

INTRODUCCIÓN

El desarrollo de la tecnología digital ha crecido velozmente y ha permitido mejorar muchos procesos en distintos ámbitos de la sociedad.

Desafortunadamente, en el ámbito de las instituciones de gobierno de los distintos niveles (federal, estatal y municipal), el uso de las tecnologías de la información ha sido, en mi particular punto de vista, muy lento respecto a la evolución de las mismas, lo cual ha derivado en servicios deficientes para la población e incluso en toma de decisiones equivocadas o no oportunas.

Afortunadamente, la misma rapidez del crecimiento tecnológico, ha obligado a muchas autoridades a poner atención en este aspecto, y han implementado programas de modernización.

Precisamente, uno de los programas de modernización implementados por las autoridades, implicó la creación de un software para consultar los expedientes catastrales de manera digital en el Instituto Catastral del Estado de Oaxaca (ICEO).

El desarrollo del Sistema de Gestión de Expedientes Digitales (SIGED) del ICEO, el cual es el objeto del presente trabajo, en mi opinión, es un excelente ejemplo para ilustrar las distintas etapas de construcción de software, ya que se aplicaron las técnicas que en su momento fueron elegidas como las más adecuadas, y porque muestra el impacto positivo del factor humano que forma parte del proceso de desarrollo cuando se logra una alta integración.

El SIGED, es uno de los pocos sistemas de una institución de gobierno, que me atrevo decir, está “bien hecho”, y que sin temor a equivocarme, las autoridades del gobierno estatal y del propio instituto involucradas en su desarrollo pueden presumir como un logro de la administración.

A manera personal, el SIGED representa una gran experiencia profesional como Ingeniero en Sistemas Computacionales, y es uno de los sistemas que forman parte de mi “carta de presentación”, y que por eso fue mi elección como trabajo para obtener el grado.

En el capítulo 1 se presentan los antecedentes del problema, los justificantes de la propuesta de solución, los objetivos a alcanzar y la metodología a seguir.

El capítulo 2 aborda toda la parte conceptual considerada para el desarrollo de la aplicación.

La información generada del análisis y diseño del sistema se describe en el capítulo 3.

El capítulo 4 nos muestra las interfaces del sistema y el código escrito como parte del desarrollo, aclarando que sólo se incluyeron los correspondientes a los módulos de Definir Dispositivos de Escaneo y Escanear Expediente.

Por último, se presentan las conclusiones del proyecto, las cuales enuncian las metas alcanzadas y las sugerencias de mejoras.

En la parte de anexos, se agregan los fundamentos legales considerados y el código completo de los módulos mencionados arriba.

CAPÍTULO 1. PROBLEMATIZACIÓN

1.1. Planteamiento del problema

El ICEO (Instituto Catastral del Estado de Oaxaca) es una entidad del Gobierno del Estado de Oaxaca, perteneciente a la Secretaría de Finanzas, la cual se encarga de registrar, además de mantener, la información cualitativa y cuantitativa de los bienes inmuebles que se encuentran dentro del territorio del Estado.

Para realizar dicha tarea, el ICEO ofrece un conjunto de trámites y servicios, con los que los usuarios, particulares o inmobiliarios, pueden integrar o actualizar los datos de sus propiedades.

Cada uno de dichos trámites y servicios, da origen a un expediente, en el que se concentra toda la documentación requerida, así como la generada por el ICEO durante el proceso del mismo, y que, cuando llega a su conclusión, es almacenado en el Archivo Central del Instituto.

Cabe mencionar que las instalaciones del Archivo se encuentran fuera del edificio del ICEO.

Estos expedientes, forman los antecedentes históricos de los bienes inmuebles a través del tiempo y se consideran oficiales. La datos contenidos en ellos son de carácter confidencial, y no se pueden otorgar más que a los titulares registrados y/o acreditados para ello.

Debido a que los procesos de registro y actualización catastral, han sido cambiantes a lo largo de la historia, existen ciertas inconsistencias e irregularidades en la información del padrón.

Esto implica que cuando se recibe una solicitud de trámite o servicio de una propiedad que se encuentra en esas condiciones, es necesario recurrir al Archivo, para buscar los antecedentes respectivos, analizarlos e intentar resolver su situación.

El acceso a los expedientes almacenados, se hacía a través de una solicitud por escrito dirigida al encargado del Archivo. En la solicitud se indicaban algunos datos

clave que permitieran la localización del expediente correspondiente. Una vez, que se recibía esta solicitud en el Archivo, se registraba en una hoja de Excel, para fines de bitácora y se procedía a su localización.

Si el expediente era encontrado, se preparaba y se enviaba al personal que lo había solicitado. Si no era encontrado, se respondía por escrito de dicha situación. Este proceso tardaba al menos una semana, dependiendo de la cantidad de solicitudes de expedientes enviadas al Archivo, afectando el tiempo de atención a las solicitudes de trámites ingresadas al ICEO.

Con fines de hacer un poco más ágil la consulta de los expedientes, las autoridades del ICEO, permitieron realizar solicitudes de manera económica, vía telefónica y por correo electrónico, pero sin dejar de manejarlo en papel. Esto permitió iniciar la búsqueda de los expedientes solicitados casi inmediatamente pero no redujo significativamente el tiempo.

Además de la lentitud del acceso a los antecedentes, el hecho de que el expediente físico saliera de su resguardo, se corría el riesgo de extravío, daño y alteración intencionada o accidental.

Otro inconveniente, era la dificultad de proporcionar información oportuna sobre un expediente prestado, como saber quién lo solicitó y desde cuándo lo tenía en posesión.

Pregunta de Investigación

¿La implementación de un sistema informático para consultar expedientes de forma electrónica ayudará a reducir el tiempo de acceso al expediente y evitará los riesgos de daño, extravío así como la alteración del mismo?

1.2. Justificación

Como se ha mencionado, el proceso de consulta de expedientes resulta ser un punto clave para el ICEO, tanto en eficiencia en la atención de los trámites como en la seguridad de los expedientes físicos y de los datos contenidos en ellos.

Debido a esto, las autoridades del ICEO se encargaron de buscar una solución que ayudara a resolver estos problemas, determinando que era necesaria una herramienta informática para realizar las consultas de forma electrónica y que implicaba realizar la digitalización de los expedientes resguardados en el archivo del instituto, proponiendo el proyecto de creación del SIGED (Sistema de Gestión de Expedientes Digitalizados).

Con la implementación del SIGED, se espera que el tiempo del proceso de consulta de expedientes se reducirá a 3 días o menos, y por consiguiente la atención de las solicitudes de trámites se hará más rápidamente, beneficiando a los usuarios a solventar dichos trámites en un tiempo más corto, logrando con ello el cumplimiento de la reglamentación del ICEO de atenderlos en un plazo no mayor a 8 días hábiles (Ley de Catastro para el Estado de Oaxaca, 2014).

Además, al permitir a los usuarios poder consultar electrónicamente los expedientes se evitará la manipulación directa de los legajos físicos, de esta forma se eliminarán totalmente los riesgos de daño, alteración o extravío, y se podrá dar mayor confianza a los propietarios de que la información se encuentra bien resguardada.

Otro aspecto importante es que al utilizar cuentas de usuario para el personal que requiera consultar los expedientes, el SIGED mejorará el control sobre quiénes y cuándo los consultan, e implementará de esta forma, una medida para evitar que personas no autorizadas accedan a los datos contenidos en ellos.

Adicionalmente, el SIGED, tomará en cuenta los lineamientos del Modelo Óptimo de Catastro, el cual se generó como parte del plan de homologación y modernización de los registros públicos y catastros del país, contemplado en el Plan Nacional de Desarrollo 2007-2012, y esto coadyuvará a que el ICEO y el

Gobierno del Estado cumplan con los puntos de dicho plan de homologación y modernización.

1.3. Objetivos

Objetivo General

Desarrollar un sistema informático que permita la consulta de expedientes del ICEO de manera electrónica logrando con ello disminuir el tiempo actual del proceso de consulta, aumentando el control sobre los accesos y evitando los daños, alteraciones o pérdidas de los expedientes.

Objetivos específicos

Disminuir el tiempo del proceso actual de consulta de expedientes de una semana o más a 3 días o menos.

Mejorar el control del acceso a los expedientes usando autentificación y un syslog.
Eliminar los riesgos de daños, alteraciones o pérdidas de expedientes usando una versión electrónica de ellos.

1.4. Metodología

Para el desarrollo e implementación del SIGED fue necesario utilizar los siguientes recursos:

Financieros: para la adquisición de los escáneres con los que se digitalizan los expedientes, para las licencias de las librerías utilizadas y del framework de desarrollo, para los servidores de almacenamiento que contienen las digitalizaciones, para el sueldo del equipo de desarrollo.

Humanos: se necesitó apoyo del personal encargado del archivo para conocer la estructura de los expedientes y el sistema de archivo que utilizan para la administración de los mismos; también se recurrió al personal del área jurídica del ICEO quienes son los usuarios que más requieren realizar consultas de los expedientes, para conocer los datos clave que permitan ubicar un expediente; para el desarrollo del sistema se formó un equipo de 4 programadores; para la implementación se requirió de 5 digitalizadores y 2 capturistas.

Tecnológicos: fue necesario aumentar la capacidad de almacenamiento de los servidores storage actuales para poder soportar las digitalizaciones de los expedientes a largo plazo; además se necesitó la licencia de la librería del formato DJVU que se utiliza en la digitalización.

Para llevar a cabo el desarrollo del SIGED, se guío con la siguiente planeación:

ACTIVIDAD	NOVIEMBRE				DICIEMBRE				ENERO				FEBRERO				MARZO				ABRIL				MAYO					
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4		
ANÁLISIS																														
DISEÑO																														
PROGRAMACIÓN																														
PRUEBAS																														
IMPLEMENTACIÓN																														

Análisis: proceso actual de consulta, gestión de los expedientes en el archivo, identificación de datos clave para localizar un expediente, identificación de estándares o normas sobre digitalización, conocimiento de los lineamientos técnicos de los datos del Modelo Óptimo de Catastro, revisión de la metodología de desarrollo de software utilizada.

Diseño: documento de visión, especificación de requerimientos, diagrama de casos de uso, especificación de casos de uso, diagramas de actividades, diagramas de clases, diagrama de secuencias, diagrama de interfaces, arquitectura del sistema.

Programación: set-up del entorno de desarrollo, creación de la estructura del proyecto, creación de la base de datos, distribución de las funcionalidades entre el equipo, programación de las funcionalidades.

Pruebas: plan de pruebas, ejecución de las pruebas, ajustes del sistema.

Implementación: set-up de los servidores, instalación de la aplicación en los equipos clientes.

Después de analizar las ventajas y desventajas de los formatos compactos para la digitalización de documentos PDF y DJVU, se determinó usar éste último, lo que conllevó a definir el lenguaje a utilizar para el desarrollo de la aplicación, C Sharp (C#), por el soporte que la librería ofrece para dicho lenguaje.

Para el caso de la base de datos, se decidió utilizar PostgreSQL, por ser una herramienta libre, robusta y de fácil manejo.

El desarrollo de la aplicación se realizó basándose en la metodología SunTone de SUN, la cual da una estructura de desarrollo que era familiar para los programadores por lo que su utilización fue fácil.

CAPÍTULO 2. MARCO TEÓRICO

2.1. Normas nacionales y estatales de catastros

La Constitución Política de los Estados Unidos Mexicanos, establece que los municipios serán los encargados de administrar los bienes catastrales que se encuentren dentro de su territorio.

Sin embargo, en el Estado de Oaxaca, dada la situación económica prevaleciente en la mayor parte del territorio, existe el ICEO, el cual es una entidad de gobierno cuya finalidad es dar apoyo a los municipios que así lo deseen a gestionar sus catastros, brindándoles toda la infraestructura administrativa, jurídica, técnica y tecnológica necesaria, y cuya normatividad se encuentra en la Ley de Catastro para el Estado de Oaxaca.

2.1.1. Ley de Catastro para el Estado de Oaxaca

Esta ley, Cámara de Diputados (2014), es la que define las obligaciones y atribuciones del ICEO y de los propietarios de los bienes inmuebles comprendidos dentro del territorio del Estado, en materia catastral.

En la sección de Anexos, se enlistan los artículos y fracciones que se deben considerar como importantes en el desarrollo del SIGED.

2.1.2. Ley de Protección de Datos Personales del Estado de Oaxaca

Dada la naturaleza de los datos que se registran en materia catastral, la ley de protección de datos personales Cámara de Diputados (2008), obliga al instituto a considerar con carácter de confidencial la información contenida en los expedientes de los trámites que se realizan y archivan en el ICEO.

En la sección de anexos se listan los artículos considerados para el desarrollo del SIGED.

2.1.3. Ley de Transparencia y Acceso a la Información Pública para el Estado de Oaxaca

La información de los expedientes catastrales, también se encuentra protegida por la Ley de Transparencia del estado Cámara de Diputados (2013), siendo los artículos listados en los anexos, los que se tomaron en cuenta.

2.1.4. Modelo Óptimo de Catastro INEGI, SEDESOL (2011)

Este modelo creado por diversas entidades de gobierno a nivel federal, principalmente el Instituto Nacional de Estadística y Geografía (INEGI) y la Secretaría de Desarrollo Social (SEDESOL), se considera prácticamente el estándar a seguir para el registro de datos catastrales.

En este modelo, se definen los datos mínimos necesarios que se deben registrar en materia catastral, así como el tipo y la estructura que cada uno de ellos debe tener. También sugiere la forma de organizar y relacionar dicha información. (Capítulo 3, Apartado 3.3, página 144).

2.2. Digitalización

De acuerdo a la definición de la Real Academia Española (2016), la digitalización es la “acción y efecto de digitalizar”, y digitalizar es “registrar datos en forma digital. Convertir o codificar en números dígitos datos o informaciones de carácter continuo, como una imagen fotográfica, un documento o un libro”.

Considerando estas definiciones y el contexto del presente trabajo, se entenderá como digitalización al proceso mediante el cual un objeto material se transforma hacia un objeto inmaterial construido con algún dispositivo electrónico y que sólo puede ser manipulado por dispositivos de la misma naturaleza.

Dada esta definición, podemos decir que el proceso de convertir un documento en

papel a una imagen que pueda ser visualizada en algún dispositivo electrónico, es una digitalización del documento. A este proceso de digitalización, también se le llama escaneo del documento, debido al nombre de los dispositivos utilizados para obtener la imagen del mismo (Scanner, o Escáner). La imagen obtenida de la digitalización, se considera un documento digitalizado.

2.2.1. Atributos de una imagen digital

Cuando se realiza un proyecto de digitalización de documentos, es importante conocer los aspectos técnicos de las imágenes digitales, ya que con ellas se conformarán los documentos digitalizados.

Una imagen digital se forma con una matriz de pequeños puntos llamados píxeles, cada uno de estos puntos guarda datos de esa pequeña parte de la imagen como su tonalidad o brillo. Así lo menciona Rodríguez, Hugo (2003), en el Curso de Imagen Digital, publicado en internet por la Universidad Nacional de San Luis, de Argentina:

“La imagen digital está formada por un conjunto definido de puntos llamados píxeles.

Cada píxel de una imagen almacena la información de su tono o luminosidad, donde el tono blanco es el valor 0 y el negro el valor más alto (normalmente 255 en escala de grises), pero en formato binario.”

Otro aspecto de una imagen digital es la profundidad de color, el cual se refiere a la cantidad de tonalidades soportadas por cada pixel de la imagen. Esta cantidad se calcula en base al número de bits que se defina utilizar para cada pixel, así una profundidad de color de 8 bits permitirá que un pixel pueda tener hasta 256 tonos distintos.

En el Curso de Imagen Digital de Rodríguez, Hugo (2003), se describe este aspecto como “el número de bits que definen cada píxel, que determinan el máximo número de colores que puede tener.

Si cada píxel viene determinado por 2 Bytes (=16 bits) en vez de por un Byte,

existirán 65,536 tonos de gris, ya que el número binario 1111111111111111 corresponde a 65,536. Es lo que se denomina una profundidad de color de 16 bits.”

Otra característica a considerar es el tamaño de la imagen. Anteriormente se mencionó que una imagen se define por una matriz de puntos, dicha matriz no es más que una cuadrícula que contiene filas y columnas. El número de filas y columnas de la matriz es lo que determina el tamaño de la imagen, de esta forma, cuando se dice que una image es de 1200 x 800 pixeles de tamaño, se refiere a que la matriz que la conforma tiene 1200 columnas y 800 filas.

Rodríguez, Hugo (2003), en su Curso de Imagen Digital, dice que el tamaño de la imagen “Se define con las dimensiones en píxeles de la matriz o cuadrícula.

Si una imagen está formada por una matriz de 800 columnas por 500 filas, tiene entonces un tamaño de 800 x 500 píxeles.”

La resolución es otro elemento que es importante tomar en cuenta con una imagen digital. Este aspecto se refiere al número de píxeles que hay en la imagen en relación a una unidad de longitud, la cual generalmente es la pulgada, lo que nos indica qué tan nítida es la imagen al ser impresa o visualizada. Si una imagen tiene 150 píxeles por cada pulgada de longitud, entonces se dice que su resolución es de 150 puntos por pulgada, o en forma abreviada 150 ppp, o en inglés 150 dpi (dots per inch).

Esto lo podemos confirmar en el Curso de Imagen Digital de Rodríguez, Hugo (2003): “Es la medida de cantidad de píxeles por unidad de longitud, comúnmente píxeles por pulgada (una pulgada equivale a 2,54 cm de longitud). Se suele abreviar como “ppp” o “dpi” (dot per inch). Como la resolución mide el número de píxeles por longitud, se deduce que a mayor resolución, mayor número de puntos de imagen en el mismo espacio y, por tanto, mayor definición. Es decir: resolución es definición. Este es, posiblemente, uno de los conceptos que más se prestan a confusiones entre los aficionados, principalmente por creer que resolución es lo mismo que calidad.

Un ejemplo: si una imagen tiene unas dimensiones en píxeles de 548x366 y se

imprime o está mostrando en pantalla con unas dimensiones de 2x3 pulgadas, entonces tiene una resolución de 72 dpi. Debe quedar claro que la resolución es la relación entre las dimensiones digitales (las medidas en píxeles) y las físicas (las que tiene una vez está impresa).

La resolución no es una medida de la calidad de una imagen digital, aunque muy a menudo se utilice para ello; es una medida de nitidez o definición, de forma que cuanto más alta sea, mayor definición y viceversa. La calidad es la conjunción de dos factores: la resolución y el tamaño, y si ambas son elevadas, la calidad también lo será.”

Para la generación de la imagen digital se han desarrollado distintos algoritmos con los que se procesa electrónicamente un documento físico, cada uno de los cuales define la resolución y calidad de la imagen digital.

Cada algoritmo se asocia con un tipo de imagen digital, y este tipo de imagen es otra propiedad a considerar.

Existen dos principales tipos de imágenes: las imágenes de mapa de bits y las imágenes vectoriales.

Una imagen de mapa de bits, es la que se genera usando la matriz de pixeles, como se ha mencionado en líneas anteriores.

En cambio, una imagen vectorial, utiliza un modelo matemático para generar la información de la imagen la cual se guarda como un objeto de programación, con atributos que definen las características de cada parte de la imagen. Debido a esto, el tamaño en bytes de una imagen vectorial resulta ser mucho más reducido que el de una image de mapa de bits.

Es lo que Ordoñez Santiago, Cristian Andrés (2005), nos confirma con las siguientes definiciones:

“TIPOS DE IMÁGENES

Por la forma de manejar los datos en un archivo de imagen, se puede hablar de dos modos principales para manipular la información que integra una imagen digital. Estos modos son las imágenes de mapa de bits y las imágenes vectoriales.

IMÁGENES DE MAPA DE BITS

Las imágenes de mapa de bits (bitmaps o imágenes raster) están formadas por una rejilla de celdas. A cada una de estas celdas, que se denominan píxeles, se le asigna un valor de color y luminancia propios.

IMÁGENES VECTORIALES

Los llamados gráficos orientados a objetos son las imágenes vectoriales. Su tamaño es mucho más reducido, en comparación con los mapas de bits, porque el modo como organizan la información de una imagen es más simple que en aquellos. Dicha simplicidad radica en generar los objetos que conforman una imagen a través de trazos geométricos determinados por cálculos y fórmulas matemáticas.”

TIPOS DE COMPRESIÓN

El archivo de una imagen digital puede llegar a ser muy grande dependiendo de los valores de los atributos mencionados, lo cual implica que esto se convierta en un problema de almacenamiento e incluso de manipulación por algún software de tratamiento de imágenes. Debido a esto, se han creado técnicas que permiten reducir el tamaño de los archivos de imágenes, es decir, las comprimen.

Básicamente estas técnicas se clasifican en dos categorías: con pérdida y sin pérdida, refiriéndose a si en la compresión se pierde o no información de la imagen.

La compresión sin pérdida, la reducción del tamaño del archivo de la imagen resulta muy poca, ya que la información de la imagen es compactada sin eliminar nada de ella, por lo que al descomprimir la imagen queda como originalmente estaba.

En cambio en la compresión con pérdida, los métodos usados desechan una buena cantidad de información de la imagen implementando procesos que compensan esta pérdida, para que a simple vista no sea perceptible

Sobre este tema, Ordoñez Santiago, Cristian Andrés (2005), nos menciona:

“Estas técnicas tratan de reducir, mediante algoritmos matemáticos, el volumen del archivo para así disminuir los recursos que consuma y abreviar el tiempo de

transferencia. Estos complejos algoritmos matemáticos reducen de diversos modos los 0 y 1 que conforman una imagen digital. Asimismo, como con los formatos de imagen, las técnicas de compresión son de dominio público o pertenecen a la empresa que las desarrolló.

Su división más común es la compresión sin pérdida y la compresión con pérdida; lo cual radica en qué tanta información de la imagen se pierde al ser comprimida. Por último, hay que agregar que algunos de los formatos pueden utilizar varias de las diferentes técnicas para comprimir.”

2.2.2. Formatos para digitalización de documentos

PDF

El formato PDF es uno de los más conocidos y usados en la digitalización, debido a su alta portabilidad, esto es, que puede ser manejado en distintas plataformas de sistemas operativos.

En el portal de internet de Adobe (2016), empresa que creó este formato, nos define PDF como “un formato de archivo utilizado para presentar e intercambiar documentos de forma fiable, independiente del software, el hardware o el sistema operativo”.

La idea del Dr. John Warnock, cofundador de Adobe, “consistía en que todo el mundo pudiera capturar documentos de cualquier aplicación y enviar la versión electrónica de dichos documentos a donde quisiera, así como verlos e imprimirlas desde cualquier máquina”

Algunas de las ventajas de este formato son la posibilidad de incluir vínculos y botones, incluso formularios. Además se pueden proteger con contraseña para restringir su visualización. Y para visualizarlos se utiliza un programa gratuito disponible para todos los sistemas operativos llamado “Acrobat Reader”.

DJVU

La página oficial DJVU Org (2016) de este formato lo define como “un formato de documento digital con tecnología de compresión avanzada y alto valor de

rendimiento. DjVu permite la distribución en Internet y en DVD de imágenes de alta resolución de documentos escaneados, documentos digitales y fotografías. Los visores de DjVu están disponibles para el navegador web, el escritorio y los dispositivos PDA.”

La compresión de este formato es su principal característica, en el sitio web se menciona: “alcanza proporciones de compresión de 5 a 10 veces mejores que los métodos existentes, como JPEG y GIF para documentos en color, y 3 a 8 veces en comparación con TIFF para documentos en blanco y negro”.

En cuanto al tamaño de los archivos, se comenta: “Las páginas escaneadas a 300 DPI a todo color se pueden comprimir de 25 MB hasta 30 a 100 KB. Las páginas en blanco y negro a 300 DPI suelen ocupar de 5 a 30 KB cuando se comprimen”

2.2.3. Escáner (Scanner)

Dentro de los dispositivos para obtener documentos digitales, se encuentra el escáner.

Este dispositivo convierte un documento físico a un documento digital o imagen usando una técnica basada en sensores de luz, esto es, algo similar a una fotocopia pero en lugar de arrojar la copia en papel, la guarda en un archivo de imagen.

En la publicación de Giménez, Joaquín (2005), se menciona lo siguiente sobre el escáner: “Un escáner transforma imágenes reales, físicas y con precisión infinita en imágenes finitas y con una precisión determinada que pueda ser procesada por una computadora”.

Existen varios tipos de escáneres, entre ellos están los escáneres llamados de alta velocidad o de alta producción, los cuales son dispositivos que tienen la capacidad de “leer” los documentos físicos con una rapidez bastante alta (hasta 210 páginas por minuto) y que además son capaces de procesar documentos ilimitadamente.

2.2.3.1. Estándares de adquisición de imágenes

La necesidad de los programas de edición de imágenes de poder obtener las imágenes a manipular, directamente del escáner sin necesidad de cerrar la aplicación y obtenerla desde otra, llevó a la creación de distintos estándares que permitieran esto.

Estos estándares son TWAIN, WIA e ISIS. Todos ellos funcionan como una interfaz entre las aplicaciones que requieren obtener una imagen del escáner y el hardware propio del escáner.

Los fabricantes de escáneres deciden el o los estándares de adquisición de imágenes a los que darán soporte.

Gracias a estos estándares, es posible implementar aplicaciones particulares de digitalización masiva de documentos.

2.3. Marco legal estatal existente sobre documentos digitalizados

Actualmente, las leyes estatales no contemplan ninguna reglamentación sobre los documentos digitalizados. A nivel nacional, tampoco existen normas que regulen o establezcan los lineamientos que se deben utilizar en la digitalización de documentos oficiales.

Debido a esto, se recurrió a recomendaciones de otros organismos nacionales e internacionales, para definir las propiedades de los documentos digitales, los cuales se refieren a continuación: BARRERA RIVERA, MÓNICA MARÍA DEL ROSARIO (2009), Bermúdez Muñoz, María Teresa (2011), Archivo General de Castilla y León (2010), IFLA E ICA (2002).

2.4. Software

Escuetamente se puede decir que software es un programa de computadora. Esto lleva a decir qué es un programa de computadora, y diremos que es un conjunto

de instrucciones definidas para realizar ciertas tareas en una computadora. Los términos sistema y aplicación, se pueden considerar sinónimos de software.

De acuerdo a Sommerville, Iam (2005), “un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes”

2.4.1. Clasificación del Software

Básicamente la clasificación del software se hace considerando el fin general para el que se fabricaron, y dado que este fin puede ser muy diverso, lo cual llevaría a tener un gran número de clases de software, se ha tomado como punto de separación el hardware, de esta forma se pueden englobar las variedades de software dependiendo de su posición respecto al hardware.

Un grupo se enfoca a interactuar de manera directa con el hardware, facilitando este trabajo a los programas que se engloban en el grupo contrario. A este grupo se le conoce como Software de Sistemas.

Algunos ejemplos de programas de este tipo son los sistemas operativos, los compiladores.

Pressman, Roger S. (2010), lo escribe así: “Software de sistemas: conjunto de programas escritos para dar servicio a otros programas. Se caracterizan por una gran interacción con el hardware de la computadora, uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación, recursos compartidos y administración de un proceso sofisticado, estructuras complejas de datos e interfaces externas múltiples.”

El otro grupo que no es software de sistemas, se refiere a los programas que se enfocan a dar solución a tareas “más humanas” y usan como plataforma al software de sistemas para poder realizar su función. Este grupo es conocido como

Software de Aplicación.

Sobre este grupo, Pressman, Roger S. (2010) dice: “Software de aplicación: programas aislados que resuelven una necesidad específica de negocios. Las aplicaciones en esta área procesan datos comerciales o técnicos en una forma que facilita las operaciones de negocios o la toma de decisiones administrativas o técnicas. Además de las aplicaciones convencionales de procesamiento de datos, el software de aplicación se usa para controlar funciones de negocios en tiempo real (por ejemplo, procesamiento de transacciones en punto de venta, control de procesos de manufactura en tiempo real).”

Ejemplos de esta clase de software son: los distintos programas de oficina, comerciales o libres, programas de diseño gráfico, y programas hechos a la medida para un particular.

2.4.2. Arquitecturas de Software

Tratando de hacer una analogía con la arquitectura de una casa, la arquitectura de software, se refiere a las distintas formas de “acomodar” los elementos que conforman un programa, de manera que esa distribución sea la más adecuada dependiendo del tipo de programa a construir.

Tal y como se hace en la arquitectura de construcciones, una construcción destinada a oficina requiere cubrir distintas necesidades que una destinada a hogar, o a una destinada a bodega.

Así, un programa administrativo no necesita cubrir las mismas necesidades de un juego.

Sobre este punto, Hanmer, Robert (2013), define Arquitectura de Software dependiendo del punto de vista de quién la esté tratando: “para el desarrollador, significa la estructura del sistema que se está construyendo. Para un desarrollador de framework, es la forma del sistema que se crea con el framework. Para un tester, es la forma de lo que necesita ser probado”.

Y en términos más generales, dice: “es la estructura de alto nivel de la solución a

un problema que el cliente quiere resuelto”.

“Es la forma en que las piezas encajan para construir una solución de algún negocio o necesidad técnica que el cliente quiere resuelta”

Existen diferentes modelos de arquitecturas de software, en los que las aplicaciones a desarrollar pueden basarse para su construcción. Algunas de las más usadas son:

Cliente – Servidor: Es una arquitectura en dónde las funciones de la aplicación se dividen en un Cliente y un Servidor.

Arquitectura de tres niveles: Es una especialización de la anterior, pero en ésta, las funcionalidades se dividen en 3 capas: la interfaz de usuario, la lógica de negocio y la persistencia, y cuya relación entre ellas es en el orden en que se mencionaron.

2.4.3. Sistema de digitalización

Basándose en los conceptos mencionados en los apartados anteriores, se puede definir un sistema de digitalización, como un software de aplicación cuyo objetivo es crear documentos digitalizados y con el cual se puedan gestionar, entendiendo por gestionar la posibilidad de crear, actualizar, consultar y eliminar dichos documentos.

2.4.4. Base de Datos y Sistemas Gestores de Bases de Datos

Prácticamente todos los programas requieren almacenar datos, ya sea datos utilizados por el propio programa para ejecutar ciertas tareas o bien datos que los usuarios registran.

Ese conjunto de datos, organizados y relacionados, es conocido como Base de Datos.

El autor Silberschatz, Abraham (2002), dice que una Base de Datos es “una colección de datos interrelacionados”.

Los programas que se encargan de manipular y gestionar el almacenamiento de los datos utilizados por un programa o sistema, son llamados Sistemas Gestores de Bases de Datos.

También Silberschatz, Abraham (2002), define un sistema gestor de bases de datos (SGBD) como “un conjunto de programas para acceder a los datos” guardados en bases de datos.

Además, menciona “los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información.”

2.4.5. Lenguaje de Programación

Para construir un software (programa), se utilizan los lenguajes de programación. Un lenguaje de programación es un conjunto de reglas, instrucciones y estructuras que ayudan a escribir la lógica de un programa.

2.4.6. Ambiente de desarrollo (plataforma)

Para el caso del SIGED, el equipo de desarrollo consideró utilizar el siguiente ambiente de desarrollo, el cual fue determinado principalmente por el formato de digitalización seleccionado para la creación de los documentos digitales, el formato DJVU, y por el conocimiento técnico y experiencia de los integrantes.

Metodología de desarrollo: Basada en el modelo de Suntone

Lenguaje de programación: .Net C#, usando el entorno de desarrollo Visual Studio .Net

Sistema Gestor de Base de datos: PostgreSQL, usando el administrador pgAdmin

Formato de digitalización: DJVU

Arquitectura de software: Cliente-Servidor

CAPÍTULO 3. METODOLOGÍA DE DESARROLLO DEL SISTEMA

En este capítulo se incluyen los artefactos (documentos y diagramas) utilizados durante la fase de análisis y diseño del SIGED.

Dado que este sistema fue desarrollado por un equipo de trabajo de 4 programadores, siendo yo uno de ellos, se limitará la inclusión de los documentos y diagramas relacionados a las funcionalidades de las cuales estuve a cargo: Escanear Expediente y Definir Dispositivos de Escaneo, del módulo de Digitalización de Expedientes.

3.1. Descripción de la metodología de desarrollo Suntone

El “White Paper” de SUN MICROSYSTEMS (2001) lo define como “un modelo que organiza el análisis arquitectónico a través de 3 dimensiones: niveles, capas y cualidades sistémicas.” Las cuales describe como sigue:

“Niveles: Es una organización lógica o física de componentes ordenados en cadena como proveedores y consumidores de servicios. Los componentes dentro de un nivel típicamente consumen los servicios de los proveedores de un nivel adyacente y a su vez prestan servicios a los consumidores de un nivel adyacente.”

“Capas: Las capas, como los niveles, representan una bien ordenada relación entre los límites de cada interfaz. Mientras que los niveles representan cadenas de procesamiento entre componentes, las capas representan relaciones contenedor/componente en la implementación y despliegue de los servicios”

“Cualidades sistémicas: Son las estrategias, herramientas y prácticas que entregarán el requisito de calidad de servicio entre niveles y capas (por ejemplo disponibilidad, escalabilidad, seguridad y capacidad de gestión)”

3.2. Artefactos utilizados

En el análisis y diseño del SIGED se utilizaron los siguientes artefactos que la metodología Suntone contiene:

Documento de Especificación de Requerimientos: En este documento se detallan los requerimientos funcionales y no funcionales del sistema, se identifican los riesgos y los actores, y se enuncia una primera aproximación de los casos de uso, así como un mapeo de cada uno de ellos contra los requerimientos identificados.

Diagrama de Casos de Uso: Este diagrama permite relacionar los casos de uso con los actores, además de que se empiezan a organizar los casos de uso por módulos.

Documento de Especificación de Casos de Uso: En este documento se describen cada uno de los casos de uso en forma de “receta”. Este documento permite refinar aún más los casos de uso, identificando nuevos casos de uso y eliminando casos de uso no necesarios.

Diagramas de Actividad: Con estos diagramas se realiza una validación de la especificación de casos de uso, ayudando a identificar posibles omisiones en la descripción de los mismos, o bien, posibles duplicidades de procesos.

Diagramas de Clases: Representan las entidades abstractas que intervendrán en los procesos del sistema, identificando las relaciones entre ellas. Con este diagrama se logra derivar el esquema de la base de datos que se utilizará en el sistema.

Diagramas de Secuencia: Con estos diagramas se representan los procesos indicando las clases que intervienen en cada paso de los mismos, e identificando las interfaces necesarias para el sistema.

3.3. Especificación de requerimientos

Requerimientos Funcionales

Características Principales

Características Esenciales

- Crear expedientes digitalizados de los expedientes físicos (documentos y planos).
- Consultar los expedientes digitalizados por campos clave (número de registro, cuenta predial, clave catastral, nombre del contribuyente, curp, año) necesarios por cada área del ICEO.
- Operar escáner que soporte los estándares TWAIN.
- Gestionar los usuarios del sistema

Características de Alto Valor

- Crear bitácora de movimientos.
- Permitir a los usuarios modificar su contraseña.

Características de Seguimiento

- Relacionar la base de datos del Padrón de Catastro con los expedientes digitalizados.
- Generar estadísticas de productividad de los escaneos por capturista en el proceso de digitalización masiva.
- Gestionar los catálogos del sistema.
- Permitir a los usuarios recordar sus datos de inicio de sesión.
- Realizar la actualización del sistema automáticamente.

Actores

Estos son los roles de personas y sistemas que interactúan con el sistema.

Nombre Actor	Descripción
Operador de escáner.	Personal encargado de obtener las imágenes de los expedientes físicos y asignar los expedientes digitalizados al personal de captura. Este personal realizará una digitalización masiva de los expedientes que se encuentran en el Archivo central del ICEO.
Capturista.	Personal encargado de identificar los metadatos de los expedientes digitalizados y capturarlos para su integración con dicho expediente. Este personal recibirá en una cola de captura los expedientes asignados por el digitalizador.
Digitalizador Unitario.	Usuario que realiza el proceso de digitalización y captura de metadatos de los expedientes que se integran a diario, tiene la facultad de eliminar y modificar expedientes.
Consultor.	El consultor buscará los expedientes digitalizados en la interfaz con los criterios de búsqueda disponibles. Dicho rol tiene limitaciones de solo lectura a dichos expedientes.
SysAdmin.	El administrador del sistema analizará las bitácoras del sistema (accesos al

	sistema de consulta, eliminación de expedientes, creación de expedientes, modificación de expedientes), así como llevara a cabo la administración de los usuarios.
Sistema Integral de Catastro.	El SIC operara las funciones de consulta de expedientes digitalizados.
Supervisor de digitalización.	Revisa productividad en el proceso de Digitalización masiva, además de realizar estadísticas.

Actor: *Operador de escáner*

- Descripción del Empleado – El ICEO contratará a este personal de manera temporal para llevar a cabo la tarea de la digitalización masiva.
- Descripción del Trabajo – Digitalizador. Este personal manipulará los expedientes físicos para su digitalización y verificará que se lleve a cabo de manera correcta el proceso de digitalización.
- Objetivo Aplicación – Generar imágenes de los expedientes físicos.
- Promedio de tiempo de uso – 8 horas.
- Frecuencia de Uso – 5 días por semana.
- Educación – Licenciatura.
- Conocimientos – Computación y Uso de Escáner.
- Experiencia comprobable Computadoras – PC Escritorio.
- Experiencia Sistema Operativo – Windows xp, Windows 7.
- Experiencia con Aplicaciones Similares – Escaneo de Documentos e Imágenes.
- Entrenamiento adicional – Curso del SIGED.
- Rotación – Nula.
- Cumplimiento esperado – Deberá desempeñar de manera analítica y responsable la digitalización de los expedientes físicos.

Actor: Capturista

- Descripción del Empleado – El capturista está coordinado con el Digitalizador Masivo. Su labor empieza cuando en su cola de espera figuren expedientes digitalizados.
- Descripción del Trabajo – Visualizar en pantalla los documentos digitalizados que se encuentren en la cola, seleccionar un expediente y buscar dentro del expediente los metadatos requeridos por la interfaz.
- Objetivo Aplicación – Integrar los metadatos de los expedientes digitalizados.
- Promedio de tiempo de uso – 8 horas.
- Frecuencia de Uso – 5 días por semana.

- Educación –Licenciatura.
- Conocimientos – Básicos de computación y captura de datos.
- Experiencia Computadoras – PC Escritorio.
- Experiencia Sistema Operativo – Windows xp, Windows 7.
- Experiencia con Aplicaciones Similares – Captura de Datos.
- Entrenamiento adicional – Curso del SIGED y Curso para ubicar los metadatos dentro del expediente digitalizado.
- Rotación – Nula.
- Cumplimiento esperado – Cumplir con las capturas diarias asignadas de los expedientes digitalizados y realizar una captura eficiente.

Actor: Digitalizador Unitario

- Descripción del Empleado – El digitalizador unitario es personal de confianza que se encuentra dentro de la estructura del ICEO y que pertenece al Departamento de Jurídico.
- Descripción del Trabajo – Digitalizar los expedientes físicos que se completan en el Departamento de Jurídico, capturar los metadatos para integrar al expediente digitalizado , modificar expedientes y eliminar expedientes.
- Objetivo Aplicación –Digitalizar, Capturar y administrar los expedientes que son integrados en el Departamento de Jurídico.
- Promedio de tiempo de uso – 2 horas/día.
- Frecuencia de Uso – Diario.
- Educación –Licenciatura.
- Conocimientos – Básicos de computación, escaneo de documentos y captura de datos.
- Experiencia Computadoras – PC Escritorio.
- Experiencia Sistema Operativo – Windows Xp, Windows 7.
- Experiencia con Aplicaciones Similares – Captura de Datos.
- Entrenamiento adicional – Curso del SIGED y Curso para ubicar los metadatos dentro del expediente digitalizado.
- Rotación –Anual
- Cumplimiento esperado – Realizar los procedimientos de manera eficiente y oportuna para contar con una base de expedientes digitalizados actualizada.

Actor: Consultor

- Descripción del Empleado – Cualquier empleado que se encuentre dentro de la estructura organizacional del ICEO y que sea aprobado por las autoridades correspondientes podrá obtener el rol de consultor.

- Descripción del Trabajo – El consultor tiene diferentes actividades que desarrolla dentro de sus áreas y para las cuales necesita consultar los expedientes digitalizados.
- Objetivo Aplicación – Consultar de manera rápida y segura los Expedientes Digitalizados. Uso primario de la aplicación –Seleccionar un expediente del conjunto de expedientes digitalizados a través de un criterio de búsqueda para posteriormente visualizarlo en pantalla para su lectura. Sin opción a descargar.
- Promedio de tiempo de uso – 2 horas
- Frecuencia de Uso –1 vez por día ,5 veces por semana.
- Educación – Preparación universitaria con carrera a fin de cualquier índole con alto grado de ética.
- Conocimientos –Conocimientos básicos de computación.
- Experiencia en Sistemas Operativo –Windows xp, Windows 7.
- Experiencia con Aplicaciones Similares –Ninguna.
- Entrenamiento adicional – El personal recibirá capacitación para el uso del sistema SIGED
- Rotación –Nula.
- Cumplimiento esperado –Se espera que siga el procedimiento para el uso del usuario y contraseña, establecido para evitar mal uso de los expedientes.

Actor: SysAdmin

- Descripción del Empleado – En la estructura organizacional del ICEO es la persona que administra los sistemas de cómputo dentro de la institución.
- Descripción del Trabajo –Su labor es dar de alta usuarios de acuerdo a su perfil, generar bitácoras de acceso a los expedientes, dar mantenimiento al sistema.
- Uso primario de la aplicación –Dar de alta usuarios, generar bitácoras de acceso.
- Promedio de tiempo de uso – 1 horas
- Frecuencia de Uso – 1 veces por semana.
- Educación – Ingeniería en Sistemas
- Conocimientos –Conocimientos avanzados de computación.
- Experiencia Computadoras –Manejo de base de datos Oracle, manejo de programación Asp, Jsp, .NET, JAVA, PHP y administración de servidores apache tomcat.
- Experiencia en Sistemas Operativo –Windows xp, Windows 7, Windows server, Linux.
- Experiencia con Aplicaciones Similares –Ninguna.

- Entrenamiento adicional – El personal recibirá capacitación para el uso, operación y modificación del sistema SIGED.
- Rotación –Nula.
- Cumplimiento esperado –Se espera que siga el procedimiento para seleccionar los expedientes correctos a eliminar o modificar.

Actor: Sistema Integral de Catastro.

- Uso primario de la aplicación –Lanzar la aplicación de acuerdo a ciertos parámetros.
- Promedio de tiempo de uso – 1 horas
- Frecuencia de Uso – 1 veces por semana.

Actor: Supervisor de digitalización.

- Descripción del Empleado – En la estructura organizacional del ICEO es la persona que administra los escaneos en la digitalización masiva.
- Descripción del Trabajo –Su labor es eliminar y modificar expedientes creados con anterioridad mayor a 1 día, también tiene la facultad de generar estadísticas de la operación de las personas.
- Uso primario de la aplicación –Dar de baja expedientes, modificar expedientes, generar estadísticas.
- Promedio de tiempo de uso – 2 horas
- Frecuencia de Uso – 5 veces por semana.
- Educación – Carrera técnica en computación.
- Conocimientos –Conocimientos intermedios de computación.
- Experiencia Computadoras –Digitalización de expedientes.
- Experiencia en Sistemas Operativo –Windows xp, Windows 7..
- Experiencia con Aplicaciones Similares –Ninguna.
- Entrenamiento adicional – El personal recibirá capacitación para el uso, operación y modificación del sistema SIGED.
- Rotación –Nula.
- Cumplimiento esperado –Se espera que siga el procedimiento para seleccionar los expedientes correctos a eliminar o modificar.

Casos de Uso

Nombre de Caso de Uso	Prioridad	No.	Descripción
Escanear expedientes físicos.	E	1	Se muestran los procesos para obtener imágenes de los documentos que integran los expedientes, así como opciones adicionales para el escaneo.

Capturar metadatos del expediente	E	2	Se mostraran los procesos que realizara el capturista encargado de integrar los metadatos al expediente digitalizado.
Consulta de Expedientes	E	3	Muestra los procesos de los usuarios que requiera visualizar los expedientes digitalizados.
Gestión de Expedientes Digitalizados	E	4	Describe las acciones de modificar, eliminar expedientes y generar estadísticas.
Iniciar Sesión	E	5	Métodos necesarios para autenticarse en el sistema y adquirir los privilegios correspondientes.
Gestión de Usuarios	E	6	El administrador del sistema creara, eliminara y actualizara los usuarios que accederán al sistema.
Cambiar contraseña de usuario	A	7	El usuario podrá cambiar su contraseña mediante el sistema.
Consultar Bitácora.	A	8	Muestra las operaciones realizadas por los usuarios en el sistema.
Recordar datos de inicio de sesión.	S	9	El usuario recibirá los datos de inicio de sesión a través de correo electrónico con una contraseña temporal.
Actualizar sistema	S	10	Actualiza el sistema de manera automática.
Gestión de Catálogos.	S	11	Administración de los datos del Sistema.

E - Esencial, A – Alto Valor, S – Seguimiento.

Requerimientos Detallados por Caso de Uso

Identificador	Descripción del Requerimiento
E1-1	Proveer de una interfaz para obtener las imágenes de los expedientes físicos realizada por el Operador del escáner.
E1-2	Asignar elementos de la cola de expedientes a los capturistas para integración de metadatos (Digitalización masiva).
E2-1	Proveer de una interfaz para capturar los metadatos de las imágenes de los expedientes obtenidos por el operador del escáner.
E3-1	Crear interfaz en la cual se pondrán realizar consultas a los expedientes digitalizados.
E3-2	Filtrar búsquedas por cuenta catastral, cuenta predial, número de trámite y fecha.
E3-3	El sistema permitirá revisar el historial de expedientes pertenecientes a una cuenta catastral.(Esta es una consulta predefinida por clave catastral y fecha)
E4-1	Agregar, eliminar y actualizar expedientes.
E5-1	<p>El sistema permitirá el acceso solo al personal autorizado y con los privilegios correspondientes.</p> <p>PRIVILEGIOS CORRESPONDIENTES</p> <p>Operador de escáner</p> <ul style="list-style-type: none"> • Acceso a Escanear expediente físico. • Acceso a correcciones de expedientes escaneados del usuario. <p>Capturista</p>

	<ul style="list-style-type: none"> • Acceso a captura de metadatos de expedientes. • Acceso a correcciones de captura de metadatos. <p>Digitalizador unitario</p> <ul style="list-style-type: none"> • Acceso a Escanear expediente físico. • Acceso a correcciones de expedientes escaneados del usuario. • Acceso a captura de metadatos de expedientes. • Acceso a correcciones de captura de metadatos. <p>Consultor</p> <ul style="list-style-type: none"> • Acceso a consulta de expedientes. <p>Jefe de unidad jurídica</p> <ul style="list-style-type: none"> • Acceso a consulta de expedientes. • Acceso a Eliminar expedientes marcados. • Acceso a Aplicar cambios a expedientes marcados. <p>Notificador</p> <ul style="list-style-type: none"> • Acceso a consulta de expedientes • Acceso a Marcar expedientes para eliminar. • Acceso a Marcar expedientes para eliminar. <p>SysAdm.</p> <ul style="list-style-type: none"> • Acceso a bitácora del sistema. • Acceso a administración de usuarios. • Administrar catálogos. <p>SIC(Sistema Integral Catastral) y otros sistemas externos</p> <ul style="list-style-type: none"> • Consulta expedientes. <p>SEGURIDAD A IMPLEMENTAR</p> <ul style="list-style-type: none"> • Se utilizará la autenticación SHA-2 para el cifrado de contraseñas en la base de datos.
E6-1	Crear usuarios con su rol respectivo.
E6-2	Modificar usuarios.
E6-3	Eliminar usuarios.
E6-4	Crear roles de usuario.
E6-5	Modificar roles de usuario.
E6-6	Eliminar roles de usuario.
A7-1	El usuario tendrá la opción de cambiar su contraseña dentro del sistema las veces que sea requerida.
A8-1	Generar bitácora de operaciones de expedientes (acceso, eliminación y modificación).
A8-2	El sistema guardará en una bitácora los movimientos realizados en el sistema, tales como consultas de Expedientes Digitalizados y accesos al sistema.
S9-1	El sistema enviará datos de inicio de sesión al correo electrónico personal del usuario cuando requiera recordarlos.

S10-1	El sistema tendrá un módulo del lado del cliente en donde verificara que exista alguna actualización del mismo usuario el cual la descargara y la instalara.
S11-1	Agregar elemento a catálogo.
S11-2	Eliminar elemento de catálogo.
S11-3	Modificar elemento de catálogo.

Requerimientos No Funcionales

Rendimiento

Versión Actual

Identificador	Descripción del Requerimiento
E1-101	Para el escaneo de documentos el procedimiento no debe sobrepasar los 5 segundos por hoja tamaño oficio en caso de planos el tiempo no debe exceder 90 segundos.
E2-101	El tiempo de actualización de la interfaz con respecto a la cola de expedientes es de 10 ms.
E2-102	La interfaz gráfica para captura de metadatos debe tener un tiempo de respuesta de no más de 5s para el almacenamiento de los datos.
E3-101	El resultado de la consulta al expediente depende del tamaño del expediente y de la velocidad de transferencia de datos en la red, se espera que este tiempo no exceda los 30 segundos.
E3-102	El sistema debe soportar un máximo de 30 usuarios simultáneos, es decir, veinte conexiones a base de datos activas.
E4-101	Las modificaciones a expedientes por corrección eliminación o actualización deben verse reflejadas en la base de datos en un tiempo límite de 30 segundos.
E5-101	La carga del sistema con formulario de autenticación no debe sobrepasar los 20 segundos y debe dar respuesta de inicio de sesión al usuario en un tiempo de 5 segundos.
E6-101	La gestión de usuarios (crear, eliminar y modificar) debe tener un tiempo de respuesta máximo de 20 segundos, no habiendo restricción en cuanto el número de registros a almacenar.
A8-101	El resultado por consulta no debe exceder los 10 segundos.

Versión a Futuro

El número de usuarios que utilizan el sistema

Escalabilidad

Versión Actual

Identificador	Descripción del Requerimiento
E1-102	La aplicación de digitalización en su fase masiva soportará 2 equipos que trabajaran de manera simultánea y consistirán de 1 operador de escáner y 4 capturistas.
E1-103	El escáner para obtener las imágenes de los expedientes está limitado a 5000 hojas por día.

E3-103	La función de consulta soportara 30 usuarios simultáneos.
--------	-----------------------------------------------------------

Versión a Futuro

Identificador	Descripción del Requerimiento
E2-103	La función para la digitalización masiva soportara un número máximo de 5 equipos de digitalización con un máximo de 10 capturistas por equipo
E1-104	El escáner para obtener las imágenes de los expedientes podrá procesar 7,000 hojas por día.
E3-104	La función de consulta soportara 100 usuarios simultáneos.

Disponibilidad

Versión Actual

Identificador	Descripción del Requerimiento
E6-101	El SIGED será operable en cualquier momento para todos los usuarios.

Versión a Futuro

Identificador	Descripción del Requerimiento
E6-102	El SIGED estará disponible durante los períodos de trabajo definidos por el ICEO. En caso de intentar acceder fuera de ese período el sistema deberá informar sobre la negación del acceso. Sólo los usuarios con rol de Administrador de Sistema tendrán acceso 24 x 7 x 365.

Confiabilidad

Versión Actual

Identificador	Descripción del Requerimiento
E2-104	Un expediente digitalizado no debe existir sin sus metadatos registrados.
E6-103	Los usuarios deben tener asociados al menos un rol
E2-105	El sistema evitará el registro de metadatos duplicados
E3-105	La consulta de los expedientes digitalizados requerirá un usuario autenticado.

Versión a Futuro

No se agregarán requerimientos de confiabilidad para futuras versiones.

Seguridad

Versión Actual

Identificador	Descripción del Requerimiento
E1-105	Para evitar extravíos o daños a expedientes físicos Se definirá un procedimiento de escaneo en el que se distingan dos acciones particulares, una para el

	separado del expediente y otra para el escaneo como tal.
E3-106	Para realizar cualquier consulta a expedientes los usuarios deben estar identificados en el sistema.
E3-107	Se establecerá un algoritmo de encriptación para la contraseña.
E3-108	El acceso al sistema se realiza especificando usuario y contraseña con lo cual se configuran los correspondientes privilegios de usuario.
E3-109	El usuario se desconectará automáticamente cuando la aplicación se cierre inesperadamente.
A8-102	Se realizará un historial de operaciones hechas sobre el sistema por los usuarios, especificando tipo de operación, responsable y fecha.

Versión a Futuro

No se agregarán requerimientos de seguridad para futuras versiones.

Manejabilidad

Versión Actual

Identificador	Descripción del Requerimiento
101	El SIGED podrá ser usado con el ejecutable del sistema, el cual se instalará en las PC's con la herramienta de instalación correspondiente.
E5-102	El SIGED mostrará las interfaces correspondiente al rol con el que se autentifique el usuario.

Versión a Futuro

No se agregarán requerimientos de manejabilidad para futuras versiones.

Usabilidad

Versión Actual

Identificador	Descripción del Requerimiento
102	El diseño de la interfaz gráfica se realizará sobre un esquema de "Diseño Centrado en el Usuario", es decir, Se tratará de centrar el proceso de diseño en el usuario, implicando a los usuarios en el diseño para confirmar que se está desarrollando un sistema que en realidad satisface sus necesidades.
103	El diseño de la interfaz gráfica de usuario se ajustará a las directrices de logos e iconos que se especifican en el ICEO.
104	Los expedientes digitalizados serán mostrados en algún visor de imágenes.
105	El sistema debe estar escrito en lenguaje Español.

Versión a Futuro

No se agregarán requerimientos de usabilidad para futuras versiones.

Capacidad de Mantenimiento

Versión Actual

Identificador	Descripción del Requerimiento
S10-101	Se pretende implementar una serie de operaciones que faciliten la configuración de futuros requerimientos en cuanto base de datos y módulos del sistema.
106	Se establecerá un formato de codificación de manera que el código fuente incluya el nombre del módulo, el nombre del autor, fecha de creación y el propósito del módulo.
107	Se realizará un manual de usuario del sistema.

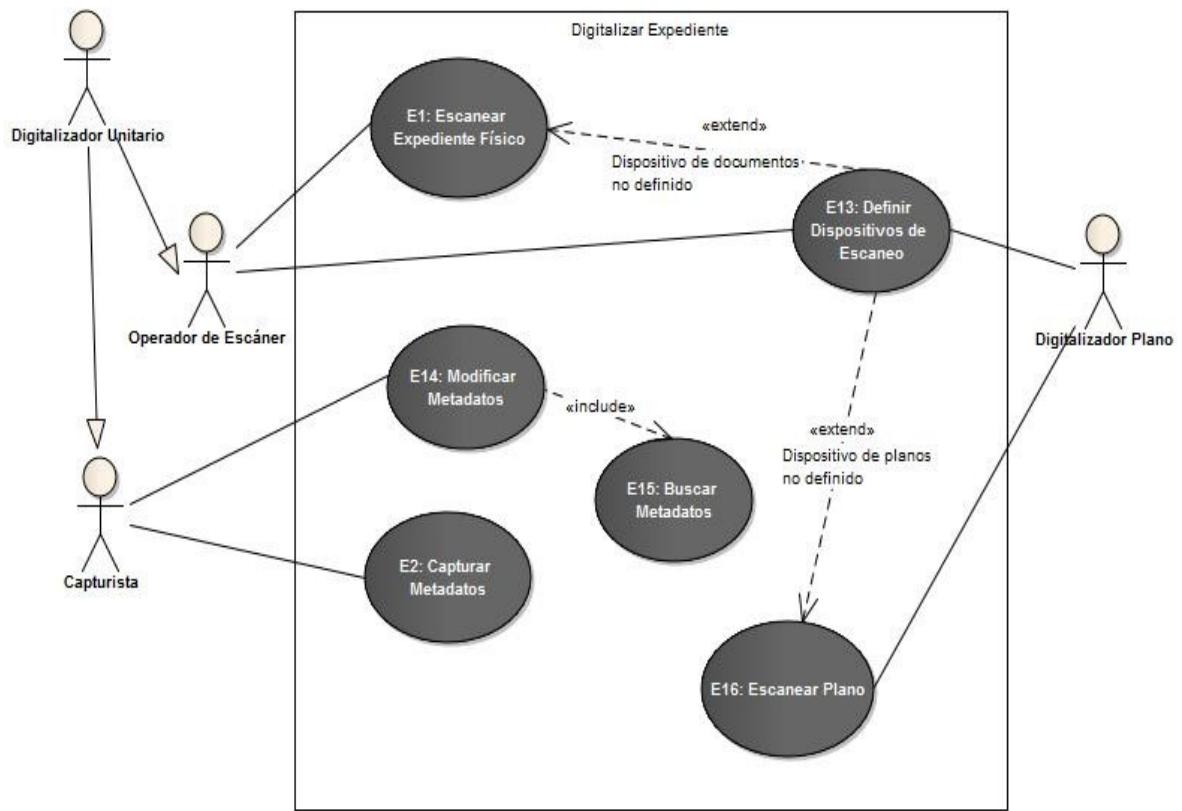
Versión a Futuro

Identificador	Descripción del Requerimiento
S11-101	El sistema contará con una gestión de catálogos que permitirá realizar cambios eficientes en las variables comunes del sistema.

Extensibilidad

Identificador	Descripción del Requerimiento
S10-102	El SIGED se podrá actualizar realizando una reinstalación en los clientes de manera local.
S10-103	En cada inicio de sesión se buscará si existen actualizaciones del sistema disponibles y se dispondrán a través de ftp los archivos que sean necesarios para llevar a cabo la actualización.

3.4. Casos de uso



3.4.1. Especificación de casos de uso

Caso de Uso Id y Nombre	E1: Escanear Expediente Físico
Descripción	Caso de uso para obtener las imágenes de los documentos físicos que compondrán el expediente digitalizado.
Actores	<ul style="list-style-type: none"> • Operador de Escáner • Digitalizador Unitario
Prioridad	Esencial
Pre-condiciones / Suposiciones	<ol style="list-style-type: none"> 1. El Usuario debe tener una sesión del sistema iniciada y estar en la pantalla de Bienvenida. 2. El Expediente Físico debe estar preparado (sin grapas u otros elementos que impidan su escaneo) y puesto en la bandeja del escáner. 3. Los expedientes físicos NO deben tener la etiqueta generada por el sistema que indica que ya ha sido digitalizado. 4. El escáner de documentos debe estar instalado y conectado en el equipo del usuario. 5. El escáner de documentos debe estar definido en el sistema. 6. Deben existir metadatos capturados del expediente a escanear.
Disparador	Se requiere la digitalización de un expediente físico.
Flujo de Evento Normal	<ol style="list-style-type: none"> 1. El usuario da clic en la opción “Escanear Expediente Físico”. 2. El sistema solicita al usuario ingrese el Número de Registro del expediente a digitalizar. 3. El usuario introduce el Número de Registro. 4. El sistema verifica si el Número de Registro existe. 5. El sistema solicita al usuario colocar los documentos en el Escáner. 6. El usuario da clic en Escanear. 7. El sistema verifica que el dispositivo de escaneo de documentos esté definido. 8. El sistema se conecta con el dispositivo de escaneo de documentos. 9. El sistema verifica que exista la ubicación predeterminada de escaneos. 10. El sistema verifica que exista la carpeta de imágenes del expediente en la ubicación predeterminada. 11. El sistema inicia la adquisición de las imágenes hasta que la bandeja queda vacía (cada imagen adquirida la guarda con el

	<p>formato de nombre: NoRegistro-NoImagen).</p> <ol style="list-style-type: none"> 12. El sistema pregunta al usuario si el escaneo ha finalizado o desea seguir agregando imágenes. 13. El usuario indica que el escaneo ha finalizado. 14. El sistema muestra en pantalla las imágenes escaneadas para revisión con las siguientes opciones: Confirmar Escaneo, Cancelar, Sustituir Imágenes y Agregar más. 15. El usuario elige Confirmar Escaneo. 16. El sistema genera el archivo DJVU con las imágenes escaneadas. 17. El sistema genera la llave de encriptación. 18. El sistema aplica el proceso de encriptación al archivo DJVU. 19. El sistema guarda el archivo encriptado en la carpeta de Expedientes Digitalizados del Servidor. 20. El sistema registra la llave de encriptación en la base de datos. 21. El sistema cambia el estatus del expediente a “Digitalizado” en la base de datos. 22. El sistema borra la carpeta de imágenes del expediente de la ubicación predeterminada. 23. El sistema registra el movimiento en la bitácora. 24. El sistema informa al usuario sobre el éxito de la operación. 25. El sistema regresa al paso 2.
Flujos Alternativos	<p>Flujo alternativo #3.1: El usuario da clic en Cancelar.</p> <ol style="list-style-type: none"> 3.1.1. El sistema regresa a la pantalla de Bienvenida. <p>Flujo alternativo #4.1: El Número de Registro NO existe.</p> <ol style="list-style-type: none"> 4.1.1. El sistema informa al usuario. 4.1.2. El sistema regresa al paso 2. <p>Flujo alternativo #6.1: El usuario da clic en Cancelar.</p> <ol style="list-style-type: none"> 6.1.1. El sistema regresa al paso 2. <p>Flujo alternativo #7.1: El sistema NO encuentra dispositivos de escaneo definidos o hace falta uno.</p> <ol style="list-style-type: none"> 7.1.1. El sistema ejecuta el proceso “Definir Dispositivos de Escaneo”. <p>Flujo alternativo #8.1: El sistema encuentra una excepción al conectarse con el dispositivo.</p> <ol style="list-style-type: none"> 8.1.1. El sistema informa al usuario sobre la excepción encontrada. 8.1.2. El sistema regresa al paso 2. <p>Flujo alternativo #9.1: La ubicación predeterminada no existe.</p>

9.1.1. El sistema crea la ubicación predeterminada.

9.1.2. El sistema continúa con el paso 10.

Flujo alternativo #10.1: La carpeta de imágenes del expediente no existe.

10.1.1. El sistema crea la carpeta de imágenes del expediente.

Flujo alternativo #11.1: El sistema encuentra una excepción al adquirir las imágenes.

11.1.1. El sistema borra la carpeta de imágenes del expediente.

11.1.2. El sistema informa al usuario sobre la excepción encontrada.

11.1.3. El sistema regresa al paso 2.

Flujo alternativo #13.1: El usuario cancela la operación.

13.1.1. El sistema solicita al usuario confirmar la cancelación.

13.1.2. El usuario confirma la cancelación.

13.1.3. El sistema borra la carpeta de imágenes del expediente.

13.1.4. El sistema regresa al paso 2.

Flujo alternativo #13.1.2: El usuario NO confirma la cancelación.

13.1.2.1. El sistema regresa al paso 12.

Flujo alternativo #13.2: El usuario elige Continuar Escaneo.

13.2.1. El sistema regresa al paso 11.

Flujo alternativo #15.1: El usuario elige Cancelar el escaneo.

15.1.1. El sistema solicita confirmar la cancelación.

15.1.2. El usuario confirma la cancelación.

15.1.3. El sistema borra la carpeta de imágenes del expediente.

15.1.4. El sistema regresa al paso 2.

Flujo alternativo #15.1.2: El usuario NO confirma la cancelación.

15.1.2.1. El sistema regresa al paso 14.

Flujo alternativo #15.2: El usuario elige Agregar más.

15.2.1. El sistema regresa al paso 11.

Flujo alternativo #15.3: El usuario elige Sustituir Imágenes.

15.3.1. El sistema obtiene los nombres de las imágenes a reemplazar.

15.3.2. El sistema regresa al paso 11 (las imágenes escaneadas serán guardadas con los nombres de las imágenes seleccionadas en el paso anterior).

1. El expediente digitalizado deberá estar creado y disponible para su consulta.

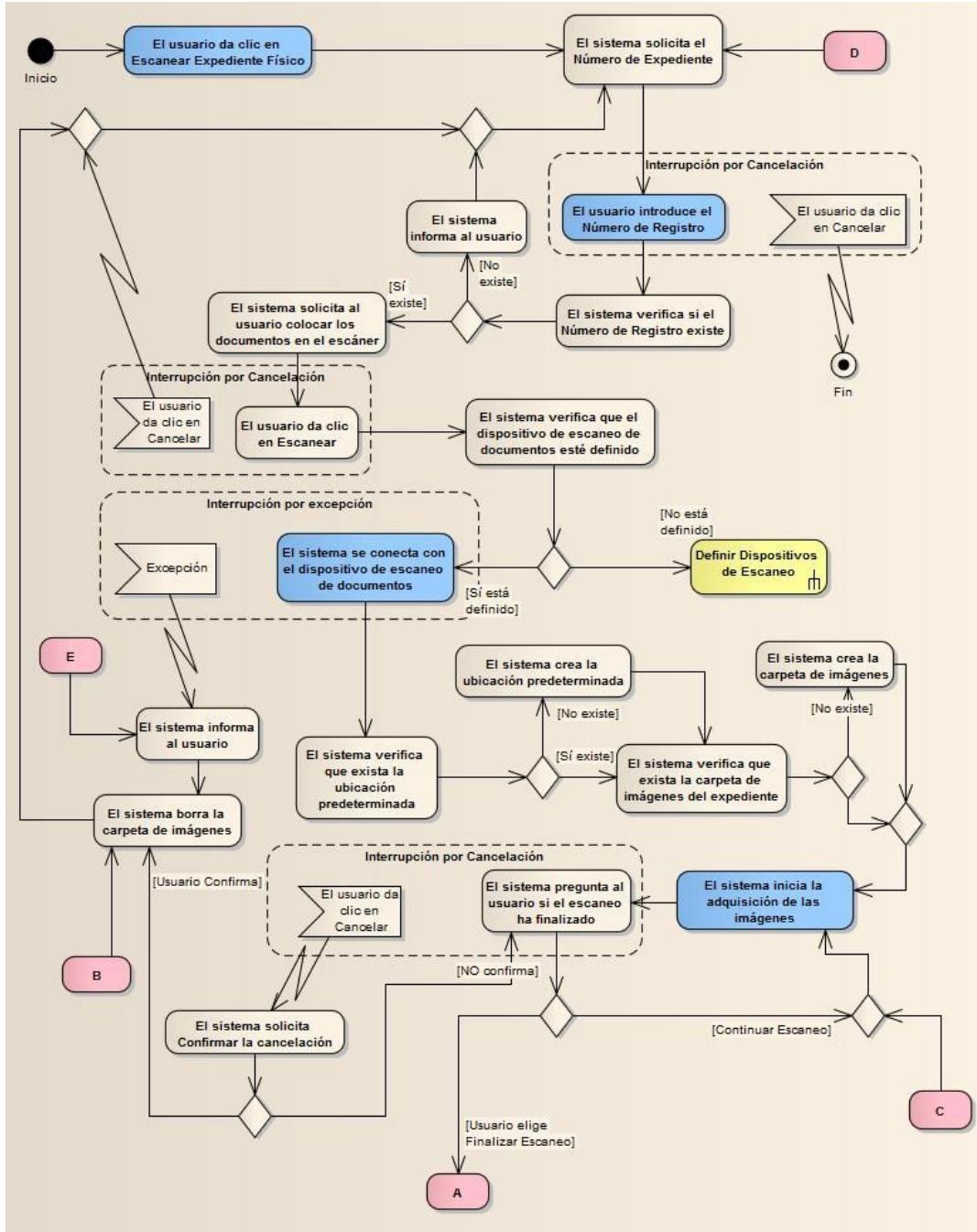
Requerimientos No-Funcionales	<ul style="list-style-type: none"> • E1-101 (Rendimiento) • E1-102, E1-103, E1-104 (Escalabilidad) • E1-105 (Seguridad) • 102, 103, 104, 105 (Usabilidad) • 106, 107 (Capacidad de Mantenimiento)
-------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

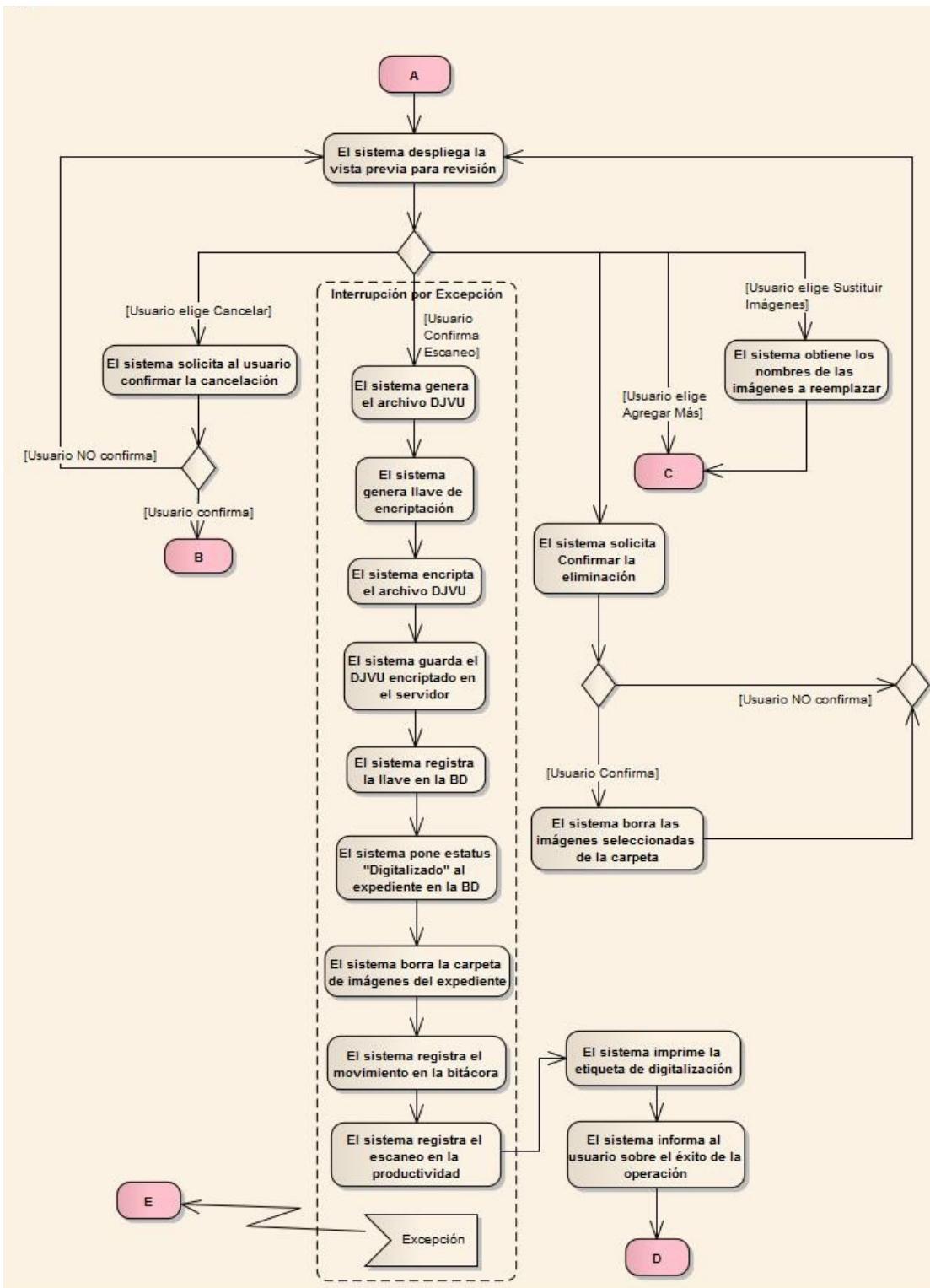
Caso de Uso Id y Nombre	E13: Definir Dispositivos de Escaneo
Descripción	Caso de uso que permite definir en el sistema qué dispositivos de escaneo se usarán y qué función tendrá cada uno (para documentos o para planos).
Actores	<ul style="list-style-type: none"> • Operador de Escáner • Digitalizador Unitario
Prioridad	Esencial
Pre-condiciones / Suposiciones	<ol style="list-style-type: none"> 1. El Usuario debe tener una sesión del sistema iniciada y estar en la pantalla de Bienvenida del Sistema. 2. Los escáneres deben estar instalados en el equipo del usuario. 3. Los escáneres instalados soportan el driver TWAIN.
Disparador	<ol style="list-style-type: none"> a) No existen dispositivos de escaneo definidos en el sistema. b) Uno de los dispositivos de escaneo se ha sustituido.
Flujo de Evento Normal	<ol style="list-style-type: none"> 1. El usuario da clic en la opción “Definir Dispositivos de Escaneo” del menú. 2. El sistema busca los dispositivos de escaneo instalados en el equipo que soportan el estándar TWAIN 3. El sistema carga las definiciones respectivas registradas de los dispositivos encontrados. 4. El usuario elige los dos escáneres que serán utilizados por el sistema y la función que realizará cada uno (para documentos o para planos). 5. El usuario da clic en el botón Guardar. 6. El sistema registra la definición de los dispositivos en la Base de Datos. 7. El sistema informa sobre el éxito de la operación. 8. El sistema regresa a la pantalla de Bienvenida.
Flujos Alternativos	Flujo alternativo #2.1: El sistema no encuentra dispositivos de escaneo instalados, conectados y encendidos en el equipo que soportan el estándar TWAIN

	<p>2.1.1. El sistema informa al usuario.</p> <p>2.1.2. El sistema regresa a la pantalla de Bienvenida del Sistema.</p> <p>Flujo alternativo #4.1: El usuario cancela la operación</p> <p>1.1.1. El sistema regresa a la pantalla de Bienvenida del Sistema.</p> <p>Flujo alternativo #5.1: El usuario cancela la operación</p> <p>5.1.1. El sistema regresa a la pantalla de Bienvenida del Sistema.</p> <p>Flujo alternativo #6.1: El sistema encuentra una excepción</p> <p>6.1.1. El sistema informa al usuario sobre la excepción encontrada.</p> <p>6.1.2. El sistema regresa a la pantalla de Bienvenida del Sistema.</p>
Post-condiciones	
Requerimientos No-Funcionales	<ol style="list-style-type: none"> Los dispositivos de escaneo quedarán definidos y listos para ser utilizados por otras funciones del Sistema. <ul style="list-style-type: none"> • 102, 103, 104, 105 (Usabilidad) • 106, 107 (Capacidad de Mantenimiento)

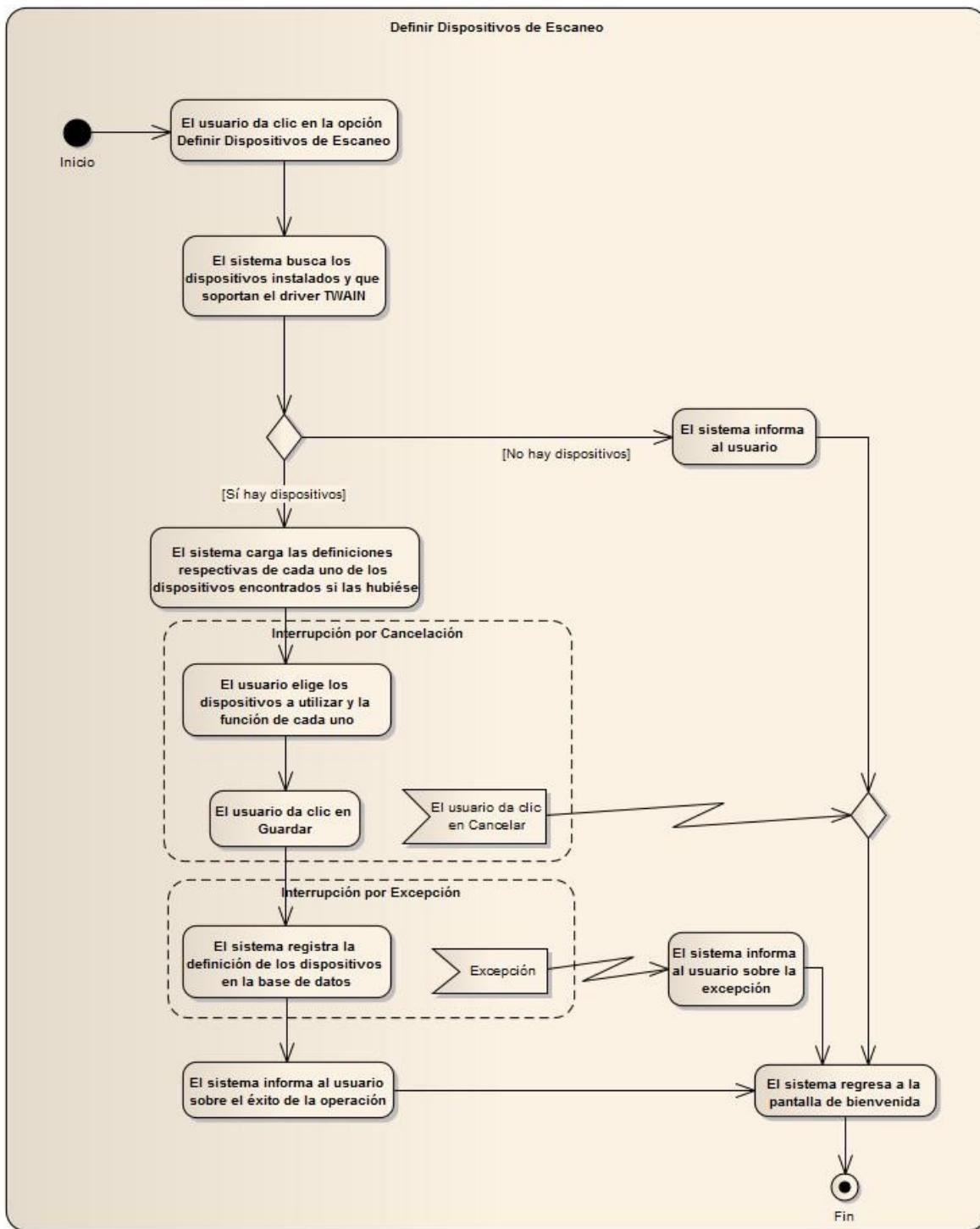
3.5. Diagramas de Actividad

Escanear Expediente Físico

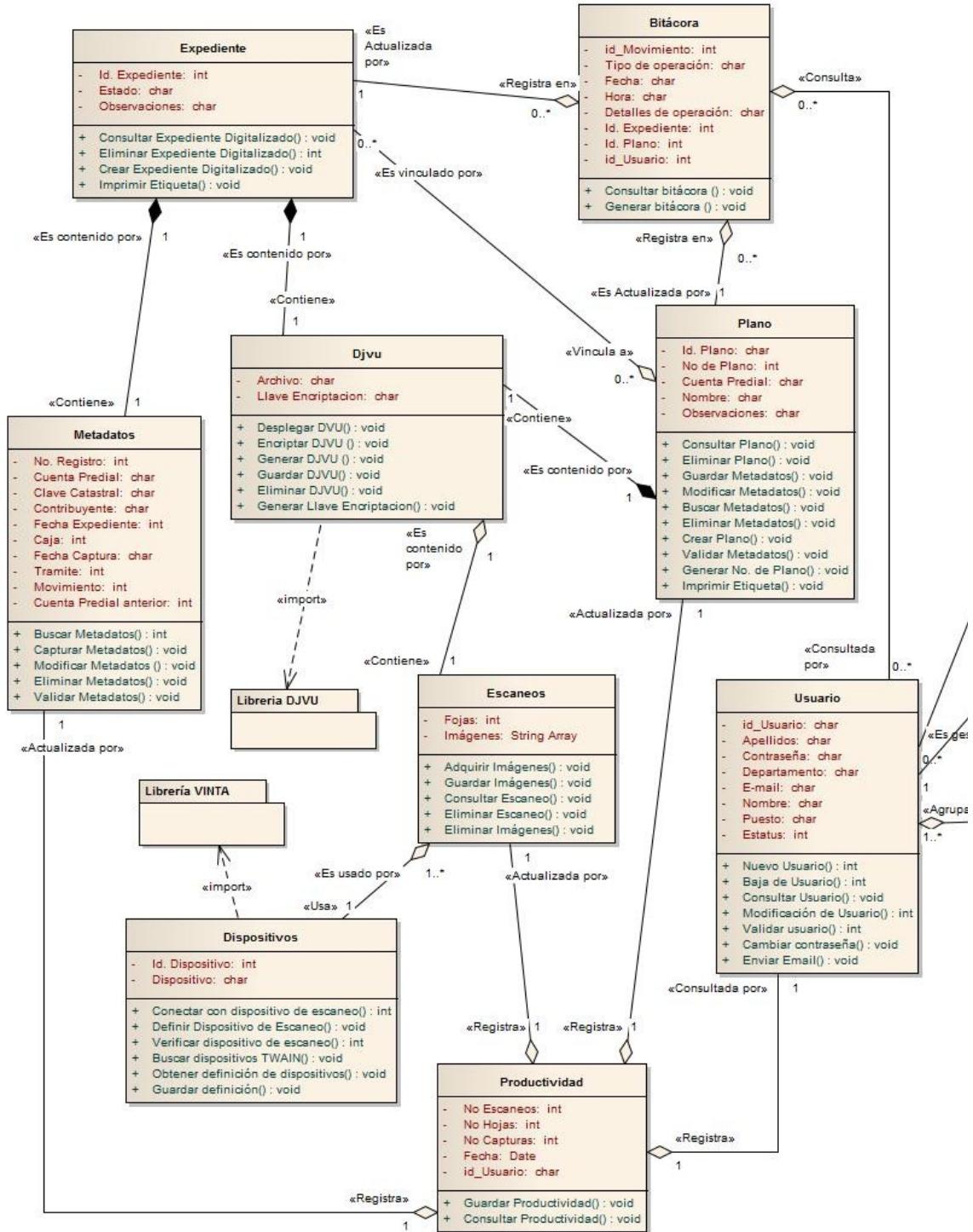




Definir Dispositivos de Escaneo

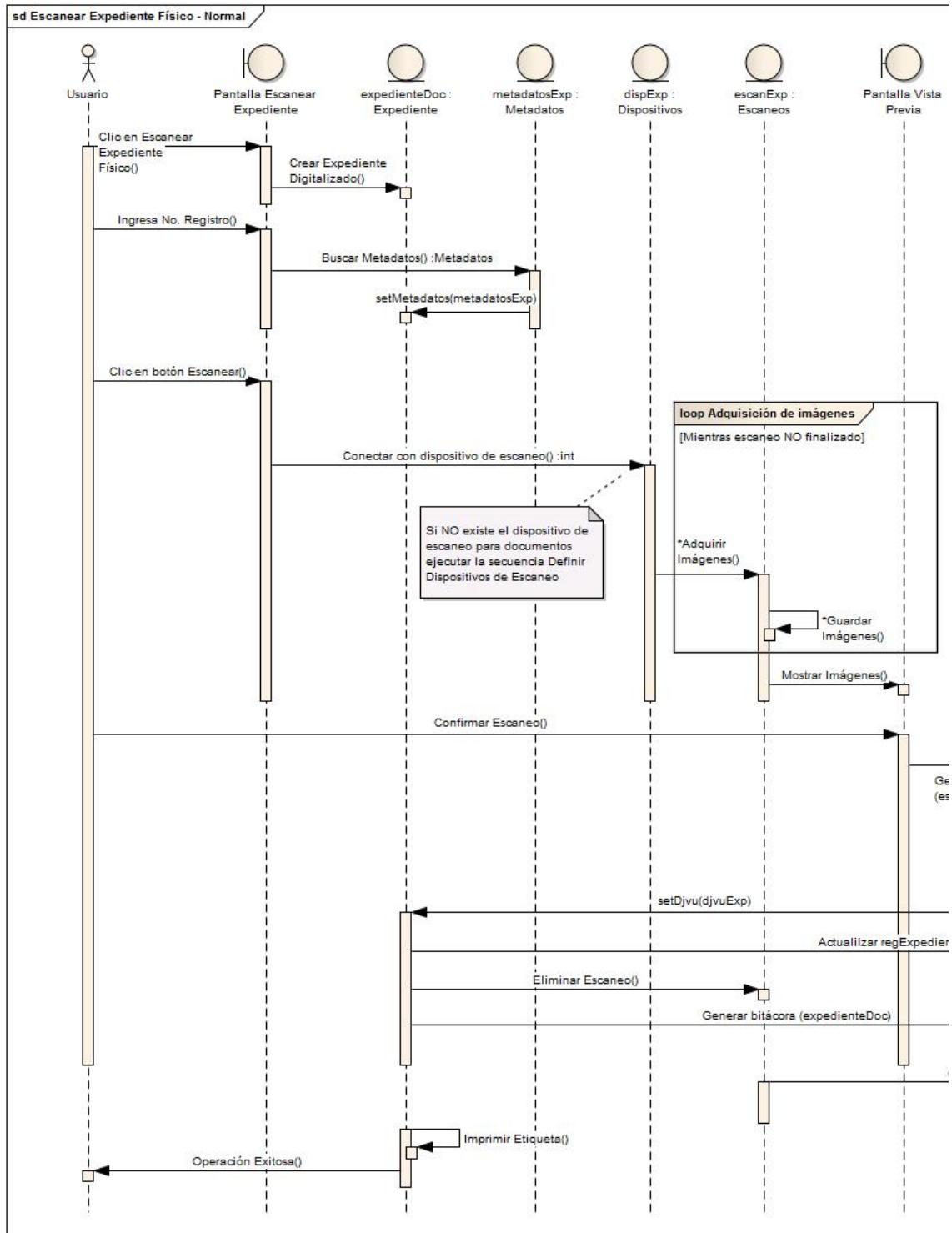


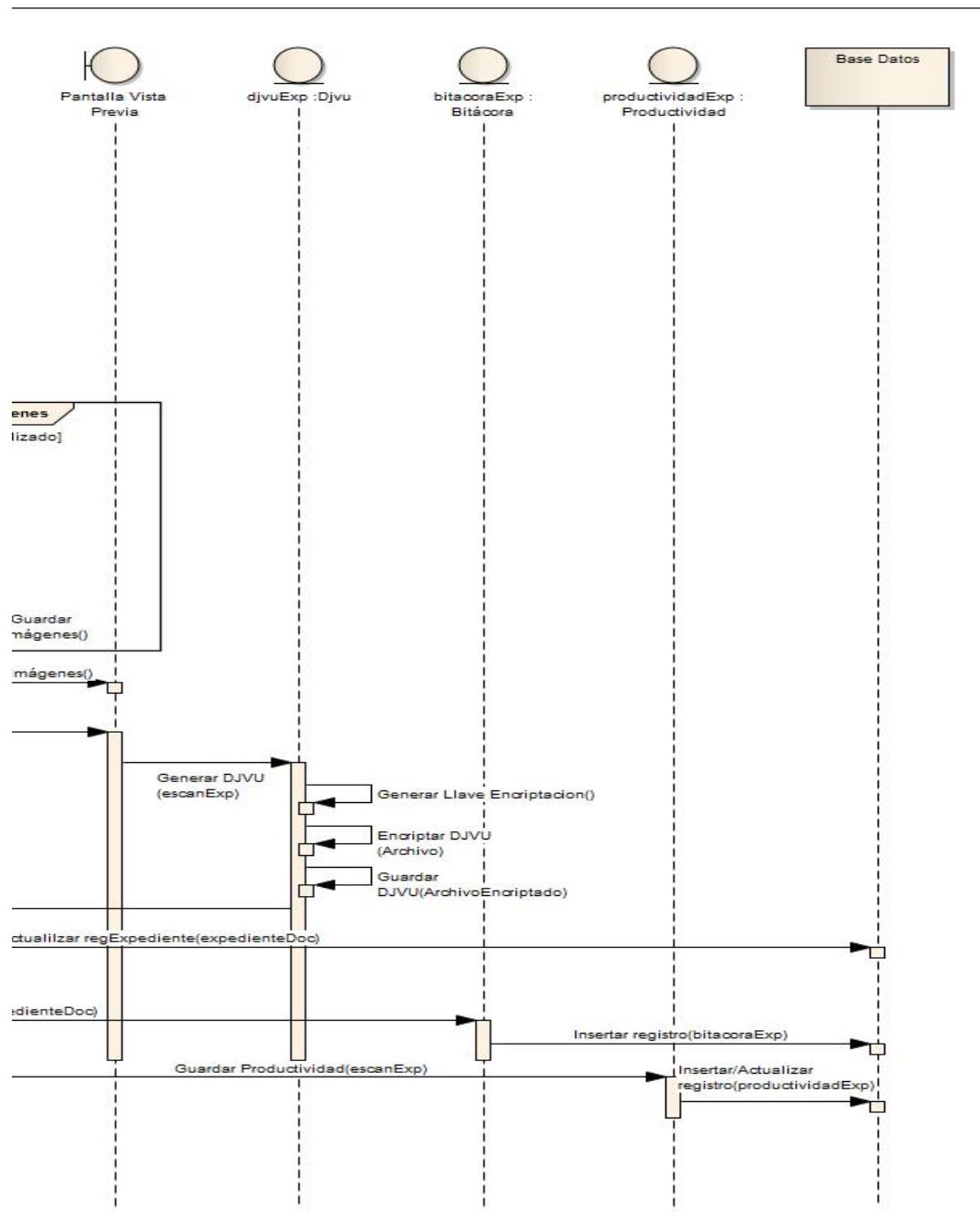
3.6. Diagrama de Clases



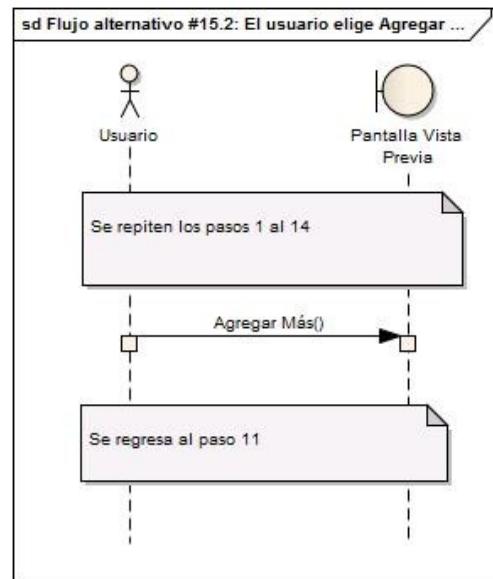
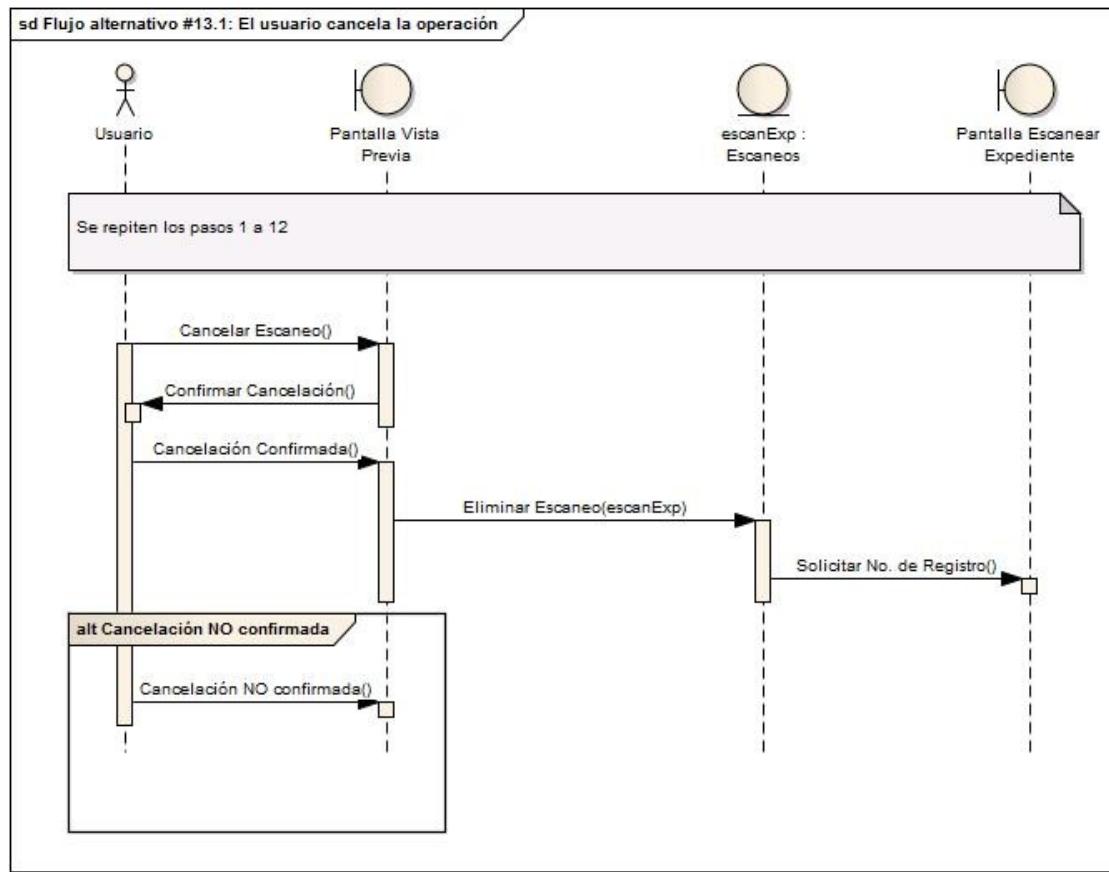
3.7. Diagramas de Secuencia

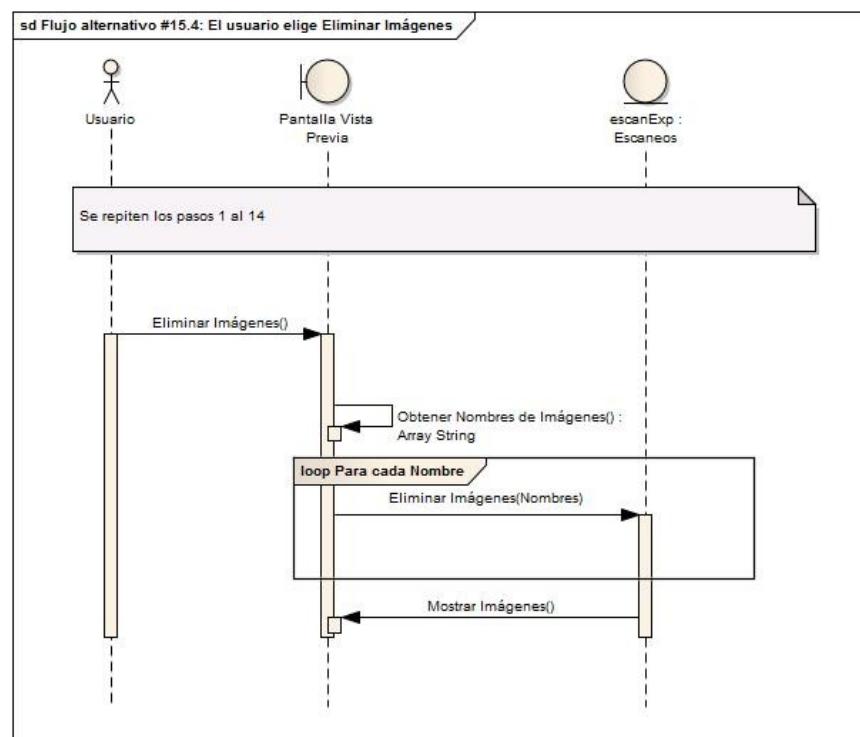
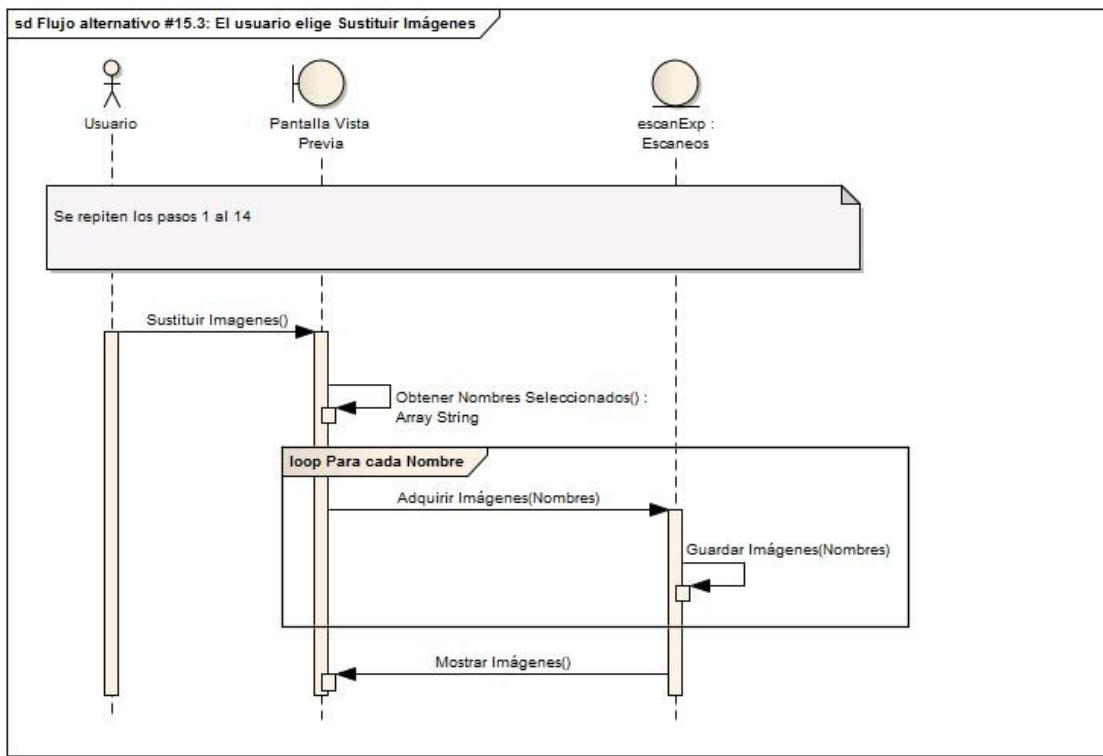
Escanear Expediente Físico – Flujo Normal



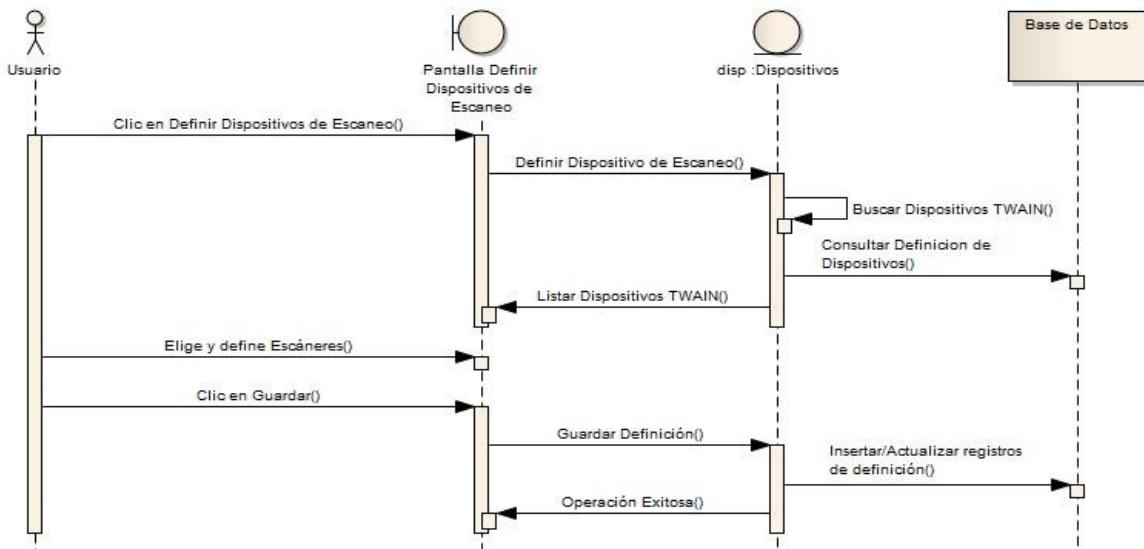


Escanear Expediente Físico – Flujos Alternativos

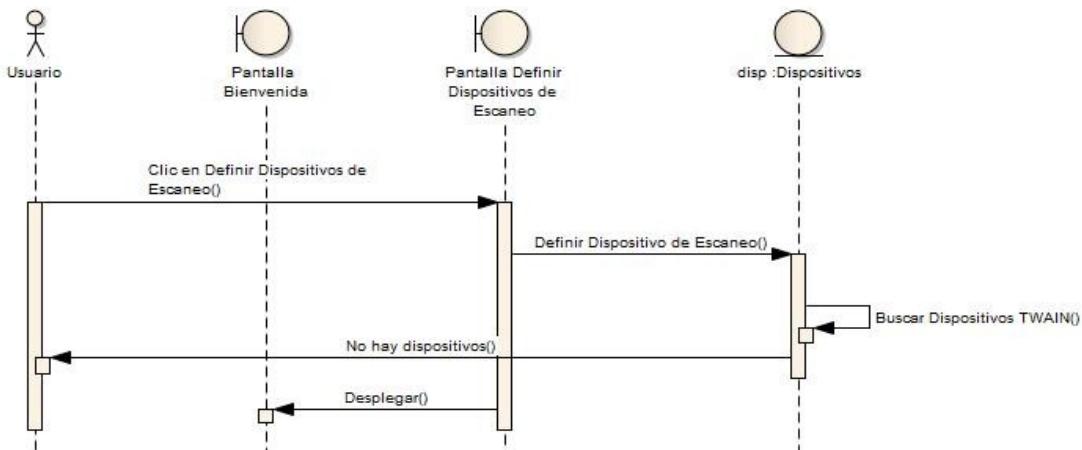




Definir Dispositivos de Escaneo – Flujo Normal



Definir Dispositivos de Escaneo – Flujo Alterno No hay dispositivos de escaneo

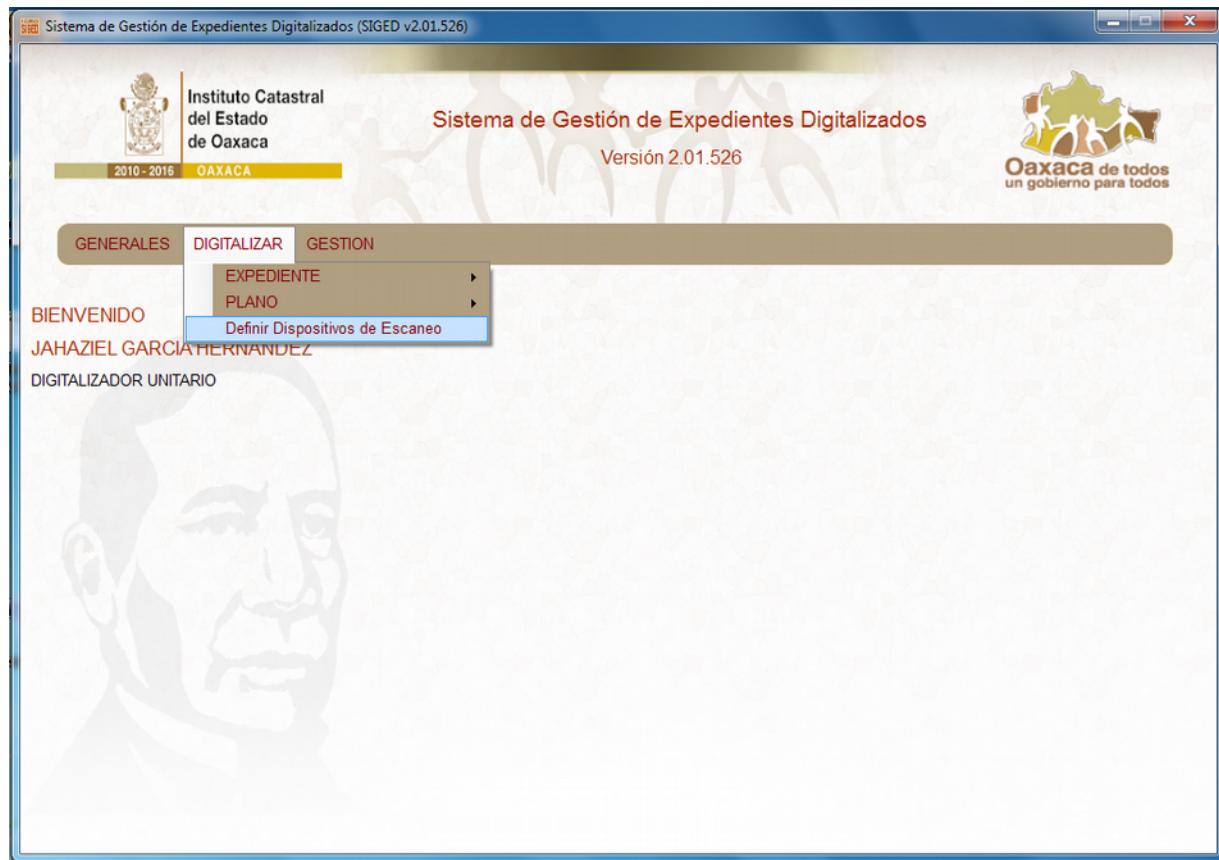


CAPÍTULO 4. DESARROLLO

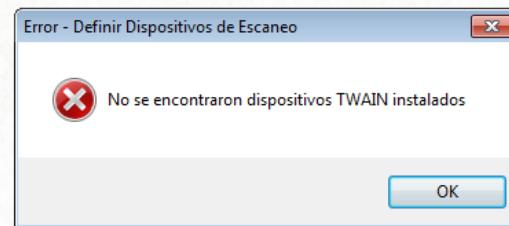
4.1. Interfaces del Sistema

4.1.1. Definir dispositivos de escaneo

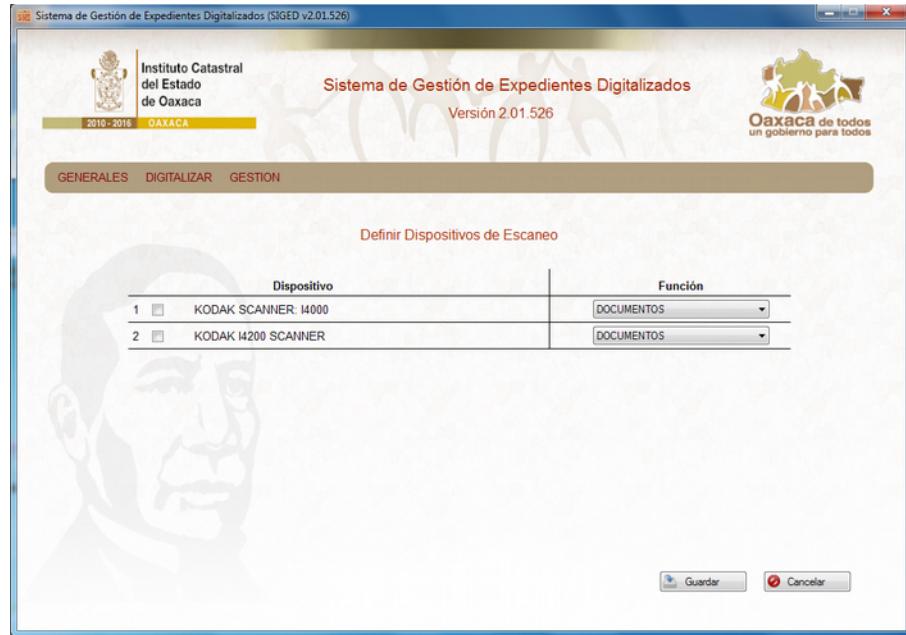
Acceso a la opción para definir los dispositivos de escaneo



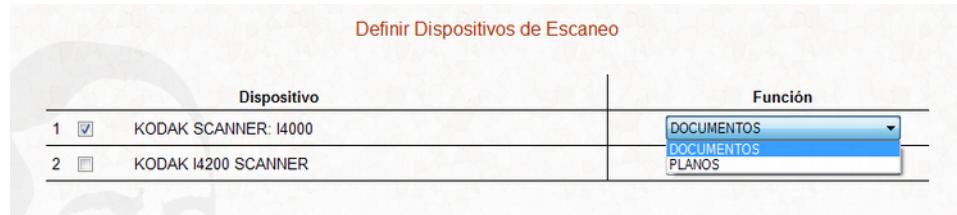
Si no se encontraron dispositivos TWAIN



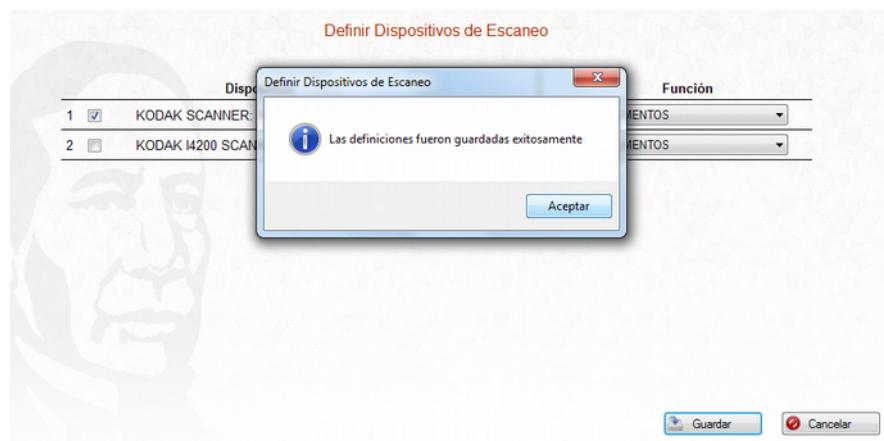
Si sí hay dispositivos TWAIN conectados y encendidos en el equipo



Funciones disponibles para asignar a los dispositivos



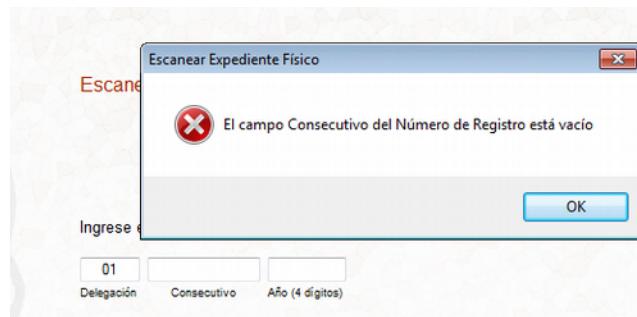
Al hacer clic en guardar



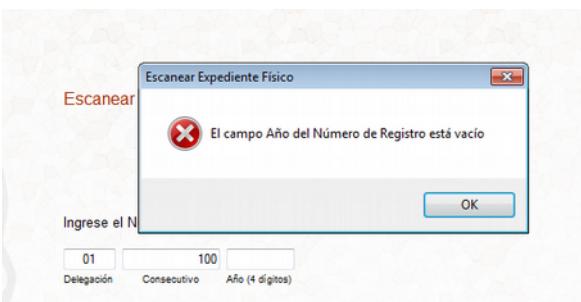
4.1.2. Escanear expediente



Si no ha ingresado el campo consecutivo del número de registro:



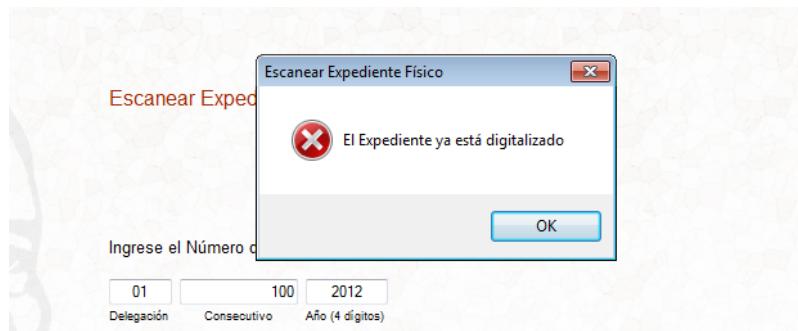
Si no ha ingresado el año del número de registro:



Si el expediente no está registrado:



Si el expediente ingresado ya está digitalizado:



Si el expediente indicado ya está siendo procesado por otro usuario:



Si el expediente ingresado es encontrado y es única coincidencia:

Escanear Expediente Físico

Verifique los datos del expediente

Núm. de Expediente:	04/112M-B.1-S61.1/2013/04731		
Núm. de Registro:	01/2640/13	Año del Núm. de Registro:	2013
Caja:	26/13		
Trámite:	1 - TRASLADO DE DOMINIO		
Movimiento:	105 - COMPRAVENTA		
Tipo:	TOTAL		
Núm. de Trámite:	004651/13		
Cuenta Predial:	84009*	Cuenta Anterior:	84009*
Clave Catastral:	001-409-00-005-010-014-000-0000		
Contribuyente:	SANCHEZ ESPINOZA MARTHA EUGENIA		
Observaciones:			

Si el expediente ingresado tiene más de un registro coincidente:

Escanear Expediente Físico

Seleccione el Expediente a escanear

	Id	Núm. Expediente	Núm. Registro	Ap. Paterno	Ap. Materno	Nombre	Año	Caja
▶	55559	04/112M-B.1-S61.1/2009/00256	01/002/09	VARGAS	MARTINEZ	MARIA MINERVA	2009	2/09
	60006	04/112M-B.1-S61.1/2009/03278	01/002/09	VARGAS	MARTINEZ	MARIA MINERVA	2009	17/09

Una vez seleccionado el registro que corresponde al expediente a digitalizar y dar clic en continuar, se muestra la pantalla al inicio de esta página, para cotejar los datos con el expediente físico.

Después de cotejar los datos con el expediente físico y hacer clic en continuar:



Si no hay dispositivo de escaneo definido:



Al dar clic al botón OK, si no se encuentran dispositivos TWAIN conectados:



Si sí hay dispositivo de escaneo definido, inicia el proceso de digitalización de los documentos:

Escanear Expediente Físico



El sistema está realizando el escaneo

Espere por favor...



Al finalizar el escaneo

Escanear Expediente Físico

El escaneo ha finalizado, por favor revise las imágenes obtenidas

The interface shows a large dashed rectangular area on the left where scanned documents would appear. To its right are several controls:

- A set of four navigation arrows: <<, <, >, and >> above a red number "1".
- A 2x3 grid of icons: a green arrow pointing left, a green arrow pointing right, a document with a blue checkmark, a green arrow pointing down, a pair of scissors, a green circular arrow, a trash can with a red X, and a red circular arrow.
- Four spinners labeled "Arriba:", "Abajo:", "Izquierda:", and "Derecha:" each with a value of "0".
- Buttons for "Reportar Error" (with a red error icon), "Agregar Más" (with a green plus icon), "Confirmar" (with a checkmark icon), "Urgente" (with a checkbox), "Aplicar" (with a checkmark icon), and "Cancelar" (with a red cancel icon).

Al confirmar el escaneo

Escanear Expediente Físico



El sistema está guardando el Expediente Digitalizado

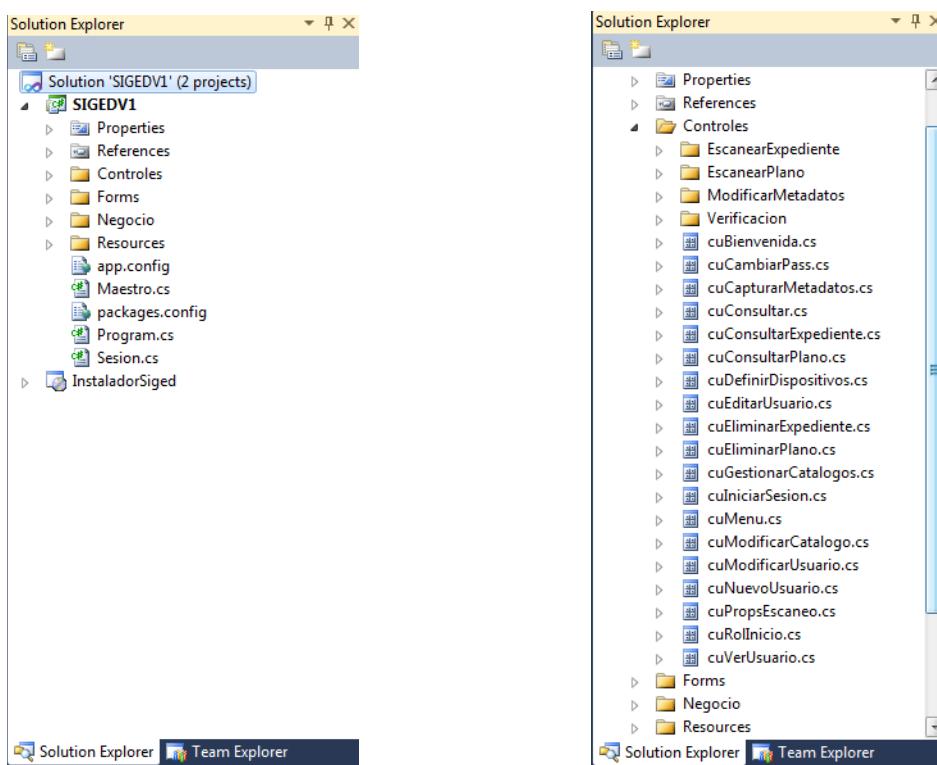
Espere por favor...

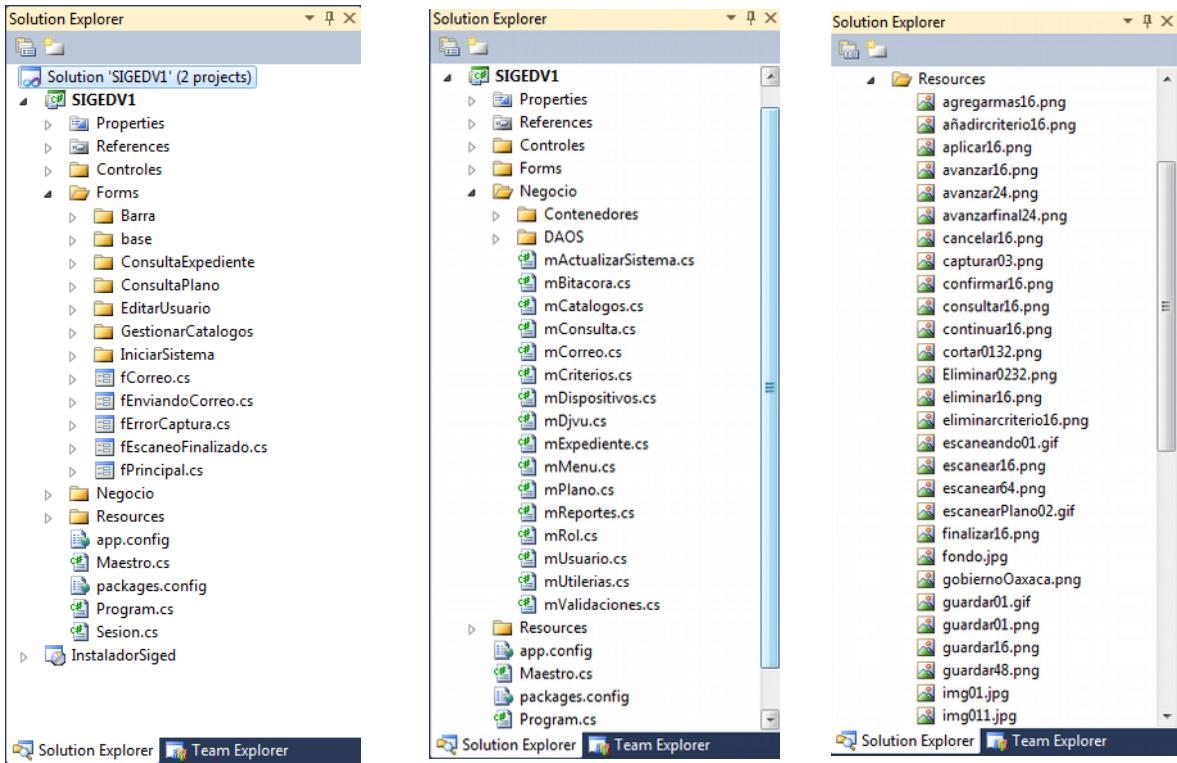
4.2. Código Fuente

A continuación se incluyen los fragmentos de código relacionados a los procesos para Definir el dispositivo de escaneo y para Escanear un expediente.

4.2.1. Estructura del código

En el desarrollo de la aplicación se usó como base el patrón de diseño MVC (Modelo Vista Controlador), por lo que se creó una estructura de carpetas para organizar los archivo de código que se asemejara al patrón:





4.2.2. Definir dispositivos de escaneo

A continuación se muestran los fragmentos de código en los que se hace uso de los elementos de la librería utilizada (VintaSoft Twain SDK) para la gestión de dispositivos de adquisición de imágenes que estén conectados a la computadora donde se esté ejecutando el sistema:

```
public void cuDefinirDispositivos()
{
    //buscar los dispositivos twain
    mDispositivos mdis = new mDispositivos();
    DeviceCollection dispositivos = mdis.BuscarDispositivosTwain();
    ArrayList dispWIA = mdis.ObtenerDispositivosWIA();
    if (dispositivos == null || dispositivos.Count == 0)
    {
        MessageBox.Show("No se encontraron dispositivos TWAIN instalados",
                       "Error - Definir Dispositivos de Escaneo", MessageBoxButtons.OK,
                       MessageBoxIcon.Error);
        ban = true;
    }
    else
    {
        ... - código ...
    }
}
```

```

        public DeviceCollection BuscarDispositivosTwain()
        {
            DeviceManager deviceManager = new DeviceManager();
            deviceManager.Dispose();
            deviceManager.Open();
            return deviceManager.Devices;
        }

        public ArrayList ObtenerDispositivosWIA()
        {
            var devMan = new WIA.DeviceManager();
            ArrayList disp = new ArrayList();
            for (int i = 1; i <= devMan.DeviceInfos.Count; i++)
            {
                if (devMan.DeviceInfos[i].Type.Equals(WIA.WiaDeviceType.ScannerDeviceType))
                {
                    disp.Add(devMan.DeviceInfos[i]);
                }
            }
            return disp;
        }
    
```

En la sección de Anexos, se puede ver el código completo de esta funcionalidad.

4.2.3. Escanear Expediente

En seguida se muestra el fragmento de código en el que se crea el archivo DJVU con las imágenes de cada una de las páginas de un expediente catastral, adquiridas desde el escáner, y que muestran el uso de la librería DJVU:

```

public cExpediente GuardarEscaneo(cExpediente cexp, cEscaneo cesc)
{
    //USO DE LA LICENCIA DEL SDK DJVU
    DjVu.LicenseManager.LicenseScript = "*****";
    cesc.Encoder.FinalizeSession();

    try
    {
        DjVu.Document doc = cesc.Encoder.GetEncodedDocument();
        doc.SaveAs(ceco.CarpetaExp + @"\" + cesc.NomDjvuFinal);
    }
    catch (DjVuException excep)
    {
        string error = excep.ToString();
    }
    //--USO DE LA LICENCIA DEL SDK DJVU

    mCatalogos mcat = new mCatalogos();
    cServidor servidorNor = mcat.ObtenerServidor();
    cServidor servidorFTP = new cServidor();
    servidorFTP = servidorNor;
}

```

```

    if (Sesion.VerificarError())
    {
        cexp.Estatus = -1;
        return cexp;
    }

    mUtilerias mut = new mUtilerias();
    mut.SubirArchivoFTP(servidorFTP, cesc.CarpetaServ, Path.Combine(cesc.CarpetaExp, cesc.NomDjvuFinal));
    if (Sesion.VerificarError())
    {
        cexp.Estatus = -1;
        return cexp;
    }

    cexp.Estatus = 2;//2 es el estatus de digitalizado
    cexp.Servidor = servidorFTP;

    //cargar hojas y paginas
    cexp.Hojas = cesc.Hojas / 2;
    cexp.Paginas = cesc.Imagenes.Count;

    //cargar hojas y paginas
    cexp.Hojas = cesc.Hojas / 2;
    cexp.Paginas = cesc.Imagenes.Count;

    //actualizar datos en la base de datos: archivo (con ruta), llave y estatus y bitacora
    dExpediente dexp = new dExpediente();
    dexp.RegistraDigitalizacion(cexp);
    if (Sesion.VerificarError())
    {
        cexp.Estatus = -1;
        return cexp;
    }

    return cexp;
}

```

En la sección de Anexos se incluye todo el código relacionado al escaneo de un expediente.

CONCLUSIONES

El SIGED, en funcionamiento desde mayo del 2012 en el ICEO, ha permitido mejorar los servicios que se ofrecen ahí. Logrando cumplir al 100% todos los objetivos para los que fue creado (reducción a 3 días o menos el tiempo de consulta, eliminar el riesgo de daño, alteración o pérdida de expedientes, y mejorando el control de acceso usando la autentificación y un syslog).

Sobre el desarrollo del SIGED, se puede concluir que la buena integración que hubo de las personas que colaboramos, fue un factor clave, para lograr el sistema

implementado, incluyendo a las autoridades del ICEO, que en todo momento apoyaron al equipo de desarrollo.

Al día de hoy, y por la experiencia adquirida de otras herramientas tecnológicas, cabe mencionar que el desarrollo del sistema pudo haberse hecho con más control, automatizando muchas tareas administrativas propias del desarrollo, e incluso en menor tiempo.

Queda como trabajo futuro los siguientes puntos:

- ◆ La integración del SIGED con el sistema integral de catastro, el cual aún está en desarrollo.
- ◆ El desarrollo de un módulo para digitalizar expedientes que se encuentran en legajos o libros.
- ◆ La implementación de pruebas unitarias e integrales, que ayuden a garantizar el buen funcionamiento del sistema.
- ◆ El uso de una herramienta de control de versiones, con la que se mejore el control de cambios del código.
- ◆ La refactorización de algunas partes de código siguiendo las recomendaciones de las buenas prácticas del desarrollo de software.

REFERENCIAS BIBLIOGRÁFICAS

- Cámara de Diputados (2014): Cámara de Diputados, Ley de Catastro para el Estado de Oaxaca, 2014
- Cámara de Diputados (2008): Cámara de Diputados, Ley de Protección de Datos Personales del Estado de Oaxaca, 2008
- Cámara de Diputados (2013): Cámara de Diputados, Ley de Transparencia y Acceso a la Información Pública del Estado de Oaxaca, 2013
- INEGI, SEDESOL (2011): INEGI, SEDESOL, Modelo Optimo de Catastro, 2011
- Real Academia Española (2016): Real Academia Española, Diccionario de la lengua española, 2016, <http://dle.rae.es/>
- Rodríguez, Hugo (2003): Rodríguez, Hugo, Curso de Imagen Digital, 2003

- Ordoñez Santiago, Cristian Andrés (2005): Ordoñez Santiago, Cristian Andrés, Formatos de Imagen Digital, 2005
- Adobe (2016): Adobe, PDF, 2016, <https://acrobat.adobe.com/mx/es/why-adobe/about-adobe-pdf.html>
- DJVU Org (2016): DjVu Org, DJVU Format, 2016, <http://djvu.org/>
- Giménez, Joaquín (2005): Giménez, Joaquín, El escáner y sus en la digitalización, 2005, http://www.mati.unam.mx/index.php?option=com_content&task=view&id=69&Itemid=47
- BARRERA RIVERA, MÓNICA MARÍA DEL ROSARIO (2009): BARRERA RIVERA, MÓNICA MARÍA DEL ROSARIO, DIGITALIZACIÓN DE ARCHIVOS HISTÓRICOS. EL CASO DE LOS ARCHIVOS JUDICIALES FEDERALES: 1815 – 2003, 2009
- Bermúdez Muñoz, María Teresa (2011): Bermúdez Muñoz, María Teresa, GUIA PARA DIGITALIZAR DOCUMENTOS, 2011
- Archivo General de Castilla y León (2010): Archivo General de Castilla y León, Recomendaciones para la digitalización de documentos en los archivos, 2010
- IFLA E ICA (2002): IFLA E ICA, DIRECTRICES PARA PROYECTOS DE DIGITALIZACIÓN, 2002
- Sommerville, Iam (2005): Sommerville, Iam, Ingeniería de Software, 2005
- Pressman, Roger S. (2010): Pressman, Roger S., Ingeniería de Software Un enfoque práctico, 2010
- Hanmer, Robert (2013): Hanmer, Robert, Pattern-Oriented Software Architecture for Dummies, 2013
- Silberschatz, Abraham (2002): Abraham Silberschatz, Henry F. Korth, S. Sudarshan, FUNDAMENTOS DE BASES DE DATOS, 2002
- SUN MICROSYSTEMS (2001): SUN MICROSYSTEMS, SUNTONE ARCHITECTURE METHODOLOGY A 3-DIMENSIONAL APPROACH TO ARCHITECTURAL DESIGN, 2001

ANEXOS

Artículos de la Ley de Catastro para el Estado de Oaxaca que se consideraron en el desarrollo del SIGED:

ARTÍCULO 1.- La presente Ley es de orden e interés públicos y de observancia general en todo el Estado, correspondiendo su aplicación e interpretación en el ámbito administrativo a las autoridades que la propia Ley establece.

ARTÍCULO 2.- Esta Ley tiene por objeto:

I. Normar las funciones relativas al Catastro de los bienes inmuebles, entendiéndose para efectos de esta ley, el suelo y las construcciones adheridas a él, ubicados en el territorio del Estado; y establecer las bases para su organización administrativa.

ARTÍCULO 3.- El catastro es un sistema de información territorial para usos múltiples, estructurado por los registros documentales, gráficos y alfanuméricos que contienen la información cuantitativa y cualitativa de los bienes inmuebles ubicados en el territorio del Estado.

ARTÍCULO 4.- Las funciones catastrales son las siguientes:

I. La identificación, descripción, delimitación, mensura y valuación de los bienes inmuebles ubicados en el territorio del Estado.

IV. La determinación de los valores catastrales de los bienes inmuebles, para efecto de las contribuciones municipales sobre la propiedad inmobiliaria, con base en los valores unitarios del suelo y de las construcciones y en los procedimientos técnicos valuatorios.

ARTÍCULO 5.- Los actos y resoluciones en materia de catastro serán tramitados en la forma, términos y procedimientos establecidos en la presente Ley, su Reglamento y en los Instructivos Técnicos Catastrales.

A falta de disposición expresa, se consideran como normas supletorias las disposiciones contenidas en el Código Fiscal del Estado, las del Código Civil y las de sus procedimientos.

ARTÍCULO 6.- Para lograr la integración del sistema de información territorial y su actualización permanente, todo propietario o poseedor de bienes inmuebles ubicados en el territorio del Estado, están obligados a manifestar:

I. La existencia de dichos bienes;

II. Los actos traslativos de dominio celebrados ante notarios y registradores;

III. Las características del inmueble.

Las manifestaciones deberán hacerse en un plazo de treinta días naturales contados a partir de la expedición de los documentos en los que se haga constar la posesión, propiedad o la fecha de celebración de los actos traslativos de

dominio, en los formatos oficiales que para el caso apruebe y expida el Instituto Catastral del Estado de Oaxaca, acompañando los documentos o planos que señala esta Ley y su Reglamento.

Dichas manifestaciones o avisos generarán el pago de derechos por servicios catastrales en términos de la Ley Estatal de Derechos para el ejercicio fiscal que corresponda, a cargo del propietario o poseedor.

No se eximen de la obligación anterior, a los propietarios o poseedores de predios que por disposición de la Ley respectiva estén exentos del pago de contribuciones a la propiedad inmobiliaria.

Para los mismos fines los gobiernos federal, estatal y municipal a través de sus dependencias, entidades, organismos, instituciones, programas, convenios y cualquier otro que tenga funciones relacionadas con la propiedad inmobiliaria o tenencia de la tierra incluida su regularización, deben manifestar los bienes inmuebles, áreas, zonas, vías o cualquier otro concepto que se encuentre vinculado con la propiedad inmobiliaria.

ARTÍCULO 7.- Todos los bienes inmuebles ubicados en el territorio del Estado, sin excepción, deberán estar inscritos en el sistema de información territorial a cargo del Instituto Catastral del Estado de Oaxaca.

ARTÍCULO 9.- Son autoridades en materia de Catastro:

I. El Gobernador del Estado.

II. El Secretario de Finanzas del Estado de Oaxaca.

III. El Director General del Instituto Catastral del Estado de Oaxaca.

IV. Los Coordinadores de Área, notificadores y verificadores adscritos al Instituto Catastral del Estado de Oaxaca.

V. Los Delegados Catastrales del Instituto.

ARTICULO 12 BIS.- Compete a los Delegados Catastrales:

I. Realizar las Operaciones Catastrales referentes a la identificación, descripción, delimitación, mensura, inscripción, valuación y revaluación de los bienes inmuebles ubicados dentro de la circunscripción territorial de la delegación a su cargo en estricto apego a las Tablas de Valores Unitarios de Suelo y

Construcciones e Instructivo Técnico de Valuación, aprobados por el Congreso del Estado;

II. Solicitar a los propietarios o poseedores de bienes inmuebles datos y documentos que les permitan ratificar o rectificar la información proporcionada, registrar información veraz, confiable y actualizada en el padrón catastral.

ARTÍCULO 13.- El Instituto Catastral del Estado de Oaxaca es un órgano descentrado de la Administración Pública Estatal, dotado de autonomía administrativa, técnica y operativa para el ejercicio de las atribuciones que esta Ley le confiere y jerárquicamente subordinado a la Secretaría de Finanzas.

ARTÍCULO 17.- El Instituto Catastral del Estado de Oaxaca tiene las siguientes atribuciones:

I. Integrar y administrar el Catastro del Estado.

...

VIII. Asignar cuando así proceda, cuenta catastral y clave catastral a cada bien inmueble.

IX. Determinar los valores catastrales de cada bien inmueble en estricto apego a las Tablas de Valores Unitarios de Suelo y Construcciones, los Planos de Zonificación Catastral y el Instructivo Técnico de Valuación aprobados por el Congreso del Estado, y asimismo conforme a lo dispuesto por la presente Ley y su Reglamento;

X. Inscribir cuando así proceda los bienes inmuebles en el Padrón Catastral y mantenerlo actualizado, como lo disponen los artículos referentes en esta Ley y su Reglamento.

XI. Registrar oportunamente los cambios que se operen en la propiedad inmueble y que por cualquier concepto alteren los datos contenidos en los registros catastrales.

...

XXX. Expedir certificaciones sobre documentos o datos asentados en los registros catastrales y notificar a los interesados las operaciones catastrales efectuadas;

...

XXXV.- Ejercer las operaciones catastrales;

...

XXXIX.- Expedir cédula de datos catastrales

ARTÍCULO 23.- Las operaciones catastrales tienen por finalidad efectuar la identificación, descripción, delimitación, localización, mensura de los bienes inmuebles, así como su inscripción en los registros catastrales, su valuación y la integración de la información relativa a sus características y elementos físicos.

ARTICULO 23 BIS.- Para los efectos de esta Ley, se considerarán operaciones catastrales las siguientes:

I. Levantamiento catastral;

II. Verificación física;

III. Elaboración y actualización de la cartografía catastral;

IV. Valuación;

V. Revaluación;

VI. Control, conservación, actualización y archivo de los registros catastrales;

VII. Inscripción en el Catastro de los bienes inmuebles y sus modificaciones;

VIII. Elaboración de propuestas de Tablas de Valores Unitarios de Suelo y Construcciones y el Instructivo Técnico de Valuación;

IX. Realizar estudios técnicos para la elaboración de las tablas de valores unitarios de suelo y construcciones;

X. Zonificación catastral;

XI. Elaboración y determinación de las tipologías constructivas; y,

XII. Certificación de los avalúos.

XIII. Cancelación de registro de cuenta catastral en el Sistema de Información Territorial por resolución judicial, o cuando se trate de duplicidad de cuentas, y éstas amparen el mismo bien inmueble a nombre del mismo poseedor o propietario; y

XIV. Deslinde catastral.

ARTÍCULO 23 Bis 1.- El cobro de los derechos que corresponden a las operaciones catastrales señaladas en el artículo 23 Bis de esta Ley, se

determinarán con base en la Ley Estatal de Derechos.

Para efectos de esta ley se entiende por:

- a) Valor de la operación: aquel que fijen las partes en el acto traslativo de dominio; dicho valor deberá ser declarado por los notarios públicos o cualquier otro fedatario público que intervengan en la celebración de contratos que tengan por objeto transmitir o modificar el dominio directo de un bien inmueble; y
- b) Tabla de Valores Unitarios de Suelo y Construcciones: aquella que contiene los valores de mercado de la propiedad raíz.

Las Tablas de Valores Unitarios de Suelo y Construcciones podrán ser objeto de revisión y actualización cada año o cuando surjan circunstancias que puedan afectar el valor de la propiedad inmobiliaria.

ARTÍCULO 51.- Los propietarios y poseedores de bienes inmuebles tienen derecho a:

I. Que se les reciban las manifestaciones, avisos, solicitudes y escritos relacionados con las funciones y atribuciones propias del Instituto Catastral del Estado de Oaxaca y éstas sean atendidas o contestadas en un término no mayor de ocho días hábiles.

...

V. Interponer los recursos previstos en esta Ley.

VI. Conocer los resultados de los avalúos que señala el artículo 17 de esta Ley; y

VII. Enterarse de los requisitos y formatos que para cada tipo de trámite catastral se requieren, así como de los derechos e impuestos que se generen por cada uno de ellos;

VIII. Los demás que establezcan esta Ley y su Reglamento.

ARTÍCULO 52.- Cuando en las manifestaciones o avisos a que se refiere esta Ley no se expresen los datos o no se acompañen los documentos o planos también requeridos, el Instituto Catastral del Estado de Oaxaca no las admitirá para su trámite.

Sin perjuicio de lo establecido en este artículo, el Instituto Catastral del Estado de Oaxaca analizará y determinará la procedencia o improcedencia de las

manifestaciones, avisos o escritos que presenten los fedatarios públicos y los particulares, cuidando en todo momento que los documentos presentados cumplan con las normas y procedimientos aplicables en las leyes vigentes.

ARTÍCULO 58.- El Instituto Catastral del Estado de Oaxaca, expedirá la información existente en los archivos catastrales a los propietarios o poseedores de los inmuebles registrados que lo soliciten, previo el pago de derechos correspondientes, en los términos de esta Ley y su Reglamento.

ARTÍCULO 60.- Los Notarios Pùblicos, fedatarios y los organismos federales, estatales o municipales encargados de la regularización de la tenencia de la tierra, antes de proceder a la autorización de la escritura o título de propiedad, deberán contar con avalúo catastral y cédula catastral.

Las autoridades registrales no harán inscripción alguna de contratos, títulos o resoluciones judiciales que tengan por objeto transmitir o modificar el dominio de un bien inmueble así como aquellos que constituyan la propiedad o generen derechos de propiedad, mientras no se les exhiba avalúo catastral y cédula catastral.

ARTÍCULO 70.- Las resoluciones y cualquier otro acto u operación catastral que deba hacerse del conocimiento de los propietarios, poseedores o sus representantes legales, se notificarán de conformidad con las disposiciones que para estos actos prevé el Código Fiscal del Estado.

ARTÍCULO 76.- Los propietarios o poseedores de bienes inmuebles ubicados en el territorio del Estado, podrán interponer ante el Instituto por sí o por conducto de sus representantes legales el recurso de revocación en contra de los actos de las autoridades catastrales, en los términos y conforme a los procedimientos previstos en el presente capítulo.

Artículos de la Ley de Protección de Datos Personales del Estado de Oaxaca considerados en el desarrollo del SIGED

“**ARTÍCULO 10.-** Los propósitos para los cuales se solicitan datos personales

deben ser especificados al momento de su obtención, y el uso subsecuente de ellos debe limitarse al cumplimiento de tales propósitos u otros compatibles con éstos, y que sean especificados en cada caso en que varíen los propósitos iniciales.

La obtención de datos personales no puede hacerse por medios desleales, fraudulentos o en forma contraria a las disposiciones legales aplicables.

ARTÍCULO 11.- Los datos personales no serán divulgados o puestos a disposición de terceros para usos diferentes a los especificados por quien los obtuvo, excepto en los casos que prevean expresamente las Leyes.

Los sujetos obligados salvaguardarán los datos personales contenidos en los padrones de beneficiarios de los programas que desarrollen, cuando la publicación de estos datos pueda inducir o produzcan discriminación o estigmatización en la sociedad.

ARTÍCULO 16.- Para el tratamiento de los datos personales, será necesario el consentimiento del titular de la información, con excepción de los siguientes casos:

- I. Cuando se trate de la realización de las funciones propias de la administración pública en su ámbito de competencia;
- II. Cuando se transmitan entre sujetos obligados, siempre y cuando los datos personales se utilicen para el ejercicio de sus facultades;
- III. Cuando exista una solicitud u orden de autoridad en materia de procuración o administración de justicia;
- IV. Cuando se trate de los datos personales de las partes en contratos civiles, laborales, comerciales o administrativos;
- V. Cuando sean necesarios para el tratamiento médico del titular;
- VI. Cuando se trate de razones estadísticas, científicas o de interés general previstas en la ley, siempre que no puedan asociarse los datos personales con el individuo a quien se refieren;
- VII. A terceros, cuando se contrate la prestación de un servicio que requiera el tratamiento de datos personales. Dichos terceros no podrán utilizar los datos personales para propósitos distintos a aquellos para los cuales se les hubieren

transmitido; y

VIII. En los demás casos que establezcan las leyes.

El titular de la información podrá revocar el consentimiento mencionado en este artículo, pero no tendrá efectos retroactivos.

Los servidores públicos, profesionales, trabajadores y otras personas físicas o morales de naturaleza privada, que por razón de sus actividades tengan acceso a Sistemas de datos personales, en poder de los sujetos obligados estarán obligados a mantener la confidencialidad de los mismos. Esta obligación subsistirá aún después de finalizar las relaciones que les dieron acceso a los datos personales. La contravención a esta disposición será sancionada de conformidad con la legislación penal.

Los datos personales relativos a la salud deberán ser operados por profesionales e instituciones de acuerdo con la legislación sanitaria local o federal, conservando la confidencialidad de los mismos de acuerdo con la presente Ley.

ARTÍCULO 18.- El titular de los datos personales, previa acreditación de su identidad, tendrá derecho a solicitar y obtener en forma gratuita, información de sus propios datos personales en intervalos no inferiores a doce meses, salvo que se acredite un interés legítimo al efecto. En su caso el solicitante cubrirá únicamente los gastos de envío y reproducción aplicables.

Se exceptúan de lo dispuesto en el párrafo anterior, los actos que por disposición de las Leyes generen el pago de derechos.

Para el caso de personas fallecidas, el derecho de acceso corresponderá a sus sucesores.

ARTÍCULO 31.- Sólo los titulares de la información o sus representantes, podrán solicitar y obtener gratuitamente, previa acreditación ante la Unidad de Enlace correspondiente el acceso, consulta, rectificación o cancelación de la información personal del titular, registrada en los sistemas de datos personales de los sujetos obligados. Se exceptúan de lo anterior, los actos que por disposición de las Leyes generen el pago de derechos.

La entrega, modificación o cancelación de dicha información, se realizará en un

plazo no mayor a 15 días hábiles contados desde la presentación de la solicitud. Este plazo podrá ser ampliado por una sola vez por un período igual, siempre que exista causa justificada. Esta disposición no se aplicará para los sistemas de datos personales que se encuentren regulados por disposiciones legales específicas.

ARTÍCULO 36.- El recurso de revisión tiene por objeto garantizar que en los actos y resoluciones de los sujetos obligados se respeten las garantías de legalidad, seguridad jurídica y el cumplimiento de esta Ley.

El solicitante a quien se le niegue el acceso, consulta, corrección, supresión o modificación de sus datos personales podrá interponer por sí mismo o a través de su representante, el recurso de revisión ante el Instituto o ante la Unidad de Enlace que haya conocido del asunto, dentro de los quince días hábiles siguientes a la fecha de la notificación. La Unidad de Enlace deberá remitirlo al Instituto dentro de los tres días siguientes a su recepción.

ARTICULO 37.- El recurso procederá cuando:

- I. El sujeto obligado no entregue los datos personales solicitados, o lo haya hecho en formato incomprensible;
- II. El sujeto obligado se niegue a modificar, corregir o suprimir los datos personales o lo haga en términos distintos a los solicitados;
- III. El solicitante no esté conforme con el tiempo, el costo o la modalidad de entrega;
- IV. El solicitante considere que la información entregada es incompleta o no corresponda a la información requerida en la solicitud; y
- V. Habiendo operado la afirmativa ficta, haya transcurrido el término de diez días hábiles sin que se le haya proporcionado la información solicitada o modificado, corregido o suprimido los datos personales.”

Artículos de la Ley de Transparencia y Acceso a la Información Pública para el Estado de Oaxaca considerados en el desarrollo del SIGED

“ARTÍCULO 58. Cualquier persona, por si, o por medio de su representante podrá

presentar, ante la Unidad de Enlace, una solicitud de acceso a la información verbalmente, mediante escrito libre o en los formatos que apruebe la Comisión ya sea vía electrónica o personalmente. La solicitud deberá contener:

- I. El nombre y nacionalidad del solicitante así como domicilio o medio para recibir notificaciones;
- II. La descripción clara y precisa de la información que solicita, así como los datos que faciliten su búsqueda y localización; y
- III. Opcionalmente, la modalidad en la que prefiere se otorgue el acceso a la información, la cual podrá ser verbalmente siempre y cuando sea para fines de orientación, mediante consulta directa, copias simples, certificadas, correo electrónico u otro tipo de medio.

En caso de que el interesado sea persona moral se deberá comprobar además, su legal constitución y que quien formula la petición en su nombre es su legítimo representante.

Si los detalles proporcionados por el solicitante no bastan para localizar los documentos o son erróneos, la Unidad de Enlace podrá requerir, por una vez y dentro de los cinco días hábiles siguientes a la presentación de la solicitud, que indique otros elementos o corrija los datos para que en un término igual y en la misma forma, la complemente o la aclare. En caso de no cumplir con dicha prevención se tendrá por concluida la solicitud. Este requerimiento interrumpirá el plazo establecido en el Artículo 64.

ARTÍCULO 68.- El recurso de revisión regulado en esta Ley es un medio de defensa jurídica que tiene por objeto garantizar que en los actos y resoluciones de los sujetos obligados se respeten las garantías de legalidad y seguridad jurídica.

El solicitante a quien se le haya notificado la negativa de acceso a la información o la inexistencia de los documentos solicitados, podrá interponer por sí mismo o a través de su representante, el recurso de revisión ante la Comisión o ante la Unidad de Enlace que haya conocido del asunto, dentro de los quince días hábiles siguientes a la fecha de la notificación. La Unidad de Enlace deberá remitirlo a la Comisión dentro de los tres días siguientes a su recepción.”

Código de la funcionalidad “Definir dispositivos de escaneo”

```
public cuDefinirDispositivos()
{
    //buscar los dispositivos twain
    mDispositivos mdis = new mDispositivos();
    DeviceCollection dispositivos = mdis.BuscarDispositivosTwain();
    ArrayList dispWIA = mdis.ObtenerDispositivosWIA();
    if (dispositivos == null || dispositivos.Count == 0)
    {
        MessageBox.Show("No se encontraron dispositivos TWAIN instalados",
                       "Error - Definir Dispositivos de Escaneo", MessageBoxButtons.OK,
                       MessageBoxIcon.Error);
        ban = true;
    }
    else
    {
        ...
```

```

        public DeviceCollection BuscarDispositivosTwain()
        {
            DeviceManager deviceManager = new DeviceManager();
            deviceManager.Dispose();
            deviceManager.Open();
            return deviceManager.Devices;
        }

        public ArrayList ObtenerDispositivosWIA()
        {
            var devMan = new WIA.DeviceManager();
            ArrayList disp = new ArrayList();
            for (int i = 1; i <= devMan.DeviceInfos.Count; i++)
            {
                if (devMan.DeviceInfos[i].Type.Equals(WIA.WiaDeviceType.ScannerDeviceType))
                {
                    disp.Add(devMan.DeviceInfos[i]);
                }
            }
            return disp;
        }

    }

    else
    {
        ban = false;
        //obtener catalogo de funciones
        mCatalogos mcat = new mCatalogos();
        dtFunciones = mcat.ObtenerFuncionesDispositivos();
        numFun = dtFunciones.Rows.Count;
        //obtener equipo
        mUtilerias mutil = new mUtilerias();
        cEquipo cequipo = mutil.CargarEquipo();
        if (Sesion.VerificarError())
            return;

        //obtener las definiciones de cada dispositivo encontrado
        definiciones = mdis.ObtenerDefiniciones(dispositivos, cequipo);
        //buscar los dispositivos wia
        definiciones = mdis.ObtenerDefinicionesWIA(dispWia, definiciones, cequipo);
        //buscar los dispositivos isis

        //si definiciones está vacía no hay dispositivos encendidos
        if (definiciones.Count == 0)
        {
            MessageBox.Show("No se encontraron dispositivos instalados",
                           "Error - Definir Dispositivos de Escaneo", MessageBoxButtons.OK,
                           MessageBoxIcon.Error);
            ban = true;
        }
    }
    InitializeComponent();
}

//Devuelve la lista de funciones de dispositivos
public DataTable ObtenerFuncionesDispositivos()
{
    dCatalogos dcat = new dCatalogos();
    return dcat.ObtenerFuncionesDispositivos();
}

```

```

public DataTable ObtenerFuncionesDispositivos()
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable lista = new DataTable();
    if (cnx != null)
    {
        try
        {
            cnx.Open();
            string sql = "select \"id_funcion\", trim(funcion) funcion from catfuncion_dispositivo where estatus=1";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            lista = ds.Tables[0];
        }
        catch (NpgsqlException e)
        {
            Sesion.BanError = true;
            Sesion.DError = "Error al obtener las funciones de dispositivos: " + e.ToString();
            lista = null;
        }
        finally
        {
            cnx.Close();
        }
    }
    else
    {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
        lista = null;
    }
    return lista;
}

public cEquipo CargarEquipo()
{
    cEquipo cequipo = new cEquipo();

    //obtener macs
    ArrayList dirMacs = ObtenerMacs();
    //obtener equipo
    dCatalogos dcat = new dCatalogos();
    cequipo = dcat.ObtenerEquipo(dirMacs);
    return cequipo;
}

```

```

public ArrayList ObtenerMacs()
{
    int i = 0;
    ArrayList DireccionesMAC = new ArrayList();
    NetworkInterface[] interfaces = null;
    interfaces = NetworkInterface.GetAllNetworkInterfaces();
    if (interfaces != null & interfaces.Length > 0)
    {
        foreach (NetworkInterface adaptador in interfaces)
        {
            if (adaptador.NetworkInterfaceType == NetworkInterfaceType.Ethernet || adaptador.NetworkInterfaceType == NetworkInterfaceType.Wireless80211)
            {
                PhysicalAddress direccion = adaptador.GetPhysicalAddress();
                byte[] bytes = direccion.GetAddressBytes();
                string mac_address = string.Empty;
                for (i = 0; i < bytes.Length; i++)
                {
                    mac_address += bytes[i].ToString("X2");
                    if (i != bytes.Length - 1)
                    {
                        mac_address += "-";
                    }
                }
                DireccionesMAC.Add(mac_address);
            }
        }
    }
    return DireccionesMAC;
}

public cEquipo ObtenerEquipo(ArrayList dirMacs)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable lista = new DataTable();
    cEquipo equipo = new cEquipo();
    string sListaMacs = "";
    foreach (string mac in dirMacs)
    {
        if (sListaMacs.Equals(""))
            sListaMacs = ":" + mac + "";
        else
            sListaMacs += "," + ":" + mac + "";
    }
    if (cnx != null)
    {
        try
        {
            cnx.Open();
            string sql = "select distinct \"id_equipo\" from macs where mac in (" + sListaMacs + ")";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            lista = ds.Tables[0];
            if (lista.Rows.Count > 0)
            {
                DataRow dr = lista.Rows[0];
                equipo.Id = Convert.ToInt32(dr["id_equipo"].ToString());
                equipo.Macs = dirMacs;
                equipo.SListaMacs = sListaMacs;
            }
            else //guardar el equipo con sus macs en la base
            {
                ...
            }
        }
    }
}

```

```

        sql = "select nextval('macs_seq')::regclass) id";
        da = new NpgsqlDataAdapter(sql, cnx);
        ds.Reset();
        da.Fill(ds);
        lista = ds.Tables[0];
        DataRow dr = lista.Rows[0];
        equipo.Id = Convert.ToInt32(dr["id"].ToString());
        foreach (string mac in dirMacs)
        {
            sql = "insert into macs values (" + equipo.Id.ToString() + "," + mac + ")";
            NpgsqlCommand query = new NpgsqlCommand(sql, cnx);
            query.ExecuteNonQuery();
        }
        equipo.Macs = dirMacs;
        equipo.SListaMacs = sListaMacs;
    }
    } catch (NpgsqlException e)
    {
        Sesion.BanError = true;
        Sesion.DError = "Error al obtener el equipo: " + e.ToString();
        equipo = null;
    } finally {
        cnx.Close();
    }
}
else
{
    Sesion.BanError = true;
    Sesion.DError = "No se pudo conectar a la base de datos";
    equipo = null;
}
return equipo;
}

public ArrayList ObtenerDefiniciones(DeviceCollection dispositivos, cEquipo equipo)
{
    ArrayList lista = new ArrayList();
    dDispositivos ddis = new dDispositivos();
    foreach (Device disp in dispositivos)
    {
        DeviceInfo dinf = disp.Info;
        if (!dinf.IsWIA)
        {
            cDispositivo cdis = ddis.ObtenerDefinicion(dinf.ProductName.ToUpper(), equipo);
            lista.Add(cdis);
        }
    }
    return lista;
}

public cDispositivo ObtenerDefinicion(string nombreDisp, cEquipo equipo)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    cDispositivo cdis = new cDispositivo();
    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select \"id_dispositivo\", nombre, funcion, estatus from dispositivo where nombre = '" + nombreDisp + "' + "
                "and \"id_equipo\"=" + equipo.Id.ToString();
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            da.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];
            cdis.Limpiar();
            cdis.Nombre = nombreDisp;
            cdis.Equipo = equipo;
            if (dt.Rows.Count > 0)
                DataRow dr = dt.Rows[0];
                cdis.IdDispositivo = Convert.ToInt32(dr["id_dispositivo"]);
                cdis.Funcion = Convert.ToInt32(dr["funcion"]);
                cdis.Estatus = Convert.ToInt32(dr["estatus"]);
        }
    }
}

```

```

        } catch (NpgsqlException e) {
            Sesion.BanError = true;
            Sesion.DError = "Error al obtener la definición del dispositivo: " + e.ToString();
            return null;
        } finally {
            cnx.Close();
        }
    } else {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
        return null;
    }
    return cdis;
}

public ArrayList ObtenerDefinicionesWIA(ArrayList dispWia, ArrayList defs, cEquipo equipo)
{
    //ArrayList defs = new ArrayList();
    dDispositivos ddis = new dDispositivos();

    foreach (WIA.DeviceInfo disp in dispWia)
    {
        cDispositivo cdis = ddis.ObtenerDefinicion(((string)disp.Properties["Name"].get_Value()).ToUpper(), equipo);
        defs.Add(cdis);
    }

    return defs;
}

public void GuardarDefiniciones(ArrayList defsSel)
{
    dDispositivos ddis = new dDispositivos();
    ddis.GuardarDefiniciones(defsSel);
}

public void GuardarDefiniciones(ArrayList defsSel)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();

    DataSet ds = new DataSet();
    DataTable dt = new DataTable();

    cDispositivo cdis = new cDispositivo();
    NpgsqlTransaction tran = null;
    string paso = "";

    if (cnx != null)
    {
        try
        {
            cnx.Open();
            tran = cnx.BeginTransaction();
            //desactivar dispositivos actuales
            cdis = (cDispositivo)defsSel[0];
            string sql = "update dispositivo set estatus=0 where \"id_equipo\\"" + cdis.Equipo.Id.ToString();
            NpgsqlCommand query = new NpgsqlCommand(sql, cnx);
            query.Transaction = tran;
            paso = "Desactivar dispositivos actuales";
            query.ExecuteNonQuery();

            //para cada definicion:
            for (int i = 0; i < defsSel.Count; i++)
        }
    }
}

```

```

//para cada definicion:
for (int i = 0; i < defsSel.Count; i++)
{
    cdis = (cDispositivo)defsSel[i];
    string sqlAx = "select count(*) c from dispositivo where nombre='" + cdis.Nombre + "' and \\"id_equipo\\"=" + cdis.Equipo.Id.ToString();
    NpgsqlDataAdapter da = new NpgsqlDataAdapter(sqlAx, cnx);
    ds.Reset();
    da.Fill(ds);
    dt = ds.Tables[0];
    DataRow dr = dt.Rows[0];

    if (Convert.ToInt32(dr["c"].ToString()) > 0)
    {
        //si existe registro, actualizar estatus a 1
        sql = "update dispositivo set estatus = 1, funcion=" + cdis.Funcion.ToString() + " where nombre='" + cdis.Nombre
              + "' and \\"id_equipo\\"=" + cdis.Equipo.Id.ToString();
        paso = "Actualizar estatus de definición";
    }
    else
    {
        //si no existe registros, insertar con estatus 1
        sql = "insert into dispositivo values (nextval('dispositivo_seq)::regclass,'" + cdis.Nombre + "','" + cdis.Funcion.ToString()
              + "',1," + cdis.Equipo.Id.ToString() + ")";
        paso = "Insertar definición";
    }
    query.CommandText = sql;
    query.ExecuteNonQuery();
}

tran.Commit();
}
catch (NpgsqlException e)
{
    tran.Rollback();
    Sesion.BanError = true;
    Sesion.DError = "Error al guardar las definiciones de dispositivos (" + paso + "): " + e.ToString();
}
finally
{
    cnx.Close();
}
else
{
    Sesion.BanError = true;
    Sesion.DError = "No se pudo conectar a la base de datos";
}
}

```

Código de la funcionalidad “Escanear Expediente”

```
private void cuEscanExp01NoReg_Load(object sender, EventArgs e)
{
    //obtener los totales del dia
    mExpediente mexp = new mExpediente();
    EENR_txtEstExp.Text = mexp.AcumuladoExpedientesDelDia().ToString();
    EENR_txtEstHojas.Text = mexp.AcumuladoHojasDelDia().ToString();
    EENR_txtDelegNR.Focus();
}

public int AcumuladoExpedientesDelDia()
{
    dExpediente dexp = new dExpediente();
    return dexp.ObtenerAcumuladoExpedientesDelDiaDeUsuario(Sesion.Usuario.IdUsuario);
}

public int AcumuladoHojasDelDia()
{
    dExpediente dexp = new dExpediente();
    return dexp.ObtenerAcumuladoHojasDelDiaDeUsuario(Sesion.Usuario.IdUsuario);
}

public int ObtenerAcumuladoExpedientesDelDiaDeUsuario(string usuario)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();

    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    int total = 0;

    if (cnx != null)
    {
        try
        {
            cnx.Open();
            string sql = @"select count(*) c from expediente where estatus=2 and id_expediente in
                           (select distinct(id_documento) from bitacora where documento= 2 and fecha=current_date
                           and id_usuario = '"+usuario+"')";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];

            //cexp.Limpiar();
            if (dt.Rows.Count > 0)
            {
                DataRow dr = dt.Rows[0];
                total = Convert.ToInt32(dr["c"]);
            }
        }
        catch (NpgsqlException e)
        {
            Sesion.BanError = true;
            Sesion.DError = "Error al obtener los datos del expediente: " + e.ToString();
        }
        finally
        {
            cnx.Close();
        }
    }
    else
    {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
    }
}

return total;
}
```

```

public int ObtenerAcumuladoHojasDelDiaDeUsuario(string usuario)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();

    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    int total = 0;

    if (cnx != null)
    {
        try
        {
            cnx.Open();
            string sql = @"select sum(hojas) s from expediente where estatus=2 and id_expediente in
                            (select distinct(id_documento) from bitacora where documento= 2 and fecha=current_date
                            and id_usuario = '"+usuario+"')";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];

            //cexp.Limpiar();
            if (dt.Rows.Count > 0)
            {
                DataRow dr = dt.Rows[0];
                string ssum = dr["s"].ToString();
                if (!ssum.Equals(""))
                    total = Convert.ToInt32(dr["s"]);
            }
        }
        catch (NpgsqlException e)
        {
            Sesion.BanError = true;
            Sesion.DError = "Error al obtener los datos del expediente: " + e.ToString();
        }
        finally
        {
            cnx.Close();
        }
    }
    else
    {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
    }
}

return total;
}

private void EENR_txtDelegNR_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!mValidaciones.ValidaNumeros(e.KeyChar.ToString()))
        e.Handled = false;
    else
        e.Handled = true;
}

private void EENR_txtDelegNR_Leave(object sender, EventArgs e)
{
    if (EENR_txtDelegNR.Text != "")
    {
        mUtilerias util = new mUtilerias();
        if (Convert.ToInt32(EENR_txtDelegNR.Text) > 0)
            EENR_txtDelegNR.Text = util.AutoRellena('0', EENR_txtDelegNR.Text,
                                                EENR_txtDelegNR.MaxLength, 1);
        else
            EENR_txtDelegNR.Text = "";
    }
}

private void EENR_txtConsecutivoNR_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!mValidaciones.ValidaNumeros(e.KeyChar.ToString()))
        e.Handled = false;
    else
        e.Handled = true;
}

```

```

private void EENR_txtAñoNR_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!e.KeyChar.Equals('\b'))
    {
        if (mValidaciones.ValidaNumeros(e.KeyChar.ToString()))
        {
            if (mValidaciones.ValidaAñoParcial(((TextBox)sender).Text + e.KeyChar.ToString()))
                e.Handled = false;
            else
                e.Handled = true;
        }
        else
            e.Handled = true;
    }
    else
        e.Handled = false;
}

private void EENR_txtAñoNR_Leave(object sender, EventArgs e)
{
    //valida año completo
    if (EENR_txtAñoNR.Text.Length < 4 && EENR_txtAñoNR.Text.Length > 0)
    {
        MessageBox.Show("El año no tiene 4 dígitos", "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Error);
        EENR_txtAñoNR.Focus();
    }
}

private void EENR_btnAceptar_Click(object sender, EventArgs e)
{
    //valida vacíos
    if (ValidaVacíos())
    {
        string noReg = "";
        string anio = "";
        if (EENR_txtDelegNR.Text.Equals(""))
            noReg = EENR_txtConsecutivoNR.Text + "/" + EENR_txtAñoNR.Text.Substring(2,2);
        else
            noReg = EENR_txtDelegNR.Text + "/" + EENR_txtConsecutivoNR.Text + "/" + EENR_txtAñoNR.Text.Substring(2,2);
        anio = EENR_txtAñoNR.Text;
        //chechar si el expediente existe en la bd
        mExpediente mexp = new mExpediente();
        if (!mexp.Existe(noReg, anio))
        {
            MessageBox.Show("El Expediente NO está registrado", "Escanear Expediente Físico",
                           MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        if (Sesion.VerificarError())
            return;

        //obtener el núm de expedientes encontrados
        int noregis = mexp.ObtenerNoRegistrosExpediente(noReg, anio);
        if (Sesion.VerificarError())
            return;

        if (noregis == 1)
        {

```

```

        if (noregis == 1)
        {
            //cargar el contenedor con el expediente, obtener los datos del expediente
            cExpediente cexp = mexp.ObtenerExpediente(noReg, anio);
            if (Sesion.VerificarError())
                return;

            //verificar que el estatus no sea ocupado o digitalizado
            if (cexp.Estatus != 0)
            {
                string mensaje = "";
                if (cexp.Estatus == 1)
                    mensaje = "El Expediente está siendo utilizado por otro proceso";
                else if (cexp.Estatus == 2)
                    mensaje = "El Expediente ya está digitalizado";
                else if (cexp.Estatus < 0)
                    mensaje = "El Expediente NO está registrado";

                MessageBox.Show(mensaje, "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Error);
                EENR_txtConsecutivoNR.Focus();
                return;
            }

            //ir al siguiente control cuEscanExp02Preparar
            Maestro.Siguiente = new cuEscanExp02ADatos();
            datos.Add("expediente", cexp);
            datos.Add("pantalla", "1");
            Maestro.Datos = datos;
        }
        else
        {
            ...
        }

        else
        {
            //guardar noreg y año
            datos.Add("noreg", noReg);
            datos.Add("anio", anio);
            datos.Add("pantalla", "2");
            //ir a lista de expedientes encontrados
            Maestro.Siguiente = new cuEscanExp020ElegirExp();
            //datos.Add("expediente", cexp);
            Maestro.Datos = datos;
        }

        fPrincipal.CambiaControl();
    }
}

private bool ValidaVacios()
{
    string error = "";
    bool retorno = true;
    if (EENR_txtConsecutivoNR.Text == "")
    {
        EENR_txtConsecutivoNR.Focus();
        error = "El campo Consecutivo del Número de Registro está vacío";
        retorno = false;
    }
    else if (EENR_txtAñoNR.Text == "")
    {
        EENR_txtAñoNR.Focus();
        error = "El campo Año del Número de Registro está vacío";
        retorno = false;
    }

    if (!retorno)
    {
        MessageBox.Show(error, "Escanear Expediente Físico",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return retorno;
}

```

```

        public bool Existe(string numRegistro, string year)
        {
            dExpediente dexp = new dExpediente();
            return dexp.Existe(numRegistro, year);
        }

        public bool Existe(string numRegistro, string year)
        {
            // Devuelve true si el expediente existe, false si no

            Conexion con = new Conexion();
            NpgsqlConnection cnx = con.Conectar();

            DataSet ds = new DataSet();
            DataTable dt = new DataTable();

            if (cnx != null)
            {
                try
                {
                    cnx.Open();
                    string sql = "select count(*) c from expediente where \"num_Registro\\"" + numRegistro +
                                " and \"anio_Expediente\\"" + year;
                    NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
                    ds.Reset();
                    da.Fill(ds);
                    dt = ds.Tables[0];
                    DataRow dr = dt.Rows[0];

                    if (Convert.ToInt32(dr["c"].ToString()) > 0)
                        return true;
                    else
                        return false;
                }
                catch (NpgsqlException e)
                {
                    catch (NpgsqlException e)
                    {
                        Sesion.BanError = true;
                        Sesion.DError = "Error al buscar si el expediente existe por número de registro: " + e.ToString();
                        return false;
                    }
                    finally
                    {
                        cnx.Close();
                    }
                }
                else
                {
                    Sesion.BanError = true;
                    Sesion.DError = "No se pudo conectar a la base de datos";
                    return false;
                }
            }
        }

        public int ObtenerNoRegistrosExpediente(string numRegistro, string year)
        {
            dExpediente dexp = new dExpediente();
            return dexp.ObtenerNoRegistrosExpediente(numRegistro, year);
        }
    }
}

```

```

public int ObtenerNoRegistrosExpediente(string numRegistro, string year)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    int registros = 0;
    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select count(*) c from expediente where \"num_Registro\\"" + numRegistro +
                         " and \"anio_Expediente\\"" + year;
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];
            DataRow dr = dt.Rows[0];

            registros = Convert.ToInt32(dr["c"].ToString());
        } catch (NpgsqlException e) {
            Sesion.BanError = true;
            Sesion.DError = "Error al buscar si el expediente existe por número de registro: " + e.ToString();
        } finally {
            cnx.Close();
        }
    } else {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
    }
    return registros;
}

```

```

public cExpediente ObtenerExpediente(string noReg, string year)
{
    dExpediente dexp = new dExpediente();
    return dexp.ObtenerExpediente(noReg, year);
}

public cExpediente ObtenerExpediente(string noReg, string year)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    cExpediente cexp = new cExpediente();

    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select e.\"id_expediente\", trim(e.\"num_Registro\") numRegistro, " +
                         "trim(e.\"cta_predial\") cuentaPredial, trim(e.\"nombre_contrib\") nombreContrib, " +
                         "trim(e.\"paterno_contrib\") paternoContrib, trim(e.\"materno_contrib\") maternoContrib, " +
                         "trim(e.\"clave_catastral\") claveCatastral, e.\"anio_Expediente\" anio, " +
                         "trim(e.\"num_Tramite\") numTramite, trim(e.\"cta_predial_anterior\") cuentaAnterior, " +
                         "e.\"id_movimiento\", trim(m.concepto) movimiento, " +
                         "t.\"id_tramite\", trim(t.\"Tipo_Tramite\") tramite, " +
                         "trim(e.\"caja\") caja, trim(e.\"Observaciones\") observaciones, e.\"fecha_captura\" fechaCaptura, " +
                         "e.\"id_expediente\", e.\"estatus\", e.\"servidor\", " +
                         "(select s.\"ip\" from catservidor s where s.\"id_ubicacion\"=e.\"servidor\") ip, " +
                         "(select s.\"puerto\" from catservidor s where s.\"id_ubicacion\"=e.\"servidor\") puerto, " +
                         "(select s.\"usuario\" from catservidor s where s.\"id_ubicacion\"=e.\"servidor\") usuario, " +
                         "(select s.\"pass\" from catservidor s where s.\"id_ubicacion\"=e.\"servidor\") pass, " +
                         "trim(e.\"llave\") llave, trim(e.\"num_expediente\") num_expediente, trim(e.\"num_reg_rep\") num_reg_rep, tipo " +
                         "from expediente e, movimiento m, trámite t where e.\"num_Registro\\"" + noReg + " and " +
                         "m.\"Id_Movimiento\\"" + e.\"id_movimiento\" and t.\"id_tramite\\"" + m.\"id_tramite\" and \"anio_Expediente\\"" + year +
                         "+ and e.estatus != 3";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);

```

```

NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
ds.Reset();
da.Fill(ds);
dt = ds.Tables[0];

cexp.Limpiar();
if (dt.Rows.Count > 0) {
    DataRow dr = dt.Rows[0];
    cexp.IdExpediente = Convert.ToInt32(dr["id_expediente"]);
    cexp.NumRegistro = dr["numRegistro"].ToString();
    cexp.CuentaPredial = dr["cuentaPredial"].ToString();
    cexp.NombreContrib = dr["nombreContrib"].ToString();
    cexp.PaternoContrib = dr["paternoContrib"].ToString();
    cexp.MaternoContrib = dr["maternoContrib"].ToString();
    cexp.ClaveCatastral = dr["claveCatastral"].ToString();
    cexp.Año = Convert.ToInt32(dr["anio"]);
    cexp.NumTramite = dr["numTramite"].ToString();
    cexp.CuentaAnterior = dr["cuentaAnterior"].ToString();
    cCatalogo mov = new cCatalogo();
    mov.IdCatalogo = Convert.ToInt32(dr["id_movimiento"]);
    mov.DescripcionCat = dr["movimiento"].ToString();
    cexp.Movimiento = mov;
    cCatalogo tram = new cCatalogo();
    tram.IdCatalogo = Convert.ToInt32(dr["id_tramite"]);
    tram.DescripcionCat = dr["tramite"].ToString();
    cexp.Tramite = tram;
    cexp.Caja = dr["caja"].ToString();
    cexp.Observaciones = dr["observaciones"].ToString();
    cexp.FechaCaptura = dr["fechaCaptura"].ToString();
    cexp.Estatus = Convert.ToInt32(dr["estatus"]);
    cServidor serv = new cServidor();
    if (dr["servidor"] != null && !dr["servidor"].ToString().Equals("")) {
        serv.Id = Convert.ToInt32(dr["servidor"].ToString());
        serv.Ip = dr["ip"].ToString();
        serv.Puerto = dr["puerto"].ToString();
        serv.Usuario = dr["usuario"].ToString();
        serv.Pass = dr["pass"].ToString();
    }
    cexp.Servidor = serv;
    cexp.Llave = dr["llave"].ToString();
    cexp.NumExpediente = dr["num_expediente"].ToString();
    cexp.NumRegRep = dr["num_reg_rep"].ToString();
    cexp.Tipo = Convert.ToInt32(dr["tipo"]);
}
} catch (NpgsqlException e) {
    Session.BanError = true;
    Session.DError = "Error al obtener los datos del expediente: " + e.ToString();
    cexp.Limpiar();
} finally {
    cnx.Close();
}
} else {
    Session.BanError = true;
    Session.DError = "No se pudo conectar a la base de datos";
    cexp.Limpiar();
}

return cexp;
}

```

```

private void cuEscanExp02ADatos_Load(object sender, EventArgs e)
{
    datos = (Hashtable)Maestro.Datos;
    cExpediente cexp = (cExpediente)datos["expediente"];
    EEDE_txtNumExpediente.Text = cexp.NumExpediente;
    EEDE_txtNumRegistro.Text = cexp.NumRegistro;
    EEDE_txtAño.Text = cexp.Año.ToString();
    EEDE_txtCuentaPredial.Text = cexp.CuentaPredial;
    EEDE_txtCuentaAnterior.Text = cexp.CuentaAnterior;
    EEDE_txtContribuyente.Text = cexp.NombreCompletoPorApe();
    mUtilerias mutil = new mUtilerias();
    EEDE_txtClaveCatastral.Text = mutil.FormateaClaveCatastral(cexp.ClaveCatastral);
    EEDE_txtNumTramite.Text = cexp.NumTramite;
    EEDE_txtTramite.Text = cexp.Tramite.IdCatalogo.ToString() + " - " + cexp.Tramite.DescripcionCat;
    EEDE_txtMovimiento.Text = cexp.Movimiento.IdCatalogo.ToString() + " - " + cexp.Movimiento.DescripcionCat;
    EEDE_txtTipo.Text = (cexp.Tipo==1)? "PARCIAL":(cexp.Tipo==2)? "TOTAL": "";
    EEDE_txtCaja.Text = cexp.Caja;
    EEDE_txtObservaciones.Text = cexp.Observaciones;
    EEDE_btnContinuar.Focus();
}

public string FormateaClaveCatastral(string clavecat)
{
    if (clavecat.Equals("") || clavecat.Length < 24)
        return clavecat;

    string formato = "";

    formato = clavecat.Substring(0, 3) + "-";
    formato += clavecat.Substring(3, 3) + "-";
    formato += clavecat.Substring(6, 2) + "-";
    formato += clavecat.Substring(8, 3) + "-";
    formato += clavecat.Substring(11, 3) + "-";
    formato += clavecat.Substring(14, 3) + "-";
    formato += clavecat.Substring(17, 3) + "-";
    formato += clavecat.Substring(20, 4);

    return formato;
}

private void EEDE_btnContinuar_Click(object sender, EventArgs e)
{
    cExpediente cexp = (cExpediente)datos["expediente"];
    cexp.Observaciones = EEDE_txtObservaciones.Text;
    datos["expediente"] = cexp;
    Maestro.Siguiente = new cuEscanExp02Preparar();
    fPrincipal.CambiaControl();
}

private void EEDE_btnCancelar_Click(object sender, EventArgs e)
{
    int pantalla = Convert.ToInt32(datos["pantalla"].ToString());
    if (pantalla == 1)
    {
        Maestro.Datos = null;
        Maestro.Siguiente = new cuEscanExp01NoReg();
    }
    else
    {
        datos.Remove("listaExpedientes");
        datos.Remove("expediente");
        Maestro.Siguiente = new cuEscanExp020ElegirExp();
    }
    fPrincipal.CambiaControl();
}

```

```

private void cuEscanExp020ElegirExp_Load(object sender, EventArgs e)
{
    datos = (Hashtable)Maestro.Datos;
    string noReg = datos["noreg"].ToString();
    string anio = datos["anio"].ToString();

    mExpediente mexp = new mExpediente();
    datos.Add("listaExpedientes", mexp.ObtenerExpedientes(noReg, anio));

    DataTable listado = (DataTable)datos["listaExpedientes"];
    ESESE_grdListado.DataSource = listado;
    ESESE_grdListado.AutoResizeColumns();
    ESESE_btnSiguiente.Focus();
}

public DataTable ObtenerExpedientes(string noReg, string anio)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();

    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select id_expediente as \"Id\", trim(num_expediente) as \"Núm. Expediente\" ,"+ 
                        " trim(\"num_Registro\") as \"Núm. Registro\" , " +
                        " trim(paterno_contrib) as \"Ap. Paterno\" , trim(materno_contrib) as \"Ap. Materno\" , "+ 
                        " trim(nombre_contrib) as \"Nombre\" , \"anio_Expediente\" as \"Año\" , caja as \"Caja\" " +
                        " from expediente where \"num_Registro\"=''' + noReg + ''' and \"anio_Expediente\"=''' + anio;
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];
        } catch (NpgsqlException e) {
            Sesion.BanError = true;
            Sesion.DError = "Error al los expedientes coincidentes: " + e.ToString();
        } finally {
            cnx.Close();
        }
    } else {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
    }
    return dt;
}

```

```

private void EESE_btnSiguiente_Click(object sender, EventArgs e)
{
    int idsel = Convert.ToInt32(EESE_grdListado[0, EESE_grdListado.CurrentCellAddress.Y].Value.ToString());

    mExpediente mexp = new mExpediente();
    cExpediente cexp = mexp.ObtenerExpediente(idsel);
    if (Sesion.VerificarError())
        return;

    //verificar que el estatus no sea ocupado o digitalizado
    if (cexp.Estatus != 0)
    {
        string mensaje = "";
        if (cexp.Estatus == 1)
            mensaje = "El Expediente está siendo utilizado por otro proceso";
        else if (cexp.Estatus == 2)
            mensaje = "El Expediente ya está digitalizado";
        else if (cexp.Estatus < 0)
            mensaje = "El Expediente NO está registrado";

        MessageBox.Show(mensaje, "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    //ir al siguiente control cuEscanExp02Preparar
    Maestro.Siguiente = new cuEscanExp02ADatos();
    datos.Add("expediente", cexp);
    Maestro.Datos = datos;
    fPrincipal.CambiaControl();
}

public cExpediente ObtenerExpediente(int idExp)
{
    dExpediente dexp = new dExpediente();
    return dexp.ObtenerExpediente(idExp);
}

public cExpediente ObtenerExpediente(int idExp)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();

    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    cExpediente cexp = new cExpediente();

    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select e.\"id_expediente\", trim(e.\"num_Registro\") numRegistro, " +
                "trim(e.\"cta_predial\") cuentaPredial, trim(e.\"nombre_contrib\") nombreContrib, " +
                "trim(e.\"paterno_contrib\") paternoContrib, trim(e.\"materno_contrib\") maternoContrib, " +
                "trim(e.\"clave_catastral\") claveCatastral, e.\"anio_Expediente\" anio, " +
                "trim(e.\"num_Tramite\") numTramite, trim(e.\"cta_predial_anterior\") cuentaAnterior, " +
                "e.\"id_movimiento\", trim(m.concepto) movimiento, " +
                "t.\"id_tramite\", trim(t.\"Tipo_Tramite\") tramite, " +
                "trim(e.\"caja\") caja, trim(e.\"Observaciones\") observaciones, e.\"fecha_captura\" fechaCaptura, " +
                "e.\"id_expediente\", e.\"estatus\", e.\"servidor\", " +
                "(select s.\"ip\" from catserveridor s where s.\"id_ubicacion\"=e.\"servidor\") ip, " +
                "(select s.\"puerto\" from catserveridor s where s.\"id_ubicacion\"=e.\"servidor\") puerto, " +
                "(select s.\"usuario\" from catserveridor s where s.\"id_ubicacion\"=e.\"servidor\") usuario, " +
                "(select s.\"pass\" from catserveridor s where s.\"id_ubicacion\"=e.\"servidor\") pass, " +
                "trim(e.\"llave\") llave, trim(e.\"num_expediente\") num_expediente, trim(e.\"num_reg_rep\") num_reg_rep, tipo " +
                "from expediente e, movimiento m, trámite t where e.\"id_expediente\"=" + idExp.ToString() + " and " +
                "m.\"Id_Movimiento\"=e.\"id_movimiento\" and t.\"id_tramite\"=m.\"id_tramite\" and e.estatus != 3";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
        }
    }
}

```

```

        da.Fill(ds);
        dt = ds.Tables[0];

        cexp.Limpiar();
        if (dt.Rows.Count > 0) {
            DataRow dr = dt.Rows[0];
            cexp.IdExpediente = Convert.ToInt32(dr["id_expediente"]);
            cexp.NumRegistro = dr["numRegistro"].ToString();
            cexp.CuentaPredial = dr["cuentaPredial"].ToString();
            cexp.NombreContrib = dr["nombreContrib"].ToString();
            cexp.PaternoContrib = dr["paternoContrib"].ToString();
            cexp.MaternoContrib = dr["maternoContrib"].ToString();
            cexp.ClaveCatastral = dr["claveCatastral"].ToString();
            cexp.Año = Convert.ToInt32(dr["anio"]);
            cexp.NumTramite = dr["numTramite"].ToString();
            cexp.CuentaAnterior = dr["cuentaAnterior"].ToString();
            cCatalogo mov = new cCatalogo();
            mov.IdCatalogo = Convert.ToInt32(dr["id_movimiento"]);
            mov.DescripcionCat = dr["movimiento"].ToString();
            cexp.Movimiento = mov;
            cCatalogo tram = new cCatalogo();
            tram.IdCatalogo = Convert.ToInt32(dr["id_tramite"]);
            tram.DescripcionCat = dr["tramite"].ToString();
            cexp.Tramite = tram;
            cexp.Caja = dr["caja"].ToString();
            cexp.Observaciones = dr["observaciones"].ToString();
            cexp.FechaCaptura = dr["fechaCaptura"].ToString();
            cexp.Estatus = Convert.ToInt32(dr["estatus"]);
            cServidor serv = new cServidor();
            if (dr["servidor"] != null && !dr["servidor"].ToString().Equals(""))
        }

        {
            serv.Id = Convert.ToInt32(dr["servidor"].ToString());
            serv.Ip = dr["ip"].ToString();
            serv.Puerto = dr["puerto"].ToString();
            serv.Usuario = dr["usuario"].ToString();
            serv.Pass = dr["pass"].ToString();
        }
        cexp.Servidor = serv;
        cexp.Llave = dr["llave"].ToString();
        cexp.NumExpediente = dr["num_expediente"].ToString();
        cexp.NumRegRep = dr["num_reg_rep"].ToString();
        cexp.Tipo = Convert.ToInt32(dr["tipo"]);
    }

} catch (NpgsqlException e) {
    Sesion.BanError = true;
    Sesion.DError = "Error al obtener los datos del expediente: " + e.ToString();
    cexp.Limpiar();
} finally {
    cnx.Close();
}
} else {
    Sesion.BanError = true;
    Sesion.DError = "No se pudo conectar a la base de datos";
    cexp.Limpiar();
}
return cexp;
}

private void EESE_btnCancelar_Click(object sender, EventArgs e)
{
    Maestro.Datos = null;
    Maestro.Siguiente = new cuEscanExp01NoReg();
    fPrincipal.CambiaControl();
}

```

```

private void cuEscanExp02ADatos_Load(object sender, EventArgs e)
{
    datos = (Hashtable)Maestro.Datos;
    cExpediente cexp = (cExpediente)datos["expediente"];
    EEDE_txtNumExpediente.Text = cexp.NumExpediente;
    EEDE_txtNumRegistro.Text = cexp.NumRegistro;
    EEDE_txtAño.Text = cexp.Año.ToString();
    EEDE_txtCuentaPredial.Text = cexp.CuentaPredial;
    EEDE_txtCuentaAnterior.Text = cexp.CuentaAnterior;
    EEDE_txtContribuyente.Text = cexp.NombreCompletoPorApe();
    mUtilerias mutil = new mUtilerias();
    EEDE_txtClaveCatastral.Text = mutil.FormateaClaveCatastral(cexp.ClaveCatastral);
    EEDE_txtNumTramite.Text = cexp.NumTramite;
    EEDE_txtTramite.Text = cexp.Tramite.IdCatalogo.ToString() + " - " + cexp.Tramite.DescripcionCat;
    EEDE_txtMovimiento.Text = cexp.Movimiento.IdCatalogo.ToString() + " - " + cexp.Movimiento.DescripcionCat;
    EEDE_txtTipo.Text = (cexp.Tipo==1)? "PARCIAL":(cexp.Tipo==2)? "TOTAL": "";
    EEDE_txtCaja.Text = cexp.Caja;
    EEDE_txtObservaciones.Text = cexp.Observaciones;
    EEDE_btnContinuar.Focus();
}

public string FormateaClaveCatastral(string clavecat)
{
    if (clavecat.Equals("") || clavecat.Length < 24)
        return clavecat;

    string formato = "";
    formato = clavecat.Substring(0, 3) + "-";
    formato += clavecat.Substring(3, 3) + "-";
    formato += clavecat.Substring(6, 2) + "-";
    formato += clavecat.Substring(8, 3) + "-";
    formato += clavecat.Substring(11, 3) + "-";
    formato += clavecat.Substring(14, 3) + "-";
    formato += clavecat.Substring(17, 3) + "-";
    formato += clavecat.Substring(20, 4);

    return formato;
}

private void EEDE_btnContinuar_Click(object sender, EventArgs e)
{
    cExpediente cexp = (cExpediente)datos["expediente"];
    cexp.Observaciones = EEDE_txtObservaciones.Text;
    datos["expediente"] = cexp;
    Maestro.Siguiente = new cuEscanExp02Preparar();
    fPrincipal.CambiaControl();
}

private void EEDE_btnCancelar_Click(object sender, EventArgs e)
{
    int pantalla = Convert.ToInt32(datos["pantalla"].ToString());
    if (pantalla == 1)
    {
        Maestro.Datos = null;
        Maestro.Siguiente = new cuEscanExp01NoReg();
    }
    else
    {
        datos.Remove("listaExpedientes");
        datos.Remove("expediente");
        Maestro.Siguiente = new cuEscanExp020ElegirExp();
    }
    fPrincipal.CambiaControl();
}

```

```

private void cuEscanExp02Preparar_Load(object sender, EventArgs e)
{
    datos = (Hashtable)Maestro.Datos;
    cExpediente cexp = (cExpediente)datos["expediente"];
    mCatalogos mcat = new mCatalogos();
    cPropsEscaneo cpes = mcat.ObtenerPropiedadesEscaneo(1);
    if (Sesion.VerificarError())
        return;

    if (cpes == null)
    {
        EEPE_btnEscanear.Enabled = false;
        MessageBox.Show("No existen Propiedades de Escaneo Configuradas\nConsulte al Supervisor o al Administrador del Sistema",
            "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Information);
        Maestro.Datos = null;
        Maestro.Siguiente = new cuEscanExp01NoReg();
        fPrincipal.CambiaControl();
        return;
    }
    propiedadesEscaneo = cpes;
    datos.Add("propiedadesEscaneo", propiedadesEscaneo);
    EEPE_btnEscanear.Focus();
}

public cPropsEscaneo ObtenerPropiedadesEscaneo(int dispositivo)
{
    dCatalogos dcat = new dCatalogos();
    cPropsEscaneo cpes = dcat.ObtenerPropsEscaneo(dispositivo);
    return cpes;
}

```

```

public cPropsEscaneo ObtenerPropsEscaneo(int dispositivo)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    cPropsEscaneo cpes = new cPropsEscaneo();

    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select pe.\"id_propiedad\", trim(pe.pagina) pagina, trim(pe.color) color, pe.profundidad, " +
                " pe.\"resolucionX\", pe.\"resolucionY\", fd.funcion, pe.sensibilidad " +
                " from propiedades_escaneo pe, catfuncion_dispositivo fd where pe.dispositivo=" + dispositivo.ToString() +
                " and fd.\"id_funcion\"=pe.dispositivo";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];
            cpes.Limpiar();
            if (dt.Rows.Count > 0) {
                DataRow dr = dt.Rows[0];
                cpes.IdPropiedad = Convert.ToInt32(dr["id_propiedad"]);
                cpes.Pagina = dr["pagina"].ToString();
                cpes.Color = dr["color"].ToString();
                cpes.Profundidad = Convert.ToInt32(dr["profundidad"]);
                cpes.ResolucionX = Convert.ToInt32(dr["resolucionX"]);
                cpes.ResolucionY = Convert.ToInt32(dr["resolucionY"]);
                cCatalogo disp = new cCatalogo();
                disp.IdCatalogo = dispositivo;
                disp.DescripcionCat = dr["funcion"].ToString();
                cpes.Sensibilidad = Convert.ToDouble(dr["sensibilidad"].ToString());
                cpes.Dispositivo = disp;
            } else
                cpes = null;
        } catch (NpgsqlException e) {
            Sesion.BanError = true;
            Sesion.DError = "Error al obtener las funciones de dispositivos: " + e.ToString();
        } finally {
            cnx.Close();
        }
    } else {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
    }
    return cpes;
}

```

```

private void EEP_E.btnEscanear_Click(object sender, EventArgs e)
{
    cExpediente cexp = (cExpediente)datos["expediente"];
    cEscaneo cesc = new cEscaneo();
    cesc.RutaComandos = Path.Combine(Environment.CurrentDirectory, "DjVu");
    cesc.Ubicacion = @"C:\Escaneos";
    cesc.CarpetaExp = cesc.Ubicacion + @"\" + cexp.NumRegistro.Replace('/', '-');
    cesc.NomArchivoParcial = cexp.NumRegistro.Replace('/', '-') + "_";
    cesc.NomDjvuFinal = cexp.NumRegistro.Replace('/', '-') + cexp.NumRegRep + ".djvu";
    cesc.CarpetaServ = cexp.Año.ToString();
    cesc.Contador = 0;
    cesc.UltimoDjvu = 0;
    cesc.Imagenes = new ArrayList();
    cesc.Recortadas = new ArrayList();
    cesc.ResetContador = false;
    cesc.Urgente = false;

    mDispositivos mdis = new mDispositivos();

    //verificar dispositivo de escaneo
    cDispositivo cdis = mdis.ObtenerDispositivoDocumentos();
    if (Sesion.VerificarError())
        return;
    if (cdis == null)
    {
        //No está definido el dispositivo para documentos
        MessageBox.Show("El dispositivo de escaneo de documentos no está definido en el sistema",
            "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        //ir al control para definir dispositivos de escaneo
        Maestro.Siguiente = new cuDefinirDispositivos();
        Maestro.Datos = null;
        fPrincipal.CambiaControl();
        return;
    }

    if (EEPE_chkPropsTwain.Checked)
        propiedadesEscaneo.ManejadorTwain = true;
    else
        propiedadesEscaneo.ManejadorTwain = false;
    //guardar el deviceManager en cesc
    cesc.DevMng = new DeviceManager();
    //conectar dispositivo y establecer propiedades de escaneo
    cesc.Dispositivo = mdis.ConectarDispositivo(cdis, propiedadesEscaneo, cesc);
    if (Sesion.VerificarError())
        return;
    //verificar ubicacion y carpeta del expediente
}

```

```

//verificar ubicacion y carpeta del expediente
mdis.VerificarUbicacion(cesc.Ubicacion, cesc.CarpetaExp);

//cargar el hash con los datos a pasar al siguiente control
datos.Add("dispositivo", cdis);
datos.Add("escaneo", cesc);
Maestro.Datos = datos;

//verificar estatus de expediente nuevamente y cambiar a ocupado
mExpediente mexp = new mExpediente();
cexp = mexp.obtenerExpediente(cexp.IdExpediente);
if (cexp.Estatus != 0)
{
    string mensaje = "";
    if (cexp.Estatus == 1)
        mensaje = "El Expediente está siendo utilizado por otro proceso";
    else if (cexp.Estatus == 2)
        mensaje = "El Expediente ya está digitalizado";
    else if (cexp.Estatus == 3)
        mensaje = "El Expediente NO está registrado";

    MessageBox.Show(mensaje, "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Error);
    Maestro.Siguiente = new cuEscanExp01NoReg();
}
else
{
    //cambiar estatus a ocupado
    mexp.CambiaEstatus(cexp.IdExpediente, 1, cexp.Estatus);
    if (Sesion.VerificarError())
        return;
    Maestro.Siguiente = new cuEscanExp03Procesando();
}

//ir al control de procesar escaneo
fPrincipal.CambiaControl();
}

public cDispositivo ObtenerDispositivoDocumentos()
{
    mUtilerias mutil = new mUtilerias();
    cEquipo equipo = mutil.CargarEquipo();
    dDispositivos ddis = new dDispositivos();
    return ddis.ObtenerDispositivo(1, equipo);
}

public cEquipo CargarEquipo()
{
    cEquipo cequipo = new cEquipo();

    //obtener macs
    ArrayList dirMacs = ObtenerMacs();
    //obtener equipo
    dCatalogos dcat = new dCatalogos();
    cequipo = dcat.ObtenerEquipo(dirMacs);
    return cequipo;
}

```

```

public cEquipo ObtenerEquipo(ArrayList dirMacs)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable lista = new DataTable();
    cEquipo equipo = new cEquipo();

    string sListaMacs = "";
    foreach (string mac in dirMacs) {
        if (sListaMacs.Equals(""))
            sListaMacs = "" + mac + "";
        else
            sListaMacs += "," + "" + mac + "";
    }

    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select distinct \"id_equipo\" from macs where mac in (" + sListaMacs + ")";
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            lista = ds.Tables[0];

            if (lista.Rows.Count > 0) {
                DataRow dr = lista.Rows[0];
                equipo.Id = Convert.ToInt32(dr["id_equipo"].ToString());
                equipo.Macs = dirMacs;
                equipo.SListaMacs = sListaMacs;
            } else {
                //obtener el id de equipo
                sql = "select nextval('macs_seq')::regclass id";
                da = new NpgsqlDataAdapter(sql, cnx);
                ds.Reset();
                da.Fill(ds);
                lista = ds.Tables[0];
                DataRow dr = lista.Rows[0];
                equipo.Id = Convert.ToInt32(dr["id"].ToString());
                //para cada mac insertar
                foreach (string mac in dirMacs) {
                    sql = "insert into macs values (" + equipo.Id.ToString() + "," + mac + ")";
                    NpgsqlCommand query = new NpgsqlCommand(sql, cnx);
                    query.ExecuteNonQuery();
                }
                Sesion.BanError = true;
                Sesion.DError = "Error al obtener el equipo: " + e.ToString();
                equipo = null;
            } finally {
                cnx.Close();
            }
        } else {
            Sesion.BanError = true;
            Sesion.DError = "No se pudo conectar a la base de datos";
            equipo = null;
        }
        return equipo;
    }
}

```

```

public cDispositivo ObtenerDispositivo(int funcion, cEquipo equipo)
{
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    cDispositivo cdis = new cDispositivo();

    if (cnx != null) {
        try {
            cnx.Open();
            string sql = "select \"id_dispositivo\", trim(nombre) nombre from dispositivo where funcion=" + funcion.ToString() +
                " and estatus=1 and \"id_equipo\"=" + equipo.Id.ToString();
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);
            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];
            cdis.Limpiar();

            if (dt.Rows.Count > 0) {
                DataRow dr = dt.Rows[0];
                cdis.IdDispositivo = Convert.ToInt32(dr["id_dispositivo"]);
                cdis.Nombre = dr["nombre"].ToString();
                cdis.Fucion = 1;
                cdis.Estatus = 1;
                cdis.Equipo = equipo;
            } else
                cdis = null;
        } catch (NpgsqlException e) {
            Sesion.BanError = true;
            Sesion.DError = "Error al obtener el dispositivo de documentos: " + e.ToString();
            return null;
        } finally {
            cnx.Close();
        }
    } else {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
        return null;
    }
    return cdis;
}

public Device ConectarDispositivo(cDispositivo cdis, cPropsEscaneo propiedadesEscaneo, cEscaneo cesc)
{
    DeviceManager deviceManager = cesc.DevMng;
    Device disp = null;
    deviceManager.Dispose();
    deviceManager.Open();
    DeviceInfo deviceInfo;
    for (int i = 0; i < deviceManager.Devices.Count; i++)
    {
        deviceInfo = deviceManager.Devices[i].Info;
        if (deviceInfo.ProductName.ToUpper().Equals(cdis.Nombre))
        {
            disp = deviceManager.Devices[i];
            //deviceManager.Devices.Current = disp;
            //break;
        }
    }

    if (disp == null)
    {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo instanciar el dispositivo de escaneo";
        return null;
    }
}

```

```

        try
        {
            disp.Open();
        }
        catch (Exception ex)
        {
            Session.BanError = true;
            Session.DError = "No se pudo instanciar el dispositivo de escaneo";
            return null;
        }

        disp = ConfigurarEscaneo(disp, propiedadesEscaneo);

        return disp;
    }

    public Device ConfigurarEscaneo(Device disp, cPropsEscaneo prop)
    {
        // set acquisition parameters
        if (prop.ManejadorTwain)
            disp.ShowUI = true;
        else
            disp.ShowUI = false;

        //propiedades fijas
        disp.DisableAfterAcquire = true;
        disp.TransferMode = TransferMode.Memory;
        try
        {
            disp.FileFormat = (Vintasoft.Twain.TwainImageFormat)ImageFormat.TIFF;
        }
        catch (TwainDeviceCapabilityException ex0)
        {
            MessageBox.Show(ex0.Message, "File Format", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        //autoborderdetection
        DeviceCapability _autoBorderDetectionCap;
        bool _isAutoBorderDetectionAvailable;
        _autoBorderDetectionCap = disp.Capabilities.Find(DeviceCapabilityId.IAutomaticBorderDetection);
        if (_autoBorderDetectionCap == null)
        {
            _isAutoBorderDetectionAvailable = false;
            //return;
        }

        try
        {
            TwainValueContainerBase autoBorderDetectionCapValue = _autoBorderDetectionCap.GetValue();
            //if (autoBorderDetectionCapValue != null)
            //    autoBorderDetectionCheckBox.Checked = autoBorderDetectionCapValue.GetAsBool();

            _isAutoBorderDetectionAvailable = true;
        }
        catch (TwainDeviceCapabilityException)
        {
            _isAutoBorderDetectionAvailable = false;
        }
    }

```

```

        if (_isAutoBorderDetectionAvailable)
        {
            try
            {
                _autoBorderDetectionCap.SetValue(true);
            }
            catch (TwainDeviceCapabilityException)
            {
            }
        }

        bool valor = _autoBorderDetectionCap.GetValue().GetAsBool();

        try
        {
            disp.DocumentFeeder.DuplexEnabled = true;
        }
        catch (TwainDeviceCapabilityException)
        {
        }

        /*****propiedades variables *****/
    }

    //definir tamaño de página
    try
    {
        Array pags = (Array)Enum.GetValues(typeof(PageSize)).Cast<PageSize>();
        for (int i = 0; i < pags.Length; i++)
        {
            string p = pags.GetValue(i).ToString();
            if (p.ToUpper().Equals(prop.Pagina.ToUpper()))
            {
                //si el tamaño no es soportado automaticamente se asigna el default
                disp.PageSize = (PageSize)pags.GetValue(i);
                break;
            }
        }
    }
    catch (Exception exc)
    {
        try
        {
            disp.PageSize = PageSize.USLEGAL;
        }
        catch (Exception exc0)
        {
        }
    }
}

```

```

//definir el color (pixeltyp)
try
{
    Array colores = (Array)Enum.GetValues(typeof(PixelType)).Cast<PixelType>();
    for (int i = 0; i < colores.Length; i++)
    {
        string p = colores.GetValue(i).ToString();
        if (p.ToUpper().Equals(prop.Color.ToUpper()))
        {
            //si el color no es soportado automaticamente se asigna el default
            disp.PixelType = (PixelType)colores.GetValue(i);
            break;
        }
    }
}
catch (Exception exc)
{
    try
    {
        disp.PixelType = PixelType.Gray;
    }
    catch (Exception exc1)
    {
    }
}

try
{
    disp.BitDepth = prop.Profundidad;
}
catch (Exception exc)
{
    //disp.BitDepth = 8;
}

try
{
    disp.Resolution = new Resolution((float)prop.ResolucionX, (float)prop.ResolucionY, UnitOfMeasure.Pixels);
}
catch (Exception exc)
{
    try
    {
        disp.Resolution = new Resolution((float)300, (float)300, UnitOfMeasure.Pixels);
    }
    catch (Exception exc2)
    {
    }
}

//revision del alimentador
DeviceCapability hayPapel = disp.Capabilities.Find(DeviceCapabilityId.FeederLoaded);
DeviceCapability estatusAlimentador = disp.Capabilities.Find(DeviceCapabilityId.FeederEnabled);
bool hayPlano = disp.FlatbedPresent;
if (hayPapel != null && hayPapel.GetValue() != null)
{
    if (!hayPapel.GetValue().GetAsBool() && hayPlano)
    {
        estatusAlimentador.SetValue(false);
    }
    else
    {
        Sesion.BanError = true;
        Sesion.DError = "El dispositivo no tiene hojas";
    }
}

return disp;
}

```

```

        public void VerificarUbicacion(string ubicacion, string carpetaExp)
    {
        //se verifican las carpetas donde se guardaran temporalmente las imagenes y djvus
        DirectoryInfo dir = new DirectoryInfo(ubicacion);
        if (!dir.Exists)
        {
            dir.Create();
        }

        dir = new DirectoryInfo(carpetaExp);
        if (!dir.Exists)
        {
            dir.Create();
        }
    }

        public cExpediente bbtenerExpediente(int id_expediente)
    {
        return (new dExpediente()).obtenerExpediente(id_expediente);
    }

public cExpediente obtenerExpediente(int id_expediente)
{
    cExpediente cExp = new cExpediente();
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();
    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    if (cnx != null) {
        try {
            cnx.Open();
            string sql = @"SELECT
expediente.id_expediente,
expediente.estatus,
expediente.caja,
expediente.'Observaciones',
expediente.clave_catastral,
expediente.nombre_contrib,
expediente.cta_predial,
expediente.fecha_captura,
expediente.'anio_Expediente',
expediente.'num_Registro',
expediente.'num_Tramite',
expediente.cta_predial_anterior,
expediente.id_movimiento,
expediente.paterno_contrib,
expediente.materno_contrib,
expediente.estatus_anterior,
expediente.tipo,
expediente.num_expediente,
expediente.num_reg_rep
FROM
expediente
where id_expediente=@id_Expediente".Replace("'", "\'");
            sql = sql.Replace("@id_Expediente", id_expediente + "");

            NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, cnx);

            ds.Reset();
            da.Fill(ds);
            dt = ds.Tables[0];
            DataRow dr = dt.Rows[0];
        }
    }
}

```

```

cCatalogo cat_mov = new cCatalogo();
cat_mov.IdCatalogo = (dr["id_movimiento"] is int) ? Convert.ToInt32(dr["id_movimiento"]) : 0;
cat_mov.DescripcionCat = "MOVIMIENTOS";
cExp.Movimiento = cat_mov;
cCatalogo cat_tra = new cCatalogo();
int id_tramite = obtenerIdTramite(cExp.Movimiento.IdCatalogo);
cat_tra.IdCatalogo = (id_tramite > 0) ? id_tramite : 0;
cat_tra.DescripcionCat = "TRAMITES";
cExp.Tramite = cat_tra;
cExp.IdExpediente = Convert.ToInt32(dr["id_expediente"]);
cExp.Estatus = Convert.ToInt32(dr["estatus"]);
cExp.Caja = dr["caja"].ToString().Trim();
cExp.Observaciones = dr["Observaciones"].ToString().Trim();
cExp.ClaveCatastral = dr["clave_catastral"].ToString().Trim();
cExp.NombreContrib = dr["nombre_contrib"].ToString().Trim();
cExp.CuentaPredial = dr["cta_predial"].ToString().Trim();
cExp.Año = Convert.ToInt32(dr["anio_Expediente"]); //se obtiene del campo regisstro-año
cExp.NumRegistro = dr["num_Registro"].ToString().Trim();
cExp.NumTramite = dr["num_Tramite"].ToString().Trim();
cExp.CuentaAnterior = dr["cta_predial_anterior"].ToString().Trim();
cExp.PaternoContrib = dr["paterno_contrib"].ToString().Trim();
cExp.MaternoContrib = dr["materno_contrib"].ToString().Trim();
cExp.Estatus_anterior = (dr["estatus_anterior"] is Int16) ? Convert.ToInt32(dr["estatus_anterior"]) : 0;
cExp.Tipo = (dr["tipo"] is int) ? Convert.ToInt32(dr["tipo"]) : 0;
cExp.NumExpediente = dr["num_expediente"].ToString().Trim();
cExp.NumRegRep = dr["num_reg_rep"].ToString().Trim();
Sesion.BanError = false;
} catch (NpgsqlException e)
{
    Sesion.BanError = true;
    Sesion.DError = "Error al Obtener los metadatos del expediente: (" + id_expediente + ")" + e.ToString();
} catch (FormatException e)
{
    Sesion.BanError = true;
    Sesion.DError = "Error de conversion de datos en el sistema " + e.ToString();
} catch (Exception e)
{
    Sesion.BanError = true;
    Sesion.DError = "Excepcion no controlada" + e.ToString();
} finally {
    cnx.Close();
}
} else {
    Sesion.BanError = true;
    Sesion.DError = "No se pudo conectar a la base de datos";
}
return cExp;
}

public void CambiaEstatus(int idExpediente, int nuevo, int anterior)
{
    dExpediente dexp = new dExpediente();
    dexp.CambiaEstatus(idExpediente, nuevo, anterior);
}

public void CambiaEstatus(int idExp, int nuevo, int anterior)
{
    //actualizar estatus del expediente
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();

    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    NpgsqlTransaction tran = null;

    if (cnx != null)
    {
        try
        {
            cnx.Open();
            tran = cnx.BeginTransaction();
            string sql = "update expediente set estatus=" + nuevo.ToString() + ", \"estatus_anterior\\"" + anterior.ToString() +
                " where \"id_expediente\\"" + idExp;
            NpgsqlCommand cmd = new NpgsqlCommand(sql, cnx);
            cmd.Transaction = tran;
            cmd.ExecuteNonQuery();
            tran.Commit();
        }
        catch (Exception ex)
        {
            if (tran != null)
                tran.Rollback();
            throw ex;
        }
    }
}

```

```

        NpgsqlCommand query = new NpgsqlCommand(sql, cnx);
        query.Transaction = tran;
        query.ExecuteNonQuery();

        tran.Commit();
    }
    catch (NpgsqlException e)
    {
        tran.Rollback();
        Sesion.BanError = true;
        Sesion.DError = "Error al actualizar estatus del expediente: " + e.ToString();
    }
    finally
    {
        cnx.Close();
    }
}
else
{
    Sesion.BanError = true;
    Sesion.DError = "No se pudo conectar a la base de datos";
}
}

private void cuEscanExp03Procesando_Load(object sender, EventArgs e)
{
    fPrincipal.BloqueaMenu();
    IniciarEscaneo();
}

public void IniciarEscaneo()
{
    Hashtable datos = (Hashtable)Maestro.Datos;
    cExpediente cexp = (cExpediente)datos["expediente"];
    cEscaneo cesc = (cEscaneo)datos["escaneo"];
    cPropsEscaneo cpes = (cPropsEscaneo)datos["propiedadesEscaneo"];

    //adquirir imagenes
    mDispositivos mdis = new mDispositivos();
    mUtilerias mutil = new mUtilerias();
    mExpediente mexp = new mExpediente();
    bool continuar = false;
    cesc.Sensibilidad = cpes.Sensibilidad;

    //CON EL SDK DJVU AHORA SE USA EL OBJETO ENCODER PARA ALMACENAR LAS IMAGENES ADQUIRIDAS
    //ESTE ENCODER SE GUARDA EN EL CONTENEDOR CESCANE
    DjVu.EncoderParams encParam = new DjVu.EncoderParams("scan600");
    //encParam.PictureEncoding = DjVu.PictureEncodingMethod.JPEG;
    encParam.EncoderMode = DjVu.EncoderMode.Picture;
    DjVu.Encoder enc = new DjVu.Encoder(encParam);
    if (cesc.Encoder != null)
        enc = cesc.Encoder;
    else
        cesc.Encoder = enc;
    //--ESTE ENCODER SE GUARDA EN EL CONTENEDOR CESCANE
}

```

```

do
{
    cesc = mdis.AdquirirImagenes(cesc);
    if (Sesion.VerificarError())
    {
        //eliminar carpeta de expediente
        mutil.EliminarCarpeta(cesc.CarpetaExp);
        //regresar el estatus anterior del expediente
        mexp.RegresaEstatus(cexp.IdExpediente);
        //ir a escanear exp 01 noregistro
        Maestro.Datos = null;
        Maestro.Siguiente = new cuEscanExp01NoReg();
        continuar = false;
        fPrincipal.ActivaMenu();
    }
    else
    {
        if (cesc.Imagenes.Count == 0)
        {
            MessageBox.Show("No se adquirieron imágenes\nConsulte al Supervisor o Administrador para que\n" +
                "cambie la sensibilidad de hoja en blanco\n" +
                "O bien, coloque otro documento en el escáner",
                "Escanear Expediente Físico", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            mexp.RegresaEstatus(cexp.IdExpediente);
            Maestro.Datos = null;
            Maestro.Siguiente = new cuEscanExp01NoReg();
            continuar = false;
            fPrincipal.ActivaMenu();
        }
        else
        {
            //checa el estatus del escaneo procesado
            if (cesc.Estatus == 1)
            {
                //preguntar si finalizar
                fEscaneoFinalizado fesf = new fEscaneoFinalizado();
                fesf.ShowDialog();
                //volver a bajar los datos para obtener la respuesta
                datos = (Hashtable)Maestro.Datos;
                string resp = datos["respuesta"].ToString();
                if (resp.Equals("1"))
                {
                    //continuar el escaneo
                    cesc.Estatus = -1;
                    //reconectar el dispositivo
                    cDispositivo cdis = mdis.ObtenerDispositivoDocumentos();
                    cesc.DevMng = new DeviceManager();
                    cesc.Dispositivo = mdis.ConectarDispositivo(cdis, cpes, cesc);
                    //ir a adquirir imagenes
                    continuar = true;
                }
                else if (resp.Equals("2"))
                {
                    //finalizar escaneo cargar el siguiente control;
                    datos["escaneo"] = cesc;
                    Maestro.Datos = datos;
                    Maestro.Siguiente = new cuEscanExp04Finalizado();
                    continuar = false;
                }
            }
        }
    }
}

```

```

        else
        {
            //cancelar el escaneo
            continuar = false;
            cesc.Estatus = 3;
            mutil.EliminarCarpeta(cesc.CarpetaExp);
            //regresar el estatus anterior del expediente
            mexp.RegresaEstatus(cexp.IdExpediente);
            Maestro.Datos = null;
            Maestro.Siguiente = new cuEscanExp01NoReg();
            fPrincipal.ActivaMenu();
        }
    }
    else if (cesc.Estatus == 2)
    {
        //confirmar cancelación
        DialogResult r = MessageBox.Show("¿Está seguro que desea cancelar el escaneo?" +
            "\n (La Carpeta Temporal de Imágenes del Expediente será eliminada)",
            "Escanear Expediente Físico", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (r == DialogResult.Yes)
        {
            mutils.EliminarCarpeta(cesc.CarpetaExp);
            //regresar el estatus anterior del expediente
            mexp.RegresaEstatus(cexp.IdExpediente);
            //ir a escan exp num registro 01
            Maestro.Datos = null;
            Maestro.Siguiente = new cuEscanExp01NoReg();
            continuar = false;
            fPrincipal.ActivaMenu();
        }
        else
        {
            //continuar el escaneo
            cesc.Estatus = -1;
            //reconectar el dispositivo
            cdispositivo cdis = mdis.ObtenerDispositivoDocumentos();
            cesc.DevMng = new DeviceManager();
            cesc.Dispositivo = mdis.ConectarDispositivo(cdis, cpes, cesc);
            //ir a adquirir imagenes
            continuar = true;
        }
    }
}
} while (continuar);

//ir al siguiente control
fPrincipal.CambiaControl();

datos.Remove("escaneo");
datos.Add("escaneo", cesc);
Maestro.Datos = datos;
}

```

```

public cEscaneo AdquirirImagenes(cEscaneo cesc)
{
    try
    {
        AcquireModalState acquireModalState = AcquireModalState.None;
        AcquiredImage acquiredImage;
        mUtilerias mutil = new mUtilerias();
        mDjvu mdjvu = new mDjvu();
        do
        {
            acquireModalState = cesc.Dispositivo.AcquireModal();
            switch (acquireModalState)
            {
                case AcquireModalState.ImageAcquired:
                    cesc.Hojas++;
                    acquiredImage = cesc.Dispositivo.AcquiredImages.Last;
                    if (!acquiredImage.IsBlank((float)cesc.Sensibilidad))
                    {
                        cesc.Contador++;
                        acquiredImage.Save(Path.Combine(cesc.CarpetaExp, cesc.NomArchivoParcial
                            + cesc.Contador.ToString() + ".tif"));
                        acquiredImage.Dispose();
                        cesc.Imagenes.Add(cesc.NomArchivoPartial + cesc.Contador.ToString());
                    }
                    break;

                case AcquireModalState.ScanCompleted:
                    cesc.Estatus = 1;
                    cesc.Dispositivo.Close();
                    cesc.DevMng.Close();
                    break;
                case AcquireModalState.ScanCanceled:
                    cesc.Estatus = 2;
                    cesc.Dispositivo.Close();
                    cesc.DevMng.Close();
                    break;
                case AcquireModalState.ScanFailed:
                    Sesion.BanError = true;
                    Sesion.DError = "Ocurrió un error al escanear el expediente";
                    cesc.Estatus = 3;
                    cesc.Dispositivo.Close();
                    cesc.DevMng.Close();
                    break;
            }
        }
        while (acquireModalState != AcquireModalState.None);
    }
    catch (Exception e)
    {
        Sesion.BanError = true;
        Sesion.DError = "Ocurrió un problema en la adquisición de imágenes: " + e.ToString();
    }

    return cesc;
}

public void EliminarCarpeta(string carpeta)
{
    DirectoryInfo dir = new DirectoryInfo(carpeta);
    if (dir.Exists)
    {
        dir.Delete(true); // el true es pa q la borre con todo y su contenido
    }
}

public void RegresaEstatus(int id)
{
    dExpediente dexp = new dExpediente();
    dexp.RegresaEstatus(id);
}

```

```

public void RegresaEstatus(int id)
{
    //regresar estatus anterior
    Conexion con = new Conexion();
    NpgsqlConnection cnx = con.Conectar();

    DataSet ds = new DataSet();
    DataTable dt = new DataTable();
    NpgsqlTransaction tran = null;

    if (cnx != null)
    {
        try
        {
            cnx.Open();
            tran = cnx.BeginTransaction();
            string sql = "update expediente set estatus=\"estatus_anterior\" where \"id_expediente\\"" + id;
            NpgsqlCommand query = new NpgsqlCommand(sql, cnx);
            query.Transaction = tran;
            query.ExecuteNonQuery();

            tran.Commit();
        }

        catch (NpgsqlException e)
        {
            tran.Rollback();
            Sesion.BanError = true;
            Sesion.DError = "Error al regresar estatus anterior del expediente: " + e.ToString();
        }
        finally
        {
            cnx.Close();
        }
    }
    else
    {
        Sesion.BanError = true;
        Sesion.DError = "No se pudo conectar a la base de datos";
    }
}

private void cuEscanExp04Finalizado_Load(object sender, EventArgs e)
{
    //bajar los datos de la sesion
    datos = (Hashtable)Maestro.Datos;
    cesc = (cEscaneo)datos["escaneo"];

    //cargar la primera imagen
    imgActual = 0;
    CargarImagen();

    EEEF_btnAnterior.Enabled = false;
    EEEF_btnInicio.Enabled = false;

    if (cesc.Imagenes.Count == 1)
    {
        EEEF_btnSiguiente.Enabled = false;
        EEEF_btnFinal.Enabled = false;
        EEEF_btnInsertar.Enabled = false;
        EEEF_btnEliminar.Enabled = false;
    }
}

private void CargarImagen()
{
    sImgActual = cesc.Imagenes[imgActual].ToString() + ".tif";
    EEEF_etqNumImagen.Text = (imgActual + 1).ToString() + " de " + cesc.Imagenes.Count.ToString();
    FileStream fs = new FileStream(cesc.CarpetaExp + @"\" + sImgActual, FileMode.Open);
    EEEF_imgImagen.Image = Image.FromStream(fs);
    fs.Close();
    fs.Dispose();
}

```

```

private void EEEF_btnSiguinte_Click(object sender, EventArgs e)
{
    EEEF_imgImagen.Image.Dispose();
    imgActual++;
    CargarImagen();
    if (imgActual == cesc.Imagenes.Count - 1)
    {
        EEEF_btnSiguinte.Enabled = false;
        EEEF_btnFinal.Enabled = false;
        EEEF_btnInsertar.Enabled = false;
    }

    if (!EEEF_btnAnterior.Enabled)
    {
        EEEF_btnAnterior.Enabled = true;
        EEEF_btnInicio.Enabled = true;
    }

    //validar si esta en imagenes recortadas
    EEEF_btnRestaurar.Visible = false;
    if (cesc.Recortadas.Contains(sImgActual))
    {
        EEEF_btnRestaurar.Visible = true;
    }
}

private void EEEF_btnAnterior_Click(object sender, EventArgs e)
{
    EEEF_imgImagen.Image.Dispose();
    imgActual--;
    CargarImagen();
    if (imgActual == 0)
    {
        EEEF_btnAnterior.Enabled = false;
        EEEF_btnInicio.Enabled = false;
    }

    if (!EEEF_btnSiguinte.Enabled)
    {
        EEEF_btnSiguinte.Enabled = true;
        EEEF_btnFinal.Enabled = true;
        EEEF_btnInsertar.Enabled = true;
    }

    //validar si está en imágenes recortadas
    EEEF_btnRestaurar.Visible = false;
    if (cesc.Recortadas.Contains(sImgActual))
    {
        EEEF_btnRestaurar.Visible = true;
    }
}

private void EEEF_btnRotar_Click(object sender, EventArgs e)
{
    Image bmp = EEEF_imgImagen.Image;
    bmp.RotateFlip(RotateFlipType.Rotate270FlipXY);
    bmp.Save(cesc.CarpetaExp + @"\" + sImgActual, System.Drawing.Imaging.ImageFormat.Tiff);
    EEEF_imgImagen.Image = bmp;
}

```

```

private void EEEF_btnEliminar_Click(object sender, EventArgs e)
{
    DialogResult r = MessageBox.Show("¿Está seguro de eliminar la imagen seleccionada?",
        "Escanear Expediente Físico", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (r == DialogResult.Yes) {
        mUtilerias mutil = new mUtilerias();
        mutil.EliminaArchivo(cesc.CarpetaExp + @"\" + sImgActual);
        cesc.Imagenes.RemoveAt(imgActual);

        if (imgActual > 0)
            imgActual--;
        CargarImagen();
        if (imgActual == 0) {
            EEEF_btnAnterior.Enabled = false;
            EEEF_btnInicio.Enabled = false;
        }

        if (imgActual == cesc.Imagenes.Count) {
            EEEF_btnSiguiente.Enabled = false;
            EEEF_btnFinal.Enabled = false;
        }

        if (cesc.Imagenes.Count == 1) {
            EEEF_btnEliminar.Enabled = false;
            EEEF_btnAnterior.Enabled = false;
            EEEF_btnSiguiente.Enabled = false;
        }
    }
}

private void EEEF_btnSustituir_Click(object sender, EventArgs e)
{
    //confirmar la sustitución
    DialogResult r = MessageBox.Show("Coloque en el escáner el documento que reemplazará a la imagen actual." +
        "\nCuando esté listo haga clic en Aceptar",
        "Escanear Expediente Físico", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
    if (r == DialogResult.OK)
    {
        cesc.Sustituir = imgActual;
        mDispositivos mdis = new mDispositivos();
        //reconectar el dispositivo
        cesc.Dispositivo = mdis.ConectarDispositivo((cDispositivo)datos["dispositivo"],
            (cPropsEscaneo)datos["propiedadesEscaneo"], cesc);

        mdis.SustituirImagen(cesc);
        if (Sesion.VerificarError())
            return;

        CargarImagen();
    }
}

private void EEEF_btnAgregar_Click(object sender, EventArgs e)
{
    DialogResult r = MessageBox.Show("Coloque los documentos adicionales en el escáner y haga clic en Aceptar",
        "Escanear Expediente Físico", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
    if (r == DialogResult.OK)
    {
        //reconectar el dispositivo
        mDispositivos mdis = new mDispositivos();
        cesc.Dispositivo = mdis.ConectarDispositivo((cDispositivo)datos["dispositivo"],
            (cPropsEscaneo)datos["propiedadesEscaneo"], cesc);
        //iniciar el proceso del escaneo
        datos["escaneo"] = cesc;
        Maestro.Datos = datos;
        Maestro.Siguiente = new cuEscanExp03Procesando();
        fPrincipal.CambiaControl();
    }
}

```

```

private void EEEF_btnRotarIzq_Click(object sender, EventArgs e)
{
    Image bmp = EEEF_imgImagen.Image;
    bmp.RotateFlip(RotateFlipType.Rotate90FlipXY);
    bmp.Save(cesc.CarpetaExp + @"\" + sImgActual, System.Drawing.Imaging.ImageFormat.Tiff);
    EEEF_imgImagen.Image = bmp;
}

private void EEEF_btnAplicaRec_Click(object sender, EventArgs e)
{
    //crear una copia del archivo original, si ya existe el original no se copia
    mUtilerias mutil = new mUtilerias();
    string original = sImgActual.Substring(0, sImgActual.Length - 4) + "_OR.tif";
    if (!mutil.ExisteArchivo(cesc.CarpetaExp + @"\" + original))
    {
        mutil.CopiarArchivo(cesc.CarpetaExp + @"\" + sImgActual, cesc.CarpetaExp + @"\" + original);
    }

    byte[] imgBytes = File.ReadAllBytes(cesc.CarpetaExp + @"\" + sImgActual);

    FileStream fs = new FileStream(cesc.CarpetaExp + @"\" + sImgActual, FileMode.Open);
    Image img = Image.FromStream(fs);
    fs.Close();
    fs.Dispose();

    int anchoOrig = img.Width;
    int altoOrig = img.Height;

    Rectangle rectOrig = new Rectangle(0, 0, anchoOrig, altoOrig);

    // El rectángulo que ocupará cada nuevo trozo
    int x = 0;
    int y = 0;
    int ancho = 0;
    int alto = 0;

    //validar vacíos
    if (EEEF_nudAbajo.Text.Equals(""))
        EEEF_nudAbajo.Text = "0";
    if (EEEF_nudArriba.Text.Equals(""))
        EEEF_nudArriba.Text = "0";
    if (EEEF_nudIzquierda.Text.Equals(""))
        EEEF_nudIzquierda.Text = "0";
    if (EEEF_nudDerecha.Text.Equals(""))
        EEEF_nudDerecha.Text = "0";
    //validar ceros
    if (Convert.ToInt32(EEEF_nudIzquierda.Text) == 0 && Convert.ToInt32(EEEF_nudDerecha.Text) == 0
        && Convert.ToInt32(EEEF_nudArriba.Text) == 0 && Convert.ToInt32(EEEF_nudAbajo.Text) == 0)
    {
        EEEF_panRecorte.Visible = false;
        return;
    }

    x = Convert.ToInt32(EEEF_nudIzquierda.Text);
    y = Convert.ToInt32(EEEF_nudArriba.Text);
    ancho = anchoOrig - Convert.ToInt32(EEEF_nudDerecha.Text) - x;
    alto = altoOrig - Convert.ToInt32(EEEF_nudAbajo.Text) - y;

    byte[] img2 = CropImageFile(imgBytes, ancho, alto, x, y);

    MemoryStream ms = new MemoryStream(img2);
    img = Image.FromStream(ms);

    img.Save(cesc.CarpetaExp + @"\" + sImgActual);
    EEEF_imgImagen.Image = img;

    //agregar a imágenes recortadas
    cesc.Recortadas.Add(sImgActual);
    //mostrar el botón restaurar
    EEEF_btnRestaurar.Visible = true;
    EEEF_panRecorte.Visible = false;
}

```

```

private void EEEF_btnRestaurar_Click(object sender, EventArgs e)
{
    //reemplazar el archivo recortado por el original
    mUtilerias mutil = new mUtilerias();
    string original = sImgActual.Substring(0, sImgActual.Length - 4) + "_OR.tif";
    mutil.CopiarArchivo(cesc.CarpetaExp + @"\" + original, cesc.CarpetaExp + @"\" + sImgActual);

    FileStream fs = new FileStream(cesc.CarpetaExp + @"\" + sImgActual, FileMode.Open);
    Image img = Image.FromStream(fs);
    fs.Close();
    fs.Dispose();

    EEEF_imgImagen.Image = img;
    EEEF_btnRestaurar.Visible = false;
    //quitar de la lista de imgs recortadas
    cesc.Recortadas.Remove(sImgActual);
}

private void EEEF_btnInsertar_Click(object sender, EventArgs e)
{
    //confirmar la inserción
    DialogResult r = MessageBox.Show("Coloque en el escáner el documento que se insertará después de la imagen actual." +
        "\nCuando esté listo haga clic en Aceptar",
        "Escanear Expediente Físico", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
    if (r == DialogResult.OK)
    {
        cesc.Sustituir = imgActual;
        mDispositivos mdis = new mDispositivos();
        //reconectar el dispositivo
        cesc.Dispositivo = mdis.ConectarDispositivo((cDispositivo)datos["dispositivo"],
            (cPropsEscaneo)datos["propiedadesEscaneo"], cesc);

        cesc = mdis.InsertarImagen(cesc);
        if (Sesion.VerificarError())
            return;

        CargarImagen();
    }
}

private void EEEF_btnConfirmar_Click(object sender, EventArgs e)
{
    //validar casilla urgente
    if (EEEF_chkUrgente.Checked)
    {
        cesc.Urgente = true;
    }

    //CARGA LAS IMAGENES AL ENCODER
    for (int i = 0; i < cesc.Imagenes.Count; i++)
    {
        cesc.Encoder.AddPage(cesc.CarpetaExp + @"\" + cesc.Imagenes[i] + ".tif");
    }

    datos["escaneo"] = cesc;
    Maestro.Datos = datos;
    Maestro.Siguiente = new cuEscanExp05Guardando();
    fPrincipal.CambiaControl();
}

```

```

private void EEEF_btnCancelar_Click(object sender, EventArgs e)
{
    //confirmar la cancelación
    mUtilerias mutil = new mUtilerias();
    DialogResult r = MessageBox.Show("¿Está seguro que desea cancelar el escaneo?" +
        "\n (La Carpeta Temporal de Imágenes del Expediente será eliminada)",
        "Escanear Expediente Físico", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (r == DialogResult.Yes)
    {
        mutil.EliminarCarpeta(cesc.CarpetaExp);
        //regresar el estatus anterior del expediente
        mExpediente mexp = new mExpediente();
        cExpediente cexp = (cExpediente)datos["expediente"];
        mexp.RegresaEstatus(cexp.IdExpediente);
        //ir a escan exp num registro 01
        Maestro.Datos = null;
        Maestro.Siguiente = new cuEscanExp01NoReg();
        fPrincipal.CambiaControl();
        fPrincipal.ActivaMenu();
    }
}

private void cuEscanExp05Guardando_Load(object sender, EventArgs e)
{
    System.Threading.Thread workerThread = new System.Threading.Thread(GuardarExpedienteDigitalizado);
    workerThread.Start();
    while (!workerThread.IsAlive);

    workerThread.Join();

    datos = (Hashtable)Maestro.Datos;
    cexp = (cExpediente)datos["expediente"];

    if (cexp.Estatus != -1)
    {
        //borrar la carpeta local del expediente
        mExpediente mexp = new mExpediente();
        //imprimir etiqueta de digitalización
        DialogResult r = MessageBox.Show("¿Desea imprimir portada de digitalización?",
            "Escanear Expediente Físico", MessageBoxButtons.YesNo, MessageBoxIcon.Information);
        if (r == DialogResult.Yes)
        {
            //mexp.ImprimirBarras(cexp);
            ImprimirPortadaDigitalizacion();
            if (Sesion.VerificarError()) ;
        }

        //informar éxito de la operación
        MessageBox.Show("El Expediente ha sido digitalizado exitosamente", "Escanear Expediente Físico",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        Maestro.Datos = null;
        Maestro.Siguiente = new cuEscanExp01NoReg();
        fPrincipal.ActivaMenu();
    }
    else
    {
        MessageBox.Show("No se pudo guardar el expediente digitalizado", "Escanear Expediente Físico",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        Maestro.Siguiente = new cuEscanExp04Finalizado();
    }
}

fPrincipal.CambiaControl();
}

```

```

        public void GuardarExpedienteDigitalizado()
        {
            datos = (Hashtable)Maestro.Datos;
            cesc = (cEscaneo)datos["escaneo"];

            mExpediente mexp = new mExpediente();
            cexp = (cExpediente)datos["expediente"];

            cexp = mexp.GuardarEscaneo(cexp, cesc);

            datos["expediente"] = cexp;
            Maestro.Datos = datos;

        }

        public cExpediente GuardarEscaneo(cExpediente cexp, cEscaneo cesc)
        {
            //USO DE LA LICENCIA DEL SDK DJVU
            DjVu.LicenseManager.LicenseScript = "*****";
            cesc.Encoder.FinalizeSession();

            try
            {
                DjVu.Document doc = cesc.Encoder.GetEncodedDocument();
                doc.SaveAs(cest.CarpetaExp + @"\\" + cesc.NomDjvuFinal);
            }
            catch (DjVuException excep)
            {
                string error = excep.ToString();
            }
            //--USO DE LA LICENCIA DEL SDK DJVU

            mCatalogos mcat = new mCatalogos();
            cServidor servidorNor = mcat.ObtenerServidor();
            cServidor servidorFTP = new cServidor();
            servidorFTP = servidorNor;

            if (Sesion.VerificarError())
            {
                cexp.Estatus = -1;
                return cexp;
            }

            mUtilerias mut = new mUtilerias();
            mut.SubirArchivoFTP(servidorFTP, cesc.CarpetaServ, Path.Combine(cest.CarpetaExp, cesc.NomDjvuFinal));
            if (Sesion.VerificarError())
            {
                cexp.Estatus = -1;
                return cexp;
            }

            cexp.Estatus = 2;//2 es el estatus de digitalizado
            cexp.Servidor = servidorFTP;

            //cargar hojas y paginas
            cexp.Hojas = cesc.Hojas / 2;
            cexp.Paginas = cesc.Imagenes.Count;
        }
    }
}

```

```

        //cargar hojas y paginas
        cexp.Hojas = cesc.Hojas / 2;
        cexp.Paginas = cesc.Imagenes.Count;

        //actualizar datos en la base de datos: archivo (con ruta), llave y estatus y bitacora
        dExpediente dexp = new dExpediente();
        dexp.RegistraDigitalizacion(cexp);
        if (Sesion.VerificarError())
        {
            cexp.Estatus = -1;
            return cexp;
        }

        return cexp;
    }

    public void RegistraDigitalizacion(cExpediente cexp)
    {
        //actualizar datos de digitalizacion del expediente
        Conexion con = new Conexion();
        NpgsqlConnection cnx = con.Conectar();

        DataSet ds = new DataSet();
        DataTable dt = new DataTable();
        NpgsqlTransaction tran = null;
        string paso = "";

        if (cnx != null)
        {
            try
            {
                cnx.Open();
                tran = cnx.BeginTransaction();
                string sql = "update expediente set servidor='" + cexp.Servidor.Id.ToString()
                    + "', estatus=2, \"estatus_anterior\"=2, "
                    + "hojas=" + cexp.Hojas.ToString() + ", paginas="
                    + cexp.Paginas + ", \"Observaciones\\"" + cexp.Observaciones.Trim() + "" +
                    " where \"id_expediente\\"" + cexp.IdExpediente.ToString();
                NpgsqlCommand query = new NpgsqlCommand(sql, cnx);
                query.Transaction = tran;
                paso = "Tabla Expediente";
                query.ExecuteNonQuery();
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}

```

```

//registrar en bitácora (id_bitacora, detalle_documento, fecha, hora, documento, id_documento, usuario
sql = "insert into bitacora values (nextval('bitacora_seq)::regclass),1,current_date," +
      "current_time,2,'" + cexp.IdExpediente + "','" +
      Sesion.Usuario.IdUsuario + "')";
query.CommandText = sql;
paso = "Tabla Bitácora";
query.ExecuteNonQuery();

tran.Commit();
}
catch (NpgsqlException e)
{
    tran.Rollback();
    Sesion.BanError = true;
    Sesion.DError = "Error al actualizar datos de digitalización del expediente: (" + paso + ") " + e.ToString();
}
finally
{
    cnx.Close();
}
}
else
{
    Sesion.BanError = true;
    Sesion.DError = "No se pudo conectar a la base de datos";
}
}

private void ImprimirPortadaDigitalizacion()
{
    CargarFuente();
    try
    {
        EEEG_prtImpresion.Print();
    }
    catch (Exception e)
    {
        Sesion.BanError = true;
        Sesion.DError = "Ocurrió un problema al imprimir la portada de digitalización: "+e.ToString();
    }
}

private void CargarFuente()
{
    PrivateFontCollection MiColecciondeFuentes = new PrivateFontCollection();
    //cargamos la fuente, el archivo de preferencia que esté en la raíz del programa para que lo ubique fácilmente
    if (File.Exists(Application.StartupPath + "\\c39hrp24dhtt.ttf"))
    {
        MiColecciondeFuentes.AddFontFile(Application.StartupPath + "\\c39hrp24dhtt.ttf");
        FontFamily FamiliaDeFuentes = MiColecciondeFuentes.Families[0];
        //parámetros la familia de fuentes y el tamaño que tendrá la fuente
        MiFuente = new Font(FamiliaDeFuentes, 40);
    }
}

```