ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

TESIS DE GRADO

"DISEÑO Y DESARROLLO DE UNA APLICACIÓN WEB PARA DIGITALIZACIÓN DE DOCUMENTOS DESDE MÚLTIPLES PLATAFORMAS, ALMACENAMIENTO CENTRALIZADO DE LAS IMÁGENES CAPTURADAS, E INTEGRACIÓN DE DICHA APLICACIÓN CON UN SISTEMA DE MANEJO DE CONTENIDO"

Previa a la obtención del titulo de:

INGENIERO EN TELEMATICA

PRESENTADA POR:

JUAN CARLOS VIZUETA VILLAVICENCIO

GUAYAQUIL - ECUADOR

2008

TRIBUNAL DE GRADO

PRESIDENTE

Ing. Holger Cevallos Ulloa

DIRECTOR DE TESIS

Ing. Pedro Fabricio Echeverría

MIEMBROS PRINCIPALES

Msc. Christina Abad

Ing. Jessica Astudillo Barahona

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Juan Carlos Vizueta Villavicencio

RESUMEN

La tesis está dividida en seis capítulos. El primer capítulo se enfocará en identificar, definir y descomponer el problema a tratar, indicará cuáles son los antecedentes, causales y efectos del problema. Este capítulo también presentará las herramientas disponibles para una solución del problema y un contraste entre las herramientas que realizan las mismas tareas para sustentar nuestra selección. Cubrirá la selección del componente de manejo de contenido, del ambiente de desarrollo, y de las interfaces de captura de imágenes.

El segundo capítulo abarca la forma en que se resolverá el problema, refiriéndose solo a la parte conceptual, se presentará la estrategia a utilizar para obtener las metas de esta tesis, que son la creación de una aplicación web que permita digitalizar imágenes de documentos desde las plataformas Windows y Linux, almacenar remotamente las imágenes digitalizadas en la base de datos de un servidor y su consecuente administración y visualización, mejorar la organización, el proceso de almacenamiento y la visualización de los documentos e integrar la aplicación como parte de un componente a un CMS (sistema de manejo de contenido por sus siglas en inglés).

El tercer capítulo explicará cómo se pasa del diseño propuesto a una implementación real del sistema y se plantearán los problemas que hayan surgido para conseguirlo. En este capítulo nos enfocaremos en cómo la aplicación realiza la conexión a las interfaces de captura de imágenes en los sistemas operativos de Linux y de Windows, el almacenamiento remoto y la integración con el Sistema de Manejo de Contenido.

En el cuarto capitulo se tratará detalladamente sobre la ejecución, utilización y los requerimientos del proyecto. Así también como los resultados de la aplicación web en los sistemas operativos ya mencionados, y en los principales navegadores de Internet tales como el Internet Explorer y el Mozilla Firefox. Las pruebas se dirigirán especialmente a la medición del tiempo de respuesta, la facilidad de interacción de la aplicación y las seguridades del sistema.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Muchas veces como personas naturales nos encontramos en trámites que necesitan de copias o digitalizaciones de nuestros documentos de identificación, facturas, recibos de pago, etc., como por ejemplo trámites en entidades públicas como el SRI (Servicio de Rentas Internas), en la Comisión de Tránsito, en el Municipio y en muchos casos entidades privadas como por ejemplo en almacenes de venta de electrodomésticos.

Casi siempre es largo y fastidioso este tipo de transacciones, imaginémonos por un momento entonces cual es el volumen de este tipo de transacciones para una empresa. Pensemos en la entrega de facturas (el SRI acepta opcionalmente también imágenes escaneadas de esas facturas) que debe hacer una empresa privada grande al SRI al finalizar un año, es una tarea bastante pesada y larga para hacerla a última hora, especialmente si la empresa privada maneja miles de miles de facturas anualmente. Las imágenes o documentos digitalizados y organizados dentro de una base de datos ayudarían tremendamente al desenvolvimiento en este tipo de tareas. También un documento digitalizado puede dar fe y constancia para otras transacciones, así como lo da una copia de la cédula. Tomando esto en cuenta, un sistema así nos podría servir para mantener registro de copias digitales de documentos en el Registro Civil, e incluso en el Trámite de Tesis

de Graduación o en la Corte Suprema de Justicia donde se manejan tantos documentos.

Esta aplicación de digitalización de documentos almacenada en la base de datos, y agregada como componente del sistema computacional de una empresa es una herramienta que puede marcar la diferencia. Más aún si esta herramienta es multiplataforma (muchas empresas corren Linux hoy en día por economía y muchas veces por estabilidad), ya que puede funcionar en la intranet de la empresa sin necesidad de ser instalada en cada estación de trabajo. Se ahorra tiempo, el que toma en fotocopiar un documento. También se ahorra el tiempo de búsqueda, ya que es más fácil buscar por la descripción de un documento, que por una hoja en una pila de carpetas. Además de que la búsqueda se puede hacer desde cualquier lugar en Internet y se ahorra recursos como el papel, tinta, etc., lo que se traduce en ahorro de dinero y en un menor costo para el ecosistema.

1. Análisis del problema

1.1 Trabajos relacionados

Existe toda una diversidad de programas en el mercado que realiza digitalización de documentos e imágenes, los más conocidos (aunque ciertamente no los mejores) son el Asistente para cámaras y escáneres de Windows y el Xsane (en interfaz gráfica GNOME) y Kooka (en interfaz gráfica KDE) en Linux, incluso se puede encontrar software que se encarga de hacer OCR (Reconocimiento de Caracteres Óptico por sus siglas en inglés), como es el caso del famoso programa ABBYY Fine Reader OCR.

El proyecto a desarrollarse como propósito de esta tesis no intenta competir con ninguna de las alternativas de software nombradas, que aunque son muy completas también son principalmente de uso general, a final de cuentas siempre su propósito es digitalizar imágenes, propósito que cumplen muy bien. El software propuesto es algo mucho más específico porque se orienta a incluirse dentro de un CMS (Sistema de Manejo de Contenido por sus siglas en inglés), donde el interés es que una compañía pueda guardar la imagen de un documento (o también una foto) en la base de datos empresarial sin tener que pasar por el sistema de archivos, o por lo menos que este proceso sea transparente al usuario. De esta manera el usuario trabaja de manera más rápida y eficiente, además el usuario podrá realizar búsquedas de las imágenes en una base de datos por medio de una

descripción que el mismo usuario le atribuye, no así en un sistema de archivos. El sistema es web, es decir no necesita instalación para ejecutarse aunque si necesita que haya aplicaciones muy comunes en la estación de trabajo, como por ejemplo un navegador web. Y por esta razón puede ser ejecutado en las plataformas o sistemas operativos ya indicados, y esta es una característica en la que sí se puede hacer una comparación, y por lo tanto podemos decir que la gran mayoría de aplicaciones de escaneo de imágenes no ofrece esta funcionalidad, muchas se enfocan sólo en Windows, y las de Linux solo funcionan en Linux.

Por lo tanto se puede decir que sobre el ámbito de desenvolvimiento de esta tesis, no se ha encontrado trabajos muy similares más allá de lo aquí descrito.

1.2. Análisis de las herramientas disponibles

El proyecto a desarrollarse tiene entre sus objetivos su ejecución en Linux y Microsoft Windows (multiplataforma), el sistema también debe dar acceso web al usuario y debe administrar y manipular imágenes. Son muchas las herramientas que se van a necesitar para estas tareas principales que se deben completar, se irá entonces hablando de cada tipo de herramienta que se necesite y de todas las disponibles cuáles se escogieron, y por qué.

Lenguaje de programación

Para desarrollar el Cliente digitalizador de imágenes no se tiene más que dos opciones que son Java y Flash, debido a que pueden ser embebidos en una aplicación web y tener permiso de uso de los recursos del sistema como es un escáner que es lo que se usaremos nosotros, pero debido a que se encontró muy poca documentación sobre la utilización de Flash para el manejo de dispositivos de captura de imágenes, [23] [29], se decidió por escoger Java.

Para que el Cliente digitalizador de imágenes pueda comunicarse con la interfaz de captura de imágenes de Linux (SANE - léase sección 1.2.1.2) y la de Windows (TWAIN - léase sección 1.2.1.1) se debe utilizar código nativo que se ejecutará en cada máquina cliente y debe ser C++ ya que se utilizará JNI (Interfaz Nativa de Java).

Lenguaje de programación web

Son tres los lenguajes de programación web que lideran el mercado, éstos son PHP, ASP de Microsoft y JSP de Java. Debido a que uno de los objetivos del proyecto es que sea multiplataforma se descarta de primera mano ASP, además el CMS que escogeremos seguramente será basado en PHP, ya que son éstos los que lideran el panorama de CMS's de código

abierto, como Joomla, Drupal, Nucleus y PHP-Nuke [33]. Además PHP es el lenguaje más popular en el Internet, según lo refleja Google Trends [7], tiene la comunidad más activa y más grande, además que el servicio de hosting (hospedaje en un servidor de Internet) basado en este lenguaje de programación web es mucho más económico y fácil de conseguir debido a la gran cantidad de oferta en el mercado. Puede ejecutarse tanto sobre Linux y Windows, ya que en ambas plataformas puede funcionar el Servidor Web Apache, que es sobre el que debe instalarse [13]. Por todas estas razones tomaremos el camino del lenguaje web PHP para realizar este proyecto de tesis.

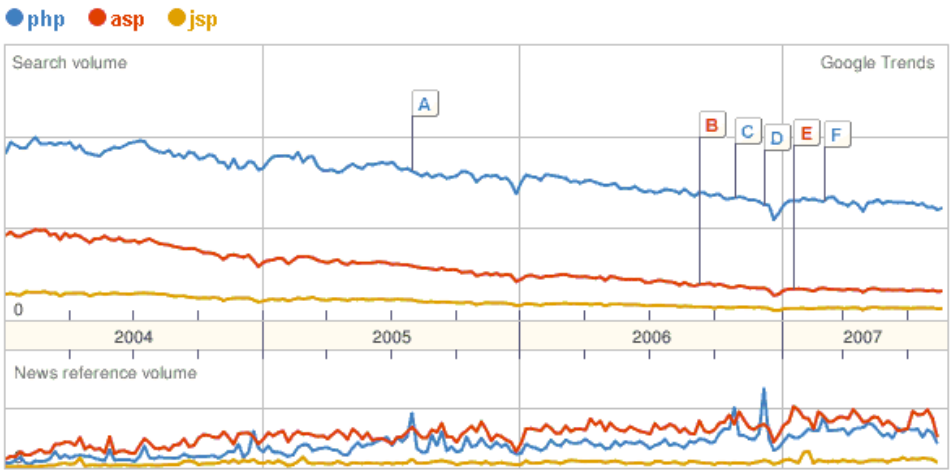


Figura 1.1. Comparación de popularidad entre los lenguajes de programación web PHP, ASP y JSP en Google Trends, realizada a finales de Agosto del 2007 (No tomar en cuenta índices con literales) [7]

Almacenamiento de datos

Debido a que utilizaremos el lenguaje de programación web PHP y por lo general las aplicaciones y los servicios de hosting (hospedaje en internet) ofrecen este servicio junto con el motor de base de datos MySQL, debido al "software bundle" (combinación de software en la instalación por defecto) que existe en la instalación de PHP con esta base de datos y en una arquitectura muy conocida en el mundo de desarrollo web de código abierto llamada LAMP [31] (Linux como sistema operativo, Apache como servidor web, MySQL como base de datos y PHP como lenguaje de programación), no hay muchas direcciones de las que escoger en este sentido.

La gran mayoría de CMS's (Sistema de Manejo de Contenido por sus siglas en inglés) dan soporte primordialmente a este motor de bases de datos, y a veces exclusivamente a éste. Eso no significa que MySQL sea una base de datos pobre de funcionalidad o estabilidad, sí es un motor de base de datos relativamente joven, pero muy poderoso y rápido; y definitivamente el más conocido y usado en el mundo de desarrollo web y de aplicaciones de código abierto [27].

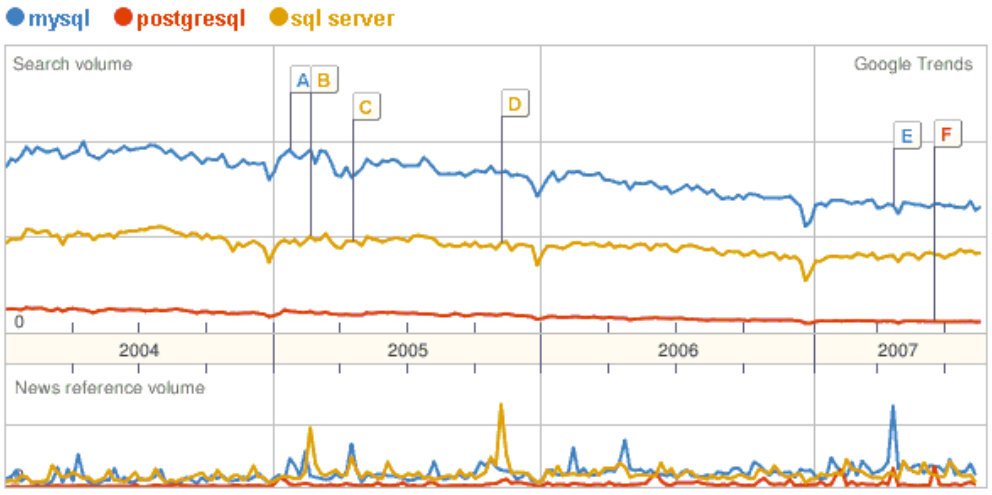


Figura 1.2. Comparación de popularidad entre motores de bases de datos MySQL, PostgreSQL y Microsoft SQL Server en Google Trends, realizada a finales de Agosto del 2007 (no tomar en cuenta índices con literales) [28]

Interfaces de captura de imágenes

Las interfaces de captura de imágenes son el nexo entre el sistema y el escaneo de imágenes en cada sistema operativo, debido a la importancia de esta tarea básica del proyecto, trataremos toda una sección (la 1.2.1) acerca de este tema. Debido a que no existe una interfaz que funcione en Linux y Windows, sino que existe una interfaz de captura de imágenes por cada plataforma (TWAIN en Windows y SANE en Linux), debemos adelantarnos al decir que tendremos que obligadamente trabajar con cada una de ellas, dependiendo de la plataforma en la que se encuentre ejecutando la aplicación de digitalización de imágenes.

Librería para digitalización de imágenes

Para facilitar, estandarizar y hacer lo más estable posible la digitalización de imágenes debemos utilizar una librería de Java para la digitalización de imágenes, las métricas principales para escoger a la que necesitamos son desempeño, flexibilidad, facilidad de uso y documentación, encontramos dos competidoras que cumplen con nuestras exigencias, éstas son la librería mms-computing y la librería Morena.

Morena, más que una librería es un Marco de Trabajo de Adquisición de Imágenes para Java, es un puente entre el hardware de adquisición de imágenes (escáneres y cámaras) y el mundo Java. Para comunicarse con el hardware, Morena utiliza las interfaces estándares TWAIN (digitalización en Windows y Mac OS X) y SANE para las plataformas basadas en Unix. Ofrece un API intuitivo y eficiente a los desarrolladores y es de licencia propietaria [29].

Mms-computing es un proyecto de software de código libre que lo que intenta es relacionar Java con tareas que muy pocos programadores realizan con él u otros dejan para realizar con lenguajes nativos, tareas como captura de imágenes, conexión telefónica a ISDN, fax, impresión, etc. [23]

	Mms-computing	**Morena**
Estabilidad	Estable	Estable
Licencia	Código Abierto (LGPL)	Pagada
Soporta TWAIN	Sí	Sí
Soporta SANE	Sí	Sí
Plataformas	Linux, Windows	Linux, Windows, Macintosh

Tabla I. Comparación de características entre Mms-computing y Morena

Finalmente nos inclinamos por mms-computing, ya que es una librería de código abierto, lo que hasta cierto punto nos permite un mayor control sobre lo que estamos haciendo y un código más acorde a nuestras necesidades.

Sistema de manejo de contenido

Uno de los objetivos principales de este sistema es integrarlo a un CMS para su uso como componente de captura de imágenes de documentos y administración de éstos. Así como sucede con las interfaces de captura de imágenes, debido a la importancia de este tema, acerca del CMS (Sistema de Manejo de Contenido) a escoger trataremos toda una sección (la 1.2.2).

1.2.1 Interfaces de captura de imágenes

Los sistemas operativos no hacen la digitalización de documentos directamente, sino que permiten que las aplicaciones se comuniquen con los

dispositivos de adquisición de imágenes a través de una interfaz (API), en Windows y Macintosh la API que se utiliza es TWAIN y en cambio en Unix y Linux se utiliza otra API que se llama SANE. En un mundo perfecto habría una misma interfaz para todos los sistemas operativos pero no es así en realidad, tanto TWAIN y SANE hacen exactamente lo mismo pero de distintas formas.

TWAIN ha existido desde 1992. Pero no existe realmente algún tipo de soporte seriamente encaminado para Unix ni Linux, debido entre otras razones principalmente al hecho de que sea un requisito para los proveedores de drivers (empresas vendedoras de escáneres) de TWAIN el entregar un cuadro de diálogo personalizado que exponga todas las características disponibles de sus dispositivos, lo que causa un problema para Unix [1].

A diferencia de Macintosh y Windows, Unix puede correr con diferentes interfaces gráficas de usuario. Más aún, una interfaz gráfica no es esencial para poder ejecutar Unix. ¿Cómo se puede lograr una reconciliación entre el énfasis de TWAIN sobre la GUI a nivel del driver y una no estandarizada o no existente GUI en las máquinas con Unix?, ese es el problema realmente.

SANE se encargó de resolver este problema. Al separar los controles del dispositivo de su representación en GUI, las aplicaciones de SANE pueden ser creadas para correr en línea de comandos o en GUI. Otro beneficio que atañe esta característica es el acceso transparente a los dispositivos de captura de imágenes a través de la red. A diferencia de TWAIN, SANE soporta acceso remoto para los dispositivos de adquisición de imágenes a través de la red [3].

	TWAIN	**SANE**
Captura de imágenes a través de la red	No	Sí
Soporte	Comercial	Comunitario
Independiente de la interfaz gráfica	No	Sí
Plataformas	Windows, Macintosh	Unix, Linux

Tabla II. Comparación entre principales características de TWAIN y SANE

A continuación ahondaremos más en estas dos especificaciones:

1.2.1.1 TWAIN

TWAIN no es un acrónimo. TWAIN proviene del libro "The Ballad of East and West" de Rudyard Kipling, nos fijamos en la parte que dice "... and never the

twain shall meet ...", donde se refiere a la dificultad que existía en esos días de conectar escáneres y computadoras personales, según la propia sección de Preguntas Frecuentes de los creadores de TWAIN. Pusieron TWAIN en mayúsculas simplemente para hacerlo más distintivo. También de acuerdo a su sitio oficial, TWAIN es un API de captura de imágenes que trabaja en los sistemas operativos Microsoft Windows y Apple Macintosh, como ya mencionamos anteriormente [2].

TWAIN necesita tres componentes de software y hardware que funcionan entre sí para habilitar la captura de imágenes, los que son la Aplicación, la Fuente y el Manejador de Fuentes.

La aplicación presenta un Menú, entre algunas de las opciones tenemos "Seleccionar Fuente" y "Capturar Imagen" que sirven la primera para seleccionar un dispositivo de captura de imágenes y la segunda para obtener una imagen de tal dispositivo. Una vez que el usuario ha seleccionado una de las opciones mencionadas, la aplicación se comunica con TWAIN mediante mensajes o también conocidos como eventos. Cuando se hace la adquisición de una imagen, TWAIN envía mensajes a la aplicación, los cuales se manejan en el bucle de manejo de mensajes (o manejo de eventos) de la aplicación.

Una fuente (fuente de datos) es un driver cuya tarea es controlar un dispositivo de adquisición de imágenes determinado y es implementado por el desarrollador del dispositivo para comunicarse de acuerdo a las especificaciones de TWAIN. Las fuentes son guardadas en archivos cuya extensión de archivo es .ds. Los archivos mencionados son almacenados en la carpeta c:\WINDOWS\twain_32.

El Manejador de Fuentes es el encargado de manejar las interacciones que se llevan a cabo entre las aplicaciones y las fuentes. Una de esas fuentes es la "fuente por defecto" (que es el escáner o cámara que se usará por defecto para escanear una imagen, parecido a la impresora predeterminada cuando se va a imprimir un documento), la misma que es usada por el manejador de fuentes en caso de no encontrarse la especificación explícita de una fuente.

La aplicación se comunica de forma directa con el manejador de fuentes, el manejador de fuentes se comunica directamente con la aplicación y la fuente, y cada fuente se comunica de manera directa con el manejador de fuentes y el hardware [2].

1.2.1.2 SANE

SANE es un API de captura de imágenes. La palabra SANE es un acrónimo para "Scanner Access Now Easy" (Acceso a Escáner Ahora Fácil). El API

provee acceso estandarizado a virtualmente cualquier escáner de imágenes, cámara, u otro tipo de dispositivo de adquisición de imágenes.

SANE fue introducido unos años después de TWAIN para dar soporte de adquisición de imágenes en plataformas Unix y Linux, debido a que TWAIN no fue capaz de hacerlo. Aunque SANE fue originado en Unix y Linux, se portó a Mac OS X, OS/2 y otros sistemas operativos.

SANE provee una interfaz estandarizada para dispositivos de captura de imágenes. Las aplicaciones que utilizan esa interfaz se conocen como “Front-ends” de SANE. Los drivers que implementan esa interfaz y controlan los dispositivos de captura de imágenes son conocidos como Back-ends de SANE [3].

El Front-end de la aplicación se comunica con el Back-end del driver a través de un intermediario conocido como libsane.so -- un enlace simbólico (también conocido como symlink) al Back-end de un driver, que controla un dispositivo de adquisición de imágenes específico. Cuando una aplicación se comunica con libsane.so, en realidad se está comunicando con cualquier driver que esté representado por libsane.so [4].

La parte más importante del ambiente SANE bien podría ser el formato usado para representar las imágenes capturadas. SANE ve a una imagen como un área rectangular dividida en filas y columnas. SANE se refiere a la intersección de una fila y una columna como un píxel cuadrático. Este píxel consiste de una o más muestras, donde una muestra representa un canal de color (rojo, verde y azul) o una sombra de gris. Cada muestra también tiene una profundidad de bit: 1, 8, y 16 bits por muestra son profundidades de bit válidas.

SANE transmite una imagen como una secuencia de tramas. Cada trama cubre el área rectangular completa pero puede contener solo un subconjunto de los canales. Por ejemplo, una trama puede contener muestras de rojo, o de verde. Alternativamente, una trama puede consistir en muestras de tres canales de colores.

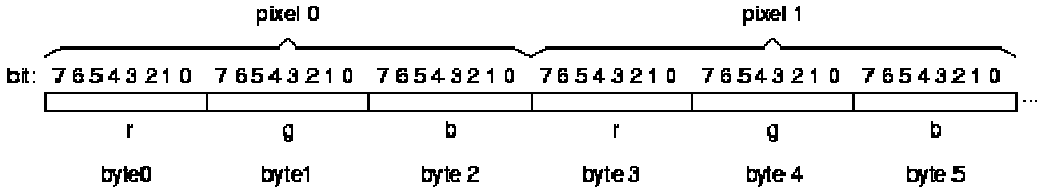


Figura 1.3. Orden de bits y bytes en los datos de una imagen en SANE [32]

1.2.2 Sistema de manejo de contenido

Los CMS (Sistemas de Manejo de Contenido por sus siglas en inglés) de código abierto pueden hacer la creación y administración de un sitio web

mucho más fácil sin tener que pagar una licencia. Pero, ¿cuál debemos utilizar en el proyecto? Haremos una comparación cautelosa entre Joomla y Drupal para poder ver sus fortalezas y debilidades.

Todo sitio web necesita contenido actualizado, navegación intuitiva, y un buen diseño gráfico. Y cada administrador quiere ser capaz de levantar un sitio web de forma rápida, hacer cambios fácilmente, y agregar contenido nuevo con un mínimo de esfuerzo. Ahí es donde un CMS entra en acción. Un CMS hace tres cosas: hace más fácil levantar un sitio web, promueve buenas prácticas para un sitio web y permite al personal no técnico hacer actualizaciones al sitio.

Uno puede conseguir todas estas funcionalidades sin un CMS, así como se puede estar en contacto con personas sin utilizar correo electrónico. Pero así como el correo electrónico, un CMS puede hacernos la vida más fácil.

Dos herramientas en particular han estado dominando la discusión de CMS's de código abierto en los últimos años: Joomla y Drupal. Ambas proveen una función sólida, funcionalidad útil para construir y mantener un sitio web.

Estas herramientas tal vez tengan más similitudes que diferencias. Ambos son sistemas CMS sofisticados y útiles que dan soporte a la mayoría de las

tareas que les interesa a los editores de contenido y los visitantes de un sitio web.

Los dos CMS's pueden ayudar a levantar una estructura del sitio web y un sistema de navegación útiles. Ambos permiten a los editores de contenido no técnicos actualizar contenido, agregar nuevas páginas o cambiar artículos en el menú de navegación. Los dos entregan soporte a un diseño gráfico completamente configurable (no existe necesidad de que los visitantes sepan qué CMS se está utilizando, o ni siquiera de que se está utilizando uno), además escogen automáticamente los artículos de contenido apropiados para mostrar a los visitantes del sitio web basándose en reglas, como por ejemplo, la página de Inicio (Home) puede mostrar las cuatro noticias más recientes o los eventos que están más cercanos por venir en la semana. Los dos CMS's facilitan el compartimiento del trabajo interno al permitir a algunos miembros del personal actualizar sólo una parte de las cosas y otros actualizar otras. Otra característica que comparten Joomla y Drupal es que proveen actualizaciones, responden preguntas, y proveen extensiones a través de una comunidad de usuarios y desarrolladores muy fuerte [6].

Joomla se esfuerza por el poder en la simplicidad. Sus programadores creen que cualquiera con un poco de conocimientos técnicos no debería tener

problema configurando y manteniendo un sitio web con Joomla, que es una herramienta amigable, y que prioriza la facilidad de uso.

Figura 1.4. Captura de pantalla de la plantilla de sitio por defecto del CMS Joomla

Joomla está diseñado para trabajar muy bien en servidores de ambientes compartidos (los que son servidores que alojan cuentas de más de un usuario cada uno resultando en paquetes más económicos y más comunes para los clientes, un servidor de ambiente compartido es lo contrario a un servidor dedicado). Su instalador se parece mucho a un simple instalador utilizado por un software de escritorio común, y la interfaz administrativa que usan los editores de contenido se parece mucho a un programa de escritorio

también. Hay pocas barreras de ingreso con Joomla, lo que significa que no debe tomarle mucho tiempo a un desarrollador web levantarlo y correrlo.

La facilidad de iniciarse en una herramienta viene con su compensación. Joomla puede ser una alternativa maravillosa para construir un sitio web sofisticado con cientos de páginas y tipos de contenido comunes, tales como noticias y eventos. Sin embargo, en su instalación sin extensiones tiene una limitada funcionalidad para tratar con estructuras de contenido dinámico sofisticadas. Por ejemplo, la navegación del sitio está limitada a no más de dos niveles de jerarquía [6].

Drupal también es muy poderoso y fácil de usar, así como Joomla está construido en PHP, y puede ser instalado en servidores de ambiente compartido, y provee un número de herramientas que permiten a los no técnicos instalar un sitio web. En general, requiere una curva de aprendizaje más pronunciada que Joomla, pero en su instalación básica ofrece más funcionalidad para sitios web más sofisticados, así como también una plataforma más rica para los programadores. Una de las fortalezas de Drupal es su amplia variedad de extensiones, otorgando funcionalidades tales como registro de eventos, noticias por correo electrónico y donaciones en línea. Drupal también tiene integración nativa a funciones no muy comunes en un CMS, como noticias por correo y donaciones en línea [6].

Figura 1.5. Captura de pantalla de la plantilla de sitio por defecto del CMS Drupal

Nuestra elección

Google Trends es una herramienta que analiza las búsquedas web en Google y que visualiza las tendencias de búsqueda a través del tiempo usando gráficos de volumen de búsqueda, dichos gráficos por lo general proveen un buen mecanismo para comparar la popularidad de dos o más frases, productos, o palabras de los usuarios de Google.

El siguiente es el gráfico que muestra Google Trends a finales de Agosto del 2007 comparando Drupal y Joomla:

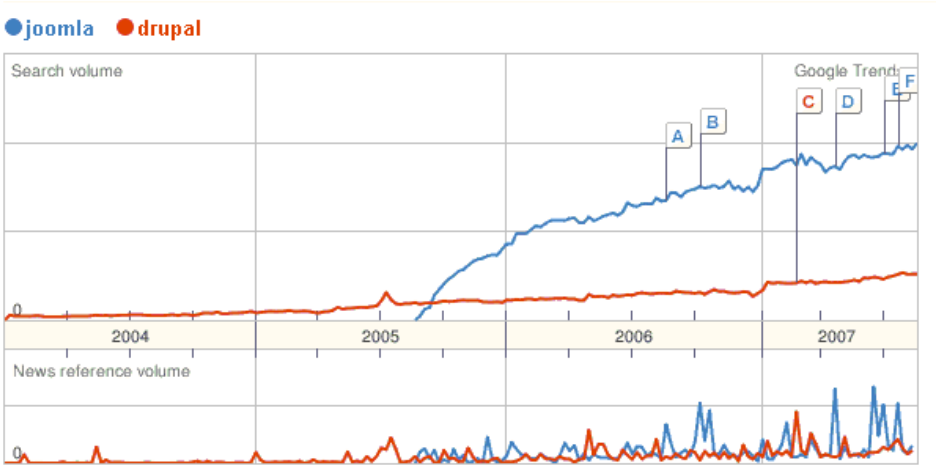


Figura 1.6. Comparación de popularidad entre Joomla y Drupal en Google Trends, realizada a finales de Agosto del 2007 (no tomar en cuenta índices con literales) [8]

Se debe también tomar en cuenta que Joomla ha estado mayor tiempo en la arena de los CMS's (Sistemas de de Manejo de Contenido por sus siglas en inglés) que el tiempo que dice el gráfico que comienza en el 2005, ya que Joomla es un proyecto que basa su código en Mambo, un CMS para el que los desarrolladores de Joomla trabajaban inicialmente, pero después, por una disputa con los dueños de la compañía Miro Corporation, dueña de Mambo decidieron salir y formar lo que hoy es Joomla, y como Mambo es un proyecto de código abierto tomaron sin problema el código de Mambo para iniciar su nuevo proyecto [34].

En el gráfico “Comparación de popularidad entre Joomla y Drupal en Google Trends” podemos notar que Joomla es mucho más popular que Drupal y también tiene un crecimiento de popularidad más rápido que el de Drupal. La razón es que Joomla tiene un balance más atractivo de soporte, funcionalidad, calidad de código, facilidad de uso, documentación, diseño de interfaz de usuario y mercadeo del producto.

Lo que buscamos para el proyecto no es el mejor CMS, aunque sí buscamos dentro de los que creemos son los mejores de su tipo. Nuestra decisión se inclina a Joomla por dos razones principales, como necesitamos desarrollar código para una extensión en un CMS, necesitamos que su comunidad sea bastante activa para que nos pueda servir de soporte para cualquier duda que podamos tener, Joomla tiene el foro de discusión enfocado a un CMS más grande del planeta, alcanzando los 100,000 usuarios inscritos (basado en cifras del mes de Abril del año 2007), que compite con algunos de los foros más grandes de internet, además de una impresionante documentación [5]. La segunda razón es que aunque encontramos extensiones de manejo de imágenes en ambos CMS's, según nuestra investigación Joomla tiene las extensiones más actualizadas y soportadas y algunas incluso con funcionalidades no muy básicas pero sí muy importantes para el desarrollo del sistema, como son búsqueda de imágenes por descripción y almacenamiento en base de datos.

1.2.3 Ambiente de desarrollo

Debido a que el proyecto de esta tesis necesita probarse tanto en Linux como en Windows, requerimos herramientas de desarrollo que en lo posible trabajen en ambas plataformas, y que también en ambas plataformas entreguen las mismas funcionalidades.

Enfocándonos en lo que es la selección del Editor de código, también conocido como IDE (Ambiente de desarrollo integrado por sus siglas en inglés), tenemos que utilizar uno que nos preste la facilidad de trabajar en PHP, Java, y C++, o sino tendríamos que buscar uno para cada lenguaje, además de que nos permita trabajar con algún sistema de versionamiento para almacenar el código de una manera eficiente y mantener un registro de los cambios hechos al código, así como también tener la oportunidad de invertir algún cambio realizado. Nos decidimos por utilizar el Eclipse (versión 3.2) que es un marco de trabajo para software de código abierto escrito principalmente en Java. En su forma por defecto (instalación básica) es un IDE para Java, los usuarios pueden extender sus capacidades al instalar plug-ins, tales como cajas de herramientas de desarrollo para otros lenguajes de programación [25]. En nuestro caso, instalamos la extensión Phpeclipse para tener la capacidad de editar código PHP. De no haberlo hecho de esta forma deberíamos haber utilizado dos o más IDE's que deberíamos haber

instalado tanto en Linux como Windows para realizar nuestras pruebas [9]. Nuestra segunda opción era utilizar el Eclipse solo para Java y el Zend Studio para PHP [24], siendo este último también multiplataforma, pero que es un software de licencia propietaria.

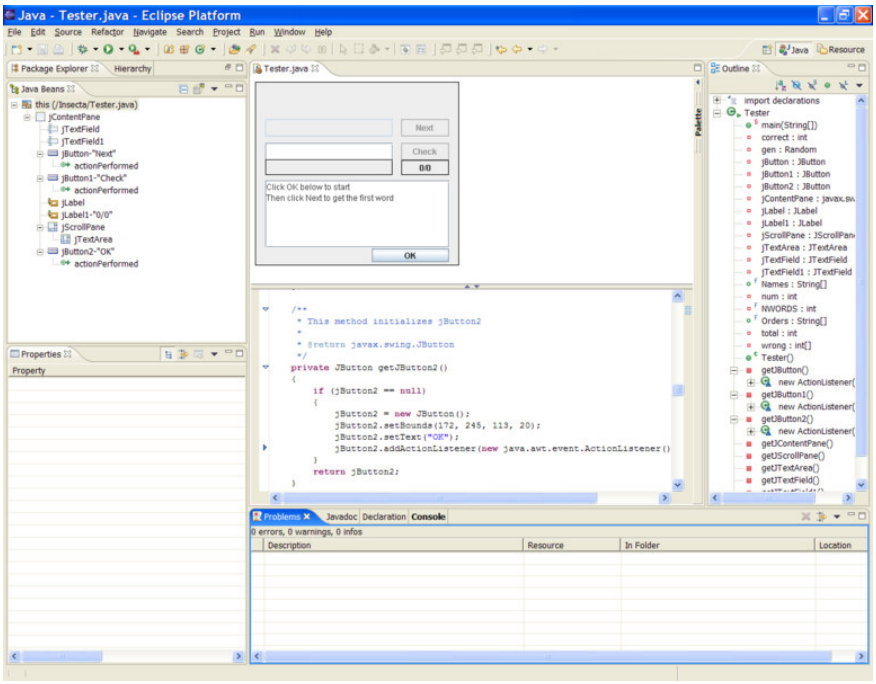


Figura 1.7. Captura de pantalla de ambiente de desarrollo integrado Eclipse versión 3.2

En materia de herramientas de versionamiento teníamos básicamente dos opciones: CVS y Subversion. El Eclipse tiene soporte por defecto para CVS y soporte para Subversion a través de la instalación de un plug-in llamado Subclipse [10], nos decidimos por Subversion dado que tiene funcionalidades

más avanzadas y es una tecnología nueva, y el CVS aunque reina aún en el mercado es un software que está de salida de éste realmente. El servidor de Subversion puede ser instalado tanto en Linux como en Windows sin ningún problema.

Para poder firmar (electrónicamente) el software necesitamos OpenSSL, que es principalmente lo que dará permiso a nuestra aplicación web para que pueda utilizar recursos internos del computador tales como el escáner (que es el caso de este proyecto), pudimos encontrar que OpenSSL se lo ofrece en Linux como un paquete lo que hace fácil su instalación, y en Windows se lo ofrece como un instalador gratuito con asistencia en forma de pasos guiados [11].

El desarrollo y pruebas los hemos realizado en Windows XP y Fedora Linux, dos de las distribuciones más utilizadas en la actualidad a nivel de escritorio, cada una en su propia plataforma.

2. Diseño de la solución

Basándonos en los objetivos y en lo descrito hasta el momento, hemos realizado un Diagrama de casos de uso que generaliza lo que se quiere obtener del sistema como conjunto (Apéndice A).

Nosotros por supuesto nos enfocaremos en la parte de digitalización de imágenes, la que detallaremos en el siguiente gráfico:

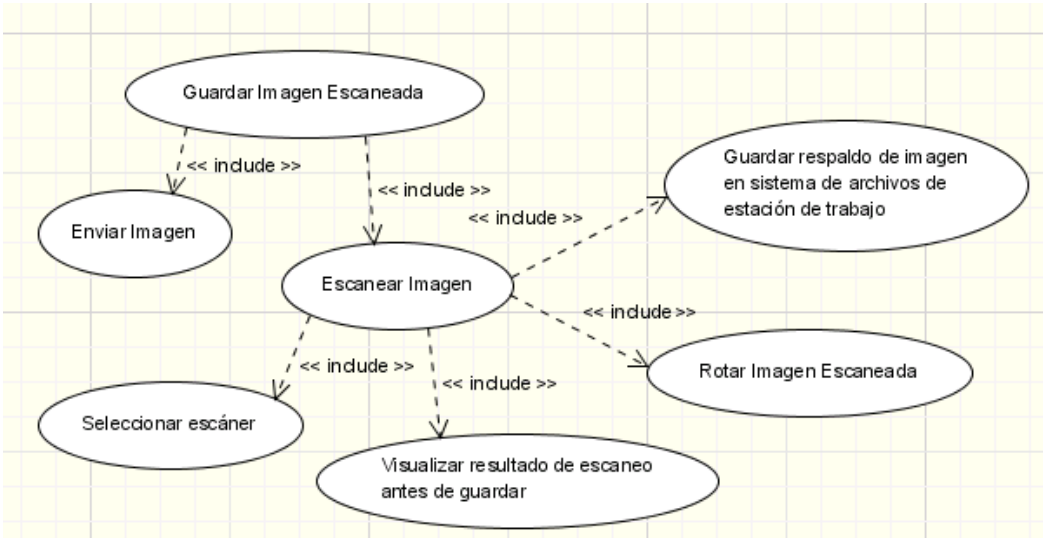


Figura 2.1. Enfoque sobre casos de uso incluidos en el caso de uso "Guardar imagen escaneada"

El sistema está formado por dos partes principales: el Cliente digitalizador de imágenes y el Servidor receptor de imágenes.

A continuación ahondaremos más sobre cada uno de ellos.

2.1. Cliente digitalizador de imágenes

Para tener una mejor idea de lo que técnicamente se intenta lograr, podemos decir que el Cliente digitalizador de imágenes es el applet que se encarga del escaneo de documentos o imágenes conectándose a las interfaces de captura de imágenes TWAIN (Windows) y SANE (Linux), dependiendo de la plataforma en la que se esté ejecutando el applet. Este cliente será embebido en el CMS, agregado como una funcionalidad de la extensión de la Galería de Imágenes en la interfaz de administración (Apéndice B).

El Cliente digitalizador de imágenes recibe la imagen ya digitalizada y se encarga de enviar tres datos importantes, que son la misma imagen en un archivo, una descripción de lo que contiene la imagen para facilitar posteriormente su búsqueda dentro de la base de datos del CMS (esta descripción es una cadena de caracteres que será entregada por el usuario), y un identificador que señala el álbum al que pertenece la imagen, lo que sirve para clasificar las imágenes para incrementar la facilidad de navegación a través de éstas. Con las especificaciones anteriores, podemos definir el Diagrama de clases (Apéndice C).

2.2. Servidor receptor de imágenes

Es el que se encargará de recibir el archivo de la imagen escaneada y almacenarlo en la base de datos del CMS, para pasar después a ser controlado por la Galería de Imágenes instalada como extensión del CMS.

El receptor de imágenes primero obviamente recibirá la imagen del Cliente digitalizador de imágenes y validará su formato y tamaño en espacio de disco, también realizará una validación de la sesión, ya que esta operación solo se puede realizar en una sesión de usuario en la Interfaz de Administración del CMS. Después de que se haya comprobado de que toda la información es válida, el Receptor se encargará de almacenar la imagen, para ponerla disponible al CMS y su extensión de Galería de Imágenes.

3. Implementación

3.1. Conexión a interfaces de captura de imágenes

Java es un lenguaje independiente a la plataforma, es decir, puede correr en múltiples plataformas sin realizar modificaciones al código fuente.

Para que Java pueda comunicarse con los dispositivos de adquisición de imágenes, primero debe reconocer el sistema operativo en que se encuentra para con eso saber con qué interfaz le toca lidiar para la consecuente captura de imagen. Para el siguiente paso entra en juego la comunicación a las librerías nativas, conexión conocida en Java como JNI (Java Native Interface), y que son las que finalmente se encargan de comunicarse a los artefactos de exploración de imágenes. Con cada interfaz la técnica de comunicación es distinta como veremos a continuación:

3.1.1. Conexión a interfaz TWAIN en Windows

Al ejecutar la aplicación de Java, lo primero en realizarse es la inicialización de TWAIN al iniciar también el programa, si la inicialización falla, la aplicación sale después de mostrar un mensaje de error al dispositivo de salida estándar (consola). Si la inicialización se da correctamente, después de un momento aparece una ventana con un menú y el panel principal de nuestra aplicación.

Dos son los escenarios principales del programa: Captura de Imagen y Selección de Fuente. Al dar clic sobre "Seleccionar Fuente..." se ejecuta jtwain.select(), función que muestra un cuadro de diálogo mostrado en la siguiente figura.

Figura 3.1. Cuadro de diálogo de selección de dispositivo de captura de imágenes para realizar digitalización entregado por TWAIN

Debido a que los nombres de las fuentes dependen de los tipos de dispositivos con soporte TWAIN conectados, se pueden observar listas diferentes de nombres de fuentes. Seleccionamos el nombre de fuente apropiado y damos clic sobre el botón Seleccionar. La fuente resaltada identifica la nueva fuente por defecto [2].

Retornamos al Panel Principal y esta vez damos clic sobre el botón de Captura de Imagen que ejecuta la función acquire(). Cuando acquire() se ejecuta, muestra un cuadro de diálogo que muestra la siguiente figura.

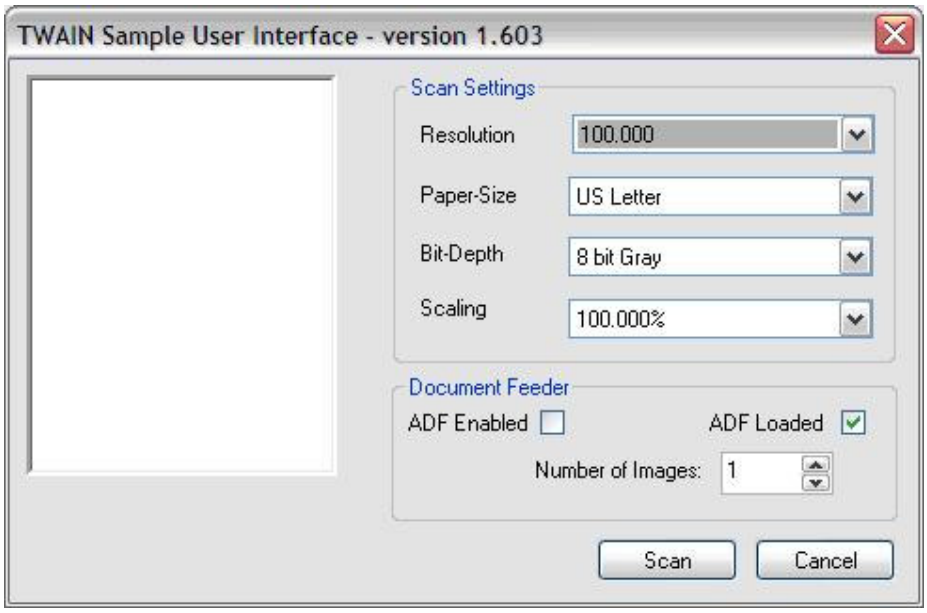


Figura 3.2. Cuadro de diálogo por defecto de captura de imágenes de interfaz TWAIN en Windows XP

El cuadro de diálogo puede cambiar dependiendo del tipo de fuente que se haya seleccionado previamente. Si el usuario da clic sobre el botón Cancelar, acquire() retorna nulo en el evento de captura y la imagen no es retornada. Pero si una imagen es capturada exitosamente, la imagen será entonces inmediatamente mostrada en su Vista Previa en tamaño reducido (thumbnail).

Si damos clic sobre el botón “Ver Imagen” en tamaño real podremos ver aparecer una ventana conteniendo la imagen recién capturada en su tamaño original, tal como vemos en la figura, la imagen estará contenida en un panel

con barras de desplazamiento para una más fácil navegación sobre la imagen sin importar su tamaño ni la resolución de la pantalla.

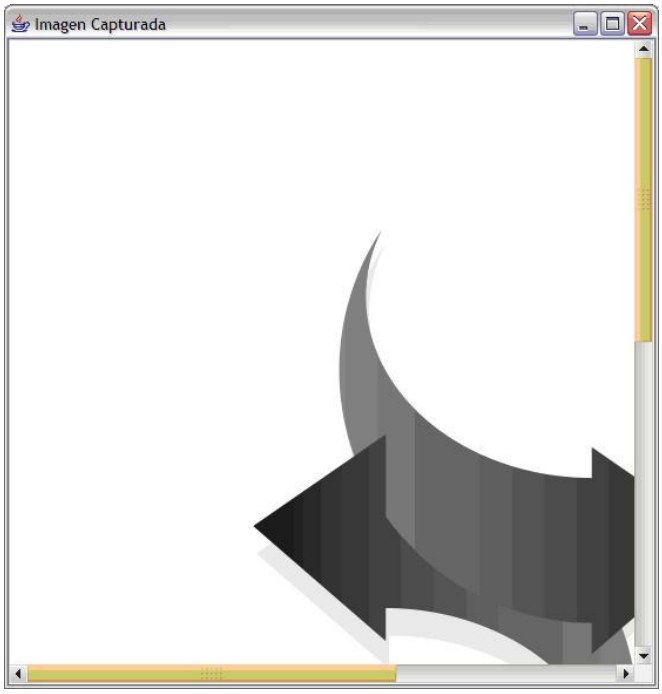


Figura 3.3. Presentación de imagen capturada en tamaño real

3.1.2. Conexión a interfaz SANE en Linux

El API de SANE fue escrito en lenguaje C. Así como TWAIN provee el archivo cabecera twain.h para ser incluido en el código fuente basado en TWAIN, SANE provee la cabecera sane.h para ser incluido en código fuente basado en SANE.

Una aplicación llama a sane_init() para iniciar su interacción con SANE. La aplicación por lo general sigue con una llamada a sane_get_devices() para obtener una lista de dispositivos disponibles. Un dispositivo será seleccionado de la lista y su nombre será pasado a la función sane_open() para abrir el dispositivo. Una vez que el dispositivo esté abierto, sus controles pueden ser establecidos o consultados, lo que ocurre dentro de un ciclo, donde cada iteración invoca a sane_get_option_descriptor() para obtener un descriptor de control, seguido por sane_control_option() para consultar o establecer el control.

Después de realizarse la configuración del dispositivo sigue la captura de la imagen. Esta tarea comienza con una llamada a sane_start(), y continúa con un ciclo donde se llama a sane_get_parameters() y después a sane_read(). La captura de imágenes continua hasta que sane_read() retorna EOF (fin de archivo), o cuando el usuario decide terminar la captura de la imagen (asumiendo que la aplicación permite la cancelación de la captura de ésta), invocando sane_cancel(). Ya habiendo realizado la captura, se llama a sane_close() y el dispositivo que se abrió se cierra. Por último se invoca a sane_exit() para romper la conexión de la aplicación con el back end de SANE.

Igual que con TWAIN tenemos una clase específica para twain, para SANE tenemos una clase específica para sane que se encarga de comunicarse con la librería nativa y hace en definitiva las mismas operaciones que la de TWAIN, pero las diferencias radican en que también en la de SANE se obtienen algunos otros datos como por ejemplo los parámetros, de los que ya hemos hablado anteriormente.

Para la selección de un dispositivo de captura de imágenes y su posterior uso para la digitalización creamos un panel con un selector que contiene la lista de dispositivos que retorna SANE con sane_get_devices().

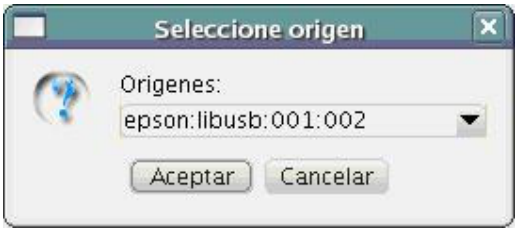


Figura 3.4. Cuadro de diálogo de selección de dispositivo de captura de imágenes generado con información obtenida de la interfaz SANE

Para la captura de imágenes creamos otro panel con los descriptores de SANE para establecer los valores de los parámetros de exploración de la imagen.

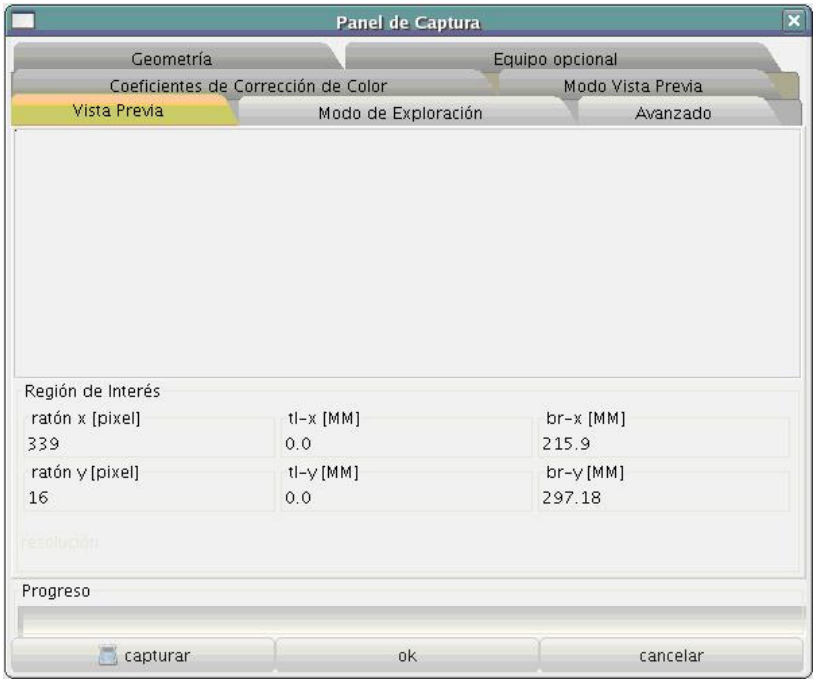


Figura 3.5. Cuadro de diálogo para realizar captura de imagen usando la interfaz SANE en Linux

Durante todo este subcapítulo (3.1) hemos explicado como trabaja cada una de las interfaces SANE y TWAIN por separado, ¿pero cómo se hace para que una aplicación en Java entienda a ambos? Realmente la modularización toma un papel principal en este aspecto, primero que nada se divide las operaciones entre las operaciones que atañen a ambos y las especializadas de cada interfaz, de ahí clases abstractas y encapsulamiento hacen el resto. Optamos por usar una misma interfaz gráfica principal para ambas interfaces de captura para que el usuario se sienta en mayor dominio del programa y mayor confianza sin importar con cual esté trabajando. De ahí la interfaz

gráfica de TWAIN como ya hemos explicado anteriormente es poco o nada modificable, entonces optamos por trabajar sobre una interfaz para SANE que se parezca a la interfaz básica de TWAIN, todo pensando en una mejor interacción con el usuario.

Para el manejo de las interfaces de captura de imágenes utilizamos una librería de Java llamada mms-computing que nos permite un alto grado de escalabilidad y modularización [23].

El panel principal de la interfaz gráfica (clase MainPanel) del sistema de la tesis además de sus funciones obvias, implementa una interfaz de Java de la librería mms-computing, llamada ScannerListener, existe una sola función en esta interfaz llamada update() que lo que hace es recibir los mensajes que envía el escáner, como es el caso de la selección de un nuevo dispositivo de captura de imágenes por defecto o la captura en sí de una imagen, en caso de recibirse una nueva imagen el programa (a través de la clase MainPanel) tiene un algoritmo que genera una imagen en miniatura de la imagen capturada para mostrarla en el panel principal, esto pasa principalmente por motivos de espacio, ya que no sabemos de antemano qué tamaño va a tener la imagen. Existen dos clases más (que forman parte del desarrollo del sistema de la tesis y no de ninguna de las librerías utilizadas) que son unos hilos (threads) que se encargan: el uno del envío (subir) de la imagen al CMS

por vía web (esta clase se llama ImageUploadThread) y el otro se encarga de crear un respaldo de la imagen en formato JPEG para ser almacenado en el sistema de archivos de la estación de trabajo del usuario (esta clase se llama ImageSaveThread). Finalmente tenemos la clase ScannerApplet, que hereda la clase JApplet y solo se encarga de envolver al MainPanel. En MainPanel también hemos desarrollado algoritmos (funciones) de rotación de imagen, que rotan tanto la imagen reducida (thumbnail) del documento como la propia imagen que se desea almacenar.

3.2. Almacenamiento remoto

3.2.1. Interfaz de envío

Una vez capturada la imagen se necesita realizar el almacenamiento del archivo, primero que nada debemos enfatizar en que un applet o similar por cuestiones de seguridad no tiene acceso directo al sistema de archivos del servidor del que proviene, podríamos en cambio utilizar una conexión directa del applet a una base de datos, pero tal acción acarrearía otro problema, tendríamos que tener el usuario y clave de esa base de datos como parte del JAR de la aplicación, lo cual sabemos que es muy inseguro por la facilidad que brinda el código de Java para ser decompilado, podríamos encriptar el código pero nunca sería 100% seguro. Entonces, ¿qué es lo que si podemos hacer que sea más seguro? La solución que se tomó fue hacer que la

aplicación haga un pedido HTTP conectándose a una página web como lo haría un navegador web al llenar un formulario de envío de archivos, además del archivo se envía en método POST los datos (identificadores) del documento al que se desea enlazar la imagen.

Antes de enviar la imagen debemos tomar el objeto que contiene a la imagen en memoria y guardarlo temporalmente en disco, luego tomamos ese archivo generado y lo enviamos.

La página web manda mensajes en respuesta al envío que nuestra aplicación en Java interpreta y así logran comunicarse Java y PHP. De ahí la aplicación hecha en PHP se encarga del almacenamiento en una base de datos y la posterior presentación de los archivos guardados.

3.2.2. Interfaz de recepción

Como ya dijimos en breves rasgos en la sección anterior, una aplicación hecha en lenguaje scripting PHP se encarga de recibir y almacenar en una base de datos MySQL las imágenes con sus datos, así mismo esta aplicación se encarga de mostrar las imágenes ya almacenadas, de esta manera el guardado es limpio y transparente para la aplicación hecha en Java, además de que permite una mayor modularización de las tareas.

La aplicación en PHP al recibir el pedido HTTP de la aplicación Java hace algunas validaciones, verifica el archivo por el tipo MIME entregado, también verifica los identificadores del documento al que pertenece la imagen.

Terminadas todas las validaciones pertinentes la aplicación en PHP se conecta a la base de datos y guarda los datos de la imagen incluyendo a la imagen misma dentro de un campo BLOB.

Para mostrar la imagen, la misma aplicación en PHP por medio de otro archivo se encarga de esta tarea, para lo que debe enviarse el identificador del documento al que pertenece la imagen en un pedido HTTP con método GET (una consulta en el mismo URL), con estos datos se puede realizar la búsqueda en la base de datos y hacer que el script entregue el contenido del archivo de la imagen, antes de eso enviamos una cabecera HTTP indicándole al navegador web que se encargue de la presentación gráfica de la imagen que se trata de un archivo JPEG, es decir, esta cabecera logra que el archivo php ya no sea visto por el navegador web como un archivo de texto plano con código HTML sino el archivo de una imagen.

3.3. Permisos de seguridad para el cliente digitalizador de imágenes

Esta aplicación utiliza periféricos de entrada del sistema cliente (escáner o cámara web), así como también necesita acceso al Sistema de Archivos

(para almacenar temporalmente las imágenes escaneadas antes de enviarlas o para guardar un respaldo de dichas imágenes), para lo que se necesita tener permisos, para conseguir esos permisos el archivo JAR necesita ser firmado digitalmente, seguiremos los siguientes pasos para firmar el JAR:

1. Descargamos OpenSSL, lo realizamos desde el sitio web del programa. OpenSSL fue diseñado principalmente para su funcionamiento en Linux, pero también existe una distribución que funciona sobre la plataforma Windows. Los siguientes pasos asumen que se utiliza el instalador disponible para Windows.
2. Instalamos OpenSSL, se verá que una nueva carpeta con nombre openssl se creará sobre la raíz (C:\openssl).
3. Agregamos la carpeta nombrada en el paso anterior a las Variables de Entorno del Sistema Operativo, en la variable PATH, para poder acceder después a OpenSSL desde cualquier carpeta sin necesidad de poner su ruta completa.
4. Ahora necesitamos crear un archivo que represente nuestra identidad para la firma que haremos sobre el archivo JAR. Abrimos una ventana de consola de DOS y ejecutamos lo siguiente:

```
C:\>openssl req –new –x509 –keyout
c:\OpenSSL\private\CAkey.pem –out
```

```
c:\OpenSSL\private\CAcert.pem -config
C:\OpenSSL\bin\openssl.cnf
C:\>cd OpenSSL
C:\OpenSSL>cd private
C:\OpenSSL\private>openssl req -new -x509 -key cakey.pem
-out cacert.pem -days 1095
C:\OpenSSL\private>openssl pkcs12 -export -in cacert.pem
-inkey cakey.pem -out cacert.p12
C:\OpenSSL\private>keytool -list -v -keystore cacert.p12
-storetype PKCS12 -storepass miclave
```

Algunos de los comandos anteriores solicitan información sobre el programador y la empresa dueña del sistema (estos datos no son importantes para la ejecución del aplicativo). Asumiendo q el nombre de su JAR es scanner1.jar, para firmarlo ejecutamos lo siguiente:

```
C:\OpenSSL\private>jarsigner -keystore cacert.p12 -
storetype PKCS12 -storepass miclave -sigfile mmsc
scanner1.jar 1
```

Si se desea volver a firmar el JAR sólo se necesita repetir la última línea cambiando el nombre del archivo JAR si es necesario [11].

Como producto de la firma del JAR, cuando se trate de acceder a éste en modo de applet desde un navegador web, antes de iniciarse su ejecución aparecerá un aviso que además de mostrar algunos de sus datos establecerá que de ser aceptada su ejecución, éste tendrá control de algunos recursos del sistema (se puede establecer la respuesta como a esta pregunta como respuesta predeterminada para futuras ejecuciones, en tal caso este mensaje ya no volverá a aparecer). Si este aviso no se muestra y simplemente el applet no se ejecuta significaría que la operación anterior no fue satisfactoria, en ese caso se debe volver a intentar el proceso desde el principio.

3.4. Integración con sistema de manejo de contenido

Para lograr la integración de la aplicación de digitalización de imágenes con el CMS Joomla sabíamos que debíamos hacerlo integrándola a una extensión de Joomla que se encuentre en la categoría de Galería de Imágenes [30], para poder mostrar y clasificar las imágenes escaneadas, nuestra meta no era integrarla a la mejor galería que encontráramos ni tampoco siquiera a la más popular, sino a la que tuviese las funcionalidades que brinden una mejor interacción al usuario tomando en cuenta la tarea objetivo de nuestra aplicación final. Como mencionamos el tema previamente en este documento, la aplicación contendrá un gran volumen de imágenes,

es primordial que el usuario encuentre fácilmente una imagen de forma rápida y sin problemas, por eso un buscador es una funcionalidad indispensable en la herramienta a seleccionar, la gran mayoría de las Galerías de Imágenes disponibles (incluyendo las de mayor renombre) tenían como usuarios potenciales a personas naturales que tienen galerías pequeñas que desean mostrar las fotos de reuniones de familiares y amigos, por eso nos decidimos por un componente para Joomla llamado HeXimage que tiene las funcionalidades básicas de una galería de imágenes (administración de álbumes e imágenes) además de los requerimientos del sistema.

HeXimage ofrece algunos módulos (los módulos en Joomla no son más que pequeños artículos de contenido que pueden ser mostrados en cualquier parte de una plantilla de Joomla permitiendo ser colocados en cualquier posición dentro de una página web del CMS) muy útiles, como son el de mostrar los álbumes de imágenes disponibles, imágenes aleatorias y por supuesto el tan requerido buscador de imágenes.

Se tuvo que agregar una nueva opción al menú del componente para poder integrar nuestra aplicación en una página web dentro de él, página a la que sólo se tiene acceso a través de la interfaz de administración de Joomla, también se tuvo que hacer algunas adecuaciones para hacer más fácil de

usar a esta sencilla extensión, como agregarle iconos para poder saltar de forma más rápida de una página de administración a otra, también hicimos la interacción del usuario más dinámica y atractiva usando código AJAX del Marco de Trabajo Dojo para mostrar las imágenes en ventanas interactivas dentro de las páginas web en la interfaz de usuario [19].

4. Características de la implementación de la aplicación

4.1. Requerimientos de la aplicación

Para ejecutar la aplicación web se necesita tener un navegador web con el plugin de Java instalado (versión 1.4.2 o superior), también necesitamos tener habilitado JavaScript, la plataforma puede ser cualquier distribución de Linux o Windows. También se necesita un mínimo de 64MB de RAM. Obviamente requerimos también de un dispositivo de captura de imágenes como un escáner, el escáner debe tener soporte para TWAIN si se trabaja en Windows, o SANE si se trabaja Linux como sistema operativo.

4.2. Procedimiento de instalación y configuración

Del lado del cliente la aplicación funciona sin necesidad de realizar ningún tipo de instalación, que no sea la instalación del navegador web o de la máquina virtual de Java y su plugin para el navegador, instalaciones que muchas veces ya se habrán realizado anteriormente por su alto grado de uso hoy en día, pero del lado del servidor si necesitaremos realizar algunos pasos para conseguir la instalación para hacer disponibles los archivos a los clientes de nuestra red por medio del protocolo HTTP.

Debido a que por lo general la instalación sobre la plataforma Linux es mucho más difícil y tediosa que en Windows, nos enfocaremos más que nada en esta instalación, además de que se prefiere en este caso tener el servidor sobre Linux por ser una plataforma mundialmente reconocida por su estabilidad, la instalación de Windows se puede realizar de forma guiada y sencilla utilizando ejecutables correspondientes para Windows de los paquetes que vamos a nombrar a continuación para Linux.

Instalación del lado del cliente

1. Instalamos un navegador web, nosotros recomendamos y utilizamos el navegador multiplataforma Mozilla Firefox. Descargamos las fuentes o el RPM del navegador y lo instalamos ejecutando

   Modo RPM [12]:

   ```
   #rpm –ivh paquete_de_mi_navegador_web.rpm
   ```

   Ó;

   Modo con Fuentes:

Descomprimimos el archivo y ejecutamos dentro de la carpeta contenedora de los archivos descomprimidos

```
#./configure && make && make install
```

2. Instalamos el JRE (Java Runtime Environment) y le instalamos el plugin para el navegador web [26].

   Descargamos el instalador de JRE 1.5.0 o el del JDK 1.5.0 (que el segundo es de desarrollo y contiene al primero).
   Suponiendo que se desea instalar el JDK se escribe lo siguiente en consola:

```
# ./jdk-1_5_0_06-linux-i586-rpm.bin
```

   Al ejecutar esta línea aparecerá una licencia que debe ser aceptada, una vez aceptada se crea un rpm en la misma carpeta con el nombre jdk-1_5_0_06-linux-i586.rpm, el que debe ser instalado así:

```
# rpm -iv jdk-1_5_0_06-linux-i586.rpm
```

Instalación del lado del servidor

1. Instalación del servidor web Apache 2.0

   Descargamos las fuentes del Apache Web Server 2.0, descomprimimos su contenido, cambiamos a la carpeta contenedora y ejecutamos la siguiente secuencia de comandos [15]:

```
#./configure --prefix=/usr/local/apache2 --enable-so
#make
#make install
```

   El valor de prefix es la ruta donde estará ubicado físicamente el programa. Para iniciar, detener y reiniciar el Apache se ejecuta:

```
#/usr/local/apache2/bin/apachectl start
#/usr/local/apache2/bin/apachectl stop
# /usr/local/apache2/bin/apachectl restart
```

   Podemos realizar una primera prueba antes de realizar modificaciones de configuración abriendo una ventana del navegador web Mozilla y poniendo la URL http://localhost, si hemos hecho todo bien debe salir un mensaje de bienvenida de Apache.

Lo que sigue es la configuración del archivo httpd.conf, la ruta de este archivo, basándonos en la ruta que establecimos para instalar el Apache es "/usr/local/apache2/conf/httpd.conf" [15]. Se lo abre con un editor de texto, por ejemplo kwrite, para ejecutar kwrite sólo se escribe en consola:

```
#kwrite
```

Ya estando en modo de edición podemos cambiar la carpeta del servidor web de su valor predeterminado "/usr/local/apache2/htdocs" establecido en DocumentRoot a cualquiera que nosotros deseemos. El valor inicial es de:

```
DocumentRoot "/usr/local/apache2/htdocs"
```

Cambiaremos la ruta a "/servidor" que será la ruta del servidor web, es decir, cuando en el navegador se escriba la dirección "http://localhost" obtendremos el contenido de la carpeta "/servidor".

A la carpeta establecida en DocumentRoot también se le establece permisos de acceso en las directivas dentro de:

```
#
# This should be changed to
# whatever you set DocumentRoot to.
#
<Directory "/usr/local/apache2/htdocs" >
```

Se cambia aquí también el valor a "/servidor".

Necesitamos agregar otras líneas de configuración para que el Apache reconozca a PHP. Donde se vea esta sección:

```
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality
# of a module which was built as a DSO you
# have to place corresponding `LoadModule'
# lines at this location so the
# directives contained in it are actually
# available _before_ they are used.
# Statically compiled modules (those
# listed by `httpd -l') do not need
# to be loaded here.
#
```

```
# Example:
# LoadModule foo_module modules/mod_foo.so
```

Se escribe justo abajo y sin comentar:

```
LoadModule php5_module modules/libphp5.so
```

Cuando una carpeta dentro del servidor web se abre desde el navegador, si es que existe un archivo index.html, éste se muestra, así también necesitamos que el Apache reconozca los archivos index.php como el archivo predeterminado de presentación, para eso agregaremos index.php al final de la siguiente línea:

```
DirectoryIndex index.html index.html.var
```

Resultado:

```
DirectoryIndex index.html index.html.var index.php
```

Existe una sección en la que se establecen los tipos de archivo que existen y las extensiones que los archivos de ese tipo deben tener para ser tratados como tales:

```
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
```

Al final de esta sección se escribe:

```
AddType application/x-httpd-php .php .phtml .inc .js
AddType application/x-httpd-php-source .phps
```

Probamos el servidor Web Apache, lo iniciamos y abrimos el navegador web (el Mozilla Firefox viene por defecto en la versiones recientes de la mayoría de distribuciones Linux), y ponemos la siguiente dirección: http://localhost, también puede usarse en vez de localhost la IP 127.0.0.1 o la IP de la máquina que ésta tenga dentro de la red, la que se puede ver ejecutando:

ifconfig

El servidor debe responder mostrando el contenido de la carpeta /servidor

2. Instalación de PHP

   Descargamos, descomprimimos, para después con los siguientes comandos compilar e instalar PHP:

```
#./configure --prefix=/usr/local/php5 --with-
apxs2=/usr/local/apache2/bin/apxs
#make
#make install
```

En el comando que inicia con./configure puede agregarse todos los parámetros adicionales que se consideren necesarios dependiendo si la instalación de PHP se la vaya a utilizar también para algún otro propósito, pero para las necesidades de este sistema las líneas de arriba son suficiente [13].

Pasos para configurar php.ini:

1. En la raíz de la carpeta descomprimida de las fuentes de PHP encontramos dos archivos que pueden ser renombrados a php.ini y establecidos como el archivo de configuración de la instalación de PHP:
    - php.ini-dist.- Es el archivo de configuración que establece el mínimo de seguridad para la instalación, este archivo es el recomendado en el caso de desarrollo
    - php.ini-recommended.- Es el archivo con el índice máximo de seguridad, este archivo es recomendado usar en modo de producción

Como nuestra instalación es de tipo producción usaremos el archivo php.ini-recommended, que lo renombraremos a php.ini, no sin antes haber hecho un respaldo de este archivo ya que vamos a realizar algunas modificaciones sobre él antes de la puesta en producción.

2. La primera modificación que se hace es cambiar las limitaciones de uso de recursos de sus valores predeterminados:

   En el sistema hay ocasiones en que se necesita consumir mucha memoria, como para crear un thumbnail (reducción) de una imagen, también se necesitará subir archivos muy grandes como videos por lo tanto se necesitará un tiempo de ejecución máximo así también bastante elevado para que el script (algoritmo de ejecución web en lenguaje php) no se llegue a interrumpir en medio de una subida de archivo o algo parecido.

   Los nuevos valores serán:

```
max_execution_time = 1000
max_input_type = 1000
memory_limit = 20M
```

3. Otra modificación indispensable es en la sección de Manejo de Datos, permitiendo subir grandes cantidades de Datos, estableceremos las variables citadas a los siguientes valores:

```
;;;;;;;;;;;;;;;;
; Data Handling ;
;;;;;;;;;;;;;;;;
.
.
.
; Maximum size of POST data
; that PHP will accept.
post_max_size = 100M
```

4. La siguiente modificación que haremos es en la sección de Subida de Archivos, la que debe quedar así:

```
;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;
; Whether to allow HTTP file uploads.
file_uploads = On
```

```
; Temporary directory for HTTP uploaded
; files (will use system default if not
; specified).
;upload_tmp_dir =
; Maximum allowed size for uploaded files.
upload_max_filesize = 100M
```

5. Hay directorios cuyo contenido (que por lo general son librerías) deben estar disponibles a muchos scripts, se debe modificar la sección de “Rutas y Directorios” de tal forma en que quede de la siguiente manera, se debe tener cuidado ya que como se puede apreciar, se debe modificar la sección de UNIX, y no la de Windows cuyo contenido en este caso no interesa.

```
;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;
; UNIX: "/path1:/path2"
include_path =
".:/servidor/micomisariato/include:/servidor/mi
comisariato/app/lib"
```

```
; Windows: "\path1;\path2"

;include_path = ".;c:\php\includes"
```

6. El archivo PHP según los parámetros en nuestra instalación debe ser ubicado manualmente en la ruta "/usr/local/php5/php.ini".

4.3. Procedimientos de utilización

Para cumplir con el objetivo principal de escanear un documento y almacenarlo en el CMS para su posterior presentación al público se debe seguir los siguientes pasos:

Comenzamos por realizar inicio de sesión en el sistema con algún usuario y su clave en el CMS Joomla.

Figura 4.1. Pantalla de acceso a la consola de administración del CMS Joomla

En el menú principal damos clic en Components -> HeXimage -> Acquire and Create Thumb

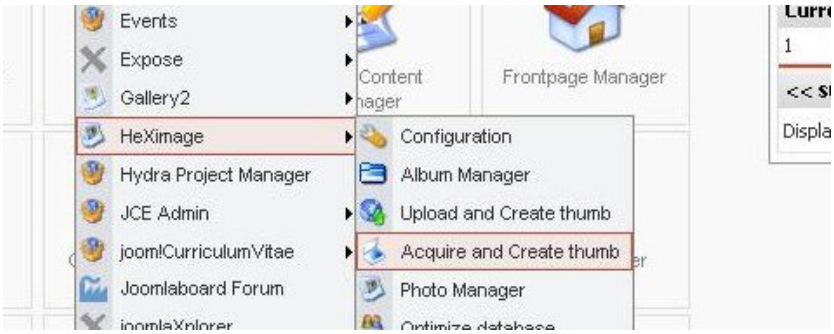


Figura 4.2. Ruta a la funcionalidad de adquisición de imágenes

Se debe esperar la carga del applet Java de captura de imágenes

Figura 4.3. Carga del applet Java de captura de imágenes

Luego, se debe aceptar el mensaje preventivo que aparece porque el applet tiene código de utilización de recursos del sistema como son el escáner y el sistema de archivos para almacenar la imagen previa a subirla al sistema.

Figura 4.4. Mensaje de advertencia de firma digital de aplicación no verificada

Entonces aparecerá la interfaz gráfica principal del sistema para su utilización

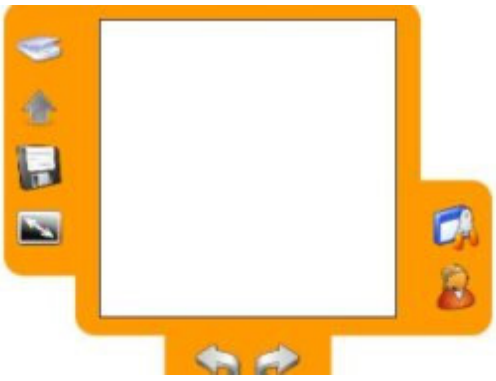


Figura 4.5. Interfaz gráfica principal del cliente digitalizador de imágenes

Si se da clic en el botón inferior derecho aparecerá la Ayuda en un panel al lado derecho del panel principal con preguntas frecuentes sobre la herramienta.

Figura 4.6. Panel principal de cliente digitalizador de imágenes con su panel de ayuda a la derecha

Al dar clic sobre el botón de Selección de Dispositivo de Adquisición de Imágenes aparecerá el siguiente diálogo:

Figura 4.7. Cuadro de diálogo de selección de dispositivo de captura de imágenes

Ya seleccionado el dispositivo, se puede proceder a dar clic en el botón de Captura de imagen y aparecerá el siguiente cuadro de diálogo (Si el sistema operativo en uso es Linux, el cuadro de diálogo que aparecerá es el de la Figura 3.5):

Figura 4.8. Cuadro de diálogo de captura de imagen en sistema operativo Microsoft Windows XP

Al digitalizar la imagen, un thumbnail (reducción) de la imagen aparecerá en el centro de la interfaz gráfica principal de la herramienta, como se muestra en la siguiente figura:

Figura 4.9. Interfaz gráfica principal de la herramienta de digitalización con un documento escaneado

Ya teniendo la imagen escaneada se habilitarán los botones para subir la imagen y para guardar un respaldo de la imagen en la estación de trabajo. Al dar clic sobre el botón de subir la imagen aparecerá un cuadro de diálogo pidiendo información pertinente para su almacenamiento en la base de datos y facilitar su posterior búsqueda.

Figura 4.10. Cuadro de diálogo solicitando información sobre la imagen antes de subirla al CMS en un álbum del componente de la galería HeXimage

Para confirmar que la imagen ha sido subida con éxito, aparecerá un mensaje a lo que la herramienta termine de subirla, pero también se podrá revisar su existencia en el sistema revisando el Administrador de Imágenes, donde se encontrará listada, si se decide publicar la imagen, ésta se hará disponible al público a través de la interfaz principal del CMS.

Figura 4.11. Administrador de imágenes del componente HeXimage del CMS Joomla

Una vez publicada la imagen, se puede verificar que esté disponible a los visitantes del sitio web del CMS, dando clic sobre el enlace al componente de la Galería de Imágenes se tendrá acceso a éste, y después dando clic sobre el álbum y sobre el thumbnail (imagen reducida) del documento se podrá

visualizarla en una ventana de AJAX que puede ser maximizada para su mejor apreciación.

Figura 4.12. Imagen capturada en su tamaño real en ventana de AJAX a lado de su thumbnail (reducción), el que se requiere dar clic para su aparición

4.4 Pruebas de la aplicación

Ya hemos mostrado qué problema trata de resolver esta tesis, también cómo escogimos atacar el problema y hemos mostrado paso a paso la implementación, instalación y el procedimiento de utilización del sistema Pero, ¿qué tan eficiente es el proyecto?, si el sistema no tiene una respuesta apropiada en un tiempo apropiado, aunque éste pueda cumplir su objetivo no será tomado en cuenta o por lo menos no con seriedad por los usuarios. ¿Qué tan fácil es de usar el sistema?, la funcionalidad y facilidad de interacción de un sistema es muy importante, si el usuario tiene que realizar muchos pasos para llegar a una tarea muy básica o de uso frecuente, su utilización se volverá muy aburrida, si los controles para realizar una tarea

están escondidos o son de difícil acceso, el usuario no podrá explotar toda la capacidad del sistema. Nos enfocaremos principalmente en el tiempo de respuesta de la aplicación, y su funcionalidad y usabilidad, además de un factor muy importante como lo es la seguridad del sistema. En algunos casos ciertas pruebas demuestran que algunas funcionalidades necesitan re-implementarse o re-diseñarse. Estas acciones tomadas a partir de pruebas poco satisfactorias también las hemos documentado en esta parte de la documentación.

4.4.1 Tiempo de respuesta

En el caso de la Galería de Imágenes recordemos que esta funcionalidad ya la entrega la extensión básica del CMS seleccionada llamada HeXimage, después de revisarla vimos que cumplía su tarea con ciertos problemas. Este componente mostraba todas las imágenes en miniatura en la página inicial de la Galería, además de un campo de selección para escoger el álbum a visualizar, lo que acarreaba un problema, si durante las pruebas el número de imágenes a contener la galería era muy grande, el tiempo de carga de tantos thumbnails (imágenes en miniatura) se iba incrementando con el pasar del tiempo, llegando seguramente en algún momento a ser insostenible, además de que no le encontrábamos mucho sentido a esa funcionalidad en la página de inicio, por eso decidimos retirarla porque iba en desmedro del tiempo de respuesta.

La principal preocupación siempre fue entorno al Digitalizador de Imágenes, ya que utiliza un applet en Java, que en un principio tomaba mucho tiempo en cargar debido a su gran tamaño en disco, aproximadamente 500 KB (este tamaño puede parecer pequeño pero no para descargarlo de un ambiente web), se logró minimizar lo que más pudimos el contenido del archivo JAR (formato de los applets en Java), quitando iconos y demás, dejando solo el núcleo de las funcionalidades indispensables en él, pasando todo lo demás a la interfaz del CMS, logrando una disminución del 30% aproximadamente. Igual siempre la primera vez que se ejecuta el applet se demora en cargar un tiempo aproximado de 10 segundos, desde la siguiente vez este tiempo disminuye dramáticamente ya que el applet es almacenado en caché de la Máquina Virtual de Java en la estación de trabajo. También tenemos que pensar que aunque el sistema puede instalarse en un servidor en Internet, está pensado principalmente para una red de área local (LAN), es decir la intranet empresarial.

En el sistema el tiempo que toma escanear una imagen no es mayor ni menor que el tiempo que le tomaría a cualquier otra aplicación, ya que para escanear todas obedecen a las mismas reglas de comunicación con la interfaz de captura de imágenes dependiendo de la plataforma que se esté utilizando.

4.4.2 Funcionalidad

Este análisis se enfocó en la parte del Cliente Digitalizador de Imágenes y en la sección de la Galería de Imágenes, que son las partes que cumplen las veces de Interfaz Gráfica para el usuario, ya que éstas son las partes más vulnerables del sistema, especialmente el Cliente digitalizador de imágenes, que realiza tareas poco comunes para una aplicación web, lo que puede llevar a ciertas situaciones de incertidumbre para el usuario. En el caso del Receptor de Imágenes, el análisis no se trata tanto de funcionalidad, sino de tiempo de respuesta, como ya lo revisamos anteriormente.

En el componente del CMS de la Galería de Imágenes utilizado, como funcionalidad nativa las imágenes se mostraban en ventanas del navegador web que se abrían al dar clic en el thumbnail (imagen en miniatura) de la imagen solicitada. Estas nuevas ventanas sólo contenían la imagen seleccionada y se abrían del tamaño de dicha imagen, era una funcionalidad muy básica y un tanto aburrida, además de que cambiaba el enfoque visual del usuario de forma abrupta. Por otro lado los thumbnails se mostraban sin el texto descriptivo asociado a cada imagen, de tal manera de que lo que veíamos de forma reducida además de que no se podía distinguir apropiadamente por su tamaño, no teníamos un texto que nos guíe a cual imagen dar clic.

Decidimos entonces utilizar un Marco de Trabajo de AJAX llamado Dojo [18] para hacer más dinámica y atractiva la interacción del usuario en la Galería de Imágenes, que es una característica primordial en este tipo de herramientas multimedia, cambiamos el uso de ventanas del navegador web por ventanas tipo AJAX, además agregamos tooltips (consejos de herramientas) a cada thumbnail de imagen con su texto descriptivo para facilitar su reconocimiento y evitar que sea seleccionado por equivocación.

Por otro lado comenzamos a probar el Digitalizador de Imágenes implementado como applet de Java embebido en una página web como parte de las funciones de la Galería de Imágenes en la Interfaz de Administrador del CMS.

En un principio habíamos decidido colocar todas las opciones de escaneo de imágenes en forma de menú y también en forma de una única barra de herramientas con iconos a todas las opciones, después, por medio de las pruebas que realizamos basándonos en el artículo “Success Rate: The Simplest Usability Metric” de Jakob Nielsen [20], nos dimos cuenta de que hacerlo de esta forma restaba facilidad de utilización al usuario.

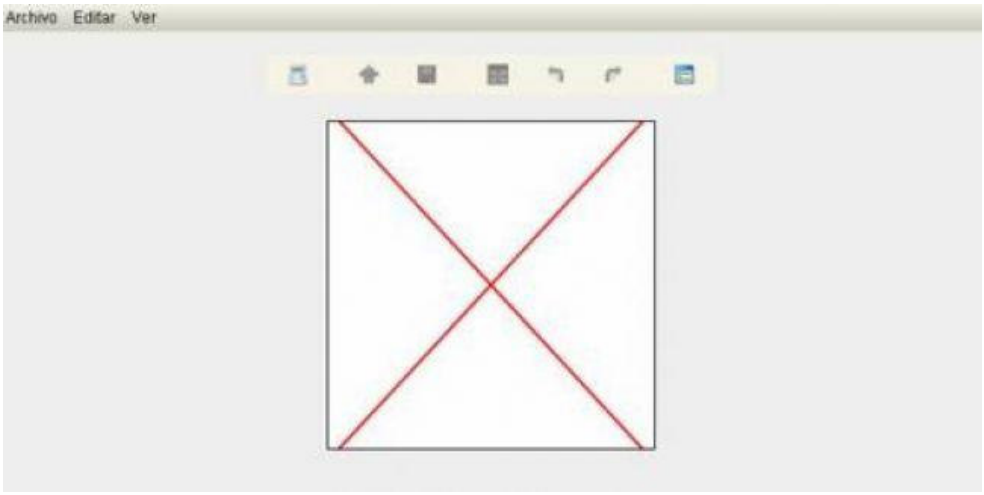


Figura 4.13. Interfaz gráfica principal que inicialmente tuvo el cliente digitalizador de imágenes

La prueba que realizamos la hicimos con cuatro usuarios (según Nielsen la cantidad de usuarios para una prueba cualitativa es suficiente que sea entre 3 y 5 [21]), a quienes les pedimos que realizaran las mismas tareas y llenamos una tabla con los resultados que fueron el nivel de éxito que cada uno tuvo en cada una de ellas.

Las tareas realizadas fueron las siguientes:

Tarea 1: Seleccionar dispositivo de captura de imágenes

Tarea 2: Escanear una imagen

Tarea 3: Visualizar la imagen en tamaño completo

Tarea 4: Rotar la imagen para ponerla en posición que el texto del documento sea legible

Tarea 5: Subir la imagen al CMS

Tarea 6: Guardar un respaldo de la imagen

Niveles de éxito: E = Éxito, F = Fallido, P = Éxito Parcial

Equivalencias de niveles de éxito: E = 1, F = 0, P = 0.5

Cálculo de taza de éxito: TazaExito = (#TareasExitosas*1 + #TareasExitoParcial*0.5 + #TareasFallidas) / #TotalTareasEjecutadas

	Tarea 1	Tarea 2	Tarea 3	Tarea 4	Tarea 5	Tarea 6
Usuario 1	E	P	E	P	F	F
Usuario 2	F	E	P	E	P	E
Usuario 3	E	E	E	P	E	F
Usuario 4	F	F	E	E	E	E

Tabla III. Nivel de éxito de usuarios por tarea en primera prueba de funcionalidad del sistema

Taza de éxito = ((1)*(13) + (0.5)*(5) + (0)*(6)) / 24 = **64.58%**

Se tomó en cuenta que el simple hecho de tener un menú en el applet ya era un tipo de ambigüedad, ya que el propio navegador web tiene su menú, y el CMS también tiene otro. Agregar otro menú haría confundir al usuario; utilizar una sola barra de herramientas en Java también era un problema, ya que no

todas las opciones realizaban tareas relacionadas. No podíamos poner una opción de Ayuda y de Escaneo de Imagen en la misma barra, además de que el applet de Java acarrea de por sí el problema de que en el aspecto visual se lo percibe como una parte sobrepuesta de la aplicación; tal vez ésta sea una razón clave de porqué su uso se ha discontinuado con el pasar de los tiempos, lo que no le ha sucedido a Flash cuyas aplicaciones sí se ven elegantemente como parte de un todo en una aplicación web. Decidimos entonces que la parte visual del applet se enfoque exclusivamente a la presentación de la imagen escaneada, retirando el menú y la barra de herramientas de Java, nos ingeniamos para desde JavaScript (AJAX) poder llamar a las funciones de escaneo de Java, así pudimos hacerlo más interactivo, utilizamos dos barras de herramientas de tipo Fisheye (como el menú de Mac OS), clasificando así las opciones, dejando las principales y más básicas más visibles, la Ayuda ya no se abre dentro de una ventana de Java sino elegantemente dentro de la misma Interfaz de Administración.

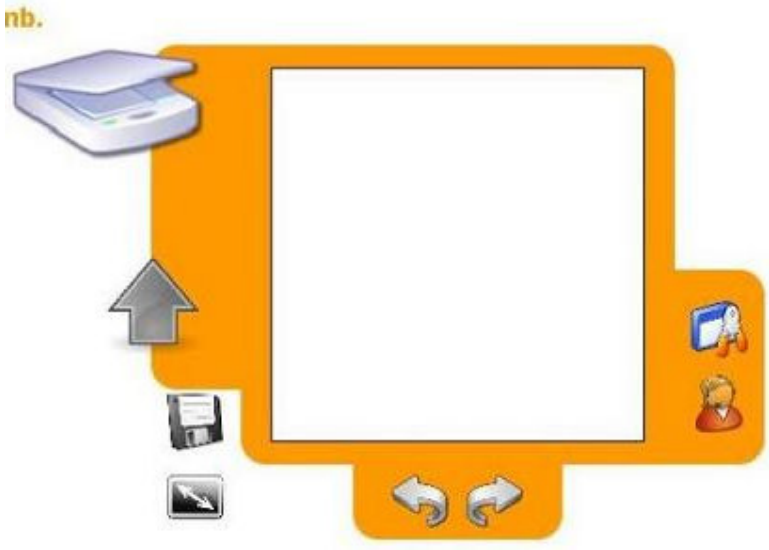


Figura 4.14. Interfaz gráfica principal que finalmente tuvo el cliente digitalizador de imágenes

Volvimos a realizar las pruebas con las mismas tareas y la misma cantidad de usuarios, los resultados se encuentran en la siguiente tabla:

	Tarea 1	Tarea 2	Tarea 3	Tarea 4	Tarea 5	Tarea 6
Usuario 1	E	E	E	E	E	E
Usuario 2	E	E	E	E	E	F
Usuario 3	E	E	E	P	P	E
Usuario 4	P	P	E	E	F	E

Tabla IV. Nivel de éxito de usuarios por tarea en segunda prueba de funcionalidad del sistema

Taza de éxito = ((1)*(18) + (0.5)*(4) + (0)*(4)) / 24 = **83.33%**

Los usuarios nos comentaron haber encontrado los controles de forma más intuitiva y estratégicamente ubicados, sintieron más confianza al realizar las tareas, lo que se reflejó en el resultado general de la nueva sesión de pruebas.

4.4.3 Seguridades de red y de almacenamiento

Después del desarrollo de nuestra aplicación decidimos instalarla en línea en un servidor en Internet, para lo que pagamos por el servicio de hosting (hospedaje) a la compañía ecuaweb.com. Tuvimos el sistema corriendo durante aproximadamente 3 meses, mientras hacíamos pruebas de instalación y de programación que sólo en un ambiente en vivo podíamos realizar. Muchas veces un sistema funciona bien en un ambiente local, pero cuando se lo instala en Internet notamos que hay algunas referencias mal colocadas y que llevan a páginas que no existen o enlaces con una ruta no existente.

Mientras la aplicación web estuvo en Internet fue hackeado (pirateado) en dos ocasiones seguidas, al parecer por el mismo hacker aparentemente proveniente de Turquía, después de una exhaustiva investigación de las

razones, nos dimos cuenta de que el error había sido el haber utilizado una versión de Joomla (la versión 1.0.12) que aunque era estable no era la versión de seguridad recomendada por Joomla (la versión 1.0.13) liberada al público en alrededor de 200 días después que la anterior, en la que se arreglaban algunos errores de seguridad. También como se recomienda en la "Lista de Chequeo del Administrador de Joomla" [22] en el sitio oficial de Joomla, cambiamos la ruta al archivo de configuración tan importante llamado "configuration.php" que inicialmente y por defecto se encuentra en la raíz de la carpeta de instalación de Joomla, lo pusimos fuera de la carpeta public_html y en vez de él colocamos simplemente un enlace a él hecho en PHP, un archivo con el mismo nombre que lo único que hace es incluirlo al archivo configuration.php, a este archivo de solo enlace le cambiamos los permisos de acceso de usuario UNIX a 440, como también se lo recomienda en el sitio oficial de Joomla, para que si alguien tiene acceso al sistema de archivos no pueda acceder al contenido de este archivo o a modificarlo.

El archivo "configuration.php" contiene información muy sensible, como son usuarios y claves de la base de datos MySQL que usa Joomla, usuario y clave de Super Administrador de la instalación de Joomla, rutas de instalación, configuración de la instalación, etc.

Otra recomendación que seguimos fue la de deshabilitar la opción de Emulación de REGISTER GLOBALS en la configuración de Joomla, la que existe para compatibilidad con extensiones viejas que usaban esta opción de PHP, que fue quitada de la instalación por defecto por los propios desarrolladores de PHP exactamente por sus problemas de seguridad.

Además desinstalamos cualquier extensión de Joomla que no fuese indispensable en el sistema para el desarrollo de la tesis, ya que una extensión mal programada podría significar un agujero de seguridad también.

Chequeamos los permisos de todos los archivos dentro de la carpeta de instalación del sistema completo, las carpetas deben tener los permisos UNIX en 755 y los permisos de los archivos deben estar en 644.

Comprobamos con un pequeño programa si había archivos creados o modificados después de la fecha del ataque para ver si el hacker no había dejado archivos que le aseguraran un futuro y fácil ingreso al sistema aunque hayamos realizado todos estos cambios. Nos dimos cuenta de que no había ninguno.

El siguiente paso que seguimos fue cambiar los usuarios y claves de MySQL, nuestra instalación de Joomla, FTP, entre otros porque sabíamos que el hacker había tenido acceso a toda esta información a través de los archivos de configuración.

La “Lista de Seguridades por chequear” de Joomla que revisamos también recomienda escoger con mucho cuidado el servicio de hosting que vamos a utilizar para Joomla, porque éste debe cumplir consideraciones mínimas en la configuración de PHP, MySQL, entre otros para la instalación de Joomla.
Una buena recomendación es realizar respaldos periódicos de toda la información en la aplicación para no perder nada en caso de un intruso mal intencionado tenga acceso al sistema. Otra buena recomendación es que los usuarios cambien sus claves con frecuencia, y utilicen claves difíciles de adivinar o piratear. También es bueno deshabilitar la cuenta FTP si no se la utiliza, porque puede convertirse en un agujero de seguridad.

Es mejor no tener que aprender a ser cuidadoso con las seguridades cuando ya el problema se ha dado y tal vez sea irreparable, mejor es tener cuidado con los programas que podamos poner en producción, y probarlos hasta la saciedad en un ambiente controlado local.

Por lo general un “hacker” (pirata) vuelve a tratar de piratear un sistema que ha pirateado antes, como nos sucedió la segunda vez que fue pirateado el sistema, pero después de seguir la Lista de Recomendaciones de Seguridades al pie de la letra no se ha vuelto a tener problema con ningún intruso en el sistema.

Conclusiones y recomendaciones

- La aplicación finalmente logró su objetivo de digitalizar imágenes de documentos en las plataformas Windows y Linux, almacenar las imágenes digitalizadas en la base de datos de un servidor al que tenga acceso el digitalizador de imágenes a través de la red y, visualizar y administrar las imágenes digitalizadas de los documentos ya almacenadas desde una aplicación web centralizada. Se mejoró la organización de los documentos por medio de métodos de búsqueda fáciles de utilizar, además de la integración del proyecto con un Sistema de Manejo de Contenido (CMS).
- La principal recomendación sería obtener un estándar de captura de imágenes que funcione sin importar la plataforma, un estándar que enfatice en las virtudes comprobadas de las interfaces TWAIN (Windows) y SANE (Linux), y de no ser así que los fabricantes de hardware de escaneo de imágenes ofrecieran drivers (software de instalación de hardware) no solo para la plataforma reinante en el mercado, la que actualmente es Windows sino para todas las principales al menos.
- Otra recomendación importante sería que Sun Microsystems haga disponible un API estandarizado también para su uso en la captura de Imágenes y no solo para la manipulación de éstas. Los API's que

ofrece Sun Microsystems se enfocan muy poco en el uso de periféricos como el escáner.

REFERENCIAS BIBLIOGRÁFICAS

[1] “About TWAIN” en Sitio Oficial de TWAIN. Diciembre del 2006. Disponible en http://www.twain.org/abouttwain.shtm

[2] “TWAIN Frequently Asked Questions” en Sitio Oficial de TWAIN. Diciembre del 2006. Disponible en http://www.twain.org/faqs.shtm

[3] “SANE - Introduction” en Sitio Oficial de SANE. Disponible en http://www.sane-project.org/intro.html

[4] “SANE Frequently Asked Questions” en Sitio Oficial de SANE. Disponible en http://www.xs4all.nl/~ljm/SANE-faq.html

[5] Artículo “Joomla's 100k Ain't Bad!” en Sitio Oficial de Joomla. Abril del 2007. Disponible en http://www.joomla.org/content/view/3134/1/

[6] “Joomla and Drupal - Which One is Right for You?”. Abril del 2007. Disponible en http://www.alledia.com/blog/general-cms-issues/joomla-and-drupal-%11-which-one-is-right-for-you?/

[7] “PHP vs ASP vs JSP Trend History” en Google Trends. Agosto del 2007. Disponible en http://www.google.com/trends?q=PHP%2CASP%2CJSP

[8] “Joomla vs Drupal Trend History” en Google Trends. Agosto del 2007. Disponible en http://www.google.com/trends?q=joomla%2C+drupal

[9] “PHPEclipse User Manual”. Febrero 2007. Disponible en http://docs.schuetzengau-freising.de/modules/xdocman/index.php?doc=xo-002&lang=en

[10] “Subclipse Frequently Asked Questions”. Febrero 2007. Disponible en http://subclipse.tigris.org/faq.html

[11] “OpenSSL Frequently Asked Questions” en Sitio Oficial de OpenSSL. Enero 2007. Disponible en http://www.openssl.org/support/faq.html

[12] “Using RPM to install packages”. Enero 2007. Disponible en http://www.rpm.org/max-rpm/ch-rpm-install.html

[13] “Instalación y Configuración de PHP” en Sitio oficial de PHP. Diciembre 2006. Disponible en http://www.php.net/manual/es/install.php

[14] “Manual de PHP” en Sitio Oficial de PHP. Junio del 2007. Disponible en http://www.php.net/manual/es/

[15] “Apache installation and configuration”. Diciembre 2006. Disponible en http://www.freeos.com/articles/2739/

[16] “How to install USB Scanner” en Gentoo Wiki. Diciembre 2006. Disponible en http://gentoo-wiki.com/HOWTO_Installing_USB_Scanner

[17] “To test a scanner or digital camera” en Sitio Oficial de Microsoft. Diciembre 2006. Disponible en http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/scanner_camera_add.mspx?mfr=true

[18] “The Dojo Book, 0.4” en Sitio Oficial de la librería para AJAX Dojo. Febrero del 2007. Disponible en http://dojotoolkit.org/book/dojo-book-0-4

[19] “Dojo 0.4x API Tool” en Sitio Oficial de la librería para AJAX Dojo. Febrero del 2007. Disponible en http://dojotoolkit.org/api

[20] “Success Rate: The Simplest Usability Metric” en Jakob Nielsen's Website. Febrero del 2001. Disponible en http://www.useit.com/alertbox/20010121.html

[21] “Why you only need to test with five users” en Jakob Nielsen's Website. Marzo del 2000. Disponible en http://www.useit.com/alertbox/20000319.html

[22] “Joomla! Administrator's Security Checklist” en Sitio Oficial de Joomla. Abril del 2007. Disponible en http://help.joomla.org/component/option,com_easyfaq/task,view/id,167/Itemid,268/

[23] “mm’s computing – Package scanner”. Octubre del 2006. Disponible en http://www.mms-computing.co.uk/uk/co/mmscomputing/device/scanner/index.php

[24] “Zend Studio: The Proven PHP Development Environment”, en Sitio Oficial de PHP. Octubre del 2006. Disponible en http://www.zend.com/products/zend_studio

[25] “About the Eclipse Foundation” en Sitio Oficial de Proyecto Eclipse. Marzo del 2007. Disponible en http://www.eclipse.org/org/

[26] “Cómo configurar Java JDK en Fedora”. Marzo del 2007. Disponible en http://www.fedora-es.com/node/248.

[27] MySQL 5.0 Reference Manual en español, Capítulo 1, “Información general” en Sitio Oficial de MySQL. Marzo del 2007. Disponible en http://dev.mysql.com/doc/refman/5.0/es/introduction.html

[28] “MySQL vs. SQL Server vs. PostgreSQL” en Google Trends. Febrero del 2007. Disponible en http://www.google.com/trends?q=mysql%2C+sql+server%2C+postgreSQL

[29] “Morena 6 product overview” en el Sitio oficial de Morena. Marzo del 2007. Disponible en http://www.gnome.sk/Twain/jtp.html

[30] “Joomla Extensions – Gallery Category” en Sitio Oficial del CMS Joomla. Noviembre del 2006. Disponible en http://extensions.joomla.org/index.php?option=com_mtree&task=listcats&cat_id=1779&Itemid=35

[31] “LAMP (software bundle)” en Wikipedia. Agosto del 2006. Disponible en http://en.wikipedia.org/wiki/LAMP_(software_bundle)

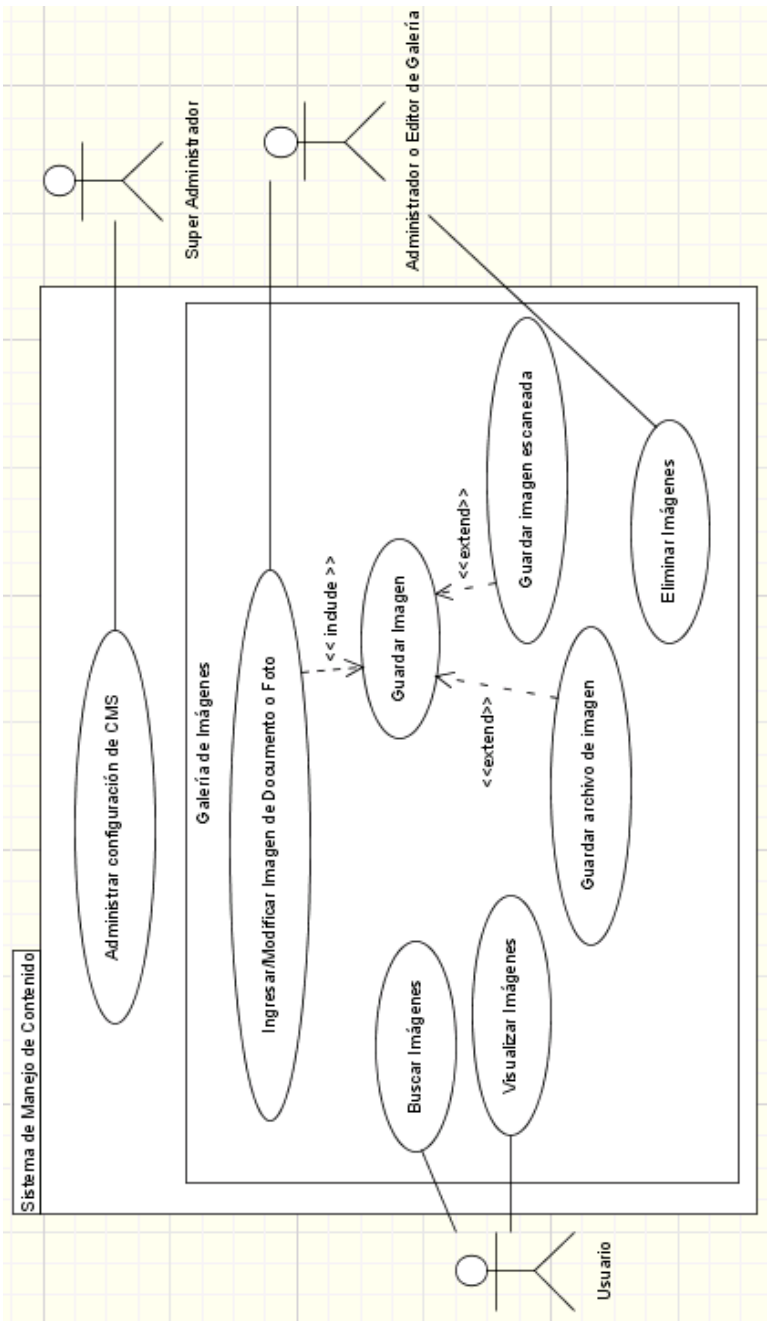
[32] “SANE Documentation - Image Data Format” en Sitio Oficial de SANE. Enero del 2006. Disponible en http://www.sane-project.org/sane2/0.07/doc008.html

[33] “CMSs not here” en OpenSourceCMS.com. Julio 2006. Disponible en http://www.opensourcecms.com/index.php?option=com_content&task=view&id=2132&Itemid=190

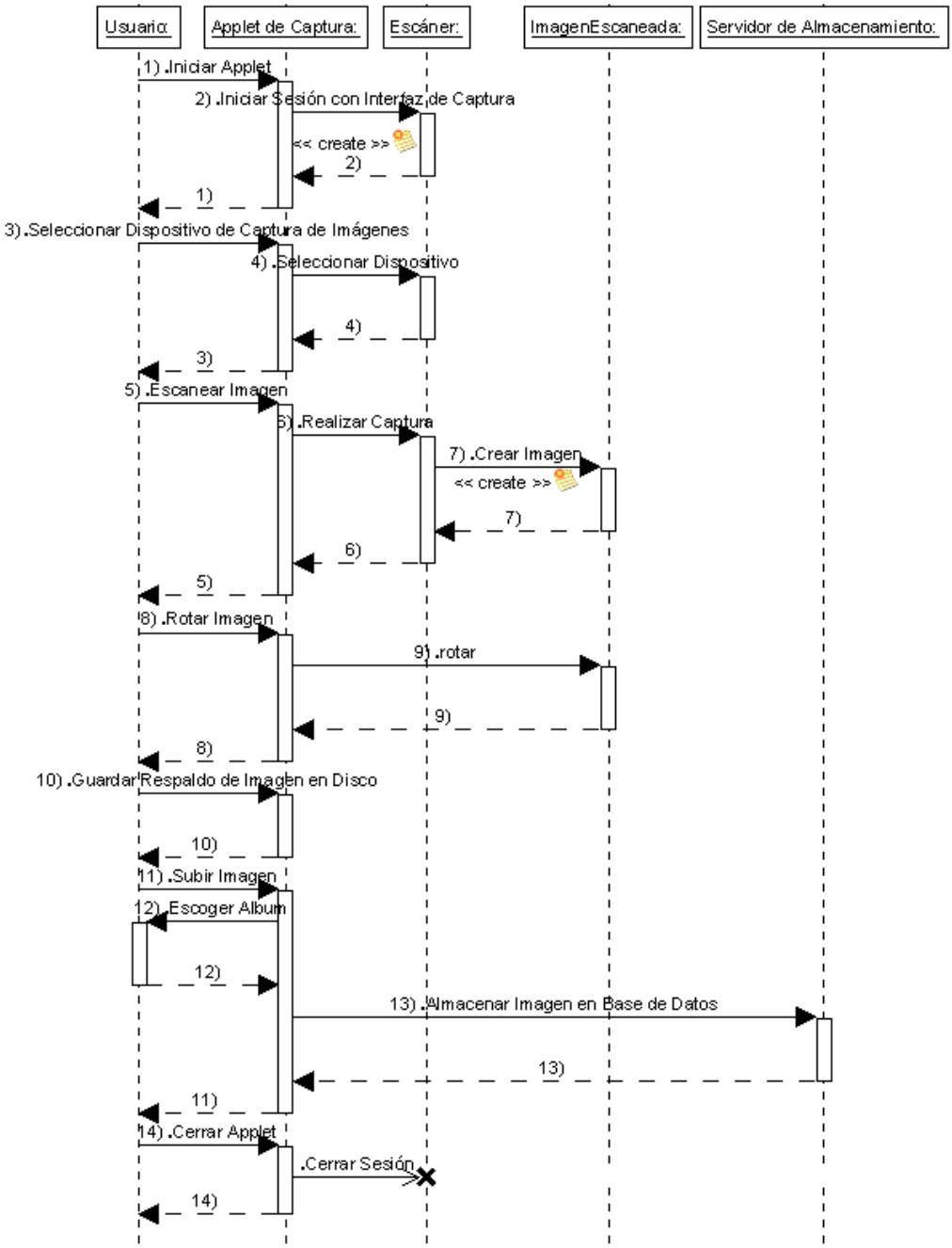
[34] Artículo “Joomla is the new Mambo” en devshed.com por Norbert Cartagena. Septiembre 2005. Disponible en http://www.devshed.com/c/a/BrainDump/Joomla-is-the-New-Mambo/

APÉNDICES

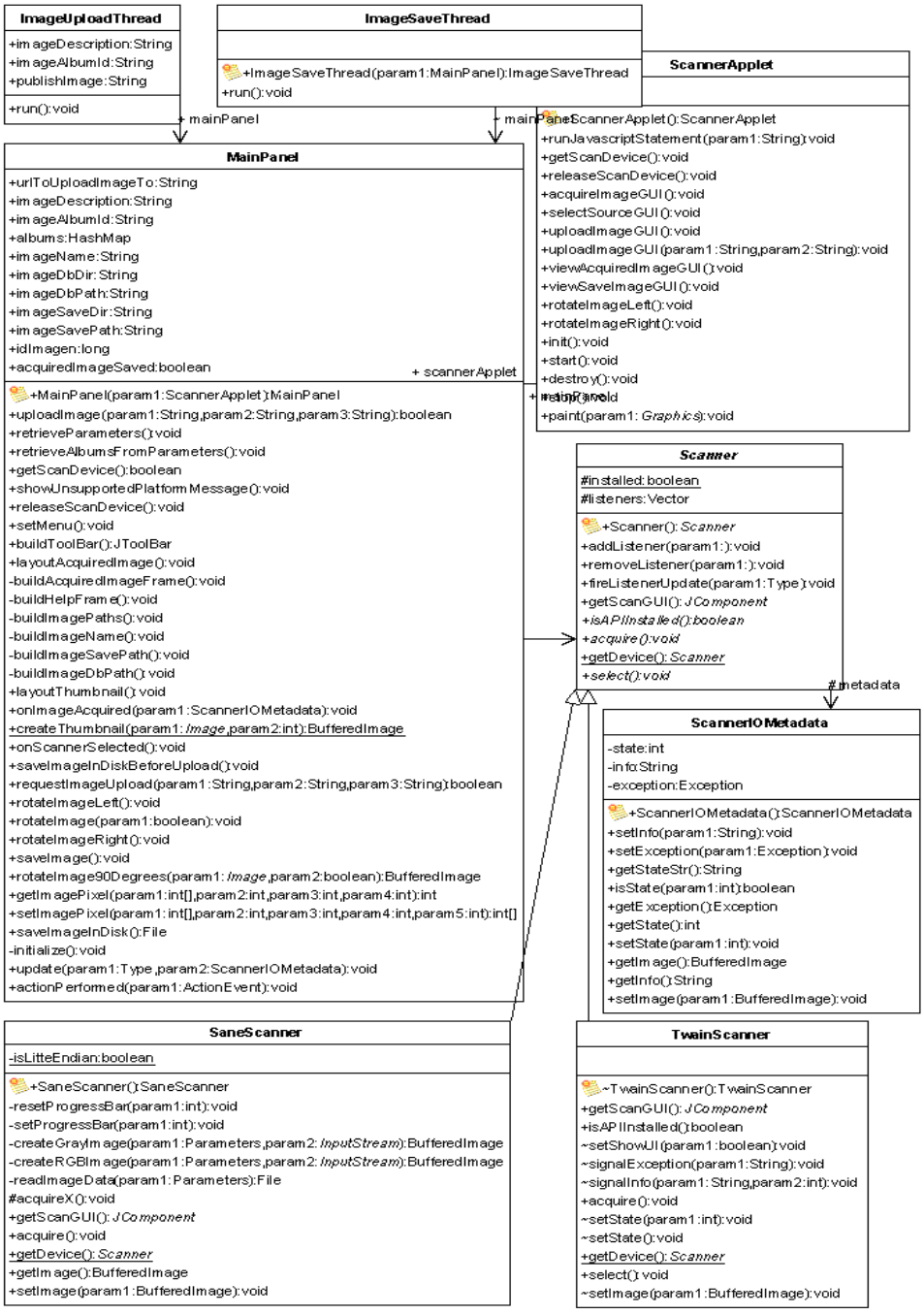
Apéndice A: Diagrama de casos de uso general del sistema



Apéndice B: Diagrama de secuencia de la digitalización y almacenamiento de una imagen



Apéndice C: Diagrama de clases de cliente digitalizador de imágenes (entidades y procesos principales)



Apéndice D: Instalación de un escáner EPSON PERFECTION 2400 PHOTO en Fedora Linux 4

Primero ejecutamos:

```
# /sbin/lsmod
```

para ver la lista de dispositivos instalados en la computadora, y estar seguro de que el escáner no ha sido instalado previamente. Después para instalar el escáner ejecutamos los siguientes comandos en consola:

```
# /sbin/insmod usbcore
# /sbin/insmod scanner vendor=0x05b8 product=0x011b
```

Después descomentamos la última línea que dice "usb0" en el archivo /etc/sane.d/epson.conf [16]. Si los pasos previos no instalan el escáner ejecutamos:

```
# /sbin/rmmod scanner
```

Este comando se ejecuta para remover el escáner que ocupa esa ubicación y proseguimos a intentar los pasos anteriores de nuevo.

Es necesario que se tenga instalado SANE en el computador para instalar un escáner si la plataforma es Linux, para chequear que Sane esté instalado como RPM, se ejecute en consola

```
# rpm –qa
```

Esto se realiza para revisar la lista de rpm's instalados y saber de la existencia de Sane en el equipo.