
- Poseen mínima redundancia (el aumento de redundancia, disminuye la transparencia del modelo y aumenta las tareas de mantenimiento).

La herramienta de modelado seleccionada fue Visual Paradigm for UML en su versión 8.0, debido a que es una herramienta para el desarrollo de aplicaciones utilizando modelado UML.

Esta herramienta es ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. (16)

Visual Paradigm ha sido concebido para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. (17)

Se caracteriza por:

1. Disponibilidad en múltiples plataformas (Windows, Linux).
2. Diseño centrado en casos de uso.
3. Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
4. Capacidades de ingeniería directa e inversa.
5. Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
6. Licencia: gratuita y comercial.
7. Varios idiomas.
8. Generación de código para Java y exportación como HTML.
9. Fácil de instalar y actualizar.
10. Soporte de UML versión 2.1.
11. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
12. Modelado colaborativo con CVS y Subversión (control de versiones).
13. Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
14. Generación de código - Modelo a código, diagrama a código.
15. Editor de detalles de casos de uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
16. Soporte ORM - Generación de objetos Java desde bases de datos.

17. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
18. Generador de informes.
19. Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
20. Integración con Visio - Dibujo de diagramas UML con plantillas (*stencils*) de Microsoft Visio.
21. Editor de figuras.

Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software. (17)

1.8.2 IDE

Un **entorno de desarrollo integrado** (**IDE,** por sus siglas en inglés), es una aplicación de software, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDEs tienen auto-completado inteligente de código. (18)

Para la implementación del sistema de reconocimiento óptico de caracteres los IDE que se tuvieron en cuenta fueron NetBeans y Eclipse.

Eclipse

Es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Sin embargo, también se puede usar para otros tipos de aplicaciones cliente. (19)

El IDE Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera

para componentes de software. Adicionalmente al permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y sistemas de gestión de base de datos. (19)

Eclipse dispone de un editor de texto con un analizador sintáctico. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (*wizards*) para creación de proyectos, clases, test, refactorización, etc... La última versión estable de Eclipse hasta febrero del 2015 es la 4.4.2. (19)

NetBeans

Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (20)

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (actualmente Sun Microsystems es administrado por *Oracle Corporation*). (20)

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs[2] de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. (20)

[2] API: La interfaz de programación de aplicaciones (API, por sus siglas en inglés), es el conjunto de subrutinas, funciones, procedimientos o métodos, en la programación orientada a objetos, que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

El NetBeans es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente. (20)

La plataforma ofrece servicios reusables comunes para las aplicaciones de escritorio, permitiendo a los desarrolladores centrarse en la lógica de sus aplicaciones. Algunas de las características de la aplicación son:

- Gestión de la interfaz de usuario (menús y barras de herramientas)
- Gestión de configuración de usuario
- Gestión de almacenamiento (guardar o cargar algún tipo de dato)
- Gestión de ventana
- Marco Asistente
- Biblioteca de clases
- Herramientas de desarrollo integrado

NetBeans IDE es libre, de código abierto, multiplataforma con soporte integrado para el lenguaje de programación Java. La última versión estable de NetBeans hasta noviembre del 2014 es la 8.02. (20)

Selección del IDE

Luego de realizar un análisis a lo expuesto anteriormente se decidió utilizar como IDE de desarrollo el **NetBeans** en su versión 8.0. Se escogió este entorno de desarrollo integrado porque soporta la implementación de todo tipo de aplicaciones desarrolladas en java y brinda muchas facilidades al programador. Se tuvo en cuenta además que el equipo de desarrollo ha estado trabajando con esta herramienta, por lo que tiene experiencia y dominio sobre la misma.

1.8.3 Otras herramientas utilizadas

También se utiliza la herramienta **WEKA** en su versión 1.4 para realizar pruebas a la combinación de los algoritmos de extracción de características con los clasificadores que esta tiene implementados. También se utiliza como framework de desarrollo.

Weka (*Waikato Environment for Knowledge Analysis*, en español entorno para análisis del conocimiento de la Universidad de Waikato) es una plataforma de software para el aprendizaje automático y la minería de datos escrito en Java y desarrollado en la universidad de Waikato. Weka es una herramienta de software libre distribuida bajo la licencia GNU-GPL. (21)

El paquete Weka contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades. (21)

Los puntos fuertes de Weka son:

- Está disponible libremente bajo la licencia pública general de GNU.
- Es portable porque está completamente implementado en Java y puede correr en casi cualquier plataforma.
- Contiene una extensa colección de técnicas para pre-procesamiento de datos y modelado.
- Es fácil de utilizar por un principiante gracias a su interfaz gráfica de usuario.

Weka soporta varias tareas estándar de minería de datos, especialmente, pre-procesamiento de datos, clustering, clasificación, regresión, visualización y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un fichero plano (*flat file*) o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales, aunque también se soportan otros tipos). Weka también proporciona acceso a bases de datos vía SQL gracias a la conexión JDBC (*Java Database Connectivity*) y puede procesar el resultado devuelto por una consulta hecha a la base de datos. (21)

Bibliotecas de clases

iText en su versión 5.5.5: Es una biblioteca de clases para el manejo y la creación de archivos PDF’s con java. (22)

OpenCV en su versión 2.4: Es una biblioteca de clases abierta desarrollada por Intel. Proporciona funciones de alto nivel para el procesado de imágenes. OpenCV ofrece muchos tipos de datos de alto nivel como árboles, grafos, matrices, etc. (22)

1.9 Conclusiones parciales

En el presente capítulo se estudiaron los principales conceptos asociados al objeto de estudio, lo que permitió conocer a nivel de detalle el proceso que realizan los sistemas OCR. Se analizaron varios de estos sistemas lo que posibilitó conocer el estado actual de los mismos, así como sus principales características. El análisis de las diferentes metodologías de desarrollo de software permitió concluir que la programación extrema XP es la que más se adecúa para el desarrollo del sistema. La selección del lenguaje de programación Java permitirá la implementación de un sistema potente, apoyándose en NetBeans 8.0 como IDE, aprovechando todas las facilidades que brinda a los programadores que utilizan este lenguaje. A su vez, la herramienta CASE Visual Paradigm for UML 8.0 permitirá realizar un diseño adecuado que ayudará a los desarrolladores a agilizar el proceso de implementación.

Capítulo 2 “Análisis y Diseño”

2.1 Introducción

Con el desarrollo del presente capítulo se pretenden identificar las funcionalidades del sistema para el reconocimiento óptico de caracteres, así como los requisitos de hardware que se necesitan para hacer uso del mismo. Se realizará la planificación del tiempo de desarrollo y se crearán los diferentes artefactos que ayudarán a entender el funcionamiento del sistema. Además se hace un estudio de cada fase del reconocimiento óptico de caracteres y se expone la propuesta de solución.

2.2 Modelo del Dominio

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. Es una representación visual de clases conceptuales o de objetos reales en el dominio de interés. (24)

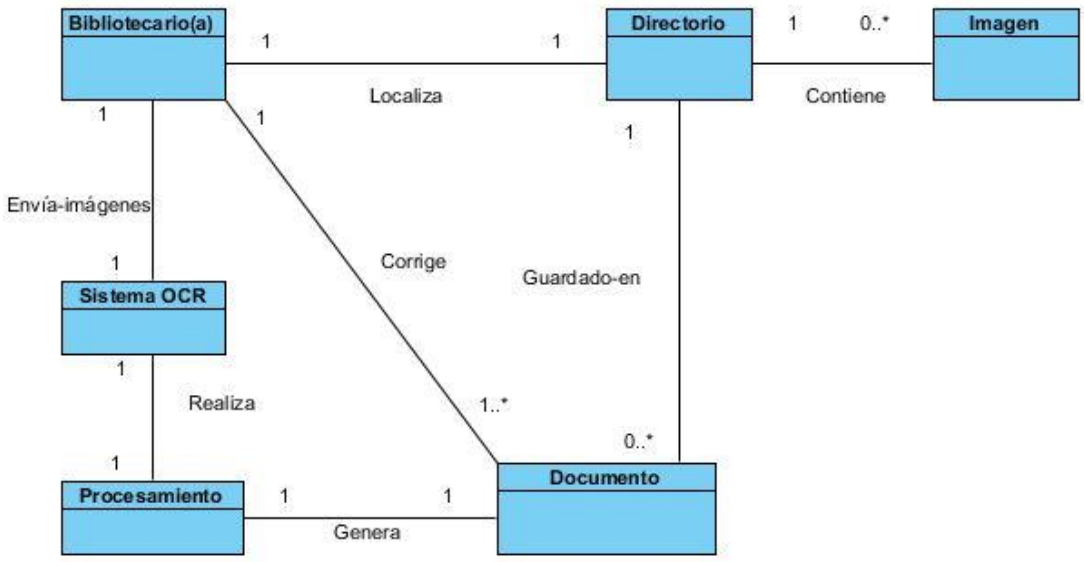

Ilustración 4 "Diagrama del modelo del dominio"

2.2.1 Descripción de los principales conceptos

Bibliotecario(a): Es la persona encargada de realizar la búsqueda del directorio donde se encuentran las imágenes, enviar o cargar la imagen en la aplicación y de la revisión del documento que genera la aplicación.

Sistema OCR: Es el que contiene las bibliotecas de clases y algoritmos para el tratamiento de las imágenes que contienen texto.

Procesamiento: Es el proceso que se le realiza a las imágenes para extraerles el texto que contienen.

Documento digital: Es el resultado de haber procesado una imagen que contiene texto.

Directorio: Lugar donde se guardan las imágenes para ser procesadas y los documentos digitales resultantes.

Imagen: Es el resultado de haber escaneado un documento en formato duro.

2.3 Captura de requisitos del sistema

Según *IEEE, 1990* un requisito es una ¨ *Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente*.¨

Aspectos importantes

- Los requisitos son independientes del diseño, deben enseñar qué debe hacer el sistema más que el cómo debe hacerse.
- No hay una única manera de expresar los requisitos de un sistema.

2.3.1 Requisitos funcionales (RF)

RF1: Cargar imagen: Permite cargar imágenes a la aplicación.
RF2: Guardar imagen: Permite guardar la imagen mostrada en el panel de la aplicación.
RF3: Preparar imagen: Aplica el pre-procesamiento a la imagen.
RF4: Segmentar imagen: Permite segmentar la imagen.

RF5: Extraer características: Permite extraer las características de los caracteres contenidos en la imagen.

RF6: Identificar caracteres: Permite realizar el reconocimiento de los caracteres.

RF7: Conformar y exportar el texto: Conforma el texto con los caracteres encontrados en la imagen y lo exporta en el formato de texto seleccionado.

2.3.2 Requisitos no funcionales (RNF)

Requisitos de Hardware

RNF1: Microprocesador: Pentium IV o superior

RNF2: Memoria RAM: 1 GB o superior

Requisitos de Software

RNF3: Multiplataforma (cualquier distribución de Linux y Windows XP o superior)

RNF4: JRE v1.8

RNF5: Biblioteca de clases: OpenCV v2.4

Requisitos de diseño e implementación

RNF6: Lenguaje de programación: Java

RNF7: IDE: NetBeans 8.0

2.3.3 Historias de Usuarios

A continuación se presenta la historia de usuario perteneciente al primer requisito funcional, el resto de las historias de usuarios se pueden consultar en los anexos.

Historia de Usuario	
Número: HU_1	**Usuario:** bibliotecaria.
Nombre historia: Cargar y guardar imagen.	
Prioridad en el negocio: media	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: David Caballero.	

Descripción Permite cargar imágenes a la aplicación para ser procesadas.
Observaciones:

Tabla 4 "Historia de usuario 1: Cargar y guardar imagen"

2.4 Planificación

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente y a los programadores. La planificación es una fase corta, en la que el cliente y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario. El resultado de esta fase es un plan de entregas.

2.4.1 Velocidad del proyecto

Es una medida de la capacidad que tiene el equipo de desarrollo para evaluar las historias de usuario en una determinada iteración. Esta medida se calcula totalizando el número de historias de usuario realizadas en una iteración (12). Véase la Tabla 5 para ver la velocidad del proyecto.

	Iteración 1	Iteración 2	Iteración 3	Iteración 4	Iteración 5	Iteración 6
Horas	120	120	120	120	120	120
Semanas	3	3	3	3	3	3
Horas semanales	40	40	40	40	40	40
Historias de usuario	1	1	1	1	1	1

Tabla 5 "Velocidad del proyecto"

2.4.2 Plan de entregas

El cronograma de entregas establece las historias de usuario que serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta reunión “Juego de planeamiento” (“Planning game”), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente. Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. (22)

Después de realizadas las historias de usuarios se procede a realizar la planificación de la fase de implementación. Las historias de usuarios se implementarán en el mismo orden en que aparecen. En la fase de implementación cada tres semanas se tendrá un producto para entregar al cliente. Véase en la Tabla 6 como queda conformado el plan de entregas.

Iteración	Entregable	Fecha inicio	Fecha fin
1	Ventana para cargar o guardar la imagen.	12/1/2015	1/2/2015
2	Imagen pre-procesada.	2/2/2015	22/2/2015
3	Imagen segmentada.	23/2/2015	15/3/2015
4	Características de los elementos de la imagen.	16/3/2015	5/4/2015
5	Caracteres identificados.	6/4/2015	26/4/2015
6	Texto conformado y exportado.	27/4/2015	17/5/2015

Tabla 6 "Plan de entregas"

2.4.3 Plan de iteraciones

El objetivo de la planificación de la iteración es plantear las historias de usuarios que se desarrollaran en cada iteración. Las iteraciones son de una longitud fija, de una a tres semanas de largo. Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteración son las historias de usuario y la velocidad del proyecto principalmente. Al final de cada iteración se debe tener un entregable para mostrarle al cliente y este vea el avance del proyecto y pueda realizarle las pruebas de aceptación adecuadas. (23)

El proyecto se realizará en 18 semanas divididas en 6 iteraciones y todas con una duración de 3 semanas. Véase en la Tabla 7 como queda conformado el plan de iteraciones.

Iteración	No. HU	Historia de usuario	Semanas estimadas
1	HU 1	Cargar y guardar imagen.	3
2	HU 6	Preparar imagen.	3
3	HU 7	Segmentar imagen.	3
4	HU 8	Extraer caracteres.	3
5	HU 9	Identificar caracteres.	3
6	HU 10	Conformar y exportar texto.	3

Tabla 7 "Plan de iteraciones"

2.5 Análisis de las fases de un OCR

2.5.1 Propuesta según diferentes autores

Fase de pre-procesamiento

Los autores Christos Nikolaos, Vassili Loumos, Ioannis E. Anagnostopoulos y Eleftherios Kayafas en (24) para realizar el pre-procesamiento llevan la imagen a escala de grises y luego le realizan el proceso de binarización mediante el método de Sauvola que aparece explicado en (25).

El autor Ved Prakash Agnihotri en (26) para realizar el pre-procesamiento lleva la imagen a escala de grises, luego le realiza el proceso de binarización y por último hace la detección de bordes. En el artículo no se especifica como calcula el umbral para realizar la binarización.

Los autores Julinda Gllavata, Ralph Ewerth y Bernd Freisleben en (27) para realizar el pre-procesamiento de la imagen la llevan a escala de grises, a continuación la convierten al espacio de color YUV y por último el valor de umbral de luminancia se aplica para difundir los valores de luminancia en toda la imagen y aumentar el contraste entre las regiones posiblemente interesantes y el resto de la imagen, en el artículo no se especifica el valor del umbral utilizado.

Fase de segmentación

Los autores Christos Nikolaos, Vassili Loumos, Ioannis E. Anagnostopoulos y Eleftherios Kayafas en (24) para realizar la segmentación aplican la técnica *sliding concentric windows* (SCWs por sus siglas) que se utiliza para la detección de las regiones de interés en la imagen, el funcionamiento de esta técnica aparece explicado en el documento referenciado.

El autor Ved Prakash Agnihotri en (26) para realizar esta fase segmenta en caracteres aislados mediante la asignación de un número a cada caracter utilizando un proceso de etiquetado de componentes conexos. Este etiquetado proporciona información sobre el número de caracteres en la imagen. Cada caracter individual cambia de tamaño uniforme a 90x60 píxeles para el posterior reconocimiento.

Los autores Julinda Gllavata, Ralph Ewerth y Bernd Freisleben en el artículo (27) realizan la segmentación mediante las propiedades geométricas de los caracteres (alto, ancho), esto se hace para descartar aquellas regiones cuyas características geométricas no caen en el rango predefinido de valores. Luego

generan una imagen de borde binario a partir de la imagen de bordes, borrando todos los píxeles fuera de los cuadros de texto predefinidos.

Fase de extracción de características

El autor Dinesh Dileep en (28) para realizar la extracción de características define un universo de discurso para cada caracter que no es más que una matriz ajustada al esqueleto del caracter una vez definido esto para todos los caracteres de la imagen se aplica el método ZONING que se explica en el artículo referenciado. Mediante este método se divide la imagen en 9 ventanas de igual tamaño, se extraen las características de cada zona de manera individual, con esto se obtiene información acerca de los detalles del esqueleto de los caracteres.

Los autores Jagruti Chandarana y Mayank Kapadia en (29) utilizan la matriz de píxeles de la imagen binarizada como vector de características.

Los autores Tanzila Saba, Amjad Rehman y Ghazali Sulong en (30) para realizar la extracción de características delimitan las regiones de los caracteres y conforman una matriz donde toman valor “1” (negro) los pixeles que se encuentran en un primer plano y “0” (blanco) los que están en el fondo, para obtener las características el carácter es dividido en rectángulos de un número específico de filas y columnas, las características extraídas se almacenan en el vector de características.

Fase de reconocimiento de caracteres

Los autores Christos Niquelaos, Vassili Loumos, Ioannis E. Anagnostopoulos y Eleftherios Kayafas en (24) realizan el reconocimiento de caracteres con el uso de redes neuronales artificiales, en este caso entrenan dos redes neuronales, una para el reconocimiento alfabético y otra para el reconocimiento de números.

Los autores Divya Gilly y Kumudha Raimond en (31) para realizar el reconocimiento de caracteres utilizan el método de comparación de plantillas que es una técnica que compara partes de las imágenes una contra la otra. En este método se correlacionan las plantillas con la imagen original. El proceso de correspondencia mueve la imagen de plantilla a todas las posiciones posibles en una imagen de origen más grande y calcula un índice numérico que indica el grado en que la plantilla coincide con la imagen en esa posición. La correlación es una medida de grado en el que dos variables están de acuerdo. Las variables son los valores de los píxeles correspondientes en las dos imágenes (plantilla y fuente). El valor

de correlación está entre -1 y 1. El valor superior de correlación indica que existe una fuerte relación entre las dos imágenes.

Los autores Bharat Bhushan, Simranjot Singh, Ruchi Singla en (32) para realizar el reconocimiento de caracteres proponen la utilización de una red neuronal artificial.

2.5.2 Propuesta de solución del presente trabajo

Fase de Pre-procesamiento

Para realizar la fase de pre-procesamiento primeramente se lleva la imagen a escala de grises utilizando la función *cvtColor* de *OpenCv* en su versión 2.4 empleando la fórmula de la luminancia la cual consiste en multiplicar cada canal de un pixel determinado por ciertos coeficientes como se muestra en la Ecuación 1. En la Ilustración 5 se muestra el resultado de convertir una imagen en RGB a escala de gris, la imagen correspondiente a la letra (A) representa la imagen en RGB y la imagen correspondiente a la letra (B) es el resultado de llevar la imagen a escala de gris.

$$0.299 * \mathbf{R} + 0.587 * \mathbf{G} + 0.114 * \mathbf{B}$$

Ecuación 1 "Conversión a escala de gris"

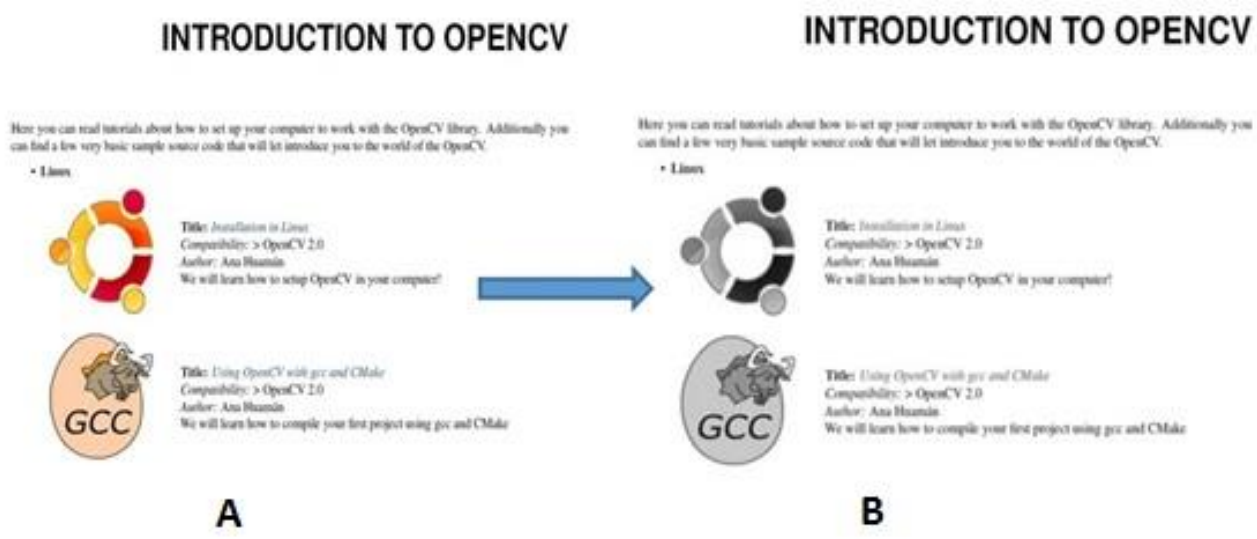

Ilustración 5 "Conversión de una imagen en el espacio de color RGB a escala de gris"

Luego la imagen en escala de gris se convierte a binaria aplicando el método global del valor umbral donde se elige un valor umbral para toda la imagen. El umbral es calculado mediante la suma total de los píxeles de toda la imagen entre la cantidad de píxeles encontrados. La imagen resultante tendrá cada pixel en 0 (*negro*) si el valor encontrado es mayor o igual que el umbral y 255 en otro caso. En la Ilustración 6 se muestra el resultado de este proceso, la imagen correspondiente a la letra (A) representa la imagen en escala de gris y la imagen correspondiente a la letra (B) representa la imagen anterior convertida en una imagen binaria.

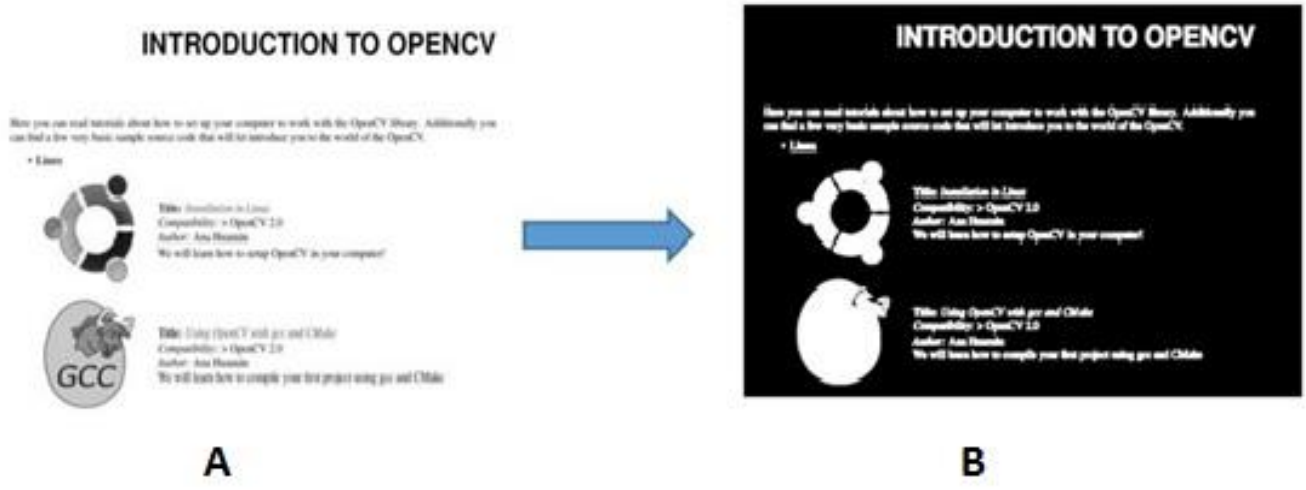

Ilustración 6 "Conversión de una imagen de escala de gris a binaria"

Posteriormente la imagen binaria es adelgazada empleando el algoritmo de *Zhang Suen* (33) para utilizarla en la fase de segmentación. Este método se basa en ir degradando los bordes de los objetos detectados hasta que el grosor de los mismos sea de un pixel de forma tal que estos conserven su topología. En la Ilustración 7 se muestra con la letra (A) la imagen binaria y con la letra (B) se muestra la imagen luego de realizarle el adelgazamiento.

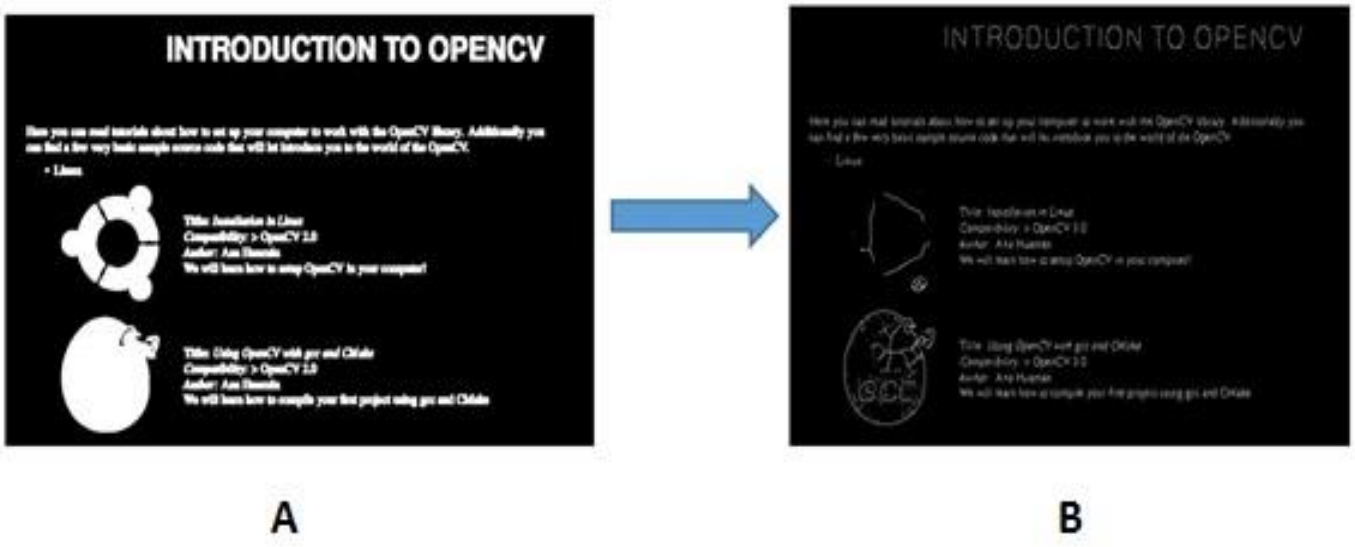

Ilustración 7 "Adelgazamiento de una imagen binaria"

Fase de Segmentación

En esta etapa se utiliza la imagen binaria para detectar los bordes o contornos de los objetos contenidos en la imagen (texto y figuras) aplicando el método de *Canny* (34). Luego de operar con cuyo método se aplica la funcionalidad *boundingRect* de OpenCv2.4 la cual encierra en rectángulos los diferentes contornos o bordes encontrados en la imagen. Posteriormente se recalculan las regiones (rectángulos) solapadas obteniendo como resultado una única región. En la **¡Error! No se encuentra el origen de la referencia.**, la imagen que corresponde a la letra "A" representa las regiones solapadas y la imagen correspondiente a la letra "B" representa el resultado de recalcular las regiones solapadas, obteniéndose como resultado una única región.

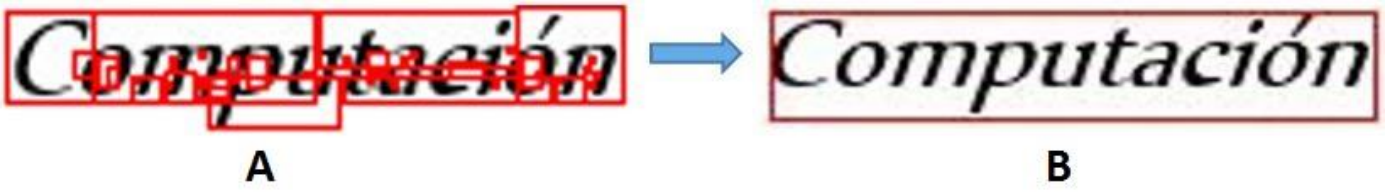

Ilustración 8 "Eliminación de los rectángulos solapados"

Para segmentar el contenido de la imagen se hizo un estudio por casos donde se determinaron tres posibles casos: Caso 1: que la imagen solo contenga texto. Caso 2: que la imagen solo contenga figuras. Caso 3: que la imagen contengo texto y figuras.

Para determinar a qué caso específico pertenece una imagen se realizó un análisis estadístico tomando una muestra de 60 imágenes que solo contenían texto para extraer algunos valores que sirvieron de umbral para la toma de decisiones (ver Tabla 8).

$H_{max}(l_i)$	Alto de la línea de texto de mayor altura en la página p_i.
$H(p_i)$	Alto de la página i.
$\mu = \max_{\forall i} \frac{H_{max}(l_i)}{H(p_i)} = 0.152$	Umbral que representa una cota superior de la proporción de una línea de texto con respecto a la altura de la página.
$\rho = \max_{\forall i} H_{max}(l_i) = 45$	Umbral que representa una cota superior de la altura de una línea de texto.

Tabla 8 "Fórmulas utilizadas para calcular los umbrales"

En la **¡Error! No se encuentra el origen de la referencia.** se muestra el pseudocódigo del algoritmo utilizado para determinar a qué caso pertenece la imagen de la página analizada.

Sea:

- $altura = \{a_0, a_1, \ldots, a_n\}$ colección de altura de todos los rectángulos que representan las regiones de los objetos en la página.
- $valorMax = \max_{a_i \in altura} a_i$
- $valorMin = \min_{a_i \in altura} a_i$
- $H(p)$ altura de la página.
- μ umbral que representa una cota superior de la proporción de una línea de texto con respecto a la altura de la página.
- ρ umbral que representa una cota superior de la altura de una línea de texto.

if $(\frac{valorMax}{H(p)} \leq \mu \vee valorMax \leq \rho)$ {

$\mathrm{p} \in Caso1$

} else if$(\frac{valorMin}{H(p)} > \mu \vee valorMin > \rho)$ {

$\mathrm{p} \in Caso2$

}else{

$\mathrm{p} \in Caso3$

}

Ilustración 9 "Algoritmo utilizado para determinar los casos"

Para el caso 3 (texto e imagen), se calcula el umbral de *Otsu* (35) con los valores de las alturas de los rectángulos contenidos en la imagen para determinar un valor que permita decidir según la altura de cada rectángulo cuál pertenece a la región de un texto y cuál pertenece a la región de una figura.

Luego de determinado qué región (rectángulo) pertenece a un texto o a una figura se procede a ordenar los rectángulos que representan regiones de texto de arriba a abajo y de izquierda a derecha. Los rectángulos que representan figuras se colocan al final. Luego auxiliándose de este orden se conforman las líneas de textos y cada rectángulo contenido en las líneas de textos representa una palabra candidata.

Para segmentar los caracteres se utiliza la imagen adelgazada y se toman en cuenta las regiones de las palabras candidatas en dicha imagen (ver **¡Error! No se encuentra el origen de la referencia.**a). Para la detección de las columnas candidatas por donde segmentar los caracteres es necesario tener en cuenta que los caracteres se clasifican en dos tipos. El primer tipo son los caracteres compuestos de bucles o semi-bucles como ‘p’, ‘a’, ‘c’, denominados como caracteres cerrados. El segundo tipo los que no están compuestos por bucles o semi-bucles como ‘u’, ‘v’, ‘n’ son denominados caracteres abiertos. Una

conexión entre dos caracteres se denomina ligadura (ver **¡Error! No se encuentra el origen de la referencia.**).

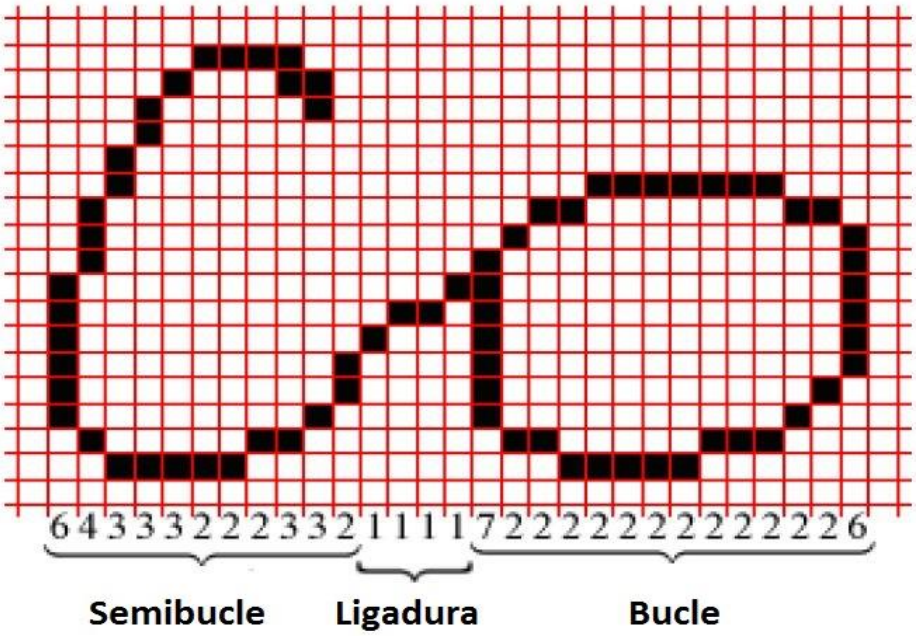

Ilustración 10 "Proyección vertical en imagen adelgazada"

Para determinar las columnas candidatas por donde segmentar se realiza un barrido vertical por la región que contiene a la palabra y se determina para cada columna la cantidad de píxeles que pertenecen al texto (ver **¡Error! No se encuentra el origen de la referencia.**). Las columnas con cero píxeles que pertenecen al texto son columnas que dividen la región por un espacio. Sin embargo las columnas que solo contienen un pixel que pertenece al texto son columnas que pueden pertenecer a una ligadura o a un carácter abierto. Por esta razón las columnas con valor cero se dice que son columnas candidatas con valor de certeza de 100% y las columnas con valor 1 se dicen que son columnas candidatas con valor de certeza de 0% (ver **¡Error! No se encuentra el origen de la referencia.**b). En una etapa posterior se comprueba cuáles columnas con valor de certeza 0% segmentan la palabra de forma correcta.

Luego a partir de una secuencia de columnas candidatas consecutivas o bastantes cercanas se determina una única columna a través de la siguiente expresión: $\frac{(s_i+s_f)}{2}$ donde s_i es el índice de la columna inicial de

la secuencia y s_f es el índice de la columna final de la secuencia. (Ver **¡Error! No se encuentra el origen de la referencia.**c).

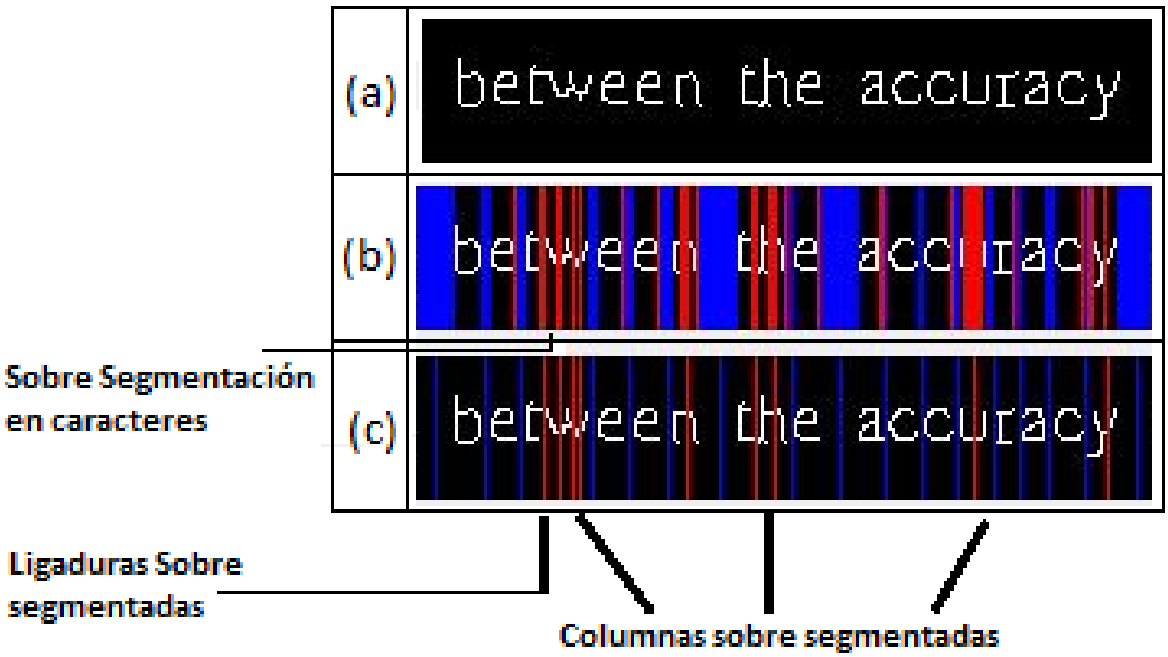

Ilustración 11 "a) Imagen adelgazada, b) secuencia de columnas candidatas para la segmentación de caracteres, c) columnas extraídas de las secuencias de columnas candidatas para la segmentación. Las columnas de color azul representan las columnas candidatas con valor de certeza 100% y las rojas las columnas candidatas con valor de certeza 0%."

Para reconocer la secuencia de columnas candidatas se utiliza el autómata de la Ilustración 12. En este autómata se pasa de un estado a otro con columnas con valores 0, 1 u otro valor. El nodo N1 es el inicial y N7 es el nodo terminal. Cuando el autómata pasa a los nodos m1 o m2 significa que en la secuencia se ha encontrado una columna no candidata. Si la cantidad de columnas no candidatas consecutivas es mayor que un umbral τ especificado entonces se pasa al nodo terminal. Si en la secuencia se encuentra una columna con valor mayor que un umbral σ especificado se pasa inmediatamente al nodo terminal. En el software desarrollado $\tau = 3$ y $\sigma = 3$ esto permite reconocer secuencias de columnas candidatas de la forma: 000111,111, 0020200, etc. Permitiendo reconocer secuencias conectadas que de otra forma

estarían desconectadas por ruidos introducidos en la imagen. El autómata devuelve el índice de una única columna candidata que representa a toda la secuencia.

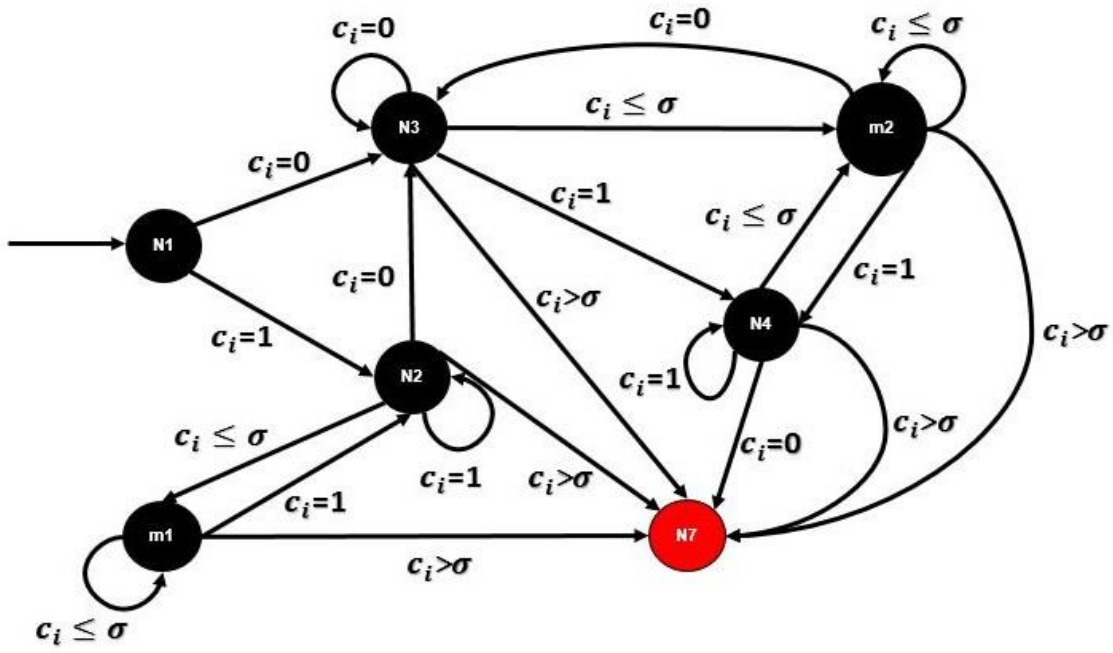

Ilustración 12 "Representación gráfica del autómata empleado para el reconocimiento de las secuencias de columnas candidatas para la segmentación."

Luego se verifica cada columna de corte con 0% de certeza para determinar si segmenta correctamente los caracteres. Para esto se divide de diversas formas utilizando las columnas de corte con certeza de 0% (columnas rojas) cada región delimitada por dos columnas de corte con certeza 100% (columnas azules) buscando reconocer una combinación de caracteres que represente a toda la región (ver Ilustración 13). Para buscar esta combinación se diseñó un algoritmo recursivo que aprovecha las ventajas de la programación dinámica (*memoization*), ver pseudo-código en la Ilustración 14.

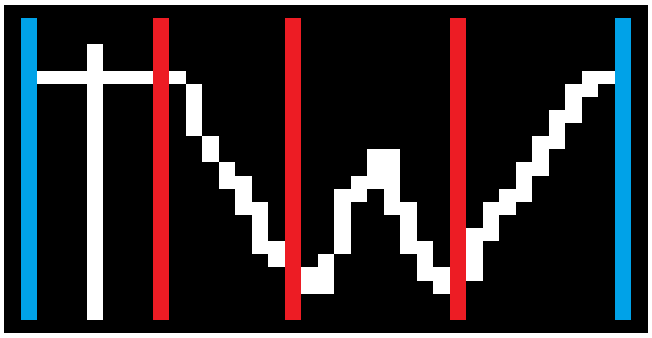

Ilustración 13 "Región delimitada por dos columnas de cortes con valor de certeza 100%. En esta región el clasificador reconoce la "t" y la "w" al segmentar por las regiones marcadas en la imagen"

Sea:

c	columnas de cortes cantidatas
cInicio	Índice de la columna de corte con certeza 100%
cFin	Índice de la columna de corte con certeza 100% que le sigue a cInicio
tabla[][]	tabla donde se almacena en la posición [**i, j**] la combiación de caracteres reconocidos en la región delimitada desde la columna i hasta la columna j.
Caracteres	Combinación de caracteres.
esLógico(i, j)	Devuelve verdadero si se puede identificar un caracter válido en la región segmentada desde desde la columna i hasta la columna j.
Caracter(i, j)	Devuelve el caracter reconocido en la región delimitada desde la columna i hasta la columna j.

```
Caracteres auxMemoizedAnalisisColumnasCandidatas (cInicio, cFin, c, tabla)
{
        if ( tabla[cInicio][cFin] no está vacía )
            return tabla[cInicio][cFin];

        if ( esLógico(cInicio, cFin) )
        {
            tabla[cInicio][cFin] = Caracter(cInicio, cFin);
        }else{
            for ( k = cFin - 1; k > cInicio; k--)
            {
              if (esLógico(cInicio, k)  ^  auxMemoizedAnalisisColumnasCandidatas (k, cFin, c, tabla) )
              {
                 tabla[cInicio][k] = Caracter(cInicio, k);
                 tabla[cInicio][cFin] = Caracter(cInicio ,k) + Caracter(k, cFin);
                 return tabla[cInicio][cFin];
              }
            }
          tabla[cInicio][cFin] = No esLógico;
        }
    return tabla[cInicio][cFin];
}
```

Ilustración 14 "Pseudocódigo del algortimo recursivo"

Fase de Extracción de Características

En esta etapa se extraen las características de cada letra en la imagen adelgazada utilizando el método de extracción de características *Zoning* (36), que consiste en dividir cada región de la imagen que representa una letra en NxM cuadrículas (8x8 en el software desarrollado), y asignar a cada cuadrícula el promedio de los píxeles que contiene cada una. Posteriormente estas características son utilizadas para la etapa de reconocimiento.

Fase de Reconocimiento

Luego de un experimento realizado se decidió utilizar para el reconocimiento el clasificador IBK que está basado en el algoritmo de K-vecinos más cercanos (KNN, por sus siglas en inglés), ya que fue el clasificador que mejor resultado arrojó. El KNN es un método de vecindad basado en casos o instancias y de clasificación supervisado. Es usado para la clasificación de objetos (elementos). Consiste en extraer información de un conjunto de datos conocidos para clasificar nuevos datos o agrupar existentes. Trata de imitar el sentido común humano a través de experiencias pasadas y de solucionar problemas presentes con analogía de problemas pasados. Este método supone que los vecinos más cercanos dan la mejor clasificación y esto se hace utilizando todos los atributos. Para esta etapa se utiliza una base de datos de imágenes de los distintos símbolos utilizados para la clasificación con un total de 72 símbolos. Los símbolos son equivalentes a las letras del abecedario español, incluyendo las letras minúsculas y las letras mayúsculas, las vocales tildadas y los números del 0 al 9, también los caracteres especiales como el punto, la coma, punto y coma, slach, dos puntos, paréntesis, corchetes, signos de exclamación, además de operadores matemáticos como la suma y la igualdad. El promedio de la cantidad de imágenes por símbolos es de 60 imágenes aproximadamente. La menor cantidad de imágenes con que cuenta un símbolo es de 5 imágenes y la mayor cantidad es de 336 imágenes. (37)

Mejorar Reconocimiento

En el reconocimiento se encontraron una serie de errores que se repetían con frecuencia y fueron solucionados aplicando reglas y heurísticas. A continuación se muestra una tabla que contienen los errores de reconocimiento de algunos caracteres, y las descripciones de las reglas definidas.

Resultado del reconocimiento	**Resultado ideal**	**Causas del error**	**Regla**
'**r**' '**1**'	'**n**'	Mala segmentación mediante la columna de	Si luego de '**r**' y '**1**' consecutivos le sigue algún carácter que no sea

		corte con valor de certeza 0% intercalada en la 'n'.	número entonces '**r**' y '**1**' son sustituidos por '**n**'. Ejemplo: **cor1templa = contempla.**
Prue ba	**Prueba**	Mala segmentación.	Si dos palabras están en la misma línea de texto y relativamente cerca entonces se elimina el espacio entre ellas conformando una única palabra.
. .	**:**	El proceso de segmentación separa el carácter **:** en dos puntos separados	Si dos caracteres en la misma línea se encuentran uno encima del otro se verifica el carácter reconocido en cada caso y se conforma un solo carácter. Ejemplo: '¡', '**;**', '**?**', '**:**'

Tabla 9 "Reglas utilizadas para mejorar el reconocimiento."

Exportación del documento

Para la exportación del documento se utiliza *iText* en su versión 5.5.5. *iText* es una biblioteca de clases desarrollada en java de código abierto que soporta la generación de documentos HTML, RTF y XML, además de documentos PDFs. También cuenta con varias clases para generar gran variedad de fuentes, tablas y establecer marcas de agua que pueden ser utilizados en el cuerpo del documento. (38).

2.6 Diseño

2.6.1 Metáfora

El sistema abre el directorio donde se encuentran las imágenes, luego las carga y le realiza los procesos de pre-procesamiento, segmentación, extracción y reconocimiento de caracteres para finalmente entregar como salida un documento digital en formato PDF que contenga el texto de la imagen de entrada. Con la realización de este proceso el usuario puede interactuar con este documento mediante un editor de texto.

2.7 Arquitectura del sistema

La arquitectura de software es una vista del sistema que incluye los componentes principales, la conducta de esos componentes y las formas en que los componentes interactúan y se coordinan para alcanzar la

misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. (39)

2.7.1 N-capas

Para la organización estructural del sistema para el reconocimiento óptico de caracteres se decidió utilizar la arquitectura n-capas.

El estilo arquitectural de n-capas se basa en una distribución jerárquica de los roles y responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la interacción con otras capas y las responsabilidades y funcionalidad que implementan. (40)

Entre sus características se encuentran:

- Descomposición de los servicios de forma que la mayoría de las interacciones ocurren solo entre capas vecinas.
- Las capas de una aplicación pueden residir en la misma máquina o pueden estar distribuidas entre varios equipos.
- Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidas.
- Cada nivel agrega las responsabilidades y abstracciones del nivel inferior.

A continuación se muestra la arquitectura del sistema:

Ilustración 15 "Arquitectura del sistema"

Capa presentación: Esta capa es la responsable de presentar al usuario los conceptos de negocio mediante interfaces de usuarios. Se relaciona con la capa de negocio del sistema, le envía las solicitudes que realiza el usuario para que esta las procese y poder mostrar los resultados que el usuario espera. Desde el punto de vista del usuario final esta capa es la "aplicación" en sí, de nada sirve el haber realizado un buen trabajo si no se facilita su explotación de la manera más satisfactoria posible para el usuario.

Capa del negocio: Esta capa es la responsable de representar los conceptos del negocio. En esta capa está contenido el sistema en sí, que a su vez se relaciona con la capa presentación, también contiene el componente **Utilidades** que ayuda al sistema, ya que este tiene implementado funcionalidades que se utilizan en todo el procesamiento, también contiene el componente **Objetos del dominio** que contiene las clases que modelan los objetos con los que se trabajan, facilitando al sistema el trabajo con los elementos que estas clases modelan, contiene además el componente **Conformar documento de salida** que lo utiliza en la etapa final del procesamiento, con la ayuda de este componente conforma el documento con el texto contenido en la imagen de entrada.

2.7.2 Tuberías y filtros

Se utiliza también la arquitectura tuberías y filtros para representar de forma más explícita el proceso que realiza el sistema, mostrando de esta forma las fases por las que pasa una imagen para ser procesada.

Es una arquitectura que provee una estructura para procesar flujos de datos, en la que los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Entre las ventajas del uso de este estilo arquitectónico se tiene que es simple de entender e interpretar, fuerza un proceso secuencial y los filtros se pueden empaquetar y hacerlos paralelos o distribuidos.

El sistema para el reconocimiento óptico de caracteres se divide en varios pasos secuenciales a los que también se les conoce como fases, estas fases pueden ser los filtros, ya que la imagen que entra en cada fase no es la misma que sale. A continuación se muestra cómo queda conformada esta arquitectura con respecto al presente trabajo

Entrada: Imagen que contiene texto.
Filtro 1: Pre-procesamiento de la imagen de entrada.
Filtro 2: Segmentación de la imagen.
Filtro 3: Extracción de características.
Filtro 4: Reconocimiento de caracteres.
Filtro 5: Conformación del texto.
Salida: Documento con el texto y figuras contenido en la imagen de entrada.

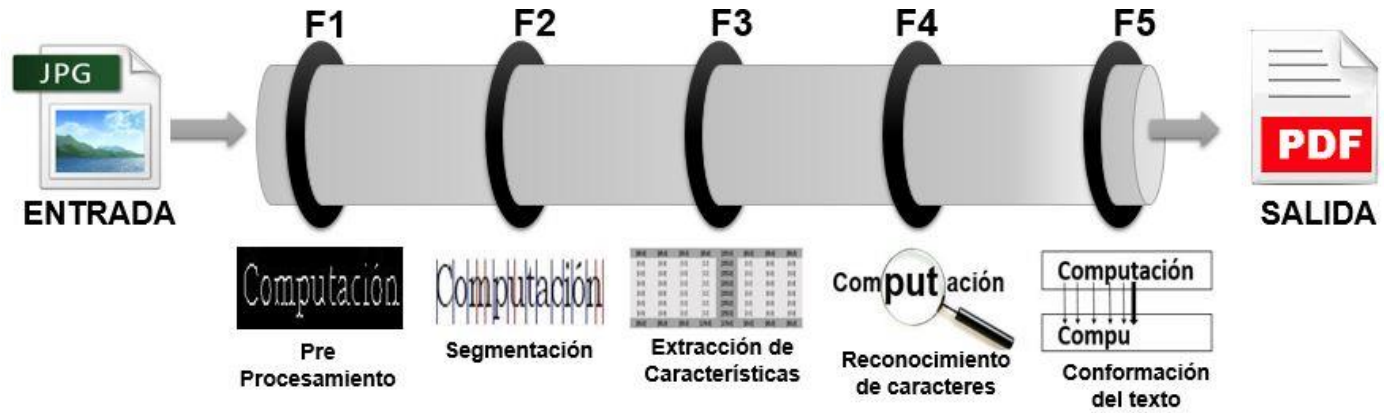

Ilustración 16 "Arquitectura de tuberías y filtros"

2.8 Diagrama de clases del sistema

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases. (41)

El propósito de un diagrama de clase es describir las clases que conforman el modelo de un determinado sistema. Dado el carácter de refinamiento iterativo que caracteriza un desarrollo orientado a objetos, el diagrama de clase va a ser creado y refinado durante las fases de análisis y diseño, estando presente como guía en la implementación del sistema. (42)

A continuación se muestra el diagrama de clases del presente trabajo:

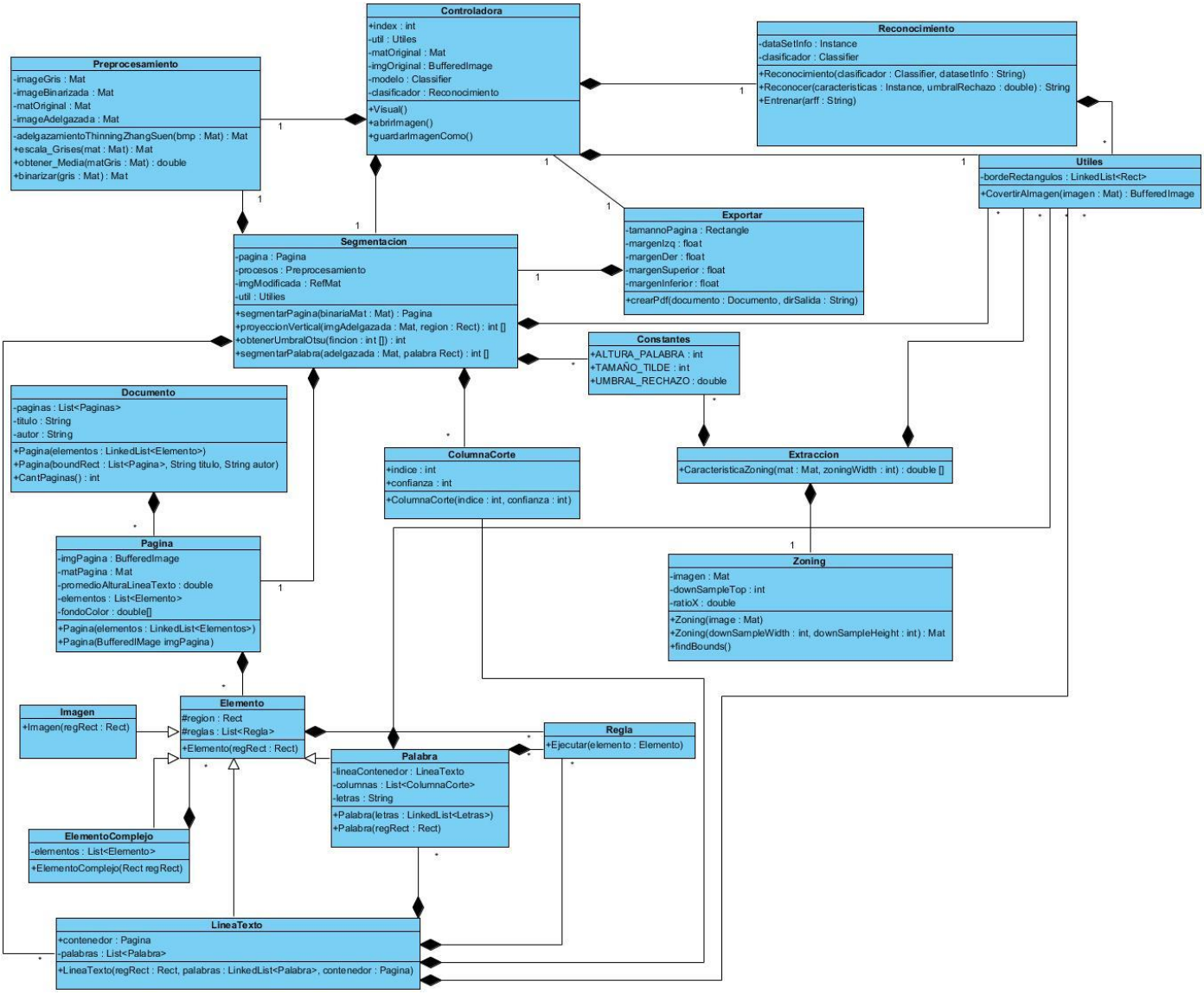

Ilustración 17 "Diagrama de clases del sistema"

2.9 Patrones de diseño utilizados

2.9.1 Patrones GRASP

En el diseño de la aplicación se utilizaron los patrones GRASP (General Responsibility Asignment Software Patterns en español significa Patrones Generales de Software para Asignar Responsabilidades). Los mismos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (43). Dentro de los patrones de diseño GRASP se utilizó:

- **Experto:** Este patrón se evidencia en la clase **Documento**, ya que esta se vale de su propia información para realizar las tareas encomendadas. En este caso si se quisiera saber la cantidad de páginas existentes, **Documento** es la clase correcta para asumir esta responsabilidad.

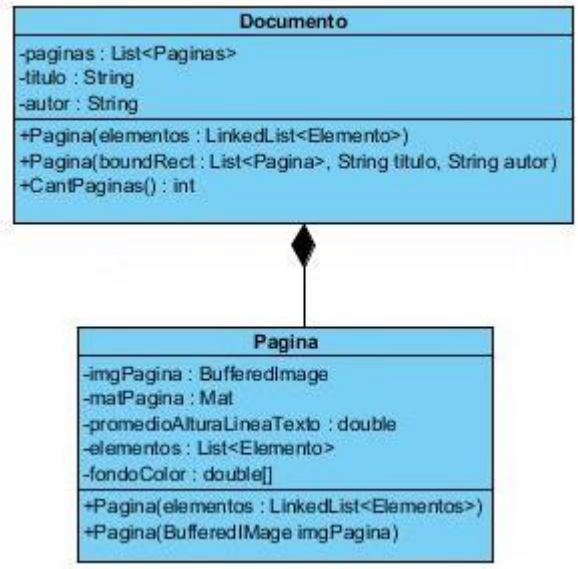

Ilustración 18 "Evidencia del patrón experto"

- **Creador:** Es evidenciado en la clase **Documento**, ya que en la misma se crean varias instancias de la clase **Pagina** (ver ilustración 18). Esta responsabilidad se le asigna a la clase **Documento** debido a que es la que tiene la información necesaria para crear estas instancias.
- **Alta Cohesión:** Este patrón se pone de manifiesto en las clases **Segmentacion**, ya que todas sus funcionalidades están orientadas a realizar el proceso de segmentación.(ver ilustración 19)

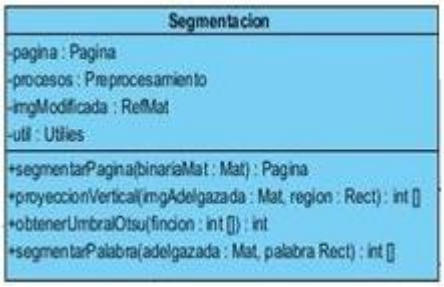

Ilustración 19 "Evidencia del patrón alta cohesión"

- **Bajo Acoplamiento:** Este patrón se pone de manifiesto en la clase **Zoning**, ya que esta clase no depende de otras para realizar la responsabilidad que le fue asignada.
- **Controlador:** Este patrón se pone en evidencia en la clase **Controladora**, ya que esta clase tiene la responsabilidad de recibir o manejar los mensaje de los eventos del sistema.

2.9.2 Patrones GoF

Los patrones GoF que se utilizaron en el desarrollo del sistema de reconocimiento óptico de caracteres son:

Patrones de Creación: El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

- **Singleton** (instancia única): Se pone de manifiesto en la clase **Preprocesamiento** garantizando la existencia de una única instancia de dicha clase la cual puede ser accedida y utilizada de forma global.

```
public class Preprocesamiento {

    private static Preprocesamiento instancia;

    private Preprocesamiento() {...2 lines }

    public static Preprocesamiento getInstancia() {
        if (instancia == null) {
            instancia = new Preprocesamiento();
        }
        return instancia;
    }
```

Ilustración 20 "Fragmento de código en el que se evidencia el uso del patrón"

- **Composite:** Se pone de manifiesto en la relación de las clases **Elemento** y **ElementoComplejo**, donde permite construir objetos complejos mediante composición recursiva de objetos similares, en este caso además de la clase **ElementoComplejo** también heredan de la clase **Elemento** las clases **LineaTexto**, **Palabra** e **Imagen**.

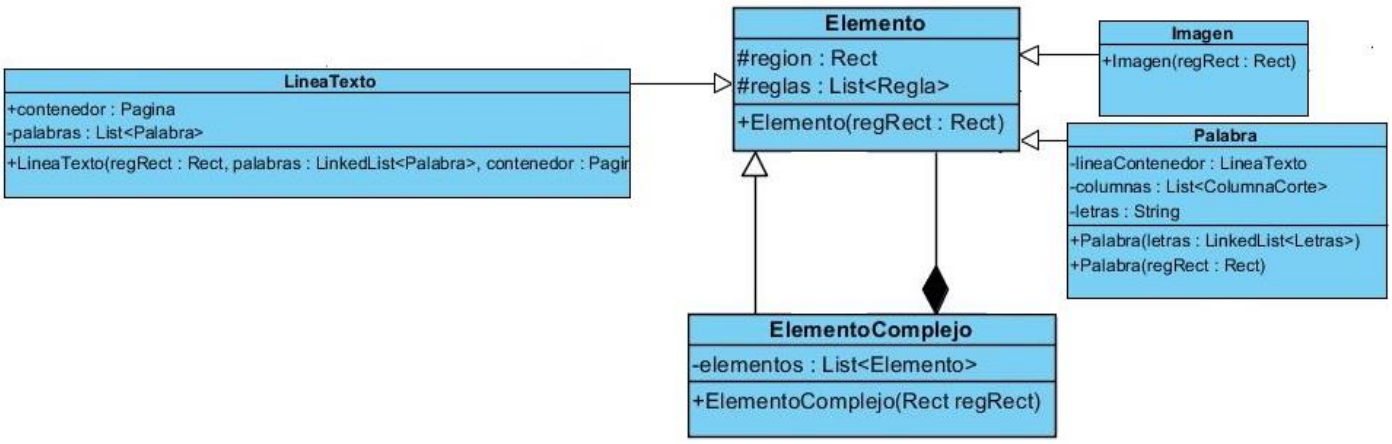

Ilustración 21 "Fragmento del diagrama de clases que muestra el uso del patrón"

2.10 Tarjetas CRC

CRC significa clase, responsabilidad y colaboración. A continuación se muestra la tarjeta CRC correspondiente a la clase **Pagina**. El resto de ellas se pueden consultar en los Anexos.

Nombre de la clase: Pagina	
Responsabilidad	**Colaboradores**
segmentarCaracteres	Elemento LineaTexto
promedioAlturaLineaTexto	Elemento LineaTexto
verificarReglas	Elemento LineaTexto

Tabla 10 "Tarjeta CRC correspondiente a la clase Pagina"

2.11 Conclusiones parciales

Luego de realizar el presente capítulo, se puede concluir que con la creación de los artefactos y la definición de las funcionalidades del sistema así como las historias de usuario correspondientes a las mismas se facilitó el trabajo de los desarrolladores y les permitió conocer los requerimientos del cliente con respecto a la aplicación. La planeación realizada le permitió a los desarrolladores organizar el proceso de entregas y cumplir con lo acordado. Además con la arquitectura n-capas se garantizará un desarrollo organizado y con la utilización apropiada de los patrones de diseño seleccionados se obtendrá como resultado un sistema para el reconocimiento óptico de caracteres con calidad.

Capítulo 3 “Implementación y pruebas”

3.1 Introducción

Luego de haber realizado el análisis, el diseño y el modelado de los artefactos que generan estas actividades, se inicia con la implementación del sistema para el reconocimiento óptico de caracteres. Se describe el estilo de codificación utilizado, se presenta el diagrama de componente que permite comprender la estructura del sistema y por último se realizan pruebas para comprobar la efectividad del producto y se muestran los resultados de los casos de prueba realizados.

3.2 Implementación

3.2.1 Tareas de ingeniería

A continuación se muestran las tareas de ingeniería correspondientes a la primera iteración, el resto se pueden consultar en los anexos.

Iteración 1	
Historia de Usuario	**Tareas**
Cargar Imagen	• Buscar el directorio de donde se quieren cargar las imágenes. • Verificar que existan imágenes en el directorio. • Verificar que la imagen esté en un formato que el sistema reconozca.

Tabla 11 "Caso de prueba de la historia de usuario cargar imagen"

3.2.2 Estándar de codificación

Reglas de codificación

Para la implementación del sistema para el reconocimiento óptico de caracteres, se tuvo en cuenta el estándar de codificación para el lenguaje de programación Java propuesto en (44). Entre las reglas que se definen se encuentran que:

- Las líneas de código no deben exceder los ochenta caracteres.

- Se debe hacer solo una declaración por línea de código.
- Debe aparecer un espacio en blanco después de cada coma en las listas de argumentos.
- Se debe usar siempre una línea en blanco entre métodos.
- Ningún espacio en blanco entre el nombre de un método y el paréntesis que abre su lista de parámetros.
- La llave de apertura aparece al final de la misma línea de la sentencia de declaración.
- La llave de cierre empieza una nueva línea para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura.

Convenciones de nombres

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. También pueden dar información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código. (44)

Identificadores	Reglas para el nombre	Ejemplo
Paquetes	El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas.	utilidades presentación
Clases o Interfaces	Los nombres de las clases deben ser sustantivos. Cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos abreviaturas.	Segmentacion; ColumnaCorte;
Métodos	Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.	Limpiar(); getImageGris();

Variables	Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúscula. Los nombres de variables no deben empezar con los caracteres guion bajo "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje. Los nombres de las variables deben ser cortos pero con significado. Los nombres de variables de un solo caracter se deben evitar, excepto para variables índices temporales. Nombres comunes para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres.	imageAdelgazada imagen
Constantes	Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas separando las palabras con un subguión ("_"). (Las constantes ANSI se deben evitar, para facilitar su depuración)	TAMAÑO_TILDE ALTURA

Tabla 12 "Convenciones de nombre"

3.3 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. (45)

A continuación se muestra el diagrama de componentes correspondiente al presente trabajo.

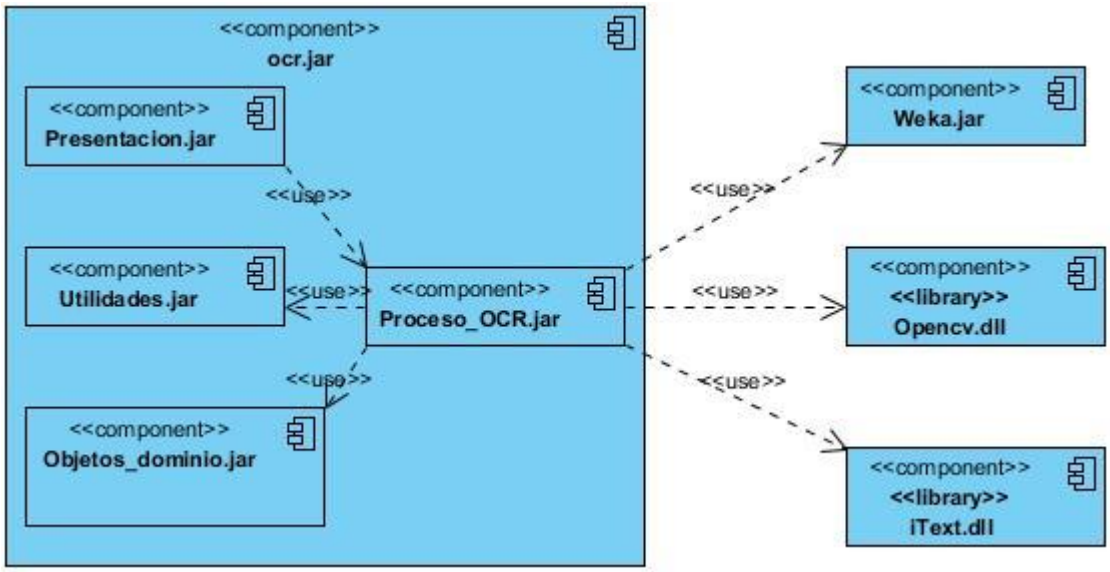

Ilustración 22 "Diagrama de componentes"

3.3.1 Descripción de los componentes

Proceso_OCR.jar: es el componente principal del sistema, gestiona la lógica del negocio y es el encargado de realizar llamadas a los servicios para darle respuesta al usuario. Para realizar el procesamiento y dar respuesta al usuario hace uso de los componentes: **Objetos_dominio.jar**, **Utilidades.jar**. Se relaciona además con los componentes externos **Weka.jar** utilizando de este sus algoritmos de clasificación, con el componente **OpenCV.dll** que contiene implementada funciones para el tratamiento de imágenes y con **iText.jar** que lo ayuda a conformar el documento de salida.

Presentacion.jar: Este es el componente que representa las interfaces de usuario. Se relaciona con el componente **Proceso_OCR.jar**, le envía las solicitudes que realiza el usuario para que este las procese y poder mostrarle el resultado que espera.

Objetos_dominio.jar: Este componente contiene las clases que modelan los objetos con los que se trabajan, facilitando al sistema el trabajo con estos elementos.

Utilidades.jar: Este componente es utilizado para realizar el procesamiento, ya que tiene implementado funcionalidades que se utilizan en todas las fases por las que pasa una imagen para ser procesada.

Weka.jar: Es un componente externo que ayuda al sistema con los algoritmos de clasificación que este tiene implementados.

OpenCV.dll: Este componente externo tiene implementada funcionalidades para el tratamiento de imágenes, es una biblioteca de clases que es muy utilizada por el componente principal en su procesamiento.

iText.dll: Este componente externo es una biblioteca de clases, este componente ayuda al sistema a generar el documento de salida del proceso.

3.4 Pruebas

Se realizaron pruebas a 5 clasificadores para escoger el de mejor resultado, para ser empleado en el reconocimiento de caracteres. Para esto se utilizó la herramienta Weka, donde se encuentran implementados estos clasificadores. Los clasificadores a los que se le realizaron las pruebas son: *BayesNet, Multilayer Perceptron, IBk, Random Commitee.*

Para lograr una mejor visualización de los resultados y facilitar la comparación de los mismos se elaboró una tabla (Tabla 13**¡Error! No se encuentra el origen de la referencia.**). Donde se muestra la cantidad de confusiones de cada clasificador, los caracteres en los que se confundió y el por ciento de efectividad.

Clasificadores	Cantidad de confusiones y caracteres en los que se confundió	Por ciento de efectividad
BayesNet	23 veces (n con ,), 15 veces (e con 1), 30 veces (1 con l), 31 veces (1 con n), 16 veces (c con 4), 16 veces (é con 4), 31 veces (s con 5), 16 veces (e con 0 y o), 17 veces (í con i), 15 veces (h con b), 35 veces (i mayúscula por e1), 18 veces (- por el .), 37 veces (i mayúscula por el .), 22 veces (p mayúscula con p minúscula).	84.56%
MultilayerPerceptron	16 veces (- por el .), 16 veces (u minúscula por u mayúscula), 20 veces (u mayúscula por u minúscula),	92.54%

	32 veces (i mayúscula por .).	
IBk	30 veces (i mayúscula por .), 16 veces (- por el .), 16 veces (u minúscula por u mayúscula), 18 veces (u mayúscula por u minúscula).	93.37%
RandomCommittee	18 veces (- por el .), 37 veces (i mayúscula por .), 23 veces (p mayúscula con p minúscula), 17 veces (u minúscula por u mayúscula), 19 veces (u mayúscula por u minúscula), 15 veces (y mayúscula por y minúscula).	92.89%

Tabla 13 "Resultados de las pruebas para seleccionar un clasificador para el reconocimiento de caracteres"

Conclusiones de las pruebas a los clasificadores

Luego de realizadas las pruebas a los clasificadores se puede concluir que:

- El clasificador con mayor por ciento de efectividad es el IBK con 93,37%.
- El clasificador con menor por ciento de efectividad fue BayesNet con 84,56%.
- En el caractere que más se confunden estos clasificadores es la i mayúscula que la confunden con el punto.
- El clasificador IBK es el que menos confusiones comete.
- El clasificador BayesNet es el que más confusiones comete.

Después de haber analizado los resultados de las pruebas se decide que el clasificador que será utilizado en el sistema para el reconocimiento óptico de caracteres es IBK, ya que fue el que mejores resultados arrojó en la pruebas realizadas, además las confusiones de este clasificador se pueden solucionar con las reglas heurísticas especificadas en la Tabla 9, esto garantiza que el clasificador aumente su por ciento de efectividad.

En el entrenamiento realizado el clasificador IBK tuvo un alto grado de confusión con respecto a los caracteres “i” mayúscula y el “.”, con un total de 30 confusiones, esto es a causa de que el patrón de estos caracteres después de aplicado el *Zoning* son muy similares (ver ilustración 23), además las bases de datos de estos caracteres tienen pocas imágenes, lo cual dificulta el entrenamiento del clasificador en estos casos. Para mejorar este resultado se recomienda aumentar la cantidad de imágenes de 5 a 10 veces la dimensión del vector de características extraído (64 características) para cada clase. Así el

clasificador tendría un mejor aprendizaje, lo que permitiría ser más exacto a la hora de clasificar dos caracteres con patrones semejantes.

Imagen "I"

Imagen "1"

Ilustración 23 "Representación de los patrones "I" y "1" con los valores de las características extraídas por Zoning."

Comparación con otros sistemas

Una vez terminado el sistema para el reconocimiento óptico de caracteres UCI_OCR[3] se realizó una comparación con el software NSOCR en su versión 5.8.790, para comprobar si se cumplieron los objetivos trazados.

Caso	Cantidad de imágenes	Nombre del OCR	Problemas en el reconocimiento	Cantidad de equivocaciones	Por ciento de efectividad	Tiempo de ejecución
Imágenes con solo texto.	5	NSOCR	Confunde la i minúscula con la mayúscula, separa palabras en dos fragmentos.	15 equivocaciones de un total de 524 caracteres.	97,2%	2,53 segundos por cada imagen.
		UCI_OCR	No reconoce el	10	98,1%	5,71

[3] Nombre del sistema para el reconocimiento óptico de caracteres desarrollado en el presente trabajo.

			fragmento “fi”.	equivocaciones de un total de 524 caracteres.		segundos por cada imagen.

Tabla 14 "Comparación entre los sistemas NSOCR y UCI_OCR"

Cantidad de equivocaciones: cantidad de caracteres que confunde.

Por ciento de efectividad: se calcula dividiendo la cantidad de equivocaciones entre la cantidad de caracteres contenidos en la imagen.

Conclusiones de la comparación realizada

Luego de la comparación realizada se puede concluir que:

- El sistema NSOCR realiza el procesamiento de las imágenes en menor tiempo que el UCI_OCR.
- El sistema NSOCR se equivoca en el procesamiento de imagen en el caso de que sea solo texto más que el UCI_OCR según su por ciento de efectividad.
- El sistema NSOCR no reconoce las regiones de figuras contenidas en la imagen.
- El sistema UCI_OCR reconoce correctamente las regiones de figuras contenidas en las imágenes.
- El sistema NSOCR comete más errores en el procesamiento de imágenes que contienen textos y figuras que el UCI_OCR, según si por ciento de efectividad.

Después de analizar los resultados obtenidos con las pruebas, se puede decir que el sistema pare el reconocimiento óptico de caracteres UCI_OCR cumple con el objetivo propuesto para la investigación, ya que reduce los errores cometidos por otros sistemas en el reconocimiento de caracteres.

3.4.2 Pruebas de efectividad

Evaluación del rendimiento del sistema:

Para evaluar las prestaciones y el rendimiento del sistema se tomó una muestra de 100 imágenes de cada posible caso (imágenes con solo texto, imágenes con solo figuras e imágenes con figuras y textos). Estas imágenes fueron procesadas por el sistema para evaluar su rendimiento respecto al tiempo de respuesta. Las pruebas se realizaron en una computadora Intel Core i3 a 3GHz, 2GB de Memoria RAM DDR3, con Sistema Operativo Windows 8 Enterprise. A continuación se muestran una tabla (Tabla 15) con los resultados de las pruebas realizadas.

Cantidad de imágenes	Casos	Segundos	Minutos
100	Imagen con solo texto	571,77	9,53
100	Imagen con solo figuras	393,75	6,56
100	Imagen con texto y figuras	782,95	13,05

Tabla 15 "Pruebas de efectividad para imágenes que solo contienen texto".

El sistema para el reconocimiento óptico de caracteres propuesto

3.4.2 Pruebas de aceptación

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, buscan una cobertura de la especificación de requisitos y de las historias de usuario. Estas pruebas no se realizan durante el desarrollo, se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente. Las pruebas de aceptación pueden tener lugar a lo largo de semanas o meses, descubriendo así errores latentes o escondidos que pueden ir degradando el funcionamiento del sistema. Estas pruebas son muy importantes, ya que definen el paso a nuevas fases del proyecto como el despliegue y mantenimiento. (48)

A continuación se muestra el caso de prueba diseñado para la primera historia de usuario. Los restantes casos de pruebas se pueden apreciar en los anexos.

Caso de Prueba de Aceptación	
Código: CP_HU1	**HU1:** Cargar y guardar imagen
Responsable de la prueba: Ronald Domínguez Marrero	
Descripción: Prueba que verifica si el sistema carga y guarda las imágenes.	

Condiciones de ejecución: Deben existir imágenes en el directorio.
Entrada / Pasos de ejecución: 1. Entrar en el directorio donde se encuentran la imágenes que se quieres procesar. 2. Reconocer si existen imágenes en el directorio. 3. Seleccionar y cargar una imagen del directorio. 4. Verificar que la imagen es cargada correctamente en el sistema. 5. Preguntar al usuario si desea guardar el texto generado 6. Verificar que la imagen fue guardada correctamente.
Resultado esperado: Imagen cargada en el sistema y guardada en directorio
Evaluación de la prueba: Satisfactoria

Tabla 16 "Prueba de aceptación de la historia de usuario 1"

3.5 Conclusiones parciales

En este capítulo se definió el estilo de codificación utilizado en la implementación del sistema, lo que permitió establecer una mayor organización y claridad en el código escrito. Se probaron diferentes clasificadores, lo que permitió seleccionar el que mejor resultado alcanzó para utilizarlo en la fase de reconocimiento de caracteres del sistema desarrollado. Se realizó además una comparación con el software NSOCR, que permitió comprobar que con el desarrollo del trabajo se cumplieron los objetivos trazados.

Conclusiones generales

Luego del trabajo realizado se puede concluir que:

- El análisis realizado a los sistemas similares demostró que es necesario implementar un sistema para el reconocimiento óptico de caracteres propio de la institución.
- El análisis realizado a las diferentes herramientas y tecnologías permitió seleccionar aquellas más adecuadas para llevar a cabo el desarrollo del sistema.
- Se implementó un sistema para el reconocimiento óptico de caracteres a partir de los requisitos funcionales establecidos por la institución, haciendo uso de la arquitectura y tecnologías definidas en el trabajo.
- Las pruebas realizadas permitieron seleccionar el mejor clasificador, que en este caso fue el IBK, para garantizar una alta efectividad en el reconocimiento de los caracteres.
- La comparación realizada con el software NSOCR permitió demostrar que el sistema UCI_OCR cumple con los objetivos propuestos.
- Las pruebas de aceptación realizadas permitieron comprobar que el sistema realiza todas sus funcionalidades satisfactoriamente.

Recomendaciones

Luego de concluir con la implementación del sistema para el reconocimiento óptico de caracteres, para mejorar el sistema y seguir profundizando en el tema se plantean las siguientes recomendaciones:

- Realizar pruebas con otros métodos de extracción de características para mejorar el proceso de reconocimiento.
- Agregar clases a la base de datos para el entrenamiento.
- Realizar mejoras para que reconozca las tablas y el texto que estas contienen.
- Agregar a las base de datos los caracteres utilizados en las fórmulas matemáticas.
- Agregar diccionario de palabras.
- Aumentar la cantidad de imágenes generadas con diferentes fuentes y tamaño de letras en la base de datos de 5 a 10 veces el tamaño del vector de características (64 características) en caso de usar el Zoning como método de extracción de características.
- Realizar un estudio para exportar el documento a otros formatos.

Bibliografía

1. **Navarro, Joaquim Arandis.** *Reconocimiento Óptico de Carácteres.*

2. **Cam, Celso Gonzáles.** *La Importancia de la Digitalización de Archivos para la Biblioteca1.* 2007.

3. **Nuance. [En línea] http://www.nuance.es/particulares/producto/omnipage/index.htm.**

4. ABBYY. [En línea] http://www.abbyyeu.com/es/.

5. Download.com. [En línea] [Citado el: 29 de 5 de 2015.] http://descargar.cnet.com/windows/nicomsoft/3260-20_4-6291121-1.html.

6. TopOCR. [En línea] http://www.topocr.com/.

7. Abate, Eloy. Linux para mi,Linux para todos. *OCR en Ubuntu.* [En línea] 25 de 11 de 2010. [Citado el: 20 de 3 de 2015.] http://linuxmaniatico.blogspot.com/2010_11_01_archive.html.

8. Compartolid. [En línea] [Citado el: 20 de 3 de 2015.] http://www.compartolid.es/tesseract-ocr/.

9. Softpicks. [En línea] [Citado el: 20 de 3 de 2015.] http://softpicks.com.es/software/Audio-Video/MP3/GOCR-Windows-Frontend_es-44102.htm.

10. IrisLink. [En línea] http://www.irislink.com/c3-2115-58/Readiris-14--OCR-Software--Scan--Convert---Manage-your-Documents-.aspx.

11. Prezi. *Optical Character Recognition.* [En línea] [Citado el: 13 de 5 de 2015.] https://prezi.com/wanqrdi0pgax/ocr/.

12. PIATTiNI. 1996.

13. Letelier, Patricio. *Metodologías ágiles para el desarrollo de software: eXtreme Programming(XP).*

14. Prezi. *Los lenguajes de programación.* [En línea] 1 de 10 de 2013. [Citado el: 19 de 3 de 2015.] https://prezi.com/_6yptyqi-pr7/los-lenguajes-de-programacion/.

15. Infor. [En línea] http://www.infor.uva.es/~jmrr/tgp/java/JAVA.html.

16. Alegsa, Leandro. *Herramienta de modelado.*

17. Software.com.ar. [En línea] [Citado el: 20 de 1 de 2015.] http://www.software.com.ar/visual-paradigm-para-uml.html.

18. EcuRed. [En línea] [Citado el: 20 de 1 de 2015.] www.ecured.cu/visual_paradigm.

19. SDE. *Entorno de desarrollo integrado.* [En línea] [Citado el: 19 de 3 de 2015.] http://www.itec-sde.net/es/search_results?search=%23Integrated_development_environment.

20. IBM, Clave. *Eclipse, herramienta Universal.* [En línea] [Citado el: 19 de 3 de 2015.] http://clave.zintegra.com/tag/ibm/.

21. Bligoo. *NetBeans.* [En línea] [Citado el: 19 de 3 de 2015.] http://netjava.bligoo.com/netbeans.

22. Difercast. *Weka.* [En línea] [Citado el: 19 de 3 de 2015.] https://difercast.wordpress.com/author/difercast/page/3/.

23. usmp. *INFOFIA.* [En línea] 4 de 2006. [Citado el: 26 de 5 de 2015.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info56/info56itext.html.

24. Sergio De La Llana Alamar, Samuel Molina Vinci, Sergio Sanchez Martinez. web-sisop. *INTRODUCCIÓN A LAS LIBRERIAS OPENCV .* [En línea] [Citado el: 26 de 5 de 2015.] http://web-sisop.disca.upv.es/imd/cursosAnteriors/2k3-2k4/copiaTreballs/serdelal/trabajoIMD.xml.

25. Craig Larman, Prentice Hall. *Modelo del dominio.* 2003.

26. Pérez, María José Pérez. *Guía comparativa de metodologías ágiles.*

27. Wake, William C. *Extreme Programming Explored.*

28. Christos Nikolaos E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Vassili Loumos, Eleftherios Kayafas. *A License Plate-Recognition Algorithm for Intelligent Transportation System Applications.* 2006.

29. Pietikäinen, J. Sauvola and M. *Adaptive document image binarization.* 2000.

30. Agnihotri, Ved Prakash. *Offline Handwritten Devanagari Script Recognition.* India : s.n., 2012.

31. Julinda Gllavata, Ralph Ewerth and Bernd Freisleben. *A Robust Algorithm for Text Detection in Images.*

32. Dileep, Dinesh. *A FEATURE EXTRACTION TECHNIQUE BASED ON CHARACTER GEOMETRY FOR CHARACTER RECOGNITION.*

33. Jagruti Chandarana, Mayank Kapadia. *International Journal of Emerging Technology and Advanced Engineering Optical Character Recognition.*

34. Tanzila Saba, Amjad Rehman and Ghazali Sulong. *IMPROVED STATISTICAL FEATURES FOR CURSIVE.*

35. Divya gilly, Dr. Kumudha raimond. ***License Plate Recognition- A Template Matching Method.***

36. Bharat Bhushan, Simranjot Singh, Ruchi Singla. ***License Plate Recognition System using Neural Networks and Multithresholding Technique.***

37. Suen, T.Y ZHANG and C.Y. Fast Pasallel Algorithm for Thinning Digital Patterns. [ed.] Robert M. Haralick. ***Image Processing and Computer Vision.*** **1984, Vol. 27, 3.**

38. Canny, J. A Computational Approach To Edge Detection. ***IEEE Trans. Pattern Analysis and Machine Intelligence.*** **1986, págs. 679–698.**

39. Otsu, Nobuyuki. A threshold selection method from grey level histograms. New York : s.n., 1979, págs. 62–66.

40. Taxt, Anil Kumar Jain and Torfinn. Feature extraction Methods for Character Recognition- A Survey. 1996, Vol. 29, 4, págs. 641-662.

41. Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. 1992.

42. Ing. Yasnay Hernández Marrero, Dr. Joaquín Danilo Pina Amargos, Msc. Raisa Socorro Llanes, Ing. Joan Jaime. ***Librería iText para la generación de PDF dinámicos.*** **2008.**

43. Reynoso, Carlos Bily. ***Introducción a la Arquitectura de Software.***

44. Cesar de la Torre Llorente, Unai Zorrilla Castro,. ***Guia de arquitectura N-capas.***

45. JAVIER GUTIÉRREZ DIYARZA, Héctor Pérez pedrero. Academia.edu. ***Lenguaje Unificado de Modelado.*** **[En línea] [Citado el: 13 de 5 de 2015.] http://www.academia.edu/6713179/Lenguaje_Unificado_de_Modelado.**

46. Peñalvo, Francisco José García. ***Diagramas de clase en UML.***

47. Software, Departamento Central de Ingeniería de. ***Flujo de trabajo. Captura de requisitos. Modelo de negocio.*** **Habana : s.n., 2004. Vol. Modelo de negocio.**

48. Santana, Leonel Castillo. ***Procedimiento para los estándares de codificación. Java y CSharp.***

49. SparxSystems. ***UML 2 Component Diagram.*** **[En línea] [Citado el: 13 de 5 de 2015.] http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_componentdiagram.html.**

50. Pons, Yanet Fernández. monografías. [En línea] [Citado el: 12 de 5 de 2015.] http://www.monografias.com/trabajos36/pruebas-de-aceptacion/pruebas-de-aceptacion2.shtml.

51. Bianchini, Prof. Adelaide. ***Conceptos y definiciones de hipertexto.*** **1999.**

52. Imbaquingo, Daisy Elizabeth. ***Capítulo II Algoritmos de Reconocimiento.***

53. Gris, Alex. Alex Gris – Usabilidad, Promoción y Analítica web. ***150 conceptos tecnológicos que debes de conocer.*** **[En línea] 19 de 10 de 2012. [Citado el: 13 de 5 de 2015.] http://www.alexgris.com/150-conceptos-tecnologicos-que-debes-de-conocer/.**

54. GNU. [En línea] [Citado el: 13 de 4 de 2015.] https://www.gnu.org/philosophy/free-sw.es.html.

55. slideshare. [En línea] [Citado el: 13 de 4 de 2015.] http://es.slideshare.net/adrianmartinrin/software-privado-1561531.

56. Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald, David Scuse. ***WEKA Manual for Version 3-7-10.*** **Hamilton, New Zealand : s.n., July 31, 2013.**

57. Ian H. Witten, Eibe Frank, Mark A. Hall. Data Mining, Practical Machine Learning. 2011, 13, págs. 505, 515.

Glosario de términos

Hipertexto: Hipertexto es una base de datos. La información no consta de grupos de bytes, sino que es estructurada y de tamaño considerable, características similares a muchas bases de datos. A pesar de que la estructura de información tiene una forma distinta a las estructuras de bases de datos tradicionales, muchos sistemas de bases de datos son capaces de almacenar información utilizada en los hipertextos. Además la acción típica permitida al usuario es la de saltar entre las partes de la base de datos. Esto es diferente a la típica utilización de bases de datos, en los cuales la obtención de información se realiza a través de *query*. (49)

Binarización: La binarización de una imagen digital consiste en convertir la imagen digital en una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen. (50)

Fragmentación: La fragmentación o segmentación es la operación que permite la descomposición de un texto en diferentes entidades lógicas o fragmentos. Con la segmentación se deben localizar las zonas de interés. (50)

Adelgazamiento: El adelgazamiento consiste en ir borrando sucesivamente los puntos del borde de cada componente conexa, de forma que se preserve su topología. (50)

API: La interfaz de programación de aplicaciones (API, por sus siglas en inglés), es el conjunto de subrutinas, funciones, procedimientos o métodos, en la programación orientada a objetos, que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas. (51)

Software libre: Es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el software libre es una cuestión de libertad, no de precio. (52)

Software privado: El software propietario o software privado se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo, redistribuirlo, cuyo código fuente no está disponible o el acceso a éste se encuentra restringido. (53)