Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Caracas - Venezuela

Desarrollo de una aplicación para procesar y organizar oficios digitalizados de modificación y apertura de materias de la División de Control de Estudios de la Facultad de Ciencias de la Universidad Central de Venezuela.

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por la Bachiller
Nevily de la Cruz Aular Pérez
C.I: 17.610.938
para optar al título de
Licenciado en Computación

Tutores:

Prof. Andrés Sanoja
Prof. Héctor Navarro

Caracas, Julio / 2010.

Acta

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado titulado **"Desarrollo de una aplicación para procesar y organizar oficios digitalizados de modificación y apertura de materias de la División de Control de Estudios de la Facultad de Ciencias de la Universidad Central de Venezuela"** y presentado por la Br. Nevily de la Cruz Aular Pérez, de Cédula de Identidad 17.610.938 a los fines de optar al título de Licenciado en Computación, dejamos constancia de lo siguiente:

Leído como fue dicho trabajo, por cada uno de los miembros del jurado, se fijó el día 19 de julio de 2010, a las 11am, para que la autora lo defendiera en forma pública, lo que ella hizo en la Sala 1 de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondió a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobar.

En fe de lo cual se levanta la presente Acta, en Caracas el día 19 de julio de 2010.

_____	_____
Prof. Andrés Sanoja	Prof. Héctor Navarro
(Tutor)	(Tutor)
_____	_____
Prof. Sergio Rivas	Prof. Rhadamés Carmona
(Jurado)	(Jurado)

Agradecimientos

Gracias a Dios y a mi madre por todo el apoyo en este proceso, por ser mi mayor inspiración y por siempre darme una palabra de aliento cuando más lo he necesitado.

A mis familiares, especialmente a mis hermanas Lilibeth y Nesly y a mis adoradas sobrinas que están a mi lado día a día, animándome con sus sonrisas y contagiándome con su alegría.

A mi esposo Michel, por ser esa persona especial que me brinda su amor y apoyo a diario.

A mis amigos, que más que eso son mis hermanos, en especial Gino, Daniel, Rafael, Gabriel, Ronald, Anna y Gilberto que han sido mi apoyo durante toda la carrera.

A mis tutores y profesores por sus enseñanzas y consejos durante todo el transcurso de mis estudios universitarios.

A la Universidad Central de Venezuela por permitirme ser parte de ella y convertirme en una profesional.

Nevily Aular Pérez

Resumen

El objetivo del presente Trabajo Especial de Grado consiste en el desarrollo de un sistema que automatice el proceso de digitalización de oficios de aperturas y modificaciones de materias de la Facultad de Ciencias de la Universidad Central de Venezuela, que han sido almacenados en las instalaciones de la División de Control de Estudios, específicamente un archivo ubicado en la oficina de la directora de dicha división. El sistema está dividido en dos partes, donde se tiene una aplicación local encargada de empaquetar y enviar las imágenes al servidor, y en segundo lugar se tiene una aplicación Web encargada del procesamiento de los oficios. Para cumplir con los objetivos de la aplicación se utilizó la metodología ágil XP (Programación Extrema), la cual se basa fundamentalmente, en la construcción progresiva del software sin hacer énfasis en etapas de diseños.

Palabras Clave

Procesamiento digital de imágenes, Optical Character Recognition (OCR), aplicaciones Web, Facultad de Ciencias UCV.

Índice General

Introducción 11
Capítulo 1: Planteamiento del Problema 13
1.1 Título 14
1.2 Situación Actual 14
1.3 Planteamiento del Problema 15
1.4 Objetivos 16
1.4.1 Objetivo General 16
1.4.2 Objetivos Específicos 17
1.5 Justificación 18
1.6 Alcance 19
Capítulo 2: Marco Conceptual 22
2.1 Digitalización 23
2.1.1 Fases del Proceso de Digitalización 23
2.1.2 Procesamiento de Imágenes 24
2.1.3 Reconocimiento Óptico de Caracteres 26
2.2 Arquitectura Cliente/Servidor 29
2.2.1 Componentes 30
2.3 Protocolos de Transferencia de Datos 31
2.3.1 HTTP 31
2.3.2 FTP 32
2.4 Herramientas Tecnológicas 33
2.4.1 Herramientas Tecnológicas para Aplicación Web 33
2.4.2 Herramientas Tecnológicas para Aplicación Local 51
2.5 Programación Extrema 51
2.5.1 Tareas Principales de la Programación Extrema 51
2.5.2 Ventajas 53
Capítulo 3: Marco Aplicativo 55

3.1 Iteración 0 56
3.2 Iteración 1 59
3.3 Iteración 2 61
3.4 Iteración 3 65
3.5 Iteración 4 71
3.6 Iteración 5 76
3.7 Iteración 6 80
3.8 Iteración 7 83
3.9 Iteración 8 86
Resultados 90
Conclusiones 91
Recomendaciones 93
Referencias Bibliográficas 94
Anexos 97

Índice de Figuras

Figura 1.1: Página Principal 15
Figura 2.1: Proceso de Digitalización 23
Figura 2.2 (a): Imagen Original 24
Figura 2.2 (b): Filtro Pasa Bajos 24
Figura 2.3 (a): Imagen Original 25
Figura 2.3 (b): Filtro Mediana 25
Figura 2.4 (a): Imagen Original 25
Figura 2.4 (b): Filtro Gaussiano 25
Figura 2.5 (a): Imagen Original 26
Figura 2.5 (b): Filtro Thinning 26
Figura 2.6: Proceso OCR 27
Figura 2.7: Ajuste por Plantilla 29
Figura 2.8: Arquitectura Cliente/Servidor 30
Figura 2.9: Modelo FTP 33
Figura 2.10: Sistema Manejador de Bases de Datos 34
Figura 2.11: Patrón MVC 41
Figura 2.12: Componentes 42
Figura 2.13: Modelo Ajax 50
Figura 2.14: Ciclo XP 52
Figura 3.1: Ventana de Error 56
Figura 3.2: Vista Principal Empaquetador 57
Figura 3.3: Ventana para Seleccionar Carpeta 57
Figura 3.4: Código para Generar Paquete de Imágenes 58
Figura 3.5: Mensaje de Confirmación 59
Figura 3.6: Código para Transparencia de Datos 60
Figura 3.7: Servidor FileZilla 62
Figura 3.8: Código para Procesamiento de Paquetes 63

Figura 3.9: Código para el Almacenamiento de Imágenes 64
Figura 3.10: Código para Emisión de Oficios en formato PDF 64
Figura 3.11: Carpetas Pendientes 66
Figura 3.12: Oficios Pendientes 66
Figura 3.13 (a): Procesamiento de Oficios 67
Figura 3.13 (b): Procesamiento de Oficios 67
Figura 3.14: Código para Listar Carpetas Pendientes 68
Figura 3.15: Código para Listar Oficios sin Procesar 68
Figura 3.16: Código para Carga de Oficios y Opciones de Procesamiento 68
Figura 3.17: Código para Procesamiento OCR 69
Figura 3.18: Código para Verificación de Oficios Procesado 70
Figura 3.19: Código para verificación de Carpeta Procesada 70
Figura 3.20: Carpetas a Procesar 72
Figura 3.21: Oficios a Procesar 72
Figura 3.22: Procesamiento de Imágenes 73
Figura 3.23: Código para el Procesamiento Digital de Imágenes 74
Figura 3.24: Opciones para el Procesamiento Digital de Imágenes 74
Figura 3.25: Código para Restablecer una Imagen 75
Figura 3.26: Código para Guardar una Imagen 75
Figura 3.27: Opciones de Consulta de Oficios 77
Figura 3.28: Descarga de Oficio en Formato PDF 77
Figura 3.29: Código para Mostrar Versiones de Imágenes 78
Figura 3.30: Código para Descargar Oficios en Formato PDF 79
Figura 3.31: Código para Cambiar Imagen 79
Figura 3.32: Vista para Usuarios 81
Figura 3.33: Vista para Administradores 82
Figura 3.34: Vista para Súper-administrador 82
Figura 3.35: Código para Generar Opciones de Usuarios Dinámicamente 83
Figura 3.36: Vista para Opciones de Búsquedas 84
Figura 3.37: Vista para Resultado de Búsqueda 84

Figura 3.38: Vista para Detalle de Búsqueda 85
Figura 3.39: Código para Búsquedas 86
Figura 3.40: Vista para Opciones de Emisión de Reportes 87
Figura 3.41: Código para la Búsqueda de Emisión de Reportes 88
Figura 3.42: Código para la Emisión de Reportes en Formato PDF 88

Índice de Tablas

Tabla 2.1: Historias de Usuarios 52
Tabla 3.1: Planificación Iteración 0 56
Tabla 3.2: Planificación Iteración 1 59
Tabla 3.3: Planificación Iteración 2 61
Tabla 3.4: Planificación Iteración 3 65
Tabla 3.5: Planificación Iteración 4 71
Tabla 3.6: Planificación Iteración 5 76
Tabla 3.7: Planificación Iteración 6 80
Tabla 3.8: Planificación Iteración 7 83
Tabla 3.9: Planificación Iteración 8 86
Tabla 4.1: Resultados del Proceso OCR Mediante el Uso de Distintos Filtros 90

Parte I

Introducción y Propuesta

Introducción

Con el continuo desarrollo de las tecnologías informáticas, muchas organizaciones se han visto en la necesidad de migrar a sistemas informáticos automatizados, que les permitan ofrecer mayor calidad de servicio al menor tiempo.

Muchas organizaciones deben transcribir los datos manualmente ya que, en la actualidad no existen mecanismos suficientes para realizar este proceso automáticamente, corriendo el riesgo de cometer errores y tardar mucho tiempo en hacerlo.

La División de Control de Estudios de la Facultad de Ciencias de la Universidad Central de Venezuela, ha venido migrando sus procesos progresivamente a sistemas informáticos automatizados, mejorando así su calidad de servicio.

En la actualidad, la División de Control de Estudio de la Facultad de Ciencias ha decidido automatizar el proceso de apertura y modificación de materias, pero se requiere toda la información histórica de dicho proceso almacenada en un archivo físico localizado en la sede dicha organización, esta información data del año 1950 hasta la actualidad. Por esta razón, se requiere digitalizar los datos para poder incluirlos dentro del sistema de manejo de aperturas y modificación de materias.

Debido a que es un archivo histórico se maneja un gran número de oficios donde se encuentran los datos y sería muy tedioso y consumiría gran cantidad de tiempo en transcribir todos los datos para poder procesarlos. Por tal motivo se desarrolló un sistema que permite apoyar la recolección de datos y su procesamiento.

En el presente trabajo de investigación se presenta un conjunto de tecnologías empleadas para desarrollar el sistema de digitalización de oficios, así como también la especificación de la elaboración de dicho sistema, siguiendo la metodología XP.

Este documento está estructurado de la siguiente manera:

- **Capítulo I: Planteamiento del problema:** En este capítulo se presenta la problemática, justificación del mismo y la solución propuesta.

- **Capítulo II: Marco Conceptual:** Se estudian todas las tecnologías y la metodología ágil utilizadas para la construcción del sistema de digitalización de oficios.

- **Capítulo III: Marco Aplicativo:** Se especifica cada una de las iteraciones de construcción del sistema, siguiendo la metodología ágil XP (Programación Extrema).

- **Capítulo IV: Resultados, Conclusiones, y Recomendaciones:** Se muestra los resultados de las pruebas realizadas al proceso OCR (Reconocimiento Óptico de Caracteres), se plantean las conclusiones del Trabajo Especial de Grado y las recomendaciones.

Capítulo 1

Planteamiento del Problema

A continuación, se presenta el contexto de la problemática, la cual está basada en la automatización del proceso de digitalización de documentos, específicamente los oficios de aperturas y/o modificación de materias que se encuentran archivados en la División de Control de Estudios de la Facultad de Ciencias (DCE). Se plantea un sistema que permita automatizar, almacenar y controlar el procesamiento de datos provenientes de imágenes que contienen la información exacta de los oficios originales.

1.1 Título

Desarrollo de una aplicación para procesar y organizar oficios digitalizados de modificación y apertura de materias de la División de Control de Estudios (DCE) de la Facultad de Ciencias de la Universidad Central de Venezuela.

1.2 Situación Actual

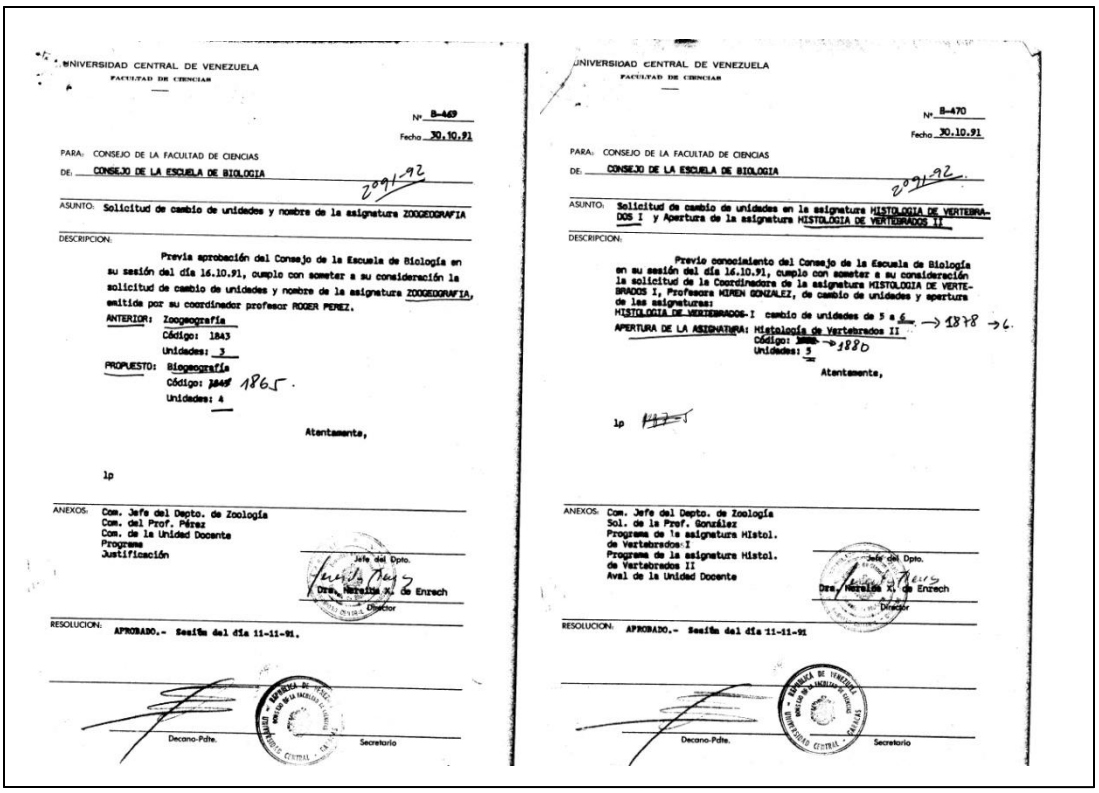
La DCE es la encargada de manejar y mantener información referente a las aperturas y modificaciones de materias de las 5 escuelas que forman Facultad de Ciencias.

Toda esta información se encuentra almacenada en un archivo físico, que data de los años 50 hasta la fecha, este está localizado en las instalaciones de la DCE, específicamente en la oficina de la Directora de dicha división (actualmente la profesora Damarys Barrante).

Durante los últimos años, el archivo físico de oficios ha estado a cargo de la Sra. Oneida Mendoza, quien archiva nuevos oficios, organiza y mantiene cada uno de ellos, estos cuentan con una página principal seguida de anexos que justifican la apertura y/o modificación de una materia en específico, programación, entre otras especificaciones.

Los oficios son almacenados por carpetas, donde cada una representa una escuela de la Facultad de Ciencias, sin ningún orden cronológico.

Los oficios están comprendidos por una página principal que trata de seguir un formato (Ver Figura 1.1), comprendido por el nombre de la universidad, la facultad, la escuela, el asunto, descripción, fecha, entre otros, pero todos difieren en cuanto a los anexos.



UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS

Nº B-469
Fecha 30.10.91

PARA: CONSEJO DE LA FACULTAD DE CIENCIAS
DE: CONSEJO DE LA ESCUELA DE BIOLOGIA
209.92

ASUNTO: Solicitud de cambio de unidades y nombre de la asignatura ZOOGEOGRAFIA

DESCRIPCION:

Previa aprobación del Consejo de la Escuela de Biología en su sesión del día 16.10.91, cumplo con someter a su consideración la solicitud de cambio de unidades y nombre de la asignatura ZOOGEOGRAFIA, emitida por su coordinador profesor ROGER PEREZ.

ANTERIOR: Zoogeografía
Código: 1843
Unidades: 3

PROPUESTO: Biogeografía
Código: 1865
Unidades: 4

Atentamente,

lp

ANEXOS: Com. Jefe del Depto. de Zoología
Com. del Prof. Pérez
Com. de la Unidad Docente
Programa
Justificación

Jefe del Dpto.
Dra. Mercedes X. de Enrech
Director

RESOLUCION: APROBADO.- Sesión del día 11-11-91.

Decano-Pdte.
Secretario

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS

Nº B-470
Fecha 30.10.91

PARA: CONSEJO DE LA FACULTAD DE CIENCIAS
DE: CONSEJO DE LA ESCUELA DE BIOLOGIA
209.92

ASUNTO: Solicitud de cambio de unidades en la asignatura HISTOLOGIA DE VERTEBRADOS I y Apertura de la asignatura HISTOLOGIA DE VERTEBRADOS II

DESCRIPCION:

Previo conocimiento del Consejo de la Escuela de Biología en su sesión del día 16.10.91, cumplo con someter a su consideración la solicitud de la Coordinadora de la asignatura HISTOLOGIA DE VERTEBRADOS I, Profesora MIREN GONZALEZ, de cambio de unidades y apertura de las asignaturas:
HISTOLOGIA DE VERTEBRADOS-I cambio de unidades de 5 a 6 → 1878 → 6.
APERTURA DE LA ASIGNATURA: Histología de Vertebrados II
Código: 1880
Unidades: 5

Atentamente,

lp

ANEXOS: Com. Jefe del Depto. de Zoología
Sol. de la Prof. González
Programa de la asignatura Histol. de Vertebrados I
Programa de la asignatura Histol. de Vertebrados II
Aval de la Unidad Docente

Jefe del Dpto.
Dra. Mercedes X. de Enrech
Director

RESOLUCION: APROBADO.- Sesión del día 11-11-91

Decano-Pdte.
Secretario

Figura 1.1 – Página Principal

1.3 Planteamiento del Problema

El manejo de oficios de aperturas y modificación de materias de la Facultad de Ciencias es manual, por lo tanto las búsquedas deben hacerse una a una hasta encontrar el oficio deseado, así como también la organización de los mismos, consumiendo gran cantidad de tiempo.

En los últimos años, la DCE ha venido migrando sus procesos progresivamente a procesos automatizados con la colaboración de los trabajadores de dicha división, profesores y estudiantes de la Facultad de Ciencias, específicamente la Escuela de Computación.

En la actualidad, la DCE ha decidido automatizar los procesos de búsqueda y organización de oficios de aperturas y modificaciones, para ello se debe digitalizar todos los datos almacenados en el archivo físico, por tal razón dicha división cuenta

con la presencia de un escáner, pero no cuenta con ninguna herramienta que permita automatizar los procesos relacionados con la organización de los oficios una vez que estos han sido escaneados.

En respuesta la problemática actual se plantea un sistema, el cual consta de dos partes principales, una aplicación local para configurar el formato de entrada de los oficios y una aplicación Web para el procesamiento de los oficios.

1. La aplicación local será realizada en lenguaje java y será la encargada configurar el formato de entrada de los oficios. Esta generará un archivo el cual contendrá la metada y empaquetara la Meta-data y las imágenes para su procesamiento.

2. La aplicación Web será realizada en Ruby on Rails y será la encargada de procesar los oficios mediante el uso de la herramienta de reconocimiento óptico de caracteres OCR, filtros que permitan el procesamiento de imágenes en pro a la mejora de la calidad de las mimas y almacenamiento de los datos, realizar consultas, manejo de usuarios, entre otros.

1.4 Objetivos

A continuación se describe el objetivo general y objetivos específicos del Trabajo Especial de Grado.

1.4.1 Objetivo General

Desarrollar un sistema que permita automatizar el proceso de digitalización y manejo de los datos de los oficios de apertura y modificación de materias almacenados en el archivo de oficios en la División de Control de Estudios de la Facultad de Ciencias de la Universidad Central de Venezuela.

1.4.2 Objetivos Específicos

- Levantar la información en conjunto con los miembros de la DCE, específicamente las personas relacionadas con el manejo del archivo de oficios y automatización de la información.

- Aplicar el método ágil, programación extrema, para el desarrollo del sistema que lleve a cabo los procesos referentes a la digitalización de los datos de los oficios.

- Diseñar e implementar una aplicación local que permita empaquetar los oficios.

- Diseñar las interfaces de usuario de cada uno de los módulos de la aplicación.

- Diseñar e implementar la base de datos donde será almacenada la información provente de los datos obtenidos del proceso de digitalización de los oficios.

- Implementar todos los módulos de la aplicación Web.

- Implementar el manejo de roles de usuario.

- Diseñar e implementar el módulo de procesamiento de datos, donde se provea de opciones, tales como: generación de formularios por parte del usuario, uso de reconocimiento óptico de caracteres (OCR, en sus siglas en ingles), entre otras opciones.

- Diseñar e implementar el módulo de procesamiento de imágenes, donde se provea al usuario de opciones, tales como: manejo de filtros para el procesamiento digital de imágenes.

- Diseñar e implementar el módulo de consultas de oficios, donde estos se muestran los oficios y sus anexos, además de las versiones (en el caso de que existan) de cada imagen.

- Diseñar e implementar el módulo de búsquedas, donde el usuario tendrá la posibilidad de realizar búsquedas por código de materia, fecha entre otros.

- Diseñar e implementar el módulo de generación de reportes, donde el usuario tendrá la posibilidad de generar reportes de acuerdo a la fecha, código de materias, etc.

- Realizar pruebas a cada uno de los módulos, en conjunto con los usuarios finales.

- Integrar todos los componentes del sistema.

- Realizar pruebas al sistema en conjuntos con los usuarios finales.

1.5 Justificación

La DCE mantiene un archivo, donde a través de los años se almacenan los oficios históricos de apertura y modificación de materias de la Facultad de Ciencias, los oficios son archivados manualmente, al igual que las búsquedas de datos.

De igual forma, por el uso frecuente de los mismos para realizar búsquedas u organizarlos estos se han ido deteriorando con el riesgo de perder datos históricos, los cuales representan un legado para la Facultad de Ciencias.

Muchas materias han sido modificadas o aperturadas con un mismo nombre o con nombres diferentes pero tratándose de la mismas materia, trayendo como consecuencia la duplicación de datos, ya que no se cuenta con un mecanismo que

permita controlar las entradas al archivo y la apertura o modificación de materias que tengan alguna relación.

Debido a la importancia del manejo y mantenimiento de los oficios históricos de apertura y modificación de materias en la Facultad de Ciencias, se desea desarrollar un sistema que permita automatizar el proceso de digitalización y manejo de los datos, donde se garantice consistencia de los datos y la disminución de tiempo de dicho proceso.

1.6 Alcance

La aplicación propuesta debe cumplir con las siguientes funcionalidades:

- Generar Meta-data y comprimir carpetas que contengan imágenes escaneadas de oficios de apertura y/o modificación.

- Recibir paquetes y organizar las imágenes recibidas por medio del archivo donde se encuentra la Meta-data.

- Administración de carpetas, controla cuales carpetas tienen oficios que aun no han sido procesados y las lista como pendientes en el menú principal de procesamiento de datos.

- Proveer a los usuarios de interfaces para el procesamiento de datos, imágenes y consulta de oficios.

- Almacenamiento de los datos de los oficios de apertura y modificación de materias obtenidos por medio del módulo de procesamiento de datos e imágenes.

- Permitir búsquedas por medio de código de materia, nombre, fecha entre otros.

- Permitir la generación y emisión de reportes.

- Manejo de roles de usuarios, dependiendo del tipo de usuario estos podrán realizar acciones especificas dentro del sistema, los menús serán generados dinámicamente dependiendo del usuario y su rol.

- Generación de formularios por medio de los usuarios, en el caso de procesamiento de datos.

Parte II

Marco Conceptual

Capítulo 2

Marco Conceptual

La finalidad de este capítulo es presentar las bases conceptuales sobre las cuales se fundamenta el Trabajo Especial de Grado. Éste se encuentra dividido en cuatro partes, las cuales serán descritas a continuación:

En el primer segmento se presenta el proceso de digitalización, se describe la fase de reconocimiento óptico de caracteres (OCR en sus siglas en ingles) y como el procesamiento de imágenes, específicamente la aplicación de filtros gráficos, mejora la calidad de las imágenes en apoyo al proceso OCR.

En el segundo segmento se presenta la arquitectura cliente/servidor, donde específicamente se expone el concepto y componentes de dicha arquitectura.

En el tercer segmento se presentan las herramientas tecnológicas utilizadas para la implementación del sistema, como por ejemplo: Ruby On Rails, RMagick, HTML, CSS, entre otros.

En el cuarto y último segmento se presenta la metodología ágil XP, utilizada para la implementación del sistema de automatización del proceso de digitalización.

2.1 Digitalización

2.1.1 Fases del proceso de digitalización

El proceso de digitalización esta divido en tres fases (Ver Figura 2.1), las cuales serán explicadas a continuación:

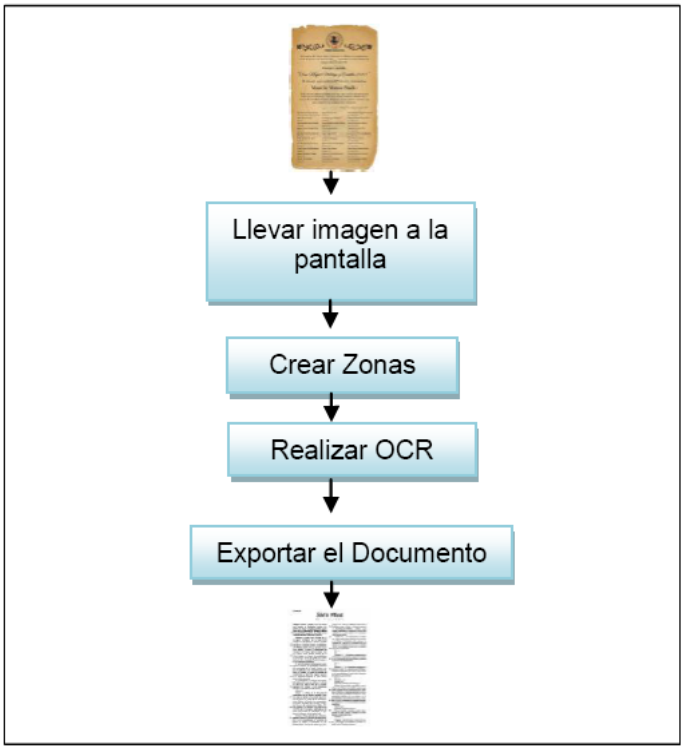


Figura 2.1-Proceso de digitalización

- Llevar imagen a la pantalla: Se utiliza un escáner, el cual se encargará de recolectar las características de la página para convertirlo en una imagen digital.

- Crear zonas: Esto se realiza para identificar las partes del documento que se quieren reconocer como texto. Las zonas seleccionadas son las únicas partes incluidas en el reconocimiento.

- Realizar OCR: En esta operación se convierte la parte seleccionada en texto editable. Durante el OCR, las aplicaciones definen los caracteres de texto que se van identificando. Una vez culminado este proceso de traducción se procede a chequear el resultado y corregir los posibles errores.

- Exportar el documento: Esta última fase se basa en exportar el resultado del proceso OCR a la aplicación deseada, bien sea un editor textos o una base de datos.

2.1.2 Procesamiento de imágenes

Para apoyar el proceso OCR es importante que la imagen tenga la mejor calidad, es por ello que se deben emplear un conjunto de técnicas para mejorar su calidad, y facilitar la búsqueda de información previa a dicho proceso [1].

Filtros

Son utilizados para la modificación de imágenes, bien sea para la detección de bordes de una escena, para la modificación del aspecto de la misma o para la eliminación de ruido de la imagen. Entre los principales se encuentran:

- Pasa bajos: filtro caracterizado por la eliminación de ruido, detalles pequeños o detalles que no sean de interés dentro de una imagen (Ver Figura 2.2(a) y Figura 2.2 (b)) [2].

Figura 2.2 (a) – Imagen original

Figura 2.2 (b) – Filtro pasa bajos

- Mediana: técnica utilizada principalmente para el suavizado de imágenes y eliminación de ruidos, presentando la ventaja con respecto a otros operadores, por ejemplo el filtro paso bajo, de que mantiene la información de borde (Ver Figura 2.3(a) y Figura 2.3 (b)) [3].

Figura 2.3 (a) – Imagen original

Figura 2.3 (b) – Filtro mediana

- Gaussiano: Este tipo de filtro permite reducir el ruido gaussiano, el cual produce pequeñas variaciones en la imagen (Ver Figura 2.4(a) y Figura 2.4 (b)) [4].

Figura 2.4 (a) – Imagen original

Figura 2.4 (b) – Filtro Gaussiano

- Adelgazamiento: Consiste en la reducción de pixeles abundantes. El proceso de adelgazamiento se basa en la eliminación de partes de una imagen (Ver Figura 2.5(a) y Figura 2.5 (b)) [5].

Figura 2.5 (a) – Imagen original

Figura 2.5 (b) – Filtro de adelgazamiento

2.1.3 Reconocimiento óptico de caracteres

Extrae de una imagen los caracteres que componen un texto, estos son traducidos mecánica o electrónicamente y almacenados en formato editable [6].

Funcionamiento

El OCR busca conjuntos de puntos que se asemejen a caracteres. Dependiendo de la complejidad de dicho programa entenderá más o menos tipos de letra, llegando en algunos casos a interpretar la escritura manual, mantener el formato original (columnas, fotos entre el texto...) o aplicar reglas gramaticales para aumentar la exactitud del proceso de reconocimiento [7]. En general hay dos métodos básicos utilizados para OCR:

- Ajuste de Matriz: Compara lo que el escáner del OCR percibe como un caracter con una biblioteca de matrices de caracteres o plantillas. Cuando una imagen coincide con una de esas matrices de puntos dentro de un determinado nivel de similitud, la imagen es etiquetada con el caracter ASCII[1] correspondiente a la matriz.

[1] ASCII Código (Estándar Estadounidense para el Intercambio de Información)

- Extracción de características: también conocido como Reconocimiento Inteligente de Caracteres (ICR en sus siglas en ingles). El sistema, básicamente, busca características generales, tales como áreas abiertas, formas cerradas, líneas diagonales, líneas entrecruzadas, estilo. Donde los caracteres son menos predecibles, caracterizados, o el análisis topográfico es superior.

Fases del proceso OCR

El proceso OCR está dividido en tres fases (ver Figura 2.6), las cuales serán descritas a continuación.

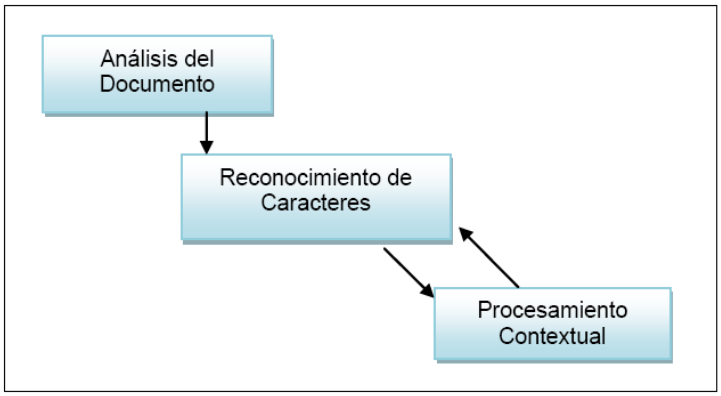


Figura 2.6 – Proceso OCR

- Análisis del documento: Se basa en la extracción del texto de la imagen, es decir, se segmenta en las diferentes componentes. La segmentación es la operación que permite la descomposición de un texto en diferentes entidades lógicas (caracteres). Estas entidades lógicas deben ser lo suficientemente invariables, para ser independientes del escritor, y lo suficientemente significativas para su reconocimiento. La fiabilidad de la segmentación de caracteres y del reconocimiento de los mismos depende de la calidad del documento original y la calidad de la imagen escaneada.

- Reconocimiento de caracteres, dos componentes esenciales en un algoritmo de reconocimiento de caracteres son:

    - El extractor de características, determina los descriptores o conjunto de características usados para describir todos los caracteres. Las características derivadas son entonces usadas como entrada del clarificador de caracteres.

    - Clasificación de caracteres: Ajuste de platilla, o ajuste de matriz, es uno de los métodos más comunes. En el ajuste de plantillas, píxeles de una imagen individual son usados como características. La clasificación es realizada mediante la comparación de un caracter de una imagen como entrada y un conjunto de plantillas (o prototipos) de cada clase de caracter (ver Figura 2.7). Cada comparación resulta en una medida de similitud entre el caracter de entrada y la plantilla (matriz). La medida incrementa la cantidad de similitud cuando un pixel en el caracter observado es idéntico al mismo pixel en la platilla de la imagen. Si el pixel difiere la medida de similitud disminuye su valor. Después de que todas las plantillas hayan sido comparadas con el caracter de la imagen observada, la identidad del caracter es asignada como la identidad de la plantilla más similar, determinada por la medida de similitud más alta.

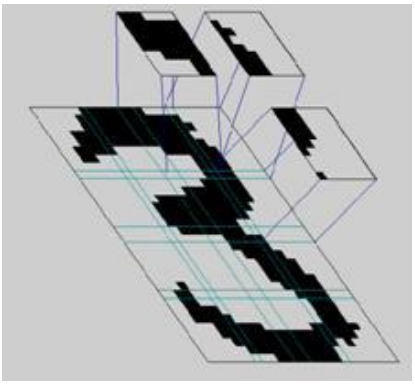


Figura 2.7 – Ajuste por plantilla

- Procesamiento contextual: El número de opciones de palabras para un campo dado puede ser limitado sabiendo el contenido de otro campo, por ejemplo, el reconocimiento del nombre de una calle en una dirección, mediante un correcto reconocimiento del código postal, limita a un léxico específico la opción del nombre de la calle. Alternativamente, el resultado del reconocimiento puede ser procesada posteriormente para corregir los posibles errores. Un método utilizado para corregir los resultados de un reconocimiento de caracteres se basa en aplicar un corrector, el cual chequea cada letra de cada palabra. Similarmente, existe otro método, el cual se basa en el uso de léxicos para verificar las palabras resultantes o reconocimientos resultantes podrían ser verificados interactivamente con el usuario [8].

2.2 Arquitectura Cliente/Servidor

Patrón arquitectónico para el desarrollo de sistemas distribuidos, el cual distribuye una aplicación entre dos o más componentes especializados cuya ejecución se distribuye entre uno o más equipos [9].

2.2.1 Componentes

Define dos tipos de entidades diferenciadas (asimétricas) que se responsabilizan de acciones diferentes: clientes y servidores. Especifica 2 tipos de procesos con roles diferenciados, donde se define un modelo de interacción basado en el concepto de servicio implementado sobre un diálogo petición-respuesta (Ver Figura 2.8).

- Cliente inicia el diálogo mediante el envío de peticiones.

- Servidor presta el servicio y responde las peticiones recibidas y especifica el modo en que se sincronizan los procesos.

- Cliente (parte activa) demanda servicios a los servidores se asume que cada petición debería obtener respuesta diseñado para soportar la interacción con el usuario final.

- Servidor (parte pasiva) espera las peticiones de los clientes procesa esas peticiones y envía una respuesta.

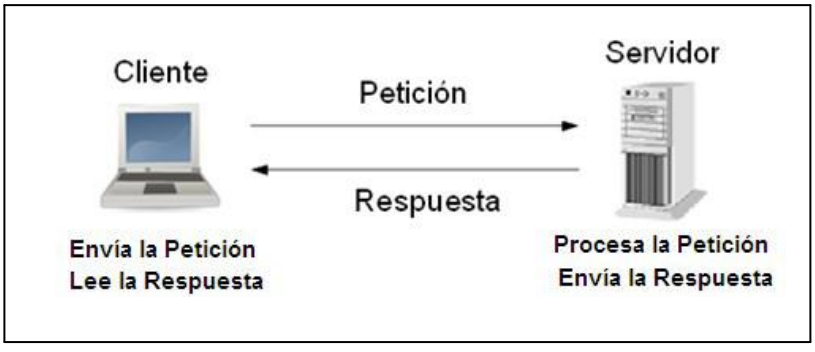


Figura 2.8 – Arquitectura Cliente/Servidor

2.3 Protocolos de transferencia de datos

2.3.1 HTTP

HTTP (en sus siglas en ingles Hypertext Transfer Protocol) es un protocolo de petición- respuesta utilizado para la transferencia de hipertexto [10].

Entre las propiedades de HTTP se pueden destacar las siguientes:

- Un esquema de direccionamiento comprensible. Utiliza el Universal Resource Identifier (URI) para localizar sitios (URL) o nombres (URN) sobre los que hay que aplicar un método. La forma general de un URL es servicio://host/fichero.ext.

- Arquitectura Cliente/Servidor. HTTP se asienta en el paradigma solicitud/respuesta. El puerto por defecto es el 80, pero se pueden utilizar otros.

Una transacción HTTP está compuesta por una cabecera, y opcionalmente, por una línea en blanco seguida de los datos. En la cabecera se especifica tanto la acción solicitada en el servidor, como los tipos de datos devueltos o un código de estado.

2.3.2 FTP

FTP (en sus siglas en ingles File Transfer Protocol) es un protocolo de red para transferencia de archivos entre sistemas conectados a una red TCP[2], basado en la arquitectura cliente/servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo [11].

Servidor FTP

Programa especial que se ejecuta en un equipo servidor conectado a una red bien sea Internet, LAN, MAN, entre otros. Su función es permitir el intercambio de datos entre diferentes servidores/clientes (ver Figura 2.9).

Cliente FTP

Un cliente FTP es un programa que se instala en el cliente, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos, generalmente por el puerto 21.

Para utilizar un cliente FTP, se necesita conocer el nombre del archivo, el equipo en que reside (servidor, en el caso de descarga de archivos), el equipo al que se quiere transferir el archivo (en caso de querer subirlo nosotros al servidor), el nombre de usuario, clave y la carpeta en la que se encuentra.

---

[2]TCP (Protocolo de Control de Transmisión) garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Microsoft Windows, DOS, GNU/Linux y Unix.

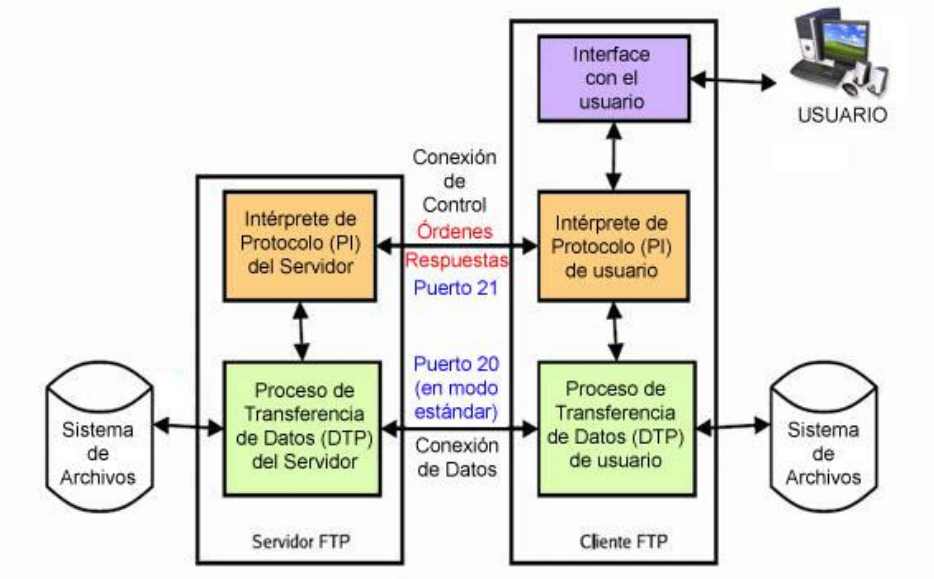


Figura 2.9 – Modelo FTP

2.4 Herramientas tecnológicas

2.4.1 Herramientas tecnológicas para aplicación Web

Sistema manejador de Bases de Datos

El sistema manejador de base de datos es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos. Se compone de un lenguaje de definición de datos (DDL: Data Definition Language), de un lenguaje de manipulación de datos (DML: Data Manipulation

Language) y de un lenguaje de consulta (SQL: Structured Query Language) (Ver Figura 2.10) [12].

El lenguaje de definición de datos (DDL) es utilizado para describir todas las estructuras de información y los programas que se usan para construir, actualizar e introducir la información que contiene una base de datos.

El lenguaje de manipulación de datos (DML) es utilizado para escribir programas que crean, actualizan y extraen información de las bases de datos.

El lenguaje de consulta (SQL) es empleado por el usuario para extraer información de la base de datos. El lenguaje de consulta permite al usuario hacer requisiciones de datos sin tener que escribir un programa, usando instrucciones como el SELECT, el PROJECT y el JOIN.

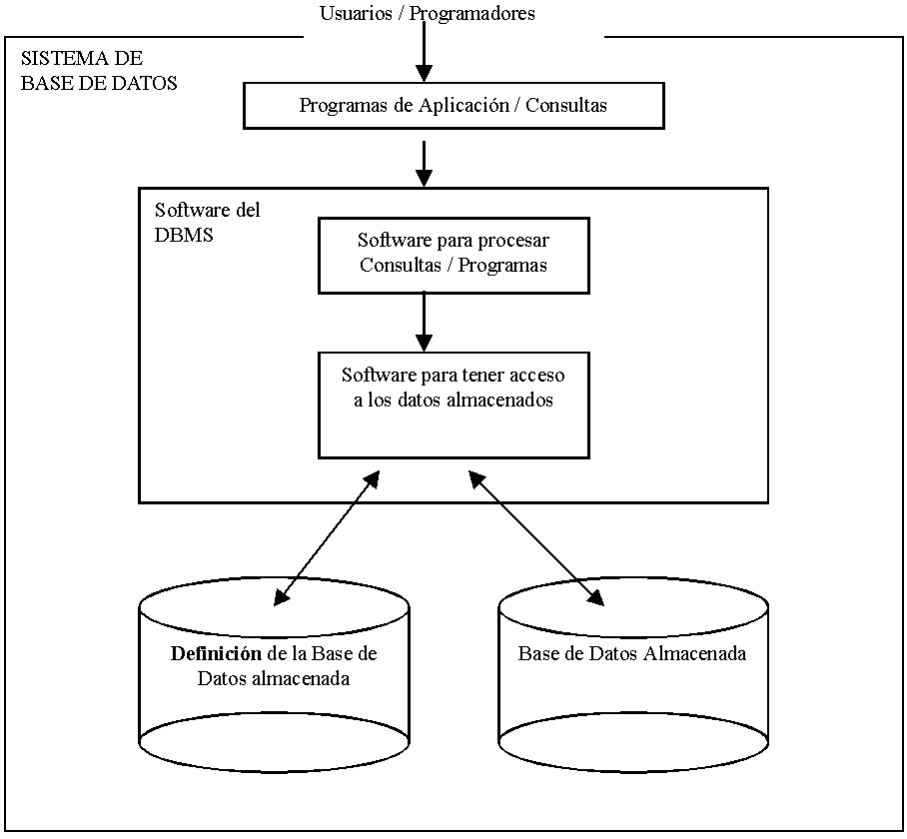


Figura 2.10 – Sistema Manejador de Bases de Datos

Mysql

MySQL es un sistema de administración de bases de datos para bases de datos relacionales. Así MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos. Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos [13].

MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información.

MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

Ruby on Rails

Framework de aplicaciones Web de código abierto escrito en lenguaje Ruby, mejor conocido como Ruby on Rails, este sigue el paradigma de la arquitectura Modelo Vista Controlador (MVC). Rails se distribuye a través de RubyGems [14].

Ruby

Lenguaje de programación interpretado, reflexivo y orientado a objetos. Combina una sintaxis inspirada en Python y Perl con características de programación orientadas a objetos similares a Smalltalk. Su implementación oficial es distribuida bajo una licencia de software libre.

En 1995, Yukihiro Matsumoto presentó la primera versión de Ruby. La principal razón para crear Ruby fue que él quería un lenguaje de secuencia de comandos que optimizara la forma en que el programador desarrollaría el software, esto quiere decir que las características de Ruby optimizan la forma como el software es desarrollado [15].

Características

- Interpretado: Es un lenguaje interpretado, por ende, siempre que se realice un cambio al código fuente, no es necesario compilar el código y correrlo para ver el efecto del cambio. Como resultado de esta característica, el ciclo código-compilación-corrida se transforma a código-corrida.

- Puramente Orientado a Objetos: Esto quiere decir que todo en Ruby es un objeto, el cual incluye tipo de datos primitivos y números. Además, Ruby soporta todas las características requeridas por un lenguaje orientado a objetos.

- Funcional: Ruby Soporta programación funcional mediante el uso de bloques.

- Tipado dinámico: Ruby decide acerca del tipo de variable a medida que el programa está corriendo mediante la búsqueda del valor contenido en la variable en ese instante.

- Administración automática de memoria: Mejor conocida como recolector de basura. Como muchos lenguajes de alto nivel, Ruby provee del recolector de basura, por ende no es necesario preocuparse por la liberación de bloques de memoria que no serán utilizados.

- Hilo de ejecución: La versión 1.8.6. de Ruby provee “casi” una plataforma independiente de hilo de ejecución usando hilos de ejecución a nivel de usuario. Se dice “casi” porque los hilos de ejecución de Ruby son simulados en la MV (maquina virtual) en lugar de correr como hilos nativos del sistema operativo.

- Reflexión: Ruby provee un programa con la habilidad de ‘Verse el mismo’ mientras corre. Esta habilidad es conocida por diferentes términos, como reflexión, introspección, etc. Usando reflexión, un programa puede modificar ciertos aspectos de sí mismo durante la ejecución, o crear un objeto nuevo completamente en tiempo de ejecución basado en los requerimientos en tiempo real.

- Altamente portable.

- Cuatro niveles de ámbito de variable: Global, clase, instancia y local.

- Manejo de excepciones: Posee una estructura de control diseñada para manejar condiciones anormales que pueden ser tratadas por el mismo programa que se desarrolla.

- Iteradores y clausuras.

- Expresiones regulares.

- Posibilidad de definir lo operadores (sobrecarga de operadores).

- Soporta inyección de dependencias: Posee un patrón de diseño orientado a objetos, en el que se suplen objetos a una clase en lugar de ser la propia clase quien cree el objeto [16].

RubyGems

Ruby cuenta con un gestor de paquetes que proporciona un formato estándar y autocontenido (llamado gem) para poder distribuir programas o librerías en Ruby, una herramienta destinada a gestionar la instalación de estos, y un servidor para su distribución.

Filosofía

- Dry: No te repitas, si es aplicado apropiadamente, reduce la duplicación de tareas dentro de un proyecto. Duplicación de cualquier tipo, dentro de un proyecto, dificulta la modificación y mantenimiento e incrementa la posibilidad de inconsistencia. En Ruby on Rails, se puede ver este principio trabajando en casi todo, desde los componentes reusables en las formas de plug-ins hasta la forma como la tablas de las base de datos son mapeadas.

- Convención sobre Configuración: También conocida como CoC, por sus siglas en ingles Convention-over-Configuration. La convención se ha hecho cargo de los frameworks de aplicaciones Web que hasta una simple tarea como la aplicación "campo obligatorio" (validaciones solo para un campo) requiere entradas en un archivo XML. En Ruby on Rails, el principio es sustituir información solo sobre aspectos que son diferentes de las configuraciones habituales de aplicaciones. El framework ORM (Mapeo Objeto-Relacional) provisto por Ruby on Rails es un ejemplo del principio de CoC. A menos que se especifique un nombre diferente para un objeto ORM,

el objeto usa el nombre de la tabla por la cual es mapeado. En Ruby on Rails, un cambio en el esquema no significa un cambio en el objeto a menos que el nombre de la tabla en si cambie.

Características

Las características basadas en la filosofía de los principios Dry y CoC son las que hacen a Ruby on Rails tan atractivo para el desarrollo de sitios Web dinámicos. Las características que muestran la forma de métodos alternativos para la implementación de varias técnicas del lado del servidor son:

- Configuración automática de la estructura de la aplicación: La estructura de cualquier aplicación no tiene que ser creada manualmente. Con tan solo un comando y la estructura completa incluyendo carpetas y archivos básicos Web como el index.html se genera automáticamente.

- Generación de código repetitivo: Toda aplicación tiene ciertos bloques de código que son esencialmente los mismos para las otras aplicaciones del mismo tipo o categoría. Tales bloques son llamados código repetitivo. Un ejemplo de código repetitivo es el bloque de código de configuración de la conexión a la base de datos. El mismo código puede ser usado por diferentes aplicaciones con el título cambiado. Aunque este es el caso, la mayoría de los frameworks no proveen ningún mecanismo incorporado para reducir este "reinvención de la rueda". Ruby on Rails evita la duplicación usando andamios. En esencia, un andamio contiene el mínimo código para realizar tareas como la conexión a la base de datos, la creación de un registro, entre otros. Los andamios reflejan el principio Dry que se adhiere a Ruby on Rails.

- Mapeado dinámico de clases a esquemas de bases de datos: Ninguna aplicación puede estar en línea sin tener una base de datos como su back-end. El framework ORM tiene fácil acceso a la base de datos. Sin embargo, el aspecto de configuración reduce algunas ventajas para el desarrollador. En el caso de Ruby on Rails, ORM no necesita ninguna configuración. En tiempo de ejecución, Ruby on Rails lee y mapea el esquema basado en los nombres de las clases y tablas correspondientes usando reflexión y meta-programación. Además, los desarrolladores obtienen mayor productividad.

- Ajax en el núcleo de la presentación: Ajax es la tecnología que ofrece interactividad a los sitios Web sin ser intrusivo. Todas las tecnologías actuales del lado del servidor dicen apoyar el Ajax, pero el apoyo es periférico y no en el núcleo. Se tendría que bajar nuevas librerías, configurarlas, y entonces comenzar el ciclo desarrollo-complicación-despliegue-prueba otra vez. Mientras que en Ruby on Rails, Ajax es parte de las librerías centrales. Así que cuando se instala Ruby on Rails, Ajax también estará disponible. Usarlas es tan fácil como usar otra librería ofrecida por Ruby on Rails.

- Baterías incluidas: Ruby on Rails contiene muchas más librerías que ofrecen a prácticamente todos los requisitos de un sitio Web dinámico, incluídos los de gestión de diseño, correo, etc.

Patrón Arquitectónico MVC

La separación de la anotomía básica de una aplicación Rails en tres componentes distintos modelo, vista y controlador significa que Rails es construido usando la arquitectura Modelo-Vista-Controlador (MVC). MVC separa los modelos de datos de una aplicación, interfaz de usuario, y controlador lógico en componentes separados (ver Figura 2.11).

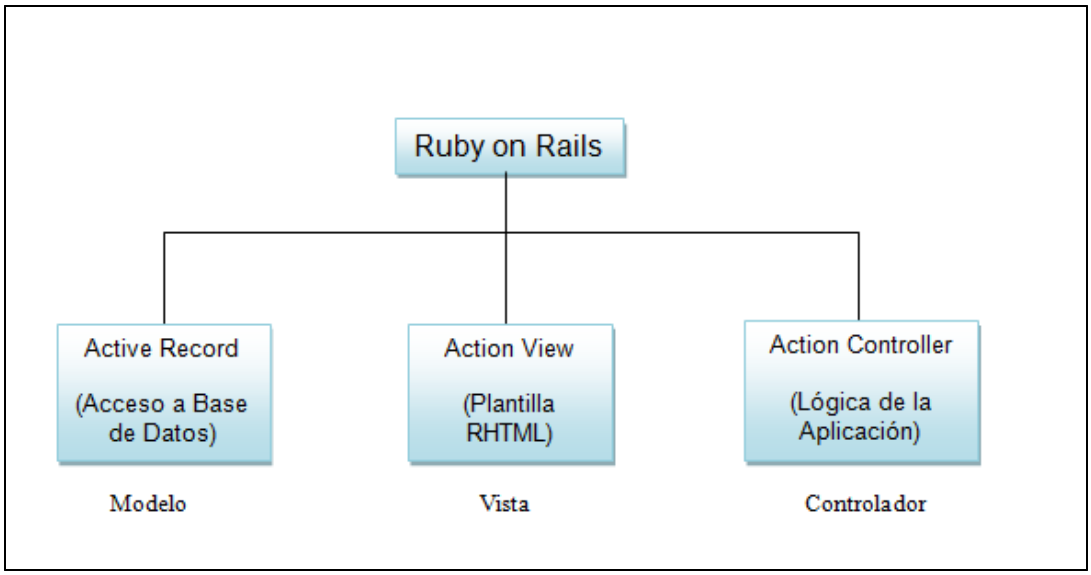


Figura 2.11 – Patrón MVC

La arquitectura MVC es importante desde la perspectiva del programador porque separa cada componente de una aplicación Rails en código base aislada que facilita la gestión sin tener que preocuparse por causar daños en otras partes de la aplicación. Por ejemplo, se puede normalmente modificar el diseño de las vistas sin tener que preocuparte de tener algún impacto en el código del controlador o del modelo. Si todo el código estuviese entrelazado a través de un simple archivo HTML que también contiene la plantilla HTML como todos los datos del modelo y del controlador, no sería gestionable y depurable. Los patrones de diseños tales como MVC son creados para facilitar la vida de los desarrolladores. Específicamente se tiene:

- Modelo: Representación especifica de la información con la cual el sistema opera.

- Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- Controlador: Responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y, en muchos casos en la vista.

Componentes

Una simple petición Web puede tomarse una gran jornada en Ruby on Rails. Cuando un usuario hace una petición de una página vía navegador, el controlador Rails (Action Controller) recibe la petición y re-direcciona a el método correcto basado en la ruta URL. Después que el método correcto es invocado, el método es ejecutado y toma todos los datos que necesita de la base de datos SQL utilizando el modelo de registro activo. Después que se tienen todos los datos necesarios, es devuelta la vista final (HTML, CSS, e imágenes) en el navegador de usuario (ver Figura 2.12).

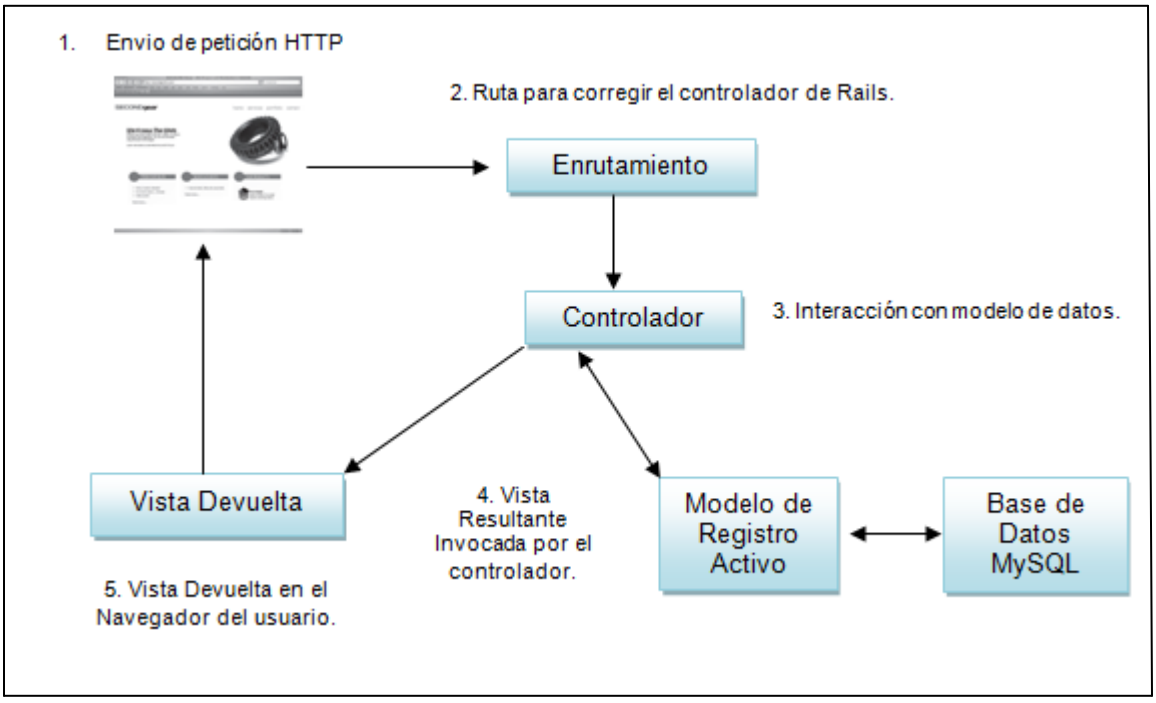


Figura 2.12 - Componentes

En la base de todas las aplicaciones están los modelos de datos que describen las reglas del negocio de las aplicaciones. Los modelos están basados en ítems reales como personas, cuentas bancarias, etc. Cada pieza de datos es representada por el uso de tablas de bases de datos y es gestionado mediante el uso del Record Activo, el cual es un objeto Rails que ofrece acceso simple a bases de datos.

El registro activo conecta el modelo de datos con las tablas de la base de datos cambiando filas de la base de datos en objetos que pueden ser manipulados. También permite describir las reglas del negocio mediante el uso de una sintaxis parecida al inglés denominada asociaciones, las cuales describen relaciones entre los diferentes modelos de datos en la aplicación [17].

Patrón MVC en Ruby on Rails

ActiveRecord

Como se mencionó previamente, el ActiveRecord no es más que un módulo para el manejo de la lógica de negocios y la comunicación con la base de datos. El juega el papel del modelo en la arquitectura MVC [18].

El ActiveRecord está diseñado para manejar todas las tareas de la aplicación referentes a la base de datos, incluyendo principalmente:

- Establecer la conexión con la base de datos.
- Recuperación de datos desde la base de datos.
- Almacenamiento de datos en la base de datos.

El módulo ActiveRecord está basado en el concepto de abstracción de base de datos, el cual se refiere a la codificación de una aplicación que no es dependiente a ninguna base de datos. El código específico para un determinado servidor de base de datos está oculto de forma segura en el ActiveRecord, y es utilizado según las necesidades. El resultado es que una aplicación Rails no está obligada a un servidor de base de datos específica. En el caso en que se necesite realizar algún cambio en la base de datos estos son efectuados solo a ella, ningún cambio en el código de la aplicación es requerida.

En Rails, el nombramiento de clases de Ruby y las tablas de las bases de datos siguen un patrón intuitivo: si se tiene una tabla llamada "usuarios", entonces la tabla almacenará los datos de objetos "usuario". Gracias al ActiveRecord, no es necesario escribir código para mapear clases con tablas de la base de datos, simplemente infiere el nombre de la tabla por el nombre de la clase. El nombre de la clase en Ruby es singular, pero el nombre de la tabla es plural.

Guardar un objeto

Para comenzar a usar el ActiveRecord, simplemente se define una clase heredada de la clase ActiveRecord : : Base. Seguidamente se instancia dicha clase y se guarda dicha instancia para que sea generado un nuevo registro en la tabla correspondiente.

Definición de relaciones entre objetos.

ActiveRecord también es el encargado de realizar el proceso de definición de relaciones entre objetos los más fácil posible. Relaciones de objetos pueden ser definidas de varias formas; la principal diferencia entre esas relaciones en el número de records que son especificados en la relación. Los principales tipos de asociaciones de bases de datos son:

- Uno a uno.
- Uno a muchos.
- Muchos a muchos.

ActionView

Componente encargado de generar las interfaces de usuario a partir de código contenido en archivos .rhtml o .erb. Estas extensiones de archivo indican que los mismos pueden contener código HTML y/o código Ruby principalmente.

ActionController

Componente que maneja el controlador de la aplicación, es decir, toda la logística de la aplicación y las interacciones entre modelo y la vista. Los controladores están compuestos por acciones, las cuales son métodos que siguen la logística de la aplicación. Cada acción tiene asociada una vista homónima, lo cual indica que el flujo de trabajo de una aplicación que funciona sobre Rails obliga a que se despliegue una vista luego de que se ejecuta una acción. Esta es una de las convenciones de Rails.

Soporte para bases de datos con Rails

El registro activo toma un modelo de clase y lo conecta con una tabla en la base de datos usando un patrón de mapeo objeto-relación. La base de datos para Rails frecuentemente usada para el desarrollo es MySQL, la cual está ampliamente disponible en múltiples plataformas, de fácil instalación, y disponibilidad gratis para el desarrollo.

Aparte de MySQL, hay adaptadores de bases de datos para otras producciones de bases de datos, incluyendo PostreSQL, SQLite, Microsoft SQL Server, Oracle, etc.

Afortunadamente, Rails hace la selección de un proveedor de bases de datos, casi en último momento, porque el registro activo no expone directamente a la base de datos.

RMagick

Es una librería que ofrece la misma interfaz para ImageMagick y GraphicsMagick, por lo tanto es indiferente cual sea utilizado. Se puede obtener RMagick instalando RMagick o RMagick-win32 gem. RMagick permite modificar y realizar efectos en imágenes desde Ruby, y por supuesto desde Ruby on Rails. Gracias a Rmagick se realizan una multitud de transformaciones a imágenes [19].

Algunas funcionalidades que provee:

- Permite disminuir el tamaño de una imagen, mediante el uso de alguno de cuatro métodos: Cambio de tamaño (resize), escala (scale), muestra (sample) y miniatura (thumbnail).
- Permite agregar texto a una imagen.
- Cambiar el formato de una imagen.
- Convertir un grupo de datos en gráficos.
- Encriptamiento de datos.

ImageMagick

Aplicación que sirve para crear, editar y componer imágenes. Puede leer, convertir y guardar imágenes en una gran variedad de formatos [20].

Características

- Puede recortar, girar cambiar de color las imágenes o incluso combinarlas con otras.
- Permite aplicar varios efectos y agregar a las imágenes de texto, líneas, polígonos, elipses, entre otros.
- Es software libre, el cual proporciona el código fuente completo que puede ser libremente usado, copiado, modificado y distribuido.
- Es compatible con varios lenguajes de programación.
- Soporta muchos formatos de imágenes (más de 100) incluyendo formatos como: GIF, JPEG, etc.

Capacidades

- Conversión entre formatos.
- Transformaciones (cambiar de tamaño, girar y recortar una imagen).
- Trasparencias (determinar partes de una imagen como invisible).
- Dibujar (agregar formas, figuras y texto a una imagen).
- Decorar (agregar un marco o borde a una imagen).
- Efectos especiales (difuminar, esmerilar o trazar una imagen).
- Texto y comentarios (insertar una descripción o un texto artístico a una imagen).
- Identificación de imagen (describe los atributos de formato de una imagen).
- Animación (crea una animación GIF a partir de un grupo de imágenes).
- Composiciones (sobrepone una imagen sobre otra).

Herramientas tecnológicas del lado del cliente

HTML

Lenguaje utilizado para definir las páginas Web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página Web. El HTML es un lenguaje de marcación de elementos para la creación de documentos hipertexto. *HTML* es un conjunto de símbolos o palabras que definen varios componentes de un documento *Web*; estos se pueden ver siempre dentro de las etiquetas "<", ">". HiperText Markup Languaje es el nombre que estas siglas representan, creado por Tim Berners-Lee en 1991 [21].

CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos [22].

CSS se utiliza para dar estilo a documentos HTML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

Javascript

JavaScript es un lenguaje de scripts, interpretado, multiplataforma y parcialmente orientado a objetos. El código JavaScript puede enlazarse o añadirse a las páginas Web proporcionando un control total y dinámico sobre ellas. Además, también permite controlar (hasta cierto punto) las aplicaciones que lo ejecutan, habitualmente navegadores [23].

JavaScript se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. JavaScript ha tenido influencia de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación Java, aunque más fácil de utilizar para personas que no programan.

JQuery

Es una libreria liviana de JavaScript, pensada para interactuar con los elementos de una Web por medio del DOM2(Document Object Model es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos). Lo que la hace tan especial es su sencillez y su reducido tamaño. Permite simplificar la manera de interactuar con los documentos **HTML**, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología **AJAX** a nuestras páginas Web [24].

Ajax

Es una técnica de desarrollo Web para crear aplicaciones interactivas. Estas aplicaciones son ejecutadas del lado del cliente, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano, permitiendo realizar

cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es asíncrona, debido a que los datos adicionales son solicitados al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest[3], objeto disponible en los navegadores actuales (ver Figura 2.13).

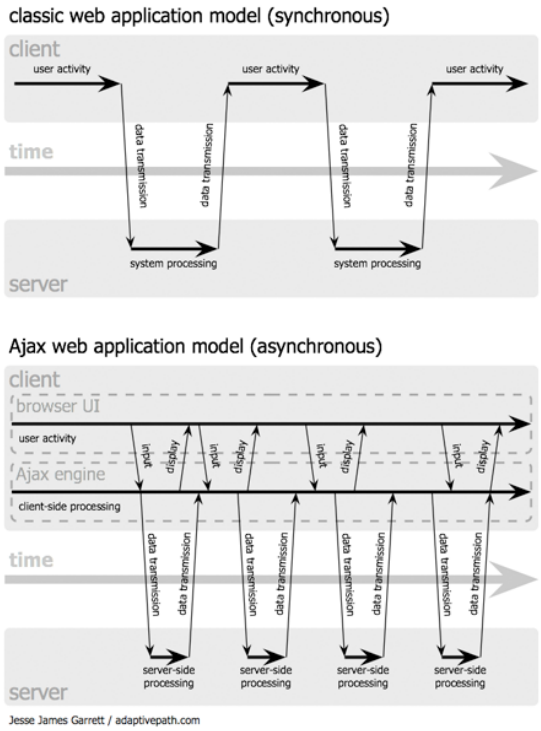


Figura 2.13 – Modelo Ajax

[3] XMLHttpRequest interfaz empleada para realizar peticiones HTTP, el uso más popular es proporcionar contenido dinámico y actualizaciones asíncronas en páginas Web.

2.4.2 Herramientas tecnológicas para aplicación local

Java

Java normalmente se refiere a la combinación de tres cosas: el lenguaje de programación Java (un lenguaje de programación orientado a objetos y de alto nivel); la máquina virtual de Java (una máquina virtual de alto rendimiento que ejecuta el bytecode en una plataforma de ordenador específica; normalmente abreviada JVM); y la plataforma Java, una JVM que ejecuta el bytecode compilado de Java, normalmente llamando a un conjunto de librerías estándar como las proporcionadas por Java Standard Edition (SE) o Enterprise Edition (EE) [25].

2.5 Programación Extrema XP

Es una metodología ágil basada en la construcción de software de buena calidad de la forma más rápida posible. La programación extrema se fundamenta en la producción de software con una fuerte arquitectura, intentando culminarlos rápidamente con gran calidad y motivando al equipo de trabajo para seguir mejorando esta tendencia [26].

2.5.1 Tareas principales de la Programación Extrema

- Codificar: Significa que, todo el software se produce mediante la elaboración de pequeñas versiones incrementales de producción corta.

- Probar: Se debe asegurar que todo lo que se hace funcione correctamente. Para ello, lo mejor es desarrollar la prueba desde el momento que se conocen los casos de uso (o, según XP, las historias del usuario).

- Diseñar: El diseño también debe ser incremental y debe estar reflejado en el software, lo cual quiere decir que la estructura de éste debe ser clara. Hay que diseñar lo que las necesidades del problema requieren.

- Planificar: La actividad de planificación comienza creando una serie de Historias de Usuario, en las cuales se describen los requerimientos del sistema en terminología del cliente, proporcionando a su vez una estimación del tiempo necesario para el desarrollo de dicha iteración (Ver tabla 2.1).

Fecha Inicio: Fecha Fin:

Número	Fecha	Descripción

Tabla 2.1 – Historias de usuarios

- Iteraciones: El método XP propone dividir el trabajo en iteraciones, las cuales se enfocan en versiones parciales del sistema hasta llegar al producto final. Los nuevos requerimientos son recibidos progresivamente y son incluidos a una nueva iteración. (ver Figura 2.14)

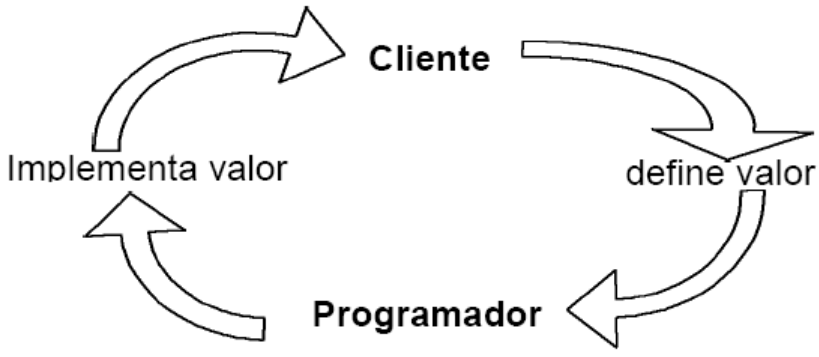


Figura 2.14 - Ciclo XP

2.5.2 Ventajas

- Se consiguen productos usables con mayor rapidez.
- El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo. Se consigue integrar todo el trabajo con mucha mayor facilidad.
- Se atienden las necesidades del usuario con mayor exactitud. Esto se consigue gracias a las continuas versiones que se ofrecen al usuario.
- Se consiguen productos más fiables y robustos contra los fallos gracias al diseño de los test de forma previa a la codificación.
- Se obtiene código más simple y más fácil de entender, reduciendo el número de errores.
- Gracias al "refactoring" es más fácil el modificar los requerimientos del usuario.

Parte III

Marco Aplicativo

Capítulo 3

Marco Aplicativo

La finalidad de este capítulo es describir cada una de las iteraciones del proceso XP durante la construcción del sistema de procesamiento de imágenes, donde cada una de ellas será especificada siguiendo las tareas fundamentales de la metodología ágil XP, las cuales son: planificación, diseño, codificación y prueba [26].

3.1 Iteración 0

- **Planificación**

En la Tabla 3.1 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 15/12/2009 Fecha Fin: 23/12/2009

Número	Fecha	Descripción
1.0	15/12/2009	Desarrollo del empaquetador de imágenes.

Tabla 3.1 – Planificación Iteración 0

- **Diseño**

Para complementar el empaquetador se diseñó una interfaz de usuario, donde se provee de un conjunto de opciones donde el usuario puede seleccionar para generar el paquete y mantener la retroalimentación con el usuario mediante un cuadro de dialogo que indica si los campos no han sido rellenados correctamente (Ver Figura 3.1), para disminuir los errores de usuarios se proveen de opciones de tipo selección, en el caso de la escuela y tipo de oficios (ver Figura 3.2). En el caso de la opción de la ruta de la carpeta donde se encuentran alojadas las imágenes, se provee de una opción tipo archivo donde la interfaz muestra una ventana para seleccionar la carpeta correspondiente (Ver Figura 3.3).

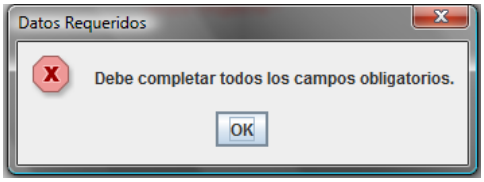


Figura 3.1 – Ventana de Error

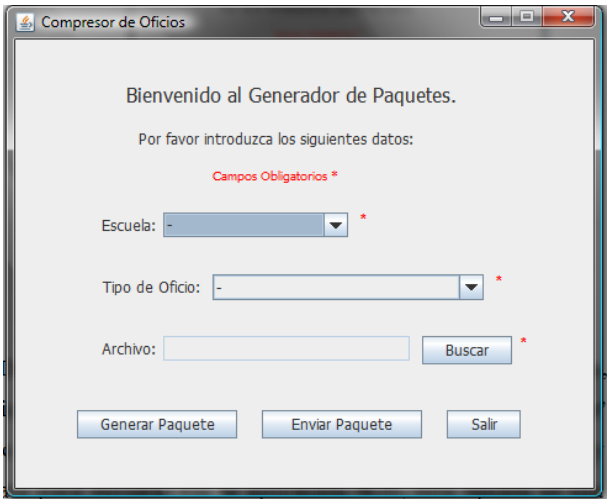


Figura 3.2 – Vista Principal Empaquetador

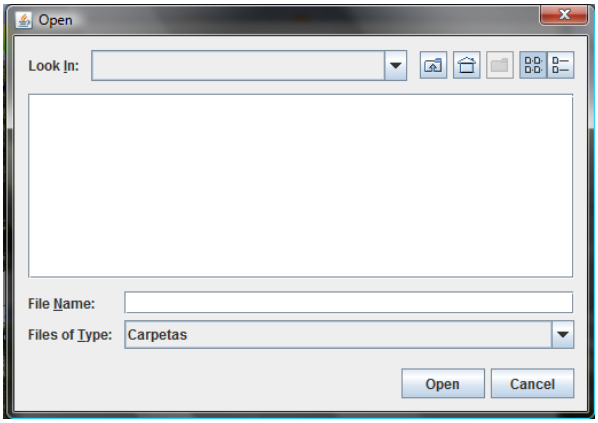


Figura 3.3 - Ventana para Seleccionar Carpeta

- **Codificación**

En esta fase se procede a desarrollar el módulo de empaquetamiento de imágenes, donde se procede a tomar los datos provenientes de las opciones seleccionadas por el usuario, en la vista principal y se genera el paquete.

Primeramente se genera un archivo Meta-data, donde se describe la escuela a la que pertenecen los oficios, tipo de oficio, nombre de la carpeta, número de imágenes y el nombre de cada una de las imágenes. Este archivo es almacenado en la carpeta donde se encuentran las imágenes y es eliminado una vez el paquete haya sido generado.

Seguidamente se procede a generar un archivo zip[4] donde se encuentran cada una de las imágenes y el archivo Meta-data (Ver figura 3.4).

```
// Comprime los archivos
for (int i=0; i<nombres_archivos.length; i++) {

        FileInputStream archivo = new FileInputStream(ruta_archivo+"\\"+nombres_archivos[i] );
        origen = new BufferedInputStream( archivo, BUFFER_TAMANO );
        // Configura la entrada en el archivo .zip
        ZipEntry entrada = new ZipEntry(nombres_archivos[i]);
        salida.putNextEntry(entrada);
        // Lee los datos del archivo fuente y escribe en el archivo zip
        int tamano;
        while( ( tamano = origen.read(buf, 0, BUFFER_TAMANO ) ) != -1 )
                salida.write(buf, 0, tamano);

        // Cierra el archivo fuente
        origen.close();
}
        // Cierra el archivo zip
salida.close();
```

Figura 3.4 – Código para Generar Paquete de Imágenes

- **Prueba:**

Las pruebas se basaron en pruebas unitarias donde se generaron diferentes paquetes, utilizando rutas aleatorias de carpetas y cada una de las opciones de escuelas y tipos de oficios.

[4]zip formato de almacenamiento sin pérdida, utilizado para la compresión de datos.

3.2 Iteración 1

- **Planificación**

En la Tabla 3.2 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 26/12/2009 Fecha Fin: 30/12/2009

Número	Fecha	Descripción
1.1	26/12/2009	Envío de paquetes al servidor.

Tabla 3.2 – Planificación Iteración 1

- **Diseño**

Las interfaces utilizadas en esta iteración son las mismas utilizadas en la iteración previa, pero se incluyo una ventana donde se la notifica al usuario cuando el proceso de empaquetamiento y envío de imágenes ha sido realizado (ver figura 3.5).

Figura 3.5 – Mensaje de Confirmación

- **Codificación**

En esta fase se procede a desarrollar el módulo de transferencia de datos al servidor, para ello se realizan dos tipos de transferencias (ver Figura 3.6), que serán explicadas a continuación:

- El primero es mediante el protocolo FTP, donde se envía el paquete zip al servidor.

- El segundo se realiza mediante el protocolo HTTP donde se le inicia la conexión con la aplicación Web, aquí se le indica donde se encuentra el archivo localmente y se envía un mensaje confirmación cuando el procesamiento de las imágenes haya sido completada por parte de la aplicación Web.

```
// transferencia de paquete al servidor
upload(ftpSERVER, "admin", "admin", file, paquete);

qparams.add(new BasicNameValuePair("archivo", rutaFTP + "\\" + file));


try {
     UrlEncodedFormEntity entity = new UrlEncodedFormEntity(qparams, "UTF-8");
     HttpPost httppost = new HttpPost("http://" + rubySERVER + "/descompresor/upload");

     httppost.setEntity(entity);
     System.out.println(httppost.getURI());

     HttpClient httpclient = new DefaultHttpClient();

     HttpResponse response = httpclient.execute(httppost);
     System.out.println(response.getEntity());

     if (entity != null) {
         long len = entity.getContentLength();
             if (len != -1 && len < 2048) {
                 System.out.println(EntityUtils.toString(entity));
             } else {

             }
     }

     if (paquete.delete()) System.out.println("Archivo Borrado");
```

Figura 3.6 – Código para Transferencia de Datos

- **Prueba**

Las pruebas se basaron en pruebas unitarias, donde se procedió al envío de paquetes provenientes de diferentes computadoras al servidor, cabe destacar que el envió de paquetes solo es posible en la misma red.

3.3 Iteración 2

- **Planificación**

En la Tabla 3.3 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 02/01/2010 Fecha Fin: 30/01/2010

Número	Fecha	Descripción
1.2	02/01/2010	Recepción de paquetes en el servidor
1.3	03/01/2010	Recepción de hipertexto y procesamiento de paquete.
1.4	15/01/2010	Almacenamiento de datos en Base de Datos
1.5	28/01/2010	Emisión de oficios en formato PDF

Tabla 3.3 – Planificación Iteración 2

- **Codificación**

En la presente iteración se procedió al procesamiento de datos provenientes del cliente, como se mencionó previamente existen dos tipos de transferencias, las cuales serán explicadas a continuación:

- Primeramente se recibe el paquete por medio del uso de un servidor FTP, específicamente FileZilla Server, este fue instalado y configurado bajo un nombre de usuario, clave y una carpeta donde serán almacenados los paquetes y sus respectivas permisologias, las cuales son: escritura y lectura (ver Figura 3.7).

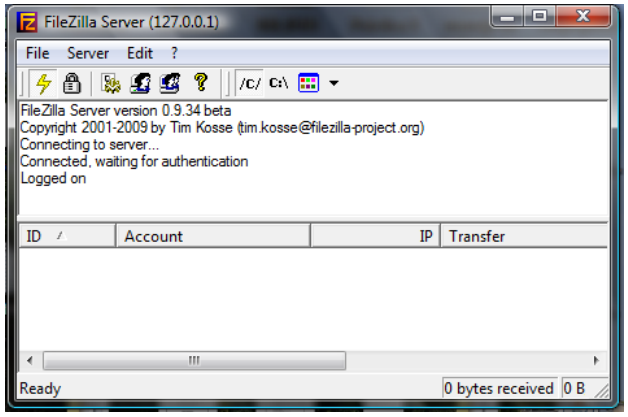


Figura 3.7 – Servidor FileZilla

- Seguidamente se procede a la recepción por parte de la aplicación Web de la dirección local del paquete, donde se procede a su procesamiento.

    - Primeramente se descomprime el archivo zip, en la carpeta public/images de la aplicación (ver Figura 3.8).

```
# crear el paquete .zip en el servidor
File.open(path, "wb") { |f| f.write(datos.read)}

#descomprime y almacena las imagenes en el servidor (en la carpeta de imagenes)
Zip::ZipFile.open(path).each do |archivo|

    archivo.extract(archivo.name)

    # verifica si el archivo es el de la metadata, en ese caso lo almacena en la carpeta de imagenes en caso contrario
    # cada archivo es almacenado en una carpeta que lleva el mismo nombre del archivo
    if archivo.name == 'Metadata.txt'

        dato = File.join(directorio, archivo.name)
        File.move(archivo.name, dato)

    else

        nombreArchivo = archivo.name
        extensionArchivo = nombreArchivo.slice(nombreArchivo.rindex("."), nombreArchivo.length).downcase
        nombreArchivo = nombreArchivo[0, nombreArchivo.length - extensionArchivo.length]
        rutaArchivo = rutaCarpeta + "\\" + nombreArchivo

        #verifica si el archivo existe dentro de la carpeta si no existe lo crea
        #en el caso de que exista arroja un mensaje de error
        if File.directory?(rutaArchivo)

        File.delete(archivo.name)
        File.delete(path)
        flash[:error] = "El archivo " + archivo.name.to_s + " ya existe"
        redirect_to :controller=> 'cargar', :action=> 'index'
        return false
```

Figura 3.8 – Código para Procesamiento de Paquetes

- Una vez que la carpeta ha sido creada se procede al almacenamiento de la información almacenada en el archivo de Meta-data referente a cada una de las imágenes, donde destacan, carpeta, escuela, tipo de oficios, ruta de la imagen, entre otros (ver Figura 3.9).

```
# Lee el nombre de la imagenes
numeroImagenes.times do

    metadata = f.readline
    metadata = metadata.delete "\n"

    # Crea registro de la imagen en bd
    nombreImagen = metadata.to_s
    imagenId = fecha.to_s + nombreImagen
    imagenId = Digest::MD5.hexdigest(imagenId)
    extensionArchivo = nombreImagen.slice(nombreImagen.rindex("."), nombreImagen.length)
    ruta = nombreCarpeta + "\\" + nombreImagen[0,nombreImagen.length-extensionArchivo.length]+"\\"+ nombreImagen
    Imagen.crear(imagenId, version, padre, fecha, ruta)
    final = "_1"+extensionArchivo.to_s

    # Verifica si es o no anexo
    if (nombreImagen[nombreImagen.length-6, nombreImagen.length] == final  || nombreImagen.count("_") < 2)

        oficios = oficios.push(imagenId)
        nombreImagen = nombreImagen[0, nombreImagen.length-extensionArchivo.length]
```

Figura 3.9 – Código para Almacenamiento de Imágenes

- Por último se procede a la emisión de archivo PDF, donde se encuentra el oficio y cada uno de sus anexos, una vez creado se procede al almacenamiento de Meta-data del oficio, donde se encuentra información que referencia a cada oficio, por ejemplo: la ruta del archivo PDF (ver Figura 3.10).

```
oficios.each do |o|

    imagen = Imagen.find(:first, :conditions => ["id=?", o.to_s])
    ids = Metadatanexo.find(:all, :conditions => ["id_imagen_principal=?", o.to_s], :order => "nombre DESC").collect{|img| img.image_id}
    imagenes = Imagen.find(:all, :conditions => ["id IN (?)", ids])

    pdf = PDF::Writer.new(:paper => "LETTER")
    pdf.margins_pt(5,5,5,5)
    pdf.add_image_from_file "public\\images\\" + imagen.ruta.to_s, 0, 0, 600, 800, nil

    if imagenes && !(imagenes.empty?)
        cont = 1
        pdf.start_new_page

        imagenes.each do |i|
            pdf.add_image_from_file "public\\images\\" + i.ruta.to_s, 0, 0, 600, 800, nil
            if cont < imagenes.length
                pdf.start_new_page
            end
            cont +=1
        end
    end
```

Figura 3.10 – Código para Emisión de Oficios en formato PDF

- **Prueba**

Las pruebas para esta iteración se basaron en la recepción de datos y el procesamiento de los mismos, verificación de los datos almacenada en la base de datos y en el servidor, específicamente en la carpeta pública de imágenes.

3.4 Iteración 3

- **Planificación**

En la Tabla 3.4 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 31/01/2010 Fecha Fin: 28/02/2010

Número	Fecha	Descripción
2.0	31/02/2010	Carga de oficios y opciones de procesamiento
2.1	15/02/2010	Generación de opciones de procesamiento y almacenamiento de datos

Tabla 3.4 – Planificación Iteración 3

- **Diseño**

El diseño de esta iteración se baso en la simplicidad y claridad para el usuario, es por ello que se planteo mostrar organizadamente las carpetas, oficios de cada carpeta que aun no han sido procesados (ver Figura 3.11 y Figura 3.12).

Nombre	Escuela	Tipo	Fecha	Opcion
Z3	Biologia	Modificaciones y aperturas	2010-06-06 13:21:03	Procesar datos
Z2	Biologia	Modificaciones y aperturas	2010-06-01 12:33:48	Procesar datos
Z1	Biologia	Modificaciones y aperturas	2010-06-01 12:32:55	Procesar datos

Figura 3.11 – Carpetas Pendientes

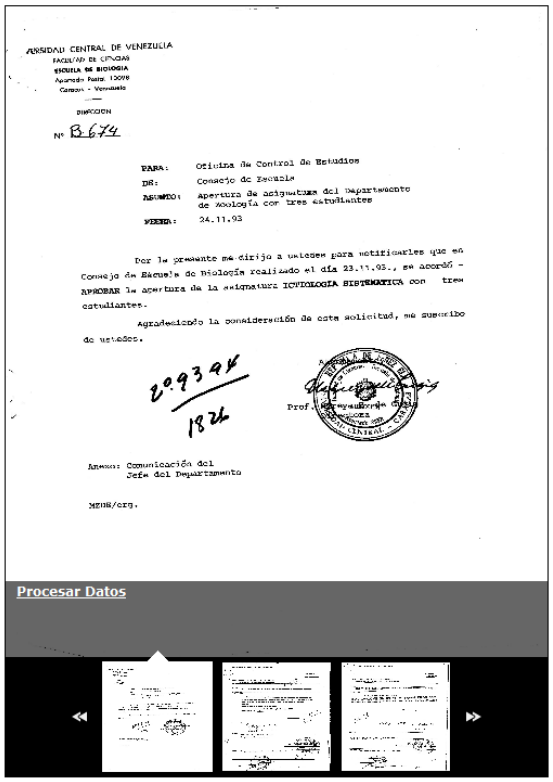


Figura 3.12 - Oficios Pendientes

En el caso del procesamiento de oficios se procedió a mostrar el oficio a procesar junto a las opciones de procesamiento (ver Figura 3.13 (a) y Figura 3.13 (b)).

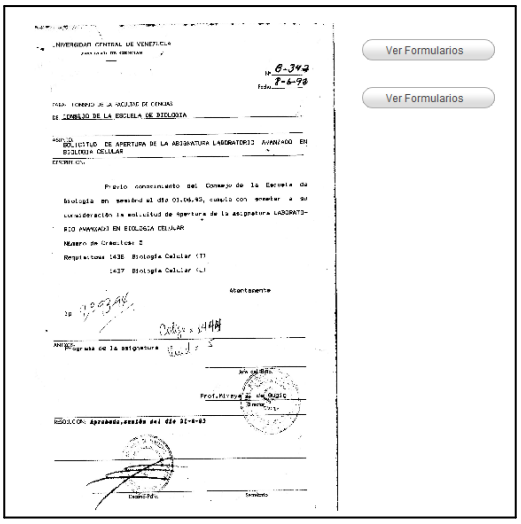


Figura 3.13 (a) – Procesamiento de Oficios

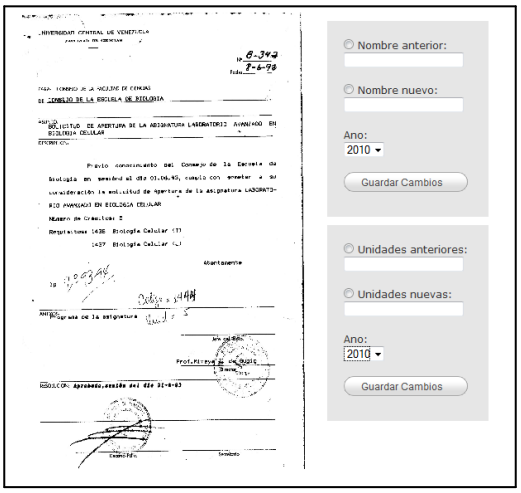


Figura 3.13 (b) – Procesamiento de Oficios

- **Codificación**

En esta iteración se procede a buscar los oficios que aun no han sido procesados para listarlos, primeramente se listan las carpetas que aún tienen oficios por procesar (ver Figura 3.14), al seleccionar una de las carpetas se listaran los oficios de dicha carpeta que aun no ha sido procesados, donde se mostrará la última versión

del oficio (ver Figura 3.15), generada por el procesamiento de imágenes, este módulo será explicado posteriormente, y por último se carga el oficio seleccionado junto a sus opciones, las cuales varían dependiendo del tipo de oficio (ver Figura 3.16).

```
#Funcionalidad dedicada a listar todas las carpetas que aun no han sido procesadas con el proceso OCR
def carpetas_sin_procesar
    @tipo = params[:tipo]
    pagina = params[:page]
    @carpetas = Carpeta.paginate(:per_page => 20, :page => pagina, :order => 'fechacarga DESC', :conditions => ["procesada = 'no'"])
    @menuPrincipal = obtener_menu_principal
    @menuSecundario = obtener_menu_secundario
end
```

Figura 3.14 - Código para Listar Carpetas Pendientes

```
imagenesId = Metadata.find(:all, :conditions => ["procesada='no' AND carpeta_id=?", id.to_s]).collect{|m| m.image_id}
@imagenes = listar_imagenes(imagenesId, "datos")
```

Figura 3.15 - Código para Listar Oficios sin Procesar

```
#Verifica si la imagen es la original
if padre
    @padre = padre
    session[:padre] = padre
end

@tipo = tipo
@imagen = Imagen.find(:first, :conditions=>["id=?",id])
@metadata = Metadata.find(:first, :conditions => ["image_id=?", padre])
```

Figura 3.16 – Código para Carga de Oficio y Opciones de Procesamiento

Para el procesamiento de datos, el usuario tiene la opción de generar los formularios dinámicamente, en el caso de oficios de tipo modificación este tendrá la posibilidad de indicar el número de modificaciones y seleccionar entre: cambio de nombre, código, unidades de créditos y requisitos, en el caso de oficios de tipo apertura el usuario tendrá la posibilidad de definir el número de aperturas.

Una vez generados los formularios el usuario puede hacer uso del proceso OCR, donde se sombrea una parte de la imágenes utilizando jQuery, específicamente Jcrop, el cual permite tomar las coordenadas de una imagen, una vez se tienen las coordenadas se procede a cortar y guardar la imagen en formato tif y bpm usando RMagick y se realiza el OCR (ver Figura 3.17).

```
imagen = Image.read("public\\images\\"+ruta.to_s).first
dato = imagen.crop!(x.to_f,y.to_f,ancho.to_f,largo.to_f)
dato.write("public\\images\\ocr.tif")

system("tesseract public\\images\\ocr.tif output -l spa")

File.delete("public\\images\\ocr.tif")

archivo = "output.txt"

f = File.open(archivo)
ocr = f.readline
@ocr = ocr.delete "\n"
f.close
File.delete(archivo)

if !@ocr
    dato.write("public\\images\\ocr.bpm")
    @ocr = %x{#{"OCRAD public\\images\\ocr.pbm"}}
    if !@ocr
        @ocr = %x{#{"GOCR public\\images\\ocr.pbm"}}
        File.delete("public\\images\\ocr.pbm")
    else
        File.delete("public\\images\\ocr.pbm")
    end
end
```

Figura 3.17 – Código para Proceso OCR

De igual forma, el usuario tiene la posibilidad de escribir los datos en cada uno de los campos de textos sin necesidad de utilizar el proceso OCR.

Para el almacenamiento de los datos, se utiliza la tecnología Ajax, donde se envían los datos de cada formulario y se procede a verificar si cumplen con los requisitos, en que caso en que cumplan con los requisitos estos serán almacenados en la base de datos y se verifica si aún quedan datos pendientes por procesar del oficio, en caso en que el oficio tenga sus datos completos se marca como procesado (ver Figura 3.18) y se verifica si todos los oficios pertenecientes a la carpeta han sido procesados de ser así se marca la carpeta como procesada (ver Figura 3.19) para que no sea referenciada cuando se listen las carpetas pendientes.

```
#Verifica si todos los datos del oficio han sido procesados
def imagen_procesada(id, tipo)

   #Verifica el tipo de oficio
   if tipo.to_s == 'modificacion'
       metadata = Metadata.find(:first, :conditions => ["image_id=?", id.to_s])
       modificaciones = Modificacion.count(:id, :conditions => ["procesada='si' AND imagen_id=?", id.to_s])

       if metadata.modificaciones.to_s == modificaciones.to_s
           return true
       else
           return false
       end
       .
       .
       .
```

Figura 3.18 – Código para Verificación de Oficio Procesado

```
#Verifica si todas las imagenes que pertenecen a una carpeta han sido procesadas
def carpeta_procesada(id, tipo)

   #Verifica tipo de oficio
   if tipo.to_s == 'modificacion'

       metadata = Metadata.find(:first, :conditions => ["image_id=?", id.to_s])
       carpeta = metadata.carpeta_id.to_s
       imagenes = Metadata.count(:id, :conditions => ["carpeta_id=?", carpeta])
       procesadas = Metadatamodificacion.count(:id, :conditions => ["procesada='si' AND carpeta_id=?", carpeta])

       if imagenes.to_s == procesadas.to_s
           return true
       else
           return false
       end
       .
       .
       .
```

Figura 3.19 – Código para Verificación de Carpeta Procesada

- **Prueba:**

Se realizaron pruebas unitarias en cada uno de los pasos, listado de carpetas, listado de oficios por carpetas, carga de oficios, opciones de procesamiento de datos, y almacenamiento de los mismos.

3.5 Iteración 4

- **Planificación**

En la Tabla 3.5 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 01/03/2010 Fecha Fin: 20/03/2010

Número	Fecha	Descripción
3.0	01/03/2010	Carga de imágenes y opciones de procesamiento
3.1	15/03/2010	Almacenamiento de imágenes

Tabla 3.5 – Planificación Iteración 4

- **Diseño**

En esta iteración se presentan todas las carpetas almacenadas, donde se permite acceder a cada una de ellas, la interfaz es parecida al listado de carpetas del procesamiento de datos (ver Figura 3.20). Una vez seleccionada la carpeta se procede a mostrar todos los oficios relacionados con dicha carpeta, nuevamente la interfaz es parecida a la interfaz utilizada para listar los oficios sin procesar (ver Figura 3.21), una vez seleccionado un oficio en particular se procede a listar todos los anexos de dicho oficios incluyéndolo, donde finalmente se permite seleccionar uno de ellos para su procesamiento.

En cuanto al procesamiento de los oficios se provee una interfaz donde se muestran cada uno de los filtros para el procesamiento digital de imágenes (ver Figura 3.22).

Nombre	Escuela	Tipo	Fecha	Opcion
Z3	Biologia	Modificaciones y aperturas	2010-06-06 13:21:03	Procesar imagenes
Z2	Biologia	Modificaciones y aperturas	2010-06-01 12:33:48	Procesar imagenes
Z1	Biologia	Modificaciones y aperturas	2010-06-01 12:32:55	Procesar imagenes

Figura 3.20 – Carpetas a Procesar

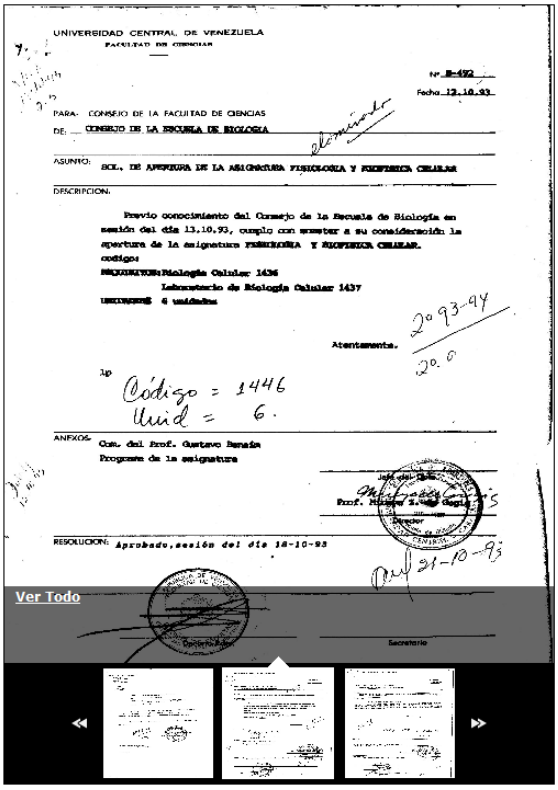


Figura 3.21 - Oficio a Procesar

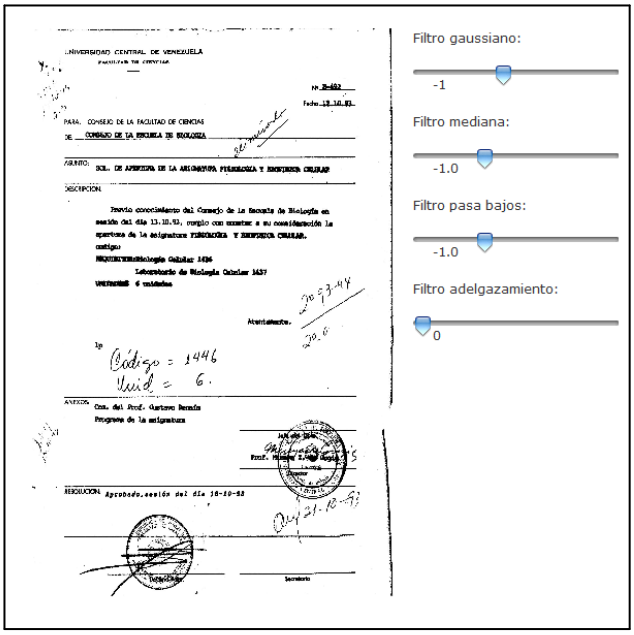


Figura 3.22 – Procesamiento de Imágenes

- **Codificación**

Para esta iteración se reutilizó parte del código utilizado en la iteración previa para listar las carpetas, imágenes y presentación de imagen detallada con las opciones de procesamiento digital de imágenes.

En el caso de procesamiento digital de imágenes de utiliza la tecnología JQuery nuevamente, específicamente slider, para la opción de valores de filtros para las imágenes, una vez se tiene el valor del filtro deseado se procede a enviar los valores, por medio de Ajax, al servidor para que este modifique la imagen dependiendo de los parámetros (ver Figura 3.23) y crea una imagen auxiliar. Una vez se realice el filtro se muestra la nueva imagen y se provee al usuario de dos opciones: restablecer y guardar (ver Figura 3.24), donde restablecer se encargara de eliminar la imagen auxiliar y cargar la imagen original (ver Figura 3.25), y guardar cambia el nombre de la imagen auxiliar con el nombre de la imagen original y la versión correspondiente (ver Figura 3.26) y almacena los cambios en la base de datos.

```
id = params[:id]
filtro = params[:filtro]
ruta = params[:ruta]
padre = session[:padre]

img = Image.read("public\\images\\"+ruta.to_s).first

if filtro.to_s == 'gaussiano'

    valor = params[:valor]
    img = img.gaussian_blur(1.0,valor.to_f)
else

    if filtro.to_s == 'mediana'

        valor = params[:valor2]
        img = img.median_filter(valor.to_f)
    else

        if filtro.to_s == 'pasaBajos'

            valor = params[:valor3]
            img = img.level(0,Magick::MaxRGB,valor.to_f)
        else

            if filtro.to_s == 'adelgazamiento'
```

Figura 3.23 – Código para el Procesamiento Digital de Imágenes

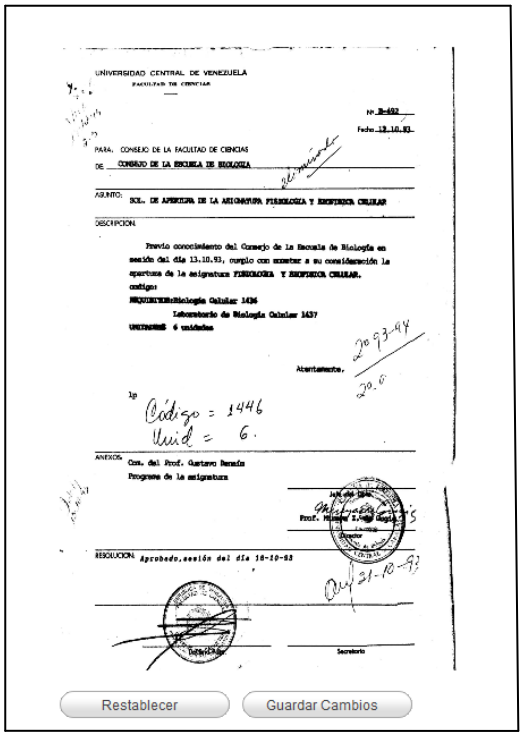


Figura 3.24 – Opciones para el Procesamiento Digital de Imágenes

```
def restablecer

    id = params[:id]
    @id = id
    ruta = params[:ruta]

    File.delete("public\\images\\"+ruta.to_s)
    @imagen = Imagen.find(:first, :conditions=>["id=?",id])

end
```

Figura 3.25 – Código para Restablecer una Imagen

```
#Verifica si es la imagen original o si es una version
if padre

    imagen = Imagen.find(:first, :conditions => ["id=?", id.to_s])
    version = imagen.version.to_s
    imagenPadre = Imagen.find(:first, :conditions => ["id=?", padre.to_s])
    rutaPadre = imagenPadre.ruta.to_s
    version = version.to_i
    version = version + 1

else

    version = 2
    padre = id
    session[:padre] = id
    imagenPadre = Imagen.find(:first, :conditions => ["id=?", id.to_s])
    rutaPadre = imagenPadre.ruta.to_s

end

img = Image.read("public\\images\\"+ruta.to_s).first
extension = rutaPadre.slice(rutaPadre.rindex("."), rutaPadre.length)
rutaPadre = rutaPadre[0,rutaPadre.length-extension.length]
rutaVersion = rutaPadre + "_version_" + version.to_s + extension
```

Figura 3.26 – Código para Guardar una Imagen

- **Prueba:**

Al igual que las iteraciones previas, las pruebas se basaron en pruebas unitarias en cada uno de las opciones de filtros para el procesamiento digital de imágenes y almacenamiento de los cambios.

3.6 Iteración 5

- **Planificación:**

En la Tabla 3.6 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 21/03/2010 Fecha Fin: 10/04/2010

Número	Fecha	Descripción
4.0	21/03/2010	Carga de imágenes y opciones de consulta
4.1	31/03/2010	Descarga de PDF
4.2	02/04/2010	Cambio de imágenes

Tabla 3.6 – Planificación Iteración 5

- **Diseño**

En esta iteración nuevamente se reutilizaron los diseños para listar carpetas e imágenes, en el caso del detalle de cada oficio se provee de opciones de consulta, una de ellas fue elaborada usando JQuery, específicamente Zoom, el cual, permite visualizar la imagen mediante una herramienta de acercamiento (ver Figura 3.27), también se la de la opción a los usuarios de visualizar todas las versiones de imágenes almacenadas después del procesamiento digital de imágenes, en el caso de que se esté visualizando una versión diferente a la original se puede acceder a la versión original, de igual forma, los usuarios tienen la posibilidad de descargar los oficios en formato PDF junto a sus anexos y por último (ver Figura 3.28), en el caso de que usuario no esté contento con la calidad de la imagen tiene la posibilidad de cambiarla.

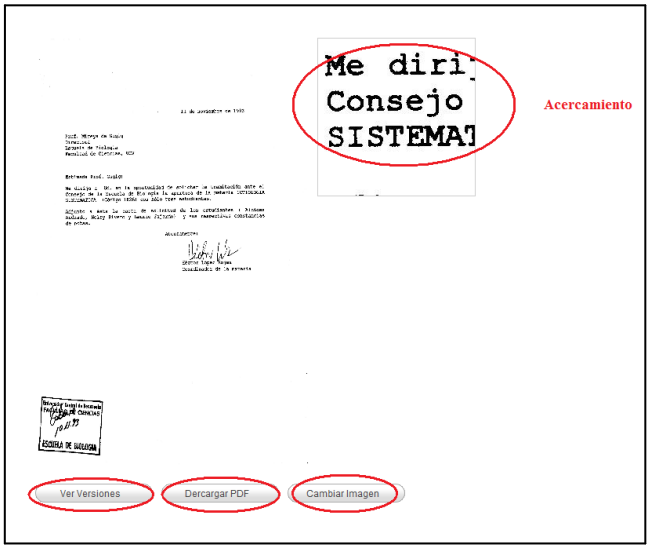


Figura 3.27 – Opciones de Consulta de Oficios

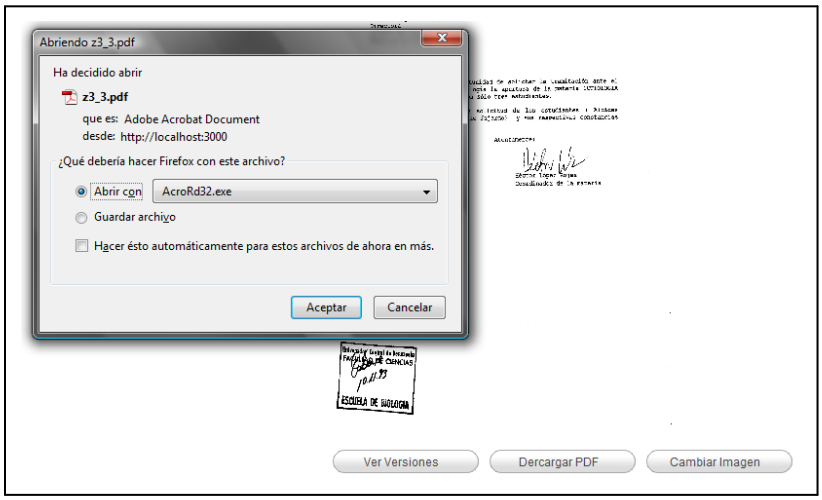


Figura 3.28 – Descarga de Oficio en Formato PDF

- **Codificación**

En esta iteración se utilizó nuevamente jQuery, para generar un acercamiento para las imágenes, donde se debe colocar el cursor sobre la imagen y se mostrará un acercamiento de la parte donde el cursor se encuentre en eso momento.

En el caso en que se esté visualizando una versión “hija” provee la opción de visualización de imagen original, donde se toma el identificador del padre de la imagen y se carga la imagen cuyo identificador coincida con el padre de la imagen “hija”.

De igual forma, se le provee la posibilidad al usuario de visualizar todas las versiones de la imagen, en el caso en que las tenga, para ello en el caso de que la imagen sea la original se toma el identificador de la imagen y se listan todas la imágenes que tengan como padre ese identificador, en el caso en que la imagen no sea la original se procede a listar todas las imágenes, que tengan el mismo padre que la imagen (ver Figura 3.29).

```
imagen = Imagen.find(:first, :conditions => ["id=?",id.to_s])

#Verifica si la imagen existe
if imagen
    padre = imagen.padre
    #Verifica si es la imagen original
    if padre.to_s == '0'
        imagenes = Imagen.find(:all, :conditions => ["padre=?", imagen.id.to_s])
        #verifica si la imagen original tiene versiones
        if imagenes
            @imagenes = imagenes
        end
    else
        @imagenes = Imagen.find(:all, :conditions => ["(padre=? OR id=?) AND !(id=?)", imagen.padre.to_s, imagen.padre.to_s, imagen.id.to_s])
    end
end
```

Figura 3.29 – Código para Mostrar Versiones de Imagen

Por otro lado, se permite la descarga del oficio en formato PDF, donde se procede a buscar dicho documento, mediante el identificador de la imagen, el cual se encuentra referenciado en la tabla donde se almacena la Meta-data del documento PDF (ver Figura 3.30).

```
def descargar_pdf
    id = params[:id]
    tipo = params[:tipo]
    principal = nil

    principal = Metadata.find(:first, :conditions => ["image_id=?",id.to_s])

    if principal
        @pdf = Oficio.find(:first, :conditions => ["imagen_principal_id=?", principal.image_id.to_s])
    else
        anexo = Metadatanexo.find(:first, :conditions => ["image_id=?", id.to_s])
        @pdf = Oficio.find(:first, :conditions => ["imagen_principal_id=?", anexo.id_imagen_principal.to_s])
    end

    send_file(@pdf.ruta, :type => 'application/pdf', :disposition => 'attachment')
end
```

Figura 3.30 – Código para Descargar Oficio en Formato PDF

Por último, se encuentra la opción de cambio de imagen, donde el usuario tiene la posibilidad de subir una imagen nueva al servidor, lo que se realiza es borrar todas las imágenes "hijas" de la base de datos y del disco y se borra la imagen original, seguidamente es sustituida por la nueva imagen (ver Figura 3.31).

```
imagenes = Imagen.find(:all, :conditions => ["padre=?", idp.to_s])

if imagenes
    imagenes.each do |i|
        id = i.id.to_s
        ruta = dir+i.ruta.to_s
        Imagen.delete(id)
        File.delete(ruta)
    end
end
nombre = imagen.original_filename
File.open(dir+nombre.to_s, "wb") { |f| f.write(imagen.read)}
extensionNueva = nombre.slice(nombre.rindex("."), nombre.length)

imagen = Imagen.find(:first, :conditions => ["id=?", idp])
id = imagen.id.to_s
ruta = dir+imagen.ruta.to_s
File.delete(ruta)

extension = ruta.slice(ruta.rindex("."), ruta.length)
ruta = ruta[0, ruta.length - extension.length]
ruta = ruta + extensionNueva
File.move(dir+nombre.to_s, ruta)
ruta = ruta[dir.length,ruta.length]
Imagen.update(id, {:ruta => ruta})
```

Figura 3.31 – Código para Cambiar Imagen

- **Prueba**

Se realizaron pruebas unitarias de cada una de las opciones, verificando que cada una cumpliera sus objetivos, probando con diferentes escenarios.

3.7 Iteración 6

- **Planificación**

En la Tabla 3.7 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 11/04/2010 Fecha Fin: 22/04/2010

Número	Fecha	Descripción
5.0	11/04/2010	Manejo de sesiones
5.2	13/04/2010	Consulta y modificación de datos de usuario
5.3	12/04/2010	Listar Usuarios y opciones
5.4	15/04/2010	Crear usuario
6.0	17/04/2010	Consultar bitácora
7.0	18/04/2010	Generar respaldo
7.1	20/04/2010	Descargar respaldo
8.0	21/04/2010	Descargar empaquetador

Tabla 3.7 – Planificación Iteración 6

- **Diseño**

Las opciones de administración de usuarios, así como las opciones generales serán cargadas dinámicamente dependiendo del tipo de usuario, se manejan tres tipos de usuarios, los cuales serán descritos a continuación:

- Usuario: los usuarios de tipo “usuario” tiene la mínima permisología dentro del sistema, sus funciones dentro del mismo se basan en la visualización de oficios y opciones de consulta y modificación de sus datos de usuario dentro del sistema (ver Figura 3.32).

- Administrador: los usuarios de tipo administrador tienen los mismos beneficios que los usuarios dentro del sistema, adicionalmente podrán procesar, cargar, consultar los oficios, así como también crear, listar y modificar a los usuarios de tipo “usuario” (ver Figura 3.33).

- Súper-administrador: los usuarios de tipo súper-administrador tienen los mismos beneficios que los administradores dentro del sistemas, adicionalmente podrán consultar, modificar y crear usuarios de tipo “usuario” y administrador, de igual forma podrán generar y descargar respaldos de la base de datos (ver Figura 3.34).

Figura 3.32 – Vista para Usuarios

Figura 3.33 – Vista para Administradores

Figura 3.34 – Vista para Súper-administrador

- **Codificación**

En esta iteración se creó el manejo de sesiones y el registro de las acciones del usuario dentro del sistema (bitácora), de igual forma las opciones para cada usuario dentro del sistema son cargadas dinámicamente una vez que el usuario este autenticado, dentro del mismo, los cuales son generados dependiendo del tipo usuarios (ver Figura 3.35).

```
def obtener_menu (rango)

    rol = session[:rol_id]
    items = Funcionalidadrole.find(:all,:conditions=> ["rol_id=? AND funcionalidad_id IN (?) ", rol, rango], :order => "funcionalidad_id ASC");
    menu = Array.new

   items.each do |i|

       id = i.funcionalidad_id
       item = Funcionalidade.find(:first, :conditions => ["id=?",id])
       menu = menu.push(item)

   end

   return menu
end
```

Figura 3.35 – Código para Generar Opciones de Usuarios Dinámicamente

- **Prueba**

Las pruebas se basaron en la verificación del manejo de usuarios, generación de menú principal y segundario, dependiendo de cada usuario, y registro de cada una de las actividades del usuario dentro del sistema.

3.8 Iteración 7

- **Planificación**

En la Tabla 3.8 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 23/04/2010 Fecha Fin: 10/05/2010

Número	Fecha	Descripción
9.0	23/04/2010	Módulo de búsquedas

Tabla 3.8 – Planificación Iteración 7

- **Diseño**

En esta iteración se proporciona al usuario una interfaz donde este podrá seleccionar diferentes opciones de búsqueda (ver Figura 3.36), una vez que se tengan los resultados se lista cada uno de ellos en forma de enlaces (ver figura 3.37), donde se podrá visualizar detalladamente cada una de los enlaces arrojados mediante la búsqueda (ver Figura 3.38).

Figura 3.36 – Vista para Opciones de Búsqueda

Figura 3.37 – Vista para Resultado de Búsqueda

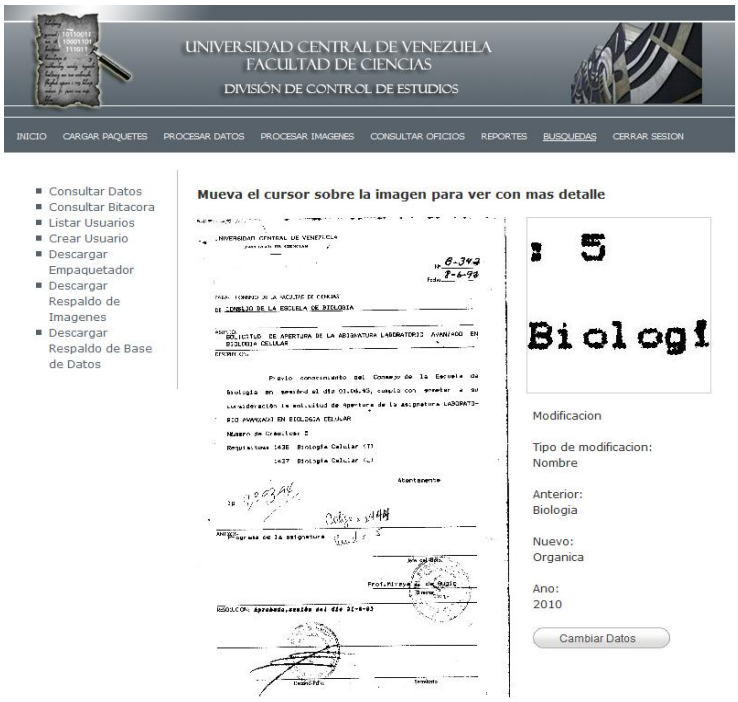


Figura 3.38 – Vista para Detalle de Búsqueda

- **Codificación**

Se provee de un campo de selección donde el usuario podrá elegir si desea buscar oficios o datos procesados, una vez que se tenga esta información se genera un nuevo formulario, el cual será cargado dinámicamente mediante el uso de Ajax, una vez que el usuario introduce sus opciones de búsquedas se procede a realizar la misma donde se verifican los datos y se procede a buscar en la base de datos las coincidencias de datos dentro de la misma (ver Figura 3.39).

```
if codigo.to_s != ""
    if ano.to_s != ""
        if nombre.to_s != ""
            #codigo, ano, nombre
            @modificaciones = Modificacion.find(:all, :conditions => ["anterior LIKE ? OR nuevo LIKE ? OR ano=? OR anterior LIKE ? OR nuevo LIKE ?",
            @aperturas = Apertura.find(:all, :conditions => ["codigo LIKE ? OR nombre_materia LIKE ? OR ano=?","%"+codigo.to_s+"%","%"+nombre.to_s+"
        else
            #codigo, ano
            @modificaciones = Modificacion.find(:all, :conditions => ["anterior LIKE ? OR nuevo LIKE ? OR ano=?", "%"+codigo.to_s+"%","%"+codigo.to_
            @aperturas = Apertura.find(:all, :conditions => ["codigo LIKE ? OR ano=?","%"+codigo.to_s+"%",ano.to_i])
        end
    else
        if nombre.to_s != ""
            #codigo, nombre
            @modificaciones = Modificacion.find(:all, :conditions => ["anterior LIKE ? OR nuevo LIKE ? OR anterior LIKE ? OR nuevo LIKE ?", "%"+codi
            @aperturas = Apertura.find(:all, :conditions => ["codigo LIKE ? OR nombre_materia LIKE ?","%"+codigo.to_s+"%","%"+nombre.to_s+"%"])
        else
            #codigo
            @modificaciones = Modificacion.find(:all, :conditions => ["anterior LIKE ? OR nuevo LIKE ?", "%"+codigo.to_s+"%","%"+codigo.to_s+"%"])
            @aperturas = Apertura.find(:all, :conditions => ["codigo LIKE ?","%"+codigo.to_s+"%"])
```

Figura 3.39 – Código para Búsquedas

- **Prueba**

Las pruebas se realizaron utilizando diferentes casos de búsquedas y verificando cada uno de sus resultados con los registros almacenados en la base de datos.

3.9 Iteración 8

- **Planificación**

En la Tabla 3.9 se muestran cada una de las historias de usuario con su respectiva fecha, realizadas en esta iteración.

Fecha Inicio: 10/05/2010 Fecha Fin: 20/05/2010

Número	Fecha	Descripción
10.0	10/04/2010	Módulo de reportes

Tabla 3.9 – Planificación Iteración 8

- **Diseño**

El usuario tiene la opción de elegir entre varias opciones donde puede seleccionar entre tipo de oficio, escuela y año (ver Figura 3.40), una vez se realice el reporte se carga un botón donde se da la opción para descargar el reporte. Una vez se presione el botón se genera una ventana donde se da la opción de descarga del reporte, una vez se descarga el reporte este podrá visualizarse.

Figura 3.40 – Vista para Opciones de Emisión de Reportes

- **Codificación**

Se toman los datos provenientes de las opciones elegidas por el usuario y se procede a realizar la búsqueda (ver Figura 3.41), una vez se tenga los datos se procede a realizar el documento PDF (ver Figura 3.42), donde se mostraran los resultados arrojados por la búsqueda.

```
carpetas = Carpeta.find(:all, :conditions => ["escuela_id=? AND tipoficio=?", escuela.to_s, tipo.to_s]).collect{|carpeta| carpeta.id
if !carpetas.empty?
    if tipo != 'No aplica'
        if tipo == 'modificacion'
            imagenesIdModificacion = Metadatamodificacion.find(:all, :conditions => ["carpeta_id IN (?)", carpetas]).collect{|id| id.image_id}
            modificaciones = Modificacion.find(:all, :conditions => ["ano=? AND imagen_id IN (?)",ano.to_s,imagenesIdModificacion])
            generar_pdf(modificaciones, nil, tipo, escuela, ano)
```

Figura 3.41 – Código para la Búsqueda de Emisión de Reportes

```
pdf = PDF::Writer.new(:paper => "A4")
pdf.compressed = true if RAILS_ENV != 'development'

if escuela
    escuela = Escuela.find(:first, :conditions => ["id=?", escuela.to_s])
    escuela = escuela.nombre
end

#encabezado
imagen = "index.jpg"
pdf.add_image_from_file "public/images/#{imagen}", 265, 730, 70, 70, nil
#pdf.add_image_from_file(image, x, y, width, height, link)
pdf.select_font "Times-Roman"


#UCV
#pdf.text "Centro" , :font_size => 14, :justification => :center
pdf.add_text(200, 710, "Universidad Central de Venezuela", 16)

#Ciencias
pdf.add_text(235, 693, "Facultad de Ciencias", 16)

#DCE
pdf.add_text(205, 676, "Division de Control de Estudios", 16)
```

Figura 3.42 – Código para la Emisión de Reportes en Formato PDF

- **Prueba**

Las pruebas se basaron en la emisión de reportes, donde se verificó que los datos arrojados por las búsquedas efectivamente coincidían con los registros almacenados en la base de datos, de igual forma se verificó que el formato del documento PDF cumpliera con los parámetros (coordenadas) de configuración para la emisión de los mismos.

Parte IV

Resultados, conclusiones y recomendaciones

Resultados

En la tabla 4.1 se puede observar una muestra de los resultados obtenidos tras el proceso OCR, utilizando cada uno de los filtros implementados para el apoyo de dicho proceso, planteados en la solución del Trabajo Especial de Grado.

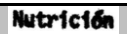
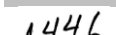
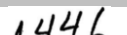
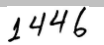
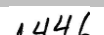
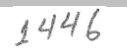
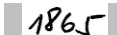
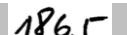
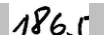
Img	Resul	%	Gauss	Resul	%	Med	Resul	%	Pasa Bajos	Resul	%	Adelga	Resul	%
	Nwtriciăn	77.7		Iutriciún	77.7		Ißtïićiû	44.4		Nßtvicičn	66.6		nulo	0
	Bioquímica	100		Bioquímica	100		Bioquímica	100		Bioquímica	100		ă äšm ăğšă	0
	ET OBOTAÑÏĆA	66.6		ETNOBOTAN I CA	100		ET OBOTAÑÏČA	66.6		ET OBOTAÑšĆA	66.6		Š ëêâ è ,	0
	IL/L/¿	0		1*/Å/g	25		IL/L/g	0		IL/L/¿	0		ğäfš ëřğ	
	FITOPATOIØGIA	84.6		FITOPATOLOGIA	100		FITOPATOIØGIA	84.6		FITOPATOIØGIA	84.6		à	0
	L843	75		1843	100		ltß	0		L843	75		ğüğš	0
	x]Yé—(‘	0		4Y6J	25		4Y6`(‘	25		x]Yé—(	0		Jîžšŷ èmğ w	0
Total		57.7			75.38			31.51			56.11			0

Tabla 4.1 – Resultados del Proceso OCR Mediante el Uso de Distintos Filtros

Conclusiones

El estudio de tecnologías actuales relacionadas con la digitalización de documentos en físicos, permitió el desarrollo del presente Trabajo Especial de Grado, el cual se baso en el desarrollo de un sistema de automatización del proceso de digitalización de oficios de aperturas y modificaciones de las materias de la Facultad de Ciencias de la Universidad Central de Venezuela.

El sistema cumple con los requerimientos planteados en este Trabajo Especial de Grado, así como también las modificaciones requeridas a los largo del desarrollo del mismo.

La metodología ágil XP facilitó considerablemente el desarrollo de la aplicación, gracias a la flexibilidad que esta provee para adaptarse a los cambios, así como la posibilidad mantenerse en contacto con los clientes constantemente, permitiendo la retroalimentación para el desarrollo del mismo. Una de las particularidades de este Trabajo Especial de Grado se basa en que no siguió todos los fundamentos de la metodología ágil, ya que el desarrollo en parejas no fue cumplido.

La elaboración del sistema de automatización fue más allá de solo procesar los datos provenientes de las imágenes escaneadas de oficios, si no que se presenta u conjunto de módulos para el mantenimiento y consulta de las imágenes, ayudando así a la perdurabilidad de los oficios originales almacenados en la División de Control de Estudios de la Facultad de Ciencias, los cuales representan un patrimonio histórico para dicha Facultad y la Universidad Central de Venezuela.

Adicionalmente, el tiempo de digitalización de oficios disminuyo, así como también el proceso de búsquedas de datos y oficios ya que este proceso también fue automatizado, otra de las ventajas de contar con un sistema automatizado es que este permite conocer las redundancias, ya que se observo que existían materias que habían

sido aperturadas varias veces con diferentes nombres y en algunos casos con el mismo nombre, pero como no se contaba con ninguna herramienta que pudiese controlar las entradas de dicho archivo, la redundancia fue aumentando.

En el caso de los filtros se verificó que el filtro de adelgazamiento no resultó como se esperaba, ya que una vez aplicado dicho filtro la imagen se torna incomprensible para el proceso OCR. El filtro que mejor resultados arrojo fue el filtro Gaussiano debido a las características de las imágenes, ya que en su mayoría tienen gran cantidad de ruido, el cual es disminuido mayormente por el filtro Gaussiano. Cuando las imágenes están en buen estado no es necesario el uso de los filtros.

Para culminar, gracias a las pruebas realizadas en cada iteración se verificaron los errores y fueron corregidos inmediatamente, de igual forma gracias a la asistencia del personal de la División de Control de Estudios, se logro cumplir con los requerimientos y objetivos planteados al principio del Trabajo Especial de Grado.

Recomendaciones

Mejorar la retroalimentación entre el sistema, específicamente el módulo de procesamiento de datos, y el usuario ya que en el caso cuando se el proceso OCR esta en ejecución se muestra una barra de estado pero esta no permanece durante todo el proceso OCR, por ende el usuario tiene que esperar unos segundos después de que la barra ha desaparecido para poder visualizar el resultado de dicho proceso.

Desarrollar un módulo de emisión de oficios de aperturas y modificaciones para la División de Control de Estudios de la Facultad de Ciencias (DCE), donde se integre con el sistema de procesamiento y organización de oficios y así continuar con la automatización de los procesos de la DCE.

Desarrollar un módulo para la emisión de reportes y búsquedas donde se refleje el historial de las materias mediante el uso de minería de datos.

Referencias Bibliográficas

[1] Anónimo. (2009). Procesamiento digital de imágenes. Wikipedia. Extraído el 15 de noviembre de 2009 desde http://es.wikipedia.org/wiki/Procesamiento_digital_de_im%C3%A1genes

[2] Dr. Sugar E. Visión Computacional. Departamento de Computación ITESM Cuernavaca.

[3] Anónimo. Procesamiento Morfológico de Imágenes. Universidad Nacional de San Luis, Argentina.

[4] Anónimo. (2009). Gaussian filter. Wikipedia. Extraído el 15 de noviembre de 2009 desde http://en.wikipedia.org/wiki/Gaussian_filter

[5] Hardy R., Wheeler J., Meckley J. (1993). Method and system for thinning images. Extraído el 15 de noviembre de 2009 desde http://www.freepatentsonline.com/5261012.pdf

[6] Anónimo. (2009). Reconocimiento óptico de caracteres. Wikipedia. Extraído el 1 de noviembre de 2009 desde http://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres

[7] Gómez C. Digitalización de imágenes. Monografías. Extraído el 23 de octubre de 2009 desde http://www.monografias.com/trabajos10/digi/digi.shtml

[8] Anónimo. OCR Systems. Resource Center for Indian Language Technology Solutions. Extraído el 2 de noviembre de 2009 desde http://www.iitg.ernet.in/rcilts/ocr2.html

[9] Universidad de Vigo. Arquitectura Cliente/Servidor. Extraído el 11 de junio de 2010 desde http://ccia.ei.uvigo.es/docencia/SCS/Tema1.pdf

[10] Vegas J. HyperText Transfer Protocol, HTTP. Extraído el 11 de junio de 2010 desde http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node13.html

[11] Anónimo. File Transfer Protocol. Extraído el 11 de junio de 2010 desde http://es.wikipedia.org/wiki/File_Transfer_Protocol#El_Modelo_FTP

[12] Uranga R. Bases de datos. Extraído el 12 de junio de 2010 desde http://www.monografias.com/trabajos12/basdat/basdat.shtml#SISTEMA 12 junio 2010

[13] Anónimo. MySQL 3.23, 4.0, 4.1 Reference Manual . Extraído el 12 de junio de 2010 desde http://dev.mysql.com/doc/refman/4.1/en/index.html

[14] Anónimo. (2009). Ruby on Rails. Wikipedia. Extraído el 6 de noviembre de 2009 desde http://es.wikipedia.org/wiki/Ruby_on_Rails

[15] Anónimo. (2009). Ruby. Wikipedia. Extraído el 6 de noviembre de 2009 desde http://es.wikipedia.org/wiki/Ruby

[16] Rajshekhar A. (2008).Building Dynamic Web 2.0 Websites with Ruby on Rails. Packt Publiching.

[17] Williams J. (2007). Rails solutions Ruby on Rails made easy. Friendsof.

[18] Lenz P. Build your own Ruby on Rails Web applications. Sitepoint.

[19] Carlson L., Richardson L. (2006). Ruby Cookbook. O'Reilly.

[20] Anónimo. (2009). ImageMagick. Wikipedia. Extraido el 15 de noviembre de 2009 desde http://es.wikipedia.org/wiki/ImageMagick

[21] Vieyra G. Conceptos de HTML. Extraído el 12 de junio de 2010 desde http://www.fismat.umich.mx/~elizalde/tesis/node49.html

[22] Anónimo. Guía Breve de CSS. Extraído el 13 de junio de 2010 desde http://www.w3c.es/divulgacion/guiasbreves/hojasestilo

[23] Anónimo. JavaScript. Extraído el 13 de junio de 2010 desde https://developer.mozilla.org/es/JavaScript

[24] Anónimo. Tutorial de jQuery. Extraído el 13 de junio de 2010 desde http://www.cristalab.com/tutoriales/tutorial-de-jQuery-c214l/

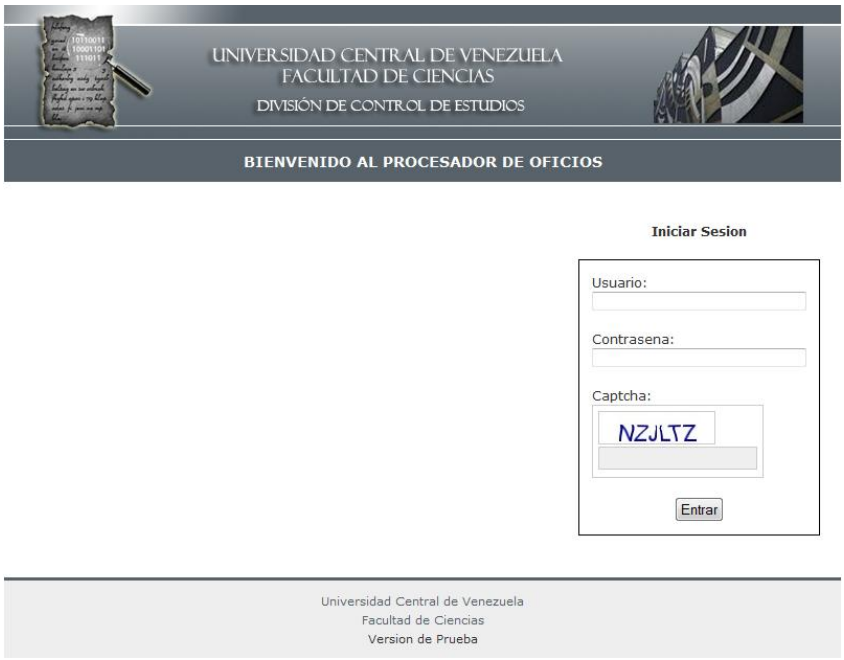
[25] Adamson C. ¿Qué es Java?. Extraído el 14 de junio de 2010 desde http://www.javahispano.org/contenidos/es/que_es_java/#java-platforms

[26] Cortizo J., Gil D., Leyva M. Extreme Programming.

Anexos

Guía de Usuario

Al ingresar al sistema Web se visualizará lo siguiente:



Donde se deberá ingresar el nombre de usuario, contraseña y los caracteres que se muestren en la parte inferior del cuadro de inicio de sesión.

Tipos de usuarios

Una este autenticado, dependiendo del rol que tenga el usuario dentro del sistema, se mostrara una serie de opciones para seleccionar en el menú principal y en el segundario.

- Usuario: para el usuario se listan opciones referentes a los consulta de datos.

- Administrador: para el administrador se listan opciones referentes a consulta, carga y procesamiento de datos, así como también manejo de usuarios, creación, listarlos, etc.
- Súper-administrador: para el súper-administrador se listan las opciones de consulta, carga y procesamiento de datos, así como también manejo de usuarios y administradores. Adicionalmente el súper-administrador tiene la posibilidad de descargar respaldos referentes a las imágenes almacenadas en el servidor y la base de datos.

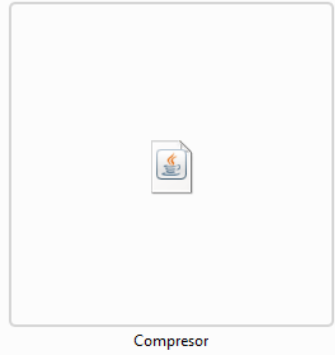
Carga de datos

Existen dos posibilidades de carga de datos una a través de la aplicación Web y otra mediante una aplicación local.

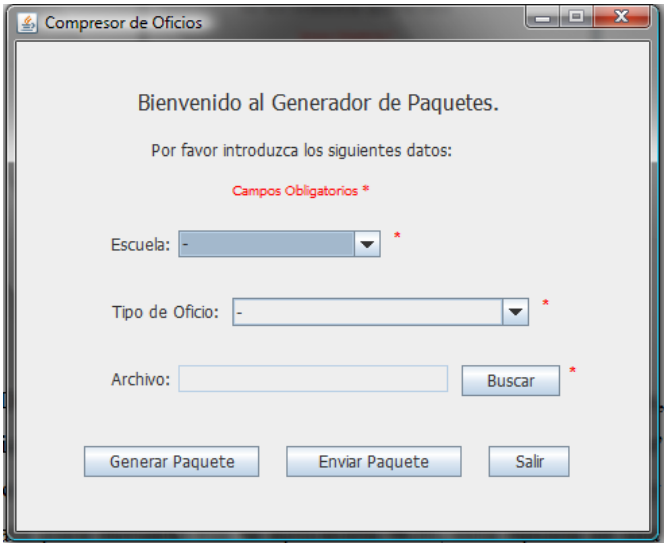
- Aplicación Web: se ingresa al módulo de carga de datos.

Una vez se haya ingresado se debe especificar el archivo (debe estar en formato zip) que va a ser cargado y se presiona el botón “subir archivo”.

- Aplicación local: para acceder a ella se debe presionar la opción de “descargar empaquetador”, en el menú segundario. Una vez descargado, descomprímalo y abra el archivo “Compresor”.



Una vez abierto se visualizará la siguiente ventana:

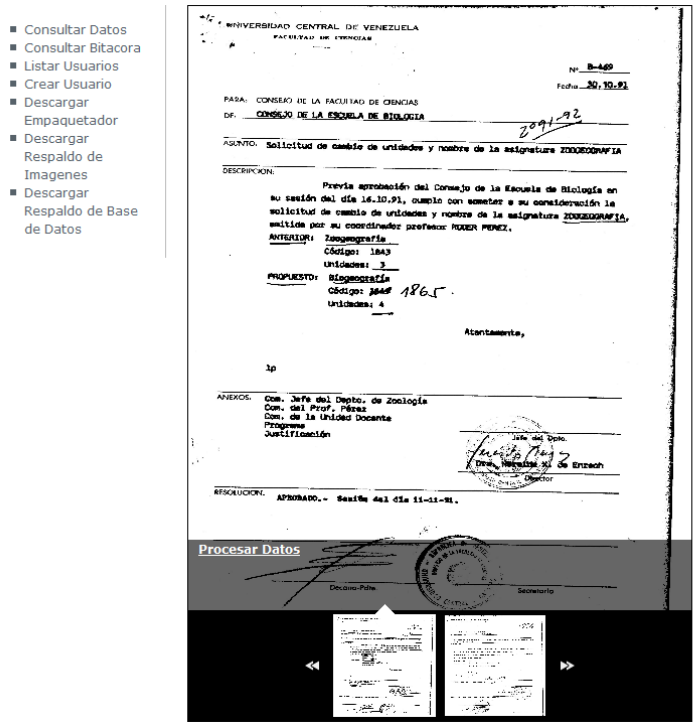


Donde deberá seleccionar cada una de las opciones requeridas, cuando todas las opciones estén seleccionadas deberá presionar "Generar Paquete" en el caso en que solo desee generar un archivo comprimido con las imágenes y su respectiva Meta-data para enviarlo vía la aplicación Web, esta opción se usa cuando no se encuentra en la misma red donde se encuentra el servidor, o "Enviar Paquete" cuando desee empaquetar y enviar las imágenes al servidor, esta opción es solo permitida cuando se realice el envío de la imágenes en la misma red donde se encuentra el servidor. Cuando el paquete este cargado se mostrar un mensaje de confirmación.

Procesamiento de datos

Para acceder al módulo de procesamiento de datos deberá presionar el ítem "Procesar Datos", localizado en el menú principal. Una vez seleccionada dicha opción, se visualizará una lista de carpetas que aun no han sido procesadas.

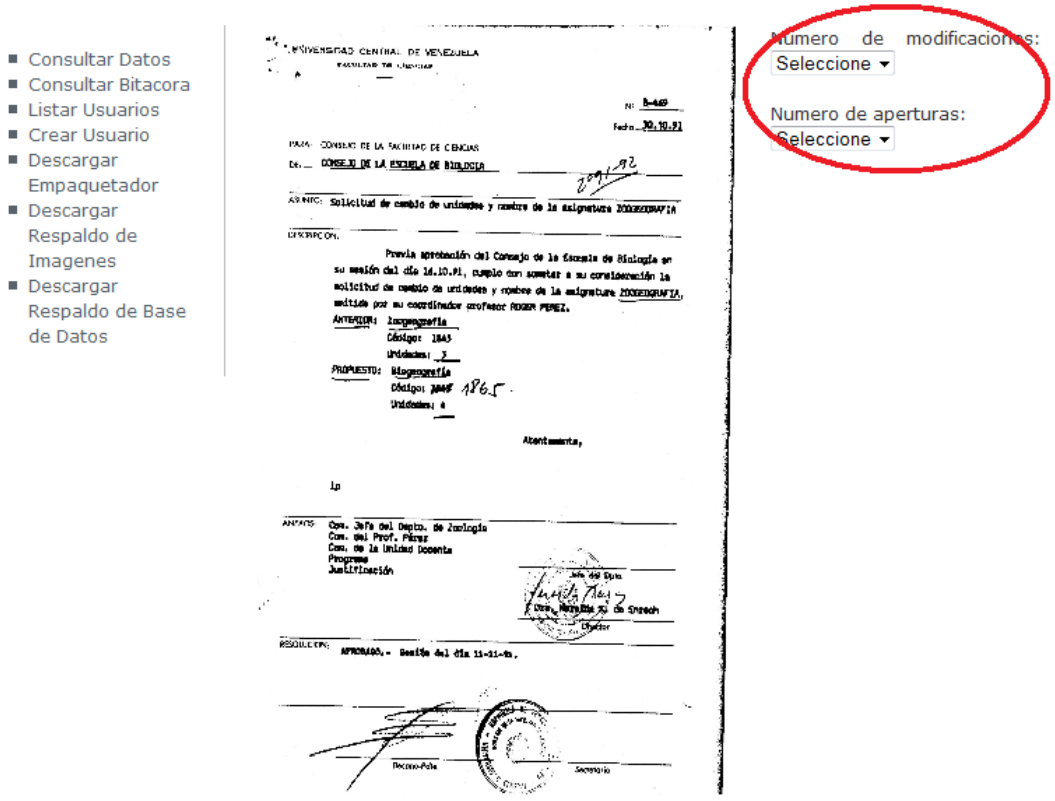
Para seleccionar una carpeta, se debe presionar la opción “Procesar datos”. Donde se visualizará una listar de imágenes pertenecientes a la carpeta seleccionada.



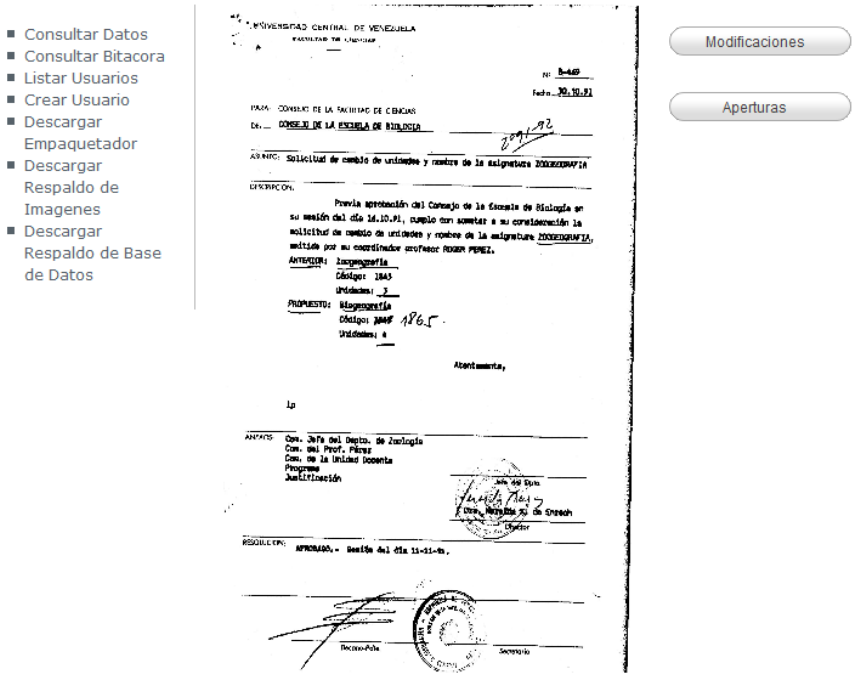
Para seleccionar una imagen en específico deberá presionar la opción “Procesar Datos”, localizado en la parte inferior izquierda de la imagen actual.

Una vez seleccionada la imagen a procesar se visualizará lo siguiente:

- Si es la primera vez que se accede al oficio o no ha sido modificado previamente se tiene una o dos selecciones, eso dependerá del tipo de oficio, en el caso de que sea de apertura o modificación se mostrará una solo selección para indicar el número de aperturas o modificaciones, en el caso en que se trate de un oficio de tipo modificación y aperturas, se mostrarán dos selecciones.

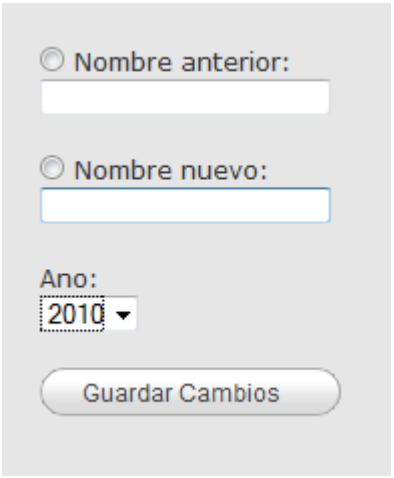


En el caso en que ya se hayan realizado cambios previos al oficio y se hayan ingresado el número de modificaciones y/o aperturas se visualizará lo siguiente:

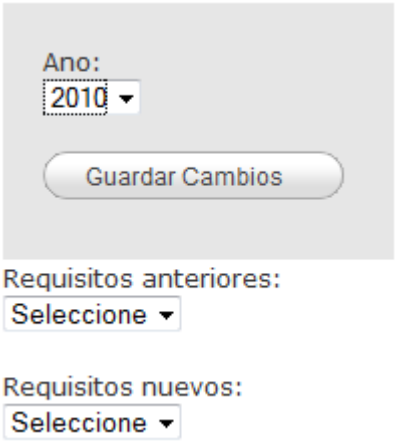


Una vez se seleccione el número de modificaciones se generaran dinámicamente, de acuerdo a la cantidad de modificaciones que debe ser procesadas, selecciones de tipo de modificación (nombre de materia, código, unidades de créditos o requisitos).

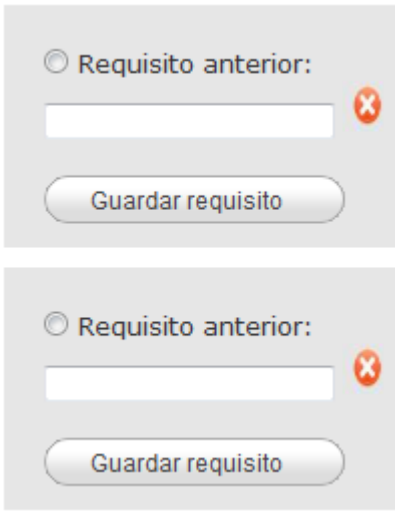
Deberá entonces, seleccionar una de las opciones de de modificación una vez seleccionada la opción (por ejemplo nombre de materia), se visualizará lo siguiente:



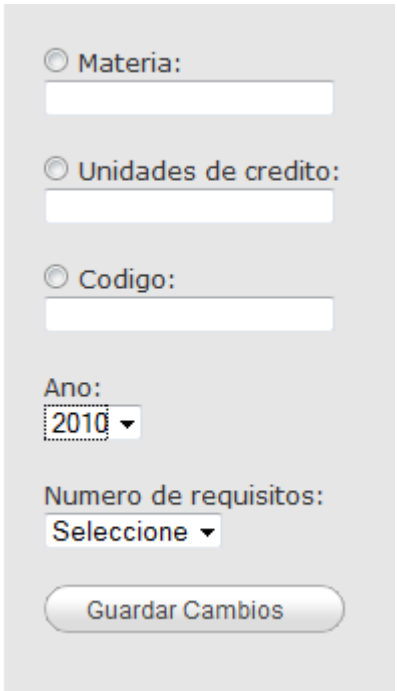
En el caso de que se trate de modificación de aperturas, se visualizará lo siguiente:



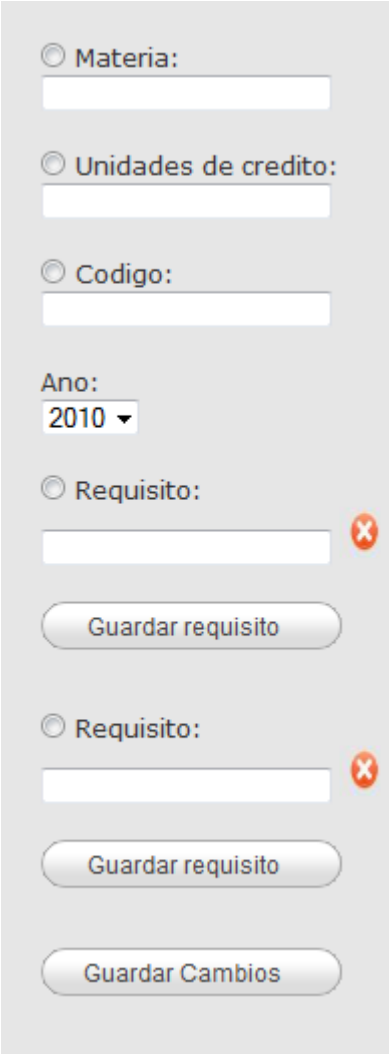
Las dos selecciones son para indicar el número de requisitos anteriores y nuevos, pero antes de seleccionar el número de requisitos es importante indicar el año del oficio. Una vez se indique el año y el número de requisitos (por ejemplo dos requisitos anteriores), se tiene lo siguiente:



Para procesar las aperturas, se debe indicar en la selección el número de aperturas o presionar el botón “Aperturas” y se visualizará lo siguiente:



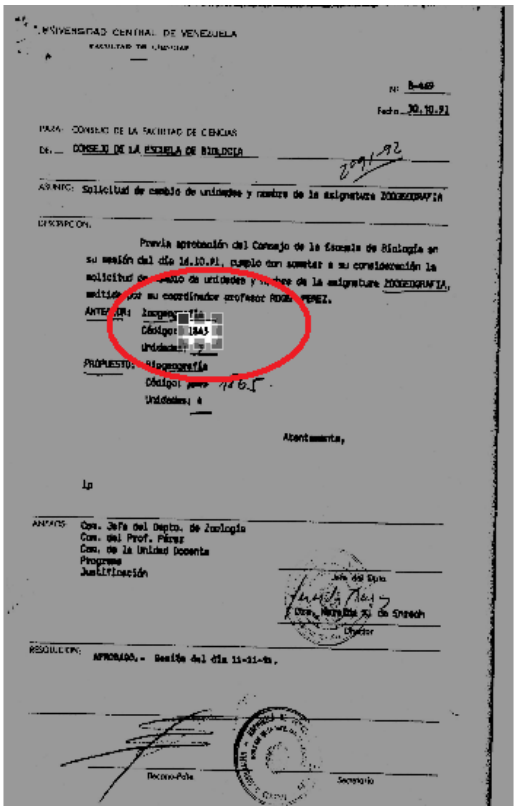
En el caso en que no se desee especificar los requisitos, no debe hacerlo, el caso en que desee se tiene una selección donde se puede indicar el número de requisitos de la materia, una vez se seleccione dicho número se tiene:



Es importante saber que se deben guardar los requisitos antes de guardar la apertura y que, en el caso de que haya más de una apertura se trabaje con una sola a la vez.

Existen dos formas de rellenar cada uno de los formularios previamente descritos, manualmente y mediante el proceso OCR.

Para hacer uso del proceso OCR deberá seleccionar la parte de la imagen donde se encuentra el texto que desea traducir.


UNIVERSIDAD CENTRAL DE VENEZUELA

Nº B-469

Fecha 30.10.91

PARA: CONSEJO DE LA FACULTAD DE CIENCIAS

DE: CONSEJO DE LA ESCUELA DE BIOLOGIA

ASUNTO: Solicitud de cambio de unidades y nombre de la asignatura BIOGEOGRAFIA

DESCRIPCION:

Previa aprobación del Consejo de la Escuela de Biología en su sesión del día 16.10.91, cumplo con someter a su consideración la solicitud de cambio de unidades y nombre de la asignatura BIOGEOGRAFIA, emitida por su coordinador profesor ROSA PEREZ.

ANTERIOR: Ingeniería
Código: 1845
Unidades: 2

PROPUESTO: Biogeografía
Código: 1845
Unidades: 4

Atentamente,

lp

ANEXOS: Con. Jefa del Depto. de Zoología
Con. del Prof. Pérez
Con. de la Unidad Docente
Programa
Justificación

Jefa del Dpto.

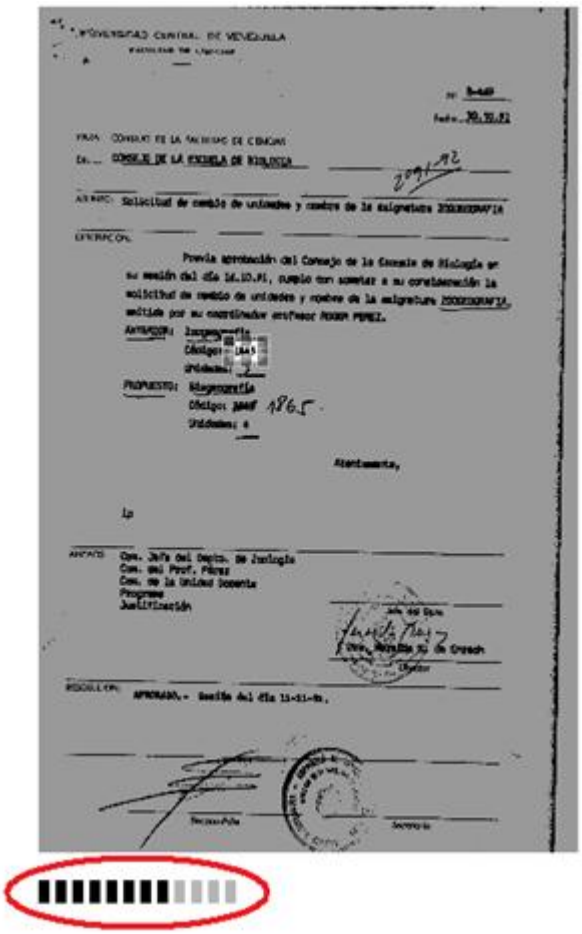
Director

RESOLUCION: APROBADO.- Sesión del día 11-11-91.

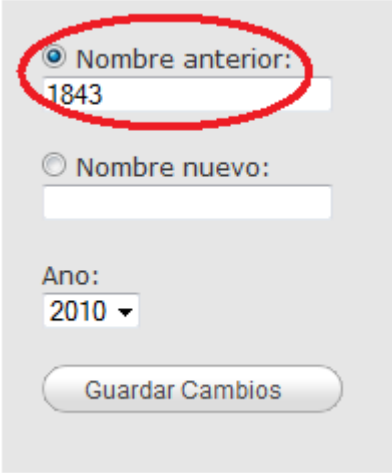
Decano-Rela

Secretario

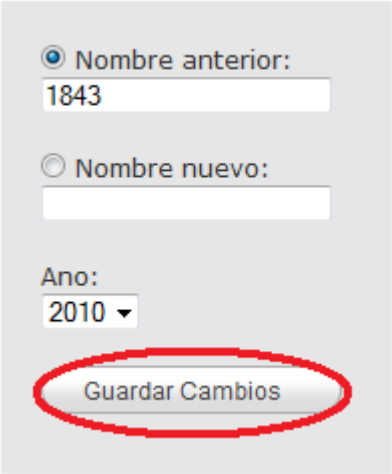
Una vez seleccionada la parte de la imagen se mostrará una barra de carga, en la parte inferior de la imagen, que indicará que el proceso OCR se está ejecutando.



Una vez la barra de carga desaparezca se selecciona el radio button que se encuentra sobre el campo de texto que se quiera rellenar con los datos generados por el proceso OCR.



Una vez se hayan realizado hayan procesado los datos reflejados en el oficio se procede a guardar cada uno de los formularios, presionando el botón “Guardar Cambios”.

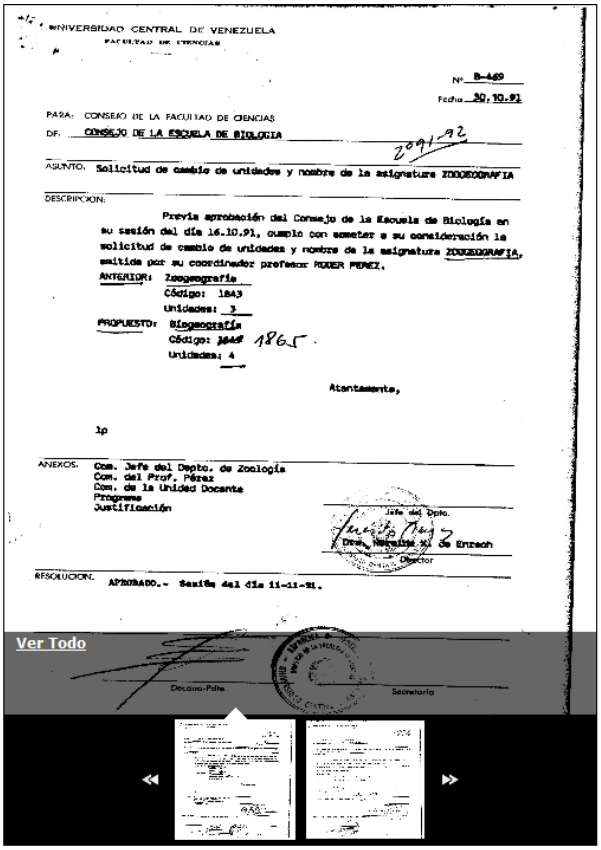


Procesamiento de imágenes

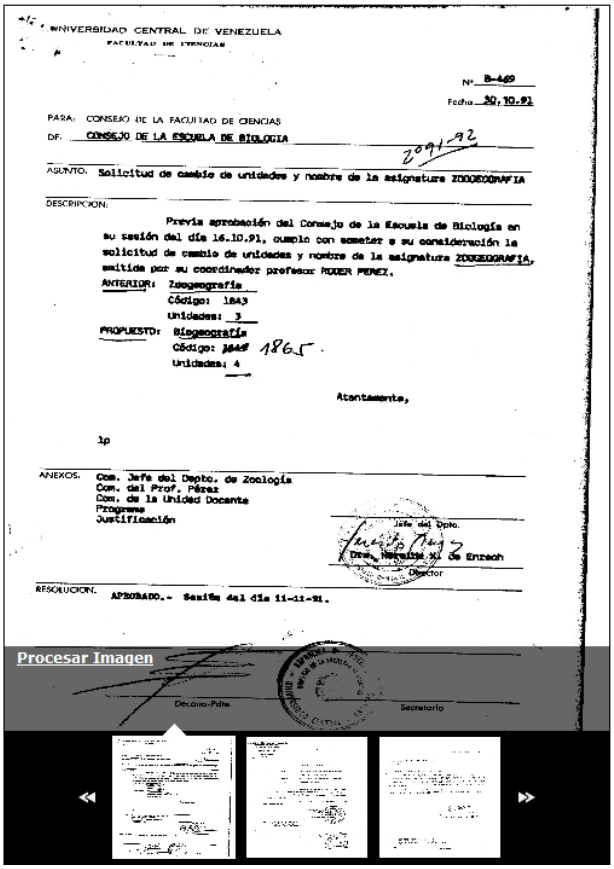
Para ingresar al módulo de procesamiento de imágenes debe seleccionar la opción “Procesar Imágenes”, la cual se encuentra en el menú principal.

Una vez dentro del módulo de procesamiento de imágenes se visualizarán todas las carpetas cargadas en el sistema.

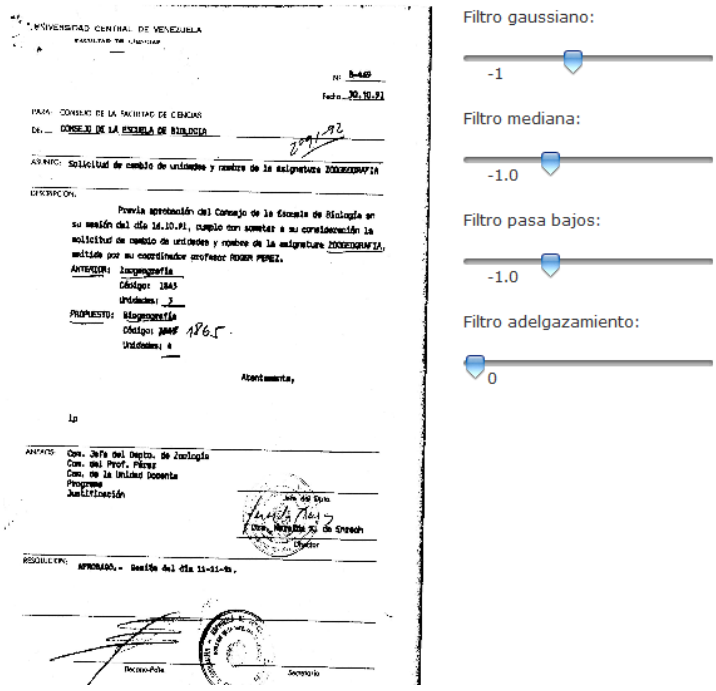
Para acceder las imágenes pertenecientes a una carpeta se debe presionar la opción “Procesar imágenes”, una vez se acceda a la carpeta se listaran los oficios principales de dicha carpeta.



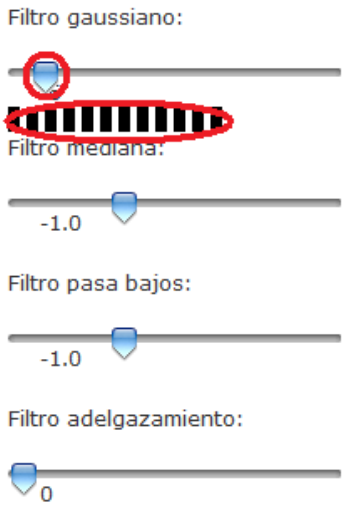
Para acceder al oficio completo (página principal y anexos), se selecciona la opción “Ver Todo”, localizada en la parte inferior izquierda de la imagen actual.



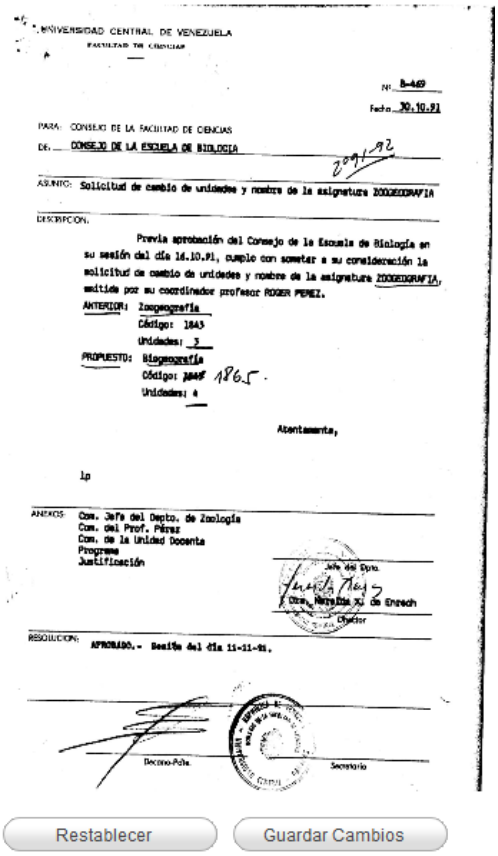
Una vez cargados el oficio completo, se tiene la opción de seleccionar una imagen para procesarla, seleccionando la opción “Procesar Imagen”. Una vez seleccionada una imagen se tienen los filtros, los cuales apoyan al mantenimiento y mejora de las imágenes.



Para utilizar los filtros, debe deslizar el icono azul, una vez se tenga el valor que desea utilizar se debe soltar el icono azul y aparecerá una barra de carga, indicando que el filtro está en proceso.



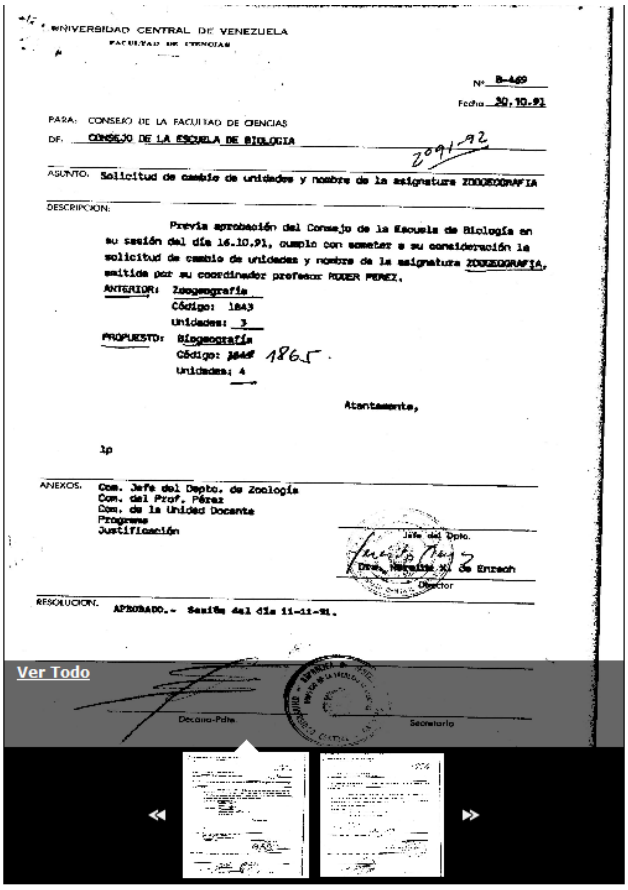
Una vez el filtro haya sido aplicado, se mostrará la imagen con los cambios y debajo de ella dos opciones “Restablecer” y “Guardar”, deberá seleccionar la opción que desee o aplicar nuevamente un filtro.



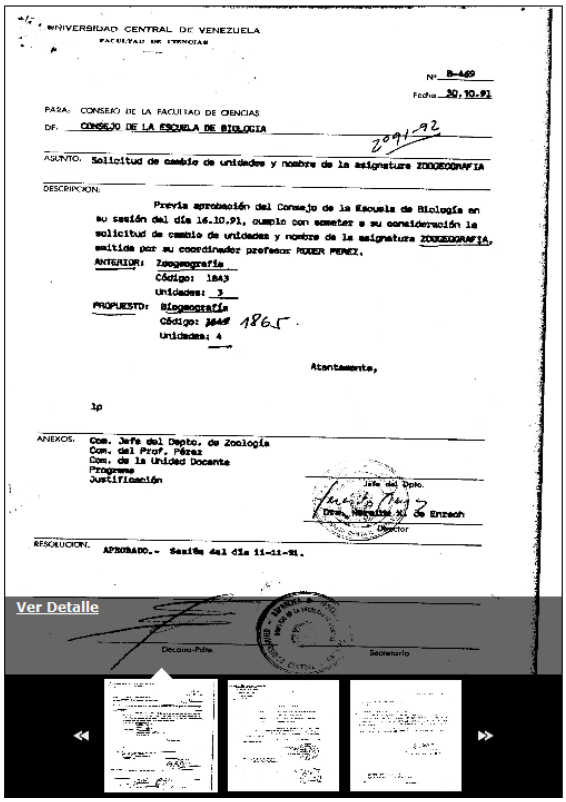
Consulta de Oficios

Para acceder a esta opción se debe seleccionar la opción “Consultar Oficios”, localizada en el menú principal, una vez se seleccione dicha opción, se visualizarán todas las carpetas cargadas en el sistema.

Para acceder a las imágenes pertenecientes a una carpeta en específico, se debe seleccionar la opción “Ver oficios”. Una vez seleccionada una carpeta se visualizarán todas las páginas principales de los oficios pertenecientes a dicha carpeta.

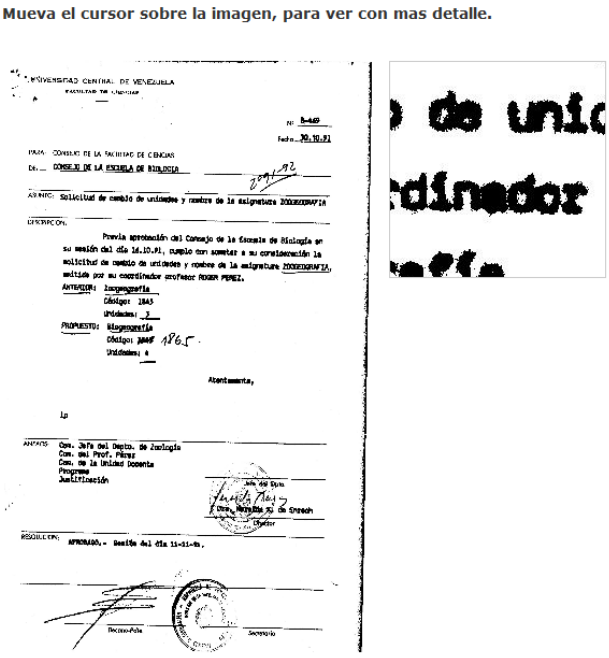


Para visualizar un oficio completo se debe seleccionar la opción “Ver Todo”. Donde se mostrará todas imágenes pertenecientes a dicho oficio.

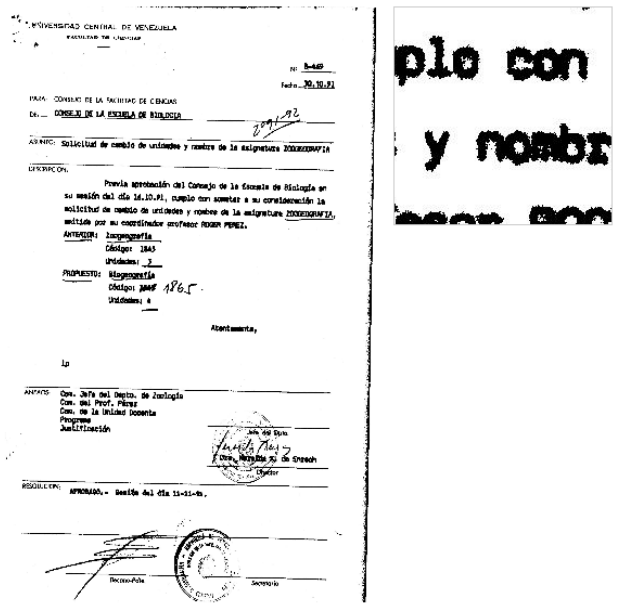


Para visualizar una página en específico, se debe seleccionar la opción “Ver Detalle”, una vez seleccionada dicha opción, se contará con diferentes opciones de visualización.

Se podrá ver el oficio con más detalle mediante el uso de un acercamiento, el cual funciona al mover el ratón sobre la imagen.

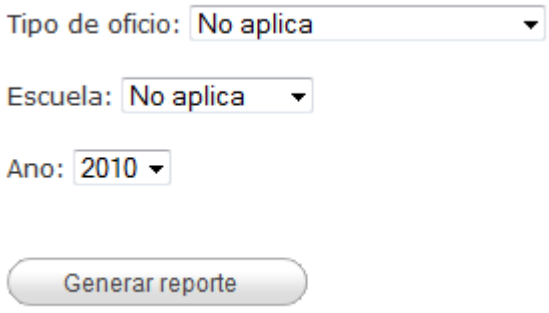


De igual forma, en la parte inferior de la imagen se visualizan las opciones de “Ver Versiones”, donde se podrán visualizar todas las versiones de la imagen generadas tras el procesamiento de imágenes (mediante el uso de filtros), la opción “Descargar PDF” permite descargar todo el oficio (imagen principal y anexos), en formato PDF y por último la opción “Cambiar Imagen” permite sustituir una imagen.

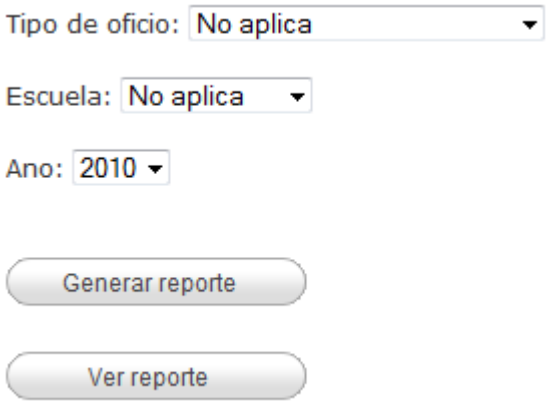


Reportes

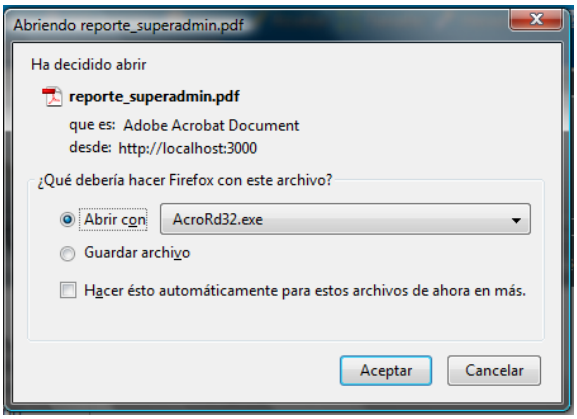
Para ingresar a este módulo se debe seleccionar la opción “Reportes”, localizado en el menú principal. Una vez se seleccione esta opción se deberá elegir entre varias opciones para generar un reporte.



Una vez seleccionadas las opciones para generar el reporte se debe presionar el botón “Generar Reporte” se mostrar un botón para visualizar el reporte “Ver reporte”.

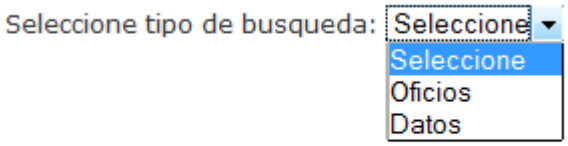


Al seleccionar la opción para ver el reporte se mostrar una ventana para abrir o guardar el reporte.

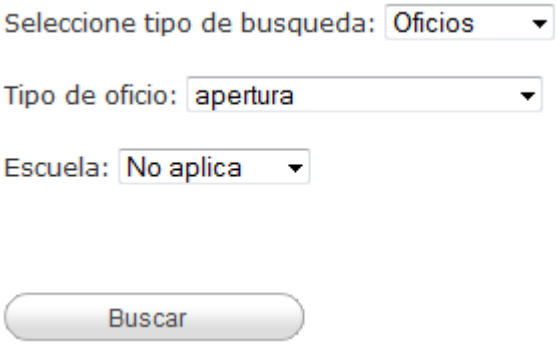


Búsquedas

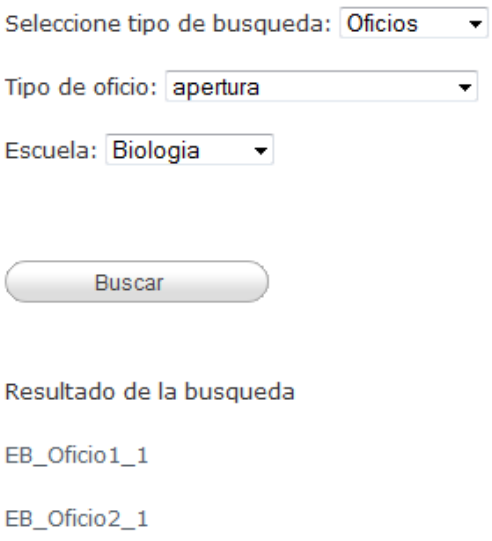
Para acceder a este módulo se debe seleccionar la opción “Búsquedas”, localizada en el menú principal. Una vez se haya seleccionada dicha opción se podrá visualizar una selección, en la cual se puede elegir entre buscar por datos (datos procesados) u oficios.



En el que caso en que se seleccione la opción de búsquedas de oficios se visualizará lo siguiente:

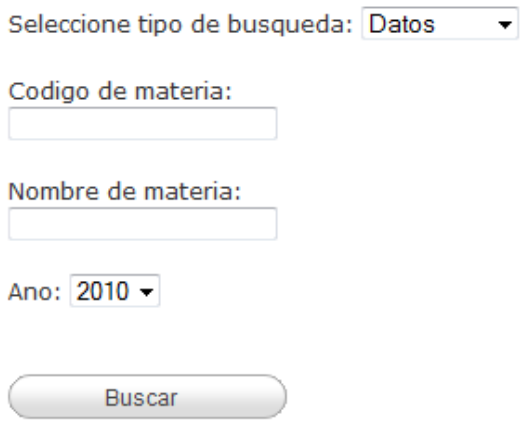


Una vez seleccionadas las opciones de búsqueda, se debe presionar el botón "Buscar", donde se mostrará todas las coincidencias con los datos seleccionados.

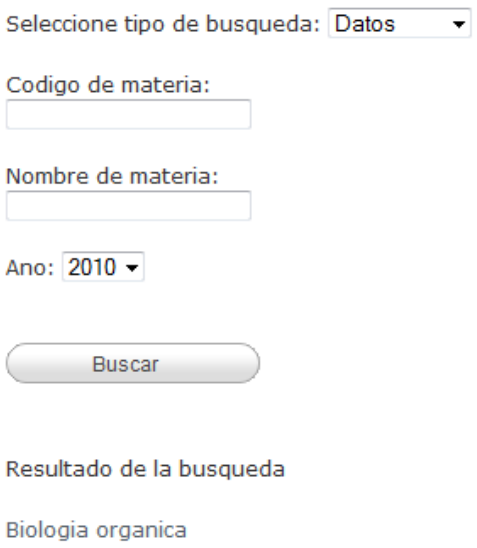


Para acceder a los resultados de la búsqueda, de hacer click sobre la opción de su preferencia. Una vez seleccionada la imagen a visualizar se mostraran todas las imágenes pertenecientes al oficio, como se muestra en el módulo de consulta de oficios.

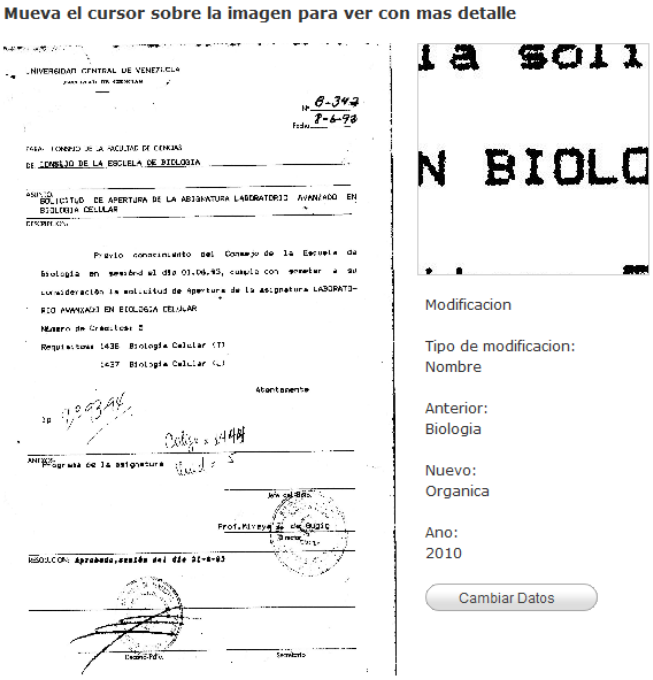
En el caso, que se desee buscar datos, se debe seleccionar en la selección principal la opción “Datos”. Una vez seleccionada dicha opción se muestran varias opciones de búsquedas.



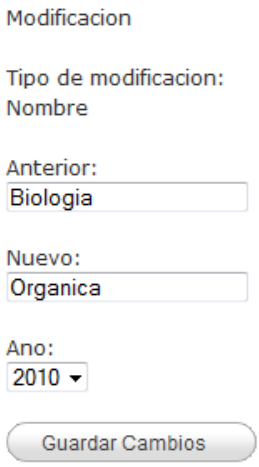
Una vez seleccionadas las opciones de búsqueda se muestran los resultados de la misma.



Para visualizar con más detalle alguno de los resultados de la búsqueda, haga click sobre la opción que desee, donde se mostrará en detalle los datos, donde se cuenta con la herramienta de acercamiento explicada en el módulo de consulta de oficios y la opción de cambio de datos, para acceder a ella presione “Cambiar Datos”.

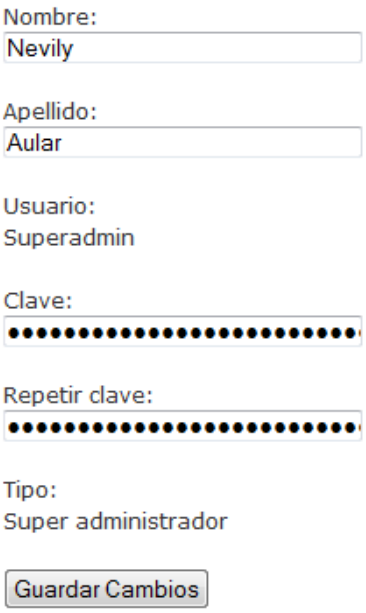


Una vez seleccionada dicha opción, se permitirá modificación de los datos.



Consultar Datos

Para acceder a este módulo debe seleccionar la opción “Consultar Datos”, localizado en el menú segundario, en él podrá consultar y modificar sus datos de usuarios.

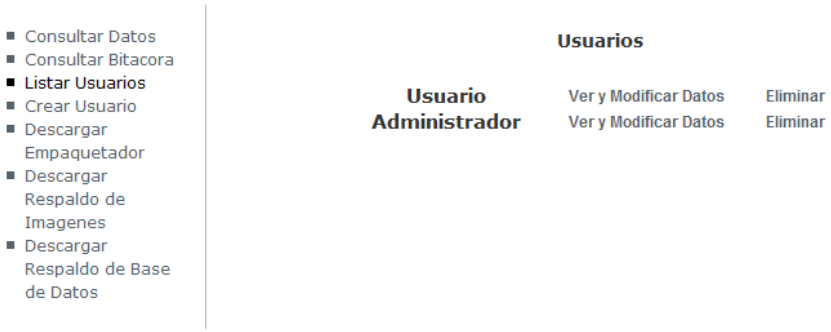


Consultar Bitácora

Para acceder a este módulo presione “Consultar Bitácora”, localizado en el menú segundario, en él podrá visualizar cada uno de las tareas realizadas en el sistema por los usuarios.

Listar Usuarios

Para acceder a este módulo debe seleccionar la opción “Listar Usuarios”, localizado en el menú segundario. Este módulo permite listar los usuarios y administradores (en el caso del súper-administrador), eliminarlos y modificar sus datos.

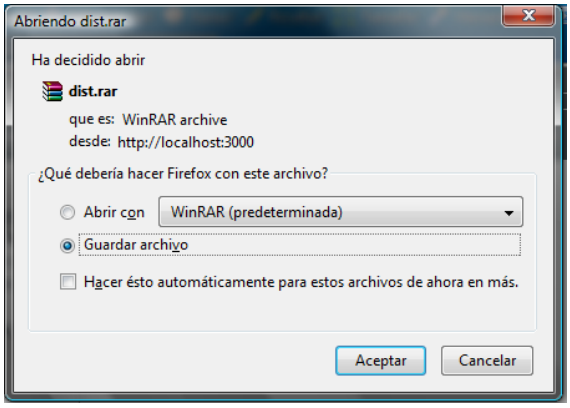


Crear Usuario

Para acceder a este módulo debe seleccionar la opción “Crear Usuario”, localizado en el menú segundario, solo el súper-administrador podrá crear administradores.

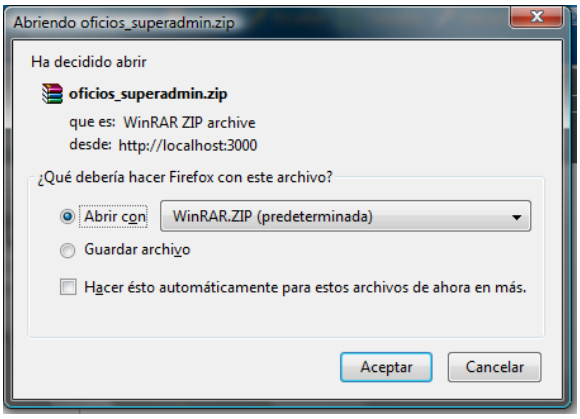
Descargar Empaquetador

Para acceder a esta opción debe seleccionar “Descargar Empaquetador”, localizado en el menú segundario. Una vez seleccionada dicha opción se mostrará una ventana externa para descargar el empaquetador.



Descargar Respaldo de Imágenes

Para acceder a este módulo debe seleccionar la opción “Descargar Respaldo de Imágenes”, localizado en el menú segundario. Una vez seleccionada la opción se mostrará una ventana externa para descargar el respaldo de las imágenes.



Descargar Respaldo de Base de Datos

Para acceder a este módulo debe seleccionar la opción “Descargar Respaldo de Base de Datos”, localizado en el menú segundario. Una vez seleccionada la opción se mostrará una ventana externa para descargar el respaldo de la base de datos.