Universidad de las Ciencias Informáticas

Facultad 2

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

Título: Implementación del Módulo Digitalización de Documentos del Arkheia.

Autor(es): Rosalia Gómez Barrientos

Asiel Gil Martínez

Tutor: Ing. Yanet Del Risco Batista

La Habana, 2013

"Año 55 de la Revolución"

Frase

El único autógrafo digno de un hombre
es el que deja escrito con sus obras.

José Martí

Declaración de Autoría

Declaramos ser los autores de la presente tesis, reconociendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de manera exclusiva.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____	_____
Rosalia Gómez Barrientos	Asiel Gil Martínez
Firma Autor	Firma Autor

_____

Firma Tutor

Ing. Yanet Del Risco Batista

Agradecimientos

Rosalia

A mis padres, por su inmenso amor y cariño, por consentirme como hija más pequeña, por darme siempre las palabras de apoyo cuando las cosas parecen derrumbarse. A ellos por hacer de mí la persona que soy, por esto y más los adoro.

A mi tutora Yanet, por escogerme como una de sus tesistas, por su apoyo incondicional, por brindarme parte de su tiempo a pesar de tanto trabajo,

A mi compañero de tesis Asiel, por ser un buen amigo, por su ayuda en la realización de este trabajo.

A mis hermanos, por apoyarme y darme las fuerzas para seguir adelante. A mi hermana Yani, por ser mi ejemplo y darme el placer de ser tía de dos maravillosos niños.

A mi esposo "Mi cosi", por ser mi mejor amigo, compañero y la luz que me guía cuando todo parece oscuro. Por estar siempre conmigo a pesar de la distancia, por su confianza, amor y comprensión. A ti cosi por abrirme las puertas del amor.

A mi tía Disney, por acogerme en su casa durante los años de la carrera, por ser mi punto de apoyo en los momentos difíciles, por sus consejos y ser una madre para mí. Al resto de mi familia por apoyarme siempre.

A la familia de mi esposo por acogerme tan bien y ser parte de mi familia.

A mis amigas y compañeras Dayna, Elvita, Yeni, por acompañarme en los momentos de felicidad y tristeza. A mi compañera de cuarto Daima por soportarme 5 años. A mis compañeros de aula, a los que perduraron y los que no, gracias por el tiempo que hemos compartido.

A todos mis profesores por contribuir a mi formación profesional y guiarme en todo momento.

A mis compañeros del proyecto Archivo (Jany, Alain, Yarelis, Tan, Travieso, Dubalón, Aile, Leandro y Yaidel) porque de una forma u otra contribuyeron en la realización de este trabajo.

A todos gracias.

Agradecimientos

Asiel

A mis padres, mi hermana Aliria, Alex, mi sobrina mi Tía Rebeca , mis primos Yosvany y Nene, a mis Abuelos. Lo que soy y seré es gracias a ustedes.

A todos los profesores, del proyecto especialmente a Leandro, Yaidel y Frank.

A mi tutora Yanet, por todo el tiempo dedicado y la confianza depositada en nosotros.

A mi compañera de tesis Rosalia, por ser una buena amiga, por su ayuda en la realización de este trabajo.

A todos gracias.

Dedicatoria

Rosalia

A mis padres para darle otro motivo que los haga sentir orgullosos del fruto que han creado.

A mi familia por apoyarme en todo momento.

A mi esposo, porque lo AMO.

Asiel

A mis padres, mi hermana Aliria, mi sobrina y mi Tía Rebeca.

Resumen

Los archivos son instituciones que tienen como finalidad gestionar, atesorar, conservar y difundir el patrimonio documental. Para facilitar el manejo de la información en estas instituciones son utilizados los Sistemas de Gestión de Documentos de Archivos (SGDA). El desarrollo de los SGDA ha motivado la digitalización de los documentos de archivo, utilizando la copia digital para su difusión y consulta.

La Oficina de Asuntos Históricos del Consejo de Estado (OAHCE) atesora en sus depósitos información gran valor histórico. Esta oficina utiliza el SGDA Arkheia, en el cual se encuentran descritos y digitalizados varios fondos personales. A los investigadores que acuden a esta institución se les dificulta la búsqueda de información, al no poder acceder a la información contenida en los documentos digitalizados.

El módulo Digitalización de Documentos de Arkheia se encarga de la extracción de los caracteres en estas imágenes digitalizadas utilizando el motor de reconocimiento óptico de caracteres Tesseract. Para obtener una mayor calidad en el procesamiento se integró la biblioteca de visión artificial OpenCV lográndose mejorar el indicador de eficacia del OCR.

Con la implementación de este módulo se obtuvo a partir de las imágenes un documento digital que contiene en texto editable las letras y las palabras de esta, facilitando el acceso a la información en la OAHCE.

Palabras Claves: digitalización, documentos, OCR, información

Índice de Contenidos

Índice de Contenido

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

La historia de la evolución humana ha podido construirse gracias a la recopilación de la información en disímiles soportes a través del tiempo. Al lugar destinado al almacenamiento de los documentos con valor administrativo, legal, fiscal, científico, económico, político, cultural y/o histórico se denomina Archivo.

La finalidad de los archivos es gestionar, atesorar, conservar y difundir el patrimonio documental. Pueden almacenar documentos históricos recibidos por donación, depósito, transferencia y adquisición. Una forma de facilitar el manejo de la información es a través de la utilización de los Sistemas de Gestión de Documentos de Archivos (SGDA).

El desarrollo de los SGDA ha motivado la digitalización de los documentos de archivo, utilizando la copia digital para su difusión y consulta. Con la digitalización se facilitan los procesos de almacenamiento y transportación. Mediante esta se reduce la manipulación de los originales y permite la estabilización de los mismos en entornos inertes para conservarlos retrasando su deterioro. Además, disminuye el espacio físico y no existe riesgo de robo o destrucción de papeles, copias, etc.

Un documento digital puede ser el resultado de haber procesado con un "scanner" un documento originalmente impreso. El resultado, en primer lugar es 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 "textual" digital.

Para obtener un documento digital es necesario un programa que transforme en texto editable las letras y las palabras contenidas en la imagen. Los programas que cumplen esta función son conocidos como Reconocimiento Óptico de Caracteres (OCR). Finalmente, el resultado será una copia digital del documento, no como imagen, sino como documento textual con todas las posibilidades que su condición digital le otorga.

La Oficina de Asuntos Históricos del Consejo de Estado atesora en sus depósitos información de valor incalculable, que ha sido custodiada y conservada a lo largo de casi 50 años. Su misión consiste en atesorar, conservar y servir la documentación de valor permanente que posee. Esta oficina cuenta con varios fondos personales de líderes de la Revolución Cubana como Fidel, Raúl, Celia y el Che. Estos

fondos se encuentran completamente digitalizados y descritos siguiendo la norma ISAD (G), en el SGDA Arkheia. La norma ISAD (G) está compuesta por 26 campos, de los cuales solo 6 se consideran de carácter obligatorio.

A esta oficina acuden diariamente un grupo de investigadores solicitando acceso a determinados fondos para consultar documentos de interés en sus investigaciones. A estos usuarios se les dificulta la búsqueda de información, pues deben conocer los datos exactos del documento que están buscando para poder acceder a este. Esto ha motivado a los especialistas de esta institución a la búsqueda una solución que permita extraer el contenido de los documentos digitalizados para facilitar el acceso a la información.

Esta situación problemática lleva a enunciar el siguiente **problema a resolver**:

¿Cómo digitalizar los documentos extrayendo los caracteres de las imágenes digitales incorporadas al Arkheia?

El **Objeto de Estudio** comprende el proceso de digitalización de documentos y el **Campo de Acción** está orientado al proceso reconocimiento óptico de caracteres en imágenes digitales.

Para dar solución al problema planteado se plantea como **objetivo general**: Desarrollar el módulo Digitalización de Documentos del sistema Arkheia. Se desglosa en los siguientes **objetivos específicos**:

- Realizar el estudio del estado del arte para proponer el motor OCR sobre el que se va a montar el sistema.
- Implementar el módulo Digitalización de Documentos del sistema Arkheia.
- Realizar pruebas de caja negra del módulo y pruebas de efectividad al OCR.

El presente trabajo sustenta la siguiente **idea a defender:** Si se implementa el módulo Digitalización de Documentos para el Arkheia se facilitará el acceso a la información.

Para dar cumplimiento de los objetivos se proponen las siguientes **tareas a desarrollar**:

- Estudio del estado del arte para proponer el motor OCR sobre el que se va a montar el sistema.
- Estudio de los sistemas del digitalización de documentos que usan el motor OCR seleccionado.
- Descripción de las herramientas a usar en la implementación del módulo Digitalización de Documentos del Arkheia.

- Descripción de los requisitos del módulo Digitalización de Documentos.
- Descripción de los casos de uso del módulo Digitalización de Documentos.
- Elaboración de los diagramas de clases del módulo Digitalización de Documentos.
- Elaboración de los diagramas de interacción del módulo Digitalización de Documentos.
- Implementación del módulo Digitalización de Documentos del Arkheia realizando Reconocimiento Óptico de Caracteres.
- Confección de los casos de pruebas basados en casos de uso.
- Ejecución de las pruebas de caja negra del módulo Digitalización de Documentos
- Ejecución de las pruebas de efectividad al módulo Digitalización de Documentos.

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

Analítico-Sintético: En el presente trabajo se lleva cabo el análisis de las teorías y documentos relacionados con la digitalización de documentos, permitiendo la extracción de los elementos más importantes que con este tema se relaciona. Fue utilizado este método en el análisis realizado para seleccionar el motor OCR.

Análisis Histórico – Lógico: En el presente trabajo se realiza un análisis de cómo ha evolucionado la digitalización de documentos, y el desarrollo de los archivos en toda su trayectoria.

Modelación: Se emplea en la realización del diseño que dio paso a la implementación del módulo Digitalización de Documentos.

De los **Métodos Empíricos,** se utilizó el método de **Experimento:**
Se utilizó el método Experimento en su forma Transformador, que se refiere a revelar la realidad y actuar sobre ella para transformarla (Martinto, 2013). Se encuentra presente en las pruebas que comprobarán la efectividad del trabajo realizado, de acuerdo con los resultados, se decide si es necesario modificarlos para lograr la efectividad que se persigue.

Introducción

En este trabajo se ve reflejada la estructura básica del experimento como método de investigación que da lugar a luchas alternativas pero que de forma general consta de las partes siguientes:

- Constatación del estado inicial
- Introducción del factor de cambio
- Constatación del estado final
- Comparación del estado inicial con el final

Aportes prácticos esperados:

Se espera con la realización de este trabajo la implementación del módulo Digitalización de Documentos del Sistema de Gestión de Documentos de archivo Arkheia, que solucione el problema planteado y se facilite lo siguiente:

- Conservación de los documentos: evita la manipulación directa de los documentos ya que se accede a su copia digital.
- Convertir en texto las imágenes escaneadas: permite que en el futuro se puedan encontrar los documentos usando palabras incluidas en su contenido

Estructura del trabajo:

Capítulo 1. "Fundamentación Teórica" Este capítulo incluye un análisis del estado del arte de los sistemas OCR. Se describen los principales conceptos asociados al dominio del problema. Se describen los lenguajes de modelado, metodología de desarrollo de software y herramientas utilizadas en la solución del problema.

Capítulo 2. "Características del Sistema" En este capítulo se representa el modelo de dominio, dando paso al diseño del sistema a desarrollar, además de dar una propuesta para la solución del mismo. Se realiza la captura de los requisitos funcionales a partir de los cuales se definen los casos de uso del sistema.

Capítulo 3. "Análisis y Diseño" En este capítulo se realiza el modelado de algunos artefactos generados en los flujos de trabajo Análisis y Diseño, diagramas de clases de análisis, de paquetes, de interacción y diagramas de clases del diseño, los cuales guían la implementación del sistema a partir de la arquitectura definida.

Capítulo 4. “Implementación y Pruebas” En este capítulo se realizarán los diagramas de componentes y de despliegue, además de una breve explicación del funcionamiento de los componentes del sistema. Se realizarán las pruebas necesarias para comprobar la efectividad de la aplicación desarrollada, y se plasmarán lo resultados obtenidos.

Capítulo 1. "Fundamentación Teórica"

1.1. Introducción

En este capítulo se realiza un estudio del estado del arte de sistemas para el reconocimiento óptico de caracteres, enfocados en la extracción del texto de una imagen digitalizada. Se describen las ventajas y desventajas de cada uno, con el objetivo de seleccionar el que más se ajusta a las necesidades de este trabajo y justificar la decisión tomada. Finalmente se especifica el lenguaje de modelado, metodología y herramientas utilizados como parte de la propuesta de solución.

1.2. Marco Conceptual

A continuación se hace mención a los conceptos que brindan mejor comprensión de la investigación realizada.

¿Qué es digitalizar?

Digitalizar es la acción de convertir en digital información analógica. En otras palabras, es convertir cualquier señal de entrada continua en una serie de valores numéricos. La información digital es la única información que puede procesar una computadora, generalmente en sistema binario, es decir de ceros (0) y unos (1) (Map, 2012).

Digitalizar Documentos:

Digitalizar un documento va a depender del tipo de información. Si es una imagen fotográfica en papel, por ejemplo, puede digitalizarse a través de un escáner. Pero si es sonido o la voz de una persona se puede digitalizar por medio de un micrófono, que lo trasmite a la placa de sonido, donde se digitaliza (Map, 2012).

En cuanto a los documentos de textos, suelen digitalizarse empleando los sistemas OCR que reconocen los símbolos escritos y lo convierten en textos editables en la computadora (Map, 2012). La realización de este trabajo se centra fundamentalmente en este tipo de digitalización.

¿Qué es un software o motor OCR?

Son aplicaciones dirigidas a la digitalización de textos en imágenes. Identifican automáticamente símbolos o caracteres que pertenecen a un determinado alfabeto, a partir de una imagen para almacenarla en

forma de datos, con los que puede interactuar (editar, seleccionar, copiar y pegar) mediante un programa de edición de texto (España, 2007).

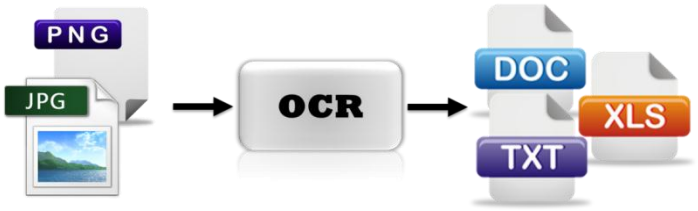

Figura 1. Proceso OCR

Metadatos:

No es más que un dato estructurado sobre la información, o sea, información sobre información, o de forma más simple, datos sobre datos. Son datos que se pueden guardar, intercambiar y procesar por medio del ordenador y que están estructurados de tal forma que permiten ayudar a la identificación, descripción clasificación y localización del contenido de un documento o recurso que, por tanto, también sirven para su recuperación (Lapuente, 2010).

1.3. Análisis de motores OCR

A continuación se describen tres motores de OCR (Tesseract, GNU Ocrad y GOCR). Es importante conocer las ventajas y desventajas que muestran cada uno, con el propósito de seleccionar el más conveniente a la solución del módulo Digitalización de Documentos. Para la determinación de cuál de estos motores debe ser implementado en este caso, se hace referencia a varias tablas que contienen la información necesaria para esta decisión.

1.3.1. Tesseract

Es un software para el reconocimiento óptico de caracteres. Originalmente desarrollado por Hewlett-Packard [1]como software propietario entre 1985 y 1995, tras diez años sin ningún desarrollo, fue liberado como código abierto en el año 2005. Actualmente está siendo liderado por Google, bajo la dirección de

[1] **Hewlett-Packard**: más conocida como **HP**, es una de las mayores empresas de tecnologías de la información del mundo. Fabrica y comercializa hardware y software además de brindar servicios de asistencia relacionados con la informática.

Ray Smith, su desarrollador original. Tesseract es uno de los mayores expertos en la tecnología OCR, distribuido bajo la licencia Apache 2.0. Está considerado como uno de los motores OCR libres con mayor precisión disponible actualmente (Bedwyr, 2008). Es multiplataforma, capaz de leer una amplia variedad de formatos de imagen y convertirlos a texto en más de 40 idiomas**.**

Sin embargo, posee algunas limitaciones

- No posee interfaz gráfica: la interacción con el usuario es a través de una consola, por lo que se necesita que el usuario tenga conocimientos avanzados del sistema operativo y la aplicación.
- Procesa solo una imagen a la vez: al procesar solo una imagen, cuando se está digitalizando un documento que tiene asociado varias páginas se debe realizar esta operación tantas veces como imágenes se tenga.
- El idioma por defecto es inglés: para procesar documentos en otros idiomas, es necesario descargarlos e instalárselos.
- El fichero de salida es en formato bruto, sin permitir la corrección ortográfica.

1.3.2. GNU Ocrad

Es un programa de OCR basado en un método de extracción de caracteres. Ocrad lee una imagen en formato pbm (mapa de bits), pgm (escala de grises) o ppm (color), y produce texto en formato byte (8-bit) o UTF-8 (Diaz, 2011).

Ocrad Incluye un analizador de composición, capaz de separar las columnas o bloques de texto que forman normalmente las páginas impresas. Puede ser usado como aplicación autónoma en modo texto, o como complemento de otros programas. Esta desarrollado sobre Unix (Diaz, 2011).

1.3.3. GOCR

Es un motor OCR desarrollado bajo la Licencia Pública GNU. Convierte las imágenes escaneadas a un archivo de texto. Joerg Schulenburg comenzó el programa, y ahora dirige un equipo de desarrolladores.

GOCR puede abrir diferentes formatos de imagen y se puede utilizar con diferentes *front-end*[2], lo que hace que sea muy fácil de portar a diferentes sistemas operativos y arquitecturas (Schulenburg, 2010).

El tamaño de fuente que soporta es de 20-60 píxeles. Los formatos de imagen que admite son pnm, pbm, pgm, ppm, otros formatos como pnm.gz, pnm.bz2, png, jpg, tiff, gif, bmp se convierten automáticamente utilizando el *netpbm-prog.* Este sistema presenta dificultades con letras cursivas, texto manuscrito. Los resultados obtenidos cuando las imágenes contienen grandes ángulos de inclinación no son satisfactorios, por lo que se necesita de un gráfico de alta calidad para conseguir buenos resultados(Schulenburg, 2002).

1.4. Resultado de análisis para seleccionar el motor OCR

La siguiente tabla muestra un resumen de las características principales a tener en cuenta en la comparativa de los motores OCR descritos anteriormente.

Tabla 1 Comparación entre motores OCR (Talledo, 2011)

Características	Ocrad	GOCR	Tesseract
Precisión en el reconocimiento	Buena	Buena	**Excelente**
Tiempo de respuesta	Excelente	Mala	**Buena**
Sistema Operativo	Unix	Windows, Linux	**Windows, Linux, Mac OS**
Licencia que utiliza	GNU(General Public License)	GNU(General Public License)	**Apache License 2.0**
Calidad de imagen de entrada	Buena	Excelente	**Buena**
Última actualización en repositorio	22/01/12	31/05/12	**03/11/12**

[2] **front-end**: es la parte del software que interactúa con el o los usuarios

Precisión en el reconocimiento: se refiere al comportamiento de estos motores OCR en la extracción de caracteres, a la cantidad de caracteres reconocidos correctamente.

Tiempo de respuesta: esta característica permite conocer cuál es el más rápido en el reconocimiento. Ser más rápido no implica tener mejor precisión.

Sistema Operativo: se refiere a los sistemas operativos que pueden utilizarse en la ejecución de estos sistemas. El que puede ejecutarse en todos los sistemas operativos sería el más conveniente.

Licencia que utiliza: analiza el tipo de licencia que utiliza el motor OCR. La licencia conveniente es la que permite integrarse con software propietario y no sea viral.

Calidad de imagen de entrada: se refiere a cómo influye la calidad de la imagen en la extracción de caracteres. El que acepte una imagen con menor calidad sería el más conveniente para documentos históricos.

Última actualización en el repositorio: se refiere al más actualizado. Esto influye en las demás características, pues en este caso si se agregan nuevas funcionalidades contantemente los resultados son mejores.

Para determinar cuál de los motores OCR arroja mejor resultado, se realizó un análisis de ejecución y una medición de los errores generados. Se utilizaron dos imágenes con diferentes tipos de fuente para realizar la comparativa, teniendo en cuenta la siguiente característica:

Eficacia: permite seleccionar cual realiza correctamente o con menor margen de error el proceso de OCR para las imágenes que se prueban. A continuación se muestra una comparativa donde se recoge el porcentaje de error en el proceso de detección y extracción de caracteres. El motor OCR más eficiente es que menor porcentaje de error posea.

Tabla 2. Resultado de eficacia con Imagen 1 (Talledo, 2011)

OCR	Total caracteres	Reconocidos correctamente	No reconocidos	Porcentaje de error
GOCR	37	35	2	5,41
Ocrad	37	29	8	21,62
Tesseract	**37**	**36**	**1**	**2,70**

Tabla 3. Resultado de eficacia con Imagen 2 (Talledo, 2011)

OCR	Total caracteres	Reconocidos correctamente	No reconocidos	Porcentaje de error
GOCR	37	34	3	8,10
Ocrad	37	30	7	18,92
Tesseract	**37**	**37**	**0**	**0**

Después de valorar los aspectos descritos de los tres motores OCR en estudio, se decide usar Tesseract, para el desarrollo del módulo Digitalización de Documentos. Los resultados obtenidos expresan los motivos que enuncia esta decisión.

- Cuenta con un grupo de desarrollo que continuamente introduce nuevas versiones con mejoras sobre el sistema.
- Licencia que permite adaptaciones o futuros usos comerciales.
- Es más eficiente porque ofrece resultados con un pequeño porcentaje de error.

1.5. Análisis de sistemas que integran a Tesseract.

Tesseract es usado como motor OCR por un gran número de aplicaciones de digitalización de documentos que van más allá del simple reconocimiento de caracteres, incluyen varias funcionalidades añadidas. En este epígrafe se realiza un estudio de estas, haciendo énfasis en las nuevas funcionalidades que introducen para sistemas de este tipo.

FreeOCR:

Es un programa gratuito de reconocimiento de caracteres para Windows, tiene soporte para la mayoría de los escáneres, puede procesar PDFs escaneados e imágenes con una gran variedad de formatos. La salida puede ser un fichero de texto plano o un documento Word.

Usa la versión 3.0.1 del motor OCR Tesseract. Este sistema se distribuye bajo la licencia Apache 2.0. Incluye un instalador para Windows y muy fácil de usar, soporta múltiples páginas, PDF y documento de fax (FreeOCR, 2011).

GImageReader

Brinda una interfaz de usuario para Tesseract y soporta la selección de columnas y partes del documento. Posee la capacidad de mostrar imágenes y archivos PDF, permite seleccionar un área para el reconocimiento en varios lenguajes, revisar la ortografía y elimina saltos de línea en el texto de salida. Puede realizar el escaneo de los archivos desde el propio programa (Medin, 2012).

OCRFeeder

Permite realizar corrección ortográfica sobre el texto extraído y definir cuadros delimitadores para realizar la corrección. Soporta varios idiomas y permite que el usuario seleccione el idioma para el reconocimiento. Posee un componente que permite definir los estilos de los párrafos y el texto del documento. Limpia las imágenes de entrada, para lograr una mayor efectividad del OCR. El documento con el texto extraído pude ser exportado a varios formatos. OCRFeeder fue desarrollado como proyecto de tesis de Maestría en Ciencias de la Computación de Joaquim Rocha (Rocha, 2011).

TextRipper

Es una aplicación OCR que permite escanear documentos con varias páginas y seleccionar múltiples archivos. Esta aplicación trae en su instalación Tesseract como motor OCR por defecto, pero puede integrarse opcionalmente con OCRAD (TextRipper, 2007).

YAGF

Es una interfaz gráfica para Tesseract, que permite el reconocimiento de texto en la plataforma Linux. Con YAGF se pueden escanear imágenes a través de XSane, importar páginas de documentos PDF, realizar el procesamiento de reconocer textos con caracteres cuneiformes [3]desde un centro de mando único. YAGF también hace que sea fácil de escanear y reconocer varias imágenes en secuencia (YAGF, 2013).

Los sistemas analizados anteriormente no brindan una solución al problema planteado por ser aplicaciones de escritorio que no brindan una API[4] de integración.

1.6. Metodología, Herramientas, Lenguajes y Tecnologías

1.6.1. Metodología de desarrollo

Una metodología de desarrollo de software, es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software; indica quién debe hacer cada actividad, cuándo hacerla y qué se debe hacer (Letelier, 2003).

La metodología seleccionada para el desarrollo del sistema Arkheia es RUP (Proceso Unificado de Desarrollo). El Proceso Unificado es el conjunto de actividades necesarias para trasformar las requisitos de usuario en un software. Utiliza el Lenguaje Unificado de Modelado (UML) para modelar los artefactos que se generan durante su ciclo de vida (Jacobson, 2000).

[3] **Cuneiformes**: Tipo de escritura muy antigua que se escribió originalmente sobre tablillas de arcilla húmeda, mediante un tallo vegetal biselado en forma de cuña, de ahí su nombre.
[4] **API**: Application Programming Interface (conjunto de funciones y que ofrece una biblioteca para ser utilizada por otro software como una capa de abstracción.)

El ciclo de vida del software en RUP comprende cuatro fases: inicio, elaboración, construcción y transición. La fase de inicio define el alcance del proyecto y desarrolla los casos de uso del negocio. La elaboración define el plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

RUP se hace único por sus tres aspectos fundamentales:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

Los casos de uso representan los requisitos funcionales, se especifican, se diseñan y se utilizan para conducir el proceso de desarrollo. Los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba. La arquitectura establece lo que se tiene que hacer; esquematiza los niveles significativos de la organización, se centra en el armazón del sistema. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto (Jacobson, 2000).

El sistema Arkheia está dividido por módulos que serían los miniproyectos, comienzan con los casos de uso y continúan a través del trabajo de desarrollo subsiguiente análisis, diseño, implementación y prueba, que termina convirtiendo en código ejecutable los casos de uso que se desarrollaban en la iteración. De esta forma se coordina mejor el trabajo con un proceso que proporciona una guía para tener ordenadas las actividades, que dirija las tareas de cada desarrollador por separado y el equipo como un todo, y especifique los artefactos que deben desarrollarse en cada flujo de trabajo para documentar futuras generaciones.

Es conveniente el uso de esta metodología pues realiza la evaluación en cada fase, que permite la realización de cambios; funciona bien en proyectos de larga duración y en los que se ven involucradas muchas personas; sigue los pasos intuitivos necesarios a la hora de desarrollar el software y permite un seguimiento detallado en cada una de las fases de desarrollo.

1.6.2. Herramientas de Modelado

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades del proceso de software, como en análisis de requerimientos, modelado de sistemas, la depuración y las pruebas. Las herramientas CASE también incluyen algunas guías de procesos para los ingenieros de software (Sommerville, 2005).

Para generar y representar los artefactos del sistema se seleccionó Visual Paradigm For UML 8.0 como herramienta de modelado visual.

Visual Paradigm For UML 8.0: Es una herramienta de modelado que se caracteriza por su flexibilidad ya que es multiplataforma, puede ejecutarse sobre diferentes sistemas operativos. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Utiliza UML como lenguaje de modelado, ofreciendo soluciones de software que permiten a las organizaciones desarrollar las aplicaciones de forma rápida y satisfactoria (Paradigm, 2013).

El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos. Por esta razón se elige Visual Paradigm como herramienta de modelado para representar los artefactos de la aplicación, por las facilidades que brinda para la modelación de sistemas, además, el equipo de desarrollo está familiarizado con el uso de esta herramienta.

1.6.3. Herramientas de Desarrollo

Springsource Tool Suite (STS) 2.5: Es un entorno de desarrollo integrado multiplataforma de código abierto desarrollado por Spring Source que soporta una amplia gama de lenguajes de programación (SpringSource, 2011).

STS es una herramienta libre, destinada a construir aplicaciones empresariales enriquecidas por los proyectos de Spring Source. No sólo incluye herramientas para el desarrollo en lenguaje Java, sino también, para Spring, Groovy and Grails y Scala (Springsource, 2013).

TortoiseSVN 1.7: Es un cliente gratuito de código abierto para el sistema de control de versiones Apache Subversion. Esto significa que TortoiseSVN administra archivos y directorios a lo largo del tiempo. Los archivos se almacenan en un repositorio central. El repositorio es un servidor de archivos, que registra

todos los cambios que se hayan hecho a sus archivos y directorios. Esto le permite al usuario recuperar versiones antiguas de sus archivos y visualizar el historial de los cambios sobre estos (Stefan Küng, 2011).

1.6.4. Sistema Gestor de Base de Datos

PostgreSQL 9.1: PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más usado. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (PostgreSQL, 2010).

Apache Tomcat 7.0.30: Es un contenedor web desarrollado en Java con código abierto. Es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java (Apache, 2013).

Se seleccionó Apache Tomcat como contenedor web ya que puede ser utilizado en entornos que dispongan de la Máquina Virtual de Java, además es un software libre que puede ejecutarse sobre múltiples sistemas operativos, permite que el sistema desarrollado esté disponible por el tiempo que resulte necesario y porque puede ser integrado con el marco de trabajo seleccionado.

1.6.5. Tecnologías

Grails 2.1.1: Grails es un framework libre, para el desarrollo de aplicaciones web, desarrollado sobre el lenguaje de programación Groovy (Glen Smith, 2009)**.**

Para el desarrollo web de la aplicación Arkheia se escoge Grails, construido sobre cinco pilares: (Brito, 2009)

- Groovy para la creación de propiedades y métodos dinámicos en los objetos de la aplicación.
- Spring para los flujos de trabajo e inyección de dependencia.
- Hibernate para la persistencia.
- SiteMesh para la composición de la vista.
- Ant para la gestión del proceso de desarrollo.

Desde el punto de vista del diseño, Grails se basa en dos principios fundamentales: (Brito, 2009)

Convención sobre configuración:
Aunque en una aplicación existen archivos de configuración, es un poco probable que sean editados manualmente, el sistema se basa en convenciones. Por ejemplo todas las clases de la carpeta grails-app/controllers serán tratados como controladores, y se mapearan en las urls de la aplicación (Brito, 2009).

Don't repeat yourself ("No te repitas"):
La participación de Spring Container en Grails permite inyección de dependencias mediante loC (Inversión de control), de forma que cada actor en la aplicación debe definirse una única vez, haciéndose visible a todos los demás de forma automática (Brito, 2009).

Una aplicación Grails contiene lo necesario para desarrollar desde el primer momento. En ella se definen un conjunto de paquetes ordenados, la carpeta principal llamada grail-app que contiene todos los artefactos que el entorno irá generando (Brito, 2009).

La carpeta grails-app contiene:

Conf: Contiene los parámetros de configuración de la aplicación.

Controllers: Contiene todos los controladores.

Domain: Contiene las clases de dominio o entidades.

i18n: Contiene los archivos para internacionalización, es decir, los diferentes idiomas.

services: Contiene los servicios de Grails.

Taglib: Contiene las librerías de etiquetas.

views*:* Se almacenan las vistas de la aplicación.

La utilización de este framework simplifica el trabajo del desarrollador, agilizando el proceso de desarrollo. Cuenta con la integración GORM (Grails Object Relational Mapping) que se encargan de controlar el ciclo de vida de las entidades.

Bootstrap: Es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript, es de código abierto con licencia Apache. Ha sido desarrollado por Mark Otto y Jacbod Thornton de Twitter que recientemente liberó su versión 2.0. La mayor ventaja es que podemos crear interfaces que se adapten a los distintos navegadores, apoyándonos en un framework potente con numerosos componentes webs que ahorran esfuerzo y tiempo (Rodríguez, 2012).

1.6.6. Gestor Documental

Nuxeo: Es un gestor documental de escala empresarial basado en Java. Está preparado para un alto grado de modularidad y rendimiento escalable, permitiendo administrar los documentos de manera colaborativa estableciendo versiones y ciclos.

Nuxeo es utilizado como software de gestión documental, páginas web, registros, imágenes y desarrollo colectivo de contenido debido a sus disimiles características que facilitan el desarrollo de estas aplicaciones, entre las que se encuentran (Nuxeo, 2011):

- Gestión de documentos.
- Gestión de contenido web.
- Versionado a nivel de repositorio.
- Gestión de registros.
- Gestión de imágenes.
- Publicación integrada.
- Flujo de trabajo basado en jBPM.
- Búsquedas implementadas con el motor Lucene.
- Servidores descentralizados.
- Soporte de varios idiomas.
- Empaquetamiento de aplicación portable.
- Soporte multiplataforma (Windows, Linux, Solaris, Mac OS).
- Interfaz gráfica basada en navegadores de Internet.
- Integración de escritorio con Microsoft Office y OpenOffice.Org.
- Soporte de clustering.

1.6.7. Lenguajes

UML: El Lenguaje Unificado de Modelado para la construcción de modelos. Permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema de notaciones (que, entre otras cosas, incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos (Larman, 1999).

Java: Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales. Es orientado a objetos, distribuido, interpretado y robusto (Jalón, 2000).

Una de las características más importantes es que los programas "ejecutables", creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos (Sun, 2000).

Java ha sido probado, ajustado y ampliado por toda una comunidad. Más de nueve millones de los desarrolladores de Java la convierten en la comunidad de desarrollo más grandes y activas del planeta. Con su versatilidad, eficacia y portabilidad, Java se ha convertido en incomparable para los desarrolladores (Java, 2013):

Groovy: Es un lenguaje dinámico y ágil para la JVM (Máquina Virtual de Java), tiene las capacidades y fortalezas de Java pero con características adicionales muy poderosas inspiradas en lenguajes cómo Python, Ruby, Smalltalk entre otros. Pone a disposición de los desarrolladores las nuevas características de la programación en la mayoría de los casos sin necesidad de una curva de aprendizaje elevada. Integra todas las características de Java existentes (Groovy, 2013).

Javascript: Es un lenguaje de programación interpretado del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado (Alvarez, 2001)**.**

Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente (Alvarez, 2001).

Las principales características del Javascript son:

- Es simple, no hace falta tener unos amplios conocimientos de programación para poder hacer un programa en Javascript.
- Está orientado a objetos de forma limitada ya que no maneja los conceptos como la herencia, los métodos, como el C++.
- Es dinámico, Javascript responde a eventos producidos por el propio usuario en tiempo real (Risco, 1998).

HTML: Es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. HTML también le indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto, las cuáles conectan diferentes documentos-ya sea en su computadora o en otras- así como otros recursos de Internet (Musciano, 1999).

1.6.8. Librerías

OpenCV: OpenCV (Open Source Computer Visión Library) es una biblioteca libre de visión artificial, que proporciona un alto nivel de funcionalidades. Esta biblioteca está escrita en C y C++ y corre en las plataformas de Linux, Windows y Mac OS X. Proporciona un alto nivel de funciones para el procesado de imágenes. Esta librería permite a los programadores crear aplicaciones poderosas en el dominio de la visión digital (Bradski, 2008).

OpenCV es multiplataforma. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de Visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica. Sus principales características son (Bradski, 2008).

- Manejo de estructuras básicas.
- Procesamiento de imágenes.
- Análisis estructural.
- Análisis de movimiento y seguimiento de objetos.
- Reconocimiento de objetos

- Calibración de cámara y reconstrucción de escenas 3D.
- Manejo de escenas de vídeo.
- Construcción de interfaces gráficas de usuario (GUI).

Para el desarrollo del módulo Digitalización de Documentos se utilizó la funcionalidad *Threshold* que posee OpenCV para el preprocesamiento de las imágenes. Se convierte la imagen a una escala de grises (blanco y negro), a las cuales no se le distinguen muy bien los caracteres debido al deterioro o suciedad. A continuación se describe el funcionamiento de este método:

Threshold (image, image, threshold,colour,cv.CV_THRESH_BINARY): El primer parámetro es la imagen de origen, el segundo la imagen de destino, el tercer parámetro es el punto de umbral, es un valor numérico sobre el cual se realiza una comparación con el resto de los píxeles. Para los últimos parámetros se definen valores de blanco (255) y negro (0), con el objetivo de hacer notar los caracteres del fondo de la imagen.

1.7. Conclusiones parciales

En este capítulo se especificaron los conceptos relacionados con los procesos a informatizar para un mejor entendimiento del negocio. Se analizaron soluciones informáticas de sistemas OCR y se seleccionó Tesseract como motor OCR para el desarrollo del módulo Digitalización de Documentos. Se analizaron aplicaciones que usan a Tesseract como motor OCR y se concluyó que ninguna de las analizadas se ajuste a las necesidades del Sistema Arkheia. Se estudiaron y describieron las herramientas, lenguajes, tecnologías y la metodología, seleccionadas para el desarrollo del módulo Digitalización de Documentos.

Capítulo 2. “Características del Sistema”

2.1. Introducción

El presente capítulo muestra las características del sistema a desarrollar y los procesos que son objeto de automatización, se hace mención a los casos de uso del módulo Digitalización de Documentos y se describe el marco de trabajo para el desarrollo del Arkheia. Con la descripción de los casos de uso y la arquitectura se procede a diseñar el módulo mediante la elaboración de sus diagramas de clases y sus diagramas de secuencia.

2.2. Propuesta de solución

Dando respuesta a la problemática planteada y haciendo uso de las herramientas y metodología seleccionada para la construcción del sistema Arkheia se propone la implementación del módulo Digitalización de Documentos. Mediante el cual serán digitalizados los documentos de los archivos.

La digitalización de los documento en el sistema Arkheia se realiza incorporando las imágenes a un documento. El usuario comprueba la calidad de la imagen y selecciona realizar el preprocesamiento para las imágenes de baja calidad, que convierte la imagen a una escala de grises; en otro caso procede directamente al proceso de reconocimiento óptico de caracteres, que puede ser en los idiomas; inglés o español. Obteniendo finalmente un fichero de texto con los caracteres extraídos de las imágenes, que el usuario puede revisar, modificar, guardar y exportar.

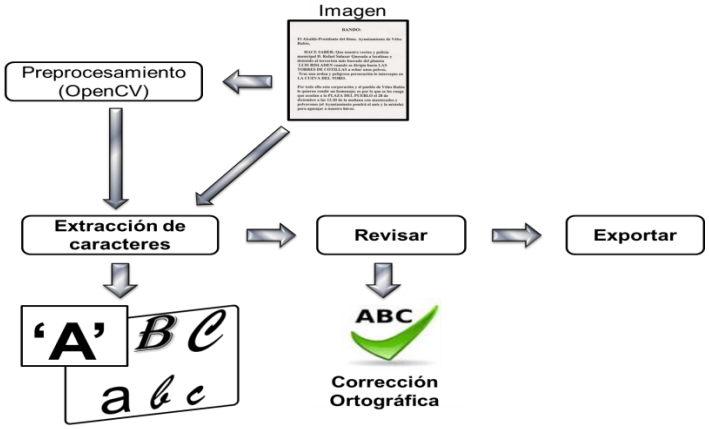

Figura 2 Propuesta de Solución

2.3. Modelo de Dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Se describe mediante diagramas UML (diagramas de clases), mostrando a los clientes, usuarios, revisores, y a otros desarrolladores las clases de dominio y cómo se relacionan unas con otras mediante asociaciones (Jacobson, 2000).

En el presente trabajo se realiza el modelo de dominio debido a que no existe un negocio definido, por lo cual, no se pueden determinar los procesos y roles del proceso de negocio, haciéndose complejo y poco estricto la descripción de los mismos. Por tanto se describirán los conceptos relacionados a las clases del dominio así como las asociaciones y atributos de las mismas.

2.3.1. Descripción de los principales conceptos

Conocer los principales conceptos que se manejan en el dominio del sistema permitirá un mejor entendimiento del modelo de dominio.

Descripción del Documento: Facilita la búsqueda de los documentos, brindando información que estos contienen.

Digitalización: Contiene el listado de las imágenes pertenecientes al documento.

Técnico de la institución: Persona que realiza la acciones de preprocesamiento, procesamiento y revisión***.***

Preprocesar con OpenCV: Incluye la librería OpenCV para el tratamiento de las imágenes, a las cuales no se le distinguen muy bien los caracteres debido al deterioro o suciedad.

Procesar con Tesseract: Contiene el motor OCR Tesseract para realizar la extracción de los caracteres que contienen las imágenes.

Caracteres de las Imágenes: Conjunto de letras o símbolos que brinda alguna información contenida en una imagen.

Revisión: Permite al usuario realizar al revisión de los caracteres extraídos, muestra sugerencias ortográficas para que éste pueda corregir los errores.

Corrector Ortográfico: Contiene varios diccionarios, los cuales mostrarán sugerencias de palabras al usuario en la revisión.

Documento Digital: Es el fichero de texto que posee los caracteres extraídos de las imágenes luego del procesamiento.

2.3.2. Diagrama de clases del Dominio

Utilizando la notación UML, un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Muestran:

- Objetos del dominio o clases conceptuales.
- Asociaciones entre las clases conceptuales.
- Atributos de las clases conceptuales.

Diagrama de clases del Modelo de Dominio

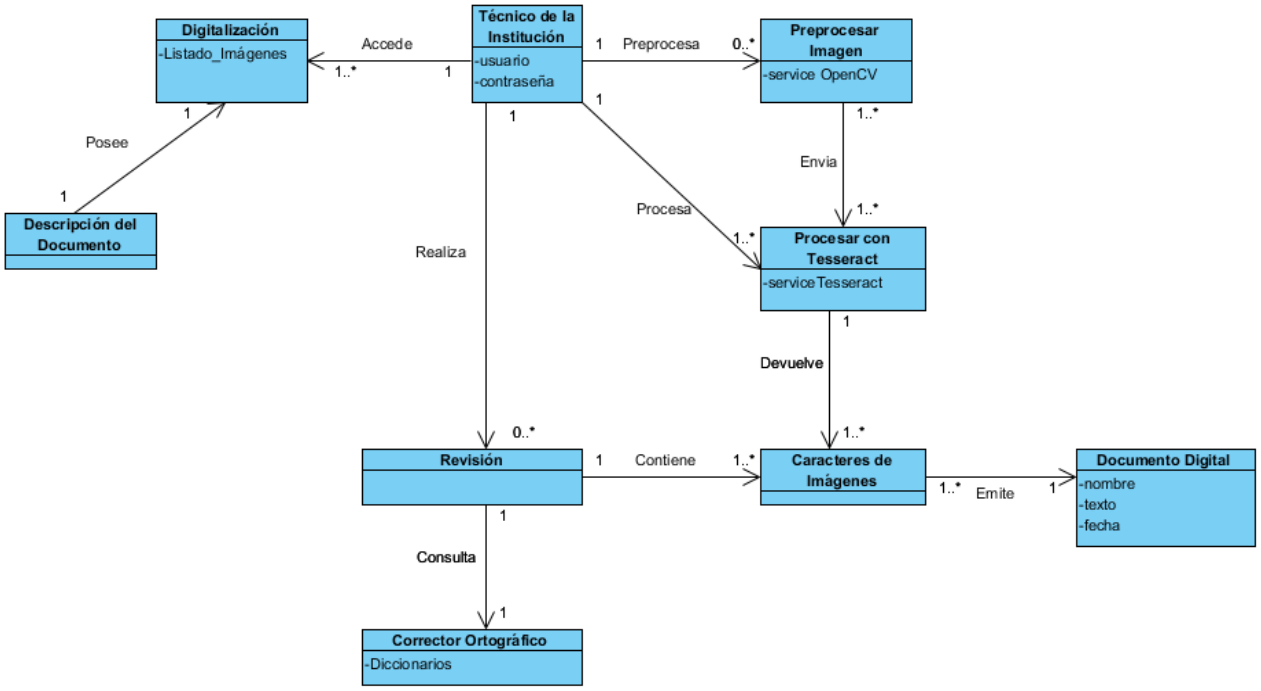

Figura 3. Diagrama de clases de Modelo de Dominio

Para realizar la Digitalización de los Documentos se debe tener en el sistema un documento descrito, la descripción no es más que las características fundamentales que posee un documento, por tanto, esta descripción posee una lista de imágenes que pertenecen al documento previamente digitalizado. Estas imágenes son procesadas con el motor OCR Tesseract para realizar la extracción de caracteres, en algunos casos son preprocesadas con OpenCV para el mejoramiento de la calidad. Luego de estos dos procesos se obtiene un fichero que contiene los caracteres de las imágenes digitales, que se puede clasificar como un documento digital, este se revisa por la persona encargada de la revisión con la ayuda de un corrector ortográfico. Es así como se obtiene finalmente un documento digitalizado.

2.4. Requisitos

Requisitos o requerimientos es la condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo; los requerimientos deben ser especificados por escrito, estar claros y precisos. Estos se clasifican en funcionales y no funcionales (Laguna, 2013).

2.4.1. Técnicas de captura de requisitos

Las técnicas utilizadas para la captura de requisitos del módulo Digitalización de Documentos, permitieron extraer los requisitos de una forma más segura, debido a que en ocasiones usuarios pueden tener dificultad para describir sus tareas, pueden dejar la información importante sin especificar, o pueden estar poco dispuestos o cooperar. Estas técnicas fueron:

Tormenta de Ideas*:* Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios, ayuda a generar una gran variedad de vistas del problema y a formularlo de diferentes formas (Sommerville, 2005). Para el desarrollo del módulo se realizaron varias reuniones donde intervinieron los roles principales del proyecto (Líder de proyecto, Arquitectos y Analistas).

Estudio de documentación: El estudio de la documentación consiste en realizar una lectura en profundidad, basada en documentos sobre el dominio del problema del sistema a desarrollar. Algunos de los principales documentos que se consultaron para el este trabajo son manuales de procedimientos y funciones que se realizan en las instituciones de archivos, normativas y legislaciones. Los documentos estudiados son:

- Legislación Archivística: Lo que el Archivero debe conocer (Navarro, 2002).
- El Archivo y el Archivero (Herrero, 1997).
- Conservación y Restauración de material (Paris, 2002).
- Propuesta de requisitos funcionales para la Gestión de Documentos Archivísticos Electrónicos en la administración central del estado cubano (Mugica, 2006).
- La Importancia de la Digitalización de Archivos para la Biblioteca (Cam, 2007).

2.4.2. Requisitos Funcionales

Los requisitos funcionales que se describen a continuación detallan lo que debe hacer el módulo. Especifican la manera en que éste debe reaccionar a determinadas entradas y como debe comportarse en situaciones particulares. El 70% de los requisitos corresponden a la técnica Tormenta de Ideas, el resto al Estudio de la Documentación realizado.

RF1: Buscar documento

El sistema debe permitir realizar la búsqueda de los documentos, mostrar al usuario y este selecciona el que desea digitalizar.

RF2: Adicionar imágenes.

El sistema debe permitir la incorporación de una o varias imágenes a un documento seleccionado.

RF3: Procesar la imagen.

Cuando la imagen que se adiciona a un documento no es de muy buena calidad el sistema debe permitir realizar un proceso que mejore la calidad de la misma.

RF4: Extraer los caracteres de las imágenes.

Con la ayuda de un motor OCR, el sistema debe permitir la extracción los caracteres que contienen las imágenes.

RF5: Revisar digitalización.

El sistema debe permitir la revisión de los caracteres extraídos que no fueron reconocidos correctamente por el motor OCR, con la ayuda de un corrector ortográfico se deben mostrar sugerencias para corregir estas palabras.

RF6: Señalar las palabras dudosas.

El sistema debe señalar las palabras que no fueron reconocidas correctamente para que el usuario pueda corregir.

RF7: Exportar a pdf.

El sistema debe permitir que los ficheros de textos con los caracteres extraídos sean exportados a pdf.

RF8: Seleccionar el Idioma en el que se desea realizar la digitalización.

El sistema debe permitir al usuario seleccionar el idioma en que desee realizar la extracción de caracteres, ya sea en inglés o español.

RF9: Eliminar imagen.

El sistema debe permitir que el usuario elimine una imagen que se encuentre asociada a un documento.

RF10: Guardar revisión.

El sistema debe permitir que se guarden los cambios realizados en la revisión.

2.4.3. Requisitos No Funcionales

Los requisitos no funcionales, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada /salida y las representaciones de datos que se utilizan en las interfaces del sistema. Por lo tanto, pueden especificar el rendimiento del sistema, la protección, la disponibilidad, y otras propiedades emergentes. Se identificaron en este trabajo los siguientes requisitos no funcionales:

RNF1. Requerimientos mínimos de Hardware:

El sistema debe contar con dos servidores, uno para la base de datos y otro para la aplicación. Estos deben tener 2 GB de RAM, un micro a una frecuencia de 2.3 GHZ, 100 GB de disco duro el de base de datos y 50 GB el de aplicaciones, para el servidor de Nuxeo la capacidad del disco duro depende de la documentación que se desee registrar en el sistema.

RNF2. Requerimientos de Software:

En las computadoras de los usuarios solo se requiere de un navegador Web Mozilla Firefox 17 o Google Chrome 23 además de un lector de pdf Adobe Reader 10 o en sus versiones superiores. En el caso del servidor de aplicaciones web se deberá tener instalado cualquier versión igual o superior al Apache Tomcat 7.0.30 y en el servidor de base de datos se debe tener instalado PostgreSQL 9.1, además de una Máquina Virtual de java 7 y como sistema operativo Windows o Linux.

RNF3. Requerimientos de apariencia o interfaz externa:

El sistema debe tener una apariencia profesional y un diseño gráfico sencillo, con la utilización de las tonalidades de los colores rojo, blanco y gris fundamentalmente. El texto incluido en las páginas de la aplicación debe ser de tipo arial 11.Debe mostrar mensajes de información tanto para el éxito o fallido de una operación

RNF4. Restricciones en el diseño y la implementación:

Se hace uso del estándar de codificación que propone el framework utilizado.

- Framework de desarrollo Grails 2.1.1.
- Framework de Presentación Bootstrap.

RNF5. Requerimientos de Seguridad:

- **Confidencialidad:**

La información controlada por el sistema debe estar protegida de acceso no autorizado por lo cual se establecerá un nivel de acceso a la aplicación mediante la gestión de roles.

- **Integridad:**

La información solo puede ser modificada por quien está autorizado y de manera controlada.

RNF7. Requerimientos de portabilidad:

El sistema podrá ser usado bajo los sistemas operativos Linux o Windows.

2.4.4. Técnicas de validación de requisitos

Este proceso tiene por finalidad comprobar que los requisitos del software poseen todos los atributos de calidad; que sean consistentes, completos, precisos, realistas, verificables y definan lo que el usuario desea del producto final. La realización de estas actividades en este momento pretende evitar los altos costos que significaría el tener que corregir una vez avanzado el desarrollo.
Técnicas utilizadas para validar los requisitos:

Revisiones de requisitos:
Esta técnica permite verificar cada uno de los requisitos en cuanto a anomalías y omisiones, este proceso fue llevado a cabo por los analistas del proyecto Archivo y fueron detectados algunas inconveniencias como:

- Presencia de errores ortográficos y de concordancia.
- Ocurrencia de palabras ambiguas.

Como resultado surgieron algunas recomendaciones de cambios de textos para facilitar la comprensión de los mismos. Todas las anomalías fueron solucionadas y aceptadas, logrando una mayor interpretación de las funcionalidades del sistema.

Construcción de prototipos: Se muestran los prototipos de interfaz a los usuarios finales, con el objetivo de comprobar si el sistema cumple sus necesidades reales, y que entienda fácilmente la propuesta de requisitos.

Fueron aceptados el 90% de los requisitos propuestos, el resto fue modificado teniendo en cuenta los siguientes aspectos:

- Nombres de los componentes de la interfaz.
- Cambios de tipos de componentes.
- Eliminación y adición de botones.

La utilización de esta técnica facilitó la construcción de las vistas del sistema y el entendimiento de la propuesta de requisitos por parte de los analistas, arquitectos y jefe de proyecto.

2.5. Resumen de los Casos de Uso

A continuación se describen los casos de uso que reflejan el funcionamiento del sistema en cuanto a la gestión de la digitalización de los documentos del Arkheia. Es necesario el entendimiento de estos casos de uso para diseñar el módulo y para su posterior implementación. Se listan los casos de usos definidos y un resumen que describe su comportamiento.

Tabla 4. Resumen de los Casos de Uso

Caso de Uso	Nombre	Descripción
1.	Procesar Imagen	Este caso de uso tiene como objetivo procesar la imagen con OpenCV, permitiendo que la imagen sea mejorada para un mejor reconocimiento óptico de caracteres.
2.	Extraer caracteres de imágenes	Este caso de uso se encarga de extraer los caracteres de las imágenes asociadas a un documento y guardarlas en un fichero de texto dentro del Gestor de Contenidos Empresariales (Nuxeo).
3.	Revisar Digitalización	El objetivo de este caso de uso es revisar el documento extraído. Este inicia cuando el usuario desea revisar algún documento de los que fueron digitalizados, selecciona la opción “Revisar Digitalización”, en el documento modifica los errores señalados, finalmente lo guarda con dichos cambios.

4.	Exportar	Este caso de uso tiene como objetivo exportar el documento extraído a otro formato (pdf). Este le permite al usuario el exportar el fichero texto en formato pdf.
5.	Buscar Documento	Este caso de uso se encarga de buscar el documento cuando el usuario introduce los criterios de búsqueda del documento que desea.
6.	Identificar Palabras Dudosas	El objetivo de este caso de uso es identificar las palabras que el OCR no pudo reconocer correctamente, se muestran al usuario estas palabras con la ayuda de un corrector ortográfico.
7.	Eliminar Imagen	Este caso de uso permite al usuario eliminar una imagen asociada a un documento.
8.	Adicionar Imagen	Este caso de uso permite al usuario seleccionar las imágenes que desea asociar a un mismo documento.
9.	Seleccionar Idioma	Este caso de uso permite al usuario seleccionar el idioma en que desea realizar el reconocimiento de los caracteres.

2.5.1. Descripción de los Casos de Uso

Un caso de uso es la representación abstracta de una funcionalidad del sistema que provee un resultado de valor desde el punto de vista de sus actores (Neyem, 2013). A continuación se describen dos de los

casos de uso más críticos del módulo Digitalización de Documentos, las demás descripciones de casos de uso aparecen especificadas en los **Anexos**.

Tabla 5 Descripción CUS 1-"Adicionar Imagen"

Objetivo	Este caso de uso permite asociar las imágenes al documento.	
Actores	Usuario del sistema	
Resumen	El CU se inicia cuando el usuario selecciona las imágenes que desea adicionar a un documento, el sistema las adjunta estas imágenes a un mismo documento y termina el caso de uso.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Postcondiciones	Se asocian varias imágenes a un documento	
Flujo de eventos		
Flujo básico "Adicionar imagen"		
	Actor	Sistema
1.	Selecciona la opción "Examinar"	
2.		Muestra un cuadro de diálogo, que permite al usuario seleccionar las imágenes que desea adicionar al documento.
3.	Escoge la imagen que desea asociar (este paso se repite hasta que el usuario haya subido todas las imágenes) y oprime el botón	

	"Adicionar".	
4.		Muestra una interfaz con la opción: • Mejorar la imagen
5.	Si selecciona la opción Mejorar la imagen: Ver CU extendido "Procesar Imagen"	
6.	Oprime el botón "Extraer caracteres"	
7.		Ejecuta el CU incluido "Extraer Caracteres"
8.		Guarda las imágenes asociadas al documento y un fichero de texto que contiene los caracteres extraídos en el Gestor de Contenidos Nuxeo.
9.		Termina el CU.
Flujos alternos		
Actor		Sistema
6a. Cancelar		
Oprime el botón "Cancelar"		
		Cancela la operación y regresa al paso 1 del flujo normal de eventos "Adicionar Imagen".

Tabla 6 Descripción CUS – "Revisar Digitalización"

Objetivo	Revisar la digitalización realizada con el objetivo de corregir los errores.
Actores	Técnico

Resumen	El CU inicia cuando el técnico desea revisar algún documento de los que fueron digitalizados, busca el documento que desea revisar y selecciona la opción “Revisar”, modifica los errores que posee, finalmente lo guarda con dichos cambios y termina el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	• Debe estar autenticado el usuario en el sistema. • Se debe haber al menos una imagen adicionada a un documento.
Postcondiciones	Se modifica el documento y se guardan los cambios.

Flujo Normal de eventos

Flujo básico “Revisar Digitalización”

	Actor	**Sistema**
1.	Selecciona “Revisar”, de las acciones de la imagen asociada al documento.	
2.		Muestra el documento con su respectiva imagen, mostrando la opción: “Corrección Ortográfica”: ver sección 1: “Corrección Ortográfica”
3.	Modifica el documento.	
4.	Oprime el botón “Guardar”: ver sección 2: “Guardar Revisión”	
5.		Termina el CU.

Flujos Alternos

2a "Cancelar"		
	Actor	**Sistema**
1.	Oprime el botón "Cancelar"	
2.		Cancela la operación y regresa al paso 1 del flujo normal de eventos "Revisar Digitalización".
Sección 1: "Corrección ortográfica"		
	Actor	**Sistema**
1.	Oprime el botón "Corrección Ortográfica"	
2.		Ejecuta el caso de uso incluido "Identificar las palabras dudosas"
3.		Señala las palabras encontradas que no pertenecen al diccionario.
4.		Muestra un cuadro de diálogo con sugerencias de palabras a corregir.
Sección 1: "Guardar revisión"		
	Actor	**Sistema**
1.	Oprime el botón "Guardar"	
2.		Guarda los cambios realizados al documento original.

2.5.2. Diagrama de Casos de Usos

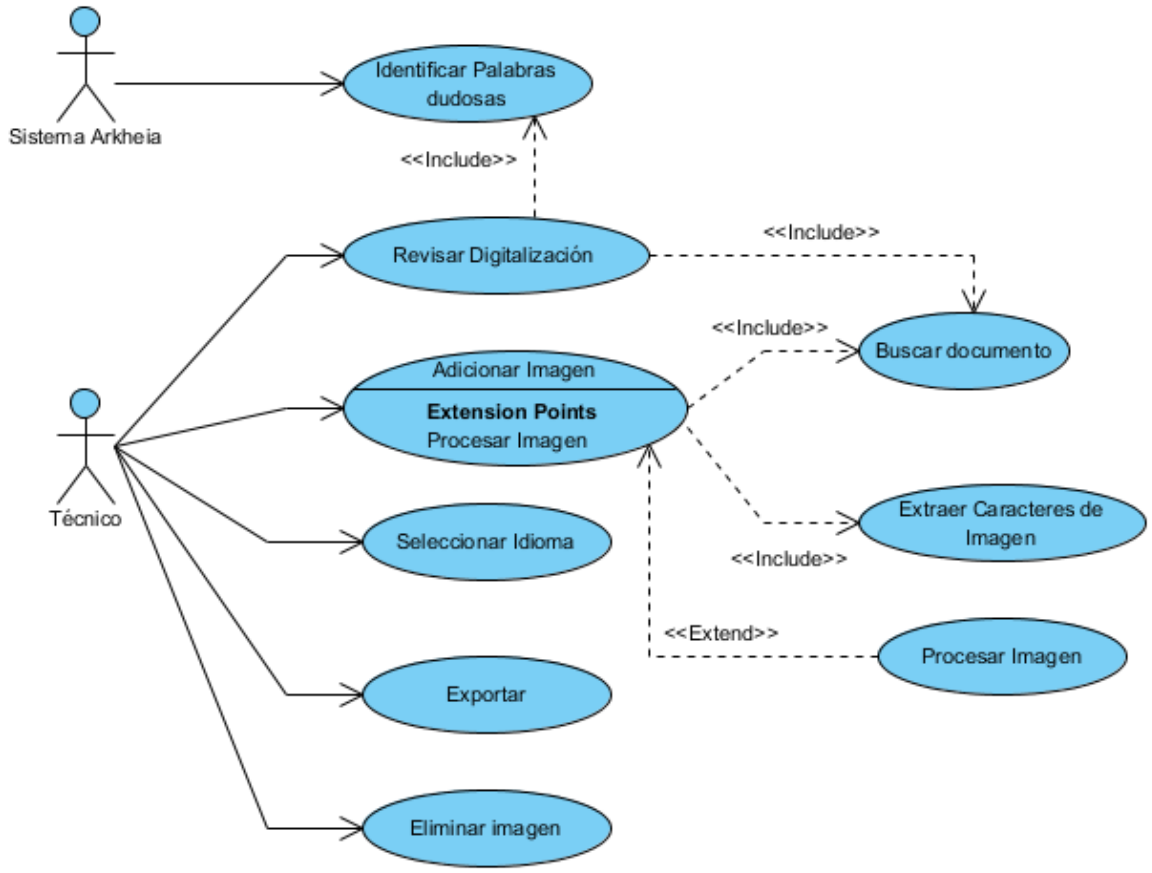

Figura 4 Diagrama de Casos de Uso para el módulo Digitalización de Documentos

2.5.3. Descripción de actores

Un actor representa un conjunto coherente de roles que los usuarios de los casos de uso juegan al interactuar con el sistema. Normalmente, un actor representa un rol que es jugado por una persona, un dispositivo de hardware, una base de datos o incluso otro sistema que interactúe (Neyem, 2013).

Tabla 7. Actores del sistema

Autor	Descripción
Técnico	Es el usuario que interactúa directamente con el sistema, una vez autenticado puede digitalizar un documento, analizarlo y exportarlo.

Sistema Arkheia	Se encargará de realizar acciones automáticas, identificará las palabras que no fueron reconocidas correctamente.

2.6. Conclusiones Parciales

En este capítulo se representó un modelo de dominio para un mejor entendimiento del contexto del sistema. Se propone una solución que responde a los requisitos funcionales especificados en la captura de requisitos. A partir del estudio de los casos de usos del sistema se concretó el módulo Digitalización de Documentos, dando paso al análisis y diseño del mismo.

Capítulo 3. “Análisis y Diseño”

3.1 Introducción

En el presente capítulo se describe el Análisis y el Diseño del sistema, que explican cómo transformar los requisitos en los productos de trabajo que especifican el diseño del software. Se realiza el modelado de los artefactos que se deben generar en estos flujos de trabajo, de análisis y de diseño, los cuales se encargan de describir cómo funcionan los casos de uso del sistema, dando paso a la implementación del mismo.

3.2 Análisis

Durante el análisis, se analizan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos, con el objetivo de conseguir una comprensión más precisa de los requisitos y una descomposición de los mismos, que sea fácil de mantener y que ayude estructurar el sistema entero (Jacobson, 2000).

A continuación se representan algunos de los artefactos que se generan en el flujo de trabajo Análisis para los casos de uso descritos anteriormente.

3.2.1. Diagramas de Análisis

El diagrama de clases de análisis representa una abstracción del diseño del sistema. A continuación aparecen representadas las clases de interfaz *Digitalización*, *Adicionar Imagen*, *Revisar Digitalización* utilizadas para la interacción entre el sistema y sus actores, la clase de control *DigitalizaciónController* representa la coordinación, secuencia, transacciones y control de los objetos, las clases entidades *Documento* y *Fichero Digitalizado* se encargan de modelar información asociado a estos conceptos.

Diagramas de clases de análisis

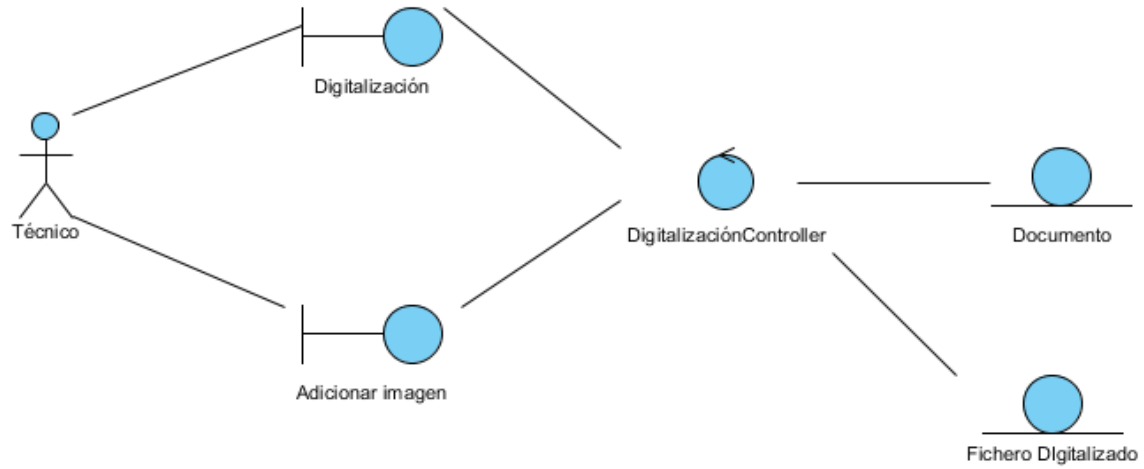

Figura 5. Diagrama de clase de análisis CU_ Adicionar Imagen

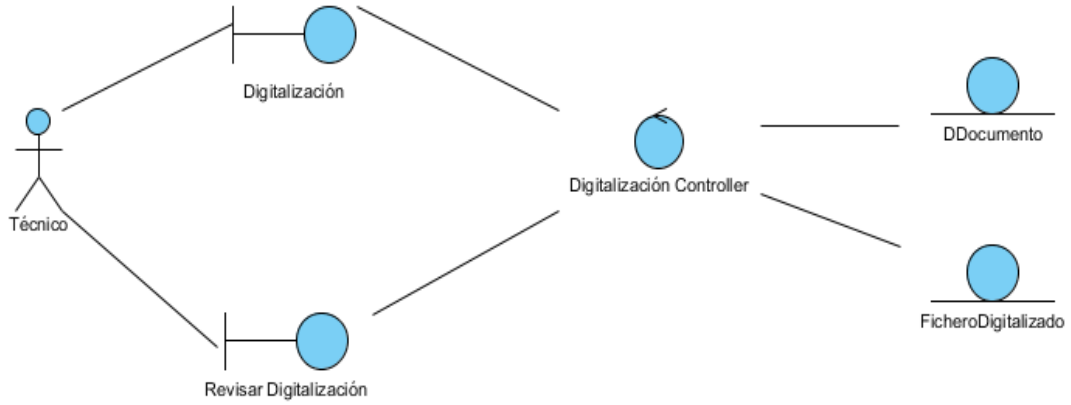

Figura 6. Diagrama de clase de análisis CU_Revisar digitalización

3.2.2. Diagrama de Paquetes

El siguiente diagrama de paquetes muestra las agrupaciones lógicas y las dependencias relacionadas con el módulo Digitalización de Documentos, facilitando una descomposición de la jerarquía lógica del módulo.

El subsistema Incorporación y Organización Manual incluye el módulo Descripción Manual de Documentos, que permite inicializar el proceso de Digitalización de Documentos con los documentos ya descritos, el paquete Nuxeo se encargará de guardar los ficheros que se generan en este proceso.

Diagrama de Paquetes del módulo Digitalización de Documentos

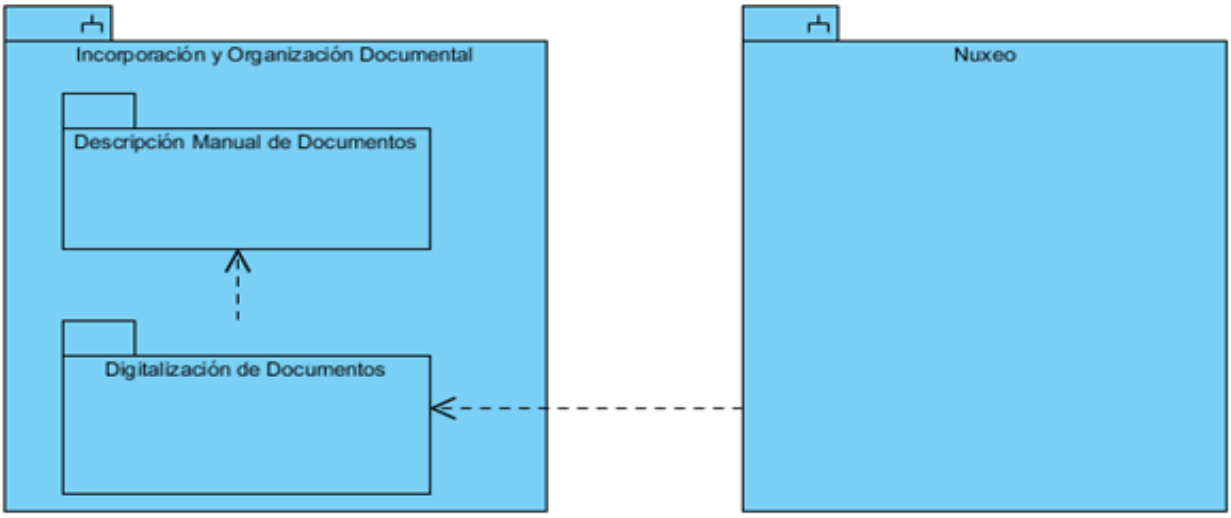

Figura 7. Diagrama de Paquetes del módulo Digitalización de Documentos

3.3. Diseño

El diseño es la continuación del análisis, es donde se modela el sistema y su forma, para que soporte todos los requisitos, incluyendo los no funcionales (Jacobson, 2000).

3.3.1. Arquitectura del Sistema

La arquitectura brinda una visión global del sistema, sus componentes y las relaciones entre ellos. Esto permite entenderlo, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar.

3.3.1.1. Descripción de la Arquitectura del Sistema:

La aplicación Arkheia consta de 4 subsistemas que incluyen 9 módulos. En la figura 9 se muestra como se encuentra estructurado el sistema.

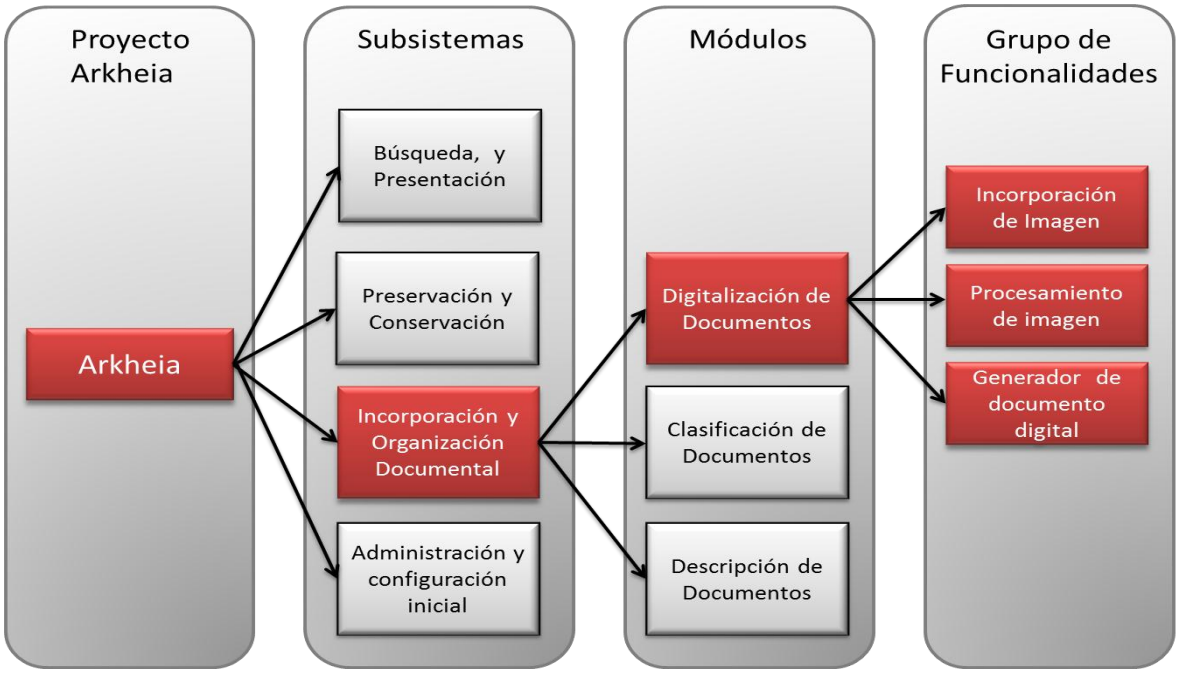

Figura 8 Estructura del sistema

El modulo Digitalización de Documentos es el módulo que pertenece al subsistema Incorporación y Organización Documental, dentro de sus funcionalidades se encargará de incorporar las imágenes digitalizadas, procesarlas y exportar el resultado como un documento digital.

La arquitectura para el módulo Digitalización de Documentos es la que define Grail (figura 11), que está conformada de tres capas lógicas principales: Capa Web, Capa de Servicios y Capa de Datos, basadas en el patrón **Modelo Vista Controlador (MVC)**. Cada capa brinda sus utilidades a la capa inmediatamente superior y se sirve de las prestaciones que le proporciona la inmediatamente inferior. Esta arquitectura en capas por su diseño proporciona la facilidad de modificar cada capa todo lo posible sin infligir daños o alteraciones a la capa inmediata.

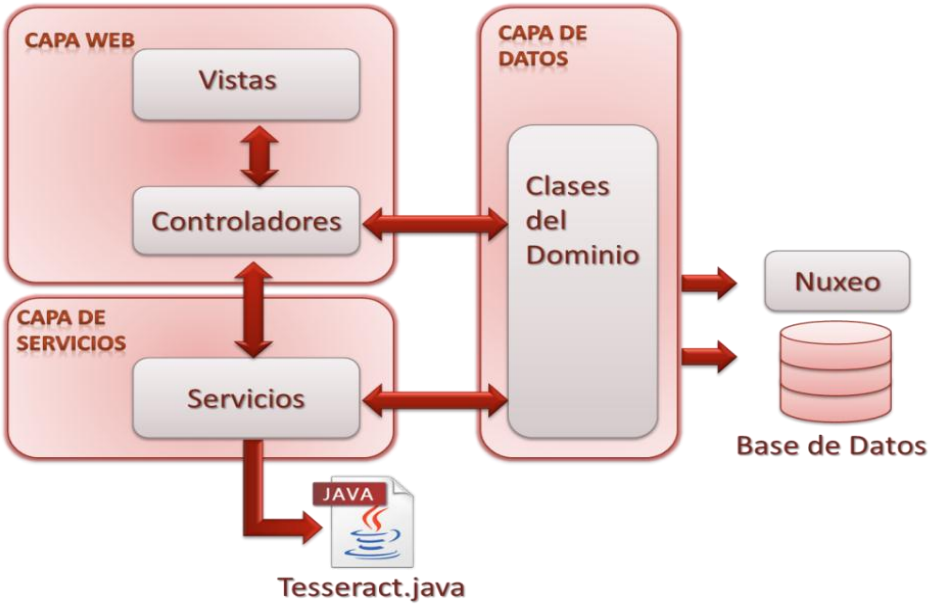

Figura 9 Arquitectura en capas según Grails.

Capa web: Se encarga de manejar dos partes, las Vistas y las Controladoras del patrón MVC.

- **Vistas**: Muestran una interfaz al usuario.
- **Controladoras**: Gestionan la aplicación mediante la recepción de las acciones del usuario.

Capa de datos: Se encarga de mantener los datos persistentes en la Base de Datos y Nuxeo.

Capa de servicios: Contiene los componentes encargados de implementar, manejar y validar la lógica de negocio de nuestra aplicación. Los servicios se ubican en la carpeta grails–app/services.

Cuando se trabaja con Grails se generan componentes en cada una de las capas y es el entorno el que se encarga de conectar unos con otros y garantizar su buen funcionamiento, obteniendo componentes fáciles de manejar, dando paso a la reutilización del código en mayor medida.

3.3.2. Patrones utilizados en el diseño

El **patrón** es una pareja de problema solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. Muchos patrones ofrecen orientación

sobre como asignar las responsabilidades a los objetos ante determinada categoría de problemas (Larman, 1999).

Modelo Vista Controlador

Este patrón es utilizado por Grails en el diseño y establece que los componentes de software deben organizarse en tres capas distintas:

- **Modelo, o capa de datos**: La capa está compuesta por las clases del dominio que representan y gestionan los datos manejados por la aplicación, se encargan de leer y escribir en la base de datos. Estas clases se ubican en la carpeta grail-app/domain (Brito, 2009).
- **Vista, o capa de presentación**: Los componentes de esta capa son responsables de mostrar al usuario el estado actual del modelo de datos, y presentarle las distintas acciones disponibles (Brito, 2009). En esta capa se encuentran los siguientes componentes:
  GSPs (Grails Server Pages / Páginas Servidoras de Grails): Ficheros con extensión .gsp, basados en las páginas JSP (Java Server Pages / Páginas Servidoras de Java), que agrupan los ficheros CSS (Cascade Style Sheet - Hojas de Estilo en Cascada), JavaScript y generan código html.
  TagLibs: Ficheros donde se definen las implementaciones de etiquetas personalizadas, para su utilización en los ficheros GSP.
- **Capas de control**: Contendrá los componentes que reciben las órdenes del usuario, gestionan la aplicación de la lógica de negocio sobre el modelo de datos, y determinan que vista debe mostrarse a continuación, así como la información que estas contienen (Brito, 2009).

Patrones GRASP

Representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 1999). GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades).

- **Experto**: Se utilizó este patrón en el diseño de las clases del domino *DDocumento,* esta clase tiene toda la información perteneciente a un documento, es la única que puede brindar cualquier información del mismo, como su descripción o revisión.

- **Controlador:** Se utilizó este patrón en las clases controladoras de Grails. Un ejemplo es la clase *Digitalización.controllers* que se encarga de manejar las acciones y el flujo de información entre las vistas, los servicios y el modelo de datos.
- **Alta Cohesión:** La utilización de este patrón se pone de manifiesto en la capa de servicios, colabora con las clases que lo utilizan para llevar a cabo las tareas, de esta forma se delegan responsabilidades (lógica del negocio) a los servicios y se liberan las clases controladoras de tantas operaciones, logrando que no pierdan la cohesión.
- **Bajo Acoplamiento**: La utilización de este patrón permite que en caso de producirse una modificación en alguna de las clases, esto implique lo menos posible al resto. Las clases de bajo acoplamiento no dependen de muchas otras.

Patrón Singleton

Por defecto, en Grails todos los servicios utilizan este patrón (Brito, 2009).Crea una instancia de un servicio durante todo el ciclo de vida de la aplicación, sería como un mecanismo de visibilidad global de dicha instancia. Esto permite mantener los datos persistentes de la variable correspondiente a la clase que se inyecta.

3.3.3. Diagrama de Clases de Diseño

A continuación se representarán los diagramas de clases de diseño con estereotipos web para los casos de uso Adicionar Imagen y Revisar Digitalización, los cuales describen gráficamente las especificaciones de las clases y las interfaces.

Se utilizan los estereotipos Página Cliente, Formulario y Páginas Servidoras, cada uno de estos tendrá una función específica dentro de la aplicación:

- **Página Cliente:** Interactúan con el usuario a través del navegador.
- **Formulario:** Colección de elementos de entrada que son parte de una página cliente. Envía los datos a la clase controladora.
- **Páginas Servidoras***:* Contiene código que se ejecuta en el servidor. Responsables de crear la página cliente.

Se representa, además, en este diagrama la clase controladora del módulo, encargada de recibir o manejar las peticiones de los usuarios u otros eventos del sistema, generan las interfaces de usuario y gestionan la lógica del negocio, apoyándose en los servicios. La clase servicio es la que implementa la lógica del negocio y el acceso a datos, utilizando las clases de dominio. Las clases de dominio representan los conceptos u objetos del sistema, y sus datos persisten en la base de datos.

Diagramas de clases del diseño con estereotipos web

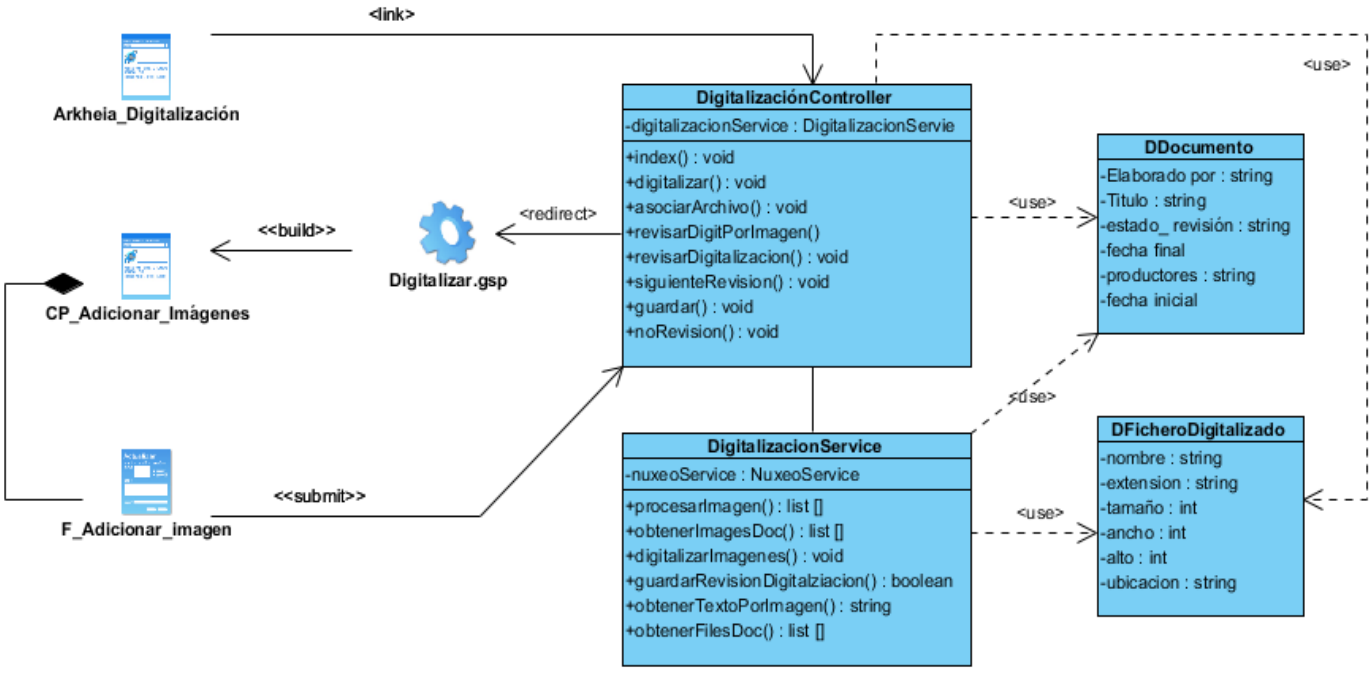

Figura 10. Diagrama de clases del diseño con estereotipos web CU_ Adicionar Imagen

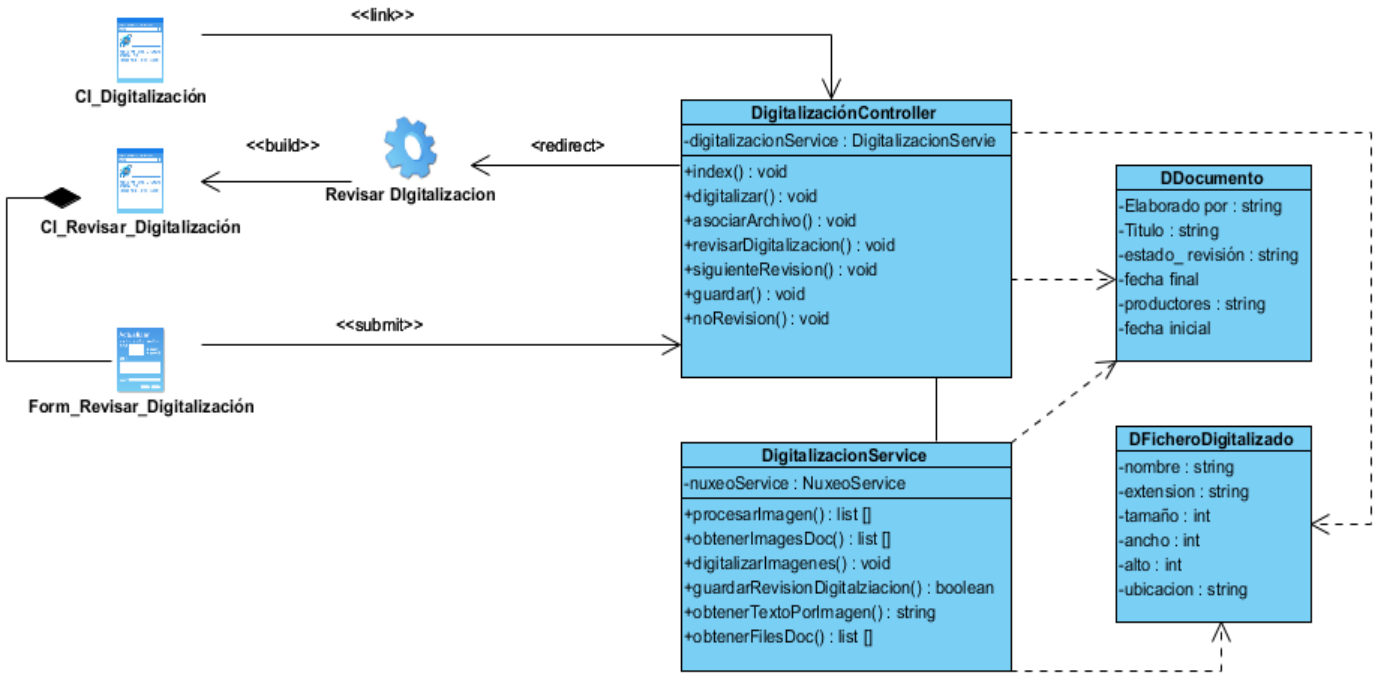

Figura 11. Diagrama de clases del diseño con estereotipos web CU_ Revisar Digitalización

3.3.4. Diagramas de interacción

Estos diagramas se utilizan para mostrar la vista dinámica del sistema, en ellos se describe como los grupos de objetos colaboran para conseguir algún fin, es decir, la forma de como un actor u objetos (clases), se comunican entre sí ante una petición a través de mensajes. Existen dos tipos de estos diagramas: los diagramas de colaboración y secuencia.

En este trabajo se representarán los diagramas de secuencia por comunicar más información conceptual.

Diagramas de Secuencia:

En el diagrama de secuencia que se muestra a continuación se representa el flujo de suceso del caso de uso “Revisar Digitalización”. En el mismo se encuentran representados a través de mensajes lo pasos necesarios para la revisión de los caracteres extraídos. Inicialmente el actor del caso de uso accede una interfaz principal, quien recibe la solicitud y envía una petición al controlador principal del módulo, este gestiona la revisión ordenando a una página servidora que genera la vista correspondiente, para que el

usuario realice acciones como corregir la ortografía y guardar los datos modificados en el servicio destinado a la gestión de contenidos.

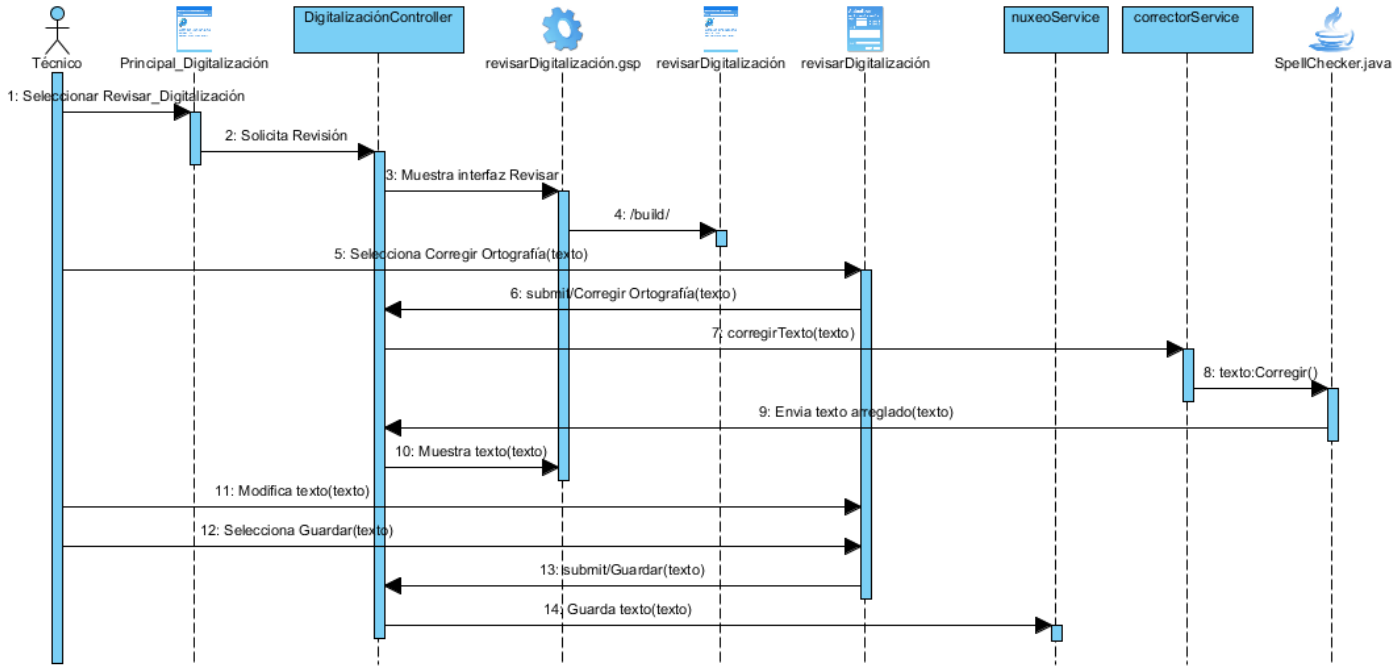

Figura 12 Diagrama de Secuencia para el CU Revisar Digitalización.

3.3.5. Validación del Diseño

Las métricas concebidas para evaluar la calidad del diseño y su relación con los atributos de calidad definidos para el módulo Digitalización de Documentos es la métrica de Tamaño Operacional de Clase (TOC), la cual se describe a continuación:

Métrica TOC: se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado. Estos atributos se describen a continuación:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de dominio o concepto.
- **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** consiste en el grado de reutilización presente en una clase, dentro de un diseño de software.

Tabla 8. Tamaño Operacional de Clase (TOC)

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 9. Rango de valores para determinar la categoría relacionada con los atributos (Responsabilidad, Complejidad de Implementación, Reutilización) de calidad de la métrica TOC.

Categoría	Criterio
Responsabilidad	
Baja	< =Prom. (5.27)
Media	Entre Prom. Y 2* Prom
Alta	> 2* Prom
Complejidad de implementación	
Baja	< =Prom
Media	Entre Prom. y 2* Prom
Alta	> 2* Prom
Reutilización	
Baja	> 2*Prom
Media	Entre Prom. y 2* Prom
Alta	<= Prom

Resultado de la evaluación con la métrica TOC:

Los elementos que demuestran estos resultados se encuentran en el **Anexo 9**

Figura 13. Representación en % de los resultados obtenidos de los intervalos de las operaciones de cada clase:

Resultado en % de la métrica TOC con el atributo Responsabilidad:

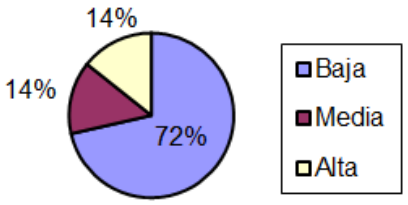

Figura 14.Atributo Responsabilidad

Resultado en % de la métrica TOC con el atributo Complejidad:

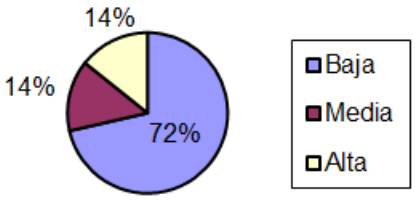

Figura 15. Atributo Complejidad

Resultado en % de la métrica TOC con el atributo Reutilización:

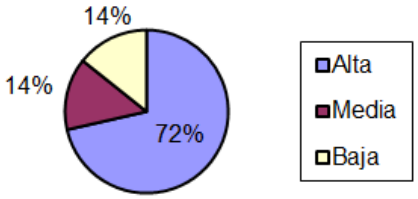

Figura 16. Atributo Reutilización

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad aceptable, teniendo en cuenta que la mayoría de las clases posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad si son satisfactorios (72% de las clases), de manera que se puede observar cómo se comporta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

3.4. Conclusiones parciales

Teniendo en cuenta que el flujo de trabajo de análisis y de diseño es uno de los más importantes en el ciclo de vida de cualquier software, se modelaron algunos de sus artefactos para los casos de uso más significativos, adquiriendo mejor visión de los principios que guiaron la implementación, con los componentes especificados en el marco de trabajo. Los resultados obtenidos en la validación del diseño fueron satisfactorios, demostrando que es simple y que los atributos de calidad alcanzan niveles favorables.

Capítulo 4. “Implementación y Prueba”

4.1 Introducción

Luego de haber terminado los flujos de trabajo de Análisis y Diseño y el modelado de los artefactos que generan estos. Se inicia la implementación del módulo Digitalización de Documentos, como parte de este flujo de trabajo se representan en este capítulo los diagramas de componentes y de despliegue correspondientes al módulo. Se realizan las pruebas correspondientes para comprobar la efectividad del producto final.

4.2 Implementación

En la implementación los elementos del diseño, fundamentalmente las clases, se implementan en términos de componentes, como ficheros de código fuente, librerías, entre otros. También se describe como dependen los componentes unos de otros.

Como parte de la implementación del módulo Digitalización de Documentos se representan los diagramas de componente y de despliegue.

4.2.1. Diagrama de Componentes

En el diagrama que se muestra a continuación se representan los archivos de código fuente y sus dependencias de compilación. Representa el módulo Digitalización de Documentos como componente principal y su relación con otros componentes del sistema Arkheia. El módulo utiliza la descripción de los documentos que contiene el módulo Descripción Manual, hace uso de la Plantilla Arkheia la cual brinda un estilo general a la vista del sistema, TagLib es la librería de etiquetas utilizada para encapsular la presentación, se utiliza en todo el sistema brindando reusabilidad y uniformidad en las vistas, Nuxeo como gestor documental que permite la gestión de los documentos digitalizados y la Base de Datos como componente para el almacenamiento y gestión de los datos.

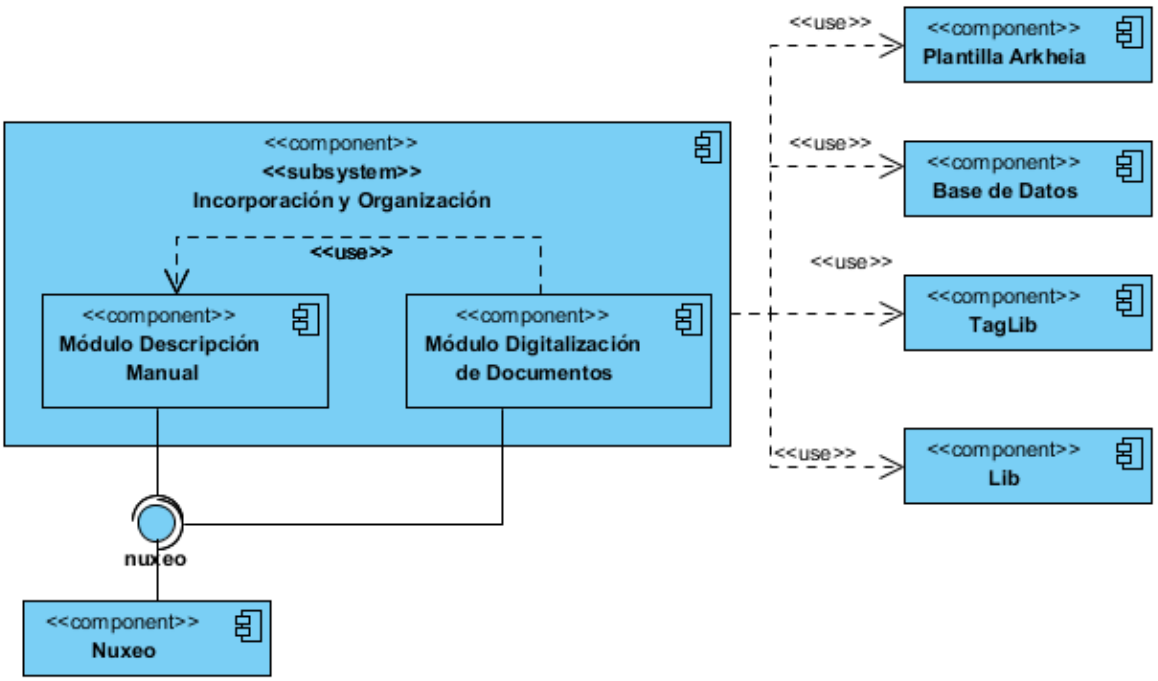

Figura 17 Componentes relacionados con el módulo Digitalización de Documentos

El modulo está compuesto por una clase controladora *DigitalizaciónController.groovy*, como se muestra en el siguiente diagrama, que establece la relación de todos los componentes que se utilizan en la implementación del mismo, *DigitalizacionService.groovy* que contiene todos los servicios utilizados y es responsable de implementar la lógica del negocio, *Tesseract.java* como motor OCR, *OpenCV* para el tratamiento de las imágenes, el componente *CorrectorOrtográfico* para la revisión de los caracteres extraídos.

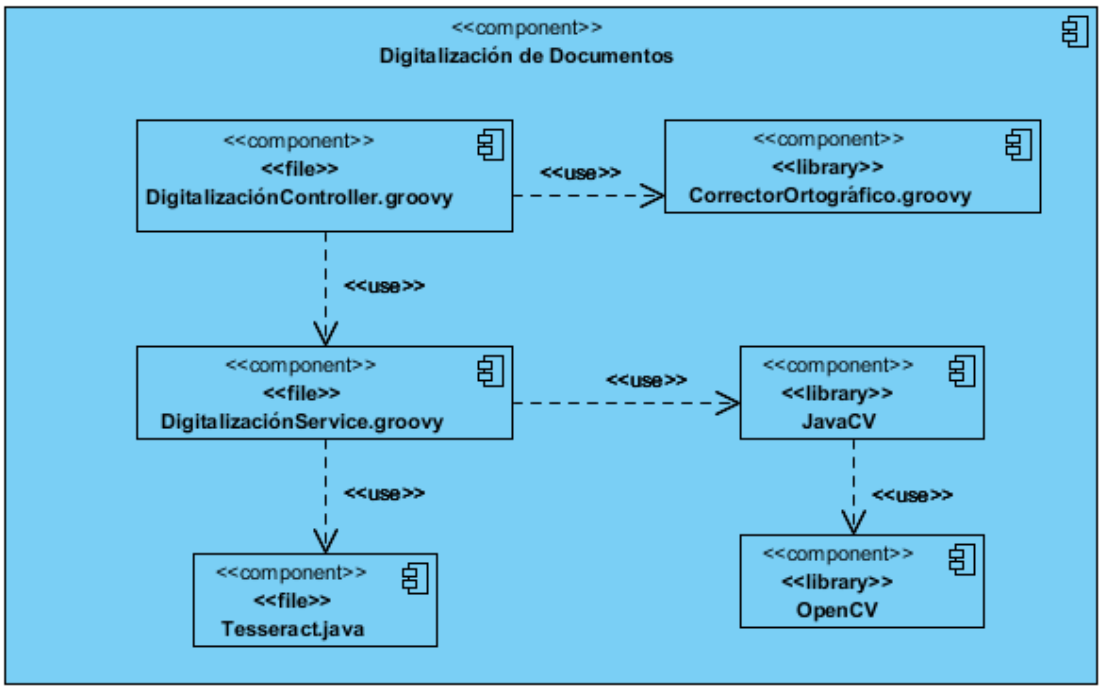

Figura 18 Diagrama de Componentes del módulo Digitalización de Documentos

4.2.2. Descripción de los Componentes

El componente Digitalización de Documentos, contiene como componente principal la clase controladora *DigitalizacionController.groovy*, y otro componente muy importante encargado de implementar la lógica del negocio es la clase *DigitalizacionService.groovy,* a continuación se explica el funcionamiento de cada uno de estos componentes.

4.2.2.1. La clase DigitalizacionController.groovy

Es el componente principal que se implementa en este módulo, gestiona la lógica del negocio redireccionando y realizando las llamadas a los servicios para dar respuesta al usuario.

Hace uso de los siguientes servicios:

nuxeoService*:* es el servicio que interactúa con el gestor documental Nuxeo gestionando los documentos.

digitalizacionService: interactúa con Nuxeo, Tesseract, Base de Datos, OpenCV facilitando su acceso.

descripcionManualService: se utiliza este servicio para obtener la ubicación de los documentos.

pdfService: es utilizado para generar los caracteres extraídos como documentos PDF.

Las principales funcionalidades que se implementan en esta clase:

buscarDocumento(): busca el documento por los criterios de búsqueda que introduce el usuario, que serán atributos que poseen el documento (elaborado por, productores, estado de revisión).

asociarArchivo(): obtiene las imágenes de un documento adicionadas por el usuario, las procesa utilizando el servicio *digitalizarImagen()*, guarda el resultado como un fichero de texto en Nuxeo y en la Base de Datos.

revisarDigitalizacionPorImagen()*:* tiene como parámetro una imagen, la cual es seleccionada por el usuario si desea revisar su digitalización. Esta funcionalidad se encarga de hacer una consulta al Nuxeo y obtener el fichero de texto asociado a esta imagen, lo muestra para que sea revisado.

guardarRevicion():utilizando el texto revisado como parámetro, esta funcionalidad permite guardar este texto con sus modificaciones realizadas.

exportPdfDoc(): con un documento que tiene como parámetro, obtiene sus textos asociados, genera un html y crea un documento pdf, con el objetivo que se muestren en el pdf los cambios realizados por el usuario, por ejemplo las palabras subrayadas o en otro formato.

arkheiaSecurityService: se utiliza para acceder a los datos del usuario autenticado.

4.2.2.2. La clase DigitalizacionService.groovy

La clase controladora es responsable de gestionar la lógica del negocio, sin embargo quien tiene la responsabilidad de implementarla es la clase *DigitalizacionService.groovy*, es en ella que se extraen los caracteres de las imágenes y se guardan en ficheros de texto que pueden ser gestionados por los usuarios, este lo puede eliminar, modificar y exportar.

Las principales funcionalidades que posee son las siguientes:

procesarImagen(): tiene como parámetro una lista de imágenes y la dirección de donde se encuentran, llamando a la clase Tesseract.java procesa cada una de estas imágenes y las guarda en la misma dirección con el mismo nombre la imagen, facilitando la búsqueda de estos ficheros.

obtenerTextoPorImagen(): con una imagen que tiene como parámetro y el documento al que pertenece busca el texto que tiene asociado en el Nuxeo.

eliminarDigitalizacion(): con una imagen que tiene como parámetro y la dirección donde se encuentra en Nuxeo, se encarga de eliminarla de este fichero.

exportarPdf(): se encarga de realizar la consulta para obtener el texto que se desea exportar.

Esta clase hace uso de los siguientes componentes:

Tesseract.java

El objetivo general de este trabajo se refiere a la integración del sistema Arkheia con el motor OCR Tesseract, el componente fundamental para integrar el Tesseract con la aplicación es clase Tesseract.java que funciona como una interfaz.

OpenCV:

Se utiliza esta librería para el tratamiento de las imágenes deterioradas. Posee algunas operaciones las cuales se utilizarán para mejorar la calidad de la imagen y facilitar al Tesseract un mejor resultado en el reconocimiento óptico de caracteres.

4.2.3. Tratamiento de errores

En términos de programación el tratamiento de errores permite al programador controlar los errores ocasionados durante la ejecución del programa. En el módulo Digitalización de Documentos se gestionan los errores a través de las excepciones, las cuales informan de que hacer cuando algo está fuera de lo normal, en cada método se capturó la excepción en una variable *flash* (figura 8), luego se muestran al usuario a través de la vistas.

```
try{...}
catch (Exception e){
    flash.error=e.message
```

Figura 19. Try Catch implementado en cada método

4.2.4. Diagrama de Despliegue

El despliegue de la solución para el módulo Digitalización de Documentos se realizará según como se muestra en el siguiente diagrama. Se representan los elementos del sistema en tiempo de ejecución por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software. Los elementos software podrán ejecutarse cumpliendo con las restricciones que aparecen en el diagrama.

Descripción de los dispositivos Hardware:

PC_Cliente Usuario Interno: Computadora que será empleada por el usuario autenticado en la institución con los permisos de digitalización.

Servidor Web: Se encarga de atender las peticiones de los usuarios para facilitar las respuestas.

Servidor de Base de Datos: Almacena toda la información que se gestiona en la institución.

Gestor Documental Nuxeo: Almacena los ficheros que pertenecen a los documentos.

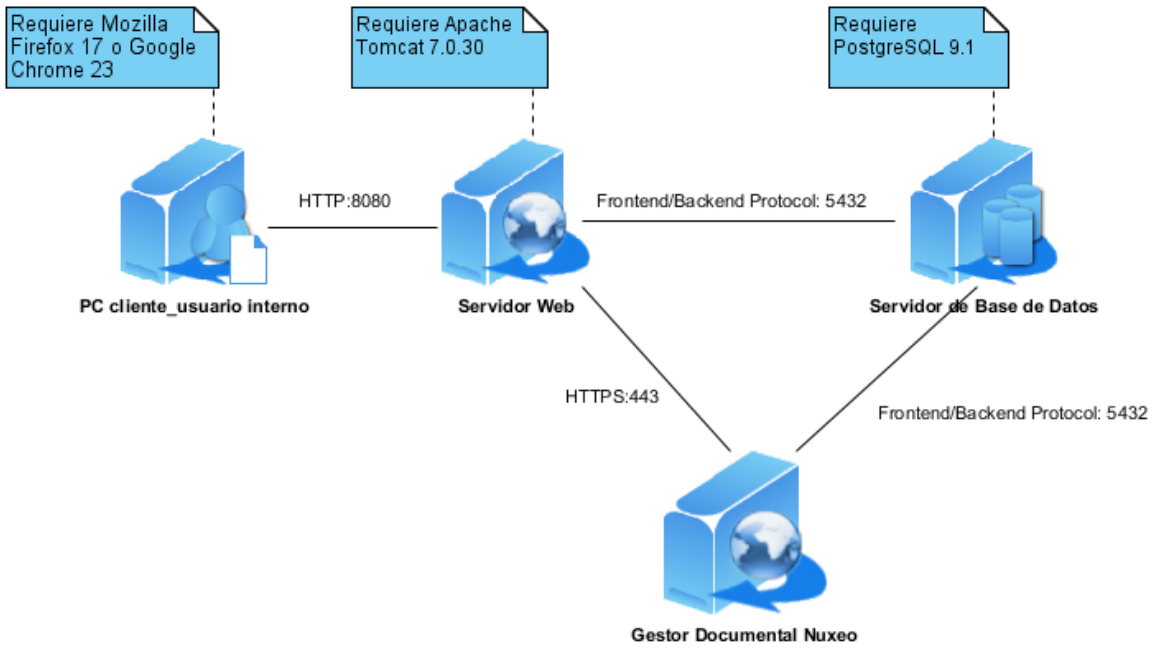

Figura 20. Diagrama de despliegue para el módulo Digitalización de Documentos.

4.3 Prueba

Las pruebas se centran principalmente en la evaluación o la valoración de la calidad del producto, verifican el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros. Esto reduce ampliamente el riesgo asociado con el despliegue de un software de baja calidad (Jacobson, 2000).

4.3.1. Método de Prueba: Pruebas de Caja Negra

La prueba de Caja Negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, se centra principalmente en los requisitos funcionales del Software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 1995). Se utilizan estas pruebas en el módulo para comprobar que la aplicación se comporta según los requerimientos establecidos por el cliente.

4.3.2. Técnica de Prueba: Diseño de caso de prueba

Los casos de pruebas son un conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados cuyo propósito es identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso) (Pressman, 1995).

El caso de prueba se diseña basado en las funcionalidades detalladas en los casos de uso. Este diseño se elabora antes de la realización de las pruebas funcionales de la aplicación. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, que hace más fructífera la ejecución de las pruebas.

Tabla 10. Escenarios a probar para el CU Adicionar Imagen.

Nombre de la sección	Nombre del escenario	Descripción	Respuesta del sistema
SC.1 Adicionar imagen	EC.1.1 Adicionar imagen	Se adicionan las imágenes a un documento.	El sistema se encargará de extraer los caracteres de las imágenes adicionadas por el usuario y guardarlos en un fichero de texto dentro del Gestor de Contenidos Empresariales (Nuxeo).
	EC. 1.2 Cancelar	Se cancela el proceso de adicionar la imagen.	Cancela la operación y regresa al flujo normal del caso de uso.

Tabla 11. Escenarios a probar para el CU Revisar digitalización.

Nombre de la sección	Nombre del escenario	Descripción	Respuesta del sistema
SC.1 Revisar digitalización	EC.1.1 Revisar Digitalización	El usuario puede revisar los caracteres extraídos, comprobar la ortografía y guardar los cambios realizados.	EL sistema permite realizar la corrección ortográfica y guarda en Nuxeo los cambios realizados.
	EC. 1.2 Cancelar	Se cancela todo el proceso de revisión.	Cancela la operación y regresa al flujo normal del caso de uso.
SC. 2 Corrección Ortográfica	EC. 2.1 Corrección Ortográfica	Son señaladas las palabras que el OCR no pudo reconocer correctamente para que el usuario pueda modificarlas.	Muestra un cuadro de diálogo con sugerencias de palabras a corregir, con las opciones de cambiarla por la sugerencia u omitirla, agregar al diccionario en caso que no se encuentre.
SC. Guardar revisión	EC. 2.2 Guardar revisión	Se guardan los cambios realizados	El sistema guarda en Nuxeo los cambios realizados.

4.3.3. Resultados

De los casos de uso probados se detectaron las siguientes No Conformidades, las cuales fueron resueltas por el equipo de desarrollo.

Tabla 12. Resultado de las pruebas.

No	No Conformidad	Ubicación	Estado
1	No se muestra un mensaje de error cuando el usuario adiciona un fichero en otro formato que no sea .jpg o .png	EC.1 Caso de uso: Adicionar Imagen	Resuelta
2.	El sistema no informa al usuario cuando ha terminado de extraer los caracteres de las imágenes adicionadas.	EC.1 Caso de uso: Extraer caracteres de imágenes	Resuelta
3	Cuando se realiza la corrección ortográfica el sistema no permite adicionar nuevas palabras al diccionario.	EC. 2.1 Caso de uso: revisar Digitalización.	Resuelta

4.3.4. Tipo de prueba: Prueba de Usabilidad

Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente y agentes, documentación de usuarios y materiales de formación.

Las pruebas de usabilidad del sistema se realizaron a través de las listas de chequeo, realizadas por el equipo de trabajo del Centro Nacional de Calidad del Software (Calisoft). Las listas de chequeo permiten recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados, estos elementos son:

- Visibilidad del sistema
- Adecuación del sistema
- Control y libertad de usuarios
- Consistencia y estándares
- Prevención de errores
- Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores
- Ayuda y documentación

- Accesibilidad
- Comprobaciones técnicas
- Semántica de la aplicación

En la realización de esta prueba se detectaron 10 no conformidades, las mismas fueron resueltas por el equipo de desarrollo.

4.4 Efectividad del motor OCR

Cuando se alcanzan los resultados y objetivos propuestos no se conoce con exactitud si se ha realizado un trabajo con toda la calidad atendiendo a los requerimientos de clientes y usuarios.

Se entiende por **efectividad** el logro de los resultados propuestos en forma eficaz. Por tal razón es necesario lograr que el proceso por el que pasan las imágenes con en el módulo Digitalización de Documentos cumpla con los objetivos propuestos, logrando optimizar los recursos con los que se cuenta. Para comprobarlo se tomarán dos imágenes, tomadas de un libro en formato duro con diferentes características en cuanto a la calidad, se procesarán y detallará el resultado con el fin de comprobar cómo se comporta la efectividad del trabajo.

4.3.4.1. Resultado del procesamiento con Tesseract sin preprocesar la imagen de entrada.

4.3.4.2. Tabla 13. Resultado del procesamiento con Tesseract

Imagen	Total de palabras en el documento	Palabras incorrectas	% de efectividad
Imagen 1	192	35	82%
Imagen 2	254	12	96%

Luego del procesamiento de la imagen con Tesseract sin verificar la calidad se observa que en la *imagen 2* el Tesseract arrojó muy buenos resultados con una efectividad de 96 %, de las palabras mal reconocidas solo algunos caracteres eran incorrectos, demostrando que con un fondo blanco los caracteres son más legibles y se obtienen mejores resultados. Sin embargo para la *imagen 1* los resultados no son muy buenos debido a que sus caracteres no son muy legibles por el deterioro que posee.

4.3.4.3. Resultado del procesamiento con Tesseract y preprocesamiento con OpenCV.

Para lograr un mejor resultado con la *imagen 1* es necesario realizar el preprocesamiento con OpenCV, así los caracteres serán más fáciles de reconocer para el motor OCR y aumentará el por ciento de efectividad, los resultados son:

Tabla 14. Resultado del preprocesamiento con OpenCV de la *imagen 1*

Imagen	Total de palabras en el documento	Palabras incorrectas	% de efectividad
Imagen 1	192	20	92%

4.5 Conclusiones Parciales

En este capítulo se representaron y explicaron los artefactos generados en la implementación del módulo, el diagrama de despliegue y el diagrama de componentes. Se definió y se implementó una política de tratamiento de errores que respalda el funcionamiento de los componentes ante el suceso de los mismos. Haciendo uso de los casos de pruebas se realizaron las pruebas de caja negra verificando que cumple con los requisitos especificados, las no conformidades encontradas fueron resueltas por el equipo de desarrollo. La prueba de efectividad del OCR demostró que es más efectivo en imágenes de buena calidad con resultados satisfactorios (96%), no siendo así para imágenes de baja calidad, problema que se soluciona con el preprocesamiento de la imagen, facilitando así la extracción de los caracteres al Tesseract y el aumento de efectividad en 8%.

Conclusiones Generales

El resultado de esta investigación es el módulo Digitalización de Documentos, responsable de la digitalización de los documentos de los archivos a través del reconocimiento óptico de caracteres, dando solución a los problemas a los que se enfrentan las instituciones de los archivos en la gestión de sus documentos. Se puede afirmar que se dio cumplimiento a los objetivos propuestos:

- Se realizó el estudio de sistemas encargados del reconocimiento óptico de caracteres, seleccionando Tesseract como parte de la solución del módulo Digitalización de Documentos por ser el que más se ajusta a las necesidades del sistema Arkheia.
- Se demostró que las aplicaciones existentes para la Digitalización de Documentos que usan el motor OCR Tesseract no brindan una solución óptima para el problema planteado en las instituciones de los archivos, evidenciando la necesidad de una nueva propuesta de solución y teniendo en cuenta algunas de las funcionalidades de estas aplicaciones.
- Las actividades de análisis y diseño del módulo Digitalización de Documentos fueron ejemplificadas a través de diagramas de análisis, clases, e interacción, cumpliendo así con los objetivos propuestos.
- Se implementó el módulo Digitalización de Documentos del sistema Arkheia a partir de los requerimientos de software establecidos con el cliente y haciendo uso de la arquitectura y las tecnologías definidas por el proyecto.
- Se diseñaron y aplicaron pruebas de caja negra, probando las funcionalidades del módulo. Las 3 inconformidades encontradas fueron resueltas por el equipo de desarrollo.
- Se demostró que la efectividad del módulo depende de la calidad de la imagen que se desea procesar, lo que conllevó a realizar el preprocesamiento a imágenes de baja calidad, y se logró aumentar en gran medida la efectividad del procesamiento.

Referencias Bibliográficas

Alvarez, Miguel Angel. 2001. DesasarrolloWeb.com. [En línea] 2001. [Citado el: 14 de 1 de 2013.] http://www.desarrolloweb.com/articulos/25.php.

Apache, Software Funadation. 2013. Apache Tomcat. *Apache Tomcat.* [En línea] Apache Software Foundation, 2013. [Citado el: 2013 de 4 de 1.] tomcat.apache.org.

Bedwyr, Canolfan. 2008. *An overview of the Tesseract OCR (optical character recognition) engine,.* s.l. : Language Technologies Unit, Bangor University, 2008.

Bradski, Gary. 2008. *Learning OpenCV, Computer Vision with the openCV Library.* 2008.

Brito, Nacho. 2009. *Manual de desarrollo web con Grails,JavaEE como siempre debió haber sido.* 2009. ISBN-978-84-613-2651.

—. 2009. *Manual de desarrollo web con Grails. JavaEE, como siempre debió haber sido.* 2009. ISBN: 978-84-613-2651.

Cam, Celso Gonzáles. 2007. *La Importancia de la Digitalización de Archivos para la.* Perú : s.n., 2007.

Diaz, Antonio. 2011. Ocrad - The GNU OCR. *Ocrad - The GNU OCR.* [En línea] Free Software Foundation, 2011. [Citado el: 8 de 5 de 2013.] http://www.gnu.org/software/ocrad/.

España, Yerbabuena Software. 2007. Athento. [En línea] 2007. [Citado el: 11 de 6 de 2013.] http://www.athento.com.

FreeOCR. 2011. FreeOCR.net. *FreeOCR.net.* [En línea] febrero de 2011. [Citado el: 10 de 12 de 2012.] http://www.freeocr.net/.

Glen Smith, Peter Ledbrook. 2009. *Grails in Action.* United States of America : Manning Publications Co., 2009. ISBN 978-1-933988-93-1.

Groovy. 2013. Groovy. [En línea] 2013. http://groovy.codehaus.org/.

Herrero, Enrique Pérez. 1997. *EL ARCHIVO Y EL ARCHIVERO.* Islas Canarias : s.n., 1997.

Jacobson, Ivar. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.

Jalón, Javier García de. 2000. *Aprenda Java como si estuviera en primero.* San Sevastian : UNIVERSIDAD DE NAVARRA, 2000.

Java. 2013. Java. [En línea] 2013. [Citado el: 14 de 1 de 2013.] http://www.java.com.

Laguna, Miguel A. 2013. Ingeniería del Software I. *Requisitos.* [En línea] 2013. [Citado el: 26 de 2 de 2013.] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.

Lapuente, María Jesús Lamarca. 2010. Hipertexto: El nuevo concepto de documento en la cultura de la imagen. *Metadatos.* [En línea] Tesis doctoral. Universidad Complutense de Madrid, 5 de 12 de 2010. [Citado el: 6 de 12 de 2012.] http://www.hipertexto.info/documentos/metadatos.htm#OMendez.

Larman, Craig. 1999. *UML y Patrones, Introduccion al analisis y diseño orientado a objetos.* Mexico : s.n., 1999.

Letelier, Patricio. 2003. *Proyecto Docente e Investigador.* s.l. : DSIC, 2003.

Map, Elvia Leticia Williams. 2012. *Propuesta para la Digitalización del Fondo Documental del Dr.Belisario Porras .* Andalucia : Universidad Internacional de Andalucia, 2012.

Martinto, Pedro Carlos Pérez. 2013. *El diseño metodológico de la investigación.* Habana, Cuba : Univercidad de Ciencias Informáticas, 2013.

Medin, Claudio. 2012. adcvBA. [En línea] 2012. [Citado el: 1 de 4 de 2013.] http://www.adcv.org.ar/prog-dgsl/entry/un-ocr-potente-y-gratuito-con-respaldo-de-google.

Mugica, Mayra Marta Mena. 2006. *PROPUESTA DE REQUISITOS FUNCIONALES PARA LA GESTIÓN DE DOCUMENTOS ARCHIVÍSTICOS ELECTRÓNICOS EN LA ADMINISTRACIÓN CENTRAL DEL ESTADO CUBANO.* Ciudad de La Habana : s.n., 2006.

Navarro, Aída Luz Mendoza. 2002. *Lrgislacicíir A rchivística: Lo que el Archivero debe conocer.* Lima-Perú : PERÚ TEXTOS SAC, 2002. 426 0944.

Neyem, Andrés. 2013. *INSTRUCTIVO PARA EL MODELADO CON CASOS DE USO.* 2013.

Nuxeo. 2011. Nuxeo. [En línea] 2011. [Citado el: 2013 de 2 de 6.] http://www.nuxeo.com.

Paradigm, Visual. 2013. BPMN and Database Tool for Software Development. *UML.* [En línea] 2013. [Citado el: 12 de 1 de 2013.] http://www.visual-paradigm.com.

Paris, José Vergara. 2002. *Conservación y restauración de material.* 2002. 84-482-3077-9 .

PostgreSQL, Sobre. 2010. PostgreSQL-es. [En línea] 2010. [Citado el: 15 de 1 de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Pressman, Roger S. 1995. *Ingenieria del Software, un enfoque práctico.* [ed.] Darrel Ince. Quinta Edicion. 1995.

Risco, Enrique Garrido-Lecca. 1998. Qué es JAVA Script ? *Qué es JAVA Script ?* [En línea] 1998. [Citado el: 14 de 1 de 2013.] http://www.pablin.com.ar/computer/cursos/cursojs/js1.htm.

Rocha, Joaquim. 2011. Gnome.org. *Gnome.org.* [En línea] 22 de 7 de 2011. [Citado el: 1 de 10 de 2013.] https://live.gnome.org/OCRFeeder/.

Rodríguez, Txema. 2012. GENBETA desarrollo y software. *Bootstrap.* [En línea] 2012. http://www.genbetadev.com/frameworks/bootstrap.

Schulenburg, Joerg. 2010. GOCR. *GOCR.* [En línea] 2010. [Citado el: 9 de 5 de 2013.] http://www.gocr.de/.

—. 2002. GOCR-documentation. [En línea] 2002. http://fossies.org/linux/privat/gocr-0.50.tar.gz:a/gocr-0.50/README.

Sommerville, Ian. 2005. *Ingeniería del Software.* Madrid : Pearson Education, 2005. 84-7829-074-5.

Springsource. 2013. Springsource. [En línea] 2013. [Citado el: 14 de 1 de 2013.] http://www.springsource.org/sts.

SpringSource. 2011. SpringSource. [En línea] 2011. [Citado el: 14 de 1 de 2013.] http://www.springsource.com/developer/sts..

Stefan Küng, Lübbe Onken, y Simon Large. 2011. *TortoiseSVN: Un cliente de Subversion para Windows.* 2011.

Sun. 2000. *El Lenguaje de Programacion Java.* 2000.

Talledo, María Fernanda Frydson. 2011. *Aplicación de visión por computador para el reconocimiento del número de placa de vehículos usando modelos de aprendizaje OCR.* ECUADOR : ESCUELA SUPERIOR POLITECNICA DEL LITORAL, 2011.

TextRipper. 2007. openDesktop.org. [En línea] 2007. [Citado el: 10 de 1 de 2013.] http://gtk-apps.org/content/show.php/TextRipper+%28aka+T-Rip%29?content=132759.

YAGF. 2013. YAGF - graphical front-end for cuneiform and tesseract. [En línea] 2013. [Citado el: 10 de 1 de 2013.] http://symmetrica.net/cuneiform-linux/yagf-en.html.