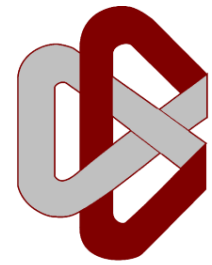


CIMAT

Centro de Investigación en Matemáticas, A.C.

Diseño e Implementación de una Calculadora de Imágenes basada en Tecnologías Web

T E S I S

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

P r e s e n t a

Alfonso Ceseña Quiñones

Bajo la dirección conjunta de:

Dr. José Luis Marroquín Zaleta

Dr. Mariano José Juan Rivera Meraz

Guanajuato, Gto. Noviembre de 2012

ALFONSO CESEÑA QUIÑONES

DISEÑO E IMPLEMENTACIÓN DE UNA CALCULADORA DE IMÁGENES BASADA EN TECNOLOGÍAS WEB

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS, A. C.

DISEÑO E IMPLEMENTACIÓN DE UNA CALCULADORA DE IMÁGENES BASADA EN TECNOLOGÍAS WEB

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS

CON ESPECIALIDAD EN COMPUTACIÓN Y MATEMÁTICAS INDUSTRIALES

PRESENTA

ALFONSO CESEÑA QUIÑONES

BAJO LA DIRECCIÓN CONJUNTA DE:

Dr. José Luis Marroquín Zaleta

Dr. Mariano José Juan Rivera Meraz

Noviembre 2012
Guanajuato, Guanajuato, México.

— A mis padres con gran admiración, a mi familia con mucho cariño, a mis maestros con mucha gratitud, a todos por su apoyo e influencia, a quien corresponda que le halle provecho, a nadie espero haber omitido.

RESUMEN

CALIMAN es un programa de procesamiento de imágenes desarrollado en el área de Ciencias de la Computación del CIMAT. Se trata de una herramienta de procesamiento de imágenes a través de operaciones con interfaz de usuario similar a una calculadora programable. Con la sucesiva inclusión de nuevas operaciones este programa se ha ido deteriorando hasta el grado de tener importantes problemas de extensibilidad y de escalabilidad. Debido a esto dicha problemática se puede enmarcar en una zona de interés común tanto al procesamiento digital de señales como a la ingeniería de software.

En el presente trabajo el interés se centra en plantear una arquitectura de software híbrida basada en tecnologías web para sustituir la arquitectura monolítica en que se encuentra la calculadora con el fin de facilitar su mantenimiento y resolver los problemas detectados. Se propone que la construcción de dicha calculadora se haga siguiendo principios disciplinados de desarrollo de software otorgando mayor preferencia a una separación explícita de responsabilidades. Todo ello aprovechando tecnologías estándar, de amplia utilización, que favorezcan el flujo de información para atender dicha separación y posibiliten la migración a otras plataformas de operación.

Concretamente, se propone un desarrollo a partir de un modelo de composición de software por medio de una orquestación de sustituciones en plantillas basadas en texto. Se presenta una prueba sustancial de concepto que reproduce la experiencia acumulada con las versiones anteriores de la calculadora a la vez que favorece el crecimiento del catálogo de funciones que ofrece, dando más prioridad a esto último. Se muestran con sumo detalle los mecanismos de esta nueva calculadora junto con una herramienta para apoyar la autoría de sus operaciones.

¡Adiós, gracias;
adiós, donaires;
adiós, regocijados amigos;
que yo me voy muriendo,
y deseando veros presto contentos en la otra vida!

— Miguel de Cervantes Saavedra [de Cervantes Saavedra, 1616]

AGRADECIMIENTOS

Agradezco a mis directores de tesis, por haberme dado la oportunidad de trabajar con ellos, por dejarme hacer y deshacer, aprecio entrañablemente su paciencia y apoyo. Sirva este trabajo como un testimonio de gratitud.

A mis maestros, a lo largo de lo que llevo de vida, por inspirarme a no soltar la estafeta que mantiene encendida la llama de mi curiosidad.

A mi familia y amigos, pues aunque solo me ven entretenido de sol a sol frente a un extraño y mágico aparato del que sale luz, siempre están ahí. Que vean reflejado en este documento el resultado de su apoyo incondicional.

Un agradecimiento especial al Consejo Nacional de Ciencia y Tecnología por su espléndido programa de Becas Nacionales del cual fui beneficiario con la beca número 245045.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

ÍNDICE DE LISTADOS

ACRÓNIMOS

CONACYT Consejo Nacional de Ciencia y Tecnología

CALIMAN Calculadora de Imágenes

CIMAT Centro de Investigación en Matemáticas, A.C.

RAM Random Access Memory o Memoria de Acceso Aleatorio

GPU Graphics Processing Unit o Unidad de Procesamiento Gráfico

COM Component Object Model o Modelo de Objetos de Componentes

SDK Software Development Kit o Kit de Desarrollo de Software

API Application Programming Interface o Interfaz de Programación de Aplicaciones

BMP formato de archivo de imagen Windows Bitmap

JPEG formato de archivo de imagen Joint Photographic Experts Group

GIF formato de archivo de imagen Compuserve Graphics Interchange Format

TIFF formato de archivo de imagen Tagged Image File Format

PNG formato de archivo de imagen Portable Network Graphics

RGB Modelo de Color (Red Green Blue)

IETF Internet Engineering Task Force

RFC Request for comment

TCP Transmission Control Protocol o Protocolo de Control de Transmisión

IP Internet Protocol o Protocolo de Internet

CGI Common Gateway Interface o Interfaz común de portales

HTTP Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto

W3C World Wide Web Consortium

HTML Hypertext Markup Language o Lenguaje de Marcado de Hipertexto

XML Extensible Markup Language o Lenguaje de Marcado Extensible

DOM Document Object Model o Modelo de Objetos de Documentos

CSS Cascading Stylesheets u Hojas de estilo en cascada

URL Uniform Resource Locator o Localizador Uniforme de Recursos

JSON Javascript Object Notation o Notación de Objetos de Javascript

HTML5 Hypertext Markup Language, versión 5

GCC GNU Compiler Collection

MinGW Minimalist GNU for Windows

PRÓLOGO

Este trabajo de tesis trata del diseño e implementación de un sistema de procesamiento de imágenes con apariencia de calculadora. El desarrollo de dicho sistema sirve de marco para discutir la aplicación de tecnologías web para renovar el software o ampliar su ciclo de vida.

Esta tesis se presenta en cinco capítulos, de los cuales, los tres primeros son de carácter teórico con el propósito de dar a conocer al lector el contexto del presente trabajo y los capítulos restantes bosquejan el funcionamiento del sistema en la práctica.

ORGANIZACIÓN DEL DOCUMENTO

La organización del documento es la siguiente:

CAPÍTULO 1: Se expone lo relacionado con la concepción, preparación y planeación del proyecto. Se encuentra información referente a la justificación, hipótesis y objetivos (generales y específicos) así como la descripción, alcances y limitaciones junto con las contribuciones de este trabajo.

CAPÍTULO 2: Se plantea un contexto de discusión y para ello se sondea brevemente el estado del arte, se citan varias aplicaciones existentes y sus implementaciones para enterarse de los trabajos afines a nuestro proyecto que se están produciendo en otros lugares.

CAPÍTULO 3: Se desarrolla la parte teórica del proyecto. Para cimentar la propuesta se mencionan algunos conceptos de arquitecturas de software, sistemas distribuidos y tecnologías web, así como de la manipulación de cadenas de texto y procesamiento digital de imágenes, terminando con características deseables de un producto de software y a partir de ellos se determinan las herramientas que se van a emplear para atacar el problema en cuestión.

CAPÍTULO 4: Enmarcado en los conceptos presentados en el capítulo anterior, se introduce y discute la arquitectura propuesta motivo de este trabajo. Se da una descripción detallada de las piezas de la línea de producción y se da seguimiento paso a paso a un ejemplo que ilustra la manera en que se desempeñan los artefactos componentes de la solución propuesta.

CAPÍTULO 5: Se presentan los resultados obtenidos. Se muestra evidencia de la ejecución del proyecto incluyendo capturas de pantalla acompañadas de descripciones sencillas de cada una de ellas. Se discuten detalles y se exponen conclusiones sobre los resultados obtenidos por la aplicación.

APÉNDICE A: Se presentan algunas tablas y listas de referencia miscelánea.

APÉNDICE B: Una referencia para el desarrollador, en donde se da una completa descripción de alto nivel de la calculadora y como añadir una operación a la misma.

Parte I

TEORÍA

En esta parte se presenta el problema, su contexto y su marco conceptual.

1 INTRODUCCIÓN

En este capítulo se presenta una descripción del problema detectado y aspectos generales de la solución que se propone.

1.1 PLANTEAMIENTO DEL PROBLEMA

Hay, en diversos campos de acción humana tales como el ámbito académico, o ambientes de producción industrial, o en ciertas aplicaciones médicas, entre otros, ciertos problemas cuya solución puede involucrar técnicas de procesamiento de imágenes.

Un ejemplo concreto en el campo industrial es evidente en la aplicación de dichas técnicas al control de calidad de ciertos productos por inspección visual asistida; siendo el caso que el desarrollo de herramientas especializadas para llevarlo a cabo puede ser extremadamente costoso, y en esas situaciones, una potente solución de bajo costo resultaría atractiva.

Por otra parte, en el campo de la medicina se pueden encontrar aplicaciones de visualización y análisis de la anatomía humana o aquellas en que se hacen conteos de ciertas partículas en imágenes microscópicas. Ahí, los mejores resultados se deben a cuidadosos tratamientos de dichas imágenes con instrumentos que permiten ajustar valores de umbrales o delimitar zonas de acción o tolerancia en determinados procesos, lo que sugeriría la necesidad de herramientas con cierta flexibilidad.

En el campo académico, particularmente en el área de procesamiento de imágenes y las áreas de investigación relacionadas, tales como las que involucran Visión Computacional y Gráficas Computacionales, se requieren algoritmos matemáticos que suelen presentarse en la forma de sofisticadas operaciones para la manipulación de imágenes; siendo en ese contexto que la exploración de métodos se vería beneficiada con una herramienta que ofrezca una gran variedad de ellos y que permita emplearlos fácilmente. Más aún, la enseñanza de dichos métodos se puede complementar con un ambiente de trabajo que permita hacer experimentos; que beneficiaría a personas con experiencia mínima si sus posibilidades son presentadas de una manera que pudiera serles familiar o intuitiva, como en una calculadora de bolsillo.

Se parte de un sistema multifuncional de procesamiento de imágenes llamado Calculadora de Imágenes (CALIMAN) con la apariencia de una calculadora de bolsillo. Dicho sistema es pertinente en las situaciones antes mencionadas. Se detectaron algunas deficiencias en su

implementación, que se identificaron como problemas de extensibilidad y de escalabilidad. En este contexto la extensibilidad se refiere a su capacidad para tener un catálogo más amplio de operaciones y la escalabilidad se refiere a su capacidad de enfrentar situaciones de tamaños distintos con un grado similar de éxito. El problema de extensibilidad es evidente al examinar el código fuente y encontrar referencias fijas a las operaciones dispersas en muchos lugares de manera tal que es difícil agregar una operación para las personas que no estén familiarizadas con el resto del código fuente[1]. El problema de escalabilidad es evidente al usar el programa y encontrar que solamente se pueden desplegar doce imágenes a la vez y que únicamente se puede modificar este comportamiento compilando de nuevo. Todo esto debido a que la arquitectura de software en que se basa su construcción es monolítica. Una arquitectura monolítica es aquella en que una sola entidad funcional tiene a su cargo varias responsabilidades[2].

Evidentemente, dado que no es un proceso inmediato añadir funciones nuevas, se afecta una extensibilidad deseable en una herramienta que atiende un campo del conocimiento en constante cambio. También se ve reducida su potencial adopción en más ámbitos de aplicación al no dar cabida a tareas que se orienten a situaciones distintas a las que vienen incluidas de fábrica. Respecto a esta deficiencia, es muy notorio que han pasado algunos años desde que se liberó la versión más reciente mientras que al mismo tiempo han ido apareciendo nuevos métodos de procesamiento de imágenes que bien podrían formar parte del catálogo de operaciones ofrecidas y no se han incluido debido a esta dificultad.

Concretamente por la parte de la escalabilidad, hay un límite específico, fijo, muy bajo, de doce operandos imagen que se pueden tener disponibles en un momento determinado. Además de que dichas imágenes a su vez están severamente restringidas en sus características, resaltando que la cantidad de canales de color con los que se opera está limitado a uno (como escala de grises). Tampoco se puede trabajar con imágenes en formatos distintos al formato de archivo de imagen Windows Bitmap (BMP) sin antes pasar por un proceso externo de transformación. Finalmente, no es fácil actuar sobre imágenes grandes por las dificultades técnicas que implica tener tanta memoria disponible a la vez.

En suma, se identifica una problemática de deterioro en un sistema de software de aplicación específica. Debido a esto los problemas enunciados se pueden enmarcar en una zona de interés común tanto al procesamiento digital de señales como a la ingeniería de software.

1 Asumiendo que dichas personas solamente están interesadas en implementar la operación de su interés y no en estar buscando por todo el código fuente donde hay que hacer las modificaciones necesarias.

2 Que van desde las responsabilidades comunes a un sistema de software (al nivel de acceso de lectura o escritura hacia y desde archivos y memoria), hasta las responsabilidades propias que le dan utilidad y propósito a dicho sistema.

1.2 JUSTIFICACIÓN DEL PROYECTO

Con el antecedente del interés que suscita una herramienta que efectúa tratamientos numéricos en imágenes, permitiendo realizar ajustes y pruebas o demostraciones de funcionamiento de algoritmos de procesamiento de las mismas, este proyecto pretende la adopción de una arquitectura híbrida de software para implementar una calculadora de imágenes tal que supere las deficiencias encontradas, antes mencionadas, además de que facilite las tareas de ajuste de configuración a sus usuarios y las labores de extensión de su funcionalidad a sus desarrolladores. Todo lo anterior con miras a favorecer su empleo en contextos más amplios y a la vez dar lugar a ahorros importantes de tiempo en su desarrollo y mantenimiento.

A este respecto se observan capacidades importantes, que son de interés valorar, en el empleo de tecnologías web, basadas principalmente en cadenas de texto, por lo que se justifica explorarlas con una aplicación concreta que brinda una solución al problema detectado. Concretamente, es de interés considerar la separación explícita de responsabilidades que ofrecen para el desarrollo de software.

La importancia de este proyecto radica en la exhibición de un caso específico en que se pueden aprovechar las tecnologías web para ofrecer una solución al deterioro potencial del software.

1.3 HIPÓTESIS

Es posible implementar una calculadora de imágenes a través de la composición de software, por medio de plantillas basadas en texto, apoyada en una arquitectura de software híbrida que sustituya a la arquitectura monolítica. De esta manera, otorgando mayor preferencia a la separación explícita de responsabilidades, es posible facilitar su mantenimiento y fomentar la extensión de sus propósitos. Todo ello aprovechando tecnologías estándar, de amplia utilización, que favorezcan el flujo de información para atender dicha separación y posibiliten la migración a otras plataformas de operación.

1.4 OBJETIVO GENERAL DEL PROYECTO

Implementar una calculadora de imágenes usando tecnologías web, siguiendo principios disciplinados de desarrollo de software que faciliten su mantenimiento y extensibilidad para aminorar su deterioro.

1.5 OBJETIVOS ESPECÍFICOS DEL PROYECTO

Una calculadora de imágenes que tenga capacidad para admitir nuevas operaciones y que para ello sea construida sobre una infra-

estructura que prefiera la separación explícita de responsabilidades debe considerar los siguientes aspectos:

PROPORCIONAR LOS MEDIOS PARA AÑADIR OPERACIONES. Mediante un grupo de formularios con los que se puedan especificar cantidad y tipos de datos tanto de sus entradas como de sus salidas.

FACILITAR LOS MEDIOS PARA ACCEDER A LAS OPERACIONES. Mediante artefactos de activación de mecanismos tales como botones debidamente etiquetados.

SUMINISTRAR LOS MEDIOS PARA AGRUPAR OPERACIONES. Mediante una interfaz gráfica en la que se puedan acomodar los botones en grupos, ya sea por sus características de operación o por el tipo de datos de sus resultados tanto para tener asociaciones que hagan sentido para los usuarios como para facilitar su ubicación.

INSTRUMENTAR LOS MEDIOS PARA INTRODUCIR PARÁMETROS A LAS OPERACIONES. Mediante artefactos de inserción de datos tales como campos de texto y cajas desplegables de opciones.

PROCESAR LAS IMÁGENES DE ACUERDO A LAS OPERACIONES ELEGIDAS POR EL USUARIO. Mediante un módulo de procesamiento especializado para cada tipo de operación.

DOTAR DE LOS MEDIOS PARA PROGRAMAR SECUENCIAS DE OPERACIONES. Mediante un lenguaje sencillo en que se puedan expresar tanto las operaciones como sus parámetros.

PROVEER LOS MEDIOS PARA MOSTRAR RESULTADOS DE OPERACIONES. Mediante un tablero de nichos que sirva de escaparate para exhibir las imágenes resultantes de las operaciones que se vayan efectuando.

SERVIRSE DE MEDIOS DE TRANSFERENCIA Y ALMACENAMIENTO DE INFORMACIÓN. Mediante el uso de bibliotecas de rutinas así como estructuras de datos con las que se pueda implementar una infraestructura que permita atacar algunas de las deficiencias detectadas. Particularmente para el manejo de distintos formatos de imágenes, o de imágenes con más de un canal de color o la posibilidad de manejar distintas cantidades de imágenes y de actuar sobre imágenes grandes (limitado por las capacidades de la memoria y el disco duro).

1.6 DESCRIPCIÓN DEL PROYECTO

Este proyecto es una tesis de desarrollo tecnológico que está dirigida principalmente a aquellas personas que suelen hacer experimentos de procesamiento de imágenes y aquellas con interés en usarlo como una plataforma de enseñanza o exploración en campos de aplicación afines. Se trata de construir una prueba de concepto sustancial del sistema que en su funcionamiento cumpla con los objetivos fijados.

1.7 ALCANCES Y LIMITACIONES

El énfasis de este trabajo es en la herramienta más que en sus aplicaciones, esto es, se le da más peso a la infraestructura de software que a las operaciones de procesamiento de imágenes y sus detalles de implementación.

Se procura reproducir la experiencia acumulada con las versiones anteriores de la calculadora a la vez que se favorece el crecimiento del catálogo de funciones que ofrece, dando más prioridad a esto último.

Concretamente, se implementa un conjunto de operaciones tal que sirve como evidencia del modo propuesto de composición de software y el funcionamiento de la arquitectura así como un asistente para añadir nuevas operaciones e instrucciones paso a paso para llevarlo a cabo.

En general el sistema está limitado principalmente por las capacidades de las plataformas de hardware y software en que se implementa.

Por simplicidad, con fines ilustrativos, se limita la implementación al cliente web Google Chrome y el servidor web Xitami, ambos para Windows. Es importante notar que muchas de las fuentes de información se basan en referencias a documentos en la web[3]. El código fuente en C debería ser compatible con los compiladores que observen el estándar[4] C99.

1.8 CONTRIBUCIONES

En el caso muy particular de un sistema de software para el procesamiento de imágenes, se plantea una arquitectura de software híbrida basada en tecnologías web para sustituir una arquitectura monolítica y aminorar el deterioro.

3 Se incluye este tipo de referencias por el fuerte empleo que se hace de tecnologías web en este trabajo y porque su contenido no ha sido vertido formalmente en libros. Siempre prefiriendo no incluir referencias a documentos de validez dudosa por la volatilidad de su presencia, se procuró elegir documentación que ha permanecido disponible en línea por mucho tiempo y que tiene gran aceptación en la comunidad de desarrolladores web, sin embargo, eso no representa ninguna garantía de que se pueda encontrar para su consulta en el futuro. Se ha cuidado que la validez de este trabajo no se vea comprometida por la ausencia de dichos documentos.

4 También conocido como ANSI C.

Se propone un desarrollo a partir de un modelo de composición de software por medio de una orquestación de sustituciones en plantillas basadas en texto.

La aportación principal es la ampliación del ciclo de vida de la calculadora de imágenes a través de la simplificación del proceso de añadidura de operaciones y de la apertura de la posibilidad de adaptar fácilmente su configuración para su uso en varios contextos.

Concretamente:

- se reduce la complejidad del proceso para añadir nuevas operaciones,
- se amplia la cantidad de formatos de imágenes y sus canales de color,
- se hace flexible la cantidad y tamaños de nichos para mostrar imágenes,
- se gana la posibilidad de ejecutar CALIMAN en otras plataformas distintas a Windows.

También se añaden algunas mejoras a las funciones accesorias existentes en la calculadora, entre ellas: la posibilidad de scripts con variables y herramientas para el análisis interactivo de imágenes.

2 CONTEXTO

En este capítulo se contemplan algunos trabajos afines con la intención de establecer una valoración respecto a otras soluciones existentes que le den pertinencia y validez a nuestro proyecto.

2.1 ANTECEDENTES

Este proyecto se originó de la necesidad de tener actualizado el programa CALIMAN del Centro de Investigación en Matemáticas, A.C. (CIMAT) y fue en el seno de la asignatura de Proceso de Señales donde se lanzó la propuesta de realizar una tesis de desarrollo tecnológico que tuviera como tema el remozamiento de dicho programa.

La posibilidad de actualizar CALIMAN por medio de tecnologías web invita a dar un breve vistazo al estado del arte de las aplicaciones relacionadas [1].

2.2 ESTADO DEL ARTE

2.2.1 *Aplicaciones web (Rich Internet Applications)*

Es fascinante ver la transición de algunos programas o aplicaciones, que usualmente han sido de plataforma de escritorio, a plataformas de internet, como es el caso del paso de Microsoft Office a estar disponible gratuitamente integrado al sitio web de cliente de correo Outlook.com (antes Hotmail) [Murray, 2011], o competidores que nacieron ya en esa plataforma como Google Docs [Steven Holzner, 2009]. De interés por nuestro tema se encuentran en internet algunos servicios de procesamiento de imágenes que además de hacerlas disponibles para su consulta pública, permiten hacerles algunos ajustes o retoques, así como aplicarles efectos o añadirles adornos como es el caso de Adobe Photoshop Express [Adobe Systems, 2012], o bien otros como IPOL [CMLA et al., 2012] que muestran artículos arbitrados con código fuente y ejemplos que se pueden ejecutar en línea.

Es importante remarcar que no es el caso que dichas aplicaciones suplan por completo a las tradicionales; si bien algunas están relativamente limitadas en cuanto a sus capacidades respecto a las correspondientes de escritorio, se trata de convenientes complementos que

1 Respecto al momento del cierre de esta edición.

ofrecen otras ventajas como ubicuidad o bajos costos de mantenimiento.

Lo que interesa de estas aplicaciones es que dejan de estar atadas a un sistema operativo o a una plataforma de hardware para poder hacer uso de ellas pues están construidas basándose en una separación explícita de responsabilidades en donde, a grandes rasgos, una parte sólo hace procesamiento, mientras que otra parte sólo hace despliegues. Esto es de gran valor en sistemas que tienen potencial para crecer rápidamente.

2.2.2 *Sistemas de administración de contenidos web*

Se trata de plataformas de desarrollo de sitios web que usan plantillas para presentar contenidos. Exhiben varias facetas personalizables para consumo del contenido por una parte y para su edición por otra. Son de amplia utilización en periódicos, catálogos en sitios web de tiendas por internet, así como en enciclopedias de contribución colectiva como Wikipedia [The Wikimedia Foundation, 2012] y sitios de bolsas de trabajo y anuncios clasificados.

Drupal [Buytaert, 2012] y Joomla! [Open Source Matters, 2012] son plataformas de renombre en este rubro por su madurez, la inmensa variedad de campos de aplicación, su amplia utilización pública y su bajo o nulo costo.

Lo interesante de estas plataformas es su empleo de plantillas de texto para mantener una imagen consistente en todo el sitio web mientras que abstraen las dificultades asociadas con la administración de los contenidos, su estructura y su presentación. Esto es de gran valor por los ahorros de mantenimiento y extensión que supone.

2.2.3 *Tecnologías de composición de software*

La idea de usar componentes para ensamblar grandes unidades funcionales ha tenido gran éxito en la industria electrónica como lo evidencia la amplia utilización de los circuitos integrados; aún así, su extrapolación a la industria del software ha encontrado obstáculos en su adopción por las dificultades que implica la flexibilidad inherente al software. Por poner un ejemplo, simplificado por motivos de exposición, se puede observar que en los componentes electrónicos es relevante, entre otros detalles, cuidar la coincidencia aproximada de sus voltajes y amperajes o atender sus tiempos de respuesta o que no se excedan ciertos umbrales o que no se acomoden en ciertas configuraciones; mientras que en los componentes de software, las estructuras y tipos de datos pueden cambiar mucho de un componente

a otro, e incluso de una versión a otra del mismo componente. Entonces, a menos que se congelen las versiones, es más difícil estandarizar los componentes de software que los componentes de hardware. No obstante, se han hecho intentos monumentales para echar a andar esa idea por los ahorros que suponen para la industria por lo que vale la pena dar una hojeada a sus principales exponentes que gozan de un amplio público.

JAVABEANS: Son clases escritas en el lenguaje de programación Java siguiendo una convención adecuada para que se comporten como componentes de software reusables. Algunos ejemplos de dicha convención son: usar constructores sin argumentos que favorezcan la instanciación de los objetos; el empleo de métodos con nombres estándar para ajuste y obtención de valores de ciertas propiedades para ofrecer una interfaz que permita tanto la manipulación de los mismos desde otros objetos así como su almacenamiento y posterior recuperación de su estado interno de manera independiente a la plataforma. Esto permite construir de manera relativamente sencilla programas de soporte tales como herramientas de configuración o inspección y edición de propiedades. Para dar lugar a su flexibilidad, suelen requerir gran cantidad de código de soporte[2] lo que resulta inconveniente para la especificación de tareas sencillas.

COM: Es un estándar para la construcción de archivos ejecutables basados en un modelo de componentes que encapsulan clases instanciables y que implementan interfaces a través de las cuales los objetos se comunican. Es independiente del lenguaje de implementación de las clases mientras se siga un estándar binario que especifica una distribución específica de las estructuras de datos en memoria, una convención de llamadas y un sistema de tipos [Polberger, 2009].

Lo que interesa de estas tecnologías es el hecho de que para el intercambio de datos entre componentes debe haber un contrato o protocolo bien definido. Esto es de valor al constituir una pista orientadora para implementar este tipo de ideas.

2.2.4 *Herramientas de procesamiento de imágenes*

Por último damos una ágil mirada a las herramientas similares a la que nos ocupa atendiendo principalmente a su catálogo de funciones y las posibilidades de programarlas, y de manera secundaria los grupos que las desarrollan así como su tipo de licencia de uso.

2 Una metáfora para dicho código de soporte son los andamios de la industria de la construcción por lo que suele conocerse como 'andamiaje'.

CALIMAN: es un programa de procesamiento de imágenes desarrollado por el Grupo de Ciencias de la Computación del CIMAT, en el marco del programa REDII auspiciado por el Consejo Nacional de Ciencia y Tecnología (CONACYT), es de distribución libre y está en su versión beta 0.1. El nombre mismo es un juego de letras para representar el nombre más largo CALculadora de IMAgeNes. Dicha calculadora ofrece un despliegue con varios nichos para albergar imágenes y una botonera o panel de botones similar al de una calculadora científica. Una rutina de trabajo típica consiste en primero seleccionar un nicho de destino y una operación de la botonera. Luego asignar valores convenientes a los parámetros de dicha operación sobre imágenes cuyo resultado pudiera ser almacenado en el mismo u otros nichos. También cuenta con la capacidad de programar la secuencia de operaciones por medio de un script.

Hay en el mercado varias herramientas muy maduras que se emplean para procesamiento de imágenes, entre ellas encontramos algunas muy relevantes, a saber:

PHOTOSHOP: Es el programa comercial de procesamiento de imágenes digitales más conocido en el mundo, está disponible en las plataformas PC y Mac OS, es desarrollado y comercializado por la empresa Adobe Systems que ofrece un Software Development Kit o Kit de Desarrollo de Software (SDK) para el desarrollo de plugins [3] en lenguaje C; además uno de sus plugins permite la programación de secuencias de procesamiento por lotes en forma de scripts. El acceso al catálogo de funciones que ofrece es principalmente a través de menús. [Brad Dayley, 2012]

GIMP: También conocido como GNU Image Manipulation Program es un programa de procesamiento de imágenes digitales cuyo código fuente está disponible gratuitamente en varias plataformas bajo la licencia GPL y también admite plugins para extender su funcionalidad, entre ellos uno llamado Script-Fu para construcción de secuencias de procesamiento por lotes. El acceso al catálogo de funciones que ofrece es principalmente a través de menús. [Jason van Gumster, 2010]

OPENCV: Es una biblioteca de funciones en C y C++ especializada en Visión Computacional que tiene rutinas de procesamiento digital de imágenes. Fue desarrollada inicialmente por Intel y ahora mantenida por el grupo de Willow Garage y está disponible en varias plataformas bajo la licencia BSD. El acceso al catálogo de funciones que ofrece es principalmente a través de inclusión de las bibliotecas en programas hechos en C o C++. [Gary Bradski, 2008]

3 Los plugins son programas con funcionalidades específicas que están construídos para ser embebidos en programas anfitriones.

HARPIA: Harpia es una herramienta, de uso gratuito para fines no comerciales, que ofrece una forma visual de programar algoritmos de procesamiento de imágenes empleando módulos prefabricados con soporte para emplear funciones de OpenCV y capturar video de dispositivos en GNU/Linux y MS-Windows. Es desarrollado por el grupo Industrial Intelligent Systems de la Universidad Federal de Santa Catarina en Brasil. El acceso al catálogo de funciones que ofrece es principalmente a través de una representación gráfica de un árbol del que se extraen bloques para agregarlos y conectarlos a otros bloques en un grafo por medio del mecanismo de arrastrar y soltar. [S2i-Industrial Intelligent Systems, 2007]

MATLAB: Presenta un entorno integrado de programación para computación numérica, visualización y análisis de datos que ofrece una gran variedad de métodos y permite emplearlos fácilmente, no obstante no los presenta de fábrica de manera intuitiva o sugerente como en una calculadora. Es desarrollado y comercializado por The MathWorks, Inc. El acceso al catálogo de funciones que ofrece es principalmente a través de llamadas a las mismas dentro de programas hechos en el lenguaje integrado.

LA BIBLIOTECA CIMG: Es una biblioteca de funciones en C++ gratuita y de código abierto distribuida bajo una licencia CeCILL-C compatible con la GNU LGPL por lo que puede ser usada en aplicaciones comerciales; esta biblioteca permite acceder a varios formatos de imágenes, acceder a sus pixeles, transformarlas y filtrarlas, dibujar texto, curvas, objetos 3D en ellas, calcular estadísticas de ellas, entre otras funciones; es ligera constando de un solo archivo de encabezados CImg.h y por ello funciona en sistemas operativos distintos bajo compiladores distintos, además de tener una estructura ordenada que permite incrementar las capacidades de la biblioteca. [Centre National de la Récherce Scientifique, 2012]

IMAGEMAGICK: Es un conjunto de programas de línea de comandos que se distribuye bajo la licencia Apache 2.0 y se puede usar para leer y escribir imágenes en más de 100 tipos distintos de formatos, ajustar sus colores, transformarlas o aplicarles efectos especiales así como dibujar líneas, polígonos, elipses y curvas Bézier en ellas. Entre sus funciones tiene un analizador de expresiones que describen operaciones en las imágenes lo que en cierta forma permite programar sus parámetros. [Montabone, 2010]

GRAPHICSMAGICK: Surgió como un trabajo derivado en una etapa temprana de ImageMagick por lo que comparte muchas de sus funciones pero se distribuye bajo la licencia MIT. Aunque tiene acceso a menos formatos que su predecesor, sus operaciones están imple-

mentadas para ejecución multi-hilos usando OpenMP. [GraphicsMagick Group, 2012]

Lo que interesa ver de todas las herramientas mencionadas es su capacidad para procesamiento de imágenes y la similaridad en sus catálogos de funciones. Además de que pueden acceder a varios formatos de imágenes y se pueden programar por medios incorporados a ellas o se pueden integrar en programas en la forma de bibliotecas o mediante llamadas a su conjunto de funciones a través de una Application Programming Interface o Interfaz de Programación de Aplicaciones (API). Finalmente muchas de ellas están disponibles para su uso sin tener que pagar regalías o costosas licencias, además de que se les pueden hacer modificaciones, adaptaciones o extensiones pues su código fuente está disponible también.

Se puede entonces apreciar el valor de una herramienta que facilite el acceso a muchas operaciones y que tenga capacidad para añadirle otras de ser necesario.

2.3 ALGUNAS CARACTERÍSTICAS DE CALIMAN

En esta sección enunciamos algunas de las características de CALIMAN que motivan los objetivos y contribuciones de este proyecto.

2.3.1 *Lo que CALIMAN es o tiene*

- Es un programa que emplea una metáfora de una calculadora para organizar un conjunto de operaciones de procesamiento de imágenes.
- Permite acceder a las operaciones por medio de botones y pasarles parámetros por medio de cajas de diálogo.
- En el mismo proceso de sistema operativo presenta la interfaz gráfica de usuario y efectúa el procesamiento de imágenes.
- Tiene un intérprete de scripts para programar secuencias de operaciones.
- Tiene una cantidad fija de nichos para mostrar imágenes en escala de grises (aunque originalmente fueran en color).

2.3.2 *Lo que CALIMAN no es o no tiene*

- CALIMAN no es Photoshop, esto es, aunque puede ser empleado como herramienta complementaria para actividades artísticas o de diseño gráfico, no es éste su propósito principal, ya que el

primero está más orientado principalmente a los sectores académicos o industriales mientras que el segundo tiene un público artístico o comercial.

- No es un producto multiplataforma, esto es, sólo puede ejecutarse en el sistema operativo Windows.
- No incorpora medios para añadir nuevas operaciones ni para cambiar la configuración del programa en tiempo de ejecución.
- Para incluir nuevas operaciones en el catálogo de CALIMAN hay que seguir un procedimiento bien establecido pero largo que involucra tener familiaridad con su código fuente. Concretamente, las modificaciones correspondientes se realizan en lugares dispersos, además de que exige al programador la habilidad para trabajar cómodamente con producciones gramaticales para ampliar el lenguaje del intérprete de script de CALIMAN. Es notable que dichas producciones gramaticales se basan en las correspondiente a expresiones numéricas del lenguaje C.
- Para modificar la disposición o tamaño de los nichos hay que recompilar el código fuente de CALIMAN.
- No lee ni escribe más formatos de imagen que el estándar BMP.

Esta lista de aspectos generales describe a grandes rasgos el conjunto de problemas a los que se les ofrece una solución en este trabajo.

3

MARCO REFERENCIAL

En este capítulo se exponen los conceptos en los que se basa este proyecto.

Se presentan algunas nociones acerca de las tecnologías involucradas, pasando por arquitecturas de software, sistemas distribuidos y tecnologías web hasta llegar a invocación de procesos, manipulación de cadenas de texto y procesamiento digital de imágenes. Al terminar se enuncia una lista de características deseables en el software.

Estas nociones dan sustento teórico al proyecto. La intención no es exhibir una pasarela de definiciones sin ton ni son, sino mostrarlas ordenadas siguiendo un hilo conductor de manera tal que gradualmente vayan motivando o dando sentido al esfuerzo que nos ocupa, en algunos casos indicando su relevancia o bien aterrizando las ideas con ejemplos enfocados a CALIMAN. Además cabe subrayar que se pueden encontrar explicaciones más profundas o elaboradas en las fuentes bibliográficas, aunque allá pudieran leerse fuera del contexto de nuestro interés[1].

3.1 SISTEMAS DE SOFTWARE

A grandes rasgos, la teoría de sistemas ofrece formulaciones conceptuales que aplicadas a la realidad permiten, por un lado analizar fenómenos complejos para hacerlos comprensibles y manejables; y por otro lado sintetizar entidades sofisticadas en términos de sencillos ingredientes.

En general, se considera sistema a un conjunto cuyos elementos interactúan para lograr un objetivo común actuando a través de procesos, consumiendo recursos y pasando por varios estados [von Bertalanffy, 2006].

En particular, el análisis de las computadoras bajo el enfoque sistémico arroja que hay acciones de sus sistemas componentes que se efectúan por medio de recursos y procesos tanto de hardware como de software, donde los correspondientes de hardware son difíciles de cambiar una vez fijos, mientras que los respectivos de software son susceptibles a cambios y es por esta flexibilidad que son de interés para este trabajo.

1 El propósito de mostrarlas es más informativo que normativo. Por ejemplo, al mencionar una línea de producción de software o el estándar Component Object Model o Modelo de Objetos de Componentes (COM) no significa que se va a construir una implementación de los mismos, sino que se va a tomar de ahí alguna idea valiosa para emplearla en nuestra propuesta.

Los procesos de software que tienen como meta resolver un problema se pueden enmarcar en una serie de pasos lógicamente ordenados en lo que se conoce como algoritmos.

Dichos pasos se pueden enunciar como sigue:

- Inicio o planteamiento del objetivo del algoritmo
- Definición de entradas o capturas de datos a alimentar.
- Aplicación de fórmulas, funciones, procedimientos para lograr transformaciones.
- Obtención de salidas o resultados parciales o finales.
- Verificación de condiciones de paro o retroalimentación al sistema.

Los algoritmos operan sobre entidades con características cuantitativas o cualitativas que son descritas a través de datos, esto es, por medio de valores tomados de los conjuntos de discurso de dichas entidades. Un dato es una representación simbólica o un atributo de una entidad.

En general, se emplean modelos para hacer tratable la complejidad de los fenómenos y poder decir algo de ellos, considerando un modelo como una representación simplificada de algún fenómeno que atiende, de entre sus características esenciales, las que son pertinentes al problema en que se esté trabajando.

Un modelo de datos es una representación de las entidades involucradas en los algoritmos, las operaciones para manipularlas y las relaciones entre ellas. Son operaciones típicas sobre modelos de datos su creación, eliminación, modificación y consultas a sus contenidos.

Para el desarrollo de estos sistemas de software es útil concebirlos en términos de modelos de datos y algoritmos para describir ciertas acciones o procesos de interés y poder considerarlos como componentes.

3.2 COMPOSICIÓN DE SOFTWARE

La aplicación de la teoría de sistemas a la descripción de las acciones de los sistemas de software fomenta la construcción estructurada de grandes unidades funcionales a través de la descomposición de algunos aspectos de los problemas en partes simples o más manejables y la posibilidad de su posterior composición para generar soluciones.

3.2.1 *Reutilización de artefactos*

Históricamente se ha buscado mejorar la producción del software por medio de la reutilización de diversos artefactos tales como:

- subrutinas a partir de 1960,
- módulos a partir de 1970,
- objetos a partir de 1980,
- componentes a partir de 1990,
- servicios a partir de 2000
- y recientemente con Líneas de Productos de Software[2] [Clements, 2001].

Entendiendo el mejoramiento de la producción como la obtención de beneficios organizacionales tales como:

- disminución de costos,
- disminución del 'time to market',
- reducción de riesgos,
- mayor calidad de los productos,
- ganancias en productividad,
- uso más eficiente de los recursos humanos,
- personalización en masa,
- incremento de la satisfacción del cliente,
- mantener presencia en el mercado,
- presencia en nuevos mercados.

Con tan imponente lista de beneficios se puede apreciar lo importante que es efectuar composiciones con artefactos reutilizables.

3.2.2 *Limitaciones de los Componentes*

La clave para la combinación exitosa de sistemas es la comprensión de las características de los ambientes exógenos de cada integrante en la forma de contratos, también conocidos como interfaces, que son listas de condiciones de intercambio de información. Por otro lado se puede llegar a tener una gran cantidad de módulos con requisitos muy dispares por lo que su composición puede tornarse difícil. Concretamente, con la estrategia de 'Divide y vencerás' se consigue una

2 Líneas de Productos de Software también conocidas como LPS están constituídas por software que comparte un conjunto administrado de características que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrolladas a partir de un conjunto de activos esenciales de una forma preestablecida. [Linda M. Northrop, 2012]

separación de intereses, pero aunque idealmente los intereses debieran estar localizados en los componentes se dan casos en donde hay intereses que se mezclan en diferentes componentes o aspectos, o se tienen componentes que satisfacen varios intereses[3], o bien la realización de un interés se encuentra dispersa en distintos componentes[4]. Para mantener un equilibrio en este proceso de composición/descomposición es que se plantean arquitecturas de software.

3.3 ARQUITECTURA DE SOFTWARE

Las composiciones de software orquestadas con propósitos definidos se conocen como arquitecturas de software. La arquitectura de un sistema es requisito para gozar del beneficio de reutilización de estructuras, algoritmos y otros componentes [Bosch, 2000].

Entre los propósitos para los que se emplea una orquestación de composiciones, de relevancia para este trabajo, es el manejo de situaciones en las que las interrelaciones entre los componentes están dispuestas de tal manera que cumpliendo ciertas restricciones exhiben su funcionamiento esperado pero que, bajo otras condiciones más relajadas, llegan a entorpecer el funcionamiento esperado del sistema. Esto suele suceder en situaciones donde se dejan fijos en el código fuente los valores de ciertas variables, sin posibilidad de cambiarlos desde afuera o que se adapten internamente al contenido presente en tiempo de ejecución. Por ejemplo, si se dejan fijas las dimensiones de despliegue de las imágenes y se permite que se abran imágenes de varios tamaños, sucede que algunas, cuyas proporciones corresponden a las de las dimensiones fijas, tendrán la apariencia esperada, mientras que las que no correspondan tendrán una apariencia desproporcionada.

Otro de los propósitos también importante es el manejo de las responsabilidades que tienen los componentes respecto a los aspectos que atienden. En ocasiones ocurre que hay componentes que tienen demasiadas responsabilidades por atender a una gran cantidad de aspectos. Más específicamente, hay algunas construcciones de software que se erigen con ciertos propósitos muy específicos en una primera instancia y que tras un tiempo de uso con éxito se observa que algún aspecto de su comportamiento pudiera ser reutilizado para otros propósitos pero que su transición hacia esos otros intereses se ve complicada por la forma en que sus componentes se ocupan de los aspectos. En otras palabras, si un sólo componente o un grupo muy reducido de ellos es responsable por el funcionamiento de todos los aspectos o de la mayoría de ellos, se considera a dicha construcción como de arquitectura monolítica, lo que es comparable por su falta de flexibilidad a una pieza de hardware; esta característica no es conveniente

3 Problema conocido como Tangling.
4 Problema conocido como Scattering

en todo aquello que se considere como software. Concretamente, si se tiene un cierto tamaño para un arreglo rectangular de campos de texto para introducir valores en una operación, como en el caso de la construcción de un kernel de convolución, y se tiene una función para el cálculo de la suma de todos esos valores y se desea emplear este artefacto con un arreglo rectangular de tamaño mayor, hay que implementar cada una de las conexiones desde cada uno de los nuevos campos de texto a cada una de las variables correspondientes en memoria para que almacenen su valor y luego actualizar la función que calcula la suma para tomar en cuenta las variables recién introducidas. Esto se haría para cada tamaño nuevo que se deseara de dicho arreglo rectangular. Una tarea de esta naturaleza resulta tediosa aún a corto plazo.

Algunos modelos de arquitecturas que procuran evitar la dureza mencionada de la arquitectura monolítica se enmarcan en lo que se conoce como arquitectura por capas, que pretende separar explícitamente las responsabilidades en capas de atención. De entre ellos, el modelo que es relevante para este trabajo es el de arquitectura de cliente-servidor.

3.4 ARQUITECTURA DE CLIENTE-SERVIDOR

Una arquitectura de software que considera explícitamente responsabilidades separadas es la de cliente-servidor, donde la parte del cliente, sirve de intermediaria entre el usuario y la parte del servidor; de manera tal que el cliente se limita a hacer solicitudes y mostrar resultados mientras que el servidor se encarga de los procesos que conducen a obtener dichos resultados [Microsoft Patterns & Practices Group, 2009]. A la parte cliente se le suele llamar front-end y a la parte servidor, back-end; en este trabajo se adopta esta última nomenclatura para distinguir los artefactos producidos de los instrumentos (servidor y cliente web) que se empleen para apoyar su funcionamiento como se podrá apreciar más adelante en la sección de discusión de la implementación.

3.5 SISTEMAS DISTRIBUIDOS

Entre las arquitecturas de software por capas, hay algunas aplicadas a la distribución tanto geográfica o física como lógica de los componentes de ciertos sistemas. Por su naturaleza se deben construir con mucho cuidado ya que en ellas es de importancia considerar la robustez ante imprevistos, la tolerancia a fallos e inconsistencias causadas por la distribución de componentes. Una de ellas, de amplia aplicación, se conoce como World-Wide Web [Andrew S. Tanenbaum, 2007].

3.6 WORLD-WIDE WEB

La World-Wide Web es un ejemplo de sistema masivamente distribuido que ha resultado muy exitoso. En esencia es un conjunto de computadoras interconectadas que comparten información a través del Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto (HTTP). Dicha información suele presentarse en lo que se conoce como páginas web. A un conjunto de páginas web relacionadas por venir del mismo proveedor y exhibir una misma apariencia se le conoce como sitio web. Al contenido típico de las páginas web se le conoce como hipertexto pues puede incluir medios no textuales como imágenes o fragmentos de audio y video además de hacer referencia a otras páginas web en un entramado semejante a una telaraña. A la actividad de seguir los vínculos o referencias suele llamársele 'navegar en las páginas web' en un guiño metafórico a los navegantes exploradores.

Es importante observar que en lo que se refiere a su consumo a través de un cliente web, las páginas web pueden considerarse como un medio de comunicación en donde el receptor es la persona que las consume, el medio es el World-Wide Web, el emisor es quien las produce, y ellas mismas son el mensaje; siendo además asíncronas y no-lineales, en el sentido de que en cualquier momento se les puede consultar y se puede escoger, por medio de la navegación, cuales solicitar respectivamente.

Los estándares y recomendaciones que se observan en las implementaciones de la World-Wide Web son usualmente administrados por una organización conocida como el Consorcio World-Wide Web o World Wide Web Consortium (W3C) por sus siglas en inglés y en otros casos por la Internet Engineering Task Force (IETF). En particular, la IETF difunde ciertos documentos conocidos como Request for comment (RFC), para invitar a la comunidad interesada a discutir las propuestas de tecnologías de internet. Por su parte, el W3C es una organización dedicada a promover y recomendar estándares para las tecnologías web que van desde protocolos, objetos, modelos, lenguajes e interfaces para interoperabilidad entre tecnologías de la información. Su relevancia para el proyecto reside en que éste se basa en algunos de sus instrumentos, principalmente, el cliente web y el servidor web, ya que soportan de manera conveniente la separación de responsabilidades que se necesita.

3.6.1 *Instrumentos de la World-Wide Web*

HIPERTEXTO: Es un formato de medios de información electrónicos basado en texto que se despliega en los clientes web conocidos como navegadores web con la posibilidad de establecer vínculos (también conocidos como hipervínculos) entre secciones o páginas y na-

vegar entre ellas, además de permitir incluir tablas, imágenes y otros medios como sonidos o videos.

PROTOCOLO HTTP: Se trata del Protocolo de Transferencia de Hipertexto cuya versión 1.1 está definida en el documento RFC2616. Controla la transferencia de datos desde y hacia un servidor web por medio de comandos como GET, POST o PUT.[Fielding et al., 1999] Está basado en texto y es de relevancia el manejo especial de cadenas con caracteres no estándar usando un código conocido como URL-encoding.

URL: Uniform Resource Locator o Localizador Uniforme de Recursos (URL) está definido en el documento RFC1738. Son cadenas de texto que sirven para representar de manera compacta la ubicación de los recursos que se pueden acceder vía Internet. En el contexto de la World-Wide Web se conocen como direcciones de páginas web. [Berners-Lee et al., 1994]

URL-ENCODING La especificación de los URL (RFC1738) indica que los caracteres permitidos en los URL se deben limitar a un subconjunto del US-ASCII, a saber:

- los alfanuméricos (ver Tabla 1)
- algunos caracteres especiales (ver Tabla 2)
- algunos caracteres reservados con significado especial en los URL (ver Tabla 3)

El resto de los caracteres debe codificarse usando una tripleta cuyo primer carácter sea % seguido de dos dígitos hexadecimales tomados de estos: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F en los casos en que su uso sea inseguro por que se puede modificar en su transmisión, como los caracteres de espacio que pueden aparecer o desaparecer dependiendo de la configuración de los agentes de transporte (v.g. el servidor web); o que no sean alfanuméricos, como los caracteres de control (v.g. tabulaciones, retorno de carro); o que no se encuentren entre los símbolos antes citados. En algunos contextos, a esta codificación también se le denomina *protección de símbolos*, pues el cambio de representación evita que se pierdan o modifiquen.

SERVIDOR WEB Es un programa que:

- está a la espera de conexiones y atiende las solicitudes hechas en el HTTP;
- usualmente envía los archivos especificados en los URL tomando como raíz un directorio especificado en su configuración;

- en algunos casos, dependiendo de la ubicación o de la extensión del archivo solicitado, en vez de enviar dicho archivo de inmediato, ejecuta un programa pasándole algunas variables de entorno especiales, espera un cierto tiempo a que termine su ejecución[5] y envía al cliente la salida que arroje dicho programa; esto último siguiendo las reglas de intercambio de información de lo que se conoce como Common Gateway Interface o Interfaz común de portales (CGI).

CGI Son las iniciales en inglés de la Common Gateway Interface que está definida en el documento RFC3875. Un servidor web que suele usarse como portal intermediario a un sistema de información tradicional como una base de datos requiere un contrato donde se especifiquen los parámetros descriptores de la solicitud hecha por el cliente, de manera que el servidor sea responsable de administrar la conexión, transferencia de datos y asuntos relacionados con la recepción de la solicitud del cliente mientras que el script o programa de CGI se encargue de los asuntos de la aplicación tales como acceso a datos o procesamiento de documentos [Robinson and Coar, 2004]. Concretamente, los datos enviados usando el comando GET de HTTP se pasan a los programas llamados por el servidor web a través de la variable de entorno `QUERY_STRING`[6], mientras que los datos enviados usando el comando POST de HTTP se les transfieren por medio de su entrada estándar. Finalmente el servidor web lee la salida estándar de tales programas para enviar su contenido al cliente web.

CLIENTE WEB Es un programa que:

- usualmente se comunica con un servidor web usando el HTTP;
- típicamente sirve de intermediario entre usuarios y servidores web (aunque algunas operaciones también pueden llevarse a cabo en un sistema de archivos local);
- del usuario recibe e interpreta cadenas de texto o movimientos y pulsos del ratón o dispositivo apuntador, o bien gestos de contacto en una pantalla táctil, según los dispositivos de entrada que estén disponibles;
- de la interpretación que hace de las entradas proveídas por el usuario ejecuta acciones de navegación entre páginas;
- al usuario presenta los contenidos de archivos, ya sean interpretados o no, dependiendo de su formato y los plugins instalados

5 Usualmente, dependiendo de la configuración del servidor, en caso de que el tiempo de espera del servidor sea menor al tiempo de ejecución del programa llamado, el servidor mata el proceso de dicho programa y envía al cliente un mensaje de error con el código 500 (Internal Server Error).

6 Estas variables de entorno se pueden leer mediante la función `getenv` en programas en lenguaje C.

o capacidades internas que dan soporte al despliegue de distintos tipos de formatos;

- pudiera tener una máquina de ejecución de scripts en lenguajes como Dart, VBscript o Javascript.

El caso de uso más típico consiste en que el usuario introduzca en el cliente web las cadenas de texto conocidas como URL o direcciones de páginas web para que enseguida se hagan las solicitudes correspondientes a un servidor web que las atienda; seguido de una espera mientras se transfieren los archivos producto de la respuesta para que finalmente el cliente web haga una interpretación del contenido de dichas páginas web y las exhiba al usuario. También hay clientes web sin interfaz gráfica de usuario aunque usualmente forman parte de programas que exploran de manera automática la World-Wide Web con fines de indexado, verificación de enlaces y creación de sitios espejo, entre otros.

Los navegadores web son programas muy sofisticados que atienden los aspectos de presentación, estructura, contenido y comportamiento de las páginas web. Cuando en este trabajo se usa el término navegador web se hace referencia a aquellos clientes web que tienen una interfaz gráfica de usuario y una máquina de ejecución de scripts en Javascript, por lo que el término cliente web se emplea de manera más general.

DOM El navegador web expone algunos de los componentes de su interfaz de usuario como objetos del lenguaje Javascript en lo que se conoce como Document Object Model o Modelo de Objetos de Documentos (DOM) por sus siglas en inglés. El DOM[7] está representado internamente en una estructura jerárquica de árbol. Es muy importante remarcar que existen unas funciones Javascript que permiten acceder y modificar el DOM en tiempo de ejecución. Una de las más importantes es `document.getElementById` pues al pasarle la cadena que identifica a un elemento del DOM regresa un apuntador o referencia al nodo DOM correspondiente y con esta referencia se pueden acceder los valores de sus propiedades. Si se escribe un valor en una de dichas propiedades el cambio se ve reflejado en el despliegue correspondiente a ese elemento DOM como se explicará posteriormente.

7 El DOM es una recomendación de la W3C y cada navegador implementa su propia interpretación de dichas recomendaciones, de manera que entre navegadores hay elementos en común y otros muy particulares no estándar, generalmente introducidos para tener ventajas competitivas por lo que, para fijar ideas, este trabajo se centra en el DOM del navegador Google Chrome, que a la fecha de cierre de esta edición es uno de los más avanzados en cuanto a cumplir con dichas recomendaciones en sus implementaciones. Debido a ello puede haber características que funcionen bien en Chrome, pero que no funcionen o lo hagan parcialmente en otros navegadores, además no es el propósito de este trabajo que la implementación propuesta se desempeñe igual en todos los navegadores porque es poca la aportación a la solución del problema planteado respecto al esfuerzo que se emplee para lograrlo.

JAVASCRIPT Javascript es un lenguaje de programación que, incorporado en el navegador web, atiende el aspecto del comportamiento de los componentes de la interfaz gráfica. De sumo interés para el proyecto es que brinda la capacidad de modificar propiedades de los elementos de las páginas web que estén expuestos por medio del DOM; su sintaxis [ECMA-ST, 2011] es similar a la del lenguaje C [ISO/IEC, 2010] aunque sus tipos de datos se toman del contexto y no hay que declararlos explícitamente salvo en contadas ocasiones. Además las funciones son también asignables a variables ganando con ello un comportamiento dual como objetos y funciones.

Hay en Javascript una función denominada `eval` cuyo argumento es una cadena de texto que es interpretado como código fuente Javascript. Entonces, si se tiene la cadena `"4+5"`, el resultado de `eval('4+5');` es el valor numérico 9; o si se tiene la cadena `"var a = 5;"`, el resultado de `eval('var a=5;');` es una variable de nombre `a` con el valor 5 asignado. Esto es muy conveniente porque permite que se construyan cadenas de texto a modo, representando fragmentos de programa que se pueden inyectar al programa en ejecución teniendo vigencia inmediata.

Para nuestros propósitos es una gran ventaja que un intérprete-compilador de este lenguaje esté integrado al navegador pues la compilación es automática y para echar a andar las actualizaciones a los programas basta con volver a cargar la página, además de que los navegadores modernos cuentan también con poderosas herramientas de depuración y perfilación integradas. Una desventaja es que por razones de seguridad en los navegadores web no hay soporte inmediato para almacenamiento local de datos en disco, aunque esto se puede lograr indirectamente a través de una transferencia de los mismos vía un servidor web a un programa que esté escrito en algún lenguaje que sí tenga dicho soporte[8] y que esté encargado de grabarlos o leerlos; en ese caso, dicho programa intermediario es lanzado a ejecución por el mismo servidor web.

CSS Las Cascading Stylesheets u Hojas de estilo en cascada (CSS) atienden el aspecto de la apariencia de la interfaz gráfica de las páginas web. Por medio de las especificaciones dadas en ellas se puede dotar de distintas presentaciones a la misma estructura y contenido[Lie and Bos., 1999]. Las hojas de estilo son listas de selectores de elementos de documentos Hypertext Markup Language o Lenguaje de Marcado de Hipertexto (HTML) seguidas de especificadores de valores de propiedades de presentación. Los ejemplos más típicos de su empleo son para indicar los colores, tamaños, tipos y estilos de las letras, esto es, se indica si deben aparecer gruesas, o inclinadas o con adornos o de algún tamaño una y las demás de otro. También se emplean para distribuir o determinar la posición de los componentes de

8 C tiene soporte para acceso al sistema de archivos.

la interfaz gráfica en la pantalla. Para los propósitos de este trabajo, por medio de CSS se puede imitar la apariencia de la calculadora en un navegador web.

HTML HTML, es un formato de datos basado en texto que atiende el aspecto de dar estructura a los documentos web por medio de marcado de segmentos de texto con etiquetas especiales. Dichas etiquetas están delimitadas por los caracteres < y >. Hay etiquetas que consisten en un par de marcas, una de inicio y otra de cierre; también hay otras etiquetas que no envuelven texto y la marca de cierre está incluida en la misma etiqueta.

Por ejemplo, una etiqueta que hace que los caracteres envueltos se desplieguen en negritas en el navegador web es `<b>`; de manera que el texto `CALIMAN` envuelto en dicha etiqueta sería `<b>CALIMAN</b>`, donde `<b>` delimita el inicio de la acción de la marca y `</b>` indica el fin de la acción de la marca; la acción del navegador web al toparse con esta marca es mostrar el texto `CALIMAN` en negritas en la pantalla. Un ejemplo de etiqueta que tiene la marca de cierre junto con la de inicio es `<br/>` que provoca en el despliegue un salto de línea. Esto se puede apreciar en la Figura 1a.

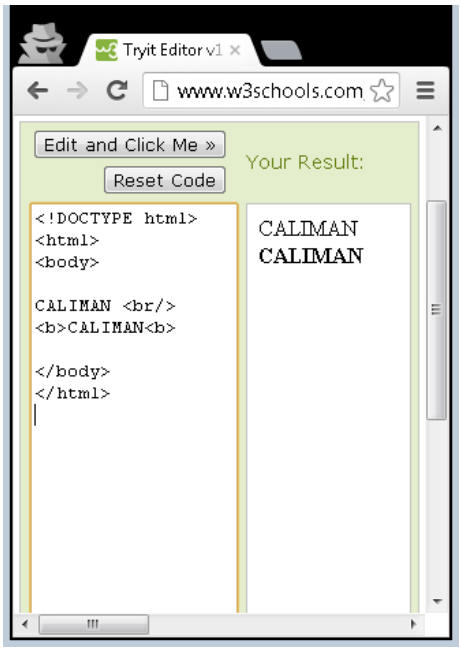
Las marcas pueden tener atributos o modificadores que son especificados por medio de pares (nombre de atributo, valor del atributo). La asignación de valores a los atributos se representa con el signo = y suele rodearse el valor con entrecomillado sencillo o doble, en particular cuando contiene espacios, a saber: `nombreDeAtributo="valor de atributo con espacios"` o `nombreDeAtributo='valor de atributo con espacios'`, o bien `nombreDeAtributo=valorSinEspacios`.

Un caso muy notorio de uso de atributos se presenta en los hipervínculos o referencias a otras páginas web por medio de la etiqueta `<a>` con su atributo `href`. Concretamente, si se conoce que la dirección de una página web es `http://localhost/` se puede incluir una referencia a esa página web en la página actual como se ilustra a continuación: envolver con la etiqueta `<a>` una cadena de texto representativa de la página a la que se quiere hacer referencia, digamos `otra página` y asignar la dirección dada a su atributo `href`, de manera que quede:

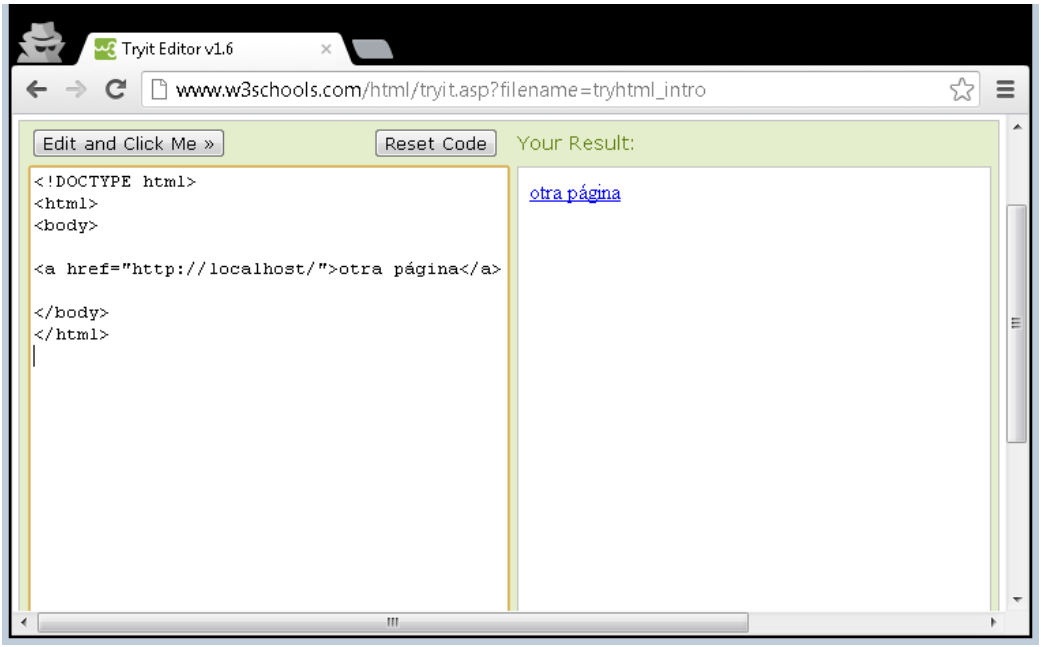
`<a href='http://localhost/'>otra página</a>`.

La acción del navegador web al toparse con esta marca es mostrar el texto `otra página` con una marca especial en la pantalla (como un subrayado) para indicar que se trata de un hipervínculo, luego el usuario puede desplazar ahí su foco de acción y presionar alguna tecla o algún botón en su dispositivo apuntador para provocar el desencadenamiento de la acción de solicitar esa página web al servidor correspondiente.

Esto se puede apreciar en la Figura 1b.



(a) Etiquetas BR y B.



(b) Etiqueta A.

Figura 1: Ejemplos de etiquetas HTML.

Por medio de la etiqueta `<script>` es posible adjuntar programas a las páginas web, casi siempre en el lenguaje Javascript, para su ejecución en el navegador web.

Uno de los atributos comunes a todos los elementos HTML que es de relevancia para este trabajo es el atributo `id`. Cada atributo id de-

be ser único[9] en cada página web. Por medio de los atributos id se puede hacer referencia directa a los elementos HTML de las páginas web en el contexto de ejecución de los programas en Javascript que se adjunten de manera que se pueden tratar a dichos elementos HTML como variables objeto en dentro de los scripts. Esto se logra por medio de la función `getElementById(id)` de Javascript que devuelve, como objeto accesible mediante Javascript, un nodo del modelo de objetos del documento correspondiente al elemento HTML cuyo atributo[10] id coincide con el dado. La importancia de esto radica en que modificar los valores de sus contenidos o los de sus atributos tiene consecuencias apreciables inmediatamente en su representación o despliegue en pantalla. Este mecanismo es un pilar fundamental de este trabajo.

Respecto a los objetos obtenidos de esta manera, son sus atributos `innerHTML` e `innerText` los que se llevan el crédito pues al usarlos en una expresión Javascript como argumentos de lectura devuelven cadenas de texto, pero al usarlos como argumentos de escritura desencadenan en el navegador web un mecanismo de interpretación y despliegue similar al que se da cuando se carga la página HTML de manera que se puede modificar la página web total o parcialmente al vuelo, esto es, sin cargarla completa de nuevo desde archivo.

Concretamente, suponiendo que se tiene el siguiente código HTML que designa una región de una página web:

`<span id='regionDePagina'></span>` y se obtiene su representación como objeto a través del Javascript:

`var objeto = document.getElementById('regionDePagina');`, entonces se puede modificar su contenido al asignar un valor a su atributo `innerHTML`: `objeto.innerHTML = 'CALIMAN';` de manera que por medio de los mecanismos antes descritos, el código HTML correspondiente queda: `<span id='regionDePagina'>CALIMAN</span>` y esto provoca que aparezca el texto `CALIMAN` en la posición correspondiente en la página web.

Se observa que lo que sea que esté envuelto por las marcas de inicio y cierre de las etiquetas puede ser accedido por medio del atributo `innerHTML` o (si se desea ignorar las etiquetas HTML internas) `innerText`. Entonces, si se modifica de manera conveniente el contenido de alguno de ellos se podrá apreciar el cambio de inmediato en pantalla; esto se aprovecha con gran beneficio en este proyecto.

Hay etiquetas que corresponden con componentes de interfaz gráfica de usuario que permiten la entrada de datos y los almacenan en su atributo `value`; de ahí se pueden tomar y almacenar en varia-

9 Si los hay repetidos, dependiendo de la implementación del navegador web, suele considerarse el primero en orden de aparición o bien continuar la ejecución como si no existiera dicho atributo id.

10 Lo que en HTML se denomina atributo es, en su representación como objeto Javascript, una propiedad o dato miembro.

bles en los programas de Javascript. De entre estas etiquetas, las más socorridas son `<input>`, `<select>` y `<textarea>`.

3.6.2 *Instrumentos de interactividad*

En abstracto, con la consideración de las páginas web como un medio de comunicación asíncrono y no lineal, se presenta la interactividad como una capacidad de control de sus mensajes (las páginas web) por parte del receptor (el navegador web) dentro de los límites del medio (su representación en el DOM) hasta el nivel establecido por el emisor (a través de la identificación de elementos por medio del atributo id). Concretamente, en nuestro caso, es de interés poder hacer modificaciones locales de las páginas web, sin tener que solicitar dichos cambios a un servidor web como ocurría en versiones anteriores del World-Wide Web, en donde los navegadores web eran meros receptores pasivos de dichas páginas; luego, con la capacidad de modificarlas dentro del navegador web se puede afirmar que se goza de interactividad local. Esto es de interés para este trabajo por el papel que juega en el cumplimiento de los objetivos propuestos. Más específicamente, con lo que se ha explicado acerca de la posibilidad de modificar lo que se muestra en pantalla a través de la debida identificación de los componentes de la interfaz de usuario y su manipulación con programas en Javascript, se tiene esa capacidad de controlar lo que ocurre en las páginas web sin depender completamente de un servidor web para lograrlo. Esto se consigue a través de los siguientes instrumentos:

EVENTOS DEL NAVEGADOR Son las respuestas del navegador a los estímulos externos como presión de teclas o movimientos y pulsos del ratón o gestos de contacto en una pantalla táctil o a señales de terminación de carga de documentos web.

MANEJADOR DE EVENTOS Son funciones que se asocian a eventos por un mecanismo similar a apuntadores a funciones de manera que el navegador les mande llamar cuando ocurra el evento al que está vinculada.

ESCUCHADOR DE EVENTOS Son propiedades de algunos objetos del DOM que tienen representación gráfica, se pueden considerar como apuntadores a funciones que serán llamadas cuando ocurran los eventos.

Cada tipo de elemento tiene un catálogo propio de eventos a los que se puede suscribir. Suscribirse a un evento significa asignar un manejador de eventos a un escuchador de eventos

Por ejemplo los botones tienen escuchadores de varios eventos; de interés para este proyecto es el evento `Click`, que ocurre cuando el

usuario pulsa el botón en la interfaz gráfica o cuando el foco de acción está ubicado en cierto botón y el usuario pulsa la tecla `[Enter]` o una equivalente en el teclado. También son de interés los eventos `KeyDown`, `KeyPress` y `KeyUp` que ocurren en distintos momentos durante la presión de teclas cuando el foco de acción está en cierto control[11] de entrada de datos.

OBJETO XMLHTTPREQUEST Es un objeto del entorno Javascript del navegador web que se encarga de la transmisión vía HTTP de mensajes cuyo contenido usualmente se espera que sea Extensible Markup Language o Lenguaje de Marcado Extensible (XML), aunque puede ser texto plano. Lo importante es que las respuestas pueden ser almacenadas en variables de Javascript y esto permite la carga de valores traídos del servidor en las variables.

JAVASCRIPT OBJECT NOTATION También conocido como Javascript Object Notation o Notación de Objetos de Javascript (JSON), es una manera de escribir objetos en texto. Junto con la función `eval` o el objeto JSON de Javascript permite inyectar objetos al programa en curso en tiempo de ejecución. La notación es `{`, `}` para designar los objetos; `:` para separar los nombres de las variables de sus valores; `[`, `]` para designar los arreglos; `"` o `'` para envolver las cadenas, y `,` para separar elementos. Por ejemplo se puede obtener un objeto en Javascript que tenga una variable de nombre a con valor 5, una cadena de nombre b con valor `'xy'` y un arreglo de nombre c con cuatro elementos, uno numérico con valor 7, el siguiente con un arreglo vacío, seguido de una cadena `'z'`, y un objeto anónimo vacío, con la notación `{'a':5,'b':'xy','c':[7,[],'z',{}]}`. Suponiendo que dicho objeto se asignó a la variable `obj` se puede acceder a la cadena `'z'` dentro del arreglo mediante la expresión `obj.c[2]`.

3.7 CADENAS DE TEXTO

Las cadenas de texto son secuencias indexadas de símbolos tomados de una colección finita de ellos [Aho, 2006; Kelley, 1995; García et al., 2001]. Por convención el primer símbolo de las secuencias tiene índice cero. Las subsecuencias cuyos índices sean consecutivos se conocen como subcadenas. La longitud de una cadena es la cantidad de símbolos en su secuencia. La cadena de longitud cero se llama cadena vacía. Tiene sentido considerar el símbolo i-ésimo de una secuencia o bien especificar un símbolo o subcadena y encontrar el índice de su ocurrencia en la secuencia.

11 Los componentes de la interfaz gráfica de usuario con los que se puede interactuar se conocen como controles. Por ejemplo los botones y campos de texto son conocidos como controles de entrada de datos.

Para fijar ideas sea la cadena `CALIMAN`. Su longitud de cadena es 7. Su primer elemento, de índice 0, es la letra C. Su cuarto elemento, de índice 3, es la letra I. La subsecuencia de índices 2,3,4,5 es la subcadena `LIMA`. Por otra parte el índice de la subcadena `IMAN` es 3, que corresponde a la posición su primer carácter en la cadena en consideración.

Se clasifican los caracteres por su naturaleza en alfanuméricos y no alfanuméricos. Los caracteres alfanuméricos se clasifican a su vez en caracteres numéricos y caracteres alfabéticos. Por su parte los caracteres alfabéticos se pueden dividir en caracteres de letras mayúsculas y caracteres de letras minúsculas, o en caracteres acentuados y caracteres no acentuados. Otras clasificaciones, de haberlas, no son relevantes para este trabajo. Para nuestros propósitos, la clasificación dada es valiosa para el caso de que una cadena de texto contenga una lista de palabras separadas por ciertos caracteres no alfanuméricos cuyas posiciones se pueden obtener para con ellas separar la cadena dada en subcadenas que contengan a una sola palabra cada una, mismas que pueden ser almacenadas en un arreglo para procesamientos posteriores; esta situación es parte importante del proceso propuesto como se verá más adelante.

Entonces, con las secuencias de símbolos se pueden formar nuevas secuencias de símbolos extrayendo subcadenas o poniendo una secuencia antes de otra en lo que se conoce como concatenación o bien especificando un símbolo o subcadena y reemplazándolo por otra cadena ya sea en su primera ocurrencia o en todas sus ocurrencias. Todas estas manipulaciones de cadenas de texto son de interés fundamental para este proyecto por su potencia en los instrumentos de la World-Wide Web que están basados en texto.

Hasta aquí se han presentado, de cierta forma encadenados, ciertos conceptos que hablan sobre los beneficios de usar las tecnologías web con un énfasis especial hacia el lado del cliente. No obstante, en la parte del servidor hay también algunos conceptos que es importante tener presentes.

3.8 PROGRAMAS Y MÓDULOS

Considerando un programa como un conjunto de instrucciones que se ejecutan en una computadora. Los programas de computadora se pueden analizar para su estudio en módulos, entendiendo un módulo como una porción de un programa de computadora que, por las tareas que realiza, se puede considerar independiente sin perder la capacidad de comunicación, por medio de entradas y salidas, con el resto de los módulos.

SISTEMA OPERATIVO Es un conjunto de programas que tienen como objetivo administrar los recursos de un sistema de cómputo a la vez que sirven como una capa de abstracción de los detalles de control del hardware para los usuarios [Andrew S. Tanenbaum, 1997]. Entre sus tareas de administración está la creación y terminación de procesos así como la planificación de su ejecución, la asignación y liberación de recursos de memoria y la gestión de acceso a archivos. Es pertinente considerar sus mecanismos porque finalmente tanto los navegadores como el servidor web son programas que se ejecutan en el contexto de un sistema operativo.

PROCESOS En el contexto de los sistemas operativos son las actividades que tienen un programa, estado y contexto de ejecución, así como entradas y salidas.

3.8.1 *Inicio de procesos*

A reserva de que los detalles varían de un sistema a otro, por lo regular los sistemas operativos crean los procesos asignando espacio en memoria para el código ejecutable y datos del proceso, manipulando tablas de control de procesos y cargando el código junto con las rutinas del sistema junto con los datos iniciales y una pila del proceso que contiene el entorno del proceso y los parámetros que se pasan en la invocación del programa; esto último lo realiza un módulo cargador que debe estar incluido en el código ejecutable.

3.8.2 *El módulo cargador*

En la compilación de todos los programas, durante su fase de enlazado, el programa enlazador añade un módulo que constituye el punto de entrada a la ejecución del programa y cuyo propósito es ejecutar rutinas preparatorias y de verificación (tales como constatar que haya suficiente espacio en memoria para almacenar el programa, asignar el espacio para las variables estáticas o globales; establecer el vínculo con los archivos estándar `stdin`, `stdout`, texttstderr; fijar referencias a las variables de entorno del sistema operativo, entre otras) para posteriormente comenzar la ejecución de las instrucciones dadas por el programador a partir de una función que se debe llamar `main`, pasándole datos que ha recibido a su vez del sistema operativo.

3.8.3 *Paso de datos en la invocación a procesos*

Por lo regular la invocación de procesos se efectúa por medio de una interfaz de usuario o shell que permite a los usuarios especificar por medio de una cadena de caracteres de texto tanto el nombre del archivo ejecutable como algunas palabras separadas por espacios que

podrán ser accedidas desde la función `main`, (excepto los que están rodeados de comillas ya que en ese caso los espacios incluidos entre las comillas de inicio y de cierre no se consideran como separadores de valores).

Con esta presentación de conceptos se tienen cubiertos ambos flancos, a saber: el del lado del servidor y el del lado del cliente. Falta completar el marco que da cimiento a este trabajo con la presentación de las siguientes generalidades.

3.9 SISTEMAS DE VISIÓN

Se refiere a los sistemas que involucran la solución de un problema de naturaleza visual en el que se intenta crear un modelo del mundo real a partir de imágenes del mismo. Generalmente se componen tanto de dispositivos de hardware como de módulos de software y en ellos se consideran procesos con etapas tales como adquisición de imágenes, pre-procesamiento de imágenes, extracción de características, hasta toma de decisiones y actuación/retroalimentación inteligente (como en aplicaciones de robótica móvil). Debido a ello se considera que integran en una única solución una serie de tecnologías diferentes permitiendo gran flexibilidad en el desarrollo de aplicaciones en distintas áreas del conocimiento. Suele usarse un esquema de tres niveles para estudiarlos, a saber:

VISIÓN COMPUTACIONAL Se refiere al procesamiento digital de imágenes para la extracción de características importantes que ayuden a la comprensión de las mismas y la subsecuente toma de decisiones. En el contexto de los sistemas de visión se consideran estas operaciones como de alto nivel, siendo común el empleo de técnicas de inteligencia artificial para el procesamiento de los datos. Un ejemplo sería el reconocimiento y la clasificación de un objeto en una imagen.

ANÁLISIS DE IMÁGENES Se refiere al procesamiento digital de imágenes donde las entradas del proceso solamente son imágenes y las salidas una descripción o representación diferente de la imagen o alguna característica suya. En el contexto de los sistemas de visión se considera que estas operaciones se encuentran en el nivel intermedio entre el procesamiento de imágenes y la visión computacional. Un ejemplo sería la visualización de los datos de una imagen por medio de un histograma que describa sus características de luminosidad.

PROCESAMIENTO DE IMÁGENES Se refiere al procesamiento digital de imágenes por medio de una computadora o dispositivo electrónico tal que las entradas y salidas del proceso son imágenes. En el

contexto de los sistemas de visión se considera a este tipo de operaciones como de nivel bajo o medio. En general se trata de operaciones de realce, restauración, transformación o compresión de imágenes. Un ejemplo sería la aplicación de un filtro a un imagen para realzar los bordes de los objetos ahí representados. Por otra parte los resultados del procesamiento de imágenes se basan en las propiedades de las mismas o son producto de secuencias de pre-procesamiento que tienen como objetivo mejorar el aspecto de las imágenes o hacer más evidentes en ellas algunos detalles como bordes o esquinas mediante operaciones morfológicas, operaciones de segmentación, el tratamiento estadístico, el realce de características, entre otras.

Este último nivel corresponde al interés de nuestros propósitos ya que las entradas y salidas de las operaciones que realiza CALIMAN son imágenes por lo que sólo resta hablar de manera general como representarlas y como operar con ellas.

3.10 REPRESENTACIÓN DIGITAL DE LAS IMÁGENES

Una imagen se define como una función bidimensional $f(x, y)$ que mapea posiciones discretas en un plano con sus intensidades correspondientes; a cada posición del dominio se le denomina elemento de imagen o pixel y las intensidades se representan en la computadora por medio de arreglos rectangulares de valores numéricos [Rafael C. Gonzalez, 2009]. Hay varios formatos o disposiciones de dichos arreglos que dependen de la cantidad de bits empleada para almacenar la información del nivel de intensidad de cada posición. Dichos formatos suelen estar asociados con modelos de color. Usualmente se emplean formatos de 8 bits por pixel, lo que proporciona una capacidad de 256 posibles valores que suelen presentarse como niveles de gris, donde por convención el valor 0 corresponde al más oscuro y el valor 255 al más claro; también se emplean formatos de 1 bit por pixel, siguiendo la convención de asignar el valor 0 al negro y el valor 1 al blanco; también, usando el modelo aditivo[12] RGB se pueden mezclar tres imágenes en escala de grises asignando cada una a un canal de color. Esta última es la configuración empleada por los formatos estándar de imágenes de más amplia utilización en la industria tales como el formato de archivo de imagen Joint Photographic Experts Group (JPEG), el formato de archivo de imagen Compuserve Graphics Interchange Format (GIF), el formato de archivo de imagen Portable Network Graphics (PNG), el formato de archivo de imagen Tagged Image File Format (TIFF), el BMP entre otros.

12 El modelo aditivo Modelo de Color (Red Green Blue) (RGB): Se emplea para sintetizar colores mediante la mezcla por adición de los colores rojo, verde y azul, suelen emplearse 8 bits por canal de color por lo que pueden sintetizarse $2^8 \cdot 2^8 \cdot 2^8 = 16777216$ colores.

3.11 OPERACIONES CON IMÁGENES

La literatura de operaciones con imágenes es muy amplia [Jahne, 2008], en general hay varios tipos de operaciones que se pueden realizar con las imágenes de acuerdo al dominio de discurso que se emplee que puede ser dominio de las frecuencias o dominio espacial. Las operaciones en el dominio espacial se realizan directamente sobre los pixeles de la imagen, mientras que las operaciones en el dominio de las frecuencias incluyen una aplicación de la Transformada de Fourier directa antes del proceso, e inversa después del proceso. Para nuestros propósitos lo importante es el concepto de que para efectuar las operaciones hay que tener acceso a las intensidades de los pixeles, ya sea individualmente o mediante ventanas centradas en un pixel; además que las operaciones se pueden clasificar de acuerdo a la cantidad de operandos involucrados. Por ejemplo, una raíz cuadrada de una imagen sólo involucra a una imagen origen y una imagen destino, mientras que una suma de imágenes involucra dos imágenes de origen y una de destino.

3.12 CARACTERÍSTICAS DESEABLES EN EL SOFTWARE

En esta sección se enuncian ciertas características[13] deseables en todo producto de software que sirven de guía para su implementación. En suma el principal beneficio de buscar satisfacerlas es extender el ciclo de vida del software [Pressman, 2002; Sommerville, 1992]:

MODULARIDAD: debe ser fácil modificar una parte sin afectar a otras.

EXTENSIBILIDAD: debe poder adaptarse a cambios o extensiones de su funcionalidad mediante componentes aislados o con bajo acoplamiento.

SER INTUITIVO: debe seguir el principio de la mínima sorpresa, esto es, las posibilidades ofrecidas al usuario en cada instante deben ser evidentes y los controles manipulables no deben desempeñar acciones inesperadas u ocultas.

FÁCIL DE USAR: debe ser cómodo ingresar datos o revisar resultados.

CORRECCIÓN: debe haber facilidad para solucionar los problemas que puedan presentarse, en ocasiones forzando los mecanismos de en-

13 Esta lista es sólo una guía, esto es, sólo es informativa y no es normativa, pues en la práctica las prioridades se determinan en base a los requisitos recopilados durante las fases iniciales del proyecto.

trada para que el usuario escriba lo menos posible para evitar errores o corrigiendo en la medida de lo posible algunos errores de entrada de datos o que definitivamente no se acepte entradas inválidas.

ROBUSTEZ: debe funcionar incluso en situaciones anormales (dependiendo de lo que se considere normal).

EFICIENCIA: debe tomar preferencia por hacer un buen uso de los recursos del ordenador.

PORTABILIDAD: debe poder ser transportado sobre diferentes sistemas físicos o lógicos, esto es, en diferentes plataformas hardware o software.

INTEGRIDAD: debe de contar con protección de sus propios componentes contra los procesos que no tengan el derecho de acceder.

COMPATIBILIDAD: debe ser relativamente fácil combinarse con otros programas.

REUTILIZACIÓN: debería ser, en principio, posible reutilizar sus componentes, en su totalidad o en parte, en nuevas aplicaciones.

VERIFICABILIDAD: debe ser inmediato probar que el software funcione correctamente.

Todas estas definiciones en su conjunto dan cuenta de las características que debe tener un sistema computacional de procesamiento de imágenes que se base en una arquitectura que prefiera la separación explícita de responsabilidades y son de relevancia para este trabajo pues apoyan la formulación de la propuesta de solución a los problemas detectados. En concreto, por un lado se pretende aprovechar la capacidad de modificar en tiempo de ejecución los elementos de una interfaz gráfica basada en cadenas de texto a través de la manipulación de dichas cadenas; por otro lado explotar la facilidad de pasar cadenas de texto a los programas al invocarlos con el fin de ajustar los valores de ciertas variables de control de su comportamiento; todo ello gracias a la maleabilidad de las cadenas de texto.

Parte II

PRAXIS

En esta parte se presenta la propuesta de solución al problema, sus pormenores y los resultados obtenidos.

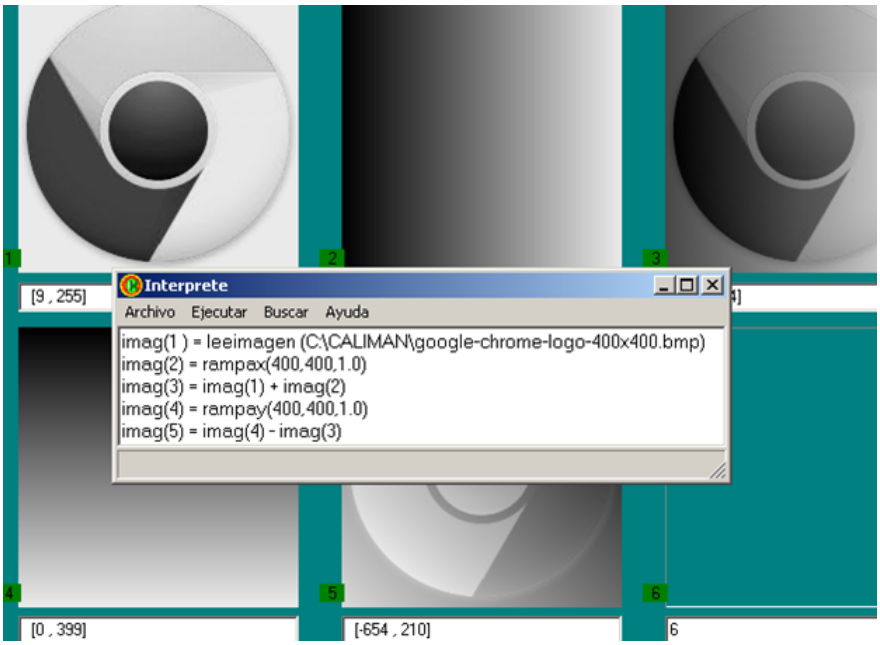
4

LA SOLUCIÓN PROPUESTA

En este capítulo se expone la arquitectura propuesta motivo de este trabajo y se discuten las razones de las decisiones tomadas. Se da una descripción detallada de las piezas de la línea de producción mediante el seguimiento paso a paso de un ejemplo que ilustra la manera en que se desempeñan los artefactos componentes de la solución propuesta.

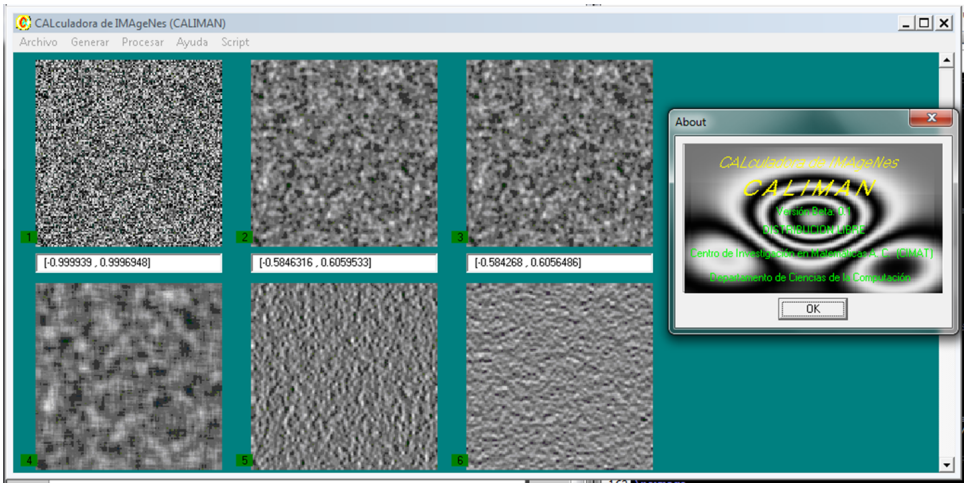
4.1 CARACTERÍSTICAS ESENCIALES

Para el desarrollo de este proyecto se empleó un proceso de arquitectura inspirado en los conocidos como Reverse Engineering descritos en Choi and Scacchi [1990], Chikofsky and Cross II [1990] y Byrne [1991] en los cuales, a grandes rasgos y desde el punto de vista de los intereses de este trabajo, se extraen modelos (por ejemplo: de datos o de procesos) del sistema actual para aproximar su arquitectura, de manera tal que posteriormente se pueda llevar a cabo una generalización con la que se facilite plantear una organización tal que favorezca la reutilización en nuevos diseños, posiblemente a través de una biblioteca de componentes reutilizables.

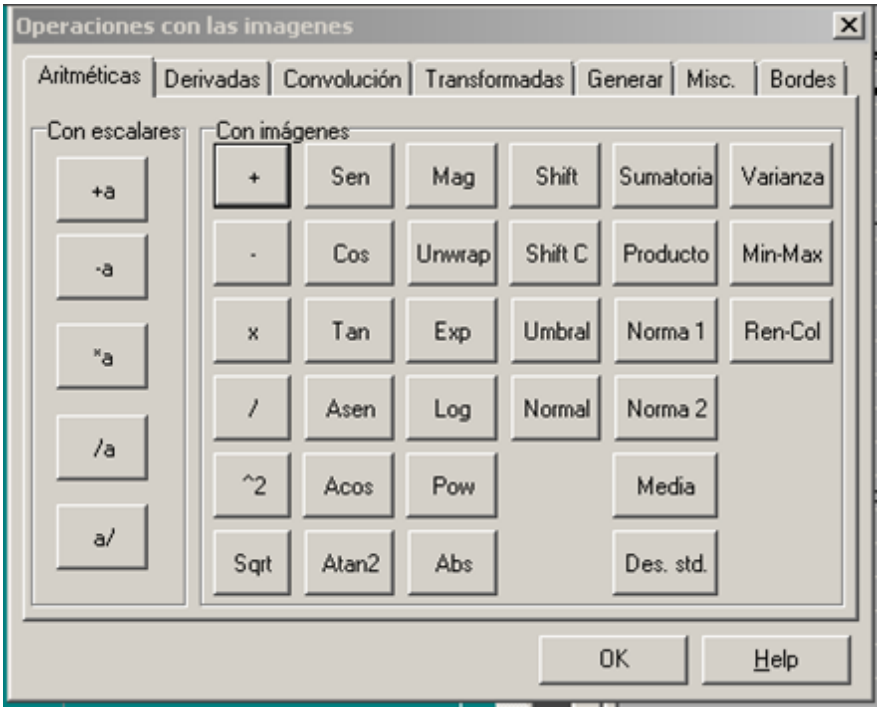


(a) Secuencia de operaciones en el script de CALIMAN.

Figura 2: El intérprete de scripts de CALIMAN.



(a) Nichos de imágenes.



(b) Botoneras.

Figura 3: Los nichos y las botoneras están alineadas en arreglos rectangulares.

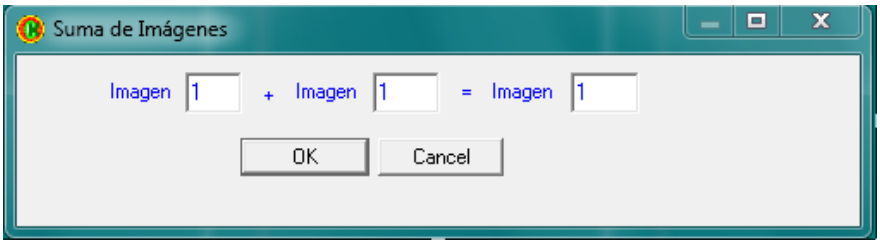
Se partió de un programa que construyó el CIMAT que en esencia consiste en:

- nichos para mostrar imágenes (ver Figura 3a) con campos de texto para despliegue de datos numéricos,
- un conjunto de operaciones al que se accede por medio de botones agrupados o separados en pestañas, con una interfaz semejante a una calculadora (ver Figura 3b),

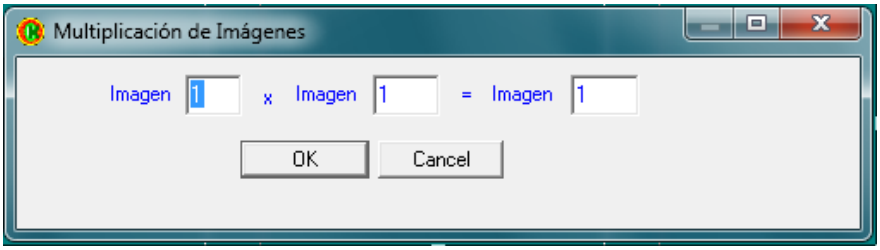
- que cada botón manda a presentar a una caja de diálogo (ver Figura 4) para introducir valores a los parámetros de las operaciones y pasarlos respectivamente a
- las funciones que realizan dichas operaciones en lenguaje C,
- una ventana opcional para introducir secuencias de operaciones con un script (ver Figura 2a).

4.2 MOTIVOS Y CONSIDERACIONES PREVIOS A LA PROPUESTA

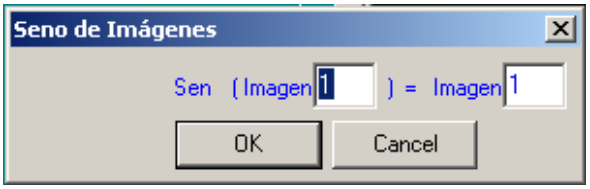
De entrada se observaron ciertas regularidades aprovechables.



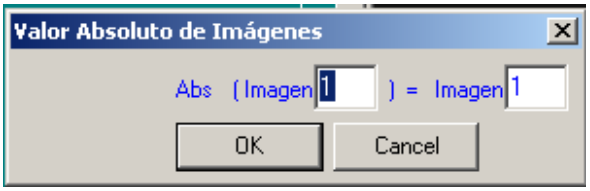
(a) Caja de diálogo de la operación suma de imágenes.



(b) Caja de diálogo de la operación multiplicación de imágenes.



(c) Caja de diálogo de la operación seno de imagen.

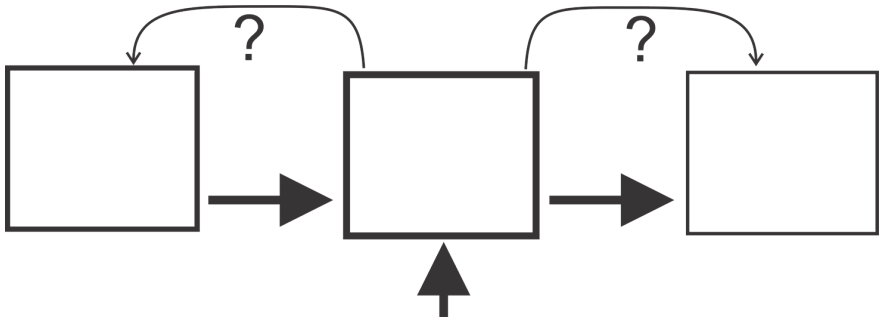


(d) Caja de diálogo de la operación valor absoluto de imagen.

Figura 4: Muchas cajas de diálogo presentan la misma estructura con distinto contenido.

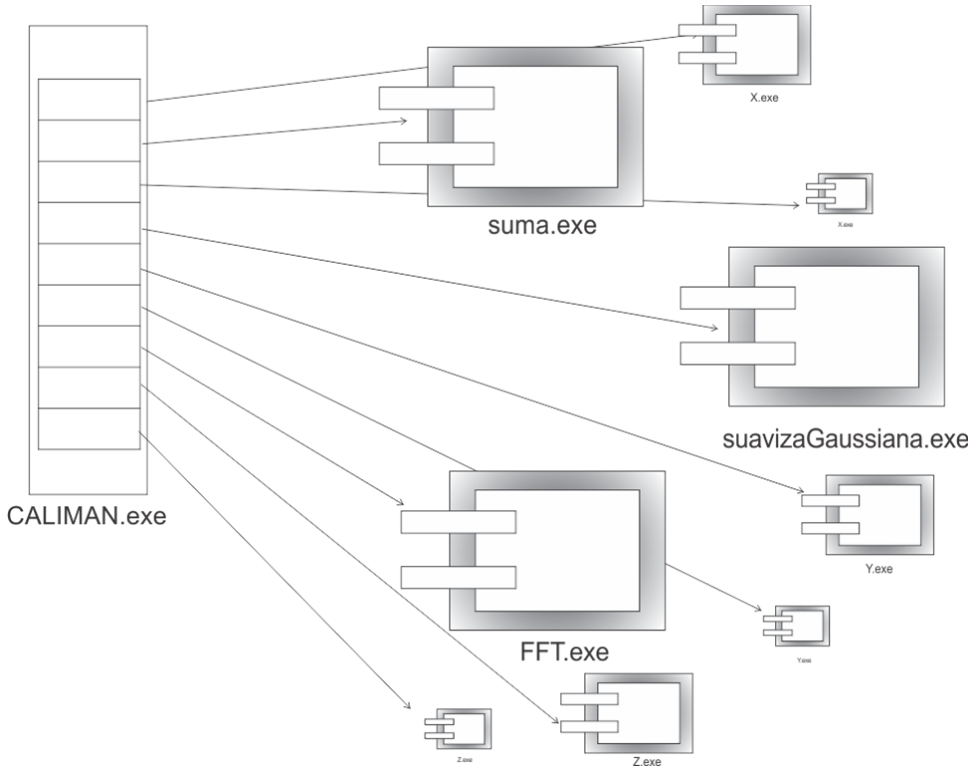
Como se puede apreciar en la Figura 4, se observó que las cajas de diálogo de muchas operaciones tienen la misma estructura pero

distinto contenido. En la Figura 3 se muestra que los nichos para las imágenes tienen todos la misma forma y que tanto los nichos como los botones de las pestañas en las botoneras de la calculadora están arreglados en un patrón rectangular. Por otra parte, se encontró que es posible, aunque no inmediato, construir una interfaz gráfica de usuario a partir de una especificación en texto con la tecnología empleada para la implementación actual de la calculadora, a saber: Lex, YACC y Borland C++ Builder. Estas dos observaciones juntas inspiraron la idea de usar plantillas basadas en cadenas de texto.



(a) Preguntas de exploración dirigidas en sentidos opuestos en cada paso.

Figura 5: Esquema de exploración.



(a) Separación del monolito en módulos individuales.

Figura 6: Esquema de separación.

Tras una exploración más profunda de esta idea, iterando en cada paso con preguntas dirigidas en sentidos opuestos (ver Figura 5a), a saber: *¿Qué se necesita para realizar lo que tengo propuesto en la iteración anterior?* o *¿De dónde viene esta propuesta?* en un sentido y en el sentido opuesto *¿Qué me sugiere lo que tengo propuesto en la iteración anterior?* o *¿A dónde me lleva lo que estoy proponiendo?*, con la intención de determinar hasta que punto se podría emplear con provecho esta generalización en plantillas, se concluyó que podría ser útil también generar plantillas del código fuente de los programas de las operaciones aparte de las plantillas de las cajas de diálogo correspondientes; de ahí la idea de separarlos en módulos individuales (ver Figura 6a).

Dicha separación trajo, a su vez, la idea de crear un módulo auxiliar *despachador*, esto es, cuya función fuera recibir los parámetros recogidos en las cajas de diálogo de las operaciones y pasarlos a sus programas para configurar su ejecución a la vez que mantuviera un estado del sistema indicando las imágenes que estuvieran abiertas y los nichos a los que estuvieran asignadas; con esto llegó la idea de también usar plantillas para generar la configuración que se pasaría a los programas.

Hasta este punto, todavía considerando no modificar mucho el estado actual de la calculadora, particularmente en lo referente a la disposición y modo de construcción de sus componentes, aún se estaba dejando de lado la capacidad de introducir scripts. Para solucionarlo se planteó la idea de recibir primero en los comandos de script los parámetros de las operaciones y luego llevar el script para su procesamiento al módulo despachador; de manera que el uso del script pasó de ser opcional a ser obligatorio, lo que condujo a la idea de hacer un parser o analizador del script e integrarlo al módulo despachador, lo que a su vez incluyó la idea de tener una lista de palabras reservadas para los comandos de las operaciones y su sintaxis correspondiente en el script. De nuevo, la idea de las plantillas se podría aplicar para los comandos del script.

Habiendo satisfecho como atender las características esenciales, aún quedaba en el aire como conseguir integrar todas esas propuestas bajo el ambiente de desarrollo ya mencionado. Concretamente, no quedaba claro como conectar los componentes de la interfaz gráfica de usuario tales como etiquetas o campos de texto con las variables correspondientes de manera automática, consistente e inmediata para ganar ventajas respecto a lo existente.

Por otra parte, afortunadamente para la situación, se encontró que las tecnologías basadas en cadenas de texto empleadas en las páginas web llenaban este hueco, lo que condujo a replantear la calculadora como una aplicación web y examinarla bajo la arquitectura de cliente-servidor. Esto obligó a tomar otras consideraciones no previstas originalmente, como el hecho de que por seguridad no es posible llamar una caja de diálogo para seleccionar archivos y conocer cual archi-

vo fue seleccionado directamente desde dentro del navegador, lo que sugirió la idea de crear un módulo que pudiera hacerlo desde afuera.

Se separaron entonces las funciones en cuanto a despliegues en pantalla y procesamiento de datos, y se clasificaron tomando la nomenclatura usual para la arquitectura propuesta como de front-end o de back-end respectivamente.

Se pudo comprobar que usando una página web se podía hacer una interfaz gráfica de usuario semejante a la que se tenía, con nichos para mostrar imágenes y campos de texto para despliegue de datos numéricos. Esto produjo la necesidad de obtener una conveniente representación de las imágenes con el rango dinámico normalizado para presentación en pantalla y un tamaño de miniatura fijo pero configurable.

De manera similar, aprovechando que en las páginas web se pueden designar regiones movibles y que se pueden ocultar a manera de ventanas, se construyeron en ellas pestañas con arreglos rectangulares de botones para acceder a las cajas de diálogo de las operaciones.

Aún más, la construcción del script y su paso al programa despachador hallaron su curso natural en las funciones de transferencia y manipulación de texto habituales de los navegadores web.

No obstante, aún con estos atajos, se pudo observar que el paso de parámetros a las funciones que realizan las operaciones así como el proceso de despliegue de las imágenes resultado de las mismas siguen una ruta más elaborada que sólo pasar el control a rutinas y hacer manipulaciones en la memoria asignada dentro del mismo proceso de sistema operativo. Esto se describe con más detalle junto con el procesamiento correspondiente al script más adelante en este capítulo en la parte de llamada a operaciones en la sección 4.4.

Con la generalización de las operaciones en plantillas se estableció una clasificación de las plantillas atendiendo a la cantidad y calidad[1] de las entradas y salidas involucradas. Por ejemplo: la suma, la resta, la multiplicación y la división de imágenes son operaciones que producen una imagen a partir de dos imágenes; mientras que la potencia, raíz cuadrada, logaritmo de una imagen son operaciones que producen una imagen a partir de una imagen; en cambio las derivadas Gaussianas son operaciones que producen dos imágenes a partir de una imagen y un número real positivo[2].

Más adelante se hallaron partes comunes en todas las plantillas que llevaron a considerar componentes[3] de plantillas. De modo que cada vez que se tuviera que producir una nueva plantilla se podrían

1 Los tipos de datos involucrados y su calidad de entrada o salida respecto al programa individual de la operación.

2 Aunque para fines de implementación, que es como se encuentran en la calculadora que se tomó como base para este trabajo, se pueden considerar los tratamientos horizontal y vertical por separado y decir en ese caso que producen una imagen a partir de una imagen y un número real positivo.

3 En el código fuente están identificadas como entidades.

reutilizar algunos de estos componentes y si hicieran falta crear unos nuevos. Para crear cada uno de estos componentes se tuvo que hacer primero una instancia completa y funcional de una operación representativa de cada tipo de plantilla, de manera que con varias plantillas se pudieran identificar partes comunes y separarlas para de ahí ubicar las posiciones invariantes y dejar marcadas aquellas donde se aplicarían los cambios en cada uso para finalmente añadirlas a una lista de donde se tomarían para formar las plantillas correspondientes. Esto último aplica en cada operación tanto para el código HTML en las cajas de diálogo como para el código de lenguaje C del programa correspondiente de tal operación.

Luego, al emplear una plantilla para dar lugar a una operación, se llenarían los lugares marcados con los valores respectivos para ajustarse a la situación actual y permitir usar varias veces el mismo componente en la misma plantilla. Más detalles acerca de como se realizan estas transformaciones se presentan más adelante en la parte de armado de la caja de diálogo en la sección 4.4.

Finalmente con todas estas consideraciones se estableció emplear una arquitectura híbrida de cliente-servidor como se describe en la siguiente sección.

4.3 FORMULACIÓN DE LA ARQUITECTURA PROPUESTA

En esta sección se bosquejan y discuten las ideas que conforman la propuesta que se desarrolló para solucionar el problema planteado en las secciones 1.1 y 2.3.

4.3.1 *Misión*

Se pretende, mediante la observación de principios disciplinados de desarrollo de software, extender el ciclo de vida de CALIMAN para seguir aprovechando los beneficios que ofrece.

4.3.2 *Visión*

Se propone una arquitectura híbrida que favorezca la separación explícita de responsabilidades, que sea jerárquica de manera tal que a nivel global sea de cliente-servidor basada en tecnologías web, mientras que a nivel de servidor sea modular y extensible, a la vez que, a nivel de cliente considere por separado los aspectos de presentación, estructura, contenido y comportamiento. Se propone que dicha separación de responsabilidades se obtenga principalmente a través de la transferencia de datos tales como cadenas de texto o por medio de archivos binarios y del empleo atinado de plantillas multinivel o anidadas, que rellenas de manera conveniente a través de sustitucio-

nes sucesivas con valores apropiados posibiliten llegar a la ejecución de tareas al nivel de las operaciones entre imágenes.

4.3.3 *Composición*

DOS FLANCOS PRINCIPALES: front-end y back-end, esto es, parte del cliente y parte del servidor. La parte del cliente se encarga del despliegue de imágenes, mientras que la parte del servidor se encarga del procesamiento de imágenes.

DOS INTERESES PRINCIPALES: usuario final y desarrollador. El usuario final es el interesado en obtener resultados de operar con las imágenes, el desarrollador se encarga de crear o implementar las operaciones y añadirlas al sistema.

4.3.3.1 *Front-end para el usuario final*

El usuario final accede al sistema a través de un programa que se ejecuta en el contexto de un navegador web con soporte del Hypertext Markup Language, versión 5 (HTML5) y Javascript, esto es, su interfaz de usuario es una página web enriquecida con interactividad[4]. Se puede decir que el front-end para el usuario final se encuentra encapsulado en una ventana de un navegador web y se compone de:

NICHOS DE PRESENTACIÓN DE IMÁGENES. Una región dedicada a mostrar los resultados de las operaciones con imágenes.

PANEL DE BOTONES O BOTONERAS. Una región dedicada a exhibir las funciones disponibles para operar con las imágenes y que es similar al tablero de botones de una calculadora de bolsillo con las operaciones clasificadas o agrupadas en pestañas.

CAJAS DE DIÁLOGO DE LAS OPERACIONES. Para la introducción de valores en los parámetros de las operaciones.

EDITOR DE SCRIPT. Un campo de texto para especificar las secuencias de operaciones a llevar a cabo ya sea de manera expresa o como resultado del empleo de las cajas de diálogo de las operaciones por parte del usuario.

4 Se hace esta distinción porque tradicionalmente las páginas web han sido de sólo lectura y en pocos casos con limitadas posibilidades de escritura o modificación a través de formularios, siempre requiriendo de un viaje redondo al servidor web para recibir las modificaciones solicitadas.

4.3.3.2 *Back-end para el usuario final*

Se encuentra al alcance del contexto de ejecución de un servidor web y se compone de:

PARSER/DESPACHADOR. Un programa encargado de interpretar las secuencias de operaciones, lanzar los programas que efectúan dichas operaciones con los parámetros correspondientes y que finalmente recoja los resultados para enviarlos al cliente para su presentación.

LISTA DE COMANDOS ACEPTADOS. Un archivo donde se enumeran los comandos aceptados por el parser/despachador y su sintaxis especificando el orden de los parámetros y determinando su propósito como de entrada o salida.

SCRIPT. Un medio para expresar secuencias de operaciones incluyendo valores de sus parámetros por medio de comandos con una sintaxis determinada definida en la lista de comandos aceptados.

ALMACÉN DE VARIABLES Y VALORES. Un archivo donde se mantiene el estado de ejecución actual, indicando que variables están definidas y con cuales valores además de cuales nichos están ocupados y por cuales imágenes. Debe estar en un archivo aparte y no en la memoria de un proceso en particular para que pueda ser modificado por el programa despachador y pueda ser solicitado por el navegador web.

MÓDULO DE OPERACIÓN CON IMÁGENES. Cada operación se lleva cabo en un programa independiente que se comunica con el resto del sistema a través de una interfaz de entrada y una interfaz de salida. La interfaz de entrada se basa en la lectura de archivos de configuración especificados en la línea de comandos para el paso de parámetros a las funciones que realizan las operaciones. También se considera parte de la interfaz de entrada la lectura de archivos con formato de matriz almacenada en binario para el acceso a imágenes y valores. La interfaz de salida consiste en la escritura de los resultados de la operación que le compete en archivos con formato de matriz almacenada en binario.

INTERFAZ/PROTOCOLO DE INTERCAMBIO DE DATOS. Una especificación de la configuración que deben recibir los programas en forma de pares llave-valor por un lado y por el otro el formato de matriz almacenada en binario para transferencia de imágenes y valores entre programas además de la especificación de las señales que se transfieren entre back-end y front-end, incluyendo el procesamiento del script en ambos flancos y la entrega de resultados; se encuentra

implementado tanto en los programas de operaciones como en el programa despachador.

MÓDULO DE MANEJO DE CAJAS DE DIÁLOGO NATIVAS DE APERTURA Y GUARDADO DE ARCHIVOS Un par de programas cliente-servidor encargados de recibir una señal del navegador web para desplegar cajas de diálogo nativas para abrir o guardar archivos y devolver al navegador web la cadena de texto que representa el nombre de archivo elegido.

4.3.3.3 *Front-end para el desarrollador*

Tiene los mismos elementos que para el front-end para el usuario final y además:

ASISTENTE PARA AÑADIR OPERACIONES. Una serie de formularios para insertar los datos requeridos para construir una nueva operación en un orden conveniente para atender las dependencias entre ellos y generar una interfaz gráfica de usuario especificada en HTML y Javascript. La interfaz gráfica de usuario de una operación consiste en un botón debidamente etiquetado y asignado a una botonera además de la caja de diálogo para especificación de valores para los parámetros de la operación. También se genera una plantilla de código fuente en lenguaje[5] C que ya incluye las funciones para abrir y guardar archivos con formato de matriz almacenada en binario, asignar y liberar recursos de memoria y otras auxiliares para leer archivos de configuración de manera tal que sólo reste añadir el código correspondiente a la operación.

4.3.3.4 *Back-end para el desarrollador*

Tiene los mismos elementos que para el back-end para el usuario final y además:

BITÁCORA DE EVENTOS. Un archivo de texto al que se escriben las ocurrencias de los eventos tales como apertura de archivos, modificación de estructuras de datos, entre otros, que es de utilidad para la depuración o mantenimiento. También se incluye en la interfaz gráfica de usuario un campo de texto donde se muestra todo lo que los programas de operaciones manden a su salida estándar durante su ejecución.

5 El desarrollador debe contar además con un compilador de lenguaje C o C++ aunque técnicamente puede ser en el lenguaje de su preferencia mientras cuente con la manera de satisfacer el protocolo de intercambio de datos que se describe en la sección 4.4. En este proyecto sólo se brinda soporte al lenguaje C en los programas de operaciones y en ocasiones C++ en otros programas auxiliares en lo que toca a infraestructura y andamiaje por medio del ambiente de programación Qt Creator.

4.3.4 *Entorno*

- El entorno operativo se centra en un cliente web que se comunica con un servidor web, mientras que el servidor web se comunica principalmente con un programa despachador, y a su vez, el programa despachador se comunica con los programas de operaciones individuales; el servidor web también se comunica con un programa para grabar archivos de texto y con el cliente de un par cliente-servidor de programas intermediarios para desplegar cajas de diálogo nativas; a su vez este cliente se comunica con su par servidor.
- El entorno del cliente web es un sistema operativo con soporte para comunicación por medio del protocolo HTTP sobre redes[6] TCP/IP,
- El entorno del servidor web es un sistema operativo con acceso a un sistema de archivos y soporte para comunicación por medio del protocolo HTTP sobre redes TCP/IP. Particularmente, el servidor web debe inyectar al ambiente CGI las variables `REMOTE_ADDR` y `SERVER_ADDR`.
- El entorno del programa despachador y los programas individuales de operaciones es un sistema operativo con acceso a un sistema de archivos y la capacidad de llamar a programas desde otros programas en ejecución[7].
- El entorno del programa cliente intermediario para desplegar cajas de diálogo nativas debe tener soporte para esquemas de comunicación entre procesos como sockets o pipes para comunicarse con su par servidor y el acceso a la salida estándar que es como se comunica de regreso con el servidor web.
- Por su parte, el usuario final sólo interactúa directamente con el cliente web, mientras que el desarrollador debe tener acceso a todo el entorno operativo.

4.3.5 *Estructura*

- Entre la parte del servidor web y el cliente web se emplea el protocolo HTTP como transporte y archivos con cadenas de texto para transferencia de datos, o archivos binarios para transferencia de imágenes.

6 Redes TCP/IP : Transmission Control Protocol o Protocolo de Control de Transmisión (TCP) , Internet Protocol o Protocolo de Internet (IP)

7 En el lenguaje C a través de la llamada a system().

- Por la parte del servidor web se emplea el estándar CGI para comunicarse con programas de operaciones individuales y con los programas auxiliares ya mencionados.
- Los programas individuales de las operaciones[8] intercambian información por medio de archivos binarios estructurados.
- Por la parte del cliente web se emplea el estándar HTML5 y programas en lenguaje Javascript para interactuar con el usuario.

4.3.6 *Mecanismo*

Por la parte del cliente se aprovecha la capacidad de modificar en tiempo de ejecución los elementos de la interfaz gráfica basada en cadenas de texto a través de la manipulación de dichas cadenas;
Por la parte del servidor se construyen cadenas de texto que se pasan por medio de archivos de configuración a los programas al invocarlos con el fin de ajustar los valores de ciertas variables que controlan su comportamiento.

Se ofrecen a continuación más detalles de lo anterior.

4.4 IMPLEMENTACIÓN DE LA PROPUESTA

En esta sección se exponen y argumentan los pormenores de implementación con cierta profundidad pero sin llegar al nivel del detalle expuesto en el código fuente, que es la fuente última de información al respecto. Es de interés particular la presentación de un ejemplo completo de la secuencia de sustituciones que se efectúan para construir una caja de diálogo, seguido de un ejemplo de análisis de una línea de script, así como una breve descripción, por su importancia, de un par de funciones del código fuente. El ejemplo elegido es el de la operación de suma de imágenes por su relativa simplicidad y porque cubre el caso del uso repetido de un componente de plantilla. Esto es de interés para apreciar como se echa a andar más de una instancia de un mismo componente.

4.4.1 *Estructuras de datos*

En esta sección se esbozan las estructuras de datos y los formatos de archivo empleados, más detalle se ofrece en el apéndice B.
En general se usa el formato de archivo de texto que llamamos *encabezado-datos*, donde el primer renglón es un encabezado conformado de valores separados por algún símbolo especial y el resto de renglones son los datos cuyas características posiblemente estén descritas en el encabezado; o el formato que llamamos *llave-valor* donde

8 Escritos en lenguaje C, pero no limitándose a él.

cada renglón tiene un par de cadenas separadas por algún símbolo especial y cuyo contenido puede ser vertido en un diccionario de pares llave-valor.

SERVIDOR En el servidor los programas de las operaciones emplean dos tipos de estructuras de datos en su funcionamiento, a saber: el diccionario de pares llave-valor, y el arreglo bidimensional.

CLIENTE Para la parte del cliente en su mayoría se emplean arreglos anidados, esto es, arreglos cuyos elementos son a su vez arreglos. Esto obedece a la necesidad de actuar con índices, además de que la construcción de esta estructura de datos es inmediata en Javascript con la JSON.

4.4.1.1 *Interfaz/protocolo de intercambio de datos*

CONFIGURACIÓN DE LOS PROGRAMAS DE OPERACIONES: Los programas de operaciones reciben un archivo con pares llave-valor separados por el símbolo =.

ARCHIVOS DE IMÁGENES Y VALORES: Los archivos para almacenar imágenes y valores deben ser de formato de matriz almacenada en binario.

SEÑALES DEL FRONT-END AL BACK-END: El navegador web envía al programa despachador una cadena de texto con símbolos protegidos que representa el script de CALIMAN. Dicha protección se hace para evitar perder algunos símbolos por la manera en que se codifican y transfieren con el HTTP. Concretamente, la protección consiste en sustituir los espacios por una cadena `___SPC___` y los signos + por la cadena `___PLUS___` y aplicar una pasada de codificación URL-Encoding; todo esto se explica con detalle más adelante. Al recibirse dicha cadena en el back-end se efectúa el proceso inverso para obtener la cadena original.

Por otra parte, el navegador web envía al programa intermediario para desplegar cajas de diálogo nativas una de las siguientes tres cadenas:

- `CALIMAN:Command:OpenFile`,
- `CALIMAN:Command:SaveFile`,
- `CALIMAN:Command:Ping`.

SEÑALES DEL BACK-END AL FRONT-END: El programa intermediario para desplegar cajas de diálogo nativas responde con cualquiera de las siguientes cadenas:

- `CALIMAN:Result:OpenFile:` concatenado a la ruta del archivo especificado para abrir,
- `CALIMAN:Result:SaveFile:` concatenado a la ruta del archivo especificado para guardar,
- `CALIMAN:Result:Pong`,
- `CALIMAN:Result:Unknown`.

El programa despachador responde con una larga cadena de texto que tiene información de los eventos ocurridos al analizar las líneas de script y llamar a los programas de operaciones. De interés son los renglones que tienen simultáneamente las cadenas `IMAGEN` y `cargarImagenEnNicho:` pues son las que indican en que nicho se debe cargar determinada imagen.

4.4.1.2 *Almacén de variables y valores*

El almacén de variables y valores es un archivo de pares llave-valor separados por el símbolo[9] ¬, donde la llave designa el nombre del nicho o de la variable almacenada.

4.4.1.3 *Lista de comandos aceptados.*

La lista de comandos aceptados es un archivo de texto cuyos renglones corresponden cada uno a un comando. Su propósito es declarar cuales comandos se aceptan y que forma tienen para ser aceptados por el programa despachador.

En cada renglón los valores de interés están separados por un símbolo[10] ¬. A continuación se muestra como se ve el renglón correspondiente al ejemplo de la suma de imágenes:

Listado 1: Ejemplo de renglón en el archivo de comandos de operación aceptados.

```
sumarImagenes¬ectoArchivoResultado1 = sumarImagenes (
    endoArchivoOperando1 , endoArchivoOperando2 )
```

En el listado se puede observar que el símbolo separador aparece una vez por lo que se tienen dos valores de interés.

El primer valor de interés es el nombre del comando. Especifica la secuencia de letras para identificar el comando y con esa misma secuencia de letras minúsculas y mayúsculas es como debe aparecer en los scripts para que sea tomado en cuenta como comando, además

9 En realidad, puede ser cualquier símbolo, letra o número, no obstante, la elección de tal símbolo se basa en la suposición de que ninguna de las cadenas de llave o de valor lo contiene.

10 En realidad, puede ser cualquier símbolo, letra o número, no obstante, la elección de tal símbolo se basa en la suposición de que ninguno de dichos valores de interés lo contiene.

el ejecutable de la operación también debe llamarse así (sin considerar la extensión).

El segundo valor de interés es la sintaxis del comando en el lenguaje de scripts de la calculadora. Especifica la secuencia de palabras para identificar los operandos, así como la especificación de su tipo indicando con prefijos si son de entrada o de salida. El prefijo `ecto` indica un operando de salida o que recoge el resultado de la operación y el prefijo `endo` indica un operando de entrada o que aporta un insumo a la operación. Los nombres sin los prefijos están definidos en el código fuente del programa despachador, y están diseñados para decir algo sobre el propósito de esa posición en la sintaxis del comando y controlar el flujo de análisis y procesamiento de las partes de las líneas de scripts. Es obligatorio cumplir con esa nomenclatura pues el programa despachador depende de ella para hacer en orden las asignaciones de valores a las entradas y salidas de las operaciones y tanto el programa de la operación como su archivo de configuración correspondiente hacen uso de esos nombres sin los prefijos.

Listado 2: Las palabras clave empleadas por el programa despachador. X es un número entero indicador del orden de aparición de dichas palabras en el comando de script.

```
ectoValorNumericoX , endoValorNumericoX , ectoArchivoResultadoX ,
     endoArchivoOperandoX , ectoValorResultadoX
```

4.4.2 *Funcionamiento*

En esta sección se da una descripción del funcionamiento del sistema a través del seguimiento de un ejemplo representativo, el de la suma de imágenes. La presentación se divide en dos partes, cada una enfocada a uno de los dos roles definidos para los posibles usuarios, a saber: el rol de desarrollador de operaciones y el rol de usuario final. La primera parte abarca desde que el usuario desarrollador decide implementar una nueva operación y termina en el momento en que la nueva operación queda disponible para el usuario final como un botón en el panel de botones y con su programa correspondiente en una ubicación en el sistema de archivos tal que estén al alcance del programa despachador. La segunda parte abarca desde que el usuario final decide usar alguna operación (inclusive la recién añadida) activando su caja de diálogo por medio del botón correspondiente en la botonera asignada, pasando por una posible edición de valores en dicha caja de diálogo o en el campo de texto del script y termina en el momento en que se despliegan los resultados de la operación.

4.4.2.1 *Para el usuario desarrollador de operaciones*

Supongamos para los efectos de este ejemplo que no está implementada aún la operación de suma de imágenes y que a continuación es la que se pretende añadir.

El usuario desarrollador de operaciones efectúa tres pasos principales, a saber:

- Registrar la operación en la lista de comandos aceptados.
- Construir una interfaz gráfica de usuario para la operación, esto es, su botón y caja de diálogo.
- Codificar la operación, compilar y copiar el ejecutable resultante al directorio donde el programa despachador espera que estén los programas de operaciones.

El usuario desarrollador de operaciones debe tener presente el tipo de operación que va a implementar en términos del tipo y cantidad de entradas y salidas, además de saber codificarla en el lenguaje C. En el caso de la operación de suma de imágenes se trata de una operación que produce una imagen a partir de dos imágenes de entrada. Para empezar se debe implementar código auxiliar para asignar la memoria para los arreglos bidimensionales en donde se almacenan las imágenes operando, para cargar ahí dichas imágenes y para que al final de la operación se graben los resultados en un archivo. Este código auxiliar de solicitud y liberación de recursos de memoria y apertura de imágenes se genera junto con la caja de diálogo como parte del proceso de sustituciones sucesivas en plantillas. En su defecto se debe implementar aparte dicho código auxiliar.

4.4.2.2 *Proceso de armado de la caja de diálogo de la operación*

Para apreciar en pleno el ejemplo siguiente se requiere cierta experiencia con HTML. A los lectores que no tengan dicha experiencia solamente se les hace saber que se trata de una secuencia de sustituciones de cadenas de texto en lugares especialmente marcados en plantillas anidadas que conducen a la construcción progresiva de la caja de diálogo de la operación.

En el listado siguiente se muestran 5 componentes de plantilla para cajas de diálogo, en orden de aparición, a saber: un componente para mostrar un título en la caja de diálogo, un componente para elegir un nicho para almacenar una de las imágenes resultantes de la operación, un componente para hacer que aparezca una cadena de texto en la caja de diálogo, un componente para elegir un nicho que servirá de operando o entrada para la operación, y un componente para hacer que aparezcan un botón de cancelación de la operación y un botón para confirmar la aceptación de los valores especificados en la caja de diálogo y enviarlos a la operación.

Listado 3: Cadenas de componentes de plantilla con lugares marcados para sustitución.

```
//strEntidadTitulo
<legend> _____OPERACION_____ </legend>


//strEntidadImagenDestino
<label for="lbImagenDestino _____PLANTILLA_____ _">
    _____OPERACION_____ </label>
<select id="lbImagenDestino _____PLANTILLA_____ _">
   _____EJECUCION_____
</select>


//strEntidadTexto
<span> _____OPERACION_____ </span>


//strEntidadImagenOrigen
<label for="lbImagenOrigen _____PLANTILLA_____ _">
    _____OPERACION_____ </label>
<select id="lbImagenOrigen _____PLANTILLA_____ _">
   _____EJECUCION_____
</select>


//strEntidadBotonesProcesarCancelar
<div style="text-align:center; padding: 10px;">
  <input type="button" id="btGenerar"  value="Procesar" onclick="
      accionBotonProcesarOperacionYMostrarBotones (
/*iPestagna*/ _____EJECUCION_____ ,
/*idNodoDOMcontenedor*/ '_____EJECUCION_____' ,
/*idNodoDOMasa*/ '_____EJECUCION_____' ,
/*strParameterizedCommand*/ '_____OPERACION_____' ,
/*arrStrIDsOrigenes*/ _____OPERACION_____ ,
/*strReplacementSignal*/ '_____PARAM_____' );">

  <input type="button" id="btCancelar" value="Cancelar" onclick="
      accionBotonCancelar (
/*iPestagna*/ _____EJECUCION_____ ,
/*idNodoDOMcontenedor*/ '_____EJECUCION_____' ,
/*idNodoDOMasa*/ '_____EJECUCION_____' );">
</div>
```

En este caso se representan por cadenas cada uno de los componentes de plantilla de manera que se puedan usar en concatenaciones o sustituciones posteriores.

Se puede observar en el listado la presencia de las marcas siguientes que indican las posiciones para sustitución:

- `_____PLANTILLA_____`,

- `_____OPERACION_____`,
- `_____EJECUCION_____`,
- `_____PARAM_____`.

Éstas indican el nivel de anidación de ese campo de sustitución en la plantilla, en ese orden, esto es, primero se sustituyen las marcas de plantillas, luego las de operación, luego las de ejecución y finalmente las de parámetros de la operación. Como se verá más adelante, la sustitución a nivel de plantilla enumera de manera genérica los recipientes de datos de acuerdo a su propósito, ya sea de entrada o de salida, mientras que la sustitución a nivel de operación asigna las características propias de la operación a la que corresponda la caja de diálogo que se está construyendo. Posteriormente, la sustitución a nivel ejecución toma valores presentes en el contexto de ejecución cuando se construye la caja de diálogo de la operación tras solicitarla por medio de su botón en la botonera correspondiente. Finalmente, la sustitución a nivel de parámetros de la operación asigna los valores dados por el usuario en las posiciones correspondientes en el comando de script para su interpretación posterior por el programa despachador.

En el último componente del listado es de interés enfocar la atención en dos funciones Javascript a las que llama el usuario final por medio de la caja de diálogo, a saber:

- `accionBotonCancelar`,
- `accionBotonProcesarOperacionYMostrarBotones`.

La función `accionBotonCancelar` solamente sustituye el despliegue actual de la caja de diálogo por el despliegue de la botonera desde la cual se llamó a ésta, lo que desaparece la caja de diálogo y hace aparecer la botonera desde la que se solicitó.

La función `accionBotonProcesarOperacionYMostrarBotones` arma la cadena de script tomando los valores de los campos indicados en el arreglo `arrStrIDsOrigenes` y sustituyéndolos en donde se encuentren las marcas indicadas por `strReplacementSignal` en la plantilla del script especificada por `strParameterizedCommand`. Luego la envía a ejecución y pone una copia en la ventana de script. Finalmente sustituye el despliegue actual de la caja de diálogo por el despliegue de la botonera desde la cual se llamó a ésta.

Esta última función es de interés porque es parte detonante del proceso de fondo que efectúa la incrustación de los datos introducidos por el usuario, a través de la caja de diálogo generada, en las posiciones debidamente indicadas en el comando de script, esto se analizará con más detalle más adelante en la sección 4.4.2.4 dedicada a las actividades del usuario final.

Dada una lista ordenada de componentes de plantilla se puede construir una plantilla para la operación actual. Esta lista la provee el desarrollador en un arreglo de objetos Javascript. En el listado siguiente aparecen los componentes en el orden que se ocupa para generar la caja de diálogo de este ejemplo.

Listado 4: Lista ordenada de componentes añadidas a la plantilla actual.

```
"[strEntidadTitulo, strEntidadImagenDestino, strEntidadTexto,
    strEntidadImagenOrigen, strEntidadTexto,
    strEntidadImagenOrigen, strEntidadTexto,
    strEntidadBotonesProcesarCancelar]"
```

Se puede observar en el listado que el orden en que aparecen los componentes de plantilla corresponde al caso que estamos analizando donde se ocupa que en la caja de diálogo aparezcan, en ese orden, un componente de salida o de destino y dos de entrada o de origen. Concretamente, lo que se quiere con esa lista es que se muestre primero el especificador de la imagen de destino seguido del símbolo $=$ y posteriormente los especificadores de las imágenes de origen separados por el símbolo $+$. Por ello entre los componentes especificadores de imágenes se intercalan componentes de plantilla para mostrar texto. Al principio de la lista hay un componente de plantilla dedicado a que se pueda mostrar un título en la caja de diálogo mientras que al final hay un componente de plantilla dedicado a que aparezcan los botones que solicitan iniciar o cancelar el proceso.

Siguiendo el proceso, de acuerdo a dicha lista se concatenan las cadenas de texto de dichos componentes para conformar la plantilla de la operación, se envuelven en un contenedor `DIV` de HTML , y se sustituyen los marcadores `_____PLANTILLA_____` por números sucesivos quedando una plantilla para una operación con dos entradas y una salida:

Listado 5: Plantilla de una operación cualquiera con dos entradas y una salida.

```
<div class="CALIMAN_cajaDeDialogo_contenedor">

<legend>_____OPERACION_____</legend>

<label for=" lbImagenDestino0_ ">_____OPERACION_____</label>
<select id=" lbImagenDestino0_ ">
  _____EJECUCION_____
</select>

<span>_____OPERACION_____</span>

<label for=" lbImagenOrigen1_ ">_____OPERACION_____</label>
<select id=" lbImagenOrigen1_ ">
  _____EJECUCION_____
```

```
</select>

<span>_____OPERACION_____</span>

<label for="lbImagenOrigen2_">_____OPERACION_____</label>
<select id="lbImagenOrigen2_">
  _____EJECUCION_____
</select>

<span>_____OPERACION_____</span>

<div style="text-align:center; padding: 10px;">
  <input type="button" id="btGenerar"  value="Procesar" onclick="
      accionBotonProcesarOperacionYMostrarBotones (
/*iPestagna*/ _____EJECUCION_____ ,
/*idNodoDOMcontenedor*/ '_____EJECUCION_____' ,
/*idNodoDOMasa*/ '_____EJECUCION_____' ,
/*strParameterizedCommand*/ '_____OPERACION_____' ,
/*arrStrIDsOrigenes*/ _____OPERACION_____ ,
/*strReplacementSignal*/ '_____PARAM_____' );">

  <input type="button" id="btCancelar" value="Cancelar" onclick="
      accionBotonCancelar (
/*iPestagna*/ _____EJECUCION_____ ,
/*idNodoDOMcontenedor*/ '_____EJECUCION_____' ,
/*idNodoDOMasa*/ '_____EJECUCION_____' );">
</div>

</div>
```

Con este nivel de plantilla armado, el desarrollador incrusta etiquetas de apoyo al usuario final para indicarle el propósito de los componentes de acuerdo a la operación que se esté implementando; esto se realiza dando una lista ordenada de cadenas con el contenido conveniente. Dichas etiquetas de apoyo se pueden especificar también mediante el asistente para añadir operaciones.

Listado 6: Lista ordenada de cadenas que representan las etiquetas de apoyo al usuario final.

```
//strNombreDesplegableDeOperacion
"Sumar imágenes"

//strLblEcto
"ecto"

//strOpLblPt1
" = sumarImagenes( "

//strLblEndo1
"endo"
```

```
//strOpLblPt2
" , "

//strLblEndo2
"endo"

//strOpLblPt3
" ) "

//strOpSyntax
"____PARAM____ = sumarImagenes( ____PARAM____,
    ____PARAM____ )"

//strOpArrDOMSources
"['lbImagenDestinoo_','lbImagenOrigen1_','lbImagenOrigen2_']"
```

De interés especial en esta lista de cadenas está `strOpSyntax` que es una plantilla para la sintaxis del comando de la operación en el script y debe coincidir al nivel de separadores (tokens no-alfanuméricos tales como espacios) con la sintaxis definida en la lista de comandos aceptados por el despachador y `strOpArrDOMSources` que es un arreglo ordenado de cadenas que representan los identificadores DOM de los campos de donde se toman los valores para incrustarlos en la plantilla del script dada.

Al efectuar la sustitución de los marcadores `_____OPERACION_____` por las cadenas de la lista anterior queda la plantilla especializada para la operación, en este caso de suma de imágenes:

Listado 7: Plantilla de la operación suma de imágenes.

```
<div class="CALIMAN_cajaDeDialogo_contenedor">
  <fieldset>
    <legend> Sumar imágenes </legend>

    <label for="lbImagenDestinoo_"> ecto </label>
    <select id="lbImagenDestinoo_">
        _____EJECUCION_____
    </select>

    <span>  = sumarImagenes(  </span>

    <label for="lbImagenOrigen1_"> endo </label>
    <select id="lbImagenOrigen1_">
        _____EJECUCION_____
    </select>

    <span>  ,  </span>

    <label for="lbImagenOrigen2_"> endo </label>
    <select id="lbImagenOrigen2_">
        _____EJECUCION_____
```

```
    </select>

    <span> ) </span>

    <div style="text-align:center; padding: 10px;">
      <input type="button" id="btGenerar"  value="Procesar"
          onclick="accionBotonProcesarOperacionYMostrarBotones (
/*iPestagna*/    _____EJECUCION_____ ,
/*idNodoDOMcontenedor*/ '_____EJECUCION_____' ,
/*idNodoDOMasa*/ '_____EJECUCION_____' ,
/*strParameterizedCommand*/ '
     _____PARAM_____ = sumarImagenes( _____PARAM_____,
     _____PARAM_____ )' ,
/*arrStrIDsOrigenes*/
['lbImagenDestino0_' ,'lbImagenOrigen1_' ,'lbImagenOrigen2_'] ,
/*strReplacementSignal*/ '____PARAM____' );">

      <input type="button" id="btCancelar" value="Cancelar"
          onclick="accionBotonCancelar (
/*iPestagna*/    _____EJECUCION_____ ,
/*idNodoDOMcontenedor*/ '_____EJECUCION_____' ,
/*idNodoDOMasa*/ '_____EJECUCION_____' );">
    </div>
  </fieldset>
</div>
```

Con todo lo anterior queda armada la plantilla de la caja de diálogo de la operación de suma de imágenes. Esta plantilla se almacena en un arreglo conocido como *de botoneras* que contiene las cajas de diálogo y botones de todas las operaciones agrupadas en pestañas o botoneras, de manera que el usuario final al presionar el botón de la operación desencadene el paso siguiente de sustituciones para obtener la representación definitiva de la caja de diálogo, que se explica más adelante en la sección 4.4.2.4 dedicada a las actividades del usuario final. Esta misma secuencia de sustituciones se aplica en todas las operaciones para obtener sus cajas de diálogo. En el apéndice B se muestra el contenido del arreglo *de botoneras* y con ello la manera en que, para el mismo ejemplo, pero directamente, sin usar plantillas se construye la misma interfaz de usuario.

4.4.2.3 *Proceso de armado de la plantilla de código fuente de la operación*

Cada componente de entrada de datos tiene asociadas secciones de código en lenguaje C cuya finalidad es dar presencia a los valores introducidos en el programa de la operación.

Se trata de 13 secciones relevantes, a saber:

- La sección de definición de la estructura para configuración del programa.

- La sección de reporte a consola de la definición de la estructura de configuración del programa.
- La sección de definición de los valores predeterminados de configuración del programa.
- La sección de asignación de los valores predeterminados de configuración a los miembros correspondientes de la estructura de configuración del programa.
- La sección de definición de las llaves a buscar en los archivos de configuración pasados al programa. Los nombres de las llaves deben de coincidir con los especificados en la lista de comandos aceptados (sin los prefijos `endo` y `ecto`).
- La sección de búsqueda de las llaves en el archivo de configuración pasado al programa y la asignación de los valores correspondientes a los miembros de la estructura de configuración del programa.
- La sección de parámetros de la función `operacionConImagenes` en su declaración.
- La sección de parámetros en la implementación de la función `operacionConImagenes`.
- La sección correspondiente al código fuente propio de la operación.
- La sección de solicitud de recursos de memoria y apertura de imágenes especificadas en la configuración.
- La sección de llamada a la función `operacionConImagenes` a la cual se le envían referencias a los recursos solicitados.
- La sección de escritura de las imágenes resultado de la operación.
- La sección de liberación de los recursos solicitados.

En las posiciones marcadas `//_____PUNTODESUSTITUCIONPLANTILLA_____` las cadenas de código se sustituyen en la plantilla de código en lenguaje C.

El código explícito de cada región para este ejemplo se muestra en la sección correspondiente en el apéndice B.

Después de quedar armada la plantilla de código fuente de la operación, se procede a añadir el código propio de la operación, seguido de compilar y colocar el ejecutable resultante en el directorio donde se encuentran los programas de las demás operaciones.

4.4.2.4 *Para el usuario final*

El usuario final efectúa tres tipos de actividades.

- Cargar o generar imágenes.
- Efectuar operaciones y análisis con las imágenes.
- Guardar una imagen, posiblemente con una normalización previa para despliegue en pantalla.

En realidad, las tres se pueden analizar desde el punto de vista enmarcado por la lista siguiente:

- Selección de operación, construcción y despliegue de la representación definitiva de su caja de diálogo,
- Asignación de valores a los parámetros de la operación,
- Conversión a comando de script,
- Procesamiento del script en el front-end,
- Transferencia de datos entre front-end y back-end,
- Procesamiento del script en el back-end,
- Llamada a operaciones,
- Retorno de valores,
- Despliegue de resultados.

4.4.2.5 *Construcción y despliegue de la representación definitiva de su caja de diálogo*

En tiempo de ejecución, cuando el usuario final pulsa el botón correspondiente a la operación en la botonera, para obtener la representación definitiva de su caja de diálogo respectiva se sustituye una lista ordenada de cadenas en la plantilla que dejó el desarrollador. Estas cadenas contienen información que sólo se conoce hasta el momento en que se solicita la aparición de la caja de diálogo por parte del usuario final. Un ejemplo de este tipo de información es el listado de nichos que están ocupados de manera que se les muestre como entradas posibles en las operaciones y evitar dar como entradas nichos que no tienen imagen cargada aún. En el listado se supone que hay imágenes cargadas en los nichos 1 a 5, el resto de los nichos está vacío.

Listado 8: Lista ordenada de cadenas que representan valores que se conocen solamente hasta el tiempo de ejecución.

```
//strHTMLNichoDestino
<option value="IMAGEN01">Imagen 1</option>
```

```
<option value="IMAGEN02">Imagen 2</option>
<option value="IMAGEN03">Imagen 3</option>
<option value="IMAGEN04">Imagen 4</option>
<option value="IMAGEN05">Imagen 5</option>
<option value="IMAGEN06">Imagen 6</option>
<option value="IMAGEN07">Imagen 7</option>
<option value="IMAGEN08">Imagen 8</option>
<option value="IMAGEN09">Imagen 9</option>
<option value="IMAGEN10">Imagen 10</option>
<option value="IMAGEN11">Imagen 11</option>
<option value="IMAGEN12">Imagen 12</option>

//strHTMLNichoOrigen
<option value="IMAGEN01">Imagen 1</option>
<option value="IMAGEN02">Imagen 2</option>
<option value="IMAGEN03">Imagen 3</option>
<option value="IMAGEN04">Imagen 4</option>
<option value="IMAGEN05">Imagen 5</option>

//strHTMLNichoOrigen
<option value="IMAGEN01">Imagen 1</option>
<option value="IMAGEN02">Imagen 2</option>
<option value="IMAGEN03">Imagen 3</option>
<option value="IMAGEN04">Imagen 4</option>
<option value="IMAGEN05">Imagen 5</option>

//iPestagna
0

//idNodoDOMcontenedor
divCALIMANBotonerasContenedor

//idNodoDOMasa
divCALIMANBotonerasAsa

//iPestagna
0

//idNodoDOMcontenedor
divCALIMANBotonerasContenedor

//idNodoDOMasa
divCALIMANBotonerasAsa
```

Al efectuar la sustitución de los marcadores `_____EJECUCION_____` por las cadenas de la lista anterior en la plantilla que dejó el desarrollador, el código HTML queda como aparece en el listado siguiente:

Listado 9: Representación definitiva de la caja de diálogo para la operación suma de imágenes.

```
<div class="CALIMAN_cajaDeDialogo_contenedor">
```

```
  <fieldset>
    <legend>Sumar imágenes</legend>

    <label for="lbImagenDestinoo_">ecto</label>
    <select id="lbImagenDestinoo_">
      <option value="IMAGEN01">Imagen 1</option>
      <option value="IMAGEN02">Imagen 2</option>
      <option value="IMAGEN03">Imagen 3</option>
      <option value="IMAGEN04">Imagen 4</option>
      <option value="IMAGEN05">Imagen 5</option>
      <option value="IMAGEN06">Imagen 6</option>
      <option value="IMAGEN07">Imagen 7</option>
      <option value="IMAGEN08">Imagen 8</option>
      <option value="IMAGEN09">Imagen 9</option>
      <option value="IMAGEN10">Imagen 10</option>
      <option value="IMAGEN11">Imagen 11</option>
      <option value="IMAGEN12">Imagen 12</option>
    </select>

    <span> = sumarImagenes( </span>

    <label for="lbImagenOrigen1_">endo</label>
    <select id="lbImagenOrigen1_">
      <option value="IMAGEN01">Imagen 1</option>
      <option value="IMAGEN02">Imagen 2</option>
      <option value="IMAGEN03">Imagen 3</option>
      <option value="IMAGEN04">Imagen 4</option>
      <option value="IMAGEN05">Imagen 5</option>
    </select>

    <span> , </span>

    <label for="lbImagenOrigen2_">endo</label>
    <select id="lbImagenOrigen2_">
      <option value="IMAGEN01">Imagen 1</option>
      <option value="IMAGEN02">Imagen 2</option>
      <option value="IMAGEN03">Imagen 3</option>
      <option value="IMAGEN04">Imagen 4</option>
      <option value="IMAGEN05">Imagen 5</option>
    </select>

    <span> ) </span>

    <div style="text-align:center; padding: 10px;">
      <input type="button" id="btGenerar"  value="Procesar"
          onclick="accionBotonProcesarOperacionYMostrarBotones (
/*iPestagna*/ 0 ,
/*idNodoDOMcontenedor*/ 'divCALIMANBotonerasContenedor' ,
/*idNodoDOMasa*/ 'divCALIMANBotonerasAsa' ,
```

```
/*strParameterizedCommand*/ '_____PARAM_____ =
    sumarImagenes( _____PARAM_____, _____PARAM_____ )' ,
/*arrStrIDsOrigenes*/
['lbImagenDestino0_','lbImagenOrigen1_','lbImagenOrigen2_'] ,
/*strReplacementSignal*/ '_____PARAM_____' );">

      <input type="button" id="btCancelar" value="Cancelar"
          onclick="accionBotonCancelar (
/*iPestagna*/ 0 ,
/*idNodoDOMcontenedor*/ 'divCALIMANBotonerasContenedor' ,
/*idNodoDOMasa*/ 'divCALIMANBotonerasAsa' );">
    </div>
  </fieldset>
</div>
```

Esta es la versión definitiva que se muestra al usuario final (ver Figura 7), quien al tenerla en pantalla puede disponerse a asignar valores y desencadenar la siguiente secuencia de eventos en caso de que presione el botón `Procesar`.

Figura 7: Representación definitiva de la caja de diálogo del ejemplo de sumar imágenes.

4.4.2.6 *Asignación de valores a los parámetros de la operación*

El usuario final manipula los controles y selectores e introduce datos en los campos de texto. Es notorio[11] como mientras no haya datos válidos en los controles de entrada de datos, se inhabilita el botón `Procesar`.

4.4.2.7 *Conversión a comando de script*

En el listado de la representación definitiva de la caja de diálogo se puede apreciar lo siguiente:

- Que los controles de entrada de datos, en este caso los selectores, tienen un `id` explícitamente asignado, a saber:

  `lbImagenDestino0_`, `lbImagenOrigen1_` y `lbImagenOrigen2_`.

11 Esto sólo se puede apreciar en el programa en ejecución en el navegador web cuando se trata de componentes con validación de datos.

- Casi hasta el final hay una sección marcada `arrStrIDsOrigenes`, que denota un arreglo de cadenas en Javascript, a saber:

  `['lbImagenDestino0_','lbImagenOrigen1_','lbImagenOrigen2_']`.

  Se observa que coinciden los `id` de los componentes de entrada de datos con las cadenas de texto de dicho arreglo.

- También hay una sección marcada `strParameterizedCommand` con la cadena:

  ```
  _____PARAM_____ = sumarImagenes(
                    _____PARAM_____, _____PARAM_____ ).
  ```

  Esta es una cadena de texto que tiene marcados los lugares en donde, tras una sustitución, aparecerán los valores asignados por el usuario final a los operandos.

- La sección marcada como `strReplacementSignal` corresponde a una cadena de texto que indica cual es el patrón a buscar para realizar las sustituciones, en este caso `_____PARAM_____`.

En el siguiente listado de código fuente se muestra una función que hace lo siguiente: dada una cadena marcada, una marca y un arreglo de cadenas, hace las sustituciones de las marcas por cada uno de los elementos del arreglo de cadenas en su orden natural.

```
//esta función recibe una cadena strEstructura con marcas
    especiales indicadas por strPatronDeSustitucion
//y una por una de sus ocurrencias las va sustitiyendo por los
    elementos del arreglo arrItemsContenido
function verterContenidoEnEstructura(strEstructura,
    arrItemsContenido, strPatronDeSustitucion) {
  var regexpression = eval('/' + strPatronDeSustitucion + '/');
  for (var iItemContenido = 0; iItemContenido < arrItemsContenido
      .length; ++iItemContenido) {
    strEstructura = strEstructura.replace(regexpression,
        arrItemsContenido[iItemContenido]);
  }
  return strEstructura;
}
```

Entonces tras la ejecución de dicha función con los ejemplos de marca, arreglo de cadenas y cadena de plantilla de ejemplo mostrada, la cadena resultante quedaría:

```
lbImagenDestino0_ = sumarImagenes(
                    lbImagenOrigen1_, lbImagenOrigen2_ ).
```

Sirve lo anterior como ejemplo de uso de esa función de sustituciones múltiples. No obstante, para obtener la cadena de script necesaria para efectuar una operación, lo que se hace en la práctica es tomar el arreglo de cadenas de identificadores de objetos del DOM, asumiendo que estos tienen algún valor válido en su propiedad `value`, y construir un nuevo arreglo con los valores almacenados en dichas propiedades según se muestra en la función listada a continuación:

```
//esta función recibe una lista de IDs de elementos DOM que
    tienen la propiedad value
//para cada elemento toma el valor almacenado en dicha propiedad
    y lo mete en un arreglo
//luego usa ese arreglo de valores y los sustituye en la cadena
    de comando de script strParameterizedCommand dada usando
//el patrón de reemplazo strReplacementSignal dado
//regresa la cadena de comando de script con los valores actuales
     tomados de los componentes de entrada de la caja de diálogo
    identificados en la lista dada como entrada
function obtenerCadenaDeScript(strParameterizedCommand, arrIDs,
    strReplacementSignal) {
  var arrStrValues = [];
  for (var iID = 0; iID < arrIDs.length; ++iID) {
    arrStrValues[arrStrValues.length] = document.getElementById(
        arrIDs[iID]).value;
  }
  strParameterizedCommand = verterContenidoEnEstructura(
      strParameterizedCommand, arrStrValues, strReplacementSignal
      );
  return strParameterizedCommand;
}
```

Entonces, suponiendo que cada uno de esos identificadores corresponde a un campo de entrada de texto de una caja de diálogo, se produciría un arreglo con los valores introducidos por el usuario y estos irían a parar a la cadena de comando resultante de la sustitución. Concretamente si en el campo de texto identificado como `lbImagenDestino0_` hay un valor `IMAGEN03`, en el campo de texto identificado como `lbImagenOrigen1_` hay un valor `IMAGEN01` y en el campo de texto identificado como `lbImagenOrigen2_` hay un valor `IMAGEN02`, se formaría un arreglo: `['IMAGEN03', 'IMAGEN01', 'IMAGEN02']` y la cadena resultante quedaría: `IMAGEN03 = sumarImagenes( IMAGEN01, IMAGEN02 )`. Se observa que los valores de los operandos son cadenas de texto.

Esta cadena es la que se envía al programa despachador para su análisis y ejecución, con un previo procesamiento de normalización que se describe más adelante; además se adjunta una copia de dicha cadena en el campo de texto de la ventana de script.

Entonces el desarrollador de operaciones debería preocuparse porque las cajas de diálogo tengan campos de entrada de datos identificables de manera única en el DOM y que se den todas esas correspondencias que permitan obtener dicha cadena de script.

Lo que se acaba de ver es pueden emplearse las cajas de diálogo de las operaciones para ir armando el script; alternativamente se puede también usar el campo de texto de la ventana de script, para ahí teclearlo o bien pegarlo desde el almacén portapapeles (clipboard).

4.4.2.8 *Procesamiento del script en el front-end*

Se construye un encabezado con los siguientes valores separados por un carácter |, a saber: timestamp estilo UNIX, un nombre de usuario, tamaño vertical y horizontal de la miniatura a generar para presentar los resultados de las operaciones. A continuación se concatena la cadena de script. De este modo el script se puede considerar como un archivo con formato *encabezado-datos*.

Antes de enviar la cadena del script al programa despachador se le aplican algunas transformaciones de normalización y protección de símbolos.

En particular:

- Los separadores de nombres de directorio en las rutas son convertidos[12] de '\' a '/'.
- Es necesario proteger ciertos caracteres debido a que la transmisión vía HTTP les modifica irreversiblemente como se indica en los párrafos siguientes. Para esto se usan ciertas sustituciones, seguidas de una codificación en URL-Encoding.

PROTECCIÓN DE LOS CARACTERES DE ESPACIOS: Se efectúa una sustitución de todos los caracteres de espacios por la cadena `___SPC___` , ya que en la etapa de preprocesamiento del script recibido en el programa despachador como parte de la normalización de las cadenas de texto se quitan todos los caracteres de espacios para evitar posiciones espurias de caracteres no alfanuméricos. Esto aplica principalmente a las rutas a un archivo o los nombres que contengan espacios, o aquellas cadenas de texto que se escriban o peguen desde el portapapeles directamente en el script, ya que los nombres introducidos desde cajas de diálogo reciben una corrección y validación previas a su aparición en el script.

PROTECCIÓN DE LOS SIGNOS +: Se efectúa una sustitución de todos los signos + por la cadena `___PLUS___`, ya que los signos + se convierten en caracteres de espacio en las transmisiones vía HTTP hacia el servidor.

4.4.2.9 *Transferencia de datos entre front-end y back-end*

Para la transferencia de ida y vuelta se usa el objeto XMLHTTPRequest del entorno Javascript del navegador web.

12 Esto significa que se asume el estilo UNIX para la escritura de las rutas de archivo ya que el correspondiente estilo de Windows podría no estar disponible en todos los sistemas.

4.4.2.10 *Procesamiento del script en el back-end*

La cadena de texto del script es recibida y procesada en el back-end por el programa despachador.

EL PROGRAMA DESPACHADOR. El propósito de este programa es lanzar, a su vez, a ejecución los programas de las operaciones pasándoles un archivo de configuración con los valores que el usuario introdujo, ya sea directamente o a través de la caja de diálogo de cada operación, para los operandos en los comandos de script. Dicho archivo de configuración consiste en pares llave-valor, uno por cada renglón.

Es necesario analizar línea por línea del script, esperando en cada una de ellas encontrar un comando de los aceptados. Se ignoran las líneas en las que no se encuentre un comando aceptado.

A grandes rasgos lo que hace el programa despachador es:

- respecto del comando encontrado tomar su sintaxis de la lista de comandos aceptados,
- encontrar los tokens separadores (no alfanuméricos),
- usarlos para encontrar los tokens de operandos en la línea del script actual,
- identificar entradas y salidas para administrar el almacén de entradas y salidas respectivamente,
- sustituir los operandos por sus valores almacenados, o almacenar los nuevos,
- armar con esos valores un archivo de configuración,
- lanzar a ejecución el programa de operación correspondiente usando la configuración construida,
- generar una miniatura del resultado obtenido y colocarla en una posición accesible al servidor web,
- reportar la dirección de esta miniatura, el nicho en que debe aparecer y algunas características numéricas de la imagen.

En más detalle, el programa despachador hace lo siguiente:

- Prepara la salida[13] a web usando la salida estándar, enviando por ahí primero la cadena correspondiente al encabezado de un mensaje HTTP especificando el conjunto de caracteres para que se transmitan sin pérdidas los caracteres acentuados.

13 Siguiendo el estándar web CGI y el protocolo HTTP

- Lee[14] la cadena de texto que recibe del navegador web por entrada estándar, y le aplica la siguiente secuencia de transformaciones, a saber:
  - La decodifica pues viene en formato URL-Encoding.
  - Sustituye las ocurrencias del carácter diagonal inversa '\' por el carácter diagonal '/' y quita todas las comillas.
  - Sustituye las ocurrencias de los símbolos `___PLUS___` por el carácter '+'.

  El resultado de esto es el script recibido.
- Separa el script recibido en renglones, donde el primer renglón es un encabezado con una lista de campos separados por el carácter '|'.
- Separa dichos campos y los almacena en un arreglo para su consumo posterior.
- Lee el archivo con la lista de comandos aceptados.
- Normaliza cada renglón de la lista de comandos aceptados con las siguientes sustituciones, a saber:
  - Sustituye las ocurrencias de `VAR ` por `VAR:` para no perder este token al quitar los espacios más adelante.
  - Quita los espacios y tabulaciones.
- Renglón por renglón del script recibido excepto el primero por tratarse de un encabezado, efectúa una búsqueda de comandos aceptados para decidir que acción tomar.

  Ya que el usuario puede introducir lo que el quiera en el campo de texto del script, esta es la última oportunidad para hacer el mejor esfuerzo para tomar lo que sirva e intentar interpretarlo como un comando.

  Entonces, para cada renglón del script recibido hay que buscar una coincidencia en la lista de comandos aceptados por el despachador y si encuentra una hay que configurar la acción correspondiente y ejecutarla.

  Entonces, al encontrar un empate se ejecuta la siguiente secuencia:
  - Normalizar la cadena del renglón actual, esto es, sustituir las ocurrencias de `VAR ` por `VAR:` y luego quitar todos los espacios.
  - Elaboración de una lista de pares llave-valor que conformarán el archivo de configuración correspondiente a la operación actual, que se desarrolla de la siguiente manera:

14 Siguiendo el estándar web CGI

- Primero hay un proceso de aprendizaje de la sintaxis del comando aceptado con el que se logró la coincidencia, esto es, se encuentran las posiciones de los tokens y sus separadores en un renglón por medio de la identificación de los cambios de alfanuméricos a no-alfanuméricos o viceversa, para tomar como separadores los no-alfanuméricos.

  El algoritmo consiste en: almacenar la primera posición, recorrer todas las posiciones de la cadena a analizar y para cada posición encontrar si el símbolo en la posición actual es alfanumérico y si el símbolo en la posición anterior es alfanumérico, con la consideración de que sólo en la primera posición se considera que el símbolo anterior es igual el actual; si la calidad de alfanumérico en el símbolo actual es distinta a la del símbolo anterior, almacenar la posición actual; finalmente después del ciclo, almacenar la última posición más uno.

- Con las posiciones de los tokens en la linea de sintaxis del comando aceptado, se obtienen las llaves en los pares llave-valor antes mencionados para lo cual se separan los tokens en no-alfanuméricos y alfanuméricos, y estos últimos por sus prefijos `ecto` y `endo` en dirección de flujo de salida y dirección de flujo de entrada respectivamente. El resultado de esto es una lista ordenada de tokens no-alfanuméricos, una lista ordenada de tokens alfanuméricos, una lista ordenada de banderas indicadoras de la dirección de flujo, `0`, `1`; donde el número `0` nos recuerda la o de output y el `1` nos recuerda la i de input.

- Con dichas listas se analiza el renglón actual del script recibido, encontrando primero las posiciones de sus tokens usando para ello la lista ordenada de tokens no-alfanuméricos obtenida anteriormente.

  El algoritmo consiste en: almacenar la primera posición, recorrer todos los tokens no-alfanuméricos y encontrar su siguiente ocurrencia en la cadena y almacenar su posición, finalmente después del ciclo, en caso de que en la lista de posiciones no esté el extremo del final, agregarlo; esto porque en las líneas de asignación de variables usando la palabra reservada `VAR` no hay token separador al final de la línea.

- Con las posiciones de los tokens en la línea actual de script recibido, se extraen los mismos como subcadenas correspondientes a esas posiciones. Dependien-

do de como fueron clasificadas en la línea de sintaxis del comando aceptado, estos valores se consideran llaves de búsqueda en el almacén de imágenes y valores, o bien como los valores mismos. Esta clasificación la dan las palabras `ValorNumerico`, `ArchivoResultado`, `ArchivoOperando`, `ValorResultado` y los prefijos `ecto` y `endo`.

- Se lee el archivo del almacén de imágenes y valores para tener la versión más actual en memoria.
- Primero se procesan los tokens marcados como entradas.
  - si el token está marcado como `ArchivoOperando` y la llave está en el almacén, se toma el valor correspondiente del almacén como nombre de archivo.
  - si el token está marcado como `ArchivoOperando` y la llave no está en el almacén, se toma desde el script como nombre de archivo.
  - si el token está marcado como `ValorNumerico` y la llave está en el almacén, se toma el valor correspondiente del almacén como valor numérico.
  - si el token está marcado como `ValorNumerico` y la llave no está en el almacén, se toma desde el script como valor numérico y se almacena también en un buffer de valores numéricos.
  - en cualquier otro caso se toma lo que haya en la línea del script actual en la posición del token actual.
- Luego se procesan los tokens marcados como salidas.
  - si el token está marcado como `ArchivoResultado` y contiene un punto, se toma desde el script como nombre de archivo.
  - si el token está marcado como `ArchivoResultado` y no contiene un punto, se genera y almacena un nombre de archivo para él. Dicho nombre de archivo es para procesamiento interno, el usuario final no se entera de esto.
  - si el token está marcado como `ValorNumerico` y la llave se encuentra en el almacén, se toma el valor correspondiente del almacén.
  - si el token está marcado como `ValorNumerico`, la llave no se encuentra en el almacén y el buffer de valores numéricos no está vacío, se toma el valor numérico desde el buffer de valores numéricos.

◇ si el token está marcado como `ValorNumerico`, la llave no se encuentra en el almacén y el buffer de valores numéricos está vacío, se toma el valor numérico desde el script.

◇ si el token está marcado como `ValorResultado` y contiene un punto, se toma el valor desde el script como nombre de archivo para guardar ahí el valor de la variable correspondiente.

◇ si el token está marcado como `ValorResultado` y no contiene un punto, se genera y almacena un nombre de archivo para guardar ahí el valor de la variable correspondiente.

◇ en cualquier otro caso se toma lo que haya en la línea del script actual en la posición del token actual.

Los valores tomados se almacenan en los pares llave-valor correspondientes dado que ya existen en ellos las llaves.

- Con la lista de pares llave-valor se genera un archivo de configuración para el programa de la operación.
- Se manda a llamar a dicho programa.
- Se genera un archivo de configuración para el programa que genera una versión miniatura o de tamaño fijo de la imagen que se abrió o generó.
- Se manda a llamar a dicho programa y la cadena de texto con el nombre de la miniatura generada se escribe en la salida estándar.
- El programa que genera las miniaturas también calcula algunas características numéricas de la imagen, estas son reportadas en la salida estándar.
- Pasa a la siguiente línea de script recibida.

- Si dicho empate falla, no hay más que hacer y se continúa con siguiente renglón del script recibido (en caso de haberlo) o se termina la ejecución.

4.4.2.11 *Llamada a operaciones, un breve ejemplo*

A continuación se presenta un breve ejemplo de como se obtiene el archivo de configuración a partir de la línea de script recibida en el programa despachador.

Suponiendo que al programa despachador le toca procesar la siguiente línea de script recibida:

```
IMAGEN01 = sumarImagenes( IMAGEN01, IMAGEN01 ),
```

mientras la siguiente cadena está en el almacén de imágenes y variables, indicando que se tiene cargada una imagen en el nicho 1:

`IMAGEN01¬135031378599549047 1AlfCeseQ_-_cargarImagen.mtx`

y con la siguiente línea de sintaxis para ese comando en la lista de comandos aceptados:

```
ectoArchivoResultado1=sumarImagenes(
                        endoArchivoOperando1,endoArchivoOperando2)
```

El proceso se lleva a cabo como sigue:

- Leer siguiente comando de script : `IMAGEN01 = sumarImagenes( IMAGEN01, IMAGEN01 )`
- Identificar el programa a llamar encontrando una coincidencia en la lista de comandos aceptados: `sumarImagenes`
- Cargar y analizar su línea de sintaxis desde la lista de comandos aceptados: `ectoArchivoResultado1=sumarImagenes(endoArchivoOperando1 ,endoArchivoOperando2)`

Listado 10: Análisis de línea de sintaxis para el comando sumarImagenes como aparece en la bitácora del programa despachador.

```
CALIMAN_despachadorDeScripts : primera ocurrencia de '
    sumarImagenes' encontrada en: 11
ectoArchivoResultado1=sumarImagenes(endoArchivoOperando1,
11111111111111111111011111111111110111111111111111111110
00000000000000000000011000000000000110000000000000000001

endoArchivoOperando2)
11111111111111111110 : mapa de alfanumericos
10000000000000000001 : mapa de cambios alfanum/no-alfanum

0, 21, 22, 35, 36, 56, 57, 77, 78 : posiciones de
    demarcacion en el comando registrado
ArchivoResultado1 (de salida) | ArchivoOperando1 (de entrada
    ) | ArchivoOperando2 (de entrada) : tokens alfanumericos
     encontrados (el comando actual no se incluye en esta
    lista)
= | ( | , | ) : tokens no alfanumericos encontrados
```

Se observa en el listado que la cadena sumarImagenes fue encontrada en la posición 11 de la línea de script recibida. Luego, aparecen tercias de renglones. En el primer renglón de cada tercia se muestra la cadena de sintaxis del comando detectado, tomada de la lista de comandos aceptados. En el siguiente renglón de la tercia se indican con unos y ceros la calidad de alfanumérico o no-alfanumérico, respectivamente, de los caracteres correspondientes del renglón de arriba. En el tercer renglón de la tercia se indica con unos que los caracteres anterior y actual

del primer renglón son de calidades distintas y con ceros que son de calidades iguales, esto es, se indican las posiciones donde se detectaron cambios de clase del carácter de no alfanumérico a alfanumérico o viceversa. Con esas posiciones es posible detectar las palabras clave. Luego se muestra como las palabras clave son clasificadas como de salida o entrada. Finalmente se muestra la lista de tokens no alfanuméricos encontrados que sirven de separadores de palabras.

La lista de tokens alfanuméricos aporta la parte de las llaves en los pares llave-valor del archivo de configuración que se construirá.

- Analizar la línea de script recibida.

  Con la lista de separadores es posible extraer los tokens en la línea de script recibida como se muestra en el siguiente listado.

Listado 11: Análisis de línea de script donde se encontró el comando sumarImagenes.

```
0, 8, 9, 22, 23, 31, 32, 40, 41 : posiciones de demarcacion
    en la línea de comando recibida :
IMAGEN01=sumarImagenes(IMAGEN01,IMAGEN01)
IMAGEN01 | = | sumarImagenes | ( | IMAGEN01 | , | IMAGEN01 |
     ) : tokens encontrados
 token[0] : IMAGEN01 | separador : (=) | comando actual:
     sumarImagenes
 token[1] : = | separador : (=) | comando actual:
     sumarImagenes
 token[2] : sumarImagenes | separador : (() | comando actual
     : sumarImagenes
 token[3] : ( | separador : (() | comando actual:
     sumarImagenes
 token[4] : IMAGEN01 | separador : (,) | comando actual:
     sumarImagenes
 token[5] : , | separador : (,) | comando actual:
     sumarImagenes
 token[6] : IMAGEN01 | separador : ()) | comando actual:
     sumarImagenes
 token[7] : ) | separador : ()) | comando actual:
     sumarImagenes
IMAGEN01 | IMAGEN01 | IMAGEN01 : nombres de nichos
    encontrados
```

En el listado aparecen primero las posiciones donde se encontraron en la línea de comando recibida, en orden, los tokens separadores obtenidos anteriormente. Esto establece una separación en tokens para finalmente extraer las llaves, indicadas como nombres de nichos encontrados, que identifican a los pares que están en el almacén de imágenes y variables.

- Obtención de valores para configuración del programa de la operación.

  Habiendo establecido la correspondencia entre nombres de llaves, se procesan los tokens de entrada y de salida para saber de donde tomar los valores para asociarlos a las llaves existentes en los pares llave-valor del archivo de configuración que se construirá.

  Listado 12: Procesamiento de tokens de entrada y de salida.

```
CALIMAN_despachadorDeScripts : caso entrada ArchivoOperando

CALIMAN_despachadorDeScripts : nombre de archivo tomado del
    almacen

CALIMAN_despachadorDeScripts : caso entrada ArchivoOperando

CALIMAN_despachadorDeScripts : nombre de archivo tomado del
    almacen

CALIMAN_despachadorDeScripts : caso salida ArchivoResultado

CALIMAN_despachadorDeScripts : nombre de archivo tomado del
    almacen tras almacenarlo tomado de una generacion de
    cadena in-situ

CALIMAN_despachadorDeScripts : strNombreDeArchivo:

CALIMAN_despachadorDeScripts : strTimestamp:1350313847633

CALIMAN_despachadorDeScripts : strUsername:AlfCeseQ

CALIMAN_despachadorDeScripts : strNombreOperacion:
    sumarImagenes

CALIMAN_despachadorDeScripts : strExtension:.mtx

CALIMAN_despachadorDeScripts : nombre generado para archivo
    :1350313847633490471AlfCeseQ_-_sumarImagenes.mtx
```

  Se observa en el listado el resultado de analizar las entradas y salidas, de manera que se determina de donde tomar los valores que se necesitan a continuación para terminar de formar los pares llave-valor del archivo de configuración que está por construirse.

  Como en este ejemplo, el resultado de la operación se almacena en el nicho 1, que ya estaba ocupado, en el almacén de imágenes y valores queda almacenada la siguiente cadena que sustituye a la anterior con la misma llave: `IMAGEN01¬1350313847633490471 AlfCeseQ_-_sumarImagenes.mtx`

- Construcción del archivo de configuración y lanzamiento a ejecución del programa de la operación.

  Con los pares llave-valor completos, se pueden generar los archivos de configuración y las líneas de comando para la ejecución de los programas de operaciones y miniaturización como se muestra en el siguiente listado. La miniaturización es el proceso de copiar una imagen en otra de un tamaño pequeño pero fijo. Usualmente se especifican tamaños de 128x128 o 200x200 pixeles para el despliegue de estas copias en miniatura en los nichos para imágenes. Esta información va especificada en el encabezado del script.

Listado 13: Conformación de los archivos de configuración y líneas de comando para la ejecución de los programas de operaciones y miniaturización.

```
CALIMAN_despachadorDeScripts : miniaturas a generar:
1350313847633490471AlfCeseQ_-_sumarImagenes.mtx


CALIMAN_despachadorDeScripts : Mapa Keys-Values:
par 'ArchivoOperando1' ,'1350313785995490471AlfCeseQ_—
    _cargarImagen.mtx'
par 'ArchivoOperando2' ,'1350313785995490471AlfCeseQ_—
    _cargarImagen.mtx'
par 'ArchivoResultado1' ,'1350313847633490471AlfCeseQ_—
    _sumarImagenes.mtx'


CALIMAN_despachadorDeScripts : Contenido del archivo de
    configuracion:
ArchivoOperando1=1350313785995490471AlfCeseQ_-_cargarImagen.
    mtx
ArchivoOperando2=1350313785995490471AlfCeseQ_-_cargarImagen.
    mtx
ArchivoResultado1=1350313847633490471AlfCeseQ_-
    _sumarImagenes.mtx


CALIMAN_despachadorDeScripts : strNombreDeArchivo:

CALIMAN_despachadorDeScripts : strTimestamp:1350313847633

CALIMAN_despachadorDeScripts : strUsername:AlfCeseQ

CALIMAN_despachadorDeScripts : strNombreOperacion:
    sumarImagenes

CALIMAN_despachadorDeScripts : strExtension:.conf
```

```
CALIMAN_despachadorDeScripts : nombre generado para archivo
    :1350313847633554288AlfCeseQ_-_sumarImagenes.conf


CALIMAN_despachadorDeScripts : Linea de comandos a ejecutar:
sumarImagenes ./runtimeConfigs/1350313847633554288AlfCeseQ_-
    _sumarImagenes.conf >> stdout.txt


CALIMAN_despachadorDeScripts : Contenido del archivo de
    configuracion de miniatura:
ArchivoOperando1=13503138476334904 71AlfCeseQ_-_sumarImagenes
    .mtx
ArchivoResultado=../../../MyWebSites/CALIMAN/miniaturas
    /13503138476334904 71AlfCeseQ_-_sumarImagenes_mtx.png
TamagnoVertical=196
TamagnoHorizontal=196



CALIMAN_despachadorDeScripts : Linea de comandos a ejecutar
    para miniaturizar:
CALIMAN_GenerarImagenMiniatura ./runtimeConfigs
    /1350313847633251171AlfCeseQ_-_MINIATURA.conf >> stdout.
    txt
```

4.4.2.12 *Retorno de valores*

Cada evento que es reportado por el programa despachador se escribe en su salida estándar por lo que es enviado al programa en el navegador web. Para el ejemplo en curso la salida se muestra en el listado siguiente:

Listado 14: Características numéricas de la imagen resultante y presentación para su carga en el nicho correspondiente.

```
CALIMAN_despachadorDeScripts : ../../../MyWebSites/CALIMAN/
    miniaturas/13503138476334904 71AlfCeseQ_-_sumarImagenes_mtx.
    png.metadatos:
R[0..510], G[0..510], B[0..510], H:128px, V:128px, AvgR:263.719,
    StdDevR:160.847, VarR:25871.8, SumR:4.32077E+006, ProdR:-1.#
    IND, Nrm1R:4.32077E+006, Nrm2R:1.56335E+009, AvgG:263.719,
    StdDevG:160.847, VarG:25871.8, SumG:4.32077E+006, ProdG:-1.#
    IND, Nrm1G:4.32077E+006, Nrm2G:1.56335E+009, AvgB:263.719,
    StdDevB:160.847, VarB:25871.8, SumB:4.32077E+006, ProdB:-1.#
    IND, Nrm1B:4.32077E+006, Nrm2B:1.56335E+009


CALIMAN_despachadorDeScripts : Relacion de nichos y miniaturas:
 cargarImagenEnNicho::IMAGEN01 ¬
     13503138476334904 71AlfCeseQ_-_sumarImagenes_mtx.png ¬R
    [0..510], G[0..510], B[0..510], H:128px, V:128px, AvgR
```

```
:263.719, StdDevR:160.847, VarR:25871.8, SumR:4.32077E+006,
ProdR:-1.#IND, Nrm1R:4.32077E+006, Nrm2R:1.56335E+009, AvgG
:263.719, StdDevG:160.847, VarG:25871.8, SumG:4.32077E+006,
ProdG:-1.#IND, Nrm1G:4.32077E+006, Nrm2G:1.56335E+009, AvgB
:263.719, StdDevB:160.847, VarB:25871.8, SumB:4.32077E+006,
ProdB:-1.#IND, Nrm1B:4.32077E+006, Nrm2B:1.56335E+009
```

Se puede apreciar en el listado como se reporta el nombre y ruta al archivo de miniatura generado junto con las palabras clave `IMAGEN` y `cargarImagenEnNicho:` que se detectan del lado del cliente para colocar la imagen en el nicho correspondiente. También se calculan algunas características numéricas de la imagen antes de ser miniaturizada y se presentan en cadenas separadas por el símbolo ¬.

Las características numéricas que se calculan son:

- Rango dinámico del canal R (rojo)
- Rango dinámico del canal G (verde)
- Rango dinámico del canal B (azul)
- Tamaño horizontal de la imagen
- Tamaño vertical de la imagen
- Promedio de intensidades en el canal R
- Desviación estándar de intensidades en el canal R
- Varianza de intensidades en el canal R
- Suma de intensidades en el canal R
- Producto de intensidades en el canal R
- Norma 1 de intensidades en el canal R
- Norma 2 de intensidades en el canal R
- Promedio de intensidades en el canal G
- Desviación estándar de intensidades en el canal G
- Varianza de intensidades en el canal G
- Suma de intensidades en el canal G
- Producto de intensidades en el canal G
- Norma 1 de intensidades en el canal G
- Norma 2 de intensidades en el canal G
- Promedio de intensidades en el canal B
- Desviación estándar de intensidades en el canal B

- Varianza de intensidades en el canal B
- Suma de intensidades en el canal B
- Producto de intensidades en el canal B
- Norma 1 de intensidades en el canal B
- Norma 2 de intensidades en el canal B

Estos datos los puede leer el usuario para cada imagen cargada en el campo de texto ubicado bajo cada nicho para imágenes.

En caso de que algún programa falle, el servidor web tiene un tiempo de espera configurable, por default de 5 minutos, pasado el cual reporta un error de ejecución con el código HTTP 500. En el cliente, el resultado de una falla en el servidor es un tiempo de espera inusualmente largo y en ocasiones una marca de imagen de miniatura no encontrada. Por ejemplo, en caso de un error de punto flotante por intento de división entre cero en alguna operación, la miniatura no se genera porque el programa de operación termina su ejecución antes de escribir la imagen resultante, y en consecuencia el programa generador de miniaturas no encuentra su correspondiente archivo de entrada por lo que termina su ejecución sin generar una miniatura. El usuario se puede dar cuenta de esto si la operación tarda más de lo usual o si solamente ve que aparece el icono de imagen no cargada que provee el navegador.

4.4.2.13 *Despliegue de resultados*

La cadena de texto que contiene todo lo que el programa despachador escribió en su salida estándar es separada en renglones. De todos los renglones sólo se toman en cuenta aquellos que contengan las cadenas: `cargarImagenEnNicho:` e `IMAGEN` ya que se asume que contienen los datos necesarios para mostrar una imagen particular en un nicho específico incluyendo los datos numéricos que se calcularon al generar la miniatura de la misma. La imagen que se muestra en el nicho es una versión miniatura o de tamaño fijo de la imagen que se abrió o generó.

Las operaciones se efectúan con las imágenes en su tamaño original, mientras que los resultados se muestran con copias en miniatura. Esto concluye la exposición de la solución propuesta.

5 RESULTADOS

En este capítulo se discuten los resultados del proyecto, se muestran evidencias de su ejecución, se dan algunas conclusiones y se plantea trabajo a futuro.

5.1 PRUEBAS

En esta sección se mencionan algunos de los aspectos del ambiente de prueba empleado así como una discusión de los resultados obtenidos.

5.1.1 *Características del ambiente de prueba.*

Debido a su arquitectura desacoplada el programa se puede ejecutar remotamente[1]. Esto significa que el cliente y el servidor pueden estar en máquinas distintas, no obstante, limitamos la mayoría de las pruebas a ejecución local y fueron pocas las pruebas, por su naturaleza especulativa, en dispositivos móviles selectos.

5.1.1.1 *Hardware y sistemas operativos*

Las pruebas se llevaron a cabo en una computadora laptop con procesador Intel Core i7 Q740@1.73GHz de 64bit, con 4GB de Random Access Memory o Memoria de Acceso Aleatorio (RAM) y sistema operativo Windows 7 Home Premium, una computadora laptop con procesador Intel Pentium III @650MHz de 32bit, con 128MB de RAM y sistema operativo Windows XP Professional con Service Pack 3, así como con dos computadoras de tableta, a saber: una Nook Color de la librería Barnes & Noble y un iPod Touch de tercera generación de la marca Apple. Poder usar CALIMAN en tan variados dispositivos fue posible gracias al empleo de tecnologías web.

5.1.1.2 *Software de sistema*

SERVIDORES WEB : Se eligieron por su disponibilidad en varios sistemas operativos y su relativa facilidad de instalación y configuración el servidor Apache HTTPD y el servidor Xitami; ambos emplean su propio archivo de configuración para modificar su comportamiento.

1 Es importante hacer notar que aunque la ejecución remota es posible, no fue prioritario darle soporte y seguimiento a tal capacidad en una primera iteración de desarrollo por ser más importante el remozamiento de un programa que originalmente fué hecho para correr localmente.

El servidor Apache es de los dos elegidos aquel cuya instalación y configuración es menos inmediata. Concretamente, en Windows, se puede ejecutar el servidor directamente desde la cuenta de usuario pero no sin tener una ventana de consola abierta, lo que provoca que todas las llamadas a los programas de operaciones abran también su ventana de consola. Por otra parte, si se instala como servicio se observa que no aparecen esas ventanas de consola en la ejecución de cada programa de operación, pero está el inconveniente de tener que instalar aparte el servidor con privilegios de administrador.

En cambio, el servidor Xitami, mientras el puerto TCP configurado esté disponible, se ejecuta en un proceso sin interfaz de usuario[2]. Esto es deseable para hacer transparente al usuario final el funcionamiento de CALIMAN.

NAVEGADORES WEB : Se eligió el navegador Google Chrome por las ventajas técnicas[3] ofrecidas por su motor de navegación web conocido como WebKit respecto a competidores como Mozilla Firefox o Microsoft Internet Explorer, aún en sus versiones más recientes. Se tomaron sólo estos navegadores de la multitud que hay debido a que son los que implementan de manera más completa los estándares y recomendaciones del W3C. El propósito de estas pruebas no fue comparar cual de los navegadores es el más rápido o mejor respecto a alguna métrica dada, sino comprobar que funcionen correctamente las características de la calculadora de imágenes.

5.1.1.3 *Objetos Testigo*

Se implementó la operación de suma de imágenes para que sirviera de testigo en todas las etapas de desarrollo de una operación. Luego de que se confirmó que funcionó la propuesta completa se implementaron las demás.

5.1.1.4 *Metodología*

Para la parte del cliente se emplearon las herramientas de depuración incluidas en el navegador web mientras que para la parte del servidor las correspondientes del ambiente de programación Qt Creator, además de un envío frecuente de mensajes a consola para dar seguimiento en tiempo de ejecución fuera de los ambientes de programación.

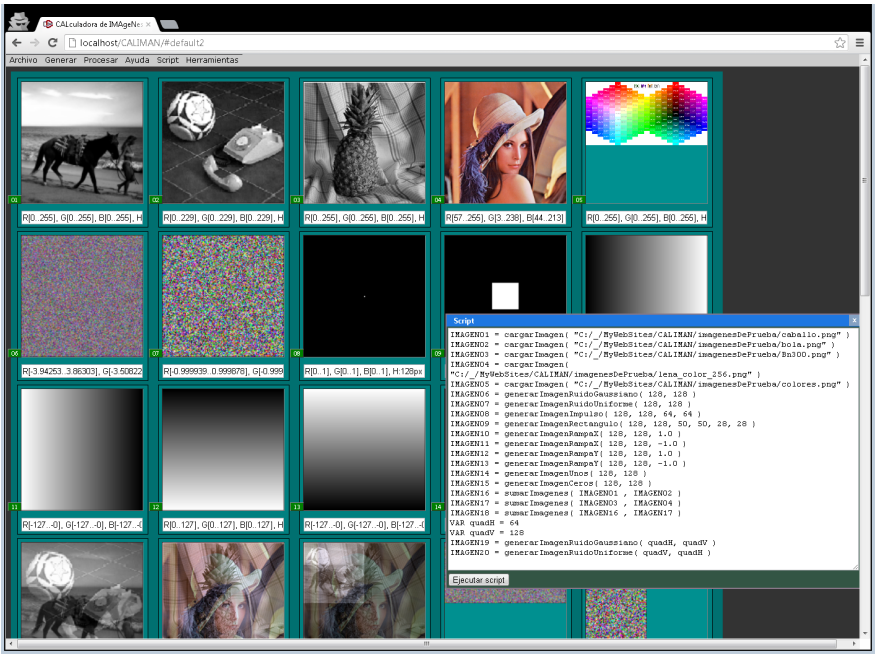
2 Excepto un menú accesible desde la bandeja de notificaciones de Windows.

3 Particularmente, este motor de navegación es el mismo que se emplea en muchos dispositivos móviles como iOS de Apple y Android de Google. Otras ventajas son su potente máquina de Javascript aunado a la riqueza de su DOM.

5.2 RESULTADOS Y DISCUSIÓN

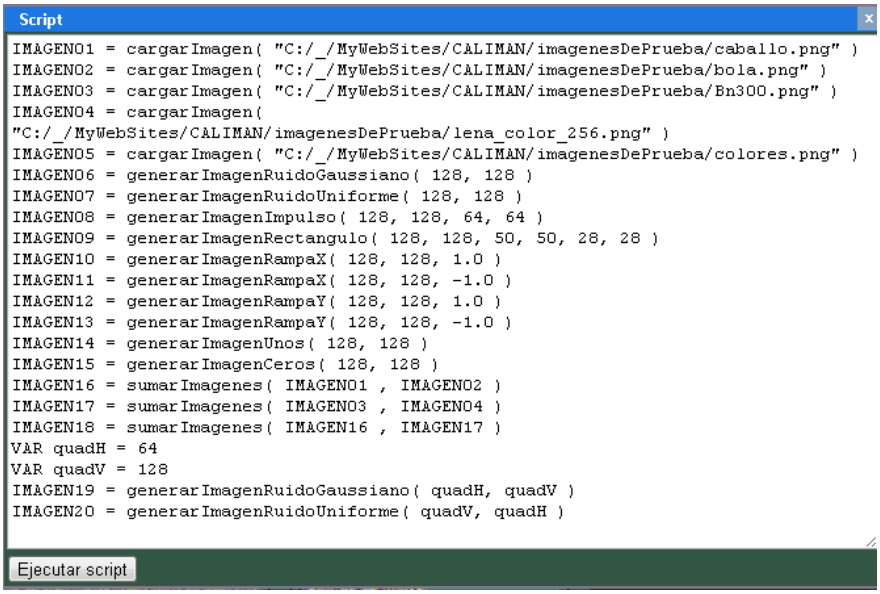
En esta sección se presentan capturas de pantalla de la ejecución del proyecto y discutimos sobre las mejoras obtenidas.

(a) Menú de bienvenida.

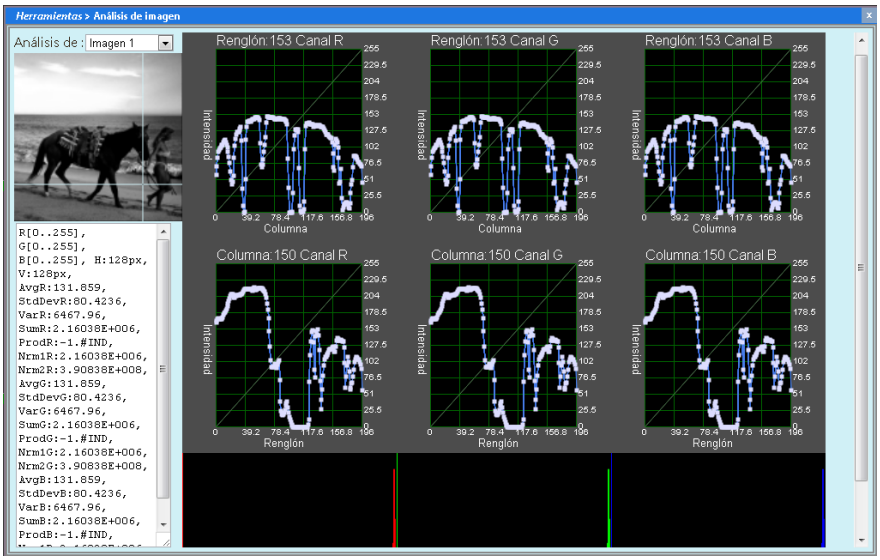


(b) Un script y sus resultados.

Figura 8: Menú de bienvenida y un script con sus resultados.



(a) Ventana para scripts.



(b) Caja de dialogo de Análisis de imágenes.

Figura 9: Ventana de scripts y de análisis de imágenes.

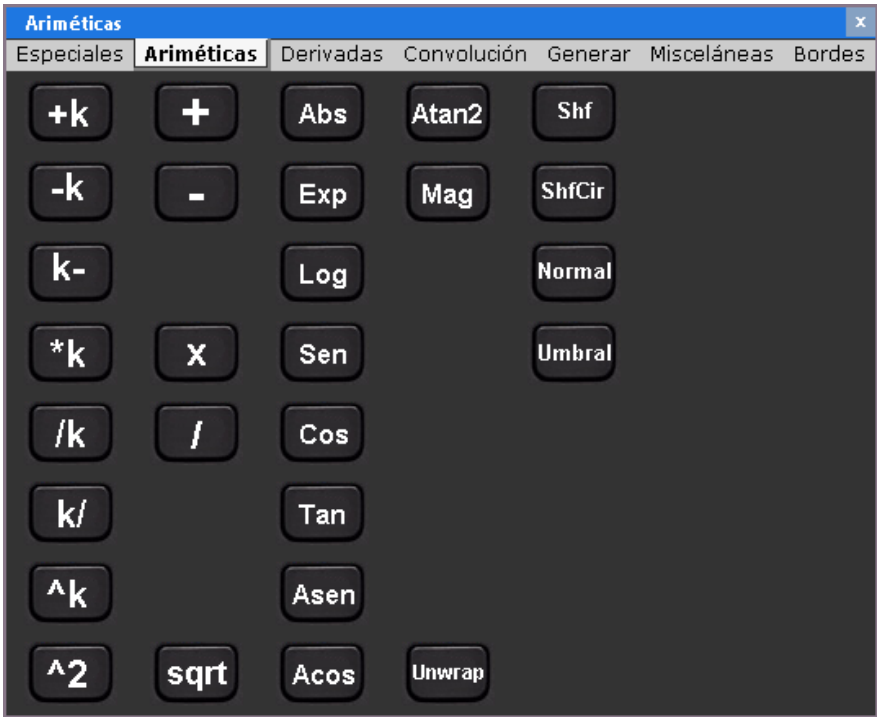
5.2.1 *Muestra de evidencias*

Como evidencia documental del programa en ejecución se presentan y describen algunas capturas[4] de pantalla representativas de su funcionamiento.

En la figura 8 se muestran las capturas de pantalla del menú de bienvenida y del resultado de la ejecución de un script.

4 En la versión electrónica de este documento se podría emplear la función de acercamiento del software lector de PDF para apreciar los detalles.

(a) Botonera Generar imágenes.



(b) Botonera Operaciones aritméticas.

Figura 10: Botoneras.

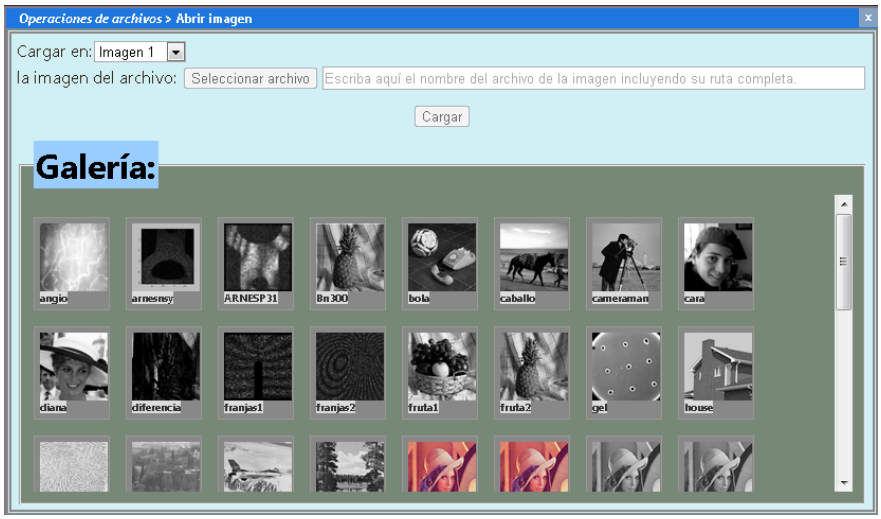
El menú de bienvenida tiene el propósito de permitir acceder a las cajas de diálogo relativas a las tareas que se consideran más relevantes al momento de iniciar una sesión en CALIMAN.

En la figura 8b se muestran los resultados de la ejecución de un script y se puede apreciar que la cantidad de nichos difiere respecto a la versión anterior de CALIMAN, además de que se muestran imágenes en color.

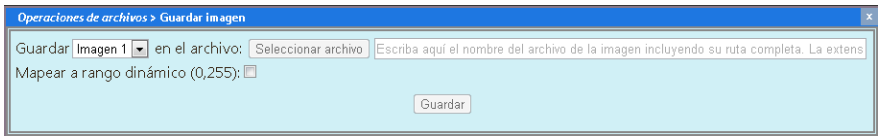
En la figura 9 se muestran las capturas de pantalla de la ventana para scripts y la caja de diálogo de análisis de imágenes.

Un acercamiento al script permite ver que ahora se pueden emplear variables. Se observa la regularidad de los comandos del script, lo cual es útil para los principiantes.

En la caja de diálogo de análisis de imágenes se puede, para la imagen seleccionada, leer las características numéricas calculadas además



(a) Caja de dialogo de AbrirImagen.



(b) Caja de diálogo de GuardarImagen.

Figura 11: Cajas de diálogo de abrir y guardar imágenes.

de poder tomar cortes horizontales o verticales por canal de color de manera interactiva y ver sus histogramas por canal de color.

En la figura 10 se muestran las capturas de pantalla de las botoneras de generar imágenes y de operaciones aritméticas.

En la figura 11 se muestran las capturas de pantalla de las cajas de diálogo para abrir y guardar imágenes.

Se puede apreciar que ahora se incluye una galería de imágenes de prueba en la caja de diálogo para abrir imágenes y que es posible (en caso de así requerirse) guardar la imagen sin aplicarle[5] previamente una normalización para despliegue en pantalla.

5.2.2 *Discusión sobre las mejoras obtenidas*

¿Cómo se miden las mejoras obtenidas con esta propuesta? Para hablar de mejoras, se tiene que hablar de mediciones, y la medición que es de interés tomar se refiere a un costo cualitativo relativo de mantenimiento del software, que contemple el esfuerzo requerido al usuario y no se calcule[6] en términos cuantitativos de tiempo o recursos humanos. En otras palabras, para estimar el costo relativo de

5 Puede ser útil en las imágenes que cumplen con tener el rango dinámico en enteros dentro del intervalo de 0 a 255.

6 Esto debido a la dificultad involucrada para obtener dichas mediciones además de la gran variabilidad inducida por subjetividad y la poca relevancia que tiene tomarlas cuando ya se poseen aproximaciones convenientes.

mantenimiento, que por la complejidad en su obtención se considera aproximadamente, más como un dato cualitativo, no se toman en cuenta la cantidad de personas o el tiempo que se requiere para llevarlo a cabo pero si se toma en cuenta el esfuerzo requerido, medido en cantidad relativa de información que el usuario debe introducir para realizarlo. Esto significa que se debe establecer en comparación con otro costo de la misma naturaleza.

Concretamente, supóngase un proceso de mantenimiento o extensión al sistema que requiere que se proporcionen $n > 0$ identificadores de objetos de tamaño t y que para introducirlos al sistema un método A requiere llevar a cabo pasos de captura de datos que incluyen a los n identificadores además de otros encabezados o información subyacente de tamaño constante $m > 0$ y otro método B sólo requiere capturar los $n > 0$ identificadores de dicho tamaño t. Se puede afirmar que el método A requiere un esfuerzo de $m + t * n$ mientras que el método B un esfuerzo de $t * n$, de donde se puede apreciar que $t * n < m + t * n$ y se puede concluir cualitativamente que el método B requiere menos costo de mantenimiento que el método A.

Para fijar ideas respecto a la comparación de beneficios se puede considerar la siguiente situación hipotética en la que se tienen que implementar una cantidad no trivial de operaciones, digamos 50, bajo las condiciones de la implementación de CALIMAN anterior a este trabajo. Si no se tiene conocimiento de cómo se arman las operaciones hay que invertir previamente un tiempo en estudiar el código fuente y las notas de ayuda, analizarles y asimilarles para finalmente decidir qué sirve para el propósito. Pasado este tiempo inicial que no necesariamente es trivial o poco, dependiendo de la experiencia de la persona, se procede al desarrollo de las operaciones, donde para cada una hay que construir el andamiaje, codificar la operación, armar su interfaz, programar la conexión de datos, y en su caso las verificaciones de dominios de discurso de entradas y salidas, etcétera. Visto así, todo este proceso es muy tedioso y rebuscado, similar al de conectar un circuito integrado a una tabla de prototipado, esto es, donde a cada componente hay que ponerle su cableado en cada una de las líneas de señal que lo requiera. Bajo estas condiciones al tener una mala conexión o la repentina necesidad de hacer alguna modificación, hay que empezar de nuevo algunas reconexiones, tal vez no todas, pero si hay que mover muchas cosas, además de que hay que saber donde, siempre cuidando que no haya faltantes y todo ello sin perder detalle de las consecuencias de los cambios. En cambio en las condiciones que se obtienen con la solución que se propone en este trabajo, tras proporcionar algunos datos esenciales de cada operación, todos los andamios y conexiones se establecen en el fondo, esto es, sin intervención directa del desarrollador, o bien, recibiendo asistencia en su construcción, potencialmente logrando con ello un ahorro considerable en tiempo y esfuerzo. En caso de haber

cambios, no es en muchos lugares donde se hacen y, aunque hay que tener mucho cuidado también, se puede decir que la asistencia que se ofrece proporciona mucho 'colchón tecnológico' provisto en la posible corrección automática o asistida de errores; esa es una de las contribuciones más fuertes de este trabajo. No obstante, no debe resultar sorprendente que el costo verdadero de producción nunca deja de existir, pues sólo se ha ocultado tras pilas de abstracciones; es como levantar la alfombra y meter abajo todo el polvo con el fin de que las visitas no lo vean o como tener un peligroso dragón encerrado en el sótano con la esperanza de que nunca pueda salir. Concretamente, se puede apreciar en la capacidad de extensión que hay límites, esto es, no se puede construir todo con estos componentes. La esperanza es que los esfuerzos alcanzados con este trabajo amplíen con mucho la cobertura que tenía el anterior y en consecuencia quede un patio más amplio para actuar. Los límites siguen ahí aunque, ahora, más lejos.

5.3 CONCLUSIONES

Se implementó una prueba sustancial de concepto de la solución a una problemática identificada en un programa de procesamiento de imágenes. Dicha problemática pudo ser atendida, a grandes rasgos, con un desarrollo tecnológico que fusionó varias ideas acerca de la composición de software por medio de plantillas basadas en texto junto con un mecanismo de despachamiento[7] dinámico; para lo cual se llevó a cabo la integración de los subsistemas propuestos que aprobaron evaluaciones de su viabilidad tales como demostraciones de funcionamiento por partes y valoraciones respecto a otras soluciones existentes; todo ello tras una investigación y descripción del estado del arte de las tecnologías pertinentes para resolverla.

Se encontró que la propuesta de solución al problema se basó en una estrategia tipo 'divide y vencerás' que se materializó a través de una arquitectura de software basada en tecnologías web. Concretamente, se observó un planteamiento consistente en varias capas o niveles de detalle. Cada uno exhibiendo su necesidad de ser tratado tanto de manera local, considerando sus minucias y mecanismos paso a paso, para dar cuenta de sus flujos internos de datos, con sus transformaciones entre formatos y el cumplimiento de los requisitos para su transferencia y almacenamiento; así como de manera global con la intención de que las piezas disjuntas por su funcionalidad, por su construcción o por la tecnología en que estuvieran basadas pudieran mezclarse para alcanzar las metas propuestas; todo ello adoptando una metodología de desarrollo semejante a la conocida como

7 El despachamiento dinámico o enlace tardío hace posible programar respecto a una especificación y posteriormente enlazar a una implementación en tiempo de ejecución.

incremental iterativa, esto es, comenzando con algo sencillo y repetir los pasos de análisis y diseño junto con incrementos en el código y pruebas de funcionamiento hasta obtener los resultados esperados de acuerdo a los objetivos fijados.

La idea que se muestra en funcionamiento en este trabajo es que usando plantillas es posible basarse en el modelo de datos para inferir de alguna manera un controlador y una vista. Se puede hablar entonces de un programador de plantillas y un programador consumidor de plantillas, de modo que el controlador y la vista no son manipulados directamente por el programador consumidor ante cambios al modelo, ya que dependen de como estén definidos, tanto las plantillas como sus modos de uso.

En una primera instancia, el programador de plantillas propone sus definiciones junto con sus modos de uso y las conexiones con los modelos de datos con una menor frecuencia que los cambios en estos; luego con sólo cambiar los datos o incluso con cambios en el modelo aunque sin salirse de las plantillas, se obtienen sin mayor esfuerzo convenientes controladores y vistas. Como consumidor de las plantillas no hay que programar explícitamente el controlador y sus vistas más allá de lo establecido en las plantillas mismas. Esto representa una ventaja en tiempo respecto al proceso de desarrollo usual, en donde dados los datos y su modelo, hay que programar sus conexiones, luego sus controladores, luego sus vistas; todo de manera explícita.

Claramente, quien define las plantillas y sus modos de uso se lleva gran parte de la carga de trabajo, pero es sólo una vez, o bien, es una tarea de baja frecuencia; que es casi siempre la más propensa a errores, ya que quien los consume llevará a cabo las tareas de alta frecuencia sobre un ambiente más controlado en el entendido de que, por diseño, se quedan casos o reglas de negocio sin atender, esto es, si las reglas de negocio no fueran compatibles con las plantillas seria el equivalente a querer martillar con un serrucho en vez de con un martillo. Se concluye que siempre hay que buscar en la medida de lo posible aplicar la herramienta más conveniente al problema. Por lo pronto el problema 'semilla' de este proyecto fue atendido debidamente.

Gracias a todo esto, en particular al uso intensivo de plantillas, se obtuvo una reducción importante de costos de mantenimiento, en términos cualitativos relativos a la magnitud del proyecto; y se ganó mucha flexibilidad ante los cambios, además de llevar el valor agregado de permitir el acceso a ciertas funcionalidades a través de dispositivos móviles tales como celulares y computadoras de tableta con navegador Web con soporte de HTML5 con Javascript.

Por otra parte, aunque las plantillas restringen la expresividad de quien hace uso de ellas para hacer composiciones, dicha limitación no es tan grande como para hacer inútiles sus productos o tan dura

como para que no se puedan adaptar de manera conveniente; en otras palabras, su cobertura es amplia, pero no infinita.

Además, la documentación en cada una de las etapas formó una parte muy importante en el seguimiento de todas sus actividades, pues al tratarse de una gran variedad de ellas, siempre es arriesgado no tener presentes todos sus detalles por lo que se recurrió incluso al apoyo de notas oportunas o ingeniosas referencias salpicadas tanto en los comentarios como en los nombres de las variables en el código fuente.

Finalmente, se observa que hay un costo ineludible tal que al simplificar un proceso hay un compromiso con complicar otros relacionados. La ventaja se gana cuando los procesos, que por necesidad se complican, quedan de alguna manera confinados en un ambiente controlado tal que son transparentes al usuario del proceso simplificado, esto es, permanecen ahí, pero pareciera como si no estuvieran.

5.4 TRABAJO A FUTURO

Pese al buen sabor de boca que pudiera dejar el proyecto al llegar a la etapa de entrega, los ánimos quedan latentes, y los deseos de mejoras o adiciones se incrementan. Particularmente, por la naturaleza de este trabajo y las distintas plataformas tanto de hardware como de software en que se puede emplear se hace necesario efectuar más pruebas en aquellas que no se exploraron a profundidad. Por otra parte, para añadir más valor a lo obtenido, sería conveniente atender varios detalles que se dejaron de lado por no satisfacer directamente los objetivos propuestos o que se omitieron por falta de tiempo.

Un ejemplo de lo anterior seria contar con capacidad de personalización de la interfaz de usuario para empatar su apariencia con su experiencia. Concretamente, si la experiencia del usuario es poca, se podrían presentar más asistentes para guiarle paso a paso en la introducción de datos para las operaciones.

Por otra parte, se dieron situaciones en las que se implementaron algunas ideas 'en sucio', esto es, sin aspirar a una implementación óptima, y se dejaron así porque cumplían con los requisitos impuestos. No obstante, siempre cabe la posibilidad de que haya mejores maneras de hacer las cosas. Por ejemplo, se podría dotar de la capacidad para crear plantillas por medio de un asistente para agilizar su producción. También se podría considerar con más atención en la eficiencia el manejo de cantidades muy grandes de operaciones o de operandos. Tal vez incluyendo un componente de paginación para poder distribuir listas grandes y evitar tener muchos elementos cargados en una sola página.

Casos aislados pero de interés los encontramos en situaciones como la que se presenta a través del script que se muestra continuación.

Listado 15: Script donde se efectúan operaciones inversas.

```
IMAGEN02 = cargarImagen( "C:/_/MyWebSites/CALIMAN/
    imagenesDePrueba/cameraman.png" )
IMAGEN03 = cargarImagen( "C:/_/MyWebSites/CALIMAN/
    imagenesDePrueba/caballo.png" )
IMAGEN01 = dividirPuntualImagenes( IMAGEN02, IMAGEN03 )
IMAGEN05 = multiplicarPuntualImagenes( IMAGEN01, IMAGEN03 )
```

Lo que hace tal script es cargar[8] la imagen del camarógrafo en el nicho 2, cargar la imagen del caballo en el nicho 3, dividir la imagen del nicho 2 entre la imagen del nicho 3 y poner el resultado en el nicho 1, multiplicar la imagen del nicho 1 por la imagen del nicho 3 y poner el resultado en el nicho 5; esto se representa simbólicamente como: $I5 = (I2/I3) * I3$. Se tiene que haciendo la manipulación algebráica aparte, se espera que $I5 = I2$, no obstante, por el rango dinámico de las imágenes se observa que hay posiciones en que la división se indefine, por lo que para evitar sobreflujos o indefiniciones, en esta implementación particular, se optó por emplear el valor numérico 1.0 como denominador al detectar posiciones con esas características, de manera tal que en los resultados se aprecia aún la silueta del caballito. Esto se podría corregir forzando a tratar las indefiniciones de alguna otra forma, por ejemplo empleando un número pequeño en vez de cero, pero aún así, no hay una definición canónica de la operación, por lo que obtener cierto resultado podría ser intencional y dependería de los motivos del implementador de dicha operación.

Consideraciones especiales con situaciones como la recién mostrada proporcionan material para proyectos futuros.

Otros esfuerzos de mayor envergadura se pueden enfocar en construir operaciones que aprovechen el poder de cómputo ofrecido por las Graphics Processing Unit o Unidad de Procesamiento Gráfico (GPU) o en construir una biblioteca con scripts reutilizables, por ejemplo cambiando en cada ocasión los nombres de los nichos o las variables involucradas pero manteniendo el comportamiento especificado. También se podrían establecer conexiones para ejecutar desde la interfaz de la calculadora algunos scripts en programas especializados como MATLAB, Octave o SciLab, o bien implementar algunas operaciones que hagan uso de bibliotecas maduras como OpenCV o que produzcan gráficas con GNUPlot.

Finalmente, con la potencia latente de ejecución de manera remota y distribuida queda abierta la posibilidad de implementar en pleno una versión hecha con los debidos cuidados y atenciones correspondientes como pudieran ser la capacidad multiusuario, multisesión y demás consideraciones derivadas de una presencia en red, como uso de ancho de banda y balanceo de cargas en el servidor, entre otras muchas.

8 Los números de los nichos empleados no son relevantes para el ejemplo.

Parte III

APÉNDICES

A

TABLAS Y REFERENCIAS MISCELÁNEAS

A.1 CADENAS DE IDENTIFICACIÓN DE CLIENTES Y SERVIDOR

- Cadenas de identificación del servidor web.

  `SERVER_SOFTWARE=Apache/2.2.22 (Win32) PHP/5.2.17 GATEWAY_INTERFACE=CGI/1.1 SERVER_PROTOCOL=HTTP/1.1`

- Cadena de identificación del navegador web Google Chrome
  `HTTP_USER_AGENT=Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.52 Safari/536.5`

- Cadena de identificación del navegador web del Nook Color
  `HTTP_USER_AGENT=Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_7; en-us) AppleWebKit/530.17 (KHTML, like Gecko) Version/4.0 Safari/530.17`

- Cadena de identificación del navegador web Safari del iPod Touch `HTTP_USER_AGENT=Mozilla/5.0 (iPod; CPU iPhone OS 5_1_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9B206 Safari/7534.48.3`

A.2 CLASES DE CARACTERES EN LOS URLS

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d
e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
y	z																		

Cuadro 1: Caracteres alfanuméricos válidos en URLs

$	-	_	.	+	!	*	’	(	)	,

Cuadro 2: Caracteres especiales válidos en URLs

;	/	?	:	@	=	&

Cuadro 3: Caracteres reservados con significado especial en URLs

B

REFERENCIA PARA EL DESARROLLADOR

En este apartado se ilustra con un ejemplo como añadir una nueva operación al catálogo ofrecido por CALIMAN. Posteriormente se presenta la distribución de archivos y directorios en una instalación típica y una detallada descripción de alto nivel del código fuente.

B.1 COMO AÑADIR UNA OPERACIÓN A CALIMAN

Dado el objetivo de añadir una operación, se muestran dos maneras de alcanzarlo, a saber:

- usando el asistente de autoría de operaciones,
- indicando que modificaciones hacer a las estructuras de datos y que archivos generar.

En ambos métodos, la información mínima requerida para añadir una operación es, a saber:

- Un nombre, una descripción y un nombre breve,
- elegir una plantilla,
- proporcionar etiquetas de apoyo al usuario final en la caja de diálogo,
- configurar un botón y asignarle un lugar en alguna botonera existente,
- especificar una ruta para almacenar la plantilla de código en lenguaje C y rellenar dicha plantilla con el código en lenguaje C de la operación misma.

En base a dicha lista de datos, el asistente de autoría de operaciones efectúa automáticamente algunas tareas que incluyen cambios en las estructuras internas de datos y en ciertos archivos además de la generación de una plantilla de código fuente en lenguaje C.

Debido a que tiene un efecto visible pese a su relativa simplicidad, el ejemplo elegido es el de la operación de negación de una imagen, esto es, aquella en donde se cambia de signo la intensidad de todos los pixeles de la imagen.

B.1.1 *Usando el asistente de autoría de operaciones*

El asistente de autoría de operaciones es una secuencia ordenada de formularios que solicitan la información mínima requerida para construir la interfaz gráfica de usuario y la plantilla de código en lenguaje C de una operación. La interfaz gráfica de usuario de una operación consiste en varios componentes, a saber: una caja de diálogo para introducir los valores a los parámetros de la operación y un botón ubicado en una botonera tales que pulsar el botón conduce a mostrar la caja de diálogo, misma que al tener los valores solicitados genera en el fondo la línea de comando del script con la que se activa el programa propio de la operación; la plantilla de código en lenguaje C incluye las funciones para abrir y guardar archivos, asignar y liberar recursos de memoria y otras auxiliares para leer archivos de configuración de manera tal que sólo reste añadir el código correspondiente a la operación. El asistente de autoría de operaciones evita a la mayoría de los desarrolladores el tener que familiarizarse con las estructuras internas con las que se representa la interfaz gráfica de usuario de la operación y le conduce cómodamente a rellenar correctamente sus plantillas con la finalidad de facilitar la ampliación del catálogo de operaciones.

B.1.1.1 *Partes del asistente de autoría de operaciones*

El asistente consiste en tres columnas, a saber:

- La primera columna muestra la lista de pasos efectuados, los pendientes e indica el paso actual.
- La segunda columna muestra indicaciones a seguir, vistas previas de los componentes de la interfaz gráfica de usuario de la operación y controles o campos para captura de los datos solicitados en cada paso.
- La tercera columna muestra la lista de valores asignados a las variables implicadas en la autoría de operaciones.

B.1.1.2 *Acceso al asistente de autoría de operaciones*

Para acceder al asistente:

- Desde el menú principal seleccionar `Herramientas`: *Se despliega el menú de herramientas.*
- En el menú de herramientas seleccionar `configurar caliman`. *Se despliega el menú de configuración.*
- En el menú de configuración seleccionar `añadir operación`: *Se despliega la primera página del asistente para añadir operaciones.*

B.1.1.3 *Introducción de datos al asistente de autoría de operaciones*

- En la primera página del asistente, pulsar el botón etiquetado `Siguiente`: *Se despliega la segunda página del asistente para añadir operaciones.*

  Esta primera página es de bienvenida para establecer un ambiente amistoso con el usuario.

- En la segunda página del asistente, introducir el nombre de la operación y pulsar el botón etiquetado `Siguiente`: *Se despliega la tercera página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

  El nombre de la operación aparece en las barras de títulos, puede contener espacios y signos de puntuación pero se recomienda que sólo consista de caracteres alfanuméricos.

- En la tercera página del asistente, introducir una descripción breve de la operación y pulsar el botón etiquetado `Siguiente`: *Se despliega la cuarta página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

  La descripción de la operación ayuda a elegir la operación, no hay restricciones en cuanto a su contenido, pero se recomienda brevedad.

- En la cuarta página del asistente, introducir un nombre breve de la operación y pulsar el botón etiquetado `Siguiente`: *Se despliega la quinta página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

  El nombre breve de la operación es muy importante pues es el que se emplea en los scripts y corresponde al nombre (sin extensión) del ejecutable de la operación y su archivo de código en lenguaje C, obligatoriamente debe consistir únicamente de caracteres alfanuméricos, sin espacios ni signos de puntuación.

- En la quinta página del asistente, elegir una plantilla acorde a los requisitos[1] de la operación: *Se despliega una vista previa en crudo de la plantilla.* Pulsar el botón etiquetado `Siguiente`: *Se despliega la sexta página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

1 Se provee un amplio catálogo de plantillas para atender las operaciones típicas de procesamiento de imágenes. La capacidad para crear plantillas por medio de un asistente queda fuera de los objetivos de este trabajo aunque esta planeado como trabajo a futuro. No obstante, tomar a través de su lectura, la experiencia vertida en las estructuras de datos que dan soporte a las plantillas incluidas puede facilitar crear nuevas en lo que se habilita un asistente.

La plantilla de la operación determina la calidad[2] y cantidad de las entradas y salidas de la operación así como el correspondiente código en lenguaje C.

- En la sexta página del asistente, tomar[3] nota de la línea de comandos de script para la operación generada en base a la plantilla seleccionada y pulsar el botón etiquetado `Siguiente`: *Se despliega la séptima página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

  La línea de comandos de script contiene la especificación explícita del orden en que se debe escribir en el script el comando de la operación así como los tipos de datos y su calidad de entrada o salida por medio de prefijos, a saber: el prefijo `ecto` indica una salida o producto de la operación, mientras que el prefijo `endo` indica una entrada o insumo de la operación.

- En la séptima página del asistente, hay que proporcionar etiquetas de apoyo al usuario final en la caja de diálogo y pulsar el botón etiquetado `Siguiente`: *Se despliega la octava página del asistente para añadir operaciones y se actualiza la lista de valores asignados.* Para modificar una etiqueta hay que seleccionar una de las marcas denominadas `<Editar aquí>`, posicionar el cursor del ratón sobre ella y presionar el botón primario, posteriormente escribir el texto deseado en el campo inmediato inferior y repetir las veces que sea necesario para el resto de las marcas.

  Las etiquetas de apoyo indican al usuario final el propósito de los distintos controles de la caja de diálogo.

- En la octava página del asistente, hay que proporcionar una etiqueta al botón de la operación en la calculadora, asignarle una posición y tamaño convenientes y pulsar el botón etiquetado `Siguiente`: *Se despliega la novena página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

- En la novena página del asistente, hay que seleccionar una botonera y una de sus posiciones disponibles para ubicar ahí el botón de la operación y pulsar el botón etiquetado `Siguiente`: *Se despliega la décima página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

- En la décima página del asistente, hay que proporcionar una ruta de directorios para almacenar y pulsar el botón etiquetado `Siguiente`: *Se despliega la undécima página del asistente para añadir operaciones y se actualiza la lista de valores asignados.*

2 Los tipos de datos involucrados y su calidad de entrada o salida respecto al programa individual de la operación.

3 De ser necesario, por ejemplo, para documentación externa.

El directorio debe estar presente y el usuario asignado al servidor web debe tener acceso de lectura-escritura, no se realizan verificaciones para saber si existe o no, y en caso de no existir no se creará, lo que podría producir resultados inesperados.

- En la undécima página del asistente, hay que proporcionar el código fuente de la operación o pseudocódigo en comentarios y pulsar el botón etiquetado `Siguiente`: *Se despliega la duodécima página del asistente para añadir operaciones, se graban las modificaciones efectuadas y se genera el archivo de plantilla de código en lenguaje C.*

  No hay restricciones ni revisiones respecto a lo que se puede escribir en el campo de texto, incluso pudiera dejarse en blanco. No obstante, se espera que sea código válido en lenguaje C, incluyendo comentarios. Lo que se escriba aquí aparecerá dentro de la función `operacionConImagenes` en el archivo plantilla[4] de código fuente generado de manera que si se va a realizar una edición posterior en dicho archivo, este puede ser el punto de inicio.

- En la duodécima página del asistente, hay que confirmar las acciones realizadas y pulsar el botón etiquetado `Terminar`: *Se recarga la página web para actualizar los valores en memoria.*

B.1.1.4 *Obtención y colocación del ejecutable de la operación*

Entre los archivos de código generados en el asistente se encuentra un proyecto del ambiente de programación Qt Creator, hay que abrirlo, asegurarse que el toolchain para el proyecto sea MinGW y compilar. Finalmente el programa producido debe ser copiado al directorio donde se encuentra el resto de los programas de operaciones de CALIMAN, por default: `/CALIMAN/WebApps/`.

Después de haber observado todas estas indicaciones, ya se tiene una nueva operación en el catálogo.

B.1.2 *Modificando las estructuras de datos internas*

IMPORTANTE: En todo momento se recomienda usar el asistente de autoría de operaciones salvo que se desee hacer alguna personalización a la interfaz de usuario o al código en lenguaje C que no esté contemplada en las plantillas.

En esta sección se describe con detalle y paso a paso como añadir una operación sin usar dicho asistente. Para esta actividad es requisito tener familiaridad con la programación en lenguaje C y en lenguaje

4 Se considera plantilla en el sentido de que sólo queda pendiente añadir el código fuente propio de la operación

Javascript, en particular con la notación JSON, así como con la manipulación de nodos DOM, la modificación de estructuras en HTML y la aplicación de estilos con CSS.

Queda hecha la advertencia de que es un proceso propenso a errores por lo que se debe tener mucho cuidado además de mantener copias de respaldo de los archivos que se van a modificar.

Al decidirse por implementar una operación por este método, lo primero que hay que tener definido es un nombre y su equivalente en nombre corto, tomando en cuenta que este último es el que se emplea en los scripts y debe corresponder al nombre (sin la extensión) del ejecutable de la operación y su archivo de código en lenguaje C, además de que obligatoriamente debe consistir únicamente de caracteres alfanuméricos, sin espacios ni signos de puntuación.

Para este ejemplo se supone que se va a añadir la operación de obtener una imagen como resultado de sumar dos imágenes.

El nombre que se le asigna es *Suma de imágenes*, con `sumarImagenes` como nombre corto.

AÑADIR UNA ENTRADA EN LA LISTA DE COMANDOS DE OPERACIÓN ACEPTADOS. Esto es añadir un renglón de valores separados por un símbolo[5] ¬ al archivo `CALIMAN_ComandosScript.txt`, donde el primer valor corresponde al símbolo (nombre breve) que identifique a la operación y el segundo valor a la forma como debe aparecer el comando en el script, esto es, su sintaxis.

Por ejemplo:

Listado 16: Ejemplo de entrada en el archivo de comandos de operación aceptados.

```
sumarImagenes¬ectoArchivoResultado1 = sumarImagenes (
    endoArchivoOperando1 , endoArchivoOperando2 )
```

En el listado el nombre breve de la operación es `sumarImagenes` y la sintaxis de la operación consiste en:

- una palabra marcada con el prefijo `ecto` que indica que ahí se especifica el resultado de la operación seguida de la palabra `ArchivoResultado1` que indica como se especifica este parámetro en el archivo de configuración para la ejecución de la operación,
- un símbolo = que sirve como separador,
- el nombre breve de la operación, en este caso `sumarImagenes`,
- un símbolo (que sirve como separador,

5 Se emplea este símbolo ¬ por que no se espera que aparezca como parte de una línea válida del script.

- una palabra marcada con el prefijo `endo` que indica que ahí se especifica un operando de la operación seguida de la palabra `ArchivoOperando1` que indica como se especifica este parámetro en el archivo de configuración para la ejecución de la operación,
- un símbolo `,` que sirve como separador
- una palabra marcada con el prefijo `endo` que indica que ahí se especifica un operando de la operación seguida de la palabra `ArchivoOperando2` que indica como se especifica este parámetro en el archivo de configuración para la ejecución de la operación.

Habiendo especificado la sintaxis del comando, se espera que en el script, cuando aparezca ese comando, deben aparecer los mismos separadores en el mismo orden.

Siguiendo el ejemplo:

Listado 17: Ejemplo de línea de script.

```
IMAGEN03 = sumarImagenes ( IMAGEN01 , IMAGEN02 )
```

En el listado, ya sea que por medio de la caja de diálogo de la operación o porque el usuario tecleó el comando en el campo de texto de la ventana del script, el comando especifica que el resultado aparecerá en el nicho denominado `IMAGEN03`, que los operandos se toman de los nichos denominados `IMAGEN01` e `IMAGEN02` y que la operación es `sumarImagenes`.

Lo que hace el programa despachador es detectar que se quiere ejecutar tal comando, luego tomarlo de la lista de operaciones aceptadas y detectar sus separadores junto con su orden para proceder a armar el archivo de configuración quitando los prefijos `ecto` y `endo` de las palabras que definen los parámetros de la operación, y usándolas como llaves en los pares llave-valor. Los valores en dichos pares se toman de las posiciones correspondientes en la línea respectiva del script.

AÑADIR UNA ENTRADA EN LA LISTA DE OPERACIONES EN LAS BOTONERAS. En el archivo `index.html` localizar la definición del arreglo `arrBotonera` que es donde se determinan las posiciones de los botones en las botoneras y las cajas de diálogo de las operaciones, añadir una entrada y llenarla como se indica.

Listado 18: Estructura de datos Javascript para la interfaz gráfica de usuario de la operación de sumar imágenes.

```
//arrBotonera : es un arreglo de arreglos
//arrBotonera[i] : es un arreglo
//arrBotonera[i][0] es una cadena de texto, es el título de la
    botonera
//arrBotonera[i][1] es un número entero, es la cantidad de
    renglones en la botonera
```

```
//arrBotonera[i][2] es un número entero, es la cantidad de
    columnas en la botonera
//arrBotonera[i][3] es un arreglo de arreglos
//arrBotonera[i][3][j] es un arreglo

// a partir de aquí empieza el ejemplo que se muestra abajo:
//arrBotonera[i][3][j][0] es un número entero, es el renglón
    donde está el botón
//arrBotonera[i][3][j][1] es un número entero, es la columna
    donde está el botón
//arrBotonera[i][3][j][2] es un arreglo
//2 arrBotonera[i][3][j][2][0] es un número entero, es el offset
    vertical de la etiqueta del botón
//3 arrBotonera[i][3][j][2][1] es un número entero, es el offset
    horizontal de la etiqueta del botón
//4 arrBotonera[i][3][j][2][2] es un número de punto flotante, es
     el tamaño de letra del botón en unidades em
//5 arrBotonera[i][3][j][2][3] es una cadena de texto, es la
    etiqueta del botón
//6 arrBotonera[i][3][j][2][4] es una cadena de texto, es el
    título de la caja de diálogo
//7 arrBotonera[i][3][j][2][5] es una cadena de texto, es el
    contenido de la caja de diálogo
//8 arrBotonera[i][3][j][2][6] es una cadena de texto, es la
    lista de campos de donde se tomarán valores de la caja de
    diálogo
// arrBotonera[i][3][j][2][7] es una cadena de texto, es la lista
     de condiciones que desactivan el boton procesar de la caja
    de diálogo
// arrBotonera[i][3][j][2][8] es una cadena de texto, es la
    descripción de la operación para los usuarios finales

[
 0,
 1,
 [
  30,
  19,
  2.4,
  '+',
  'Sumar imágenes',
  '<div class="CALIMAN_cajaDeDialogo_contenedor">
    <fieldset>
     <legend>Sumar imágenes</legend>

     <label for="lbImagenDestino1"></label>
     <select id="lbImagenDestino1" title="Elija un nicho de
         imagen para almacenar el resultado de la operación">
         ____EJECUCION____ </select>

     <span> = sumarImagenes( </span>
```

```
    <label for="lbImagenOrigen1"></label>
    <select id="lbImagenOrigen1" title="Elija un nicho de imagen
        como operando"> ____EJECUCION____ </select>

    <span> , </span>

    <label for="lbImagenOrigen2"></label>
    <select id="lbImagenOrigen2" title="Elija un nicho de imagen
        como operando"> ____EJECUCION____ </select>

    <span> ) </span>

    <div style="text-align:center; padding: 10px;">

     <input type="button" id="btProcesar" value="Procesar"
        onclick="accionBotonProcesarOperacionYMostrarBotones (
        /*iPestagna*/ ____EJECUCION____ , /*
        idNodoDOMcontenedor*/ \'____EJECUCION____\' , /*
        idNodoDOMasa*/ \'____EJECUCION____\' , /*
        strParameterizedCommand*/ \'____PARAM____ =
        sumarImagenes( ____PARAM____, ____PARAM____ )\' ,
        /*arrStrIDsOrigenes*/ [\'lbImagenDestino1\',\'
        lbImagenOrigen1\',\'lbImagenOrigen2\'] , /*
        strReplacementSignal*/ \'____PARAM____\' );return
        false;">

     <input type="button" id="btCancelar" value="Cancelar"
        onclick="accionBotonCancelar ( /*iPestagna*/
        ____EJECUCION____ , /*idNodoDOMcontenedor*/ \'
        ____EJECUCION____\' , /*idNodoDOMasa*/ \'
        ____EJECUCION____\' );return false;">

    </div>
   </fieldset>
  </div>',
  'strHTMLNichoDestino,strHTMLNichoOrigen,strHTMLNichoOrigen,
      iPestagna,idNodoDOMcontenedor,idNodoDOMasa,iPestagna,
      idNodoDOMcontenedor,idNodoDOMasa',
  '(lbImagenOrigen1.value.length <= 0 || lbImagenOrigen2.value.
      length <= 0)',
  'para obtener una imagen como resultado de sumar dos imágenes'
 ]
],
```

LLENAR LA PLANTILLA DE PLANTILLAS DE CÓDIGO FUENTE EN LENGUAJE C, COMPILAR Y COLOCAR EL EJECUTABLE EN EL DIRECTORIO DE PROGRAMAS DE OPERACIONES. La plantilla de plantillas por default está en:

`/CALIMAN/WebSites/plantillas/CALIMAN_PlantillaDePlantillasDeOperacion.cpp`

Hay 13 secciones relevantes, a saber:

- La sección de definición de la estructura para configuración del programa.
- La sección de reporte a consola de la definición de la estructura de configuración del programa.
- La sección de definición de los valores predeterminados de configuración del programa.
- La sección de asignación de los valores predeterminados de configuración a los miembros correspondientes de la estructura de configuración del programa.
- La sección de definición de las llaves a buscar en los archivos de configuración pasados al programa. Los nombres de las llaves deben de coincidir con los especificados en la lista de comandos aceptados (sin los prefijos `endo` y `ecto`).
- La sección de búsqueda de las llaves en el archivo de configuración pasado al programa y la asignación de los valores correspondientes a los miembros de la estructura de configuración del programa.
- La declaración de la función `operacionConImagenes`.
- La firma en la implementación de la función `operacionConImagenes`.
- La sección correspondiente al código fuente propio de la operación.
- La sección de solicitud de recursos de memoria y apertura de imágenes especificadas en la configuración.
- La sección de llamada a la función `operacionConImagenes` a la cual se le envían referencias a los recursos solicitados.
- La sección de escritura de las imágenes resultado de la operación.
- La sección de liberación de los recursos solicitados.

La marca `//_____PUNTODESUSTITUCIONPLANTILLA_____` indica la posición de esas secciones y es ahí donde debe escribirse el código correspondiente ya sea quitando o dejando comentada la marca.

Listado 19: Marcas de los puntos de sustitución en la plantilla de plantillas.

```
//INICIA - contenido dependiente de la aplicacion - INICIA

//_____PUNTODESUSTITUCIONPLANTILLA_____

//TERMINA - contenido dependiente de la aplicacion - TERMINA
```

En general, se recomienda tomar como ejemplo el código fuente de una operación ya implementada.

B.2 DISTRIBUCIÓN DE LOS ARCHIVOS

En esta sección se describe la estructura de directorios empleada así como la distribución en ellos de los archivos[6] de configuración y programas.

Debe haber dos directorios disjuntos[7], a saber: uno para el front-end, con los archivos accesibles desde web y otro para el back-end, con los programas de operaciones y auxiliares.

Los archivos en el directorio para el front-end deben ser accesibles en un navegador web por medio de solicitudes de URL a un servidor web, mientras que los archivos en el directorio para el back-end solamente son accesibles para ciertos usuarios del sistema de archivos, en particular el que emplea el servidor web en su ejecución.

- Hay en el directorio para el front-end:
  - el archivo de la página principal de CALIMAN, por default: `index.html`,
  - un directorio para plantillas de código fuente de operaciones en lenguaje C, por default: `plantillas`,
  - un directorio para almacenar las miniaturas de las imágenes abiertas, son las que se muestran en los nichos, por default: `miniaturas`,
  - un directorio para almacenar los archivos de los kernels de convolución, por default: `kernelsDeConvolucion`,
  - un directorio para la galería de imágenes de prueba, por default: `imagenesDePrueba`,
  - un directorio para las imágenes empleadas en la interfaz gráfica de usuario como logotipos, botones, flechas, por default: `imagenes`,
  - un directorio para los archivos de configuración de la interfaz gráfica de usuario, la pichonera, las botoneras, las cajas de diálogo de las operaciones y sus plantillas, la lista de comandos aceptados en el script, por default: `configuracion`
  - un directorio para el archivo del almacén de imágenes y variables, por default: `almacen`
  - opcionalmente el archivo del icono para la pestaña en el navegador web, obligatoriamente: `favicon.ico`,
- y en el directorio para el back-end:
  - un directorio para las bibliotecas de enlace dinámico de Qt para abrir distintos formatos de imágenes, obligatoriamente: `imageformats`,

6 Algunos archivos sólo están presentes en Windows, por ejemplo, los archivos de bibliotecas de enlace dinámico de QT con extensión .dll.

7 Esto es, que no sean uno subdirectorio del otro en ningún grado.

- un directorio para almacenar los archivos de configuración que se generan para pasar parámetros a los programas de las operaciones, por default: `runtimeConfigs`,
- el archivo de configuración de Qt que especifica la ubicación de las bibliotecas de enlace dinámico que se emplean, obligatoriamente: `qt.conf`,
- el archivo de biblioteca de enlace dinámico de las rutinas para Windows de la GNU Compiler Collection (GCC), obligatoriamente: `libgcc_s_dw2-1.dll`,
- el archivo de biblioteca de enlace dinámico de las rutinas de la Minimalist GNU for Windows (MinGW), obligatoriamente: `mingwm10.dll`,
- el archivo de biblioteca de enlace dinámico de las rutinas del núcleo de Qt, obligatoriamente: `QtCore4.dll`,
- el archivo de biblioteca de enlace dinámico de las rutinas de los componentes de interfaz gráfica de usuario de Qt, obligatoriamente: `QtGui4.dll`,
- el archivo de biblioteca de enlace dinámico de las rutinas de los componentes de red de Qt, obligatoriamente: `QtNetwork4.dll`,
- los programas de las operaciones, por default: su nombre correspondiente registrado en el archivo de comandos aceptados en el script,
- el programa encargado de desplegar las cajas de diálogo nativas de abrir y guardar archivos: `CALIMAN_DeOSAWeb.exe`,
- el programa encargado de pasar mensajes entre el cliente web y el programa que despliega la cajas de diálogo nativas, por default: `CALIMAN_DeWebAOS.exe`,
- el programa encargado de analizar el script, armar los archivos de configuración y pasarlos a los programas de las operaciones, administrar el almacén de valores y variables e indicar cuales imágenes están asignadas a cuales nichos, por default: `CALIMAN_despachadorDeScripts.exe`,
- el programa encargado de generar las miniaturas de las imágenes abiertas que serán exhibidas en los nichos, por default: `CALIMAN_GenerarImagenMiniatura.exe`,
- el programa encargado de grabar archivos de texto, empleado principalmente para archivos de configuración:

  `CALIMAN_grabarTexto.exe`,
- el programa encargado de cargar las imágenes desde su formato original al formato binario interno de CALIMAN, así como calcular algunas características numéricas de las

imágenes que se muestran acompañando a las imágenes en los nichos de la pichonera, por default: `cargarImagen.exe`,

- el programa encargado de grabar las imágenes en el formato especificado en la extensión del nombre asignado desde el formato binario interno de CALIMAN, por default: `guardarImagen.exe`,
- el programa encargado de grabar las imágenes en el formato especificado en la extensión del nombre asignado desde el formato binario interno de CALIMAN con una normalización previa para despliegue en pantalla, por default: `guardarConRangoDinamicoDePantalla.exe` .

B.3 DESCRIPCIÓN DE ALTO NIVEL DEL CÓDIGO FUENTE

El código fuente se divide en la parte de cliente y la parte del servidor. La parte del cliente está implementada en el lenguaje Javascript actuando sobre una estructura de documento HTML cuya apariencia es modificada mediante CSS. A su vez, la parte del servidor está implementada en el lenguaje C y actúa principalmente sobre el sistema de lanzamiento de procesos y el sistema de archivos.

B.3.1 *Parte del cliente*

El código fuente en el cliente se divide por el aspecto que atiende en:

- comportamiento con Javascript,
- apariencia con CSS,
- estructura con HTML,
- contenido mediante cadenas de texto e imágenes;

B.3.1.1 *Secciones del código fuente en Javascript*

De interés en este apartado es el comportamiento, por lo que dicha sección se divide a su vez por funcionalidad en:

- Funciones de validación de valores introducidos en campos de texto.
- Funciones de enlace de datos en respuesta a eventos `OnKeyUp`, `OnChange`.
- Funciones de soporte para ventanas o regiones arrastrables.
- Funciones de manejo del evento `OnScroll` para control de desplazamientos en la ventana del navegador web.

- Funciones de que usan el objeto `XMLHTTPRequest` para dar soporte a transferencias personalizadas con HTTP.
- Funciones de preparación de cadenas de texto para su transferencia.
- Funciones de manejo de cadenas de texto para su inclusión en el script de CALIMAN.
- La función de procesamiento de ida y vuelta del script de CALIMAN.
- La función que construye la siguiente línea del script de CALIMAN en base a los datos introducidos en las cajas de diálogo de las operaciones.
- La función que recibe una cadena con marcas y sustituye valores tomados de una lista en esas marcas.
- Datos globales para el cliente, a saber:
  - Rutas de programas y almacenes de datos y configuraciones.
  - Estructuras de datos de componentes de plantillas, plantillas, operaciones y botoneras.
- Funciones de procesamiento de eventos `OnLoad` y `OnBeforeUnload`.
- Funciones de alimentación de listas desplegables de opciones.
- Funciones para modificar visibilidades de componentes.
- Funciones para administración de la lista de componentes de plantillas.
- Funciones para administración de la lista de plantillas.
- Funciones para administración de la lista de operaciones.
- Funciones para administración de la lista de botoneras.
- Funciones para obtención de cadenas de números con ceros a la izquierda y cadenas alfanuméricas construidas al azar.
- La función que construye la pichonera de imágenes.
- Funciones que hacen aparecer o desaparecer imágenes en los nichos de la pichonera.
- Funciones de alimentación de listas desplegables de opciones con los nichos disponibles para las operaciones.
- Funciones para las acciones de salida de las cajas de diálogo.
- La función que construye las cajas de diálogo.

- La función que construye tablas desde arreglos rectangulares.
- Funciones que construyen las cadenas de etiquetas de las pestañas en los menús y botoneras.
- Funciones que construyen las botoneras y los botones.
- Funciones que construyen los menús, a saber:
  - Menú principal.
  - Menú de bienvenida.
  - Menú de herramientas.
  - Menú de ayuda.
  - Menú de archivo.
  - Menú de configuración.
- La función que construye la matriz de miniaturas de imágenes empleada en la caja de diálogo de 'Abrir imagen'.
- Funciones que construyen las cajas de diálogo accesorias, a saber:
  - Caja de diálogo de configurar pichoneras.
  - Caja de diálogo de configurar entorno.
  - Caja de diálogo de abrir imagen.
  - Caja de diálogo de guardar imagen.
  - Caja de diálogo de análisis de imagen.
  - Caja de diálogo para edición de funciones de transferencia de tonos.
  - Caja de diálogo de lista de variables.
  - Caja de diálogo para edición de matrices de kernel de convolución.
  - Caja de diálogo para edición de paletas de colores.
- Funciones de almacenamiento de datos de la configuración de pichoneras o de entorno.
- Funciones de validación de cadenas de texto que designan nombres o rutas de archivos de imágenes.
- Funciones de verificación de estado del servidor de ventanas nativas o tipo de ejecución (remota o local).
- Funciones de lanzamiento de las cajas de diálogo nativas para seleccionar archivos para abrir o guardar.
- Funciones de cálculo de histogramas para las imágenes en los nichos.

- Funciones de graficación de histogramas y cortes vertical/horizontal por canal de las imágenes en los nichos.
- Funciones de graficación de funciones de transferencia de tonos.
- Funciones de manipulación de nodos de funciones de transferencia de tonos.
- Funciones de manipulación de la lista de variables.
- Funciones de manipulación de entradas de matriz de kernel de convolución.
- Funciones de manipulación de listas de colores.
- Funciones del asistente de autoría de operaciones.

IMPORTANTE: Mediante `use strict` se previenen situaciones que pudieran causar problemas posteriormente como variables no inicializadas o no declaradas a la vez que se arrojan más excepciones. Mediante la herramienta JSLint se detectan irregularidades como parámetros no usados, empleo ambiguo de == o !=.

B.3.1.2 *Secciones del código fuente en HTML*

De interés en este apartado es la estructura, por lo que dicha sección se divide a su vez en:

- Región para pichoneras de imágenes.
- Región para script de CALIMAN.
- Región para botoneras y menús.
- Región para mensajes de depuración (presente pero no visible en la versión final).
- Región para el asistente de autoría de operaciones.
- Región para el menú principal.

B.3.2 *Parte del servidor*

El código fuente en el servidor se divide principalmente en:

- programa despachador/parser de script de CALIMAN,
- programas de operaciones,
- programa que lee imágenes,
- programa que graba imágenes,
- programa que genera imágenes en miniatura,

- programa grabador de archivos de texto,
- cliente de ventanas nativas,
- servidor de ventanas nativas;

B.3.2.1 *Secciones del código fuente del programa despachador/parser*

- funciones para lectura de archivos de configuración,
- funciones para manejo de cadenas de texto,
- funciones para detección y clasificación de tokens en las líneas que especifican la sintaxis de las operaciones,
- funciones para detección y clasificación de tokens en las líneas del script,
- funciones para administración del contenido del almacen de imágenes y valores,
- funciones para generación de los archivos de configuración que se pasan a los programas,
- funciones para llamada a los programas de las operaciones y de miniaturización,
- funciones de reporte de estado a bitácora,
- función de asignación y especificación de acciones de entrada en los archivos de configuración,
- función de asignación y especificación de acciones de salida en los archivos de configuración,
- función de pre-procesamiento del script recibido via web,
- funciones de control de flujo de ejecución;

B.3.2.2 *Secciones del código fuente de los programas de operaciones, lectura/escritura de imágenes y generación de miniaturas*

- funciones para lectura de archivos de configuración,
- funciones para lectura/escritura de archivos de datos con formato de matriz almacenda en binario,
- función que efectúa la operación,
- funciones de control de flujo de ejecución;

B.4 FORMATOS DE ARCHIVOS DE DATOS

B.4.1 *El formato de matriz almacenada como archivo binario estructurado*

Es el formato empleado para almacenar las imágenes de CALIMAN en archivos binarios estructurados.

Consiste en un encabezado de tres números enteros seguido de una secuencia de números de un tipo de datos específico. Dicho tipo de datos está declarado en las rutinas de lectura y escritura de este tipo de archivos en el código fuente de cada programa de operación con imágenes. La cantidad de números en dicha secuencia es el producto de los tres números del encabezado. Los números del encabezado especifican en el siguiente orden: la cantidad de renglones, la cantidad de columnas y la cantidad de capas o canales de color. Aunque los números de la secuencia pueden ser de cualquier tipo de datos, se eligió float por su tamaño estándar en arquitecturas de CPU de 32 y 64 bits.

Es muy importante notar que dichas rutinas de lectura y escritura están restringidas a un tipo de datos específico y se asume en ellas que tanto el encabezado como la secuencia de números están almacenados en posiciones contiguas en memoria. Esto significa que no son compatibles los archivos con secuencias de tipos de datos distintos en el sentido de que a priori no está definido lo que sucede al leer un archivo de un tipo con una rutina escrita para leer archivos de tipo distinto.

B.4.2 *El formato de matriz almacenada como texto ASCII*

Es el formato empleado para almacenar las imágenes de CALIMAN en archivos de texto ASCII.

El primer renglón actúa como encabezado con una lista de valores separados por espacios. El primer valor indica el número de renglones de la matriz, el segundo valor indica el número de columnas de la matriz.

Los siguientes renglones corresponden a las entradas de la matriz como listas de valores separados por espacios.

Ejemplo: Para almacenar una matriz de 3 renglones por 6 columnas. El primer renglón debe tener: `3 6` Los siguientes 3 renglones deben tener 6 columnas cada uno, separadas por espacios:

Listado 20: Ejemplo de datos numéricos en un arreglo de 3 renglones y 6 columnas.

```
1 2 3 4 5 6
6 5 4 3 2 1
1 3 5 2 4 6
```

Entonces el archivo completo queda así:

Listado 21: Ejemplo de archivo de datos numéricos con encabezado.

```
3 6
1 2 3 4 5 6
6 5 4 3 2 1
1 3 5 2 4 6
```

Importante: El primer caracter del primer renglón puede ser el símbolo #. En algunos contextos la presencia del símbolo # como primer caracter de algún renglón provoca que el resto de dicho renglón se considere como que no aporta datos útiles o que se trata de algún comentario. En nuestro caso esto es de provecho cuando la matriz almacena los datos de las coordenadas de los puntos para graficar con el programa GNUPlot en donde es de interés no incluir entre los datos el tamaño de la matriz.

B.4.3 *El formato de matriz almacenada en binario*

Este tipo de archivos es similar al de matriz almacenada como texto ASCII, con la particularidad de que los números se almacenan directamente en su representación binaria y no se puede usar este formato para graficar con GNUPlot directamente.

Hay un encabezado formado por las dos primeras palabras que determinan las dimensiones de la matriz.

B.4.4 *El formato de diccionario de pares llave-valor*

Cada renglón tiene un par de cadenas llave-valor separadas por algún símbolo que no se espera forme parte de dichas cadenas. Ejemplo de separadores típicos son los caracteres ¬ o =.

BIBLIOGRAFÍA

Inc. Adobe Systems. Photoshop express editor. http://www.photoshop.com/tools/expresseditor/, 2012.

Sethi y Ullman Aho, Lam. *Compilers: Principles, Techniques & Tools*. Addison Wesley, segunda edition, 2006.

Albert S. Woodhull Andrew S. Tanenbaum. *Sistemas operativos: Diseño e implementación*. Prentice Hall, segunda edition, 1997.

Maarten Van Steen Andrew S. Tanenbaum. *Distributed Systems: Principles and Paradigms*. Prentice Hall, segunda edition, 2007.

T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738 (Proposed Standard), December 1994. URL http://www.ietf.org/rfc/rfc1738.txt. Obsoleted by RFCs 4248, 4266, updated by RFCs 1808, 2368, 2396, 3986, 6196, 6270.

J Bosch. *Design and Use of Software Architectures*. Addison-Wesley, ACM Press, 2000.

DaNae Dayleyr Brad Dayley. *Adobe Photoshop CS6 Bible*. Wiley, 2012.

Dries Buytaert. Drupal. http://drupal.org/, 2012.

Eric J Byrne. Software reverse engineering: a case study. *Software: Practice and Experience*, 21(12):1349–1364, 1991.

www Centre National de la Récherce Scientifique. Introduction to the cimg library : C++ template image processing toolbox (version 1.5). http://cimg.sourceforge.net/CImg_slides.pdf, 2012.

Elliot J. Chikofsky and James H. Cross II. Reverse engineering and design recovery: A taxonomy. *IEEE Softw.*, 7(1):13–17, January 1990. ISSN 0740-7459. doi: 10.1109/52.43044. URL http://dx.doi.org/10.1109/52.43044.

Song C. Choi and Walt Scacchi. Extracting and restructuring the design of large systems. *IEEE Softw.*, 7(1):66–71, January 1990. ISSN 0740-7459. doi: 10.1109/52.43051. URL http://dx.doi.org/10.1109/52.43051.

L. Clements, P. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

CMLA, ENS Cachan DMI, Universitat de les Illes Balears Fing, and Universidad de la República. Image processing on line. http://www.ipol.im/, 2012.

Miguel de Cervantes Saavedra. *Historia de los Trabajos de Persiles y Sigismunda*. Centro de Estudios Cervantinos, Universidad de Alcala., 1616.

ECMA-ST. Ecmascript language specification. Technical report, Ecma International, 2011. URL http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. URL http://www.ietf.org/rfc/rfc2616.txt. Updated by RFCs 2817, 5785, 6266, 6585.

García, Pérez, Ruiz, Segarra, Sempere, and de Parga. *Teoría de Autómatas y Lenguajes Formales*. Alfaomega Grupo Editor, 2001.

Adrian Kaehler Gary Bradski. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

www GraphicsMagick Group. Graphicsmagick supported formats. http://www.graphicsmagick.org/formats.html, 2012.

ISO/IEC. Iso/iec 9899:201x committee draft programming languages : C. nov 16, 2010. Technical report, ISO/IEC, 2010. URL http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1539.pdf.

Bernd Jahne. *Digital Image Processing: Concepts, Algorithms and Scientific Applications*. Springer-Verlag, cuarta edition, 2008.

Robert Shimonski Jason van Gumster. *GIMP Bible*. Wiley, 2010.

Dean Kelley. *Teoría de Autómatas y Lenguajes Formales*. Prentice Hall, 1995.

H. W. Lie and B. Bos. Cascading Style Sheets, level 1. 17 December 1996. Revised 11 January 1999. RFC 5, jun 1999. URL http://www.w3.org/TR/1999/REC-CSS1-19990111.

Paul C. Clements Linda M. Northrop. A framework for software product line practice, version 5.0. Technical report, Software Engineering Institute, Carnegie Mellon University, 2012. URL http://www.sei.cmu.edu/productlines/frame_report/arch_def.htm.

www Microsoft Patterns & Practices Group. *Microsoft .NET Application Architecture Guide*. Microsoft Press, segunda edition, 2009. URL http://msdn.microsoft.com/en-us/library/ff650706.aspx.

Sebastian Montabone. *Beginning Digital Image Processing: Using Free Tools for Photographers*. Apress, 2010.

Katherine Murray. *Microsoft Office 365: Connect and Collaborate Virtually Anywhere, Anytime*. Microsoft Press, 2011.

Inc. Open Source Matters. Joomla. http://www.joomla.org/, 2012.

David Polberger. Component technology in an embedded system. Master's thesis, Departamento de Ciencias de la Computación, Universidad de Lund, 2009. URL http://www.polberger.se/components/.

Roger S. Pressman. *Ingeniería del software: Un enfoque práctico.* McGraw-Hill, quinta edition, 2002.

Steven L. Eddins. Rafael C. Gonzalez, Richard E. Woods. *Digital Image Processing Using MATLAB*. Gatesmark Publishing, segunda edition, 2009.

D. Robinson and K. Coar. The Common Gateway Interface (CGI) Version 1.1. RFC 3875 (Informational), October 2004. URL http://www.ietf.org/rfc/rfc3875.txt.

www S2i-Industrial Intelligent Systems. Harpia user manual version 0.1. http://s2i.das.ufsc.br/harpia/downloads/manual_harpia.pdf, 2007.

Ian Sommerville. *Software Engineering*. Addison-Wesley, Boston, MA, USA, 4th edition, 1992.

Nancy Holzner Steven Holzner. *Google Docs 4 Everyone*. Que, 2009.

Inc. The Wikimedia Foundation. Wikipedia. http://www.wikipedia.org/, 2012.

Ludwig von Bertalanffy. *Teoría general de los sistemas: fundamentos, desarrollo, aplicaciones.* Fondo de Cultura Económica, segunda edition, 2006.

COLOFÓN

Este documento se mecanografió en su totalidad en el editor de texto WinShell y se virtió en un documento electrónico en formato PDF con MikTeX Portable empleando el estilo tipográfico `classicthesis` desarrollado por André Miede que está disponible tanto para LaTeX y LyX en:

`http://code.google.com/p/classicthesis/`

.

Final Version as of 2 de julio de 2013 (`classicthesis` version 0.201211141309).

DECLARACIÓN

Declaro que el trabajo presentado en esta tesis es, hasta donde tengo entendido en un esfuerzo a conciencia para asegurarlo, original, excepto en lo que toca a los documentos, herramientas, tecnologías y demás material relacionado al que se hace referencia en el texto, además de que no ha sido sometido antes ni total ni parcialmente, para obtener un grado ni en esta ni en ninguna otra institución.

Asimismo declaro haber cumplido con los requisitos, procedimientos, reglamentos y políticas de la institución en lo referente a la elaboración del presente trabajo de tesis y su defensa pública para obtener el grado.

Guanajuato, Guanajuato, México., Noviembre 2012

Alfonso Ceseña Quiñones