**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**FACULTAD 3**

**CEGEL**

# Sistema para la digitalización de los expedientes en la Secretaría General de la Universidad de las Ciencias Informáticas

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

**Autor(es):**

**Haraid Martínez Cano**

**Alex Alain Hurtado Campuzano**

**Tutores:**

**Ing. Angel Miguel Morciego Lezcano**

**MsC. Mailen Edith Escobar Pompa**

**La Habana, mayo de 2019**

**"Año 61 de la Revolución"**

## DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____
Haraid Martínez Cano
Firma del Autor

_____
Alex Alain Hurtado Campuzano
Firma del Autor

_____
Ing. Angel Miguel Morciego Lezcano
Firma del Tutor

_____
MsC. Mailen Edith Escobar Pompa
Firma del Tutor

## DATOS DE CONTACTO

Tutor: Ing. Angel Miguel Morciego Lezcano
Correo: amorciego@uci.cu

Tutor: MsC. Mailen Edith Escobar Pompa
Correo: meescobar@uci.cu

Autor: Haraid Martínez Cano
Correo: hmartínez@estudiantes.uci.cu

Autor: Alex Alain Hurtado Campuzano
Correo: aahurtado@estudiantes.uci.cu

## DEDICATORIA

*Primeramente, le dedico esta tesis a nuestro Comandante en Jefe Fidel Castro Ruz, por haberme permitido realizar mis estudios en esta única Universidad, creándola en 12 de diciembre del 2002.*

*También se la dedico a mis padres, abuelos y familiares en general, para que cada vez se sientan más orgullosos.*

***Alex Alain***

*A mi mamá por haberme apoyado siempre en todas mis decisiones.*

*A mi papá por ser ese pilar que me dio el soporte para continuar.*

*A mis abuelas por darme su fuerza y voluntad para seguir adelante.*

*A mi hermano Gesiel por ser como un padre para mí.*

***Haraid***

## RESUMEN

Los archivos se utilizan para gestionar, atesorar, conservar y difundir el patrimonio documental. Con la digitalización de los mismos se facilita el manejo de la información, así como su preservación, almacenamiento y transportación. En la Secretaría General de la Universidad de las Ciencias Informáticas se archivan en los expedientes la trayectoria estudiantil de cada uno de sus graduados. Estos expedientes se encuentran expuestos al deterioro condicionado por el tiempo de vida de los documentos en formato duro, además de que existe pérdida de información, duplicidad de la información y la misma no se guarda en el formato establecido. Debido a estos problemas se define como objetivo de la presente investigación desarrollar un sistema que agilice el proceso de la digitalización de los expedientes en la secretaría general de la universidad garantizando la calidad de esta información.

Para ello se selecciona la metodología, herramientas y tecnologías para el desarrollo de la solución; se describen las principales funcionalidades y se hace un análisis de la arquitectura y los patrones a utilizar. Como resultado de la investigación se obtuvo un sistema que realiza la digitalización de los expedientes en la secretaría general en correspondencia con las solicitudes del cliente, validado por las actas de liberación y aceptación emitidas. Este sistema permite que el proceso de gestión de estos expedientes se realice de forma ágil garantizándose la calidad de la información que se genera en la Secretaría General.



Palabras Clave: Archivo, Digitalización, Expediente, Secretaría.

***ABSTRACT***

*The archives are used to manage, treasure, preserve and disseminate the documentary heritage. With the digitalization of the same the handling of the information is facilitated, as well as its preservation, storage and transportation. In the General Secretariat of the University of the Informatic Sciences, the graduates student trajectory is archived in the files. These files are exposed to deterioration conditioned by the life time of the documents in hard format, in addition to the loss of information, duplicity of information and it is not saved in the established format. Due to these problems it is defined as the objective of the present investigation develop a system that streamlines the process of digitalization of the files in the general secretariat of the university, guaranteeing the quality of this information.*

*For this, the methodology, tools and technologies for the solution development are selected; the main functionalities are described and an analysis of the architecture and the patterns to be used is made. As a result of the investigation, a system was obtained that carries out the digitalization of the files in the general secretariat in correspondence with the client's requests, validated by the release and acceptance certificates issued. This system allows the process of managing these files to be carried out in an agile manner, guaranteeing the quality of the information generated in the General Secretariat.*

Keywords: Archive, Digitization, File, Secretariat.

# ÍNDICE

# ÍNDICE

## ÍNDICE DE TABLAS

## ÍNDICE DE ILUSTRACIONES

## INTRODUCCIÓN

Los archivos se utilizan para gestionar, atesorar, conservar y difundir el patrimonio documental. Estos pueden almacenar documentos históricos recibidos por donación, depósito, transferencia y adquisición. Con la digitalización de los mismos se facilita el manejo de la información, así como su preservación, almacenamiento y transportación. Mediante la digitalización se reduce la manipulación de los documentos originales y permite conservarlos para retrasar su deterioro. Además, disminuye el espacio físico y el riesgo de robo o destrucción de los documentos (Mujica y Herrera 2005a).

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

Un documento digital puede ser el resultado de haber procesado con un escáner un documento originalmente impreso. Primeramente, se obtiene una imagen (fotografía digital) del documento escaneado. La imagen que se obtiene puede ser guardada en un medio electrónico, pero no tiene las capacidades de hipertexto de un documento de texto digital. Posteriormente esta imagen puede ser procesada por un sistema de reconocimiento óptico de caracteres para así convertirla en un documento digital con todas las capacidades de hipertexto (Mujica y Herrera 2005a).

La Universidad de las Ciencias Informáticas (UCI) genera y recibe una considerable cantidad de documentación como consecuencia de las actividades necesarias para el cumplimiento de sus procesos fundamentales: docencia, producción e investigación. La documentación que se genera en estas actividades constituye una evidencia de su realización que además puede tener valor fiscal, legal y en su última fase de vida, valor histórico ya que constituye la memoria de la institución.

La secretaría se encarga de proponer e instrumentar las políticas, normas, sistemas y procedimientos necesarios para salvaguardar los bienes institucionales, verificar la exactitud y seguridad de los datos. También es la que posibilita desarrollar la eficiencia del control de gestión, así como supervisar la correcta administración de los recursos financieros, humanos y materiales de la entidad, recibir y archivar documentos y estar al pendiente de la tramitación de expedientes («Secretaría Administrativa» 2018). En la secretaría de la universidad se preserva gran volumen de expedientes docentes de estudiantes y graduados. A este lugar acuden frecuentemente graduados, profesores y especialistas

solicitando acceso a estos documentos para consultar información de interés con el objetivo de realizar procesos docentes, investigativos y laborales.

El expediente docente lleva un conjunto de documentos que registra todo el quehacer del estudiante en su estancia en la institución, algunos de los cuales son (Grupo de identificación y valoración de series de los archivos universitarios 2001):

- Los datos de identificación del centro.
- Los datos del alumno o alumna.
- La fecha de apertura y el número de expediente.
- La información relativa al proceso de evaluación del alumno: los resultados de la evaluación con las calificaciones obtenidas, las decisiones de promoción de etapa, las medidas de apoyo educativo, las adaptaciones curriculares que se hayan adoptado para el alumno o alumna y, en su caso, el seguimiento del proceso de enseñanza bilingüe.
- Las fechas de entrega de las certificaciones académicas, o cualquier otro documento que se considere necesario incluir.

Una vez que el estudiante termina la carrera para guardar la información del expediente se hace necesario digitalizar esta información. En la secretaría actualmente este proceso se realiza con el uso de un escáner y una computadora, sin embargo, se presentan problemas que atentan contra el buen desempeño de este proceso:

- Se trabaja con un archivo independiente por cada página que tenga el documento original, lo que dificulta la organización de los documentos que presentan más de una página.
- Existe pérdida de la información. Muchas veces se digitalizan los documentos y no se guardan en los lugares establecidos, perdiéndose en muchos casos el trabajo realizado, por lo que hay que volverlo a hacer.
- Existe duplicidad de la información. Como parte de la no organización en algunos casos, se encuentran digitalizados los mismos documentos dentro de las computadoras en varios lugares y repetidos varias veces.
- Los expedientes deben guardarse por una estructura específica definida por el Ministerio de Educación Superior, la cual igualmente debe respetarse en los expedientes digitales. Este proceso se realiza en la secretaría de forma manual, lo que conlleva un esfuerzo de parte de los trabajadores para lograr cumplir con este requisito.
- Todo lo anteriormente expuesto conlleva a tener en la secretaría personal dedicado a largas jornadas laborales para organizar toda la documentación que se genera como parte del proceso.

En el 2016 en la secretaría general de la UCI se hizo uso de una aplicación de escritorio llamada Archivo UCI. Esta herramienta, por las pocas facilidades que mostraba para su uso, solo fue utilizada para digitalizar documentación asociada a libros y documentos que no pertenecían a los expedientes, siendo considerada por la secretaría como no adecuada para ser usada en un proceso tan delicado como la gestión y digitalización de los expedientes docentes.

La problemática antes mencionada conduce al siguiente **problema** de investigación: ¿Cómo agilizar el proceso de digitalización de los expedientes en la Secretaría General de la Universidad garantizándose la calidad de la información generada en esta área?

Definiéndose como **objeto de estudio** los procesos de digitalización y gestión de documentos digitales.

Para dar solución al problema expuesto anteriormente se plantea como **objetivo general:** Desarrollar un sistema que agilice el proceso de la digitalización de los expedientes en la secretaría general de la UCI garantizándose la calidad de la información generada en esta área.

Desglosándose en los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante el estudio de los sistemas para la digitalización de archivos.
- Desarrollar el sistema para la digitalización de los expedientes en la secretaría general de la universidad garantizándose la calidad de la información generada en esta área.
- Verificar el correcto funcionamiento de la herramienta y que la solución propuesta dé cumplimiento al objetivo general de la investigación.

Enmarcándose como **campo de acción:** los procesos de digitalización y gestión de los expedientes en la secretaría general de la UCI.

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

- Revisión bibliográfica acerca de las soluciones existentes.
- Elaboración del marco teórico del tema de investigación.
- Estudio de las herramientas para el desarrollo de aplicaciones de escritorio (Desktop).
- Análisis para definir las herramientas, métodos y metodologías de desarrollo de software para la realización del sistema.
- Identificación de los requisitos funcionales a tener en cuenta para el desarrollo del sistema.
- Identificación de los requisitos no funcionales a tener en cuenta para el desarrollo del sistema.
- Realización de los diagramas de clases.
- Realización del modelo de diseño.

- Implementación del sistema de digitalización de expedientes.
- Aplicación de las pruebas de software al sistema de digitalización de expedientes.

Los **Métodos Teóricos** utilizados para cumplir las tareas son los siguientes:

- **Analítico-Sintético:** Este método fue utilizado para el análisis de los principales conceptos asociados al objeto de estudio y las herramientas homólogas.
- **Modelación:** Se utilizó la modelación para representar por medio de diagramas los conceptos asociados a la etapa de diseño del sistema y de esta manera hacerlos más comprensibles a la hora del desarrollo.

Los **métodos empíricos** utilizados fueron:

- **Entrevista:** Se realizó una entrevista a las personas encargadas de realizar la digitalización de los expedientes en la Secretaría General, con el objetivo de obtener información acerca de este proceso y de los principales problemas a los que se enfrenta para su realización. Se materializó la entrevista informal, individual y no estructurada.

A continuación, se explicará la estructura capitular del documento donde se expondrá por cada capítulo el proceso a seguir para el desarrollo de la investigación.

**Capítulo 1**. “Fundamentación teórica”: En este capítulo se abordan elementos importantes sobre las aplicaciones para la digitalización de documentos, se presentan las definiciones de digitalización de documentos, así como algunos elementos directamente relacionados con esta. Se realiza un estudio crítico sobre las herramientas existentes destinadas a la digitalización de documentos a nivel nacional e internacional. Además, se describe la metodología, las tecnologías, las herramientas y el lenguaje a utilizar para la construcción del sistema.

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

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

# CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

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

## 1.2 Conceptos asociados

A continuación, se conformará el sustento teórico de la presente investigación; se profundizará en los conceptos fundamentales asociados a la digitalización de expedientes para una mayor comprensión de la investigación realizada.

### 1.2.1 Gestión documental

La gestión documental se extiende al ciclo de vida completo de los documentos desde su producción hasta la eliminación final y su envío al archivo para su conservación permanente. Está dirigido a asegurar una documentación adecuada, evitar lo no esencial, simplificar los sistemas de creación y uso del papeleo. Además, mejora la forma de cómo se organizan y se recuperan los documentos, proporciona el cuidado adecuado y almacenamiento de los documentos en los centros de archivo y asegurar la ordenación adecuada de los documentos que no se necesitan por mucho tiempo en la conducción de los asuntos del momento(Mujica y Herrera 2005a).

Un sistema de gestión documental (DMS por sus siglas en inglés), está diseñado para almacenar, administrar y controlar el flujo de documentos dentro de una organización. Se trata de una forma de organizar los documentos e imágenes digitales en una localización centralizada a la que los empleados puedan acceder de forma fácil y sencilla (Lluis Codina 1993).

### 1.2.2 Digitalización de documentos

La digitalización de documentos se define como la captura de una imagen física, mediante escáner o cámara digital, que una vez convertida en imagen electrónica puede ser almacenada y procesada por una computadora. Sus resultados están determinados por la resolución (densidad de puntos, o píxeles que tiene una imagen) y por la distribución luminosa en el documento, ya sea en sus niveles de grises o tonalidades de color (Elda González Mesa 2006a).

Por consiguiente, se entiende que la digitalización de documentos es el proceso de convertir un documento a una imagen que pueda ser reconocida por un computador.

### 1.2.3 Expediente

Un expediente es el conjunto de los documentos que corresponden a una determinada cuestión. También puede tratarse de la serie de procedimientos de carácter judicial o administrativo que lleva un cierto orden. Los expedientes se almacenan y archivan en lugares especiales y destinados a tal efecto y obviamente disponen de elementos, tales como números que facilitan su reconocimiento y búsqueda en un archivo (Julián Pérez Porto y María Merino 2012).

Por tanto, un expediente académico es un expediente formado por los documentos que origina la estancia en la universidad de los alumnos, desde el trámite administrativo que debe efectuar el alumno al realizar la matrícula más las incidencias posteriores hasta la obtención del título (Grupo de identificación y valoración de series de los archivos universitarios 2001).

## 1.3 Análisis de sistemas homólogos

En el mundo actualmente existen diferentes sistemas o herramientas, tanto de software libre[1] como propietarias[2]. Estas herramientas permiten dar solución a diversos problemas relacionados con la digitalización de documentos y gestión documental, reduciendo a su vez los asociados a la organización de la información dentro de la institución.
A continuación, se expone algunos de estos sistemas:

### 1.3.1 Internacionales

#### 1.3.1.1 Scan2PDF

La herramienta permite escanear documentos y guardarlos en formato PDF en el ordenador. También permite abrir imágenes que se tengan en el disco duro, y combinarlas con un documento recién escaneado, o reunir varios documentos o fotos en un solo PDF («Freeware OCR Scanner Software Scan2PDF (Scan To PDF)» 2018). Es fácil de usar y la interfaz cuenta con un diseño sencillo. Presenta las siguientes características:

- Carga imágenes desde el disco duro.
- Escanea imágenes.
- Cambia el orden de las imágenes.
- Borra imágenes de la lista.

[1] «Software libre» es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el «software libre» es una cuestión de libertad, no de precio. (Luis Miguel Arteaga 2018)

[2] Se le llama software propietario, no libre, privado o privativo al tipo de programas informáticas o aplicaciones en el que el usuario no puede acceder al código fuente o tiene un acceso restringido y, por tanto, se ve limitado en sus posibilidades de uso, modificación y redistribución. Este tipo de software se opone al más recientemente popularizado software libre, que permite que cualquiera lo modifique y lo redistribuya.(«Definición de Software propietario » Concepto en Definición ABC» 2018)

- Rota imágenes.

#### 1.3.1.2 Nuxeo

Nuxeo es un software que permite implementar con gran funcionalidad un repositorio documental corporativo**.** Su implementación es sencilla si lo que se quieren cubrir son necesidades no muy específicas («Productos de Nuxeo | Nuxeo» 2018). Esta solución te ofrece las siguientes características:

- Colaboración, flujos de trabajo, búsquedas eficientes.
- Flexibilidad, empleando una arquitectura basada en estándares.
- Robustez, utilizando Java para entornos Enterprise (J2EE) entre otras tecnologías.
- Velocidad, gracias a la integración con el rápido motor de búsqueda Lucene.
- Capacidad de evolución en captura inteligente de documentos mediante Athento.
- Seguridad gracias a la implementación de estándares como SSL, Single Sign On (SSO) y facilitando el cumplimiento con normativas como la ISO 27.001 (Seguridad de la Información) o la Ley de Protección de Datos (LOPD).

#### 1.3.1.3 Alfresco

Es una solución versátil compatible con software tanto de la vertiente Microsoft, como de la rama Linux. Posibilita la creación y gestión de contenidos empresariales desde una gran cantidad de blogs y paquetes ofimáticos (Office y OpenOffice). Además, ofrece una gran variedad de herramientas colaborativas como calendarios individuales y de equipo, comentarios de actividades y tableros de discusión. Alfresco es para las empresas ante todo colaboración, pero también constituye un gestor documental. Su base de programación, junto a Nuxeo, es Java, lo que los convierte en soluciones multiplataforma adaptables a cualquier entorno («Alfresco» 2018).

Está disponible en dos versiones: Alfresco Community y Alfresco Enterprise. Ambas versiones son Open Source, aunque su versión Enterprise, diseñada para importantes volúmenes de trabajo, es de pago. Se trata de una plataforma ECM líder en su mercado con más de 2.500 implantaciones y probada en entornos con más de 100 millones de documentos.

Entre sus principales características se encuentran:

- Facilidad de uso.
- Entorno web.
- Soporte para Gestión de Contenidos Empresariales (incluidas Gestión Documental y Gestión de Activos Digitales).
- Soporte para Gestión de Contenido Web (WCM).

- Soporte para Records Management.
- Soporte para Gestión de Contenido Social (Colaboración).
- Fácil despliegue y administración.
- Escalabilidad.

#### 1.3.1.4 Athento – Software de Gestión Documental Inteligente

Es una aplicación informática que permite el manejo, gestión, conservación, publicación y trabajo sobre documentos electrónicos. Te permite ahorrar tiempo y esfuerzo en la gestión de documentos («Software de Gestión Documental - Athento» 2018). Athento es una suite compuesta por un conjunto de módulos que permiten cubrir el ciclo de vida de los documentos desde su captura hasta su archivo:

- Athento ECM: Gestor documental
- Athento Smart Engine (SE): Captura de documentos
- Athento Rhombus: Herramienta visual de parametrización del gestor documental.

Todos los componentes de la suite de Athento están cobijados por la misma licencia de uso. Es decir, que el cliente contará con el uso de todos los módulos bajo una única suscripción.

Athento –*Smart Document Management*– permite a las empresas automatizar procesos relacionados con la captura, gestión, almacenamiento y distribución de documentos. A diferencia de otros sistemas, Athento permite a las compañías contar con un proceso de Enterprise Content Management integrado en sus diferentes fases. («Software de Gestión Documental - Athento» 2018).

Athento ha sido desarrollado mediante módulos. Sus dos módulos fundamentales son el módulo de Captura y el módulo ECM. Estos dos módulos, integrados de forma nativa, pero independientes, cubren las fases de Captura, Almacenamiento, Gestión y parte de la Distribución («Software de Gestión Documental - Athento» 2018).

#### 1.3.1.5 Kofax Express

Facilita a todo tipo de usuarios la digitalización, organización y almacenamiento de documentos, a velocidades que aceleran drásticamente el procesamiento de lotes, tanto grandes como pequeños. Express es suficientemente fácil para los principiantes y suficientemente potente para los expertos («Kofax Express | Kofax» 2018).

#### 1.3.1.6 SE Capture

Es un software que simplifica el tratamiento de aspectos como la captura, procesamiento, validación y gestión de contenido. Además, notifica automáticamente a los equipos y departamentos sobre nuevos contenidos. A través de los recursos de procesamiento automático de imágenes, reconocimiento, clasificación, extracción de datos e indexación de documentos, las organizaciones pueden agregar valor y enriquecer el contenido, acelerando los procesos de captura, evitando errores y elevando los

niveles de productividad («Software para Captura e Escaneo de Documentos | SoftExpert Captura» 2018).

- Soporta escáner, cámaras digitales y otros dispositivos compatibles con el estándar TWAIN.
- Clasifica documentos a través de categorías jerárquicas.
- Acelera los procesos de captura y reduce intervenciones manuales a través de la programación de los procesos.
- Mejora la productividad a través de la monitorización de carpetas o cuentas de e-mail, e importación automática del contenido.
- Permite establecer procesos de captura con etapas como control de calidad, separación de documentos, reconocimiento de caracteres, verificación, indexación, entre otros, de acuerdo con el tipo de documento.
- Detecta y elimina automáticamente las páginas en blanco, para optimizar el espacio de almacenamiento.
- Permite priorizar actividades de captura, a través del acompañamiento de las actividades en día y en atraso.
- Captura un único tipo de documento, o un gran volumen de documentos diferentes en procesos complejos y multidepartamentales.
- Almacena archivos en banco de datos o en directorios controlados, aplicando mecanismos de compresión y protección por contraseña.
- Convierte las imágenes capturadas en imágenes (TIFF, JPEG, FIG) o PDF rastreable.
- Ofrece visualizador nativo embarcado en la solución para la visualización de documentos PDF, TIFF, JPEG, GIF, entre otros.
- Aplica controles de seguridad por usuario, equipo, categoría, documentos, entre otros.
- Localiza documentos a través de la aplicación de filtros por usuario, departamento, metadato, entre otros.
- Presenta informes de productividad para el acompañamiento de identificación de posibles cuellos de botella en los procesos de captura.
- Permite exportar informes para Excel para manipulación de las informaciones.

#### 1.3.1.7 Dokmee

Es una herramienta segura y fácil de usar. Sistema de gestión documental diseñado para una variedad de propósitos, incluyendo la captura de documentos y el almacenamiento, búsqueda y recuperación y compartición de archivos. Dokmee se adapta a cualquier modelo de negocio al maximizar la accesibilidad y funcionalidad en los repositorios de todos los tamaños, mientras aumenta la colaboración y la comunicación entre los usuarios. Con una interfaz fácil de usar disponible en 19

idiomas. («Software de digitalización - Soluciones de Software de Gestión de Documentos > Productos > Gestión Documental» 2018)

Las integraciones entre Dokmee y otros programas, ambos basados en Windows y basada en la Web, pueden utilizar la búsqueda y recuperación, indexación, la importación y la visualización de los archivos. La barra de herramientas de fácil acceso de Microsoft Office le permite enviar archivos directamente desde Word, Excel y Outlook directamente en Dokmee. Con la Dokmee Virtual Printer, envía archivos PDF en Dokmee desde cualquier aplicación utilizando el botón de impresión. Utiliza una estructura de carpetas como Windows, de forma manual y automáticamente permite ordenar los archivos en un número ilimitado de carpetas y niveles. («Software de digitalización - Soluciones de Software de Gestión de Documentos > Productos > Gestión Documental» 2018)

Dokmee Desktop es una solución de gestión documental independiente diseñada para la pequeña oficina o el usuario de oficina en casa. Este sistema completamente funcional permite el almacenamiento y recuperación de archivos electrónicos y mucho más con un fuerte Microsoft SQL back-end incluido gratis. Tasado agresivamente para el usuario individual, Dokmee Desktop es una excelente alternativa para el almacenamiento de su información en carpetas en su equipo («Software de digitalización - Soluciones de Software de Gestión de Documentos > Productos > Gestión Documental» 2018).

Según («Software de digitalización - Soluciones de Software de Gestión de Documentos > Productos > Gestión Documental» 2018) Algunas características que tiene esta herramienta son:

- Comparte archivos en su organización.
- Almacenar y gestionar cualquier tipo de archivo en Dokmee, con construido en capacidades de visualización de PDF, TIFF, JPG, PNG, BMP, GIF, DWG, DXF, MSG, EML, y mucho más.
- Añadir anotaciones, márgenes y notas a los archivos.
- Con un visor de Microsoft Office colaborativo, abrir y editar archivos de Word, Excel, PowerPoint y Visio.
- Varios usuarios pueden ver los archivos al mismo tiempo, junto con el control de versiones para ediciones y revisiones.

### 1.3.2 Nacionales

#### 1.3.2.1 DigiPyrus

El sistema de Digitalización DigiPyrus, desarrollado por el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI), permite digitalizar los Tomos de los Registros y

Notarías de la República Bolivariana de Venezuela, contribuyendo a agilizar el acceso a la información que a diario se tramita en cada una de sus oficinas. Este sistema surge debido a la demanda de una gestión más eficiente y segura de los trámites que se realizan con los documentos archivados en las Oficinas de los Registros y Notarías Públicas. De ahí, que se haya propuesto presentar una solución que cumpla con el objetivo de dotar a esta entidad de un sistema que contenga las funciones, para realizar el proceso de digitalización de todos los documentos archivados y de esta forma obtener el fondo digital de cada una de estas (La Rosa Montes 2007).

Procesos: Los procesos que se informatizan (La Rosa Montes 2007) mediante este sistema son:

- Recepción de Documentos: Verificación y registro de los tomos provenientes del Almacén Temporal
- Preparación de documentos: Ejecución de las operaciones establecidas para garantizar que los tomos cumplan con los parámetros necesarios para el proceso de digitalización.
- Escaneo de Documentos: Conversión de los tomos físicos en objetos digitales multipáginas.
- Control de la Calidad: Inspeccionar el 100% del objeto digital multipáginas proveniente del área de Digitalización de Documentos.
- Encuadernación de Documentos: Recuperación y preservación de las condiciones iniciales de los tomos digitalizados.
- Devolución de Documentos: Registro de salida de los tomos que han sido digitalizados en el proceso.

#### 1.3.2.2 DigiDaP

DigiDAP forma parte de la Solución Tecnológica Integral para la automatización y modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela, como el subsistema que informatiza los procesos del Centro de Digitalización para el fondo documental de dicha institución. Su objetivo fundamental es garantizar la obtención de objetos digitales con valor legal a partir de la digitalización del fondo para mejorar los servicios de inscripción y certificación de antecedentes penales a los ciudadanos de la nación venezolana (González Valdés, Suárez Font y Lizama Mué 2011).

Procesos: En el modelo de referencia para la digitalización de fondos documentales de antecedentes penales (González Valdés, Suárez Font y Lizama Mué 2011) se tiene

- Recepción y Devolución de Documentos: Recepcionar expedientes al centro de digitalización, devolver desde este al archivo y eliminar expedientes que por errores se decidan quitar del proceso.

- Preparación de Documentos: Preparar los expedientes y garantizar que tengan las condiciones necesarias para pasar por el escáner en la próxima área, evaluar la calidad de las unidades documentales para determinar si proceden o no en el proceso y registrar los folios a digitalizar de cada una y emitir notas de preparación en caso de que sea necesario para esclarecer la omisión de folios a digitalizar.
- Digitalización de Documentos: Digitalizar las unidades documentales contenidas en los expedientes, rectificar errores en los trámites que hayan sido regresados en el proceso y mejorar la calidad visual de los documentos digitales.
- Control de Calidad: Revisar legalmente los trámites realizados en el centro de Digitalización, emitir la respuesta correspondiente del trámite e imprimir salidas del sistema.
- Otorgamiento: Otorgar trámites revisados legalmente, certificando la calidad del proceso realizado, así como otorgar firma manuscrita a las salidas impresas del sistema y la firma digital a los documentos digitales, obtener reportes estadísticos sobre el proceso y localizar trámites dentro del proceso y consultar el estado de los mismos.
- Encuadernación de Documentos: Encuadernar y foliar las unidades documentales dentro de los expedientes para su asentamiento en archivo.

#### 1.3.2.3 CDA

CDA es la solución tecnológica desarrollada para el Centro de Digitalización de Alfabéticas para el Servicio Administrativo de Identificación, Migración y Extranjería. Para su desarrollo se utilizó como marco de trabajo el utilizado por la solución DigiPyrus por lo que las características arquitectónicas y tecnologías de desarrollo utilizadas son similares. Su objetivo fundamental es extraer tarjetas alfabéticas para convertirlas en metadatos. Esta solución está compuesta por varios módulos, algunos de estos son (La Rosa Montes 2007):

- Almacén Temporal: permite la gestión de inventario de los recursos que entran y salen del Centro de Digitalización, así como del control de recibo y despacho de las Tarjetas de Alfabética desde las Oficinas de Identificación hacia cada Centro de Digitalización.
- Preparación: permite la creación de los lotes de documentos en el sistema, se registran los documentos rechazados por malas condiciones para entrar en el proceso de digitalización y se realiza la corrección de los documentos rechazados por mala preparación o códigos de barra ilegible.
- Digitalización: permite la digitalización de los lotes de documentos o de documentos rechazados.

- Firma digital: permite la emisión de la firma digital de los documentos, certificando que la información obtenida es verídica.

### 1.3.3 Conclusiones del análisis de los sistemas homólogos

En la tabla de resumen (ver Tabla 1) se muestra una comparación entre los sistemas anteriormente mencionados, reflejando sus ventajas y deficiencias dando paso a la argumentación de la idea del desarrollo de un nuevo sistema. Para la realización de esta tabla se tuvo en cuenta la licencia de pago, si organizaba la información, si es un sistema multiplataforma y si digitaliza o no.

*Tabla 1: Tabla comparativa de sistemas homólogos*

| Software | Indicadores | | | |
|---|---|---|---|---|
| | Organización de la Información | Digitalización | Multiplataforma | Licencia de pago |
| **Scan2PDF** | No | Sí | No | No |
| **Nuxeo** | Sí | No | Sí | Sí |
| **Alfresco** | Sí | Sí | Sí | Sí |
| **Athento** | Sí | No | Sí | Sí |
| **Kofax Express** | Sí | Sí | No | Sí |
| **SE Capture** | Sí | Sí | No | Sí |
| **Dokmee** | Sí | No | Sí | Sí |
| **DigiPyrus** | Sí | Sí | Sí | Sí |
| **DigiDAP** | Sí | Sí | Sí | Sí |
| **CDA** | Sí | Sí | Sí | Sí |

**Fuente**: Creación propia

El análisis guiado por la tabla anterior permitió conocer la existencia de sistemas de digitalización en Cuba y en el mundo, desarrollados en distintos lenguajes y con el mismo propósito. Sobre el estudio de las aplicaciones existentes a nivel internacional se tiene que no resultan soluciones factibles a tener en cuenta, pues fueron desarrolladas con software propietario lo que implica gastos muy elevados en licencia y mantenimiento. Aunque Scan2PDF no es privativo tampoco se puede considerar para la solución del problema, puesto que no es un sistema multiplataforma y no organiza la información que es una de las principales funcionalidades a tener en cuenta.

Sobre el estudio de las herramientas a nivel nacional, estas fueron desarrolladas con software propietario, por lo que no pueden ser usadas para la solución del problema expuesto en la investigación. Sin embargo, en un análisis a profundidad fueron identificadas un conjunto de características a tener en cuenta en la solución a construir como son: la adaptabilidad a cualquier entorno, la edición y clasificación de documentos digitales a través de categorías jerárquicas, la conversión de tomos físicos en objetos digitales multipáginas y la posibilidad de guardar archivos en formato PDF en directorios controlados.

## 1.4 Metodología de desarrollo de software

En el presente trabajo se realizó un estudio para la determinación de la metodología de desarrollo de software a utilizar en el desarrollo del sistema, donde se obtuvo lo siguiente:

Según (Roger S. Pressman 2002a) las metodologías de desarrollo de software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable.

Las metodologías se clasifican en 2 grupos (ágiles y tradicionales), las cuales se seleccionan en dependencia de sus características que se reflejan en la Tabla 2 que se encuentra a continuación.

*Tabla 2: Comparación de los tipos de metodologías de desarrollo de software*

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

**Fuente**: Elaboración propia

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

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

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

### 1.4.1 Definición de la metodología de desarrollo de software

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

#### 1.4.1.1 Scrum

Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming (Ken Schwaber 2004).

Scrum se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo, en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente (Ken Schwaber 2004).

Está especialmente indicada para proyectos con rápidos cambios de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Véase en la Ilustración 1 el ciclo de vida de Scrum.


*Ilustración 1: Ciclo de vida de Scrum*

**Fuente**: («Desenvolvimento Ágil» 2009)

### 1.4.1.2 Programación extrema (XP)

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software formulado por (Kent Beck 1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (José M. Batista 2009). Véase en la Ilustración 2 el ciclo de vida de XP.

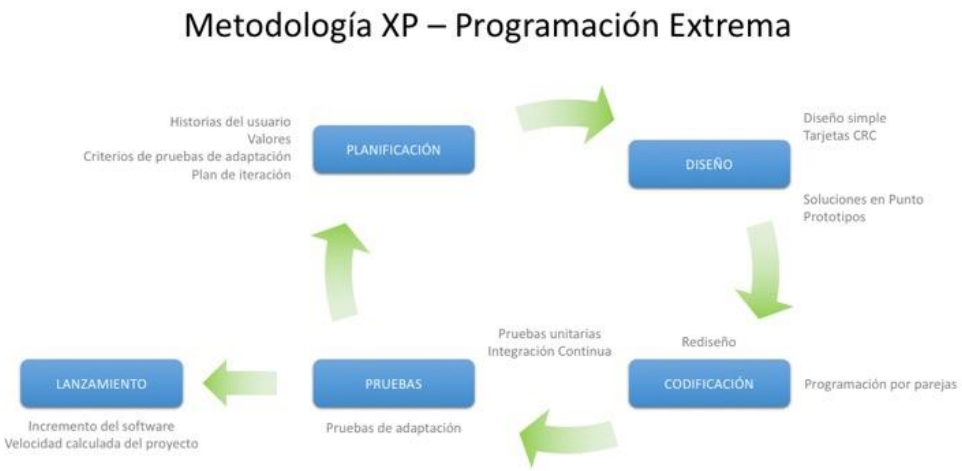



*Ilustración 2: Ciclo de vida de XP*

**Fuente**: (Rojas 2018)

### 1.4.1.3 AUP – UCI

Es una variación de la metodología AUP (Proceso Unificado Ágil o Agile Unified Process), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI, en unión del modelo CMMI-DEV v1.3, ya que está definida por el proyecto que desarrolla el módulo. Además, genera los artefactos que permiten obtener la documentación necesaria para una mejor comprensión de la herramienta a desarrollar (Tamara Rodríguez Sánchez 2015).

Esta metodología cuenta de tres fases: inicio, ejecución y cierre por las que deben transitar durante el desarrollo de las actividades productivas. La presente investigación elige la fase de ejecución, en la que se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura (Tamara Rodríguez Sánchez 2015).

Durante la ejecución se obtiene los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto**.** De las siete disciplinas que propone la metodología se eligen seis, las cuales son: requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación y pruebas de aceptación. De las disciplinas mencionadas se definen 4 escenarios de trabajo los cuales son:

**Escenario No 1**: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los caso de uso del negocio muestran como los procesos son llevados a cabo por personas y los activos de la organización (Tamara Rodríguez Sánchez 2015).

**Escenario No2:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (Tamara Rodríguez Sánchez 2015).

**Escenario No 3:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y las relaciones entre los procesos identificados (Tamara Rodríguez Sánchez 2015).

**Escenario No4:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una historia de usuario (HU) no debe poseer demasiada información (Tamara Rodríguez Sánchez 2015).

### 1.4.2 Selección de la metodología.

De lo expuesto anteriormente resultó la siguiente tabla de comparación (ver Tabla 3):

*Tabla 3: Comparación entre XP,SCRUM y AUP-UCI*

| Criterio de comparación | XP | Scrum | AUP-UCI |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **Tamaño de los proyectos** | Pequeños y medianos | Pequeños, medianos y grandes | Pequeños, medianos y grandes |
| **Tamaño del equipo** | Menores de 10 | Múltiples equipos menores de 10 | Todo tipo de equipos |
| **Estilo de desarrollo** | Iterativo y rápido | Iterativo y rápido | Iterativo y rápido |

**Fuente**: Elaboración propia

Para guiar el desarrollo de la solución se decide utilizar AUP-UCI como metodología de desarrollo. Se opta por el cuarto escenario, el cual define que se utilizan en proyectos que no modelen el negocio y se concentren en modelar el sistema mediante HU. En este escenario el cliente acompaña al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Además, se recomienda en proyectos no muy extensos, ya que una HU no debe poseer mucha información.

## 1.5 Lenguajes

En esta sección se definirán los lenguajes a utilizar para el desarrollo del sistema.

### 1.5.1 Lenguaje de modelado

El Lenguaje de Modelado Unificado (UML) es “un lenguaje estándar para escribir diseños de software. El UML puede usarse para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo” (Roger S. Pressman 2010a).

Existen muchas características opcionales en diagramas de UML. El UML ofrece dichas opciones (en ocasiones complejas) de modo que pueda expresar todos los aspectos importantes de un sistema. Al mismo tiempo, tiene la flexibilidad para suprimir aquellas partes del diagrama que no son relevantes para el aspecto que se va a modelar, con la finalidad de evitar confundir el diagrama con detalles irrelevantes. Por tanto, la omisión de una característica particular no significa que ésta se encuentre ausente; puede significar que la característica se suprimió (Roger S. Pressman 2010a).

El UML es principalmente un lenguaje para describir sistemas orientados a objetos independientes de cualquier lenguaje de programación específico. Es simple de aprender, y bastante flexible, y consistente desde el planeamiento hasta el despliegue. Los beneficios de usar UML incluyen la trazabilidad, mejorada, inteligibilidad entre los usuarios y un mantenimiento realmente simplificado («Herramientas de ingeniería de Software para desarrollo y modelado de software» 2018).

### 1.5.2 Lenguaje de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo («Lenguajes de programación» 2017).

Un lenguaje de programación se puede usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos, reglas sintácticas y reglas semánticas que definen su estructura, el significado de sus elementos y expresiones.

El lenguaje de programación que se utilizará para el desarrollo del sistema es Java. Este es un lenguaje que está diseñado para la implementación de sistemas libres, orientado a objeto, distribuido, interpretado y compilado, seguro, portable, de alto desempeño, multihilo y dinámico.

Java es orientado a objetos dando buen soporte a la reutilización de componentes de software. Es distribuido porque se ha diseñado para trabajar en ambientes de redes teniendo un repositorio centralizado de bibliotecas para distintas funciones. Mediante el código Java se pueden manipular recursos URL con la misma facilidad que C y C++ (Mujica y Herrera 2005a).

Es un lenguaje interpretado**,** ya que el compilador Java traduce cada fichero fuente de clases a código de bytes (*Bytecode*), que puede ser interpretado por todas las máquinas que den soporte a un visualizador que funcione con Java. Es robusto porque el código Java no se quiebra fácilmente ante errores de programación. Es seguro porque las mismas características antes descritas que evitan la corrupción de código evitan su manipulación. Es dinámico porque al contrario de C++ que exige que se compile de nuevo la aplicación al cambiar una clase madre, Java utiliza un sistema de interfaces que permite aligerar esta dependencia. Como resultado, los programas Java pueden permitir nuevos métodos y variables en un objeto de biblioteca sin afectar a los objetos dependientes («¿Qué es el lenguaje de programación JAVA? - Base de Conocimientos - ICTEA» 2018).

## 1.6 Herramientas para el desarrollo del sistema

En esta sección se definirán las herramientas a utilizar para el desarrollo del sistema.

### 1.6.1 Herramienta de modelado

Las herramientas de modelado de sistemas informáticos, son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Permiten crear un "simulacro" del sistema, a bajo costo y riesgo mínimo (Leandro Alegsa 2010).

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

Visual Paradigm for UML 8.0 es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. La herramienta agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como obtener ingeniería inversa de bases de datos (Dayana Mendoza Peña y Lionel Rodolfo Baquero Hernández 2016).

Se caracteriza por:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Licencia: gratuita y comercial.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversion (control de versiones).
- Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Editor de detalles de casos de uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Soporte ORM - Generación de objetos Java desde bases de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Generador de informes.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de Microsoft Visio.
- Editor de figuras.

### 1.6.2 IDE

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés), es una aplicación de software, que proporciona servicios integrales para facilitar el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDEs tienen autocompletado inteligente de código (Saez Villavicencio, Anisley de la Caridad y Ernesto Lorente Rodriguez 2009). Para la implementación del sistema de digitalización los IDE que se tuvieron en cuenta fueron NetBeans y Eclipse.

#### 1.6.2.1 Eclipse

Es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Sin embargo, también se puede usar para otros tipos de aplicaciones cliente («Eclipse, herramienta universal – IDE abierto y extensible» 2013).

El IDE Eclipse emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente al permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como látex, aplicaciones en red como Telnet y sistemas de gestión de base de datos («Eclipse, herramienta universal – IDE abierto y extensible» 2013).

Eclipse dispone de un editor de texto con un analizador sintáctico. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (*wizards*) para creación de proyectos, clases, test y refactorización («Eclipse, herramienta universal – IDE abierto y extensible» 2013).

#### 1.6.2.2 NetBeans

Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso (Mujica y Herrera 2005a).

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (actualmente Sun Microsystems es administrado por *Oracle Corporation*) (Mujica y Herrera 2005a).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs[3] de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (Mujica y Herrera 2005a).

El NetBeans es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente (Mujica y Herrera 2005a).

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

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

[3] API: La interfaz de programación de aplicaciones (API, por sus siglas en inglés), es el conjunto de subrutinas, funciones, procedimientos o métodos, en la programación orientada a objetos, que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. («¿Qué es Interfaz de programación de aplicaciones (API)? - Definición en WhatIs.com» 2018)

- NetBeans IDE es libre, de código abierto, multiplataforma con soporte integrado para el lenguaje de programación Java.

### 1.6.3 Selección del IDE

Luego de realizar un análisis a lo expuesto anteriormente se decidió utilizar como IDE de desarrollo el NetBeans en su versión 8.2. Se escogió este entorno de desarrollo integrado porque soporta la implementación de todo tipo de aplicaciones desarrolladas en Java y brinda muchas facilidades al programador. Se tuvo en cuenta además que el equipo de desarrollo ha estado trabajando con esta herramienta, por lo que tiene experiencia y dominio sobre la misma.

## 1.7 Bibliotecas de Clases

- **Jfoenix v1.4.0:** JFoenix es una biblioteca Java de código abierto que implementa Google Material Design utilizando componentes de Java («JFoenix» 2018).
- **java-scanner-access-twain v12.0.9:** Asprise Scanning and Imaging SDK ofrece una biblioteca API de alto rendimiento para que pueda adquirir imágenes y tener control total sobre casi todos los tipos de escáneres WIA TWAIN en Windows 32bit / 64bit y Mac OS X («Asprise Java Scanning and Imaging SDK for Java applets, web applications, Swing/JavaFX components, JEE enterprise applications - an API library scan images into files or memory from scanners on Windows and Mac OS X» 2018).
- **iText v5.5.5:** Es una biblioteca de clases para el manejo y la creación de archivos PDF's con Java («iText 5.5.5 | iText» 2015).
- **javaxt-core v1.6.2:** JavaXT es una colección de bibliotecas y utilidades de Java que proporciona una serie de funciones que no están disponibles en el estándar JDK («JavaXT - Open Source Extensions to the Core Java Library» 2018).
- **commons-net v1.4:** La biblioteca Apache Commons Net ™ implementa el lado del cliente de muchos protocolos básicos de Internet («Apache Commons Net – Overview» 2018).
- **jai-imageio-core v1.3.1:** La herramienta JavaTM Advanced Imaging Image proporciona operaciones de lectura y escritura de JAI que utilizan el marco de E/S de imagen de Java, flujos de entrada y salida de imágenes que utilizan las nuevas API de E/S y los complementos de lectura y escritura de imágenes («jai-imageio-core Java documentation Version 1.4.0» 2018).

## 1.8 Conclusiones del capítulo

- El estudio de los principales conceptos asociados al objeto de estudio permitió conocer en detalle el proceso que realizan los sistemas de digitalización.

- A partir del estudio de soluciones existentes se identificó que no existía ningún sistema que le diera solución al problema planteado por lo que se decide construir un sistema nuevo.
- El análisis de las diferentes metodologías de desarrollo de software permitió concluir que la metodología de desarrollo AUP-UCI es la más indicada para guiar el desarrollo del sistema.
- Se realizó un estudio de las tecnologías, herramientas y tendencias del desarrollo de software actuales, exponiendo las principales características y ventajas, lo que ayudó a seleccionar las herramientas para el desarrollo del sistema y facilitó la familiarización con los elementos del entorno de desarrollo a ser utilizados en la construcción de la propuesta de solución.

# CAPÍTULO 2. ANÁLISIS Y DISEÑO

## 2.1 Introducción

En este capítulo se describen los elementos asociados al análisis y diseño de la herramienta a desarrollar. Se determinó la arquitectura del sistema y los patrones de diseño para su construcción. A partir de la metodología seleccionada se hace referencia a la etapa de ejecución del proyecto, en la que se describen las Historias de Usuario para establecer posteriormente el orden en que serán implementadas atendiendo a su prioridad, con el fin de organizar el ciclo de vida del producto. Se obtienen los requisitos funcionales y no funcionales a tener en cuenta para la construcción del sistema.

## 2.2 Modelo conceptual

El Modelo conceptual es una representación visual de los objetos y conceptos (o partes esenciales) que se requieren en la modelación de un problema determinado y las relaciones que subsisten entre ellos. En el presente trabajo debido a que no existe un negocio definido 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 y así como las asociaciones, sin incluir las responsabilidades de las personas que ejecutan las actividades.

### 2.2.1 Descripción de los principales conceptos

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

- **Universidad**: es el centro que presenta la situación y donde se va a implantar la solución.
- **Secretaría**: es el área de la universidad donde se va desarrollar todo el proceso de digitalización.
- **Escáner**: es la herramienta mediante la cual se realiza el proceso de digitalización.
- **Expediente**: es el conjunto de los documentos que van a ser escaneados.
- **Archivo digital**: Es la representación digital del documento original.
- **Digitalización**: es el proceso que permite obtener los archivos digitales.

### 2.2.2 Descripción del proceso

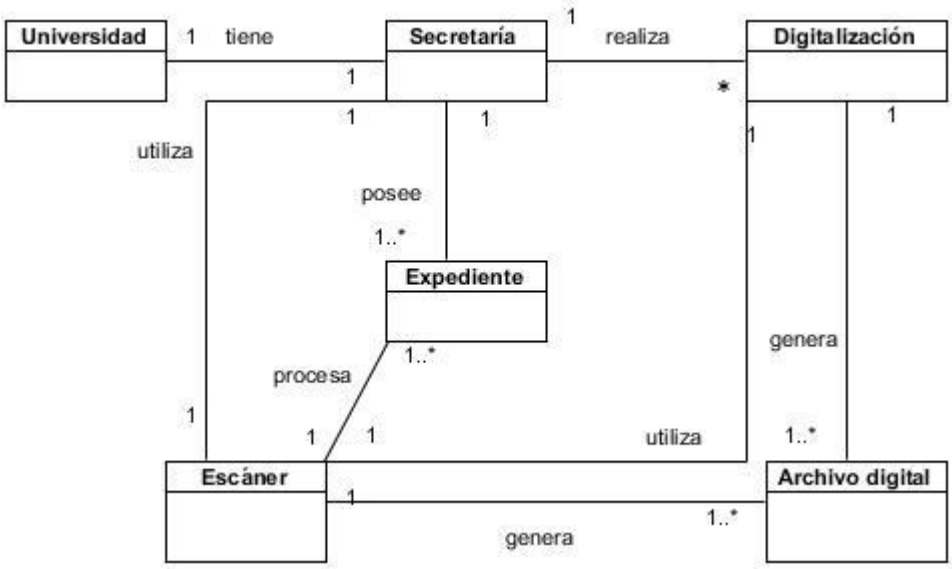



*Ilustración 3: Modelo conceptual*

**Fuente**: Elaboración Propia

## 2.3 Requisitos del sistema

En el proceso de producción de software es fundamental tener bien claros los requisitos del sistema, con el objetivo de tener definida una línea de lo que se desea desarrollar. La captura de estas especificaciones permite y asegura un sistema con eficiencia y calidad. Estas se pueden obtener utilizando la descripción de las condiciones y/o capacidades que el producto debe cumplir, debiendo ser lo suficientemente abarcadoras como para que se puedan llegar a acuerdos entre los clientes y los desarrolladores en cuanto a lo que debe o no hacer el software.

Los requerimientos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama ingeniería de requerimientos (IR) (Ian Sommerville 2011a).

Los requisitos del sistema no son más que esa condición que debe ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. Se denomina como: condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

### 2.3.1 Técnicas de capturas de requisitos

En el proceso de desarrollo de software los analistas pueden emplear varias técnicas para obtener los requisitos del cliente. Históricamente, esto ha incluido técnicas tales como las entrevistas, o talleres con grupos para crear listas de requisitos. Técnicas más modernas incluyen los prototipos, y utilizan casos de uso. Cuando sea necesario, el analista empleará una combinación de estos métodos para establecer los requisitos exactos de las personas implicadas, para producir un sistema que resuelva las necesidades del cliente. A continuación, se muestran las técnicas a utilizar en la investigación:

#### 2.3.1.1 Entrevistas

Las entrevistas son la técnica de elicitación más utilizada, y de hecho son prácticamente inevitables en cualquier desarrollo ya que son una de las formas de comunicación más naturales entre personas (Amador Durán Toro y Beatriz Bernández Jiménez 2000). Por lo general no se entrevista a toda la gente que se relacionará con el sistema, sino a una selección de personas que represente a todos los sectores críticos de la organización, con el énfasis puesto en los sectores más afectados o que harán un uso más frecuente del nuevo sistema. (Ver **¡Error! No se encuentra el origen de la referencia.**)

#### 2.3.1.2 Lluvia de ideas

Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Las sesiones suelen estar formadas por un número de cuatro a diez participantes, uno de los cuales es el jefe de la sesión, encargado más de comenzar la sesión que de controlarla. Como técnica de elicitación de requisitos puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de elicitación, cuando los requisitos son todavía muy difusos (Amador Durán Toro y Beatriz Bernández Jiménez 2000).

El principio básico es no descartar de manera apresurada ningún planteo, de modo que existe la posibilidad de que surjan otras ideas derivadas de la idea original y se generan varios puntos de vista del problema.

### 2.3.2 Requisitos funcionales (RF)

Los RF son capacidades o condiciones que el sistema debe cumplir. Expresan una especificación detallada de las responsabilidades del sistema en cuestión y permiten determinar de una manera clara, lo que debe hacer el sistema (Roger S. Pressman 2002a).

#### 2.3.2.1 Listado de Requisitos Funcionales

**RF**1 Visor Scanner-Imagen (Área de trabajo)

**RF2** Aumentar imagen

**RF3** Disminuir imagen

**RF4** Ir al inicio del documento

**RF5** Ir al final del documento

**RF6** Ir a la página anterior

**RF7** Ir a la página siguiente

**RF8** Girar a la izquierda

**RF9** Girar a la derecha

**RF10** Seleccionar origen

**RF11** Escanear documento

**RF12** Insertar página

**RF13** Adicionar página

**RF14** Eliminar página

**RF15** Configurar conexión de acceso al servidor remoto

**RF16** Probar conexión al servidor remoto

**RF17** Conectar con el servidor remoto

**RF18** Mostrar el estado actual de la conectividad

**RF19** Sincronizar los datos con el servidor remoto

**RF20** Desconectar del servidor remoto

**RF21** Abrir documento local

**RF22** Guardar documento local

**RF23** Eliminar documento local

**RF24** Renombrar documento local

### 2.3.3 Requisitos no funcionales (RNF)

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares. A menudo se aplican al sistema en su totalidad. Se aplican a características o servicios individuales del sistema. Los requerimientos no funcionales, como su nombre sugieren, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. 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 (Ian Sommerville 2011a). Por lo general los requisitos no funcionales son fundamentales en el éxito del producto; normalmente están vinculados a los requisitos funcionales, es decir, una vez que se conoce lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades o propiedades debe tener. Los requisitos no funcionales han de especificarse cuantitativamente, siempre que sea posible para que se pueda verificar su cumplimiento. A continuación, se muestran los requisitos no funcionales que se han definido para el desarrollo de la herramienta por parte del equipo de trabajo.

#### 2.3.3.1 Listado de Requisitos no Funcionales

**De hardware y software**

**RNF1** La PC tiene que tener instalado el JRE 1.8.x.

**RNF2** La aplicación tiene que ser multiplataforma

La computadora donde se va a ejecutar la aplicación debe tener los siguientes requisitos mínimos:

**RNF3** Memoria RAM de 256 Mb o superior.

**RNF4** Microprocesador dual Core a 1.2 GHz o superior.

**Portabilidad**

**RNF5** El sistema será multiplataforma por lo cual se podrá utilizar en cualquier sistema operativo.

**Disponibilidad**

**RNF6** El sistema podrá ser usado en cualquier momento por todos los usuarios autorizados las 24 horas del día los 7 días de la semana.

**Apariencia o interfaz de usuario**

**RNF7** Se requiere de una interfaz agradable, legible y sencilla de usar.

**RNF8** Se deben respetar los estándares de la institución (utilizar colores como blanco o azul en la interfaz).

**RNF9** Todas las interfaces y mensajes de información de éxito o fallo de una operación tienen que estar en idioma español.

**Usabilidad**

**RNF10** El sistema debe permitir al usuario conocer las acciones que puede realizar, a partir de íconos sugerente, alternativas textuales u otro elemento que le permita al usuario un mejor trabajo con el mismo.

**RNF11** El sistema debe ser de uso intuitivo, que le brinde facilidad de uso a los usuarios, los elementos de la interfaz como menús y zonas de trabajo se deben encontrar en posiciones fijas y con la mayor uniformidad posible entre cuadro de textos y botones.

### 2.3.4 Historias de usuario

Siguiendo con la definición de requerimientos de software de (Standards Coordinating Committee of the Computer Society of IEEE 1990) citada en la sección Introducción a los Requerimientos de Software, se puede concluir que las Historias de Usuario (HU) son requerimientos ya que expresan el problema que el sistema o producto software debe resolver. Las HU son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son

descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario (María Paula Izaurralde 2013).

Las HU son la técnica para especificar las funcionalidades que brinda el sistema y constituye una manera dinámica de realizar esta actividad. Como su nombre lo indica son especificadas por los propios usuarios y por tanto redactadas en su lenguaje; de manera sencilla y breve, evitando tecnicismos innecesarios que puedan crear confusión, aunque los programadores pueden contribuir en la tarea. Además, son la base para realizar las pruebas de aceptación, así como la estimación y planificación del proyecto.

Una vez identificadas las historias de usuario necesarias para liberar una primera versión operativa los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de programación que están escritas técnicamente y que darán solución a su HU correspondiente.

#### 2.3.4.1 Descripción de las HU

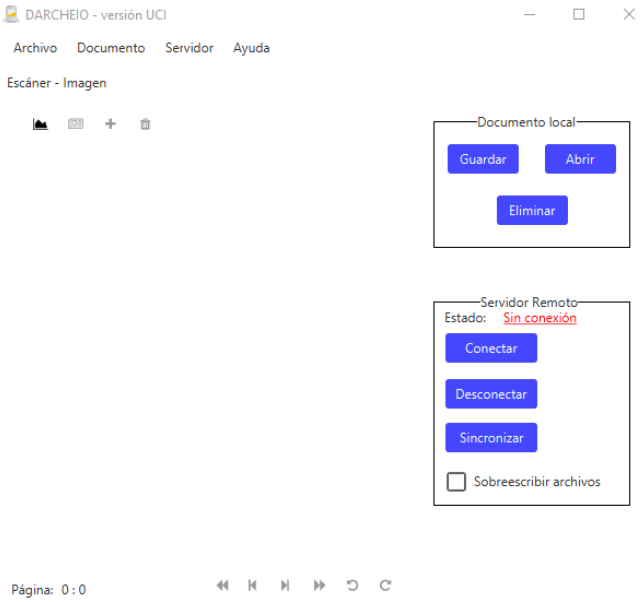
De los requisitos funcionales que se especificaron se realizaron las HU correspondientes. A continuación, se muestran dos ejemplos definidos para implementar la solución, el resto se encuentran en los anexos del documento. (Ver Anexos del 2 al 23)

##### 2.3.4.1.1 HU Visor Scanner-Imagen

*Tabla 4: HU-1 Visor Scanner-Imagen*

| **HU-1 Visor scanner-imagen** | |
|---|---|
| **Número:** 1 | **Nombre del requisito:** Visor scanner-imagen |
| **Programador:** Alex A. Hurtado Campuzano y Haraid Martínez Cano | **Iteración Asignada: 1** |
| **Prioridad: Alta** | **Tiempo Estimado: 20h** |
| **Riesgo en Desarrollo:** N/A | **Tiempo Real: 28h** |
| **Descripción:**<br>Es el área donde se va trabajar con el documento escaneado permitiendo una mejor visualización a través de las diferentes funcionalidades que este posee, por ejemplo: **aumentar imagen**, **disminuir imagen**, **ir al inicio del documento, ir al final del documento**, **página anterior**, **página siguiente, girar a la izquierda**, **girar a la derecha**, **seleccionar origen**, **escanear documento**, **insertar página**, **adicionar página**, **eliminar página.** | |
| **Observaciones:** N/A | |

**Prototipo elemental de interfaz gráfica de usuario:**




**Fuente**: Elaboración Propia

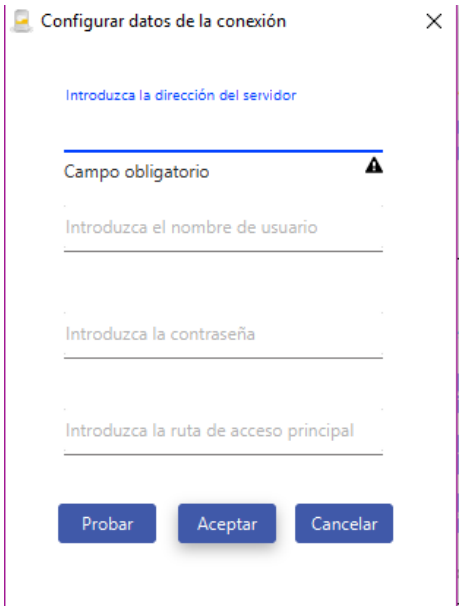
#### 2.3.4.1.2 HU Configurar conexión de acceso al servidor remoto

*Tabla 5: HU-15 Configurar conexión*

| **HU-15 Configurar conexión** | |
|---|---|
| **Número:** 15 | **Nombre del requisito:** Configurar conexión |
| **Programador:** Alex A. Hurtado Campuzano y Haraid Martínez Cano | **Iteración Asignada: 2** |
| **Prioridad: media** | **Tiempo Estimado: 20h** |
| **Riesgo en Desarrollo:** N/A | **Tiempo Real: 15h** |
| **Descripción**:<br>Permite configurar los parámetros de la conexión con el servidor. Los campos a rellenar son Dirección remota, Usuario, Contraseña y Ruta de acceso en el servidor, el primero es un campo obligatorio mientras que los otros son opcionales, en el caso de no especificar ningún usuario entonces se toma por defecto el usuario anónimo, para la contraseña ocurre de igual manera que para el usuario y la ruta de acceso al servidor es la dirección a partir de la cual se va a empezar a guardar los archivos en el servidor remoto. En la misma venta aparecen varios botones: Probar, Aceptar el cual guarda todos los datos necesarios para la conexión, y el Cancelar que cierra la ventana descartando todos los cambios. | |

**Observaciones:** N/A

**Prototipo elemental de interfaz gráfica de usuario:**




**Fuente**: Elaboración Propia

## 2.4 Patrones y arquitectura

A continuación, se definirá la arquitectura de software a utilizar para el desarrollo del sistema que no es más que una vista conceptual de toda su estructura y además se determinarán los patrones a aplicar para lograr el esquema organizativo y estructural del software.

### 2.4.1 Arquitectura de software

La arquitectura de software alude a la estructura general del software y las formas en que la estructura proporciona una integridad conceptual para un sistema. En su forma más simple, la arquitectura es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan, y la estructura de datos que utilizan los componentes. En sentido más amplio, sin embargo, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones (Roger S. Pressman 2002a).

Un estilo arquitectónico es una transformación que se impone al diseño de todo el sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En el caso en el que ha de hacerse la reingeniería de una arquitectura ya existente, la imposición de un estilo arquitectónico dará como resultado cambios fundamentales en la estructura del software, incluida la reasignación de las funciones de los componentes (Roger S. Pressman 2010a).

La arquitectura basada en capas se enfoca principalmente en el agrupamiento de funcionalidades relacionadas dentro de una aplicación en distintas capas que son colocadas verticalmente una encima de otra. La funcionalidad dentro de cada capa se relaciona con un rol o responsabilidad específica. El dividir en capas una aplicación, permite la separación de responsabilidades lo que proporciona una mayor flexibilidad y un mejor mantenimiento. Por ejemplo, en una aplicación con una capa de presentación, una capa de lógica y una capa de acceso a datos, la responsabilidad de la capa de presentación es la de interactuar con el usuario, solicitando y proporcionando la información que el usuario requiere.

Se seleccionó esta arquitectura debido a que el equipo de trabajo escogió la tecnología JavaFX la cual recomienda la utilización de una arquitectura por capas, facilitándose el uso del patrón arquitectónico modelo-vista-controlador.

#### 2.4.1.1 Modelo – Vista – Controlador

Un patrón arquitectónico se puede considerar como una descripción abstracta estilizada de buena práctica, que se ensayó y puso a prueba en diferentes sistemas y entornos. De este modo, un patrón arquitectónico debe describir una organización de sistema que ha tenido éxito en sistemas previos. Debe incluir información sobre cuándo es y cuándo no es adecuado usar dicho patrón, así como sobre las fortalezas y debilidades del patrón (Ian Sommerville 2011a).

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo (Ian Sommerville 2011a).

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa referentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo (Ian Sommerville 2011a).

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo (Ian Sommerville 2011a).

El patrón MVC es utilizado ya que para el desarrollo del software se utilizó la tecnología JavaFX 2.0, la cual recomienda y utiliza este patrón por defecto además de encargarse de la interacción entre las capas. La estructura del proyecto es la siguiente:

En la Ilustración 4 se muestra el módulo o capa de Modelo o dominio del proceso, en la cual están las clases manejadoras de conexión al escáner, entrada y salida de datos a ficheros, entre otros.

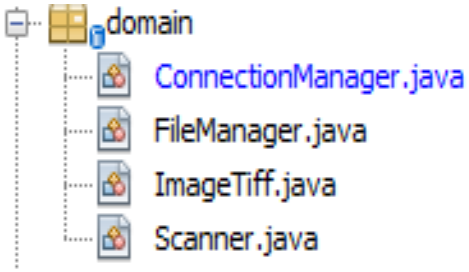



*Ilustración 4: Capa de modelo*

**Fuente**: Elaboración Propia

En la Ilustración 5 se muestra el módulo o capa de Vista o recursos, la que posee los ficheros de estilos, las imágenes y archivos de interfaz visual. Las vistas en JavaFX es dada por archivos FXML[4] que son archivos de XML[5] que definen la interfaz gráfica y que delegan a los controladores las acciones a realizar cuando el usuario haga uso de cada uno de los componentes de la interfaz de usuario.

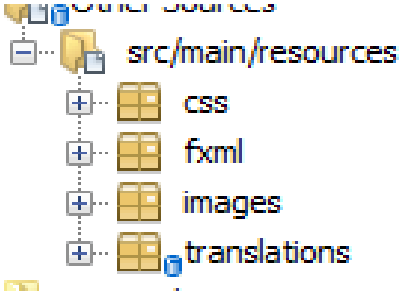



*Ilustración 5: Capa de vista*

**Fuente**: Elaboración Propia

En la Ilustración 6 se muestra el módulo o capa de controladores, la cual muestra las clases encargadas de controlar todas las acciones realizadas por el usuario al interactuar con la interfaz de usuario. El controlador decide cómo se comportará la vista y se comunicará con el modelo de negocio de acuerdo a sus necesidades.

[4] FXML es un lenguaje de marcado de interfaz de usuario basado en XML creado por Oracle Corporation para definir la interfaz de usuario de JavaFX (Greg Brown 2011).

[5] XML, siglas en inglés de eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium utilizado para almacenar datos en forma legible (ABRAHAM SILBERSCHATZ 2006).

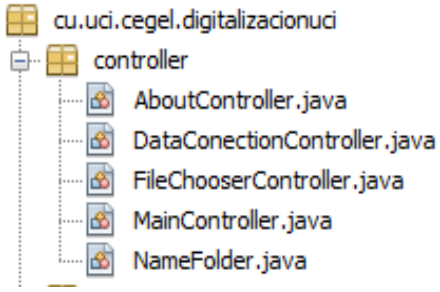



*Ilustración 6: Capa de controladores*

**Fuente**: Elaboración Propia

A continuación, se muestra una imagen explicando el uso de la arquitectura MVC en el desarrollo del sistema.

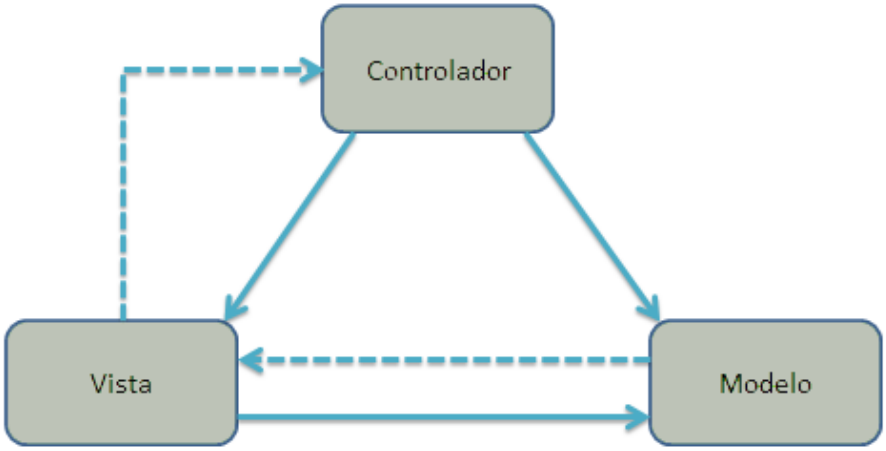



*Ilustración 7: Uso de la arquitectura MVC*

**Fuente**: Elaboración Propia

### 2.4.2 Diagrama de clases

Los diagramas de clase pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases. De manera holgada, una clase de objeto se considera como una definición general de un tipo de objeto del sistema. Una asociación es un vínculo entre clases, que indica que hay una relación entre dichas clases (Ian Sommerville 2011a).

Los diagramas de clase en el UML pueden expresarse con diferentes niveles de detalle. Cuando se desarrolla un modelo, la primera etapa con frecuencia implica buscar en el mundo, identificar los objetos esenciales y representarlos como clases (Ian Sommerville 2011a).

A continuación, se muestra el diagrama de clases que representa el sistema de digitalización a desarrollar y la relación entre sus clases:

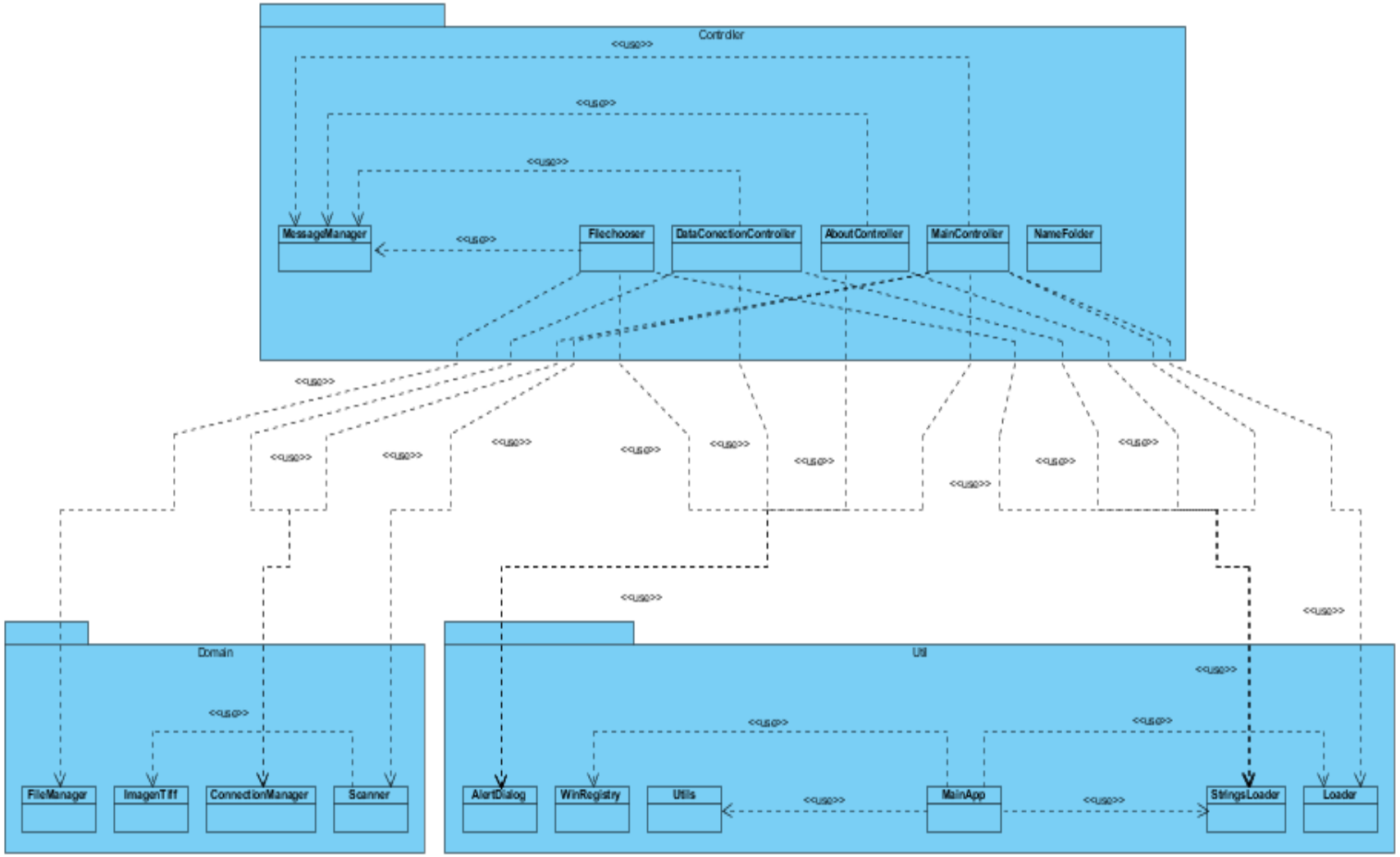



*Ilustración 8: Diagrama de clases del diseño*

**Fuente**: Elaboración Propia

### 2.4.3 Patrones de diseño

Los patrones constituyen una guía para el diseño del software. Su objetivo es la solución de problemas que ocurren repetidamente dentro de un contexto muy bien definido. Además, deben ser reusables, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. Son de gran utilidad para describir las mejores prácticas, buenos diseños, y encapsulan la experiencia permitiendo su reutilización. Definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del sistema, con el objetivo de facilitar la tarea del diseño. (Ian Sommerville 2011a)

#### 2.4.3.1 Patrones GRASP

**Experto**: propone que la clase que contenga toda la información necesaria será la responsable de la creación de un objeto o la implementación de un método. El comportamiento se distribuye entre las que contienen la información requerida, siendo más fáciles de entender, mantener y ampliar, aumentando sus posibilidades de reutilización (Larman y Moros Valle 2010). Se observa el uso de este patrón en todas las clases a utilizar en la solución ya que cada clase conoce su información y es la encargada de implementar las funcionalidades que brindan información de las mismas. El patrón experto es utilizado en las clases especializadas en acciones específicas como por ejemplo la clase FileManager, que se encarga de todo lo relacionado con entrada y salida de datos a ficheros.

```
public class FileManager {

    private static FileManager instance;

    public static FileManager getInstance() {...6 lines }

    public String Edit(String oldv, String newv, String cu

    public void Save(String path) throws IOException, Docu
```

*Ilustración 9: Uso del patrón experto*

**Fuente**: Elaboración Propia

**Controlador**: el patrón controlador funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la interfaz quien recibe los datos del usuario y los envía a las distintas clases según el método invocado (Larman y Moros Valle 2010). Este patrón fue utilizado en el paquete Controllers que es el encargado de enviar y recibir las instrucciones hacia las demás clases.

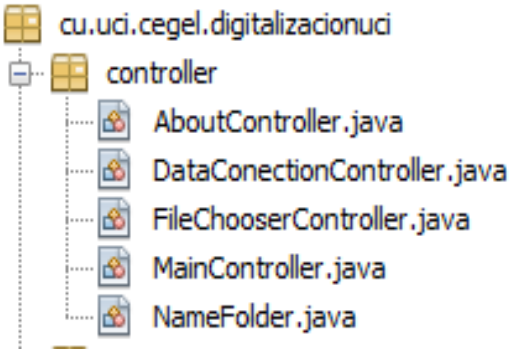



*Ilustración 10: Uso del patrón controlador*

**Fuente**: Elaboración Propia

**Bajo acoplamiento**: este patrón expresa que entre las clases deberán existir pocas ataduras, es decir, estarán lo menos relacionadas posible de forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, incrementando la reutilización y disminuyendo la dependencia entre las clases (Larman y Moros Valle 2010). El uso de este patrón se evidencia en la poca relación existente entre las clases ya que solamente hay tres clases en el diseño que tienen tres o más relaciones manteniéndose las otras 13 con una o dos relaciones solamente.

**Alta cohesión**: propone que la información que almacena una clase debe de ser coherente y debe estar, en la medida de lo posible relacionada con la clase. Al realizar un cambio en una clase con alta cohesión, todos los métodos que pueden verse afectados, estarán a la vista, en el mismo archivo. Incrementa la claridad, la reutilización y la facilidad de comprensión del diseño (Larman y Moros Valle 2010). El uso de este patrón se evidencia en las siguientes imágenes:

```
public class ConnectionManager {

    private static ConnectionManager instance;

    public static ConnectionManager getInstance()

    public ConnectionManager() {...3 lines }


    private FTPClient ftpClient;
    private String host = "";
    private String username = "";
    private String password = "";
    private String home_path = "";
    private boolean isconfigured = false;

    public boolean isConnected() {...3 lines }

    public FTPClient getFtpClient() {...3 lines }

    public String getHost() {...3 lines }

    public String getUsername() {...3 lines }
```

```
public class AlertDialog {

    private boolean clicked = false;

    public AlertDialog(StackPane stackPane,
            String textoBoton1, String textoBoton2,
            String titulo) throws IOException {...3 li

    public AlertDialog(StackPane stackPane,
            String textoBoton1, String textoBoton2,
            String titulo, String contenidoTexto,
            Type tipo) {...3 lines }
    Confirm resp = Confirm.NO;

    public Confirm GetResponse() {...3 lines }

    private void CreateAlert(StackPane stackPane,
            String bt1text, String btn2text,
            String title, String content, Type tipo) {

    String name = "";

    public String getName() {...3 lines }

    private void CreateInputDialog(StackPane stackPane
            String bt1text, String btn2text,
            String title) throws IOException {...42 li
```

*Ilustración 11: Uso del patrón Alta cohesión*

**Fuente**: Elaboración propia

#### 2.4.3.2 Patrones GOF

**Proxy**: cuando no se desea el acceso directo a un objeto sobre el que se va a aplicar determinada acción, este patrón propone la adición de un nivel que permita solamente el acceso al objeto a través de un objeto proxy sustituto, que será el responsable de controlar o mejorar el acceso al objeto real (Larman y Moros Valle 2010).

Este patrón se evidencia en el trabajo de clases como AlertDialog y StringsLoader que son clases intermedias que median el uso de los objetos de mensajes de diálogos y mensajes de texto respectivamente, por lo que para mostrar alguna ventana emergente con un mensaje se necesita una instancia de estas clases. También se puede apreciar este patrón en la clase Loader para poder cargar las interfaces visuales en el sistema, ya que dicha clase gestiona las peticiones y muestra cada una de las vistas requeridas por el usuario.

```
public class AlertDialog {

    private boolean clicked = false;

    public AlertDialog(StackPane stackPane, String textoBoton1,
            String textoBoton2, String titulo,
            String contenidoTexto, Type tipo) {...3 lines }
    Confirm resp = Confirm.NO;

    public Confirm GetResponse() {...3 lines }

    private void CreateAlert(StackPane stackPane, String bt1text,
            String btn2text, String title, String content,
            Type tipo) {...55 lines }
```

*Ilustración 12: Uso del patrón proxy*

**Fuente**: Elaboración Propia

**Singleton**: el patrón singleton es un patrón de diseño de software que restringe la creación de instancias de una clase a un objeto. Esto es útil cuando se necesita exactamente un objeto para coordinar acciones en todo el sistema (Larman y Moros Valle 2010). Solo se necesita una única instancia de la clase Scanner, por lo que cumple con los requerimientos para el patrón antes mencionado

```
private static Scanner instance;

public static Scanner getInstance() {...6 lines }

public Scanner() {...5 lines }

private List<BufferedImage> vector = new ArrayList<>();

private int currentPos = -1;

private final Imaging imaging = new Imaging("myApp", 0);
```

*Ilustración 13: Uso del patrón singleton*

**Fuente**: Elaboración Propia

## 2.5 Planificación

En esta sección se documentará todo lo que se acuerde en las reuniones con el cliente referente al desarrollo de la aplicación. El cliente establece la prioridad de cada HU de acuerdo a sus necesidades más inmediatas y cuál va a ser el contenido de la primera entrega, los desarrolladores estiman cuanto tiempo y esfuerzo requiere cada una de ellas. Algunos de los artefactos que se generan son el cronograma y el plan de entregas el cual se describe a continuación:

*Tabla 6: Plan de entregas*

| Plan de entregas | | | |
|---|---|---|---|
| **Iteración** | **Entregable** | **Fecha inicio** | **Fecha Fin** |
| **Iteración 1** | Sistema para la digitalización de expedientes v1.0 | 5/10/2016 | 30/1/2017 |
| **Iteración 2** | Sistema para la digitalización de expedientes v2.0 | 16/2/2017 | 30/3/2017 |

**Fuente**: Elaboración Propia

## 2.6 Conclusiones del capítulo

- El empleo de las técnicas para la obtención de requisitos posibilitó comprender, identificar y describir las exigencias funcionales y no funcionales que deberá cumplir el sistema. Se identificaron un total de 24 requisitos a tener en cuenta, los cuales se describieron a través de las HU.
- Los artefactos obtenidos en la fase de planificación y diseño son de gran importancia para la construcción del sistema, facilitando una definición en detalle para permitir su interpretación e implementación.
- La utilización de patrones de diseño posibilitó modelar una solución teniendo como objetivo la reutilización de código y la solución a problemas en contextos similares durante el desarrollo de software.

# CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

## 3.1 Introducción

En el presente capítulo se describe el estilo de codificación utilizado, se presenta el diagrama de despliegue y se realizan las pruebas al sistema. Se presentan los casos de prueba creados para validar las funcionalidades que responden a los requisitos identificados.

## 3.2 Implementación

Se seleccionan las HU para ser implementadas durante el transcurso de la iteración a la que pertenecen. Por estas razones, se lleva a cabo una revisión del plan de iteraciones y se modifican en caso de ser necesario. La aplicación que se desarrolle debe tener la calidad requerida para su uso y cumplir con los requisitos de software determinados en el segundo capítulo.

### 3.2.1 Estándares de codificación

Comprende todos los aspectos de la generación del código. Contribuyen al entendimiento del equipo de trabajo, limpieza del código y actividades de mantenimiento. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación permiten generar un código de alta calidad permitiendo al equipo de desarrollo efectuar mejor las revisiones posteriores. Ejemplo de estándares aplicados en el proyecto:

- Rompiendo líneas
    - Romper después de una coma.
    - Romper antes de un operador.
- Longitud de la línea : Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.
- Convenciones de nombres
    - El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel, actualmente com, edu, gov, mil, net, org, o uno de los códigos ingleses de dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981.
- Declaraciones
    - Cantidad por línea: Se recomienda una declaración por línea, ya que facilita los comentarios.

- Inicialización: Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

```
public class ConnectionManager {

    private static ConnectionManager instance;

    private FTPClient ftpClient;
    private String host = "";
    private String username = "";
    private String password = "";
    private String home_path = "";
    private boolean isconfigured = false;

    public ConnectionManager() {...3 lines }

    public static ConnectionManager getInstance()

    public boolean isConnected() {...3 lines }

    public FTPClient getFtpClient() {...3 lines }
```

```
public boolean init_connection() {...6 lines }

public void SetVariables(String host, String usernam
        , String password, String home_path) {
    this.host = host;
    this.username = (username != null &&
            !username.isEmpty() ? username : "anonym
    this.password = (password != null &&
            !password.isEmpty() ? password : "anonym
    this.home_path = (home_path != null &&
            !home_path.isEmpty() ? home_path : "/");
    isconfigured = true;
}
```

*Ilustración 14:Uso de los estándares de codificación*

**Fuente**: Elaboración Propia

### 3.2.2 Diagrama de despliegue

En el diagrama de despliegue se muestra cómo y dónde se despliega el sistema. Los elementos de diseño a nivel de despliegue indican como se ubicarán estos dentro del entorno computacional físico que soportará al software, los mismos pueden ser nodos, componentes y asociaciones (Roger S. Pressman 2002a).

Como la información manejada por la aplicación es bastante sensible se tomaron medidas para proteger la mismas de ataques informáticos. Para la sincronización con el servidor remoto se utilizó un servidor de protocolo seguro de transferencia de archivos (SFTP [6]), el cual utiliza una encriptación de los datos mediante una capa de conexión segura (SSL[7]) lo que aumenta la seguridad al ser transferidos desde el equipo local al remoto.

[6] Protocolo Seguro de Transferencia de Archivos (*Secure File Transfer Protocol*, por sus siglas en inglés SFTP): Es un programa que se ejecuta en un equipo servidor cuya función es permitir el intercambio de datos entre diferentes servidores y ordenadores («SERVIDOR FTP :: CONCEPTOS Y DESCRIPCIONES BÁSICOS» 2019)

[7] Capa de Conexión Segura *(Secure Socket Layer, por sus siglas en inglés* SSL ): Es un protocolo criptográfico que proporciona conexiones seguras en una red (Rescorla <ekr@networkresonance.com> 2019).


*Ilustración 15: Diagrama de despliegue*

**Fuente**: Elaboración Propia

### 3.2.3 Tratamiento de errores

Para que la aplicación funcionara adecuadamente frente a cualquier situación que se presentase ya sea por errores introducidos por el usuario, por el propio sistema o debido a factores externos, fue necesario tener un completo control sobre todos los posibles errores a ocurrir. Se utiliza para una mayor seguridad y confiabilidad de los usuarios con el sistema. Estos errores son controlados mediante excepciones, cada vez que se lanza una, el usuario inmediatamente es notificado de dicho error y de las posibles causas que generan el mismo. Permite manejarlas y tomar decisiones para que no se comprometan la integridad del sistema ni de los datos.

## 3.3 Validación del diseño

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas, de manera que se puedan desarrollar los remedios y mejorar el proceso del software (Roger S. Pressman 2010a).

### 3.3.1 Tamaño operacional de clases (TOC)

Las métricas basadas en el tamaño operacional de las clases se enfocan en contar la cantidad de atributos y operaciones para una clase individual y los valores promedios para el sistema orientado a objetos en su totalidad. El tamaño de clase puede ser determinada usando las siguientes medidas (Pressman, 2010):

- El número total de operaciones (las operaciones privadas de la clase, así como las heredadas) que están encapsuladas en ella.
- El número de atributos (ya sean los atributos heredados como los propios) que están encapsulados dentro de la clase.

Para medir la responsabilidad, complejidad de implementación y la reutilización se definieron los atributos de calidad y umbrales que se muestran en la Tabla 7 y Tabla 8 respectivamente.

*Tabla 7: Atributos de calidad evaluados por la métrica TOC.*

| Atributo de calidad | 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 incremento del TOC implica una disminución del grado de reutilización de la clase. |

**Fuente**: Elaboración Propia

*Tabla 8: Criterios de evaluación de la métrica TOC*

| Atributo | Categoría | Criterio |
|---|---|---|
| **Responsabilidad** | Baja | < =Prom. |
| | 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. |

**Fuente**: Elaboración Propia

#### 3.3.1.1 Resultados obtenidos de la aplicación de la métrica TOC

El siguiente gráfico refleja que la mayoría de las clases tienen de 1 a 5 procedimientos. Este resultado demuestra que el funcionamiento general del sistema está distribuido equitativamente entre la mayoría de las clases, de esta forma se garantiza que, en caso de ocurrir algún cambio en el sistema de clases del componente, estaría menos comprometido el funcionamiento correcto del mismo.

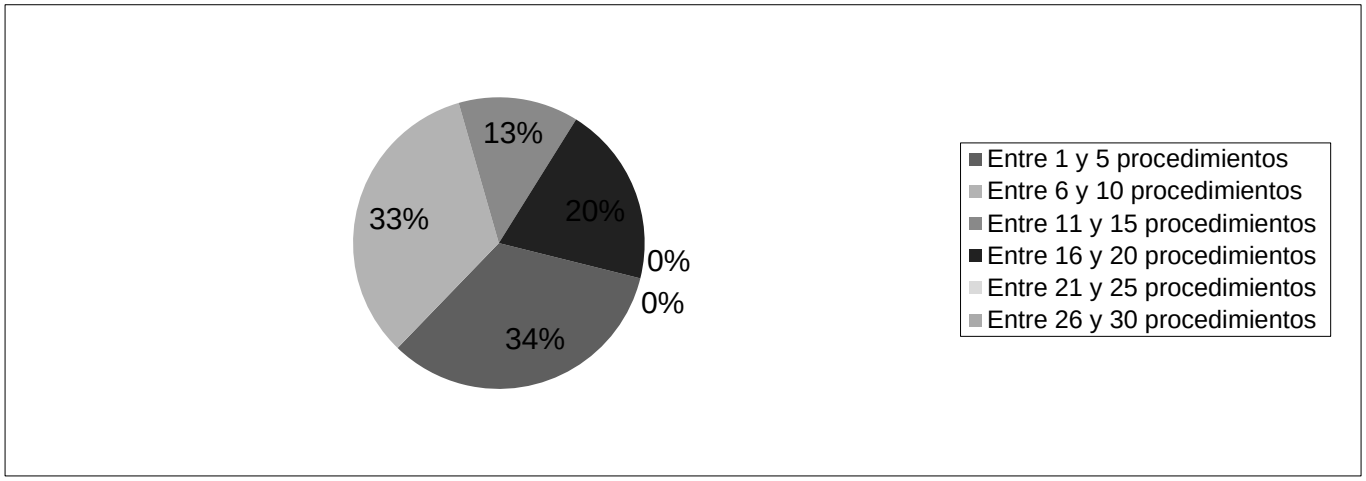



*Ilustración 16: Resultados gráficos obtenidos de la aplicación de la métrica TOC*

**Fuente**: Elaboración Propia

A continuación, se muestra la representación de la incidencia de los resultados de la evaluación de la métrica TOC en los diferentes atributos.

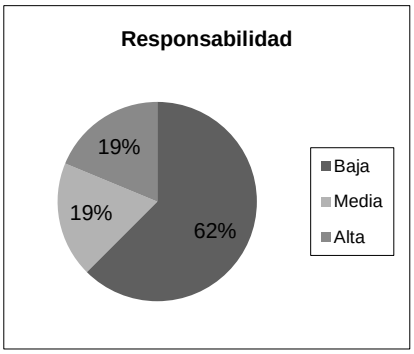



*Ilustración 17: Atributo responsabilidad con la métrica TOC*

**Fuente**: Elaboración Propia

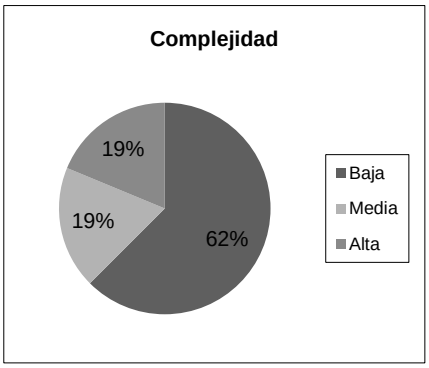



*Ilustración 18: Atributo Complejidad con la métrica TOC*

**Fuente**: Elaboración Propia

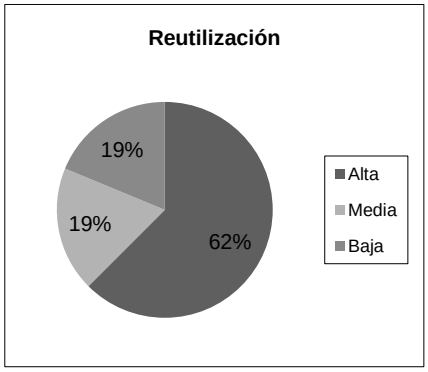



*Ilustración 19: Atributo Reutilización con la métrica TOC.*

**Fuente**: Elaboración Propia

Haciendo un análisis de los resultados obtenidos en la evaluación de la herramienta de medición de la métrica TOC, se puede concluir que el diseño del sistema tiene una calidad aceptable teniendo en cuenta que el 62% de las clases incluidas en este sistema poseen una alta reutilización. Además, este mismo por ciento de clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad y Complejidad de Implementación) en el diseño propuesto, evidenciando que las clases no tienen tanta responsabilidad, no son tan complejas, y poseen un elevado grado de reutilización. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

### 3.3.2 Relaciones entre clases (RC)

Las relaciones entre las clases están dadas por el número de relaciones de uso que tenga una clase con otras, o sea el número de dependencias que una clase tiene con otra. Mediante la cual se calcula el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas a fin de inspeccionar la efectividad del diseño, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Para medir el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas se definieron los atributos de calidad y umbrales que se muestran en la Tabla 9 y Tabla 10 respectivamente.

*Tabla 9: . Atributos de calidad evaluados por la métrica RC.*

| Atributo de calidad | Modo en que lo afecta |
|---|---|
| **Acoplamiento** | Un aumento del RC implica un aumento del Acoplamiento de la clase. |
| **Complejidad de mantenimiento** | Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase. |
| **Reutilización** | Un aumento del RC implica una disminución en el grado de reutilización de la clase. |
| **Cantidad de pruebas** | Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase. |

**Fuente**: Elaboración Propia

*Tabla 10: Criterios de evaluación para la métrica RC.*

| Atributo | Categoría | Criterio |
|---|---|---|
| **Acoplamiento** | Ninguno | 0 |
| | Bajo | 1 |
| | Medio | 2 |
| | Alto | > 2 |
| **Complejidad de mantenimiento** | Baja | < =Prom. |
| | Media | Entre Prom y 2* Prom. |
| | Alta | > 2* Prom. |
| **Reutilización** | Baja | > 2*Prom. |
| | Media | Entre Prom y 2* Prom |
| | Alta | <= Prom. |
| **Cantidad de pruebas** | Baja | < =Prom. |
| | Media | Entre Prom y 2* Prom. |
| | Alta | > 2* Prom. |

**Fuente**: Elaboración Propia

#### 3.3.2.1 Resultados obtenidos de la aplicación de la métrica RC

La gráfica muestra la representación de los resultados obtenidos agrupados en los intervalos definidos. Después de un análisis se decidió que no es posible eliminar estas dependencias por la responsabilidad arquitectónica que deben mantener las clases.

A continuación, se muestra la representación en porciento de los resultados obtenidos en la herramienta agrupados en los intervalos definidos.

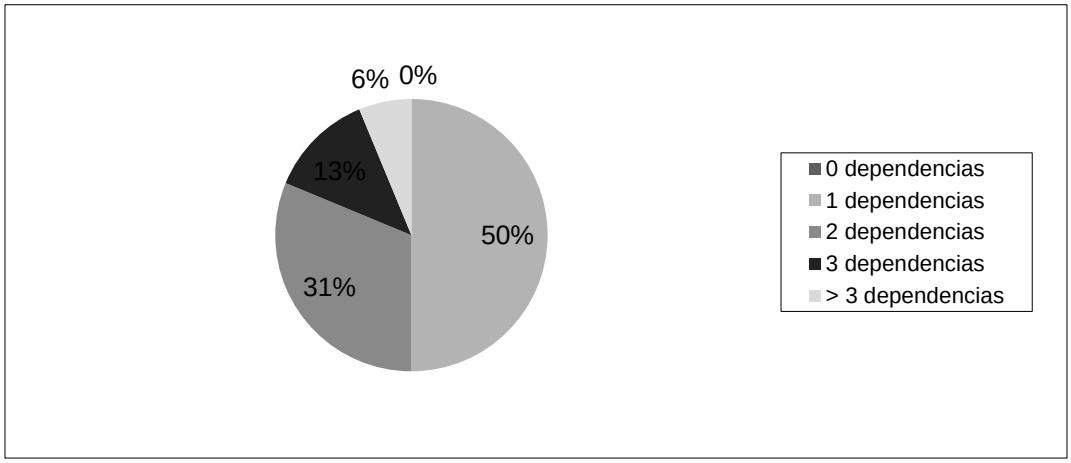



*Ilustración 20: Representación en % de la evaluación de la métrica RC*

**Fuente**: Elaboración Propia

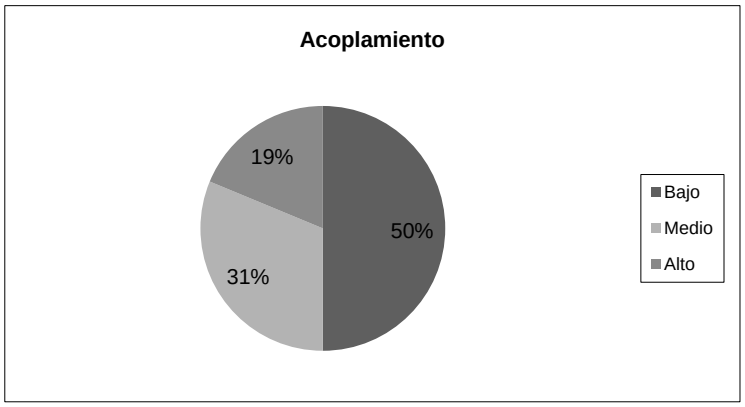



*Ilustración 21:Atributo acoplamiento con la métrica RC*

**Fuente**: Elaboración Propia

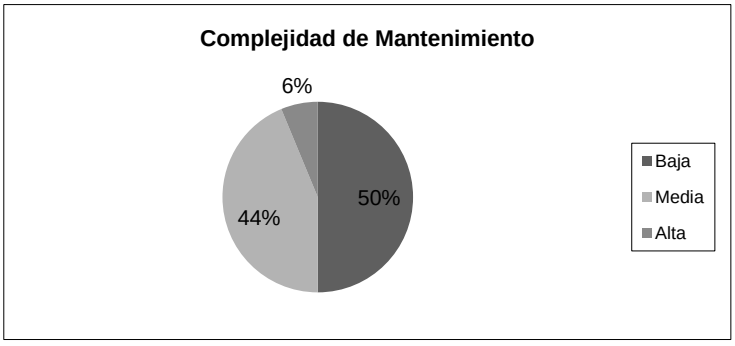



*Ilustración 22: Atributo complejidad de mantenimiento con la métrica RC*

**Fuente**: Elaboración Propia

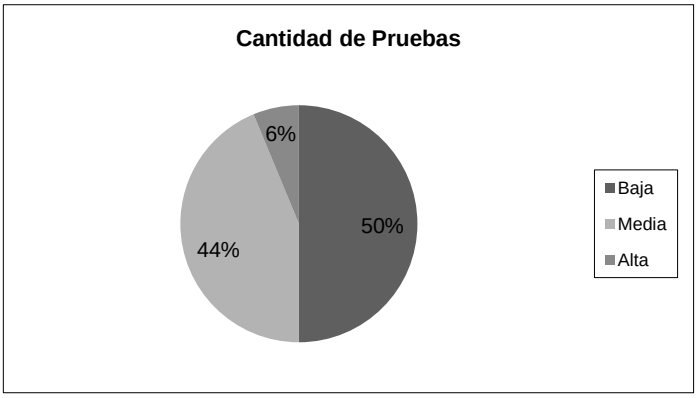



*Ilustración 23: Representación de la evaluación de la métrica RC en el atributo Cantidad de pruebas.*

**Fuente**: Elaboración Propia

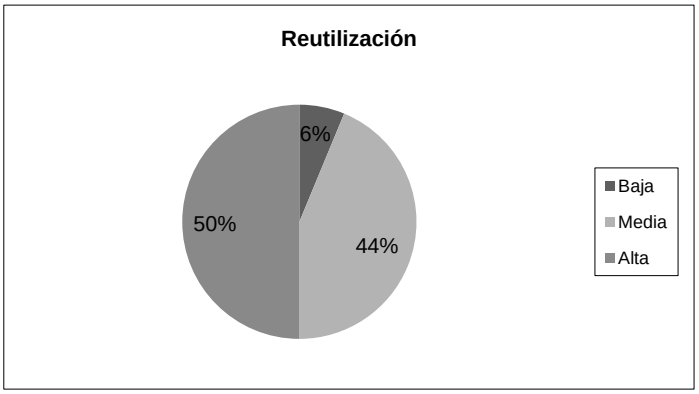



*Ilustración 24: Representación de la evaluación de la métrica RC en el atributo Reutilización*

**Fuente**: Elaboración Propia

Haciendo un análisis de los resultados obtenidos en la evaluación de la herramienta de medición de la métrica RC, se puede concluir que el diseño del sistema está entre los niveles de calidad requeridos, pudiéndose observar que el 50% de las clases posee una dependencia. Igualmente, el 50% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de mantenimiento, Reutilización y Cantidad de pruebas se comportan satisfactoriamente con un 50% de valor en las clases. Confirmando la elevada reutilización, el bajo acoplamiento, la baja complejidad y la baja cantidad de pruebas que se necesitan realizar.

## 3.4 Pruebas

Las pruebas de software (en inglés software testing) intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. El proceso de prueba tiene dos metas definidas: demostrar al desarrollador y al cliente que el software cumple con los

requerimientos, y encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación (Ian Sommerville 2011a).

Por lo general un sistema debe pasar por tres etapas de pruebas (Ian Sommerville 2011a):

1. Pruebas de desarrollo: donde el sistema se pone a prueba durante el proceso para descubrir errores y defectos.
2. Versiones de pruebas: donde un equipo de prueba por separado experimenta una versión completa del sistema antes de presentárselo al usuario final.
3. Pruebas de usuario: donde los usuarios reales o potenciales del sistema prueban al sistema en su propio entorno.

Las pruebas de software son una actividad más en el proceso de control de calidad. A continuación, se evidencian las pruebas aplicadas al sistema desarrollado.

### 3.4.1 Pruebas de caja blanca

La prueba de caja blanca del software se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles (Roger S. Pressman 2010a).

#### 3.4.1.1 Técnica de caminos básicos

La técnica de camino básico permite al equipo de desarrollo obtener una medida de la complejidad lógica del código implementado. Posteriormente el uso de esta medida servirá de guía para la definición de un conjunto de caminos independientes de ejecución, lo que garantiza que durante la prueba se ejecuten por lo menos una vez, cada sentencia del sistema desarrollado (Ingeniería de sistemas y computación 2018).

La complejidad ciclomática es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa (Ingeniería de sistemas y computación 2018).

A continuación, se muestra los pasos para realizar la técnica de camino básico:

```
85      private boolean ValidateName() {
86          if (nametext.getText().length() < 200) {
87              for (int i = 0; i < nametext.getLength(); i++) {
88                  if (nametext.getText().charAt(i) == '/'
89                          || nametext.getText().charAt(i) == '\\'
90                          || nametext.getText().charAt(i) == '<'
91                          || nametext.getText().charAt(i) == '>'
92                          || nametext.getText().charAt(i) == '*'
93                          || nametext.getText().charAt(i) == '\"'
94                          || nametext.getText().charAt(i) == '?'
95                          || nametext.getText().charAt(i) == ':'
96                          || nametext.getText().charAt(i) == '|') {
97                      return false;
98                  }
99              }
100             return true;
101         }
102         return false;
103     }
```

*Ilustración 25: Método ValidateName para técnica de Camino Básico*

**Fuente**: Elaboración Propia

1. Confeccionar el grafo de flujo: usando el método ValidateName, se realizó la representación del grafo del flujo.

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

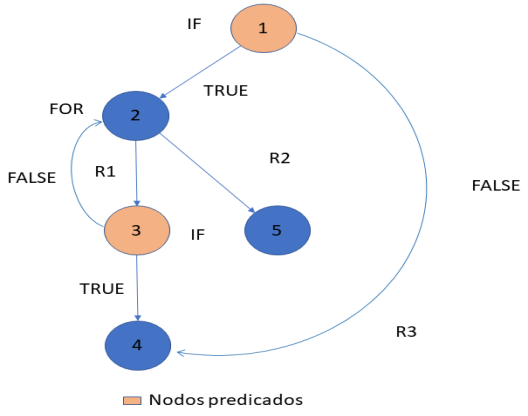



*Ilustración 26: Grafo de camino básico del método Validatename*

**Fuente**: Elaboración Propia

Después de haber realizado el grafo, se calcula la complejidad ciclomática por tres fórmulas distintas, las cuales deben dar el mismo resultado para comprobar que el cálculo sea correcto.

2. Calcular la complejidad ciclomática: proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, la complejidad ciclomática se calculó con las siguientes fórmulas:

- **V (G)= A-N+2**, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
  **V (G) = 6-5+2=3**
- **V (G) = Regiones**
  **V (G) = 3**
- **V (G) = P+ 1** , donde P son los nodos predicados.
  **V (G) =2+1=3**

3. Determinar un conjunto básico de caminos linealmente independientes: el valor de V(G) da el número de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes cinco caminos:

- Camino básico #1: 1-4
- Camino básico #2: 1-2-3-4
- Camino básico #3: 1-2-3-2-5

4. Una vez determinados los caminos independientes se debe diseñar casos de prueba para cada uno de esos caminos. Luego de ejecutados esos casos de prueba se comprueba que cada una de las sentencias es ejecutada al menos una vez y funcionan bien.

*Tabla 11: Caso de Prueba CB# 1y 2*

| **Caso de Prueba Camino Básico 1 y 2** | |
|---|---|
| Objetivo de la prueba | Comprobar si los datos introducidos por el usuario están escritos correctamente. |
| Datos de entrada | Datos introducidos por el usuario. |
| Procedimientos | Cuando se presiona la opción de guardar luego de digitalizar un documento se procede a nombrar el archivo digital y junto con esto se validan los datos introducidos por el usuario. |
| Salida esperada | No se le permite al usuario escribir estos datos y se muestra un mensaje. |

**Fuente**: Elaboración Propia

*Tabla 12: Caso de Prueba CB#3*

| Caso de Prueba Camino Básico 3 | |
|---|---|
| Objetivo de la prueba | Comprobar si los datos introducidos por el usuario están escritos correctamente. |
| Datos de entrada | Datos introducidos por el usuario. |
| Procedimientos | Cuando se presiona la opción de guardar luego de digitalizar un documento se procede a nombrar el archivo digital y junto con esto se validan los datos introducidos por el usuario. |
| Salida esperada | Se guardan los datos introducidos por el usuario |

**Fuente**: Elaboración Propia

### 3.4.2 Pruebas de caja negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo en la interfaz del software. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software (Roger S. Pressman 2010a).

Conociendo una función específica para la que fue diseñado el producto, se pueden diseñar pruebas que demuestren que dicha función está bien realizada. Dichas pruebas son llevadas a cabo sobre la interfaz del software, es decir, de la función, proporcionando unas entradas y estudiando las salidas para ver si concuerdan con las esperadas. (Ver Anexo 25)

#### 3.4.2.1 Técnica Partición Equivalente

Esta técnica divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El método se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada (Pressman, 2010).

Para aplicar esta técnica, se deben primeramente realizar los Diseños de casos prueba (DCP) con el objetivo de obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Los DCP son un conjunto de condiciones o variables bajo las cuales un analista determinará si una aplicación o una característica de esta es parcial o completamente satisfactoria (Pressman, 2010).

Como primer paso en el DCP, se muestra a continuación la descripción de las variables para el caso del requisito “Configurar conexión”.

*Tabla 13: Descripción de las variables para el RF- Configurar conexión*

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| **1** | Dirección del servidor | Campo de texto | No | Campo de texto que puede tener números o letras |
| **2** | Nombre de usuario | Campo de texto | Si | Campo de texto que solo se permite valores alfanuméricos |
| **3** | Contraseña | Campo de texto | Si | Campo de texto que posee valores alfanuméricos |
| **4** | Ruta de acceso principal | Campo de texto | Si | Campo de texto que puede tener letras o ser solo un símbolo (/) |

**Fuente:** Elaboración propia

Luego de obtener la descripción de cada una de estas variables, se procede a realizar cada uno de los escenarios a probar en el sistema. Para el caso del requisito en cuestión, se muestran estos escenarios.

*Tabla 14: Diseño de caso de prueba para el RF- Configurar conexión*

| Escenario | Descripción | Dirección del servidor | Nombre de usuario | Contraseña | Ruta de acceso remoto | Respuesta del sistema |
|---|---|---|---|---|---|---|
| **EC 1.1 Configuración correcta** | El usuario selecciona la opción configuración rellena los datos necesarios de forma correcta | V | V | V | V | El sistema guarda los datos de configuración |
| | | 127.0.0.1 | hmartinez | asdasda123*** | home | |
| **EC 1.2 Configuración incorrecta** | El usuario selecciona la opción configuración rellena los datos necesarios de forma incorrecta | I | V | V | V | El sistema te indica que hay un campo obligatorio. |
| | | | aahurtado | fdgdfgdf8**d | | |
| | | V | I | V | V | El sistema debe mostrar el mensaje "Solo se permiten valores alfanuméricos". |
| | | localhost | &^%$#%#24 | dato | / | |
| | | V | V | I | V | |
| | | dato | hmartinez | N/A | home | |
| | | V | V | V | I | |
| | | dato | hmartinez | djs*852 | N/A | |

**Fuente:** Elaboración propia

Para aplicar los DCP a la solución, se efectuaron un total de 2 iteraciones para poder alcanzar resultados satisfactorios atendiendo al correcto comportamiento del sistema ante diferentes situaciones. En la figura que se muestra a continuación, se realiza un resumen mediante una gráfica con el número total de No conformidades (NC) por iteración.


*Ilustración 27: Resultado de las pruebas de Caja Negra*

**Fuente**: Elaboración Propia

### 3.4.3 Pruebas de aceptación

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. En las pruebas de aceptación, la ejecución y aprobación final corresponden al usuario o cliente, que es el que valida y verifica que el alcance es el correcto (Ian Sommerville 2011a). (Ver Anexo 24)

## 3.5 Validación de las variables de la investigación

Partiendo del punto de que el problema a resolver definido por los autores de la presente investigación es: ¿Cómo agilizar el proceso de digitalización de los expedientes en la Secretaría General de la Universidad garantizándose la calidad de la información generada en esta área?

Para analizar la relación causa efecto entre la variable independiente y la variable dependiente, a continuación, se explicarán cómo se resolvieron las deficiencias identificadas en la situación problemática. Para ello se realiza un análisis de cómo se realizaba el proceso antes del desarrollo del sistema y después de la utilización del sistema.

**Antes**

En la secretaría el proceso de digitalización se realiza con el uso de un scanner y una computadora, sin embargo, se presentan problemas que atentan contra el buen desempeño de este proceso:

- Se trabaja con un archivo independiente por cada página que tenga el documento original, lo que dificulta la organización de los documentos que presentan más de una página.
- Existe pérdida de la información. Muchas veces se digitalizan los documentos y no se guardan en los lugares establecidos, perdiéndose en muchos casos el trabajo realizado, por lo que hay que volverlo a hacer.
- Existe duplicidad de la información. Como parte de la no organización en algunos casos, se encuentran digitalizados los mismos documentos dentro de las computadoras en varios lugares y repetidos varias veces.
- Los expedientes deben guardarse por una estructura específica definida por el Ministerio de Educación Superior, la cual igualmente debe respetarse en los expedientes digitales. Este proceso se realiza en la secretaría de forma manual, lo que conlleva un esfuerzo de parte de los trabajadores para lograr cumplir con este requisito.
- Todo lo anteriormente expuesto conlleva a tener en la secretaría personal dedicado a largas jornadas laborales para organizar toda la documentación que se genera como parte del proceso.

**Después**

El uso de un sistema de digitalización para el trabajo con los expedientes en la secretaría general de la Universidad de las Ciencias Informáticas con el fin de gestionar la gran cantidad de documentos y la información que allí se encuentra brinda los siguientes beneficios:

- Todos los documentos de están controlados y centralizados.
- Se evita la pérdida de documentos, ya que están todos seguros en un servidor documental central.
- Se evita la pérdida de tiempo debido a que varias personas tengan que recuperar los documentos desde diferentes lugares y departamentos. Los documentos son inmediatamente recuperados por la persona que está buscando la información.
- Acceso a la información y los documentos de manera segura y estructurada.
- Garantía de seguridad de la información, gracias al sistema de gestión documental los documentos digitales también se archivan de forma segura. Lo primero a tener en cuenta es que los archivos digitales son menos vulnerables que los documentos en papel porque éstos pueden volverse ilegibles o traspapelarse. Un sistema de gestión documental reduce estos

riesgos ya que digitaliza y realiza copias de seguridad. De esta forma, la digitalización de archivos garantiza la seguridad de la información.

- Fácil acceso puesto que nos permite acceder de una manera rápida a la documentación la cual debe guardar un acceso simple, lógico y coherente que nos permita localizarla de una manera rápida y directa.

## 3.6 Conclusiones del capítulo

En este capítulo se abordaron los elementos de la implementación del sistema para la digitalización de los expedientes en la secretaría general de la Universidad de las Ciencias Informáticas, guiado por la metodología de desarrollo AUP-UCI, así como las pruebas realizadas a este sistema y los resultados obtenidos arribando a las siguientes conclusiones:

- El diagrama de despliegue obtenido reflejó las relaciones existentes entre los elementos que intervienen en el proceso de digitalización de los expedientes.
- El correcto uso de los estándares de codificación permitió que el código del sistema desarrollado fuera legible para lograr una fácil comprensión, mayor organización y claridad del mismo.
- El proceso de validación del software arrojó como resultado que el sistema implementado funciona correctamente en correspondencia con las solicitudes del cliente, validado por las actas de liberación y aceptación emitidas.

## CONCLUSIONES GENERALES

En la realización de este trabajo se demostró la necesidad de desarrollar un sistema para la digitalización de los expedientes en la secretaría general de la Universidad de las Ciencias Informáticas, obteniéndose las siguientes conclusiones:

- El estudio de las soluciones existentes demostró que ninguno de estos sistemas daba solución a los problemas identificados por lo que se decide construir un sistema nuevo.
- La selección de lenguajes, tecnologías y herramientas permitió la implementación del sistema de digitalización guiado por la metodología de desarrollo AUP-UCI, obteniéndose un software que satisface las exigencias funcionales y no funcionales definidas por el cliente.
- Los artefactos obtenidos en la etapa de ejecución fueron de gran importancia para la construcción del sistema, facilitando una definición en detalle para permitir su interpretación e implementación.
- El proceso de validación del software arrojó como resultado que el sistema implementado funciona correctamente en correspondencia con las solicitudes del cliente, validado por las actas de liberación y aceptación.

**RECOMENDACIONES**

- Se recomienda el desarrollo de un módulo para gestionar la configuración manual de nuevos escáneres, así como la gestión de controladores personalizados.
- Se recomienda incorporar funcionalidades que permitan añadir una firma digital y gestionar los metadatos de los ficheros creados y realizar Reconocimiento Óptico de Caracteres (OCR).

## REFERENCIAS BIBLIOGRÁFICAS

ABRAHAM SILBERSCHATZ, 2006. *Fundamentos_de_Bases_de_Datos* [en línea]. 5ta. S.l.: s.n. [Consulta: 19 mayo 2019]. ISBN 84-481-4644-1. Disponible en: http://mateo.pbworks.com/w/file/fetch/122276985/Fundamentos_de_Bases_de_Datos_5a_Ed.-_Si.pdf.

ALEX UHU COLLI, 03:50:06 UTC. Modelo vista controlador vs Programación por n capas. [en línea]. Software. S.l. [Consulta: 1 noviembre 2018]. Disponible en: https://es.slideshare.net/alejandrouhu/mvc-vs-pnc-1-revaluacion.

Alfresco. *Alfresco* [en línea], 2018. [Consulta: 1 noviembre 2018]. Disponible en: https://www.itop.es/alfresco.html.

AMADOR DURÁN TORO y BEATRIZ BERNÁNDEZ JIMÉNEZ, 2000. *Metodología para la Elicitación de Requisitos de Sistemas Software.* 2000. S.l.: Universidad de Sevilla.

ANDRÉS NAVARRO CADAVID, JUAN DANIEL FERNÁNDEZ MARTÍNEZ y JONATHAN MORALES VÉLEZ, 2013. *Revisión de metodologías ágiles para el desarrollo de Software*. 6 abril 2013. S.l.: s.n.

Apache Commons Net – Overview. [en línea], [sin fecha]. [Consulta: 20 junio 2018]. Disponible en: https://commons.apache.org/proper/commons-net/.

Asprise Java Scanning and Imaging SDK for Java applets, web applications, Swing/JavaFX components, JEE enterprise applications - an API library scan images into files or memory from scanners on Windows and Mac OS X. [en línea], [sin fecha]. [Consulta: 20 junio 2018]. Disponible en: http://asprise.com/document-scanner-image-pdf/java-scanning-api-overview.html.

Atento. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: http://atento.com/es/.

CAMACHO, A.L., 2015. *SIGELITE DESKTOP: HERRAMIENTA DE CAPTURA DE DATOS PARA EL SISTEMA INTEGRADO DE GESTIÓN ESTADÍSTICA EN ENTIDADES CON LIMITACIONES TECNOLÓGICAS*. Tesis de grado. S.l.: UCI.

Creación y Evolución de Sistemas | Marco de Desarrollo de la Junta de Andalucía. *Creación y Evolución de Sistemas | Marco de Desarrollo de la Junta de Andalucía* [en línea], 2018. [Consulta: 1 noviembre 2018]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/ingenieria/creacion-y-evolucion-sistemas.

DANIEL RIESCO, 2016. *UML-DiagramaClaseObjeto.pdf* [en línea]. 23 junio 2016. S.l.: s.n. [Consulta: 12 marzo 2019]. Disponible en: http://www.sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf.

DAYANA MENDOZA PEÑA y LIONEL RODOLFO BAQUERO HERNÁNDEZ, 2016. *Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de*

*Casos de Uso* [en línea]. Tesis de grado. Cuba: Universidad de las Ciencias Informáticas. [Consulta: 13 noviembre 2018]. Disponible en: https://www.researchgate.net/publication/305160646.

Definición de Expediente. *Definición ABC* [en línea], 2018. [Consulta: 13 noviembre 2018]. Disponible en: https://www.definicionabc.com/derecho/expediente.php.

Definición de Software propietario » Concepto en Definición ABC. [en línea], 2018. [Consulta: 27 noviembre 2018]. Disponible en: https://www.definicionabc.com/tecnologia/software-propietario.php.

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS, 2018. *Tema 3: Patrones de Asignación de Responsabilidades (GRASP)* [en línea]. 2018. S.l.: Universidad de Sevilla. [Consulta: 23 noviembre 2018]. Disponible en: http://www.lsi.us.es/docencia/get.php?id=906.

Desenvolvimento Ágil. *Thiago Thamiel* [en línea], 2009. [Consulta: 29 noviembre 2018]. Disponible en: https://thiagothamiel.com/2009/07/16/desenvolvimento-agil/.

Diagramas de Clases y Objetos. [en línea], [sin fecha]. [Consulta: 12 marzo 2019]. Disponible en: https://www.osmosislatina.com/lenguajes/uml/clasesob.htm.

Eclipse, herramienta universal – IDE abierto y extensible. [en línea], 2013. [Consulta: 13 noviembre 2018]. Disponible en: http://clave.zintegra.com/tag/ibm/.

El patrón Modelo–vista–controlador (MVC). [en línea], 2018. [Consulta: 1 noviembre 2018]. Disponible en: https://openclassrooms.com/en/courses/4309566-descubre-la-arquitectura-mvc/4942546-el-patron-modelo-vista-controlador-mvc.

El software libre no es software gratuito – Geeky Theory. [en línea], 2018. [Consulta: 27 noviembre 2018]. Disponible en: https://web.archive.org/web/20160624124038/https://geekytheory.com/el-software-libre-no-es-software-gratuito/.

ELDA GONZÁLEZ MESA, 2006a. *La digitalización de documentos, ¿amiga o enemiga?* [en línea]. 2006. S.l.: Biblioteca Nacional de Cuba José Martí. [Consulta: 7 noviembre 2018]. Disponible en: http://revistas.bnjm.cu/index.php/anales/article/view/200. Bibliotecas. Anales de Investigación; núm. 2 (2006): Bibliotecas. Anales de Investigación

ELDA GONZÁLEZ MESA, 2006b. *La digitalización de documentos, ¿amiga o enemiga?* [en línea]. 2006. S.l.: Biblioteca Nacional de Cuba José Martí. [Consulta: 7 noviembre 2018]. Disponible en: http://revistas.bnjm.cu/index.php/anales/article/view/200. Bibliotecas. Anales de Investigación; núm. 2 (2006): Bibliotecas. Anales de Investigación

EMILIO A. SÁNCHEZ, PATRICIO LETELIER y JOSÉ H. CANÓS, 2014. *Mejorando la gestión de historias de usuario en eXtreme Programming**. 17 mayo 2014. S.l.: s.n.

FERNANDO BERZAL, 2004. *UML lenguaje de modelado*. 28 octubre 2004. S.l.: s.n.

FIDEL ALEJANDRO CEBRIÁN LA ROSA, J.S.M., 2014. *Plataforma para la construcción de sistemas de digitalización de documentos (DigiPRO)*. Tesis de grado. Cuba: UCI. TD_07335_14.pdf

Freeware OCR Scanner Software Scan2PDF (Scan To PDF). [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: http://www.scantopdf.org/.

*get.pdf* [en línea], 2018. diciembre 2018. S.l.: s.n. [Consulta: 11 diciembre 2018]. Disponible en: http://www.lsi.us.es/docencia/get.php?id=361.

GONZÁLEZ VALDÉS, S., SUÁREZ FONT, R. y LIZAMA MUÉ, Y., 2011. Diseño e implementación de los módulos de Preparación de Documentos, Digitalización de Documentos y Asociación de Metadatos del Centro de Digitalización para la División de Antecedentes Penales. [en línea], [Consulta: 13 noviembre 2018]. Disponible en: http://repositorio.uci.cu/jspui/handle/ident/TD_04302_11.

GREG BROWN, 2011. Introducing FXML. A Markup Language for JavaFX. ,

GRUPO DE IDENTIFICACIÓN Y VALORACIÓN DE SERIES DE LOS ARCHIVOS UNIVERSITARIOS, 2001. *Propuesta abreviada de identificación y valoración de series* [en línea]. junio 2001. S.l.: Conferencia de Archiveros de Universidades. [Consulta: 29 noviembre 2018]. Disponible en: http://cau.crue.org/wp-content/uploads/pabrevEXPALUMNOS.pdf.

Herramientas de ingeniería de Software para desarrollo y modelado de software. [en línea], 2018. [Consulta: 13 noviembre 2018]. Disponible en: http://www.sparxsystems.com.ar/platforms/software_development.html.

How to use Model-View-Controller (MVC). [en línea], 2012. [Consulta: 7 noviembre 2018]. Disponible en: https://web.archive.org/web/20120429161935/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

IAN SOMMERVILLE, 2009a. *Sommerville_Parte_III_Diseño.pdf*. 7ma Edición. Madrid (España): Pearson Educación S.A. ISBN 84-7829-074-5.

IAN SOMMERVILLE, 2009b. *Sommerville_Parte_II_Requerimientos.pdf*. 7ma Edición. Madrid(España): Pearson Educación S.A. ISBN 84-7829-074-5.

IAN SOMMERVILLE, 2009c. *Sommerville_Parte_IV_Desarrollo.pdf*. 7ma Edición. Madrid (España): Pearson Educación S.A. ISBN 84-7829-074-5.

IAN SOMMERVILLE, 2009d. *Sommerville_Parte_I_Visión_General.pdf*. 7ma Edición. Madrid (España): Pearson Educación S.A. ISBN 84-7829-074-5.

IAN SOMMERVILLE, 2009e. *Sommerville_Parte_VI_Gestión_del_Personal.pdf*. 7ma Edición. Madrid (España): Pearson Educación S.A. ISBN 84-7829-074-5.

IAN SOMMERVILLE, 2009f. *Sommerville_Parte_V_Verificación_y_Validación.pdf*. 7ma Edición. Madrid (España): Pearson Educación S.A. ISBN 84-7829-074-5.

IAN SOMMERVILLE, 2011a. *Ingeniería de software*. 9na Edición. méxico: Pearson Educación S.A. ISBN 978-607-32-0603-7.

IAN SOMMERVILLE, 2011b. *Sotware Engineering*. 8va Edición. China: Pearson Educación S.A. ISBN 978-0-321-31379-9.

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN, 2013. Técnicas de Pruebas de Software. *Universidad del Valle*, pp. 61.

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN, 2018. *2013A_TPSW_Clase05_CajaBlanca.pdf* [en línea]. 2018. S.l.: s.n. [Consulta: 11 diciembre 2018]. Disponible en: https://campusvirtual.univalle.edu.co/moodle/pluginfile.php/480081/mod_resource/content/1/2013A_TPSW_Clase05_CajaBlanca.pdf.

Ingenieria del software: prueba de la caja blanca y camino básico. [en línea], [sin fecha]. [Consulta: 11 diciembre 2018]. Disponible en: https://www.monografias.com/docs113/ingenieria-software-prueba-caja-blanca-y-camino-basico/ingenieria-software-prueba-caja-blanca-y-camino-basico.shtml.

iText 5.5.5 | iText. [en línea], 2015. [Consulta: 14 noviembre 2018]. Disponible en: https://itextpdf.com/changelog/555.

IT-MENTOR ITM, 2009. PruebasSoftware.pdf. [en línea]. [Consulta: 11 diciembre 2018]. Disponible en: http://materias.fi.uba.ar/7548/PruebasSoftware.pdf.

JACINTO CABRERA RODRÍGUEZ, 07:03:03 UTC. ENTORNOS DE DESARROLLO: LENGUAJES DE MODELADO. [en línea]. Software. S.l. [Consulta: 1 noviembre 2018]. Disponible en: https://es.slideshare.net/JacintoCabreraRodrguez/entornos-de-desarrollo-lenguajes-de-modelado.

jai-imageio-core Java documentation Version 1.4.0. [en línea], [sin fecha]. [Consulta: 20 junio 2018]. Disponible en: https://jar-download.com/java-documentation-javadoc.php?a=jai-imageio-core&g=com.github.jai-imageio&v=1.4.0.

JavaXT - Open Source Extensions to the Core Java Library. [en línea], [sin fecha]. [Consulta: 20 junio 2018]. Disponible en: http://www.javaxt.com/.

JFoenix. [en línea], [sin fecha]. [Consulta: 20 junio 2018]. Disponible en: http://www.jfoenix.com/documentation.html.

JOSÉ H. CANÓS, PATRICIO LETELIER y MARIA DEL CARMEN PANADÉS, 2018. *Metodologías ágiles en el desarrollo de software*. 2018. S.l.: s.n.

JOSE M. BATISTA, 2009. *PROGRAMACIÓN EXTREMA (XP) EXTREME PROGRAMMING (XP)*. 24 septiembre 2009. S.l.: s.n.

JULIÁN PÉREZ PORTO y MARIA MERINO, 2012. Definición de expediente — Definición.de. *Definición.de* [en línea]. [Consulta: 13 noviembre 2018]. Disponible en: https://definicion.de/expediente/.

KEN SCHWABER, 2004. *¿Qué_es_scrum?* 2004. S.l.: Microsoft Press. Agile Project Management with Scrum

KENT BECK, 1999. *Extreme Programming Explained : Embrace Change*. AUSTELL, GA, Estados Unidos de America: Addison Wesley Professional. ISBN 0-201-61641-6.

Kofax Express | Kofax. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: https://www.kofax.com/Products/Information%20Capture/Express/Features.

LA ROSA MONTES, 2007. *Análisis y Modelado del sistema para el proceso de digitalización de documentos de Registros y Notarias del MPPRIJ de la República Bolivariana de Venezuela*. 2007. S.l.: s.n.

LARMAN, C. y MOROS VALLE, B., 2010. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Madrid: Pearson Education. ISBN 978-84-205-3438-1.

LEANDRO ALEGSA, 2010. Definición de Herramienta de modelado. [en línea]. [Consulta: 13 noviembre 2018]. Disponible en: http://www.alegsa.com.ar/Dic/herramienta_de_modelado.php.

Lenguajes de programación. *Enciclopedia* [en línea], 2017. [Consulta: 13 noviembre 2018]. Disponible en: https://es.ccm.net/contents/304-lenguajes-de-programacion.

LETELIER, P. y PENADÉS, M.C., 2006. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *www.cyta.com.ar/ta0502/v5n2a1.htm* [en línea]. [Consulta: 1 noviembre 2018]. Disponible en: http://www.cyta.com.ar/ta0502/v5n2a1.htm.

LETELIER PENADES, 2018. Ciclo de vida de un proyecto XP. Capítulo 5. Metodología. *Ciclo de vida de un proyecto XP* [en línea]. [Consulta: 1 noviembre 2018]. Disponible en: http://oness.sourceforge.net/proyecto/html/ch05s02.html.

LLUIS CODINA, 1993. Qué es un sistema de gestión documental. *Revista internacional cientifica y profesional*, ISSN 1386-6710.

LUISA TOLOSA ROBLEDO, V.G.C., 2008. *LA DIGITALIZACIÓN EN ARCHIVOS*. 13 noviembre 2008. S.l.: s.n.

MAIA KORD, 2011. Patrones GRASP. Patrones GoF. Diferencia entre GRASP y GoF - TuxNots. [en línea]. [Consulta: 23 noviembre 2018]. Disponible en: http://tuxnots.com.ar/materias-de-la-facu/metodologia-de-sistemas/patronesgrasppatronesgofdiferenciaentregraspygof.

MARÍA PAULA IZAURRALDE, 2013. *Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario*. Tesis de postgrado. S.l.: Universidad Tecnológica Nacional Facultad Regional Córdoba.

MARTÍNEZ, A.G., 2013. *Digitalización de Documentos*. Tesis de grado. Cuba: Universidad de las Ciencias Informáticas.

Metodología XP Programación Extrema (Metodología ágil). *Diego Calvo* [en línea], 2018. [Consulta: 13 noviembre 2018]. Disponible en: http://www.diegocalvo.es/metodologia-xp-programacion-extrema-metodologia-agil/.

MIGUEL ANGEL MORENO MARTIN, 2010. *La ingeniería del software*. 28 noviembre 2010. S.l.: s.n.

Modelo – Vista – Controlador (MVC) para tus Aplicaciones JAVA. *INFORUX* [en línea], 2008. [Consulta: 1 noviembre 2018]. Disponible en: https://inforux.wordpress.com/2008/08/02/modelo-vista-controlador-mvc-para-tus-aplicaciones-java/.

Modelo de Dominio. *Tecnología y Synergix* [en línea], 2008. [Consulta: 14 noviembre 2018]. Disponible en: https://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

MUJICA, M.M. y HERRERA, M.E.P., 2005a. *Gestión documental y organización de archivos* [en línea]. S.l.: Editorial Félix Varela. [Consulta: 9 noviembre 2018]. ISBN 978-959-258-950-6. Disponible en: https://books.google.com.cu/books?id=MGuanQAACAAJ.

MUJICA, M.M. y HERRERA, M.E.P., 2005b. *Gestión documental y organización de archivos* [en línea]. S.l.: Editorial Félix Varela. [Consulta: 15 octubre 2018]. ISBN 978-959-258-950-6. Disponible en: https://books.google.com.cu/books?id=MGuanQAACAAJ.

NADER, J.R., 2014. Metodología de Desarrollo de Software: MBM (Metodología Basada en Modelos) Software Development Methodology: MBM (Methodology Based in Models). , no. 16, pp. 15. ISSN 1909-2458.

Nuxeo - Athento. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: http://www.athento.com/es/nuxeo/.

Obtención de Requerimientos. Técnicas y Estrategia. *SG Buzz* [en línea], 2018. [Consulta: 14 noviembre 2018]. Disponible en: https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

OSCAR PINZON, 2017. *Ingeniería Web: Una Metodología para el Desarrollo de Aplicaciones Web Escalables y Sostenibles*. 1 febrero 2017. S.l.: s.n.

Plataforma de servicios de contenido (ECM, DAM, Gestión de casos) | Nuxeo. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: https://www.nuxeo.com/es/.

Productos de Nuxeo | Nuxeo. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: https://www.nuxeo.com/es/products/.

¿Qué es el lenguaje de programación JAVA? - Base de Conocimientos - ICTEA. [en línea], 2018. [Consulta: 13 noviembre 2018]. Disponible en: http://www.ictea.com/cs/knowledgebase.php?action=displayarticle&id=8790.

¿Qué es el software libre? - Proyecto GNU - Free Software Foundation. [en línea], 2018. [Consulta: 27 noviembre 2018]. Disponible en: https://www.gnu.org/philosophy/free-sw.es.html.

¿Qué es Interfaz de programación de aplicaciones (API) ? - Definición en WhatIs.com. [en línea], 2018. [Consulta: 23 noviembre 2018]. Disponible en: https://searchdatacenter.techtarget.com/es/definicion/Interfaz-de-programacion-de-aplicaciones-API.

¿Qué es un IDE? – DeProgramación. [en línea], 2018. [Consulta: 13 noviembre 2018]. Disponible en: http://deprogramacion.cubava.cu/2016/02/01/que-es-un-ide/.

¿Qué es un sistema de gestión documental? *TIC Portal* [en línea], 2018. [Consulta: 28 noviembre 2018]. Disponible en: https://www.ticportal.es/temas/sistema-gestion-documental/que-es-sistema-gestion-documental.

QUIJANO, J., 2018. ¿Qué pruebas debemos hacerle a nuestro software y para qué? *Genbeta* [en línea]. [Consulta: 11 diciembre 2018]. Disponible en: https://www.genbeta.com/desarrollo/que-pruebas-debemos-hacerle-a-nuestro-software-y-para-que.

RESCORLA <EKR@NETWORKRESONANCE.COM>, E., 2019. The Transport Layer Security (TLS) Protocol Version 1.2. [en línea]. [Consulta: 20 mayo 2019]. Disponible en: https://tools.ietf.org/html/rfc5246.

REYES DUCONGER, C.O., ROSALES MUÑOZ, F., COMPANIONI GUERRA, A. y RODRÍGUEZ MORALES, S., 2011. Solución de software para la Digitalización de Documentos. [en línea], [Consulta: 13 noviembre 2018]. Disponible en: http://repositorio.uci.cu/jspui/handle/ident/TD_05014_11.

ROGER S. PRESSMAN, 2002. *Ingeniería del software. Un enfoque práctico*. 6ta Edición. S.l.: McGraw-Hill. ISBN 978-970-10-5473-4.

ROGER S. PRESSMAN, 2002b. *Ingeniería del software. Un enfoque práctico*. 6ta Edición. S.l.: McGraw-Hill. ISBN 978-970-10-5473-4.

ROGER S. PRESSMAN, 2010a. *Ingeniería del software. Un enfoque práctico*. 7ma Edición. New York: McGraw-Hill. ISBN 978-607-15-0314-5.

ROGER S. PRESSMAN, 2010b. *Software Engineering. A Practitioner's Approach*. 7ma Edición. New York: McGraw-Hill. ISBN 978-0-07-337597-7.

ROJAS, V.L., 2018. ¿Cuáles son las metodologías ágiles y por qué son beneficiosas para tu empresa? *Tentulogo* [en línea]. [Consulta: 29 noviembre 2018]. Disponible en: https://tentulogo.com/cuales-son-las-metodologias-agiles-y-por-que-son-beneficiosas-para-tu-empresa/.

ROMERO, Y.F. y GONZÁLEZ, Y.D., 2012. Patrón Modelo-Vista-Controlador. *Revista dígital delas informáticas ylas comunicaciones*, vol. 11, no. 1, pp. 11. ISSN 1729 - 3804.

SAEZ VILLAVICENCIO, ANISLEY DE LA CARIDAD y ERNESTO LORENTE RODRIGUEZ, 2009. ENTORNO DE DESARROLLO INTEGRADO LIBRE Y MULTIPLATAFORMA PARA DESARROLLAR SOFTWARE EDUCATIVO EN FORMATO MULTIMEDIA FREE

INTEGRATED ENVIRONMENT TO BUILD MULTIMEDIA AND EDUCATIONAL SOFTWARE. *International Conference "Informática 2009*,

Secretaría Administrativa. [en línea], 2018. [Consulta: 19 noviembre 2018]. Disponible en: https://administracion.cinvestav.mx/Secretar%C3%ADaAdministrativa.aspx.

SERVIDOR FTP :: CONCEPTOS Y DESCRIPCIONES BÁSICOS. [en línea], 2019. [Consulta: 20 mayo 2019]. Disponible en: http://servidorftp.es/.

SETA, L.D., 2018. Una introducción a Extreme Programming. *Dos Ideas* [en línea]. [Consulta: 13 noviembre 2018]. Disponible en: https://dosideas.com/noticias/metodologias/822-una-introduccion-a-extreme-programming.

Significado de Secretaría - Qué es, Definición y Concepto. *Que Significado* [en línea], 2016. [Consulta: 19 noviembre 2018]. Disponible en: https://quesignificado.com/secretaria/.

Software de digitalización - Soluciones de Software de Gestión de Documentos > Productos > Gestión Documental. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: https://www.dokmee.com/es-mx/productos/gesti%C3%B3ndocumental.aspx.

Software de Gestión Documental - Athento. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: http://www.athento.com/es/gestion-documental-inteligente/.

SOFTWARE, M.V.C. del patrón de A. de y INICIAL1979, M.V.C.R., 2018. Patrón Modelo Vista Controlador. *EcuRed* [en línea]. [Consulta: 1 noviembre 2018]. Disponible en: https://www.ecured.cu/Patr%C3%B3n_Modelo_Vista_Controlador.

Software Dokmee de Gestión Documental. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: https://www.ikno.com.co/index.php/software/software-dokmee-de-gestion-documental/.

Software Libre frente a Software Propietario. [en línea], 2018. [Consulta: 27 noviembre 2018]. Disponible en: http://www.cobdc.net/programarilliure/software-libre-software-propietario-legislacion-modelos-negocio/.

Software para Captura e Escaneo de Documentos | SoftExpert Captura. [en línea], 2018. [Consulta: 21 noviembre 2018]. Disponible en: https://www.softexpert.com/es/produto/captura-documentos/.

Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. [en línea], 2017. [Consulta: 11 diciembre 2018]. Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

SQLiteStudio. [en línea], 2018. [Consulta: 14 noviembre 2018]. Disponible en: https://sqlitestudio.pl/index.rvt.

STACK OVERFLOW, 2019. *testing-es.pdf* [en línea]. 17 enero 2019. S.l.: Stack Overflow Contributors. [Consulta: 13 marzo 2019]. Disponible en: https://riptutorial.com/Download/testing-es.pdf.

STANDARDS COORDINATING COMMITTEE OF THE COMPUTER SOCIETY OF IEEE, 1990. Software Requirement. *New York*,

TAMARA RODRÍGUEZ SÁNCHEZ, 2015. *Metodología de desarrollo para la actividad productiva de la UCI*. 2015. S.l.: s.n.

TAMAYO, D.G. y BARROSO, M.L.C., 2013. *Módulo de Digitalización para el Sistema Gestión de Documentos Históricos Dexcriba*. Tesis de grado. Cuba: UCI.

TINOCO GÓMEZ, O., ROSALES LÓPEZ, P.P. y SALAS BACALLA, J., 2010. Criterios de selección de metodologías de desarrollo de software. *Industrial Data* [en línea], vol. 13, no. 2. [Consulta: 13 noviembre 2018]. ISSN 1560-9146. Disponible en: http://www.redalyc.org/resumen.oa?id=81619984009.

TOLEDO, Y.M., CURIAUT, A.V. y CASTRO, E.C., 2016. *Sistema para la preservación digital a largo plazo de la información científico-técnica de la Biblioteca de la UCI*. Tesis de grado. Cuba: UCI.

Tutorial SqLite Studio | conestecurso. [en línea], 2015. [Consulta: 14 noviembre 2018]. Disponible en: https://conestecurso.wordpress.com/2015/03/31/tutorial-sqlite-studio/.

VICTOR DANIEL QUIESPE ALDAMA, 2014. *Metodología_XP*. Tesis de grado. S.l.: Universidad de Huanuco.

WEBHTTP://WWW.PRI.JOVENCLUB.CU/JC/SC, P.E., 2018. Programación Extrema (XP). *EcuRed* [en línea]. [Consulta: 13 noviembre 2018]. Disponible en: https://www.ecured.cu/Programaci%C3%B3n_Extrema_(XP).