BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Página e Interface para un Sistema de Graficación de Funciones Útil en Aplicaciones de Mínimos Cuadrados no Lineales"

TESIS
Que para obtener el título de Licenciado en Ciencias de la Computación.

Presenta:

García López Lucio

Asesor:

Mtro. Soriano Ulloa Marco Antonio

Puebla, Puebla Febrero de 2008

Introducción General

Antecedentes del proyecto

El estudio de las Ciencias Exactas, siempre nos ha guiado a realizar operaciones sobre un conjunto de datos (evaluación de ecuaciones, representaciones gráficas, etc.), ejemplo de esto, la evaluación del Método de Mínimos Cuadrados. Muchas de estas operaciones son realizadas sobre papel o con la utilización de dispositivos como la calculadora, realizando cada operación de forma individual.
La evaluación de funciones matemáticas o conjuntos de datos de forma automática requiere de la integración de todas las operaciones evaluadas de forma individual para llevarlas a cabo de manera automática.
Con esto, las personas pueden olvidarse de la evaluación de operaciones de un mismo tipo de forma constante y sólo tener que proporcionar los datos de entrada para mirar los resultados de la operación.
Para llevar a cabo la evaluación de una función matemática o conjuntos de datos, existen actualmente aplicaciones gráficas que hacen tal tarea, pero con la característica de ser aplicaciones que operan sobre un solo equipo, es decir, sólo la persona encargada de utilizar tal equipo es quien puede hacer uso de tal aplicación (Mathcad, Winplot, etc), además de requerir de la compra de licencias para su utilización. En contraste, existen en internet aplicaciones gráficas similares a los mencionados anteriormente, estas aplicaciones pueden ser compartidos entre múltiples usuarios tal como se realiza con los recursos compartidos en red como son: procesadores, impresoras, archivos de datos, etc. con la característica de ser aplicaciones que requieren de software adicional en los equipos donde se pretende utilizar tales aplicaciones, como por ejemplo, la máquina virtual de Java el cual es necesario para ejecutar aplicaciones desarrolladas con dicho lenguaje, además de contar con ciertas restricciones como es la manipulación de archivos con conjuntos de datos. El proyecto el cual centramos esta documentación, no requiere de software adicional sólo una conexión a internet y un navegador de la preferencia del usuario además de tener habilitado la aceptación de código JavaScript sobre dicho navegador.

Objetivo General

Como objetivo general tenemos la creación de una página Web y un sistema para graficación de funciones matemáticas que sirva de apoyo a proyectos de investigación por la Facultad de Ciencias de la Computación y la Facultad de Ciencias Químicas, el cual requiere de métodos de ajuste de datos por medio de Mínimos Cuadrados no Lineales.

Objetivos específicos

Como objetivos particulares tenemos:

1. Recopilación y análisis de Bibliografía relacionada con:
    a. HTML

   b. PHP
   c. Java
   d. Servidor Apache
   e. Sistema Operativo Linux
   f. Graficación
   g. Protocolo de Comunicación TCP/IP
   h. Diseño de Páginas Web y la Interfaz
   i. Documentación del sistema

2. Diseño e implantación de una página Web para acceder a un sistema de graficación.

3. Análisis, diseño e implantación de un sistema para graficación de funciones matemáticas.

4. Elaboración del documento de tesis.

Justificación

Este proyecto de tesis se justifica con los siguientes puntos:

Facultad: Es de utilidad para los alumnos de la Facultad de Ciencias de la Computación para trabajos posteriores, los cuales podrán tomar como base esta documentación, para aumentar los alcances del proyecto que engloba este trabajo, migrarlo a otra metodología y utilizar otras herramientas de desarrollo.

Sociedad: Ofrecer una nueva alternativa para el graficado de funciones en 2D y que evita realizar costosas compras de software que si son llevadas a cabo con otras aplicaciones.

Mi Persona: Aplicar los conocimientos adquiridos durante el estudio de la licenciatura, además de adquirir nuevos conocimientos y una mayor experiencia en el desarrollo y mantenimiento de software, especialmente con tecnología de código abierto.

Descripción del contenido

Contenido de este documento; el capítulo 1 presenta el marco teórico, el cual describe algunos conceptos que serán necesarios en la elaboración de este proyecto. En la sección 1.1, los conceptos mencionados están relacionados con el tema de redes e Internet, en donde se hace mención de las distintas características que tienen las redes existentes actualmente.
Se analizan las distintas capas que componen el modelo de referencia OSI y la influencia que tiene este importante modelo en el diseño de los distintos modelos de red como lo es TCP/IP.
En la sección 1.2, se analizan los distintos protocolos más utilizados en las aplicaciones cliente-servidor y en la sección 1.3 se menciona la evolución de los distintos lenguajes de programación de bajo y alto nivel.

En la sección 1.4 se proporciona el análisis de las operaciones básicas de graficación como son rotación, escalación, traslación, etc. Y finalmente en la sección 1.5 se presenta el estudio de los distintos modelos de ingeniería de software que serán de utilidad para tener un seguimiento más controlado en la creación del sistema de software.

En el segundo capítulo, sección 2.1 realizaremos el planteamiento del problema del cual nos basaremos para la toma de decisiones como es la elección de un modelo de ingeniería de software (cascada, espiral, incremental, etc.).
En la sección 2.2 llevaremos a cabo el análisis de requerimientos del usuario y del sistema los cuales nos ayudarán a tener una mayor comprensión de que es lo que realmente pretendemos llevar a cabo.

Posteriormente, en el capítulo tres nos centraremos en crear de forma general el diseño del sistema. En la sección 3.1 identificaremos las representaciones lógicas de objetos de datos además de los módulos del programa que operan directamente sobre tales objetos de datos. En la sub sección 3.1.4 son modelados las transformaciones de los objetos a través de diagramas de flujo de datos de los cuales obtenemos mayor detalle del sistema conforme avancemos en los niveles de profundidad que conforman este modelado.

En el capítulo cuatro, sección 4.1, es en donde se lleva a cabo la implementación del sistema, no sin antes haber hecho un estudio de los distintos lenguajes de programación para desarrollar aplicaciones en entornos web y elegir alguno de ellos para desarrollar el sistema de software.
En la Sección 4.2 realizaremos las pruebas pertinentes en la utilización del producto final.

En la parte final de este documento se proporciona, en dos apéndices, una breve descripción de las herramientas utilizadas en el proceso de desarrollo del producto de software. Estas herramientas serán mencionadas a través de los apéndices 1 y 2.
El apéndice 1, detalla la instalación y puesta a punto del producto de software; mientras que en el apéndice 2, describiremos brevemente las herramientas utilizadas en el desarrollo de este sistema: HTML, JavaScript, Hojas de Estilo y PHP.

Contenido

Capítulo 1: Marco Teórico

- ✓ Introducción
- ✓ Redes e Internet
- ✓ Aplicaciones Cliente/Servidor
- ✓ Lenguajes de Programación
- ✓ Graficación
- ✓ Ingeniería de Software

Introducción

Como se ha mencionado anteriormente, el propósito principal de este trabajo es la creación de una página web y una interface que sirva de apoyo en el análisis de funciones matemáticas y posteriormente aplicarles métodos de ajuste de mínimos cuadrados no lineales. Por lo anterior es necesario analizar algunos conceptos relacionados para lograr tal propósito.

En el presente capítulo se discuten los conceptos de redes e Internet además de los distintos tipos de redes que existen. También se analiza el modelo de referencia OSI el cual es empleado como prototipo en la creación de redes (por ejemplo TCP/IP) incluyendo las distintas capas que lo conforman. Se definen las aplicaciones de tipo cliente-servidor así como los distintos protocolos de comunicación que utilizan tales aplicaciones.
Posteriormente se describen las operaciones básicas de graficación, la evolución de los lenguajes de programación y finalmente se hace referencia a los principios fundamentales de ingeniería de software.

1.1 Redes e Internet

En su forma más básica, una red de computadoras es sencillamente dos computadoras que se comunican entre sí. La mayoría de las redes constan de más de dos computadoras. No obstante, los principios de comunicación son los mismos para dos, tres, o aún mil computadoras.
Una red de computadoras se establece cuando dos o mas computadoras se conectan entre sí para compartir recursos e intercambiar información, tales recursos pueden ser: documentos o bases de datos, impresoras, unidades de disco, entre otros. Las redes de computadoras están hechas con enlaces de comunicaciones que transportan datos entre los distintos dispositivos conectados a la red. Los enlaces se pueden realizar con cables, fibras ópticas o cualquier otro medio de comunicación.
Las redes de computadoras han tenido un auge extraordinario en los últimos años y han permitido intercambiar y compartir información entre diferentes usuarios a través de los distintos servicios con los cuales cuenta Internet como son: Correo electrónico, foros, chats, acceso a bibliotecas electrónicas, sistemas de procesamiento distribuido, etc.
Por tanto, Internet está compuesto por varias redes interconectadas en todo el mundo y dentro de el existen miles de tipos de computadoras que usan una gran variedad de software de red.

1.1.1 Tipos de red

Las redes pueden cubrir geográficamente grandes distancias y son fundamentalmente diferentes de aquellas que cubren distancias cortas, por tanto las redes caen en uno de los siguientes grupos: redes de área local y redes de área amplia.

1.1.1.1 Redes de área local

Conocidas también como redes LAN. Proveen comunicaciones a grandes velocidades pero sacrifican la habilidad de cubrir distancias grandes.

1.1.1.2 Redes de área amplia

En contraste, las redes de área amplia conocidas como redes WAN, conectan computadoras ubicadas en varias ciudades, estados, e incluso países, pero a cambio, operan a velocidades menores que una red LAN.

1.1.2 Topologías de red

A la forma en como están conectadas las computadoras en una red se le llama Topología. Existen diversas topologías los cuales analizaremos a continuación.

1.1.2.1 Topología en bus

Este tipo de topología, tiene la característica de que todos los ordenadores se encuentran conectados a un cable central llamado bus. Las redes de bus lineal son las más fáciles de instalar y son relativamente baratas.

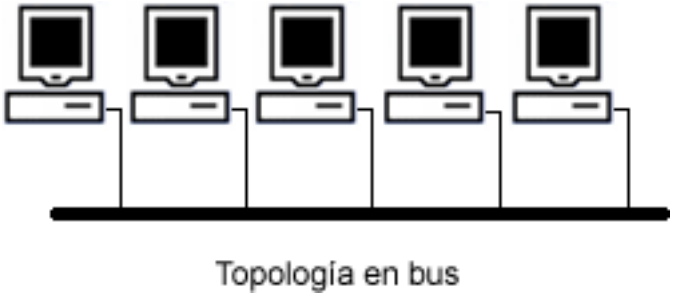


Figura 1.1.2.1[1]

1.1.2.2 Topología en estrella

Las redes con esta topología tienen una caja central de conexión llamada "hub" o concentrador el cual es el encargado de administrar todas las comunicaciones entre los ordenadores conectados a este.

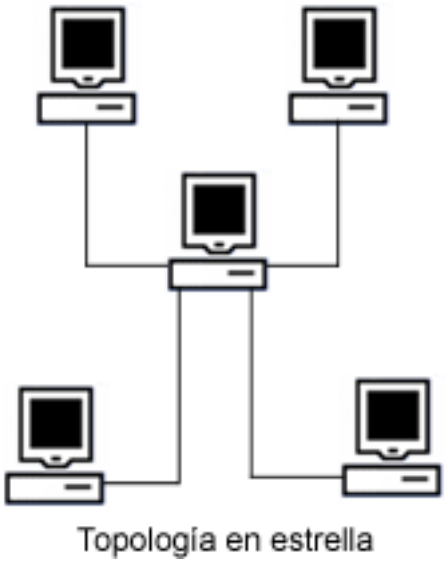
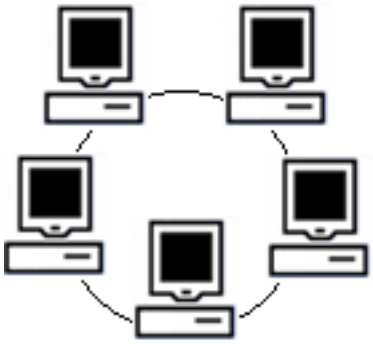


Figura 1.1.2.2[1]

1.1.2.3 Topología en anillo

En una topología en anillo cada dispositivo tiene una línea de conexión dedicada solamente con los dos dispositivos inmediatos.



Topología en anillo

Figura 1.1.2.3[1]

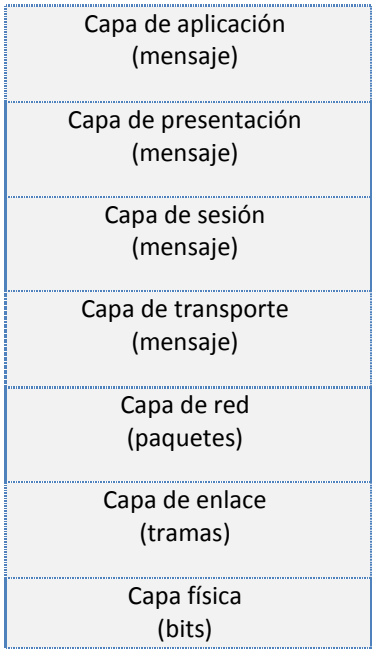
Internet es un sistema de redes interconectadas y no toma ninguna topología específica (anillo, bus, estrella, etc.), sino que conectan las redes sin importar su topología o tecnología. Para llevar a cabo una interconexión de redes se utilizan dispositivos especiales tales como son los repetidores, puentes, enrutadores, compuertas etc. Además de que estos dispositivos son utilizados para la interconexión de redes, también tienden a incrementar el desempeño como la fiabilidad y seguridad.

1.1.3 Modelo OSI

Durante las últimas dos décadas ha habido un enorme crecimiento en la cantidad y tamaño de las redes, muchas de ellas se desarrollaron utilizando implementaciones de software y hardware diferentes dando como resultado un conjunto de redes que en muchas de las veces eran totalmente incompatibles. Esta incompatibilidad generó la dificultad de que las redes pudieran comunicarse.

Para solucionar este problema, la Organización Internacional para la Normalización (ISO) creó un modelo de red que ayudara a los diseñadores de red a implementar redes que pudieran comunicarse y trabajar en conjunto, tal modelo es conocido como el modelo de referencia OSI.

El modelo OSI utiliza capas para organizar una red en módulos funcionales bien definidos, tales capas proporcionan servicios específicos a las capas adyacentes siguiendo reglas o protocolos que definen los servicios que estos ofrecen.



(Unidades de datos entre paréntesis)
Capas de la red en el modelo ISO/OSI

Figura 1.1.3[1]

Un protocolo es un conjunto de reglas y convenciones aceptadas para comunicarse.
Dentro de una red, las computadoras deben cumplir con los protocolos establecidos que controlan la comunicación, en otras palabras, los protocolos de comunicación especifican las reglas que deben cumplir los programas para transmitir y recibir datos a través de la red. En el modelo OSI se usa el nombre de cada capa para identificar el protocolo: al protocolo de la capa de transporte se le conoce como protocolo de transporte; el protocolo de la capa de red se le conoce como protocolo de red.

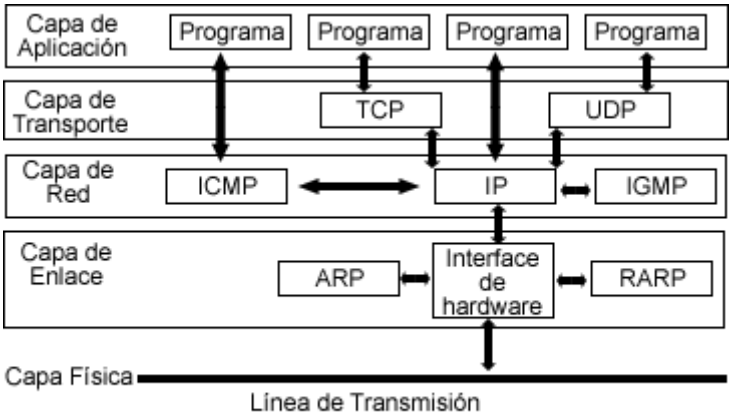
Los datos que se transmiten desde una computadora, fluyen en modo vertical desde la capa de aplicación hasta la capa física, una ves que los datos están en la capa física, los datos fluyen en modo horizontal (a través de las líneas de transmisión de la red) hasta llegar a la computadora destino. Una vez ahí, los datos fluyen hasta las capas superiores.

Los datos que viajan a través de la red se conocen como paquetes, los cuales son el resultado de dividir un mensaje en varias unidades de datos. Tales paquetes incluyen la información de origen junto con otros elementos necesarios para hacer que la comunicación sea factible y confiable en relación con los dispositivos de destino.

1.1.3.1 Protocolo TCP/IP

El modelo OSI contemplado hasta el momento no es la especificación de un diseño, ni los planos para diseñar una red específica, sino mas bien es un modelo que los desarrolladores pueden utilizar para diseñar redes; TCP/IP por ejemplo, el poderoso motor de Internet, se deriva en varias áreas de la guía de diseño del modelo de referencia OSI.
Toda la información que se mueve a través de la red, es manejada por el conjunto de protocolos TCP/IP.

La pila de protocolos TCP/IP es solo uno de los muchos que soporta el modelo en capas OSI. El modelo de referencia OSI define siete capas funcionales para el diseño de redes, sin embargo, el diseño de red TCP/IP solo utiliza cinco de las capas OSI.



Modelo de red TCP/IP con protocolos asociados

Figura 1.1.3.1[1]

Los dos protocolos más importantes son el protocolo TCP (Protocolo de Control de Transporte) y el protocolo IP (Protocolo Internet).

La capa física en la red TCP/IP es idéntica a la capa física del modelo ISO/OSI la cual incluye al medio físico de transmisión por el cual se transportan los datos por la red. Generalmente serán cables, aunque no podemos descartar la posibilidad de utilizar cualquier otro medio de transmisión como ondas o enlaces vía satélite.
La capa de enlace determina la manera en que los nodos (ordenadores) envían y reciben la información a través del soporte físico proporcionado por la capa anterior.
La capa de enlace incluye una interface de hardware y dos módulos de protocolos: el protocolo de resolución de direcciones (ARP) y el protocolo de resolución de direcciones inverso (RARP).

Por otro lado, la capa de red decide como empaquetar los mensajes para transportarlos, es decir, como fragmentar cada mensaje en paquetes de datos y enviarlos de forma independiente empleando el protocolo Internet (IP).

La capa y protocolo de transporte entrega los datos entre aplicaciones.
TCP/IP incluye dos protocolos de transporte: TCP y UDP.

Los diseñadores de TCP/IP, consideraron que los protocolos de nivel superior a la capa de transporte del modelo de referencia ISO/OSI, deberían estar incluidos en uno solo, por ende, los detalles de la capa de sesión y presentación se encuentran incluidos en la capa de aplicación del modelo TCP/IP, el cual maneja protocolos de alto nivel, aspectos de representación, codificación y control de diálogo.
La capa de aplicación es la capa más cercana al usuario final; esta capa proporciona los distintos servicios de Internet como por ejemplo el correo electrónico, páginas Web, FTP, TELNET, etc. Los programas hacen interface con los protocolos de la capa de aplicación. Sin la capa de aplicación no habría soporte de comunicación de red.

Muchas redes han sido implementadas mediante el conjunto de protocolos TCP/IP incluyendo a las agencias gubernamentales y de investigación, universidades, comercio, por mencionar algunos.

Mediante TCP/IP, se ha logrado la interconexión de redes muy distintas, independientemente de los ordenadores involucrados, sistemas operativos, etc. Con TCP/IP se tiene un método de interconexión general que es válido para cualquier plataforma, sistema operativo y tipo de red.

1.2 Aplicaciones Cliente/Servidor

La mayoría de las aplicaciones que operan en un entorno de red se clasifican como aplicaciones cliente/servidor. Estas aplicaciones, tales como FTP, los navegadores de Web y el correo electrónico tienen dos componentes especiales que les permiten operar: el cliente y servidor.
El cliente solicita los servicios y el servidor brinda los servicios en respuesta a la petición del cliente.
Una aplicación cliente/servidor funcionan mediante la repetición constante de la siguiente rutina cíclica: petición del cliente, respuesta del servidor.

Por ejemplo, un navegador Web accede a una página Web solicitando un URL, o dirección de Web en el servidor de Web remoto. Después de que ubica la dirección URL, el servidor de Web identificado por la dirección URL responde a la petición.

Netscape Navigator e Internet Explorer son probablemente las aplicaciones de navegadores de red que se utilizan más a menudo. El navegador le brinda al usuario la capacidad de navegar a través de la Web mediante hipervínculos.

Un hipervínculo es un objeto (una palabra, frase o imagen) en una página Web que, cuando el usuario hace clic en el, lo transfiere a otra página Web o a un punto determinado dentro de la página actual.

Un navegador Web es una aplicación cliente/servidor, lo que significa que requiere tanto un componente cliente como un componente servidor.
El servidor responde a la petición enviando todos los archivos de texto, audio, video y de gráficos como lo especifican las instrucciones HTML al cliente Web.
El cliente por su parte, reensambla todos los archivos para crear una vista de la página Web y luego termina la sesión.
La conexión del cliente con el servidor se mantiene sólo durante el tiempo suficiente como para procesar la transacción. En el ejemplo anterior, la conexión se mantiene lo suficiente como para descargar la página Web. En el ejemplo de una impresora, la conexión se mantiene sólo lo suficiente como para enviar el documento al servidor de impresión, una vez que se ha completado el procesamiento, la conexión se interrumpe. En los ejemplos de Telnet y FTP, la conexión establecida con el servidor se mantiene hasta que se haya ejecutado todo el proceso, en este caso el cliente finaliza la conexión cuando el usuario determina que ha finalizado.

Cada tipo de programa de aplicación se asocia con su propio protocolo de aplicación:

- La World Wide Web usa el protocolo HTTP (Protocolo de Transferencia de Hipertexto).
- Los programas de acceso remoto utilizan el protocolo Telnet para la conexión directa a las fuentes remotas.
- Los programas de correo electrónico soportan el protocolo de capa de aplicación POP3.
- Los programas de utilidades de archivos utilizan el protocolo FTP para copiar y trasladar archivos entre sitios remotos.
- La recopilación y monitoreo de datos de la red utilizan el protocolo SNMP.

El protocolo HTTP es uno de los más utilizados en Internet, este es precisamente el protocolo que se emplea en la World Wide Web fundamentalmente para la transferencia de documentos HTML e imágenes entre clientes (navegadores) y servidores Web.

Dentro de los programas de aplicación en entornos Web, el cliente es siempre quien inicia la comunicación enviando mensajes de petición al servidor, el cual responde con otro mensaje que contiene la respuesta.

1.3 Lenguajes de Programación

La comunicación entre computadoras se parece mucho a la comunicación entre humanos, ya que para que los humanos podamos entablar una conversación se requiere de un lenguaje que todos entiendan para poder saber lo que cada uno de los involucrados en la conversación están tratando de decir. Los humanos utilizan un lenguaje el cual está conformado por letras y símbolos que se combinan para formar palabras o ideas. De manera similar, las computadoras utilizan un lenguaje compuesto por dígitos binarios que representan 1 y 0. Las computadoras combinan los 1 y 0 para formar bytes de información, los cuales a su vez al ser transmitidos a otra computadora esta última puede interpretarlos con facilidad.

Con la creación de los primeros sistemas de cómputo, se ha pretendido procesar información y realizar cálculos complejos con fines muy variados como pueden ser: realizar un cálculo matemático, enviar un correo electrónico, etc.

El impacto que los sistemas de cómputo reflejan en la vida del hombre es de gran magnitud, ya que en la mayoría de los casos realizan las tareas que el mismo hombre solía realizar, con la diferencia de que estos sistemas, realizan tales actividades con un mayor rapidez y precisión.

Para que un sistema sepa lo que tiene que hacer con cierta información, se requiere de un conjunto de instrucciones para realizar la actividad de procesamiento. Ese conjunto de instrucciones se pueden procesar mediante el uso de los lenguajes de programación. Es decir, con el uso de los lenguajes de programación, el hombre puede de cierta forma decirle al sistema de cómputo lo que él quiere que la computadora haga.

Bien sabemos que la computadora hace uso de un lenguaje binario (lenguaje de bajo nivel) para comunicarse con otros sistemas de cómputo, para que el hombre tome ese mismo lenguaje y se comunique con el sistema existe una barrera muy grande por la complejidad a la que nos enfrentaríamos. Para poder evitar el uso directo de este lenguaje, se han creado ciertos lenguajes más cercanos al lenguaje utilizado por el hombre que son más fáciles de aprender y usar, con la desventaja de ser menos eficientes en términos de rapidez y memoria.

1.3.1 Evolución de los lenguajes de programación

Los lenguajes de programación se dividen en dos categorías fundamentales: bajo nivel y alto nivel. La evolución de los lenguajes de programación ha sido muy notoria, ya que con cada nuevo lenguaje de programación se requieren menos instrucciones para indicar a la computadora que realice una actividad en particular.

La primera generación de los lenguajes de programación, lo conforman solamente el lenguaje de máquina. Es considerado como de bajo nivel ya que consiste en sucesiones de dígitos binarios. Aún en la actualidad, es el único lenguaje interno que entiende la computadora.

En la segunda generación se encuentran los lenguajes ensamblador. Estos usan instrucciones como ***add*** para agregar, ***mv*** para mover, etc. Aún se utilizan estos lenguajes cuando interesa un nivel máximo de eficiencia en la ejecución. Al igual que los lenguajes de máquina, el lenguaje ensamblador es dependiente de cada máquina en particular. Esta dependencia los hace pertenecer a al conjunto de lenguajes de bajo nivel.

Los lenguajes de la tercera generación son más fáciles de aprender y utilizar, ya que se acercan más al lenguaje humano, aunque como se mencionó son menos eficientes en términos de rapidez y memoria. Estos lenguajes son totalmente independientes del hardware de la computadora, esto significa que el mismo programa puede utilizarse en varias computadoras de distinto fabricante. Los beneficios que aportan estos lenguajes son una mayor productividad del programador y claridad de la lógica.

Dentro de la cuarta generación se hace hincapié en qué es lo que de debe hacer en lugar de especificar como ejecutar una tarea. Las características distintivas de los lenguajes de cuarta generación son:

1. Uso de frases y oraciones parecidas al inglés para emitir instrucciones.
2. No operan por procedimiento por lo que permite al usuario centrarse en lo que hay que hacer, no en cómo hay que hacerlo.
3. Son utilizados para generar presentaciones de consulta y creaciones de informes.

Algunos lenguajes de programación de alto nivel, hacen uso de un compilador el cual se encarga de traducir las instrucciones del lenguaje contenidas en un código fuente a lenguaje de máquina, de manera que el programa no necesita interpretar o convertir cada instrucción. Por tal motivo es mucho más velos que un intérprete, ya que este último va traduciendo cada instrucción y no trabaja en código de máquina de forma directa.

1.4 Graficación

Los sistemas de cómputo son ampliamente utilizados hoy en día por dependencias gubernamentales, en negocios, educación, etc.
Algo que no podemos descartar es su uso en aplicaciones en cuanto a efectos gráficos se refiere. Muchos lenguajes de programación ofrecen primitivas que nos facilitan la manipulación de objetos gráficos y poder desplegarlos sobre algún dispositivo de salida. Para la manipulación de objetos gráficos, se hace uso de conceptos de píxel, transformaciones geométricas, entre otros.
Las transformaciones geométricas son utilizadas para realizar cambios en cuanto a la orientación, tamaño y forma de imágenes. Las transformaciones geométricas básicas son: traslación, rotación y escalación.

En los elementos gráficos, llevamos cuenta de nuestra posición usando coordenadas de píxel. El píxel es el elemento básico de construcción para los objetos gráficos en las computadoras, los cuales, mediante su agrupación podemos dibujar líneas, figuras, texturas, y otros objetos gráficos. Los píxeles pueden ser direccionados mediante el uso de un sistema de coordenadas similar al que se utiliza cuando se accede a renglones y columnas en una tabla (sistema de coordenadas cartesianas).

Todos los modos gráficos en la computadora inician sus coordenadas en la esquina superior izquierda de la pantalla con las coordenadas (0,0). Los valores sobre ***x*** se incrementan de izquierda a derecha y los del eje ***y*** de arriba a abajo; la amplitud de su sistema de coordenadas depende del adaptador de video que se utilice. Este sistema de coordenadas de pantalla es conocido como sistema de coordenadas de dispositivo.

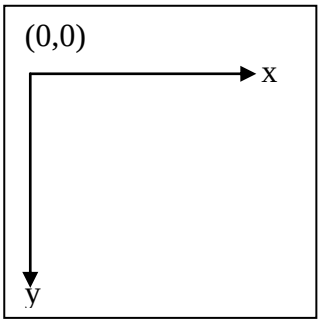


Figura 1.4[1]

Cada modo gráfico tiene una relación de aspecto asociado a el. Esta relación de aspecto está basada en la relación entre la anchura y altura de cada píxel. Tal relación es crítica cuando tratamos de dibujar en la pantalla una figura de un tamaño particular y de una forma determinada, esto porque normalmente cada píxel tiene un tamaño mayor en la altura que en la anchura, por lo tanto, cuando tratamos de dibujar un cuadrado en la pantalla, este gráfico no tendrá el aspecto de un cuadrado. Por lo tanto hay que hacer ajustes para la relación de aspecto, con los cuales se obtendrá un objeto correctamente proporcionado y posicionado.

Los problemas que podemos enfrentar al utilizar un modo gráfico específico, son: al intentar desplegar una imagen de *10 x 10* en un modo de *320 x 200* y posteriormente en un modo gráfico de 640 x 200, obviamente los resultados serían diferentes en tamaño, además, si especificamos que la imagen esté en la posición (500,150), este objeto no se vería en un modo de 320 x 200.
Para solucionar esto necesitamos de la habilidad para transformar objetos de tal manera que se exhiban proporcionados y posicionados correctamente.

1.4.1 Mapeo de coordenadas del mundo a coordenadas de pantalla

En la consideración de realizar ajustes a los objetos gráficos para que mantengan su tamaño cuando son desplegados en modos gráficos con diferentes resoluciones de pantalla, utilizamos un sistema de coordenadas estándares llamadas coordenadas del mundo (tales como metros, centímetros, pulgadas, pies, etc.) y luego convirtiéndolos a coordenadas de pantalla para que ocupen su lugar correcto antes de ser desplegados (mapeo de conjunto de datos reales a una zona de trabajo).

A este proceso de transformación de coordenadas del mundo a coordenadas de pantalla, se conoce como "*mapeo*", para ello utilizamos las siguientes ecuaciones:

$x' = a * x + b$ donde $a = \frac{R2-L2}{R1-L1}$, $b = L2 - a * L1$
$y' = c * y + d$ donde $c = \frac{T2-B2}{T1-B1}$, $d = B2 - c * B1$

Las ecuaciones anteriores, definen como un punto ***(x,y)*** en coordenadas del mundo se convierte en coordenadas de pantalla ***(x',y')***, es decir, mediante estas ecuaciones lograremos transformar los objetos representados en coordenadas del mundo hacia la pantalla.
De la misma manera, para convertir coordenadas de pantalla a coordenadas del mundo haremos uso de las ecuaciones:

$x = \frac{(x'-b)}{a}$ donde $a = \frac{R2-L2}{R1-L1}$, $b = L2 - a * L1$
$y = \frac{(y'-d)}{c}$ donde $c = \frac{T2-B2}{T1-B1}$, $d = B2 - c * B1$

Los valores de L1, T1, R1 y B1 representan el rango de valores en coordenadas del mundo, y de manera similar, L2, T2, R2 y B2, definen el tamaño del área de la apantalla en el que se colocará el objeto.

1.4.2 Transformaciones Geométricas

Una transformación es una función que define como se debe cambiar o transformar un conjunto de datos en otro.

1.4.2.1 Traslación

Una traslación define como se supone que debe moverse un objeto de un lugar a otro sin deformarlo.

La traslación de un píxel en la pantalla se debe hacer sumando un valor apropiado a cada una de las coordenadas x y y. Para hacer una traslación de un píxel, hacemos uso de las siguientes ecuaciones:

$$x\_new = x + traslate\_x$$
$$y\_new = y + traslate\_y$$

1.4.2.2 Escalación

Una transformación de escalación, altera el tamaño de un objeto. El cambio de escala a un polígono, se puede efectuar multiplicando cada una de las coordenadas del polígono original por un factor de escala. Esto se pude efectuar con las siguientes ecuaciones:

$$x\_scaled = x * scale\_x$$
$$y\_scaled = y * scale\_y$$

El factor de escalación ***scale_x***, escala objetos en la dirección de x, mientras que el factor de escalación ***scale_y*** lo hace en la dirección de y.
Se asignan valores numéricos positivos a cualquiera de los factores de escalación. Los valores menores que 1, reduce el tamaño del objeto y los valores mayores que 1, produce una ampliación. Al especificar un valor de 1 tanto para ***scale_x*** como para ***scale_y***, no se produce ninguna alteración sobre los objetos y cuando se asigna el mismo valor a ***scale_x*** y ***scale_y***, se genera una escalación uniforme.

1.4.2.3 Rotación

La rotación de un objeto alrededor de un punto, la realizamos ejecutando las ecuaciones:

$$rotate\_x = x * \cos(angle) - y * \sin(angle)$$
$$rotate\_y = x * \sin(angle) + y * \cos(angle)$$

Las variables ***x*** y ***y*** representan el punto que se está aplicando la transformación y la variable ***angle*** especifica el ángulo en que se rota el punto.

1.5 Ingeniería de Software

Bien sabemos que actualmente cada vez más productos incorporan computadoras y software de control.
El desarrollo costeable de sistemas de software es uno de los retos más importantes de la ingeniería de software. La ingeniería de software es una disciplina que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después de que se utiliza, en este proceso de producción, predomina el poder satisfacer los requerimientos de funcionalidad y desempeño.
Una propiedad de la ingeniería de software es la necesidad de una descripción exhaustiva de lo que debe producirse, un proceso detallado conocido como el “análisis de requerimientos”.

Para la construcción de un sistema de software los procesos pueden describirse sistemáticamente como: la obtención de los requisitos de software, el diseño del sistema de software, la implementación, las pruebas, la instalación, el mantenimiento y la ampliación o actualización del sistema.
El software o producto, en su desarrollo pasa por una serie de etapas que se denominan ciclos de vida, siendo necesario, definir en todas las etapas el ciclo de vida del producto, los procesos, las actividades y las tareas a desarrollar. Por tanto, podemos decir que se denomina ciclo de vida a toda la vida del software, comenzando con su concepción y terminando con el momento de desinstalación del mismo.

1.5.1 Modelos del proceso del Software

Un proceso del software es un conjunto de actividades y resultados asociados que conducen a la creación de un producto de software.
Aunque existen muchos procesos diferentes de software, todos tienen actividades fundamentales que son comunes para todos ellos, en donde tales actividades son: las especificaciones del software, diseño e implementación del software, validación y evolución del software.
Algunos de los modelos del proceso del software más comunes en la ingeniería de software son los que mencionaremos a continuación:

1.5.1.1 El modelo en cascada

En este modelo, el producto evoluciona a través de una secuencia de fases ordenadas en forma lineal y permitiendo interacciones con el estado anterior. El número de etapas en este modelo suele variar, pero en general suelen ser:

1. Especificación de requerimientos: Consiste en reunir las necesidades del producto.
2. Diseño del software: Describe la estructura interna del producto y suele presentarse en diagramas y texto.
3. Implementación: Significa programación. El producto de esta etapa es el código en cualquier nivel, incluso el producido por sistemas de generación automática de código (Lenguajes de cuarta generación u otras).
4. Integración: Es el proceso de ensamblar las partes para generar el producto completo.
5. Mantenimiento: Es el final del proceso, donde se repara y modifica la aplicación para que continúe siendo útil.

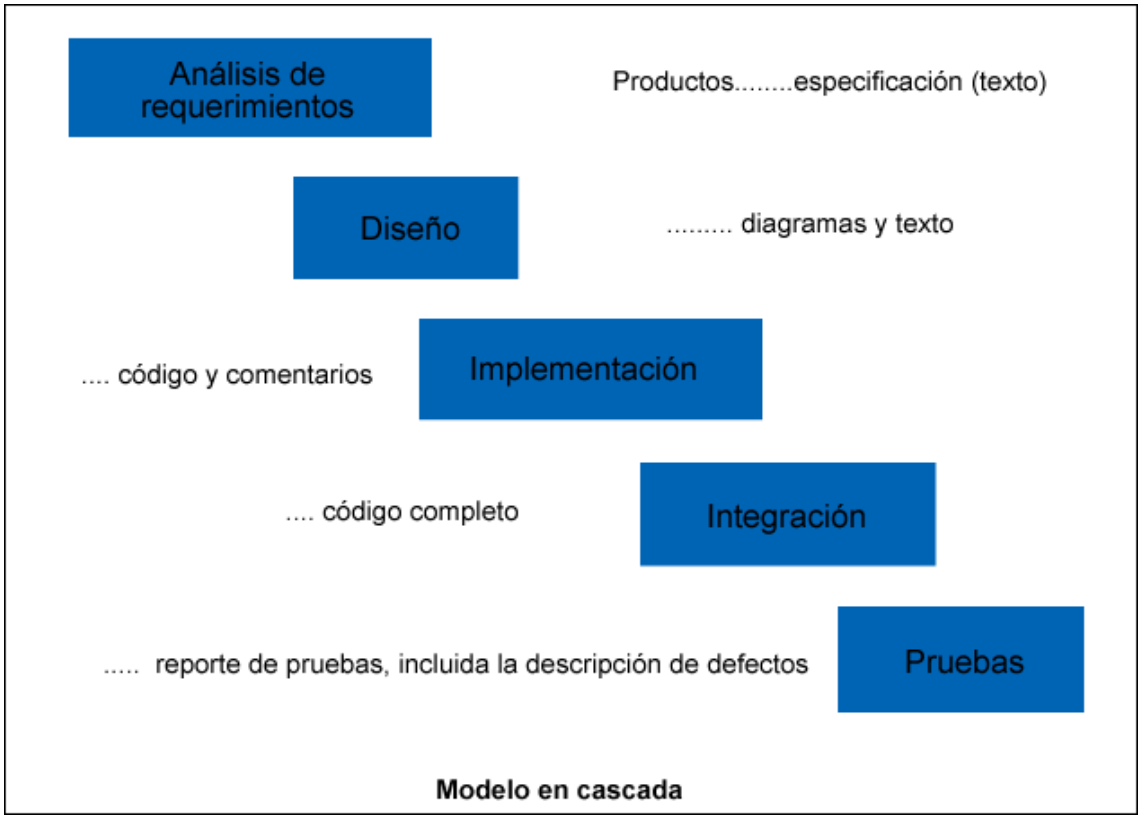


Figura 1.5.1.1[1]

1.5.1.2 Desarrollo evolutivo

Este se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado. En este modelo, más que tener actividades separadas de especificación, desarrollo y validación, estas se llevan a cabo concurrentemente.

1.5.1.3 Desarrollo orientado a la reutilización

Este enfoque se compone de un gran número de componentes de software reutilizable. Las etapas de las cuales está compuesto este modelo son: el análisis de componentes, modificación de requerimientos, diseño del sistema con reutilización y desarrollo e integración.

Para sistemas muy grandes, existe la necesidad de utilizar diferentes enfoques para distintas partes del sistema, por lo que se tiene que utilizar un modelo híbrido; los más comunes son: desarrollo incremental y desarrollo en espiral.

1.5.1.4 Desarrollo incremental

En este modelo, los clientes identifican los servicios más importantes y cuáles menos importantes que proveerá el sistema. Los servicios de prioridad más alta son los que se entregan primero al cliente. Este proceso cuenta con algunas ventajas, tales como: el cliente

no tiene que esperar a que el sistema este completo, existe un bajo riesgo de fallar en el proyecto total, entre otros.

1.5.1.5 Desarrollo en espiral

En este modelo, representa el proceso de software como una sucesión de actividades con ***retrospectiva*** de una actividad a otra. Representa el proceso como un espiral en donde el ciclo más interno podría referirse a la factibilidad del sistema, el siguiente ciclo a la definición de requerimientos del sistema, el siguiente ciclo al diseño del sistema, y así sucesivamente. Cada ciclo de la espiral se divide además en cuatro sectores: definición de objetivos, evaluación y reducción de riesgos, desarrollo y validación y por último la planeación. En la fase de desarrollo y validación es en donde se elige un modelo para el desarrollo del sistema, los cuales poden ser el modelo evolutivo, transformaciones formales, cascada, etc.
La diferencia más importante a destacar en el modelo de espiral y otros modelos es la consideración explícita del riesgo.
La idea principal de este modelo es la construcción de una nueva versión con base al resultado de la anterior.

Capítulo II: Definición del problema y especificación de requerimientos

- ✓ Introducción
- ✓ Planteamiento del problema
- ✓ Metodología

Introducción

Después de haber revisado algunos conceptos necesarios para lograr el propósito de este trabajo, es indispensable realizar el planteamiento del problema con más detalle, además de seleccionar una metodología de desarrollo para la elaboración de este proyecto.
Se elaborará también la primera fase de desarrollo del proyecto dependiendo de la metodología elegida.

2.1 Planteamiento del problema

Cuando hacemos una presentación gráfica, lo que pretendemos es presentar los datos que estamos manejando de manera que sea más fácil de interpretar, tales como: conocer cuando un proceso ha cambiado suficientemente e intentar remediar una situación, ofrecer la posibilidad de determinar por medio de un estudio de una ecuación o conjunto de datos, sus características geométricas y propiedades para una condición dada, etc.

Estas presentaciones tienen un mayor provecho al ser puestas a disposición de todo público para su libre utilización. Hasta hace algunos años toda información presentada al público con la característica antes mencionada era difícil de manejar y/o actualizar, por lo que esa información permanecía invariante por un periodo de tiempo o quizás nunca era actualizada. Actualmente, con la aparición de Internet, mucha información puede ser consultada y actualizada en todo momento y que la gente pueda ver reflejado esos cambios. Se puede con internet compartir mucha información, pero lo más destacado hasta hoy en día es que ya no solo nos limitamos a compartir información, sino que también recursos que pueden ser utilizados por muchos usuarios y que el resultado visualizado para tales usuarios dependa de la información proporcionada por ellos mismos. Un ejemplo de ello es la utilización de herramientas de graficado como la expuesta a continuación.

El software gráfico se divide en dos grandes grupos: Software de propósito especial y software de propósito general.

Los paquetes de propósito especial son diseñados para quienes no son programadores y quieren generar imágenes, gráficos, etc. de una forma rápida.
Por otro lado, los paquetes de propósito general proveen un conjunto de librerías de funciones gráficas que pueden ser usados en algún lenguaje de programación tales como C, C++, Java, PHP, etc.

El sistema a desarrollarse permitirá a los usuarios proponer funciones o importar archivos con un conjunto de datos (puntos) y obtener una gráfica que permita ver el comportamiento que toman ese conjunto de datos o funciones matemáticas a través de Internet.
Con esta característica, el sistema a desarrollar podrá ser empleado no solo para programas de investigación sino por cualquier persona que intente evaluar funciones matemáticas y obtener como resultado una gráfica que muestre el comportamiento de la función o archivo de interés.
Como se ha insistido, el sistema podrá ser utilizado vía Internet, por lo que el sistema cae en una aplicación de tipo cliente-servidor, en donde el cliente consiste en un navegador que

permita el acceso a internet y hacer uso del sistema. Por otro lado, el servidor se encargará de evaluar las peticiones de los clientes y devolver un resultado. Cabe mencionar que además de obtener una gráfica como resultado de la evaluación de un conjunto de puntos, podrán realizarse operaciones de recorte sobre la imagen después de seleccionar cierta área de interés. Y finalmente realizar operaciones de zoom y descarga del gráfico.

2.2 Metodología

Después de realizar el planteamiento del problema, diremos que la parte siguiente es elegir una metodología de desarrollo del producto de software.

La metodología se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas y documentación.

La metodología empleada para desarrollar el sistema seguirá los procesos según el modelo en cascada que como habíamos mencionado, las etapas involucradas en este modelo son: el análisis de requerimientos, Diseño, implementación, integración y pruebas.

2.2.1 Especificación de Requerimientos

La especificación de requerimientos define de forma precisa el producto de software que se va a construir. Las decisiones son tomadas en base a la información de los documentos de la propuesta del proyecto y requerimientos del usuario. El conjunto de requerimientos debe ser satisfecho por el diseño del sistema.

La especificación de requerimientos es la primera etapa del modelo en cascada. Con la especificación de requerimientos se pretende conocer las descripciones de los servicios y restricciones para el sistema. La especificación de requerimientos quedará definida en dos secciones: la especificación de requerimientos del usuario y posteriormente se desarrollará la especificación de requerimientos del sistema el cuál consiste de una especificación mucho mas detallada de los requerimientos que se hallan planteado en la especificación de requerimientos del usuario.

2.2.1.1 Especificación de requerimientos del usuario

Estas especificaciones se han obtenido de las entrevistas realizadas con los usuarios que finalmente serán quienes utilicen el producto final. Dentro de este conjunto de especificaciones, se han identificado las funciones específicas las cuales proveerá el sistema, ese conjunto de funciones se presentan a continuación:

1. Los usuarios trabajan con un conjunto de datos almacenados en archivos de texto plano (los datos son un conjunto de puntos de la forma ***(x,y)***) o bien con funciones matemáticas las cuales son evaluadas en ciertos intervalos para obtener un conjunto de puntos. La información obtenida desde los archivos almacenados en cualquier unidad de almacenamiento o desde las funciones evaluadas, se utiliza para crear una gráfica en la cual se ve reflejado en comportamiento que conservan tales conjuntos de datos. Por lo tanto, el software debe de proveer medios para

poder exportar archivos de texto plano además de los medios para que los usuarios puedan proponer funciones matemáticas que serán manipuladas por el propio sistema.

2. Una vez que los datos han sido representados sobre una gráfica, es normal que los usuarios se interesen por determinadas áreas de la gráfica excluyendo el resto de ella, por lo que es necesario contar con medios para seleccionar áreas determinadas de la gráfica además de herramientas que permitan excluir aquellas secciones que se encuentran fuera del área de interés para el usuario.
3. Contar con opciones de alejamiento y acercamiento de la gráfica resultante proporciona a los usuarios una mejor visión del comportamiento que está tomando el conjunto de datos manejado, con esto, el sistema debe de proporcionar herramientas de zoom para poder cubrir tal requerimiento.
4. Generalmente, los usuarios, después de realizar la evaluación de una función o conjunto de datos desde un fichero, están reportando observaciones obtenidas a partir de tales evaluaciones, por lo que es recomendable incluir las gráficas obtenidas en documentos de reporte. Por ello, el sistema debe proporcionar esa facilidad de descarga de la gráfica como una imagen en algún formato determinado.
5. Es importante que los usuarios tengan una guía que les permita saber qué y cómo resolver las dudas que se presenten al momento de utilizar el sistema.
6. Finalmente, los usuarios normalmente no cuentan con un equipo de cómputo propio, por lo que generalmente recurren a laboratorios de cómputo en escuelas o establecimientos de negocios para rentar una computadora y realizar sus actividades, a esto hay que sumarle que no todas las computadoras tienen el mismo software instalado y más aún, que pueden ser de distintas marcas con distinto sistema operativo. Por ende, el sistema debe proporcionar la facilidad de poder ser utilizado por varios usuarios, desde distintos lugares y funcionar con distintos equipos de cómputo con características diferentes.

2.2.1.2 Especificación de requerimientos del sistema

Como ya se mencionó anteriormente, la especificación de requerimientos del sistema consiste en una descripción más detallada de los requerimientos obtenidos en el punto anterior (especificación de requerimientos del usuario). En los requerimientos del sistema se involucran detalles como los datos de entradas y salidas, excepciones, etc.

Función:	Función matemática.
Descripción:	Al usuario se le proveerá con los recursos para definir la función matemática a evaluar y cuales serán los intervalos en el eje ***x*** y ***y*** para obtener la gráfica de la función propuesta. La definición de la función matemática deberá de corresponder con la sintaxis del lenguaje utilizado en la creación del software.
Entradas:	Intervalos en ***x*** y ***y***, función a graficar f(x), dx.
Fuentes:	El valor de los intervalos y la función son proporcionados por el usuario.
Salidas:	Grafica correspondiente a la función.
Destino:	La gráfica es almacenada como un archivo en el servidor remoto.

Función:	Importar archivos.
Descripción:	Al usuario se le proveerá además con los recursos necesarios para definir los archivos involucrados que contienen el conjunto de datos o puntos a graficar. El usuario selecciona el archivo almacenado desde su máquina local y se hará una copia para almacenarlo en un servidor remoto.
Estradas:	Archivos de texto.
Fuentes:	Los archivos son proporcionados por el usuario.
Salidas:	Gráfica correspondiente al conjunto de datos leídos desde los archivos importados.
Destino:	La gráfica resultante es almacenada como un archivo en un servidor remoto, junto con el/los archivo(s) importado(s).
Requerimientos:	Función para seleccionar y copiar archivos, además de funciones para leer su contenido.

Función:	Operaciones de zoom.
Descripción:	Se proporcionarán al usuario herramientas para realizar acercamiento o alejamiento de la gráfica.
Entradas:	Tipos de operación: alejamiento o acercamiento.
Fuentes:	El tipo de operación es seleccionado por el usuario.
Salidas:	Gráfica resultante correspondiente a un archivo o función, pero con un acercamiento o alejamiento mayor.
Destino:	La gráfica resultante es almacenada como un archivo en un servidor remoto.
Requerimeintos:	Al introducir una función o un archivo se genera la gráfica correspondiente.
Precondición:	La gráfica se despliega en la pantalla del usuario.
Postcondición:	La gráfica cambia dependiendo del tipo de operación seleccionado.

Función:	Selección de área.
Descripción:	Se proveerá al usuario con los recursos para que éste seleccione parte de la gráfica la cual es de interés para él y se eliminan aquellas áreas que no pertenecen a esa área de interés. El área de selección se realizará mediante el uso de recuadros dibujados sobre la grafica.
Entradas:	Límites del área de interés, función o archivos a graficar.
Fuentes:	Los límites son seleccionados por el usuario al dibujar un recuadro sobre la gráfica; la función o archivos son proporcionados ahora por el sistema.
Salida:	Gráfica resultante correspondiente a un archivo o función pero con nuevos límites en los intervalos en ***x*** y ***y***.

Destino:	La gráfica resultante es almacenada como un archivo en el servidor remoto.
Requerimientos:	Al introducir una función o un archivo se genera la gráfica correspondiente.
Precondición:	La gráfica se despliega en la pantalla del usuario.
Postcondición:	La gráfica cambia dependiendo de los límites del área seleccionada.

Función:	Operación de descarga.
Descripción:	Del archivo o función evaluada, se proporcionarán al usuario los recursos para obtener una copia de la gráfica resultante.
Entradas:	Solicitud de descarga.
Fuentes:	La solicitud de descarga es proporcionada por el usuario.
Salidas:	Un archivo con la gráfica correspondiente.
Destino:	El archivo es guardado en la máquina del cliente.
Requerimientos:	Al introducir una función o un archivo se genera la gráfica correspondiente.
Precondición:	La gráfica se despliega en la pantalla del usuario.
Postcondición:	La gráfica no cambia.

Para concluir con este capítulo, diremos que para poder satisfacer el requerimiento de la utilización del sistema por varios usuarios a la vez, desde distintos lugares y sin tener que preocuparse por la instalación del sistema y más aún el tipo de computadora utilizada por el usuario, el sistema a desarrollar tendrá que funcionar en red, con lo que nos lleva a planear la creación del sistema como un sitio web, al cual tendrán acceso todo tipo de usuarios.

Las herramientas requeridas para implementar y poner en funcionamiento este sistema son: Servidor con un Sistema Operativo con soporte para PHP y Apache.
Para nuestro propósito, los requerimientos de hardware mínimos son computadora personal Pentium IV a 1.8 Ghz y 256 MB en RAM, 40 GB en Disco Duro y Unidad de CD-ROM, además de conectividad TCP/IP.
Mientras que los requerimientos de software son: Sistema Operativo Linux (recomendado) o Windows, Internet Explorer o Netscape o cualquier otro navegador de internet, Apache v. 1.3.34 y PHP v. 4.42. Es importante a mencionar además que el navegador de internet debe estar habilitado con soporte para JavaScript.

Capítulo III: Diseño del sistema

- ✓ Introducción
- ✓ Modelado del sistema

Introducción

Siguiendo con la especificación de requerimientos y diseño del sistema, éste último tiene que modelarse como un conjunto de componentes y sus relaciones. Es decir, la arquitectura del sistema puede ser representada como un diagrama de bloques que muestra los principales subsistemas y la interconexión entre ellos. Este modelado puede llevarse a cabo de muchas formas, una de ellas es mediante el uso de diagramas de flujo de datos (DFD) que consiste principalmente en modelar la transformación que van sufriendo los datos a medida que fluyen por un sistema. Los diagramas de flujo de datos, pueden dividirse en varios niveles y conforme se avanza a niveles de mayor profundidad, se modela con mayor detalle la transformación de los datos en el sistema.
El diagrama de flujo de datos de nivel 0 también es conocido como el modelo fundamental del sistema o modelo de contexto y básicamente representa al elemento de software completo como una sola burbuja con datos de entrada y de salida representados por flechas [16]. Tal diagrama debe de mostrar, las posibles interacciones del sistema con entidades externas a el.
El diagrama de flujo de nivel 1 se origina de particionar el diagrama de flujo de datos del nivel 0, esto con el fin de poder mostrar más detalles del sistema. En este caso, se ha de dividir el modelo general del sistema en un conjunto de funciones que interactúan entre si y van mostrando las distintas transformaciones que van tomando los datos de entrada para poder generar finalmente la información esperada como respuesta.
En los siguientes niveles de diagrama de flujo de datos (nivel 2 y nivel 3), se realizan revisiones más minuciosas del sistema, con ellas se va generando un mayor refinamiento hasta obtener un mayor detalle del software que se está elaborando.

3.1 Modelado de Sistema

Para comenzar con el diseño del sistema, tendremos que seleccionar representaciones lógicas de objetos de datos e identificar aquellos módulos del programa que deben operar directamente sobre las estructuras de datos lógicas.

3.1.1 Identificación de entidades

De acuerdo a la especificación de requerimientos mostrada en el capítulo anterior, tenemos que las entidades encontradas son:

Usuario
Operación
Gráfica
Imagen

3.1.2 Identificar relaciones

Las relaciones identificadas entre las distintas entidades son:

El usuario realiza una operación
La operación elegida genera una gráfica
La gráfica obtenida es mapeada a una imagen
El usuario obtiene una imagen con la gráfica

3.1.3 Identificar Atributos

Operación (Tipo_operación)
Gráfica (Xini, Xfin, Yini, Yfin)
Imagen (ancho, alto)

Es importante señalar que para el usuario no se establece ningún control (el sistema estará disponible a todo usuario sin la necesidad de mantener algún registro de éste) es por ello que no se definen atributos para esta entidad.

3.1.4 Diagramas de Flujo de Datos

Aquí se modela la transformación de los objetos de datos a través de Diagramas de Flujo de Datos.

Los objetos de datos con los que se trabajó son:

Operación.
Archivos de texto.
Funciones.
Intervalos.
Imagen.

3.1.4.1 Diagrama en nivel 0

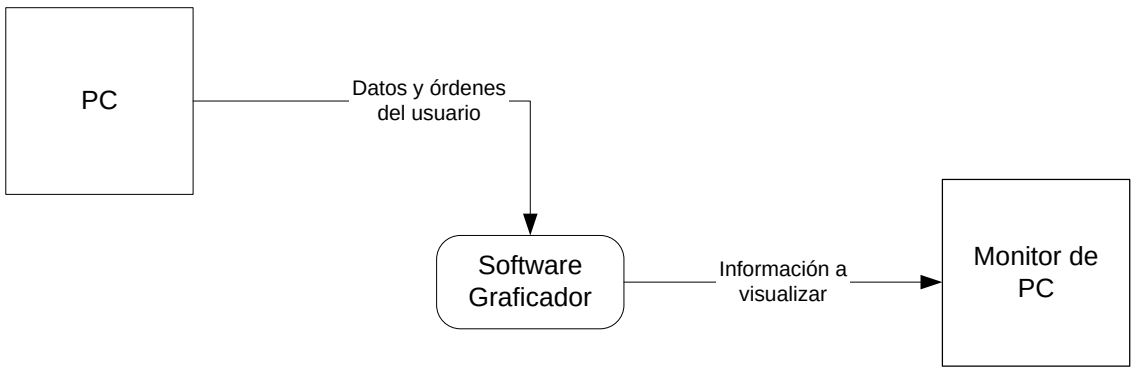


Figura 3.1.4.1[1]

En la figura anterior se muestran las interacciones del sistema con entidades externas. Puede verse que existen dos entidades externas los cuales son: la PC y el monitor de la PC, a través de la PC se introducirán las órdenes y datos del usuario los cuales serán procesados por el software y generará cierta información que podrá ser visualizada por el monitor de la PC.

3.1.4.2 Diagrama en nivel 1

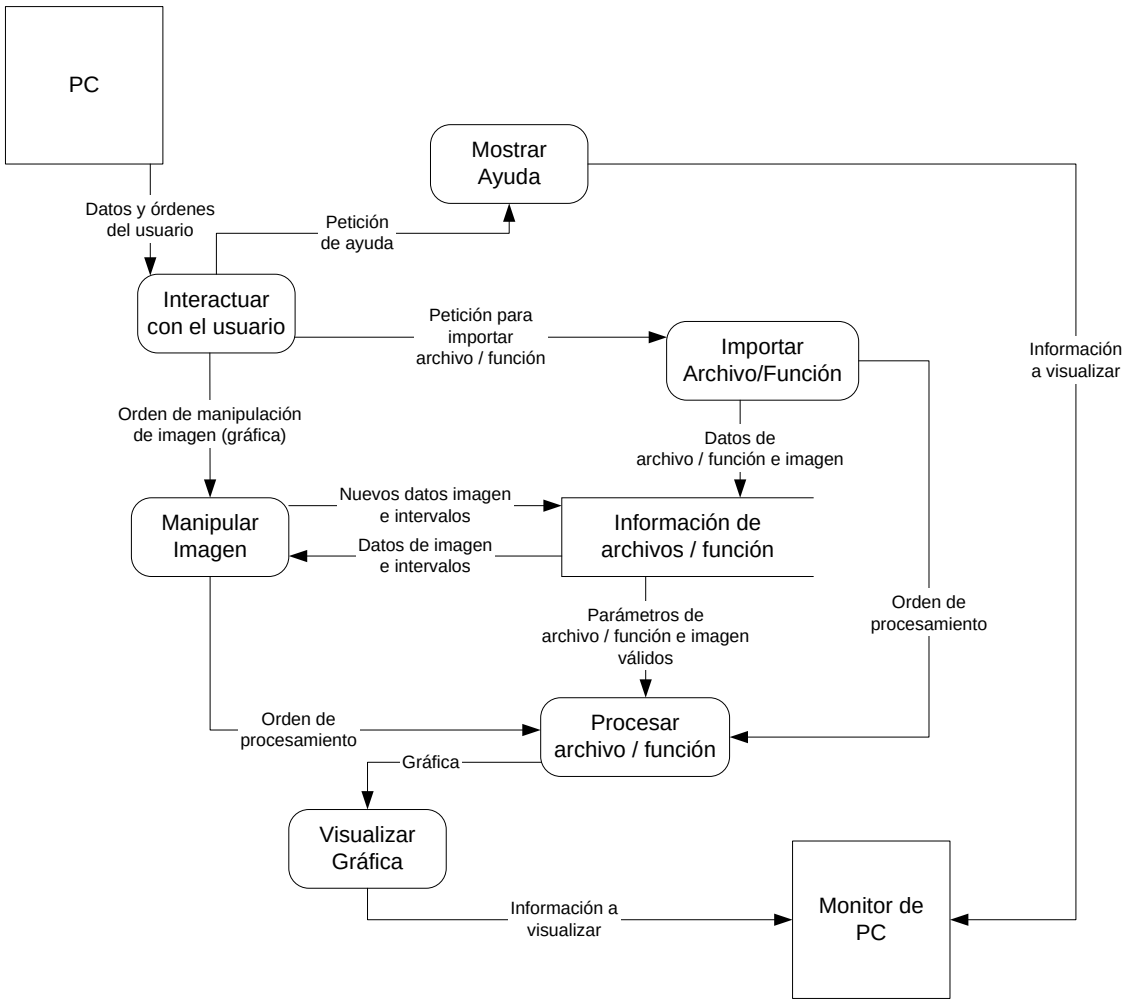


Figura 3.1.4.2[1]

En este diagrama se han identificado los distintos subprocesos y las interacciones que existen entre ellos con un pequeño almacén de datos, además de las entidades externas que interaccionan con el sistema.

3.1.4.3 Diagramas en nivel 2

- **interacción con el usuario**

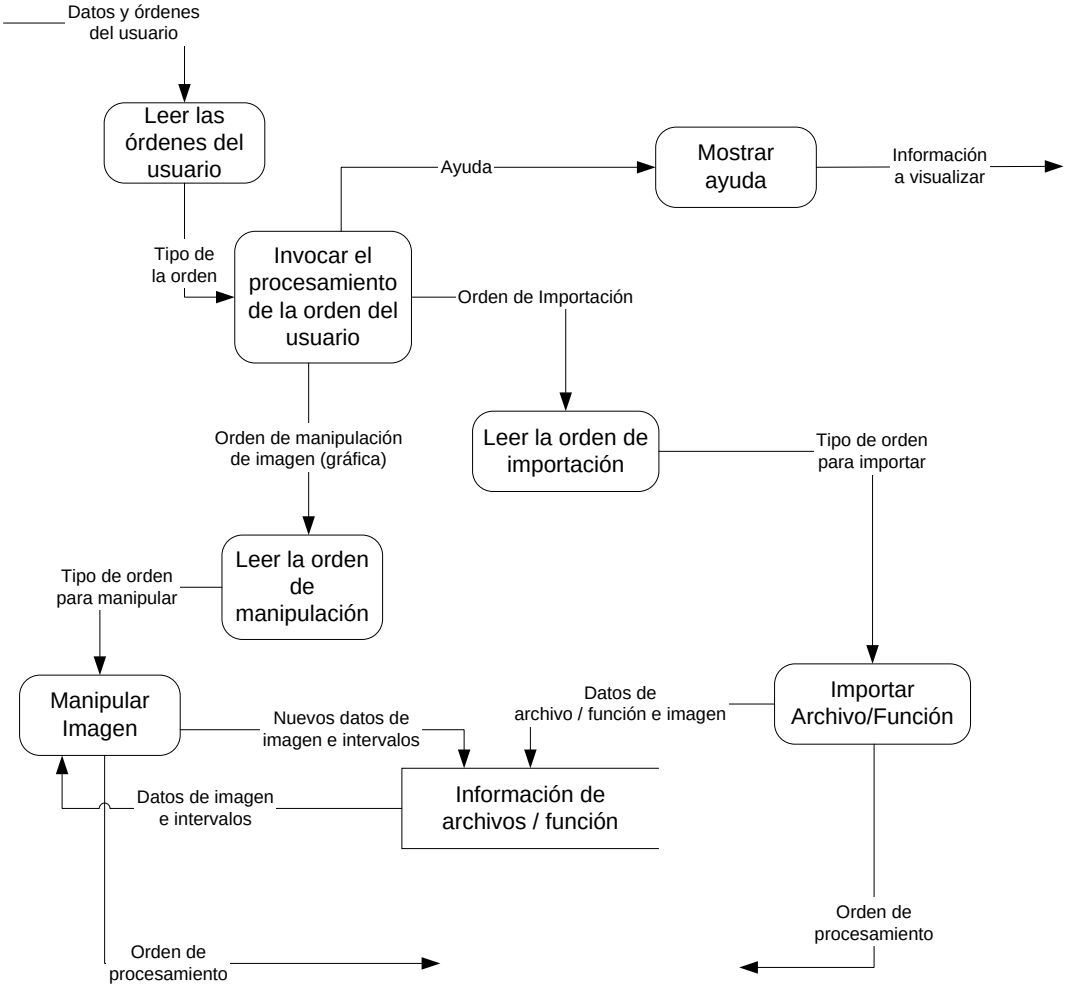


Figura 3.1.4.3[1]

En este diagrama se muestra como se ha descompuesto el proceso interactuar con el usuario del nivel 1 para mostrar más detalle de cómo el usuario puede interactuar con el sistema y elegir alguna de las posibles opciones, en este caso, el usuario podrá solicitar una pequeña guía de cómo utilizar el sistema, dar órdenes para importar archivos o funciones matemáticas o también elegir la opción de manipulación de imagen.

- **Procesamiento de archivo / función**

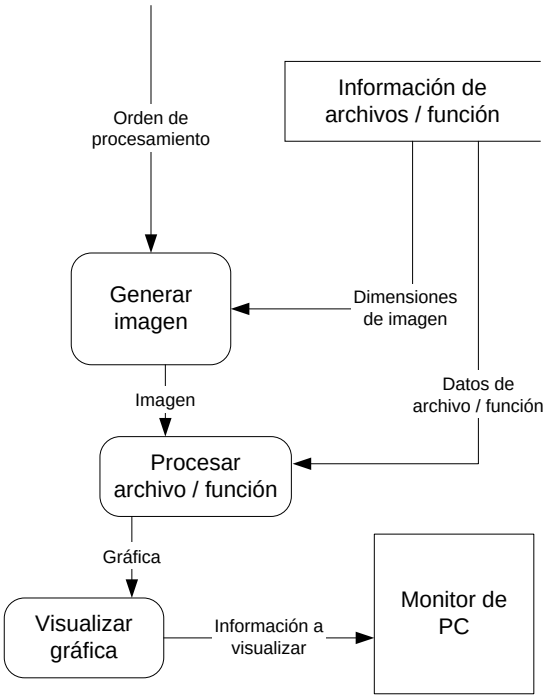


Figura 3.1.4.3[2]

En el siguiente diagrama se muestra en que consiste el proceso de procesamiento de archivo / función, el cual está compuesto de los procesos mostrados en la figura anterior. Podemos ver como es que el proceso de procesamiento de archivo / función consiste en generar inicialmente una imagen, el cual será utilizado para en el siguiente proceso (procesar archivo / función) y finalmente generar una gráfica como respuesta.

3.1.4.4 Diagramas en nivel 3

- **Manipulación de imagen**

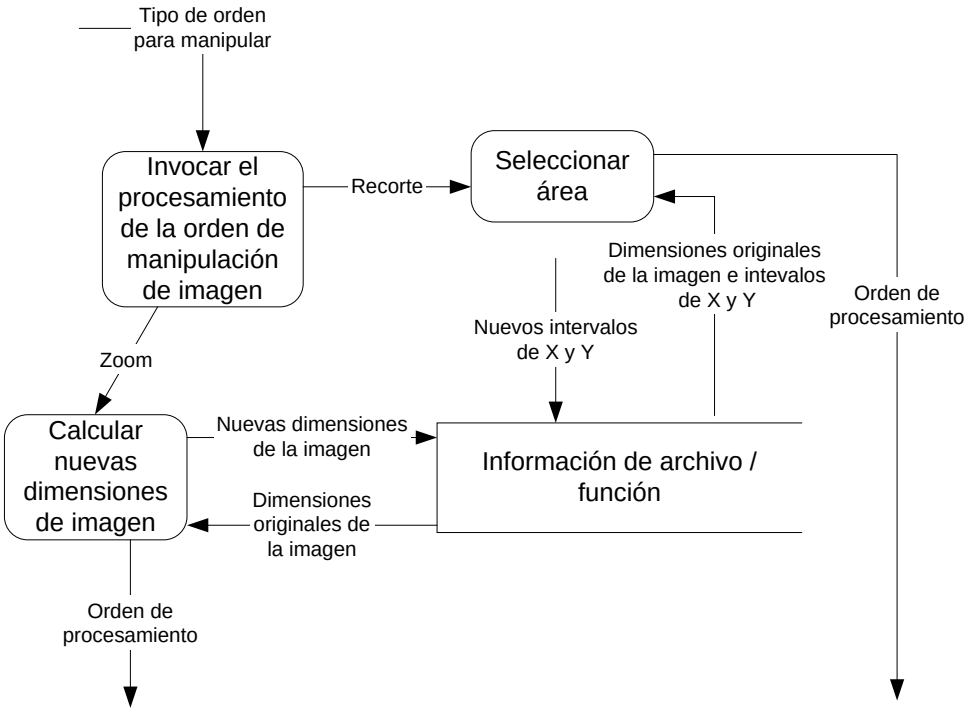


Figura 3.1.4.4[1]

En este diagrama se muestra con mayor detalle el proceso de manipulación de imagen que presentará el sistema.

- **Importación de archivo / función**

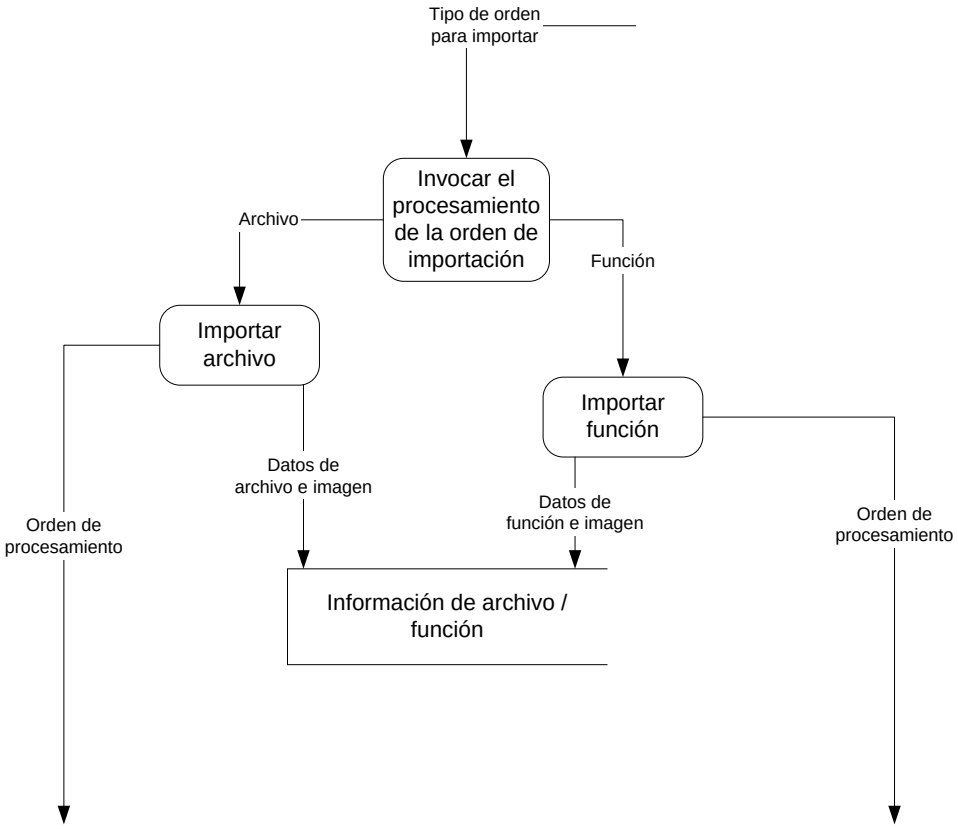


Figura 3.1.4.4[2]

Finalmente, en este diagrama mostramos con detalle las formas en que se realizarán las opciones de importación de archivo y de función.

Finalmente, de los diagramas de flujo de datos obtenemos el siguiente diagrama con los módulos principales a desarrollar:

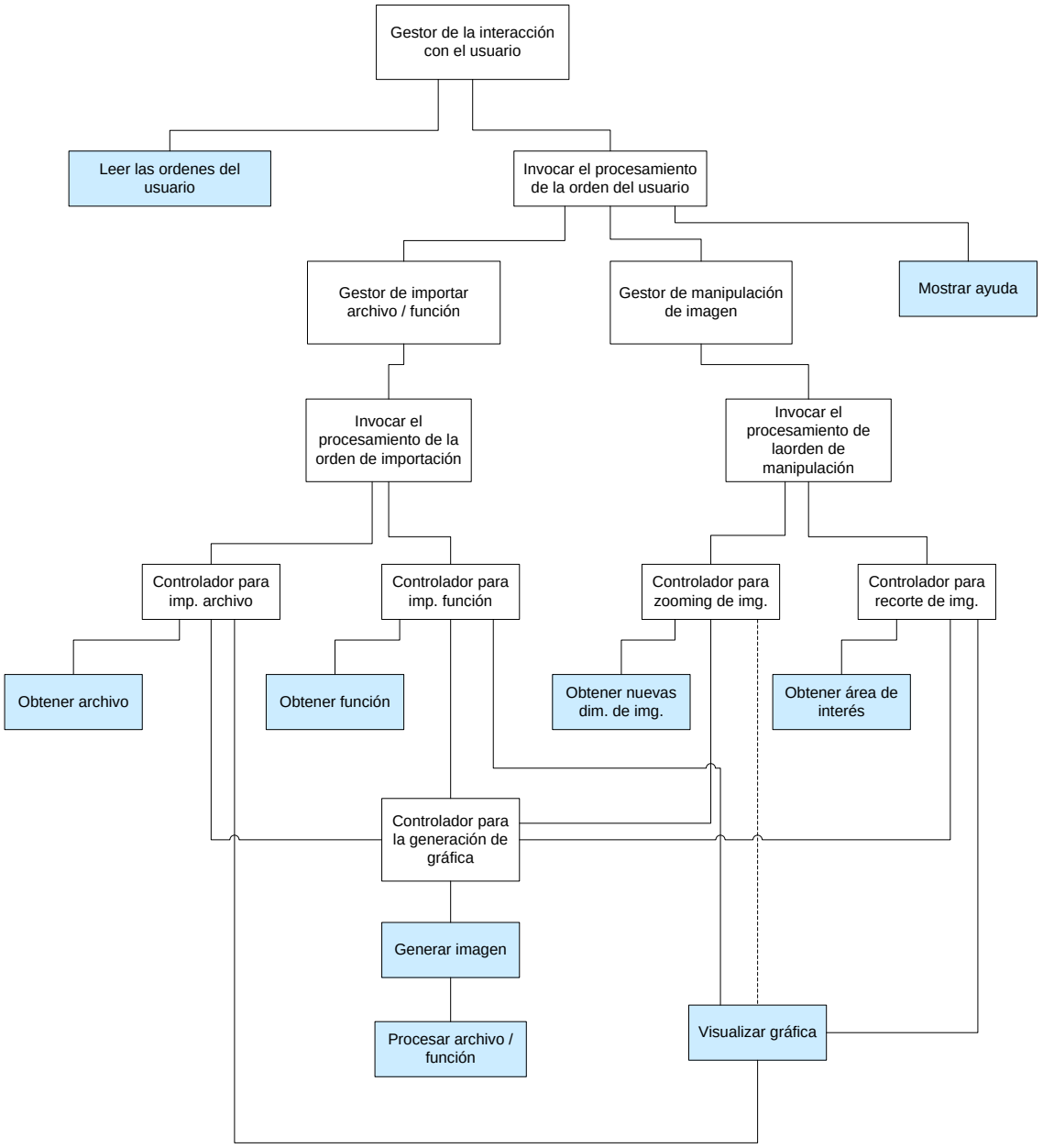


Figura 3.1.4.4[3]

En el siguiente capítulo describiremos la implementación de tales módulos.

Capítulo IV:
Implementación y pruebas

- ✓ Implementación
- ✓ Pruebas del sistema
- ✓ Conclusiones

4.1 Implementación

En el capítulo anterior, nos dedicamos a realizar el diseño del sistema y concluimos con el modelado de los módulos que han de implementarse, tales módulos serán descritos a continuación.

4.1.1 Módulo importar función

Esta función nos permite proponer funciones al sistema para que las evalúe y nos genere la gráfica correspondiente. Los parámetros que esta operación necesita para la evaluación de la función son los límites inferior y superior tanto del eje ***x*** como del eje ***y***, además de un diferencial de ***x*** (dx), que es el valor en que se incrementará la variable x en cada iteración.

Leer función y parámetros
Guardar datos leídos en repositorio

```
//Crear imagen
$img_number=imagecreate($ancho,$alto);
//Define área de trabajo (mapeo)
$REGION= new CRegionXY($amin,$amax,$bmin,$bmax);
//dibuja ejes sobre área de trabajo
$REGION->ejes();

//evaluar función
for($x=$REGION->xmin(); $x <= $REGION->xmax(); $x+=$dx){
  eval("\$y=$f_x;");
  // dibuja punto sobre la imagen creada.
  $REGION->punto($x,$y);
 }
//crea archivo png con la grafica
imagepng($res,"./image1.png");
```

4.1.2 Módulo importar archivo

En esta opción el usuario puede incorporar un conjunto de archivos (máximo dos), en donde ya se tienen los valores o puntos que se desean graficar. En estos archivos deben incorporarse los parámetros límite inferior y superior tanto del eje ***x*** como ***y***.

```
//en caso de importar dos archivos
if($archivo1 and $archivo2){
   // abre archives como solo lectura
   $fp1 = fopen("./Files/".$Archivo1, "r");
   $fp2 = fopen("./Files/".$Archivo2, "r");
   if($fp1 and $fp2){
     //obtiene limites inferior y superior X en $archivo1
```

```
    $list1x=limites('X',$fp1);
    //obtiene limites inferior y superior Y en $archivo1
    $list1y=limites('Y',$fp1);
    //obtiene limites inferior y superior X en $archivo2
    $list2x=limites('X',$fp2);
    //obtiene limites inferior y superior Y en $archivo2
    $list2y=limites('Y',$fp2);

    //Crear imagen
    $img_number=imagecreate($ancho,$alto);
    //Define área de trabajo (mapeo)
    $REGION= new CRegionXY($amin,$amax,$bmin,$bmax);
    //dibuja ejes sobre área de trabajo
    $REGION->ejes();

    //leer datos desde archivo1
    while(!feof($fp1)){
          $linea=fgets($fp1,255);
          $linea=trim($linea);
          if($linea!=""){
            $list = split( "[,]", $linea );
           // dibuja punto sobre la imagen creada
            $REGION->punto($list[0],$list[1]);
          } //cierra if
    } //cierra while
    //cierra archivo
    fclose($fp1);

    //leer datos desde archivo2
    while(!feof($fp2)){
          $linea=fgets($fp2,255);
          $linea=trim($linea);
          if($linea!=""){
            $list = split( "[,]", $linea );
           // dibuja punto sobre la imagen creada
            $REGION->punto($list[0],$list[1]);
          } //cierra if
    } //cierra while
    //cierra archivo
    fclose($fp2);
    //crea archivo png con la grafica
    imagepng($res,"./image1.png");
   }
 }else{
 //en caso de solo importar un archivo
   //abre el archivo como solo lectura
   if($archivo1!="")      $fp = fopen($archivo1, "r");
```

```
  else   $fp = fopen($archivo2, "r");
  if($fp){
   //obtiene limites inferior y superior X en $archivo1 o $archivo2
   $listx=limites('X',$fp);
   //obtiene limites inferior y superior Y en $archivo o $archivo2
   $listy=limites('Y',$fp);

   //Crear imagen
   $img_number=imagecreate($ancho,$alto);
   //Define área de trabajo (mapeo)
   $REGION= new CRegionXY($amin,$amax,$bmin,$bmax);
   //dibuja ejes sobre área de trabajo
   $REGION->ejes();

   //leer datos desde archivo
   while(!feof($fp)){
         $linea=fgets($fp,255);
         $linea=trim($linea);
         if($linea!=""){
           $list = split( "[,]", $linea );
          // dibuja punto sobre la imagen creada
           $REGION->punto($list[0],$list[1]);
         } //cierra if
     } //cierra while
   //cierra archivo
   fclose($fp);
  }//cierra if
}//cierra else
```

4.1.3 Módulo zoom

Este módulo permite realizar un acercamiento o alejamiento de la imagen para observar mejores detalles de la gráfica. En esta operación son actualizados los parámetros de alto y ancho de la imagen con la gráfica mostrada.

- Zoom +

```
alto.value=alto.value + alto.value*0.1;
 ancho.value=ancho.value + ancho.value*0.1;
//evaluar archivo o function
if($archivos)
 evaluar_archivos();
else
 evaluar_funcion();
```

- Zoom -

```
alto.value=alto.value - alto.value*0.1;
```

```
 ancho.value=ancho.value - ancho.value*0.1;
//evaluar archivo o function
if($archivos)
 evaluar_archivos();
else
 evaluar_funcion();
```

4.1.4 Módulo recorte

Esta función permite seleccionar un área de interés para el usuario. Esta selección se realiza con la ayuda del mouse y la activación de un marco de dibujo.

```
//activar marco de dibujo
marco.visible=true;

//Obtener las coordenadas del recuadro dibujado.

 var x2=marco.left + marco.width;
 var y2=marco.top + marco.height;

//Actualizar parámetros de entrada

//evaluar archivo o function
if($archivos)
 evaluar_archivos();
else
 evaluar_funcion();
```

4.1.5 Módulo descargar

Este módulo permite descargar una imagen con la gráfica en un archivo con el formato “png” para que posteriormente el usuario pueda utilizarlo en diversos trabajos.

```
//evaluar archivo o función
if($archivos)
 evaluar_archivos();
else
 evaluar_funcion();

//cambiar encabezados para la descarga de imagen

  header ('Content-type: application/png');
  header ('Content-Disposition: attachment; filename="image1.png"');
  readfile('image1.png');
```

4.2 Pruebas del sistema

Después de realizar la implementación de los módulos nos dedicaremos a realizar pruebas del sistema con ejemplos que han sido extraídos de otros trabajos relacionados con la graficación de funciones en 2D.

4.2.1 Leer las órdenes del usuario

Las principales órdenes que el usuario puede realizar sobre el sistema son aquellas referentes a los módulos descritos en el capítulo anterior. En la figura 4.1, se muestran una serie de imágenes a partir de las cuales el usuario puede seleccionar una operación, tales operaciones son:

1. Importar función: 
2. Importar archivo: 
3. Descargar imagen: 
4. Seleccionar área para recorte: 
5. Recortar imagen: 
6. Zoom: 
7. Ayuda: 

Figura 4.2.1[1]

4.2.2 Importar función

Esta operación permite al usuario la oportunidad de proponer una función para que el sistema genere una gráfica a partir de ésta. La función tiene que ser escrita en términos del lenguaje utilizado en la implementación de este sistema (PHP) por lo que se recomienda revisar la ayuda para poder proponer una función válida. En este módulo, tienen que especificarse además ciertos parámetros como son: Los intervalos en X y Y sobre los cuales queremos que nuestra función sea evaluada, además del valor en que se incrementará la variable x en cada interacción, figura 4.2.2.[1].

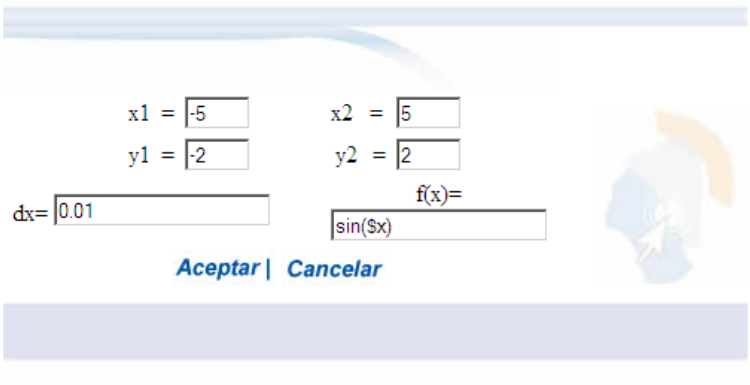


Figura 4.2.2[1]

Una vez que los parámetros sean evaluados de forma satisfactoria, el resultado será una gráfica con los valores especificados. Figura 4.2.2.[2]

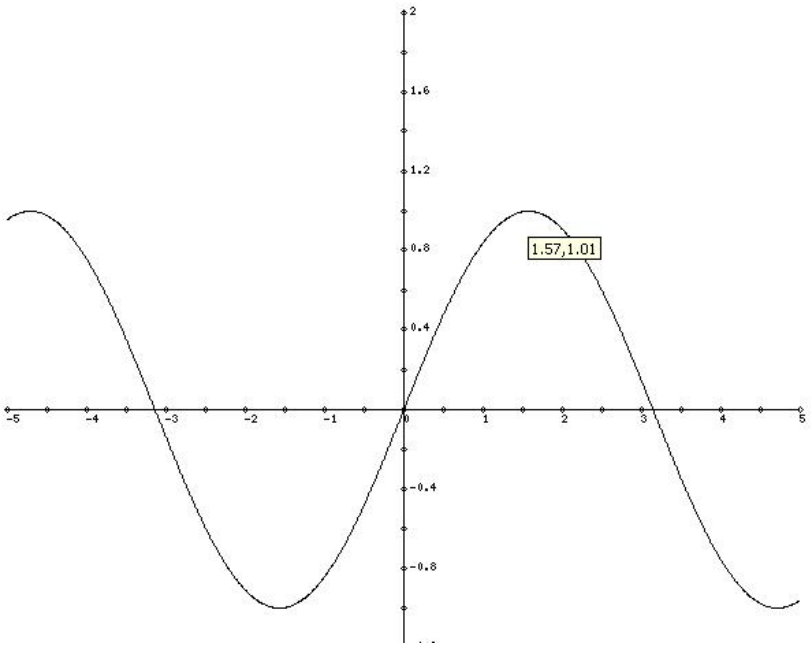


Figura 4.2.2[2]

4.2.3 Importar archivo

La operación de importar archivo, permite al usuario el mantener un archivo de texto con los datos de los valores a graficar. La sintaxis que deben tener los datos y el archivo en general es la siguiente:

IntervaloenX: [xini, xfin]
IntervaloenY: [yini, yfin]

Valor_x1, valor_y1
Valor_x2, valor_y2
…….
Valor_xn, valor_yn

Puede utilizarse cualquier editor de texto, la única restricción es que el archivo tenga la extensión “.txt”.

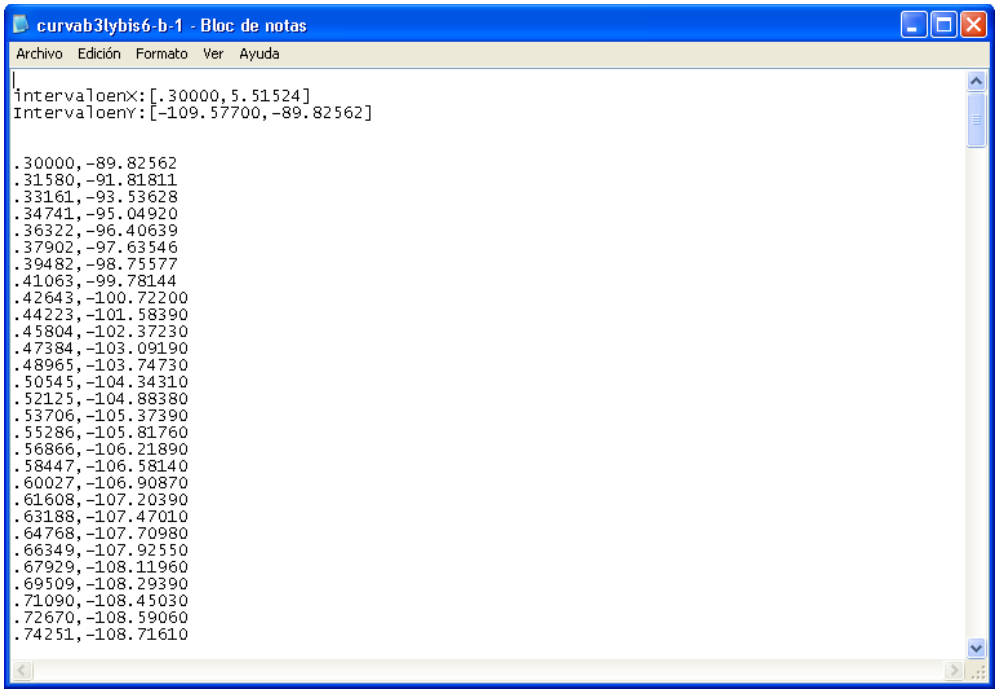


Figura 4.2.3[1]

Una vez que hemos capturado nuestros datos en un archivo de texto y elegimos la operación de importar archivo, el sistema nos muestra una ventana como la siguiente:

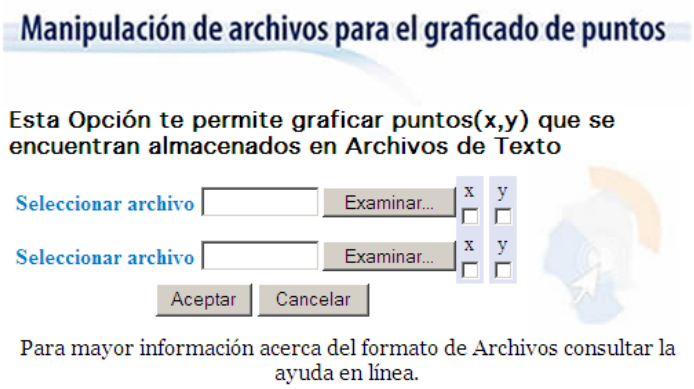


Figura 4.2.3[2]

A partir de este punto, el sistema nos pide que indiquemos el lugar en dónde se encuentra ubicado nuestro archivo y como podemos notar en la figura anterior, tenemos la oportunidad de poder importar dos archivos al mismo tiempo.
El resultado obtenido es una imagen con los valores de los archivos importados ya graficados. Figura 4.2.3[3].

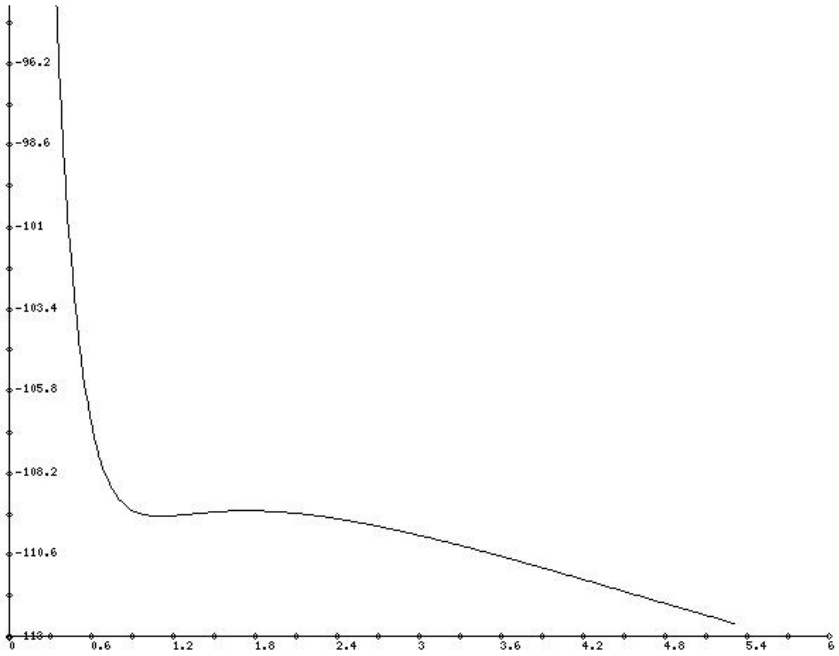


Figura 4.2.3[3]

4.2.4 Zoom de imagen

Esta operación nos permite realizar un acercamiento o alejamiento de nuestra imagen, para ello contamos con los botones:  , esta operación consiste en volver a evaluar ya sea una función o un archivo importado pero con las dimensiones de la imagen ampliada o reducida, dependiendo de la operación seleccionada.

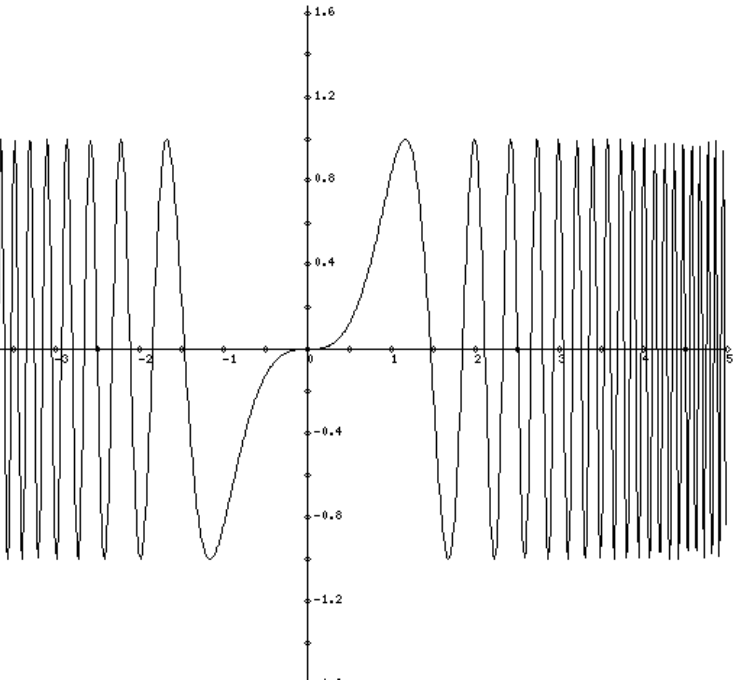


Figura 4.2.4[1] - Imagen resultante de evaluar función matemática o archivo

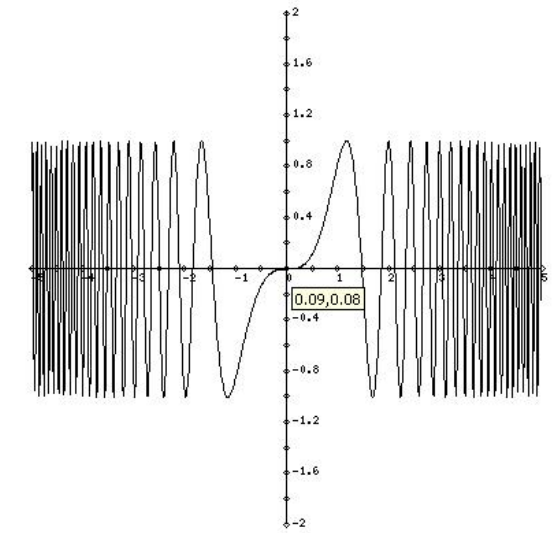


Figura 4.2.4[2] - Imagen resultante después de aplicar reducción de imagen (zoom -)

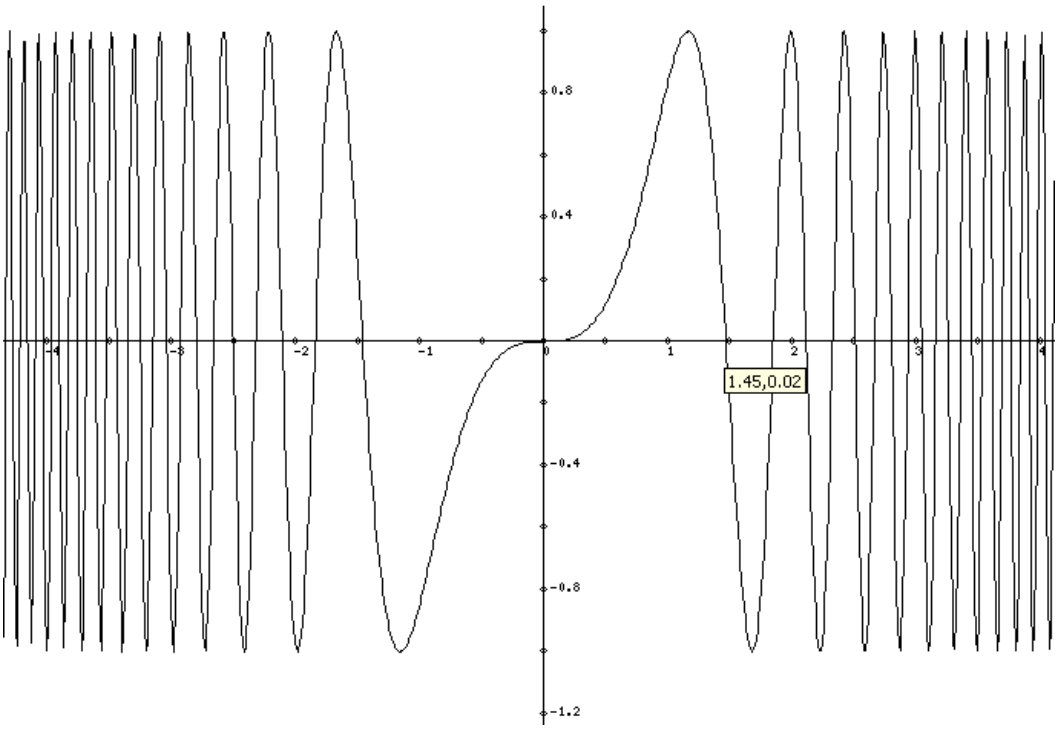


Figura 4.2.4[3] - Imagen resultante después de aplicar ampliación de imagen (zoom +)

4.2.5 Recorte de imagen

Esta operación consiste en realizar la selección del área de interés sobre la imagen que contiene nuestra gráfica. La operación de selección es realizada mediante el icono  del menú principal, esta operación permitirá que mediante el uso del mouse podamos seleccionar el área que nos interesa. Figura 4.2.5.[1]

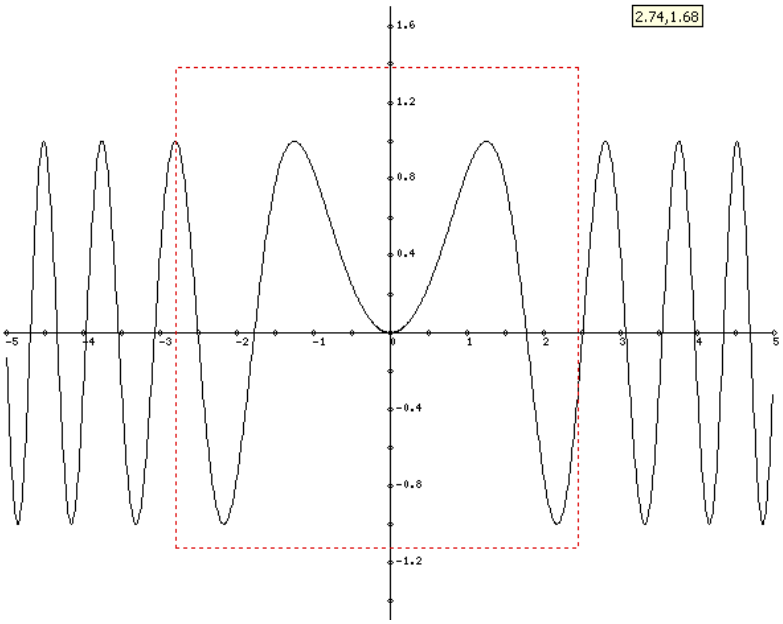


Figura 4.2.5[1] - Selección de área

Una vez que hemos seleccionado el área que realmente nos interesa, un nuevo icono será activado:  , y a partir del cual, haciendo click con el mouse sobre tal icono, este invocará nuevamente la evaluación de nuestra función o archivo para que una nueva imagen sea generada con la parte de nuestra gráfica que previamente hemos seleccionado. Figura 4.2.5[2].

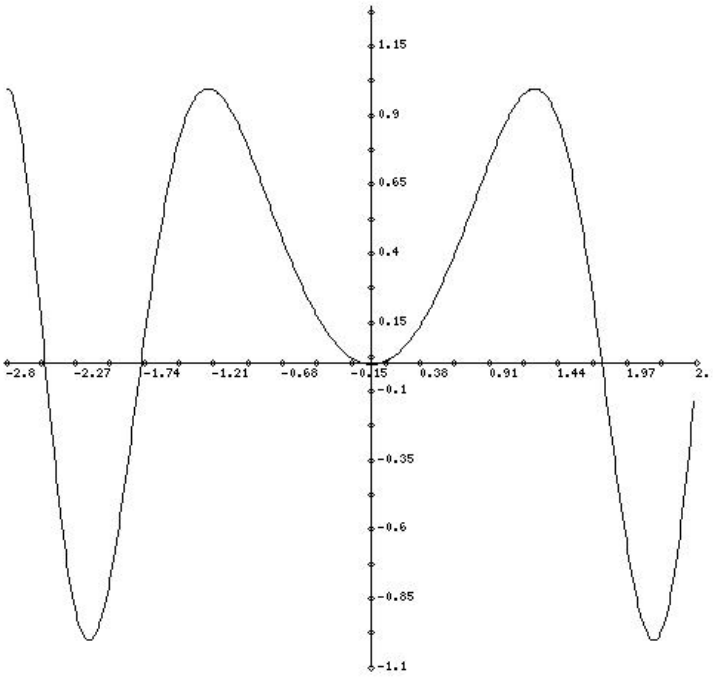


Figura 4.2.5[2] - Imagen resultante después de aplicar recorte.

4.2.6 Descargar imagen

Finalmente, la operación de descarga, esta operación consiste en volver a evaluar nuestra función o archivo, generando una imagen para descargar en nuestro equipo.

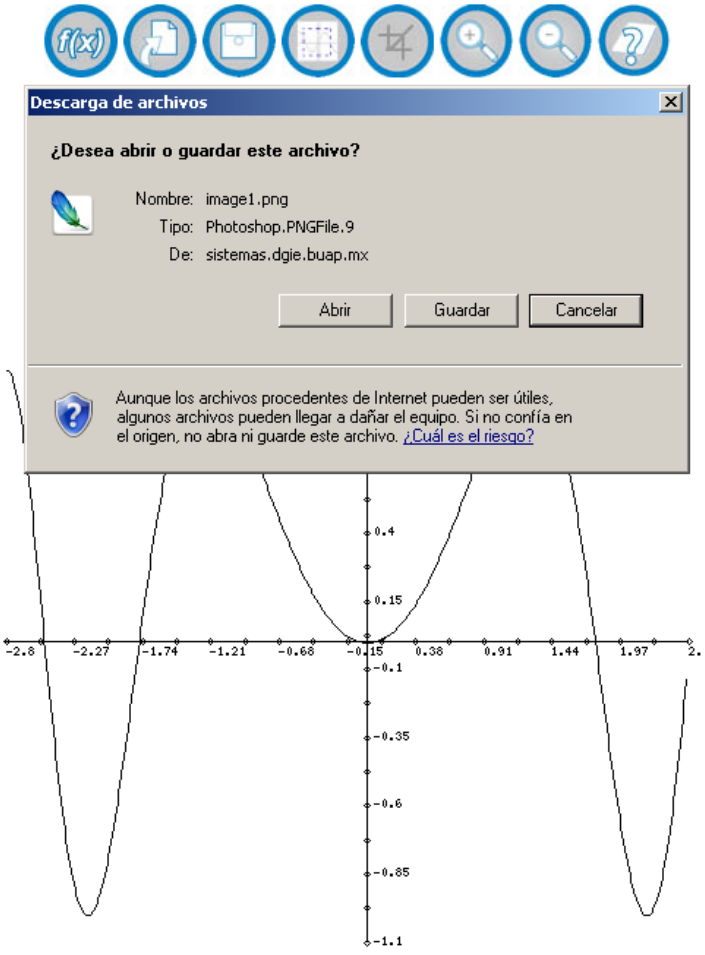


Figura 4.2.6[1]- Descarga de imagen.

4.3 Conclusiones

Existen actualmente una gran variedad de herramientas libres de costo las cuales podemos aprovechar para desarrollar aplicaciones que nos apoyen en la resolución de problemas que enfrentemos en la vida diaria. No podemos negar la existencia de software comercial enfocado a problemas específicos, para la gran mayoría de éstos los costos son muy elevados en su adquisición; las ventajas que podemos aprovechar de estos paquetes son muchas sobre todo el soporte que se aporta por parte de los desarrolladores a quienes adquieren aplicaciones de este tipo.

El sistema graficador de funciones especificado en este documento, fue elaborado con herramientas Open Source (de código abierto), y tal aplicación aunque si bien está limitada a solo generar gráficas en 2D, este es un claro ejemplo de lo mucho que es posible desarrollar con tecnología Open Source.

Como ventajas de esta aplicación podemos destacar su facilidad de acceso para hacer uso de ella, ya que no se necesita de software adicional instalado en nuestra PC más que una conexión de internet y el uso de un navegador web (especialmente Internet Explorer, Mozilla, Netscape). Además se debe mencionar que el sistema presentado en este proyecto de tesis, ofrece al usuario cierta facilidad que son propias de graficadores incluidas en sistemas más grandes, facilidades tales como:

- Zoom
- Recorte y acotamiento de áreas de interés.
- Definición del incremento sobre el eje ***x***.
- Manipulación de archivos con conjuntos de datos.

Este proyecto cubre los requerimientos establecidos y detallados en los capítulos 2 y 3, de los que podemos destacar: manipulación de archivos, herramientas de zoom, recorte y descarga de imagen. Mathcad, winplot, por mencionar algunos las cuales, son aplicaciones dedicadas al tratamiento de funciones matemáticas pero la desventaja que tienen es que estos están enfocados a sistemas operativos específicos, por lo que el adquirir uno de estos paquetes, implica tener un sistema operativo en el cual pueda ser ejecutado.

Otra ventaja que podemos aprovechar de la tecnología open Source, es que se encuentra en constante evolución por lo que podemos en algún momento ampliar las funcionalidades de nuestros sistemas.

Apéndice 1: Instalación y puesta a punto

- ✓ Introducción
- ✓ Instalación de Apache
- ✓ Instalación de PHP

Introducción

La siguiente sección está enfocada a explicar brevemente los pasos a seguir en la instalación de los requerimientos necesarios para el correcto funcionamiento de nuestro sistema.

Para poder tener acceso a este Sistema Graficador de Funciones desde la web, el principal elemento es contar con un servidor web que ofrezca la posibilidad de interactuar con nuestro intérprete PHP, recordemos que este lenguaje fue utilizado en la construcción del sistema.

El servidor web que nos ofrece esta posibilidad es Apache además de que tanto Apache como PHP están dentro de la categoría *Open Source* por lo que podemos: leer, modificar y redistribuir el código fuente de estos programas. Otra ventaja muy importante es que al instalar estos paquetes podemos configurarlos con ciertos parámetros de acuerdo a nuestras necesidades, esto hará que nuestro sistema esté funcionando sólo con los requerimientos que se requieran.
PHP y Apache pueden ser instalados en una amplia gama de sistemas operativos (Windows, Linux, Unix, etc.) para nuestros fines realizaremos la instalación sobre Linux RedHat, aunque puede ser instalado sobre cualquier otra distribución: Fedora, Debian, Suse, etc.

La instalación se realizará desde una consola con los permisos de administrador: root.

Instalación de Apache

Antes de realizar la instalación procedemos a realizar la descarga del código fuente de Apache y para ello podemos acceder a la página oficial de descarga (http://www.apache.org) en donde podremos adquirir cualquiera de las versiones disponibles.

Una vez que hemos descargado el software de instalación de Apache lo primero que haremos es descomprimir el paquete de la siguiente forma:

#tar –zxvf apache-<version>.tar.gz

Después de haberlo descomprimido tenemos que decidir en dónde quedará finalmente instalado, por ejemplo en /usr/local/apache.

A continuación realizamos la instalación de la siguiente forma;

#./configure –prefix=/usr/local/apache - -enable-module=so - - enable-module=userdir
#make
#make install

Instalación de PHP

De igual forma que Apache, podemos obtener las últimas versiones de PHP con el código fuente desde el sitio oficial http://www.php.net/downloads.php. Realizamos la acción de descomprimir el paquete y decidir en donde quedará instalado, por ejemplo en: /usr/local/php.

#tar –zxvf php-<version>.tar.gz
#./configure - - with- apxs=/usr/local/apache/bin/apx
#make
#make install
#cp php.ini-dist /usr/local/php/lib/php.ini

Después de haber terminado con la instalación de PHP y apache, la siguiente actividad a realizar es la modificación del archivo de configuración de Apache: http.conf;

En la línea con la directiva “AddType application”, añadir las extenciones que tendrán nuestros archivos escritos en PHP, la directiva queda de la siguiente forma:

AddType application / x-httpd-php .php .phtml

Para que Apache reconozca una página index.php como la página de inicio, en la directiva “DirectoryIndex” añadiremos “index.php”.

DirectoryIndex index.html index.php

Debemos indicar a Apache en dónde colocaremos nuestras páginas web para que puedan verse desde la red:

DocumentRoot /var/www/html/

El paso final es la activación del servicio de Apache:

#/usr/local/apache/bin/apachectl start

Esto hará que podamos publicar al fin nuestros sitios web y que podamos consultarlos desde cualquier computadora con acceso a internet.

Para ver que Apache está funcionando, realizaremos una prueba. Colocaremos un archivo con el nombre index.php en /var/www/html/ el cual contendrá el siguiente código PHP:

```
<?php

phpinfo();

?>
```

Y desde la barra de direcciones de nuestro navegador web: Internet explorer, Mozilla, etc. escribiremos:

http://locahost/

y lo que veremos es una página con un resumen de nuestra instalación PHP:

PHP Version 4.4.2

System	Linux phoenixx.cu.buap.mx 2.4.21-4.EL #1 Fri Oct 3 18:13:58 EDT 2003 i686
Build Date	Jun 2 2007 15:28:10
Configure Command	'./configure' '--prefix=/usr/local/php-4.4.2/' '--with-dom=/usr/local/include/' '--with-zlib-dir=/usr/local/lib/zlib-1.2.1/' '--with-zlib=/usr/local/lib/zlib-1.2.1/' '--with-gd' '--with-jpeg' '--with-xml' '--with-iconv' '--enable-mbstring' '--with-mysql=/usr/local/mysql-standard-4.1.18-pc-linux-gnu-i686-glibc23/' '--with-apxs=/usr/local/apache_1.3.34/bin/apxs'
Server API	Apache
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php-4.4.2/lib/php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20050606
Debug Build	no
Zend Memory Manager	enabled
Thread Safety	disabled
Registered PHP Streams	php, http, ftp, compress.zlib

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.3.0, Copyright (c) 1998-2004 Zend Technologies

Figura 1

Con esto hemos concluido la instalación del software necesario para que nuestro Sistema Graficador Funciones opere de la forma correcta y como paso final colocamos todos los archivos que conforman nuestro sistema en la carpeta /var/www/html y hacer uso de ella accediendo mediante un navegador escribiendo en la barra de direcciones la siguiente dirección: http://sistemas.dgie.buap.mx/~lucio/grafica/.

Apéndice 2: Herramientas

- ✓ Hiper Text Markup Language
- ✓ JavaScript
- ✓ Hojas de estilo
- ✓ Hypertext Preprocessor (PHP)

♦ Hiper Text Markup Language

Introducción

HTML son las iniciales de Hiper Text Markup Language, consiste de un conjunto de etiquetas que pueden ser incluidas en archivos de texto, las cuales definen la estructura y formato del contenido de tal documento y que finalmente pueda ser publicado en internet.

Este archivo con etiquetas es conocido como un documento web o documento HTML. Los navegadores Web (Internet Explorer, Mozilla Firefox, NetScape, etc.) interpretan las etiquetas incluidas en el documento para determinar cómo desplegar su contenido.

Etiquetas

Una etiqueta, como hemos mencionado puede especificar el formato del contenido de un documento HTML, como por ejemplo la etiqueta ***strong*** es utilizada para especificar el texto en negrita, la etiqueta ***center*** para centrar texto, tablas, imágenes, o todo el contenido de nuestro documento, todo depende de dónde especifiquemos el inicio y cierre de la etiqueta. Por tanto una etiqueta debe especificarse siempre con una apertura y un cierre, los cuales determinan el alcance de la etiqueta.

La apertura está delimitada siempre por “< > “, mientras que el cierre por “</ >”, y todo lo que indiquemos entre la apertura y cierre de una etiqueta quedará especificada con el formato que contempla la etiqueta aplicada. Por ejemplo, para especificar parte de un texto en negrita, lo hacemos de la siguiente manera:

 Hola a todos, este es un ejemplo de documento HTML.

Y el navegador interpretará lo anterior como:

Hola a todos, este es un ejemplo de documento HTML.

Para especificar un texto además de negrita en cursiva:

 Hola a todos, este es un ejemplo de documento HTML.

Teniendo como resultado:

Hola a todos, este es un ejemplo de documento HTML.

Estructura Básica

Un documento HTML está compuesto por dos partes principales: el encabezado y el cuerpo de la página los cuales están definidos mediante las etiquetas ***head*** y ***body***.
Estas dos últimas etiquetas junto con la etiqueta ***html***, son consideradas como etiquetas fundamentales que deben existir siempre en un documento HTML.

<***html***> </***html*** >, esta etiqueta abre el inicio del documento y cierra el final del mismo. Mediante éstas, el navegador se da cuenta de que se trata de un documento HTML y por tal motivo inicia la interpretación.

<***head***> </***head***>, encabezado del documento, aquí es donde se establece información de carácter informativo como por ejemplo el título de la página. Estas etiquetas se definen después de la apertura de la etiqueta ***html*** y antes de la apertura de la etiqueta ***body***.

<***body***> </***body*** >, lugar donde queda establecido el contenido global del documento. Esta etiqueta se inicia después de cerrar la etiqueta ***head*** y se extiende hasta el final del documento cerrándose antes de </***html***>

En resumen, una página Web está estructurada de la forma siguiente:

```
<html>
<head><title> Introducción a HTML </title></head>
<body>

Este es un ejemplo de documento HTML

</body>
</html>
```

Atributos

La páginas HTML, contemplan también un conjunto de atributos los cuales almacenan información que permiten personalizar el aspecto de dichas páginas, por ejemplo especificar un color o imagen de fondo a la página, especificar el color de texto y enlaces de la página, entre otros.

Lo anterior puede especificarse en la etiqueta ***body*** con los atributos:

bgcolor: especifica un color de fondo para la página.

background : para colocar una imagen como imagen de fondo en la página.

text: para especificar el color de texto en la página.

link: Color de los enlaces que no han sido visitados.

vlink: Color de los enlaces que ya han sido visitados.

alink: Color de los enlaces activos.

De lo descrito hasta ahora, podemos generar una página con un color de fondo en negro, texto y enlaces en colores claros:

<body bgcolor="#000000" text="#ffffff" link="#ffff33" alink="#ffffcc" alink="ffff00">

Los colores especificados anteriormente están contemplados en hexadecimal:

Negro: 000000

Blanco: ffffff

……..

Un ejemplo de una página con colores y márgenes personalizados es la siguiente:

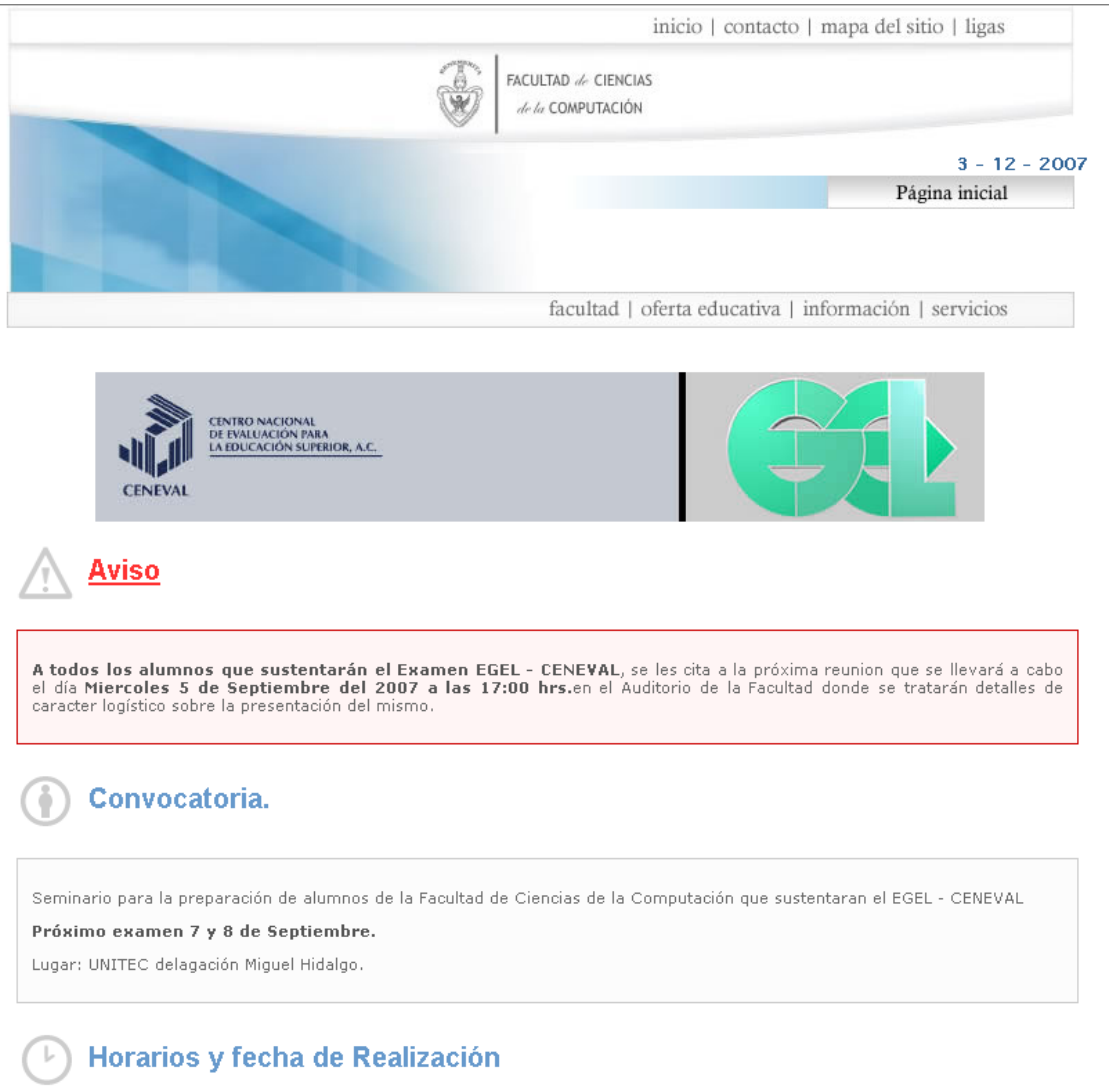


Figura 1

Formularios

Además de que HTML gestiona y muestra información mediante texto, imágenes y enlaces, existe también la posibilidad de intercambiar información con los visitantes de nuestra página; con esta nueva característica se desprenden nuevas formas de realizar compras de un artículo, llenado de encuestas, enviar comentarios al autor, etc. Para llevar a cabo lo anterior, HTML propone el uso de formularios.

Los formularios se componen de cajas de texto, botones y muchos otros elementos con los cuales el usuario puede proporcionar información que posteriormente mediante la utilización de otros lenguajes de programación más sofisticados (por ejemplo, PHP, ASP), esta información puede ser tratada de forma automática como puede ser: almacenar la información en una base de datos, enviar la información a un correo electrónico, realizar búsquedas, etc.

Los formularios son definidos mediante la etiqueta ***form,*** utilizando como apertura <***form***> y cierre </***form***>, y todos los botones y cajas de texto deben de definirse entre estas últimas etiquetas. Dentro de la etiqueta <***form***> debemos de especificar un conjunto de atributos:

name: indica el nombre dado a un formulario.

action: Define la acción a realizar una vez que el usuario ha proporcionado la información. Esta acción permitirá que la información sea enviada a un programa o script que se encargue de procesarla.

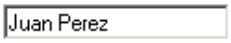
method: Especifica la forma en que el formulario será enviado, GET o POST.

enctype: Especifica la forma en que será enviada la información contenida en el formulario, "text/plain", "multipart/form-data", etc.

```
<form name="form1" action="mailto:direccion@correo.com" method="post" enctype="text/plain">
```

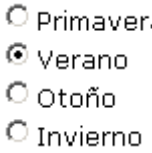
Los elementos que conforman un formulario y que funcionan como entrada de datos se denotan con la etiqueta ***input,*** estos además tienen un atributo ***name*** para determinar el nombre con el cual se hará referencia al elemento y el atributo ***type*** con varias especificaciones:

text (cajas de texto),

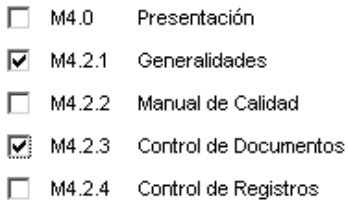


password (contraseña),

radio (botones de radio),



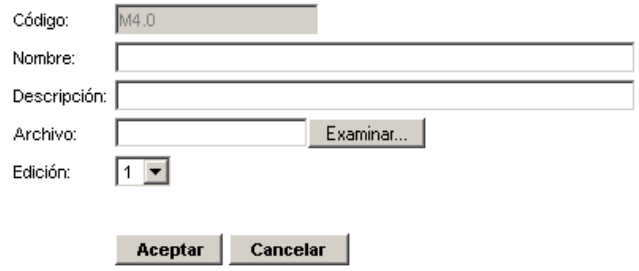
checkbox (cajas de validación),



submit (botón de envío de formulario),

reset (botón para borrado de formulario),

button (botón normal),

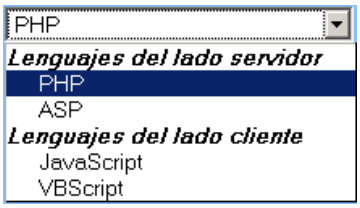


hidden (datos ocultos).

Otros elementos disponibles para el tratado con formularios son:

Áreas de texto: su utilización es mediante la etiqueta ***textarea***, además de los parámetros que contempla este elemento: ***name***, ***rows***, ***cols***, etc.

Lista de opciones: etiqueta ***select*** y adicionalmente entre la apertura y cierre (<***select***>, </***select***>) de etiqueta incluir por cada elemento de la lista, la etiqueta ***option***.



Otra de las ventajas encontradas en la utilización de formularios es el tratado de ***eventos*** que pueden fácilmente responder a funciones de JavaScript, con lo que nuestras posibilidades de uso son más extensas.

Finalmente, los archivos escritos en HTML, tienen que ser guardados con la extensión ".html".

♦ JavaScript

Introducción

JavaScript es un lenguaje de programación interpretado y por tal razón, para ejecutar las instrucciones escritas en este lenguaje se requiere de la intervención de un intérprete.
El intérprete utilizado en este caso son los navegadores web, tales como internet explorer, Netscape, Opera, etc.
Los programas escritos con este lenguaje son conocidos como scripts o guiones.
Con JavaScript podemos ir más allá de la presentación de un documento en pantalla con sus atributos tales como: colores, tipo de letra, imágenes e hipervínculos; JavaScript permite realizar menús desplegables, validar datos proporcionados por el usuario, entre otros; además de ser soportado por todos los navegadores y es sencillo de aprender, ya que solo se refiere a la inserción de código especial en un documento HTML.
En resumen podemos decir que JavaScript es un lenguaje de programación fundamentalmente para la creación de aplicaciones de internet.

Inclusión de JavaScript en las páginas HTML

Para hacer que un documento HTML incluya instrucciones en JavaScript se debe hacer uso de la etiqueta ***script*** con el atributo ***language*** asignandole el valor ***"JavaScript"***, ejemplo:

```
<script language="JavaScript">
  …Código JavaScript
</script>
```

También se puede utilizar el código JavaScript escrito en otro archivo el cual debe tener la extensión "**.js**" y posteriormente pueda ser invocado desde otra página mediante el siguiente código:

```
<script language="JavaScript" src="archivo.js">
```

Normas de escritura de JavaScript

1. Los comentarios deben empezar con el símbolo // si son de una sola línea o iniciarse con los símbolos /* y finalizar con */ si se tratase de varias líneas.
2. Las líneas de código terminan con el signo de punto y coma (;).
3. JavaScript distingue entre mayúsculas y minúsculas.
4. Las llaves ({,}) permiten agrupar código.

Variables y Operadores

Al igual que otros lenguajes de programación, JavaScrpt permite la utilización de variables para el almacenamiento de valores. Las variables tienen un determinado nombre y tal nombre puede empezar con una letra seguido de un conjunto de números, el signo "_" o también más letras.

Los valores que pueden asignarse a una variable en JavaScript pueden ser:

1. Cadenas de texto.
2. Valores numéricos.
3. Valores booleanos.

Para declarar una variable se puede utilizar:

var variable = valor;

o simplemente

variable = valor;

Por lo anterior, en JavaScritp no hace falta declarar una variable antes de utilizarla, además de que permite almacenar en una variable distinto tipo de datos en cada trozo de código.

Los operadores son elementos que permiten la realización de operaciones con los datos del código.
Los operadores empleados son:

Aritméticos	
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo (Resto de la división)
++	Incremento
--	Decremento

Lógicos	
&&	AND (Y lógico)
||	OR (O lógico)
!	NOT (NO lógico)

De comparación	
==	Igualdad
!=	Diferente
>=	Mayor o igual que
<=	Menor o igual que
>	Mayor que
<	Menor que

De asignación	
+=	Suma y asignación
-=	Resta y asignación
*=	Multiplicación y asignación
/=	División y asignación
%=	Módulo (Resto de la división) y asignación

Mensajes

JavaScript permite la integración de ventanas que pueden ser desplegadas ante el usuario para hacer que este reaccione ante una situación o nos informe ante una duda.

Los tipos de mensajes empleados son:

alert – Muestra información al usuario dando sólo la posibilidad de aceptarle.

Sintaxis: alert(“Texto del Mensaje”);

prompt – Ventana que permite dar entrada de datos por el usuario de modo que esta función devuelve un valor que se puede usar en el código si es asignado a una variable.

Sintaxis: variable = prompt(“texto del mensaje”);

confirm – Muestra un mensaje de confirmación en el cual el usuario puede elegir entre dos botones: aceptar y cancelar.

Sintaxis: confirm(“texto del mensaje”);

Estructuras de control

Algunas de las estructuras de control manejadas por JavaScript son:

- Instrucción ***if***

Realiza lo que se denomina un sí lógico. Su forma es:

```
if(condición){
..código que se va a ejecutar en caso de que la evaluación de la condición devuelva un valor TRUE.
}
```

También admite otra forma:

```
if(condición){
..código que se va a ejecutar en caso de que la evaluación de la condición devuelva un valor TRUE.
}else{
..código que se va a ejecutar en caso de que la evaluación de la condición devuelva un valor FALSE.
}
```

- Bucle ***while***

Esta instrucción permite ejecutar un conjunto de instrucciones mientras se cumpla una determinada condición.

```
 while(condición){
..código a ejecutar mientras la condición se cumpla.
}
```

- Bucle ***for***

Esta estructura es muy similar a la anterior. Permite ejecutar una serie de sentencias hasta que se cumpla una determinada condición.

```
for(valor inicial; condición; actualización){
..código que se va a ejecutar mientras la condición se cumpla.
}
```

Funciones

Existen dos tipos de funciones: las definidas por el usuario y los que están ya predefinidas por el lenguaje.

Una función es una serie de instrucciones que realizan una determinada tarea y una vez definidas, estas pueden ser invocadas desde cualquier parte del documento HTML o JavaScript.

La definición de funciones por el usuario sigue la siguiente sintaxis:

function nombre_funcion(argumento1, argumento2, …){
…Sentencias a ejecutar durante la invocación de esta función.
}

Una vez definida una función, el usuario podrá invocarla directamente mediante su nombre desde otra función o mediante el uso de eventos.

JavaScript dispone de un conjunto de funciones predefinidas los cuales no tenemos que volver a definir sólo invocarlas en el momento que se requieran. Algunas de estas funciones son ***eval y parseInt*** los cuales evalúan un texto como si se tratase de código JavaScript y convierten un texto a formato de número.

Objetos

Un objeto es una entidad el cual posee un conjunto de propiedades (los cuales describen sus atributos) y un conjunto de funciones a los cuales son llamados métodos (estos describen su comportamiento).

JavaScript dispone de una amplia gama de objetos definidos, tales como: ***Array, boolean, Date, Funtion, Global, Math, Number,*** entre otros, además de que el propio usuario puede crear nuevos objetos con sus propios métodos y propiedades.

Para acceder a las propiedades de un objeto, invocamos al objeto y su propiedad de la forma siguiente:

objeto.propiedad

Para hacer uso de los métodos de un objeto, invocamos al objeto y el método de interés mediante:

objeto.metodo()

En el uso de métodos de un objeto, tenemos que tomar en cuenta la posibilidad del uso de algún parámetro.

objeto.metodo(arg1, arg1, …)

En la sección Hiper Text Markup Language (HTML) de esta documentación se mencionó la integración de formularios en archivos HTML. Dichos formularios están compuestos por una serie de elementos como son cajas de texto, botones, entre otros.

Los elementos que componen estos formularios, JavaScript los considera como objetos del navegador, incluso la misma página es considerada como un objeto y por ende, también poseen un conjunto de propiedades y métodos a los cuales tenemos acceso, logrando finalmente obtener aplicaciones mucho más potenciales.

Eventos

Un evento es un suceso desencadenado por una acción, ya sea del usuario o del navegador. Ejemplos de eventos tenemos: presionar una tecla, pasar el puntero del mouse sobre algún objeto de la página, la carga de la página, etc.

Algunos de los objetos de la página tienen la posibilidad de detectar la ocurrencia de un evento los cuales podemos aprovechar para asociarlos a una función para que haga algo predeterminado por nosotros.

Algunos de los eventos manejados son:

Evento	Descripción	Objeto
onLoad()	Es llamado al cargar la página	Window
onUnload()	Es llamado al cerrar una página	Window
onClick()	Es llamado al pulsar con el botón izquierdo del mouse.	Link, button, radio, checkbox
onDblClick()	Es llamado al hacer doble click con el botón izquierdo del mouse	Link, button, radio, checkbox

Algo importante a destacar es que a pesar de que todos los eventos son iguales tanto en Netscape como en Explorer, no todos los eventos se aplican a los mismos objetos ya que no se contemplan el mismo número de objetos en los navegadores.

♦ Hojas de estilo

Introducción

Las hojas de estilo vienen del término CSS (Cascading StyleSheet, Hojas de estilo en cascada). Las hojas de estilo son un simple mecanismo que permiten al diseñador de páginas web hacer una unión de estilo en la que se incluyen fuentes, tamaño, color, espacios, etc., las que posteriormente puede fácilmente ser aplicadas a un documento HTML.

Definición de estilos

Las hojas de estilo están compuestas de reglas conformadas de un selector (usualmente un elemento HTML como ***body***, ***p***, ***em***) y el estilo a ser aplicado al selector.

```
Selector[:operador]{
 Atributo 1: valor 1;
 Atributo 2: valor 2;
 ……
 Atributo n-1: valor n-1;
Atributo n: valor n;
}
```

Ejemplo:

```
a:visited {
font-family: Arial, Helvetica, sans-serif;
        font-size: 11px;
        color: #000000;
        text-decoration: none;
}
```

La regla anterior indica que todos los vínculos que hayan sido visitados tendrán la fuente indicada, de tamaño 11, y en color negro.

Las hojas de estilo pueden emplearse en un documento HTML de tres formas:

1. **Interna**
   Usando los atributos ***style, class, ID,*** dentro de alguna etiqueta como ***body, H1, P,*** etc.

   ```
   <p style=”color: red; font-family:’Arial’; ”>
   Este párrafo usa estilos en rojo con la fuente Arial.
   </p>
   ```

```
<p class=”colorverde”>
  Párrafo en color verde
</p>
```

2. **Empotrada**

   Usando la etiqueta ***style*** dentro de la sección de encabezado.

```
<head>
 <style type=”text/css”>
   p.colorverde{color: green;}
   #frio {color:blue;}
   .colorrojo{color:red;}
 </style>
</head>
```

3. **Vinculada**

   También conocida como hoja de estilo externa. Permite crear estilos que se pueden aplicar a un sitio completo mediante un fichero externo cuya extensión será “***.css”***.

   Esta forma debe de aplicarse en la sección <**head**> y <**/head**> con el enlace correspondiente.

```
<link rel=StyleSheet href=”estilo.css” type=”text/css”>
@import url(http://www.pagina.com/estilo.css);
```

Capas

Otro concepto importante en el uso de hojas de estilo son las capas. Estos son elementos HTML de tipo contenedor que nos permiten crear efectos de visibilidad, tales como ocultar o mostrar objetos de un documento web y esto se consigue modificando cada una de las propiedades previamente definidas en la capa a través de JavaScript proporcionándole a la página un mayor dinamismo.

El empleo de capas se logra mediante el manejo de la etiqueta ***div*** con los atributos “***id***” y “***style***”:

```
<div id=”identificador” style=”position:,visibility:…”>
```

♦ Hypertext Preprocessor (PHP)

Introducción

PHP (Hypertext Preprocessor) es un lenguaje interpretado de alto nivel incluido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl.
PHP permite a los creadores de páginas web, escribir páginas dinámicas de una manera rápida y fácil. El código PHP se incluye entre etiquetas especiales de comienzo y fin <?php, ?> que nos permitirá entrar y salir del modo PHP.

```
<html>
  <head>
   <title> Ejemplo de PHP </title>
  </head>
  <body>
    <?php
        echo “Esto es código PHP”;
    ?>
  </body>
</html>
```

A diferencia de javascript que es ejecutado en la máquina cliente, PHP se ejecuta en un servidor web, por lo que el cliente sólo recibe el resultado de ejecutar el código PHP en tal servidor sin ninguna posibilidad de determinar que código ha producido el resultado recibido.
Con PHP podemos hacer cosas como procesar información de formularios, generar páginas con contenido dinámico, etc., pero la característica más potente de PHP es su soporte para trabajar con una gran cantidad de bases de datos además de soportar el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados.

Sintaxis básica

Hay cuatro formas de salir de HTML y entrar en el modo de código PHP:

1. **<?** echo “Esta es la más simple”; **?>**

2. **<?php** echo “Esta es otra forma de entrar en modo PHP”; **?>**

3. **<script language=”php”>**
   echo “Esta es una tercera forma”;
   </script>

4. **<%** echo “puedes también utilizar etiquetas tipo ASP”; **%>**

Las instrucciones se separan terminando cada sentencia con un punto y coma (;).

Comentarios

La forma de establecer comentarios en PHP es de la siguiente forma:

1. Las líneas //, establecen un comentario a partir de su aparición hasta el final de la línea.

2. /* …….*/, esta forma es utilizada para establecer comentarios por bloques

Ejemplos;

```
<?php

echo “esto es una prueba”; //Este es un comentario para una línea

/*
 echo “esta forma establece comentarios en bloque”;
 echo “Hola a todos”;
*/

?>
```

Tipos de datos en PHP

Una variable es una estructura que contiene datos que pueden ser recuperados y modificados durante la ejecución de un programa.

PHP al igual que cualquier otro lenguaje de programación soporta los siguientes tipos de datos:

1. Array
2. Número en punto flotante
3. Entero
4. Objeto
5. Cadena

Los tipos de datos son utilizados para saber cuál es el contexto sobre el cual están siendo utilizadas ciertas variables. En PHP el usuario no necesita especificar o declarar el tipo de dato que una variable puede albergar y en su lugar el tipo de una variable se determina por el contexto en la que está siendo utilizada, es decir, una variable en PHP puede contener distintos tipos de datos en distintas situaciones en un mismo programa.

En PHP las variables se representan con un signo de dólar seguido por un nombre de la variable.

Ejemplos:
$a, $b, $hola, $a1,…

Operadores

Existen distintos tipos de operadores, de los que destacan: aritméticos, de asignación, de comparación, lógico, entre otros.

Los operadores aritméticos básicos son:

Ejemplo	Nombre	Resultado
$a + $b	adición	Suma de $a y $b
$a - $b	Substracción	Resta de $a y $b
$a * $b	Multiplicación	Producto de $a y $b
$a / $b	División	Cociente de $a entre $b
$a % $b	Módulo	Resto de $a dividido entre $b

El operador básico de asignación es “=”.

Ejemplo:

$a= ($b + 4) / 5;

El operando de la izquierda toma el valor de la expresión de la derecha.

Algunos ejemplos de operadores de comparación son:

Ejemplo	Nombre	Resultado
$a == $b	Igualdad	Cierto se $a es igual a $b
$a != $b	Desigualdad	Cierto si $a es diferente a $b
$a < $b	Menor que	Cierto si $a es menor a $b
$a > $b	Mayor que	Cierto si $a es mayor a $b

Los operadores lógicos contemplan al And, Or, Xor, etc.

Sentencias de control

Por mencionar algunos, una sentencia puede ser una asignación, una llamada a una función, un bucle, etc. y normalmente terminan con un punto y coma.

Las sentencias pueden ser agrupadas mediante un par de llaves {}, y un conjunto de sentencias a su vez también es considerado como una sentencia.

Las sentencias de control manejadas en PHP son similares a los empleados en otros lenguajes como C o Java, de las que podemos mencionar:

if – permite la ejecución condicional de fragmentos de código.

```
<?php

  if(exp) sentencia

?>
```

Ejemplo

```
if($a < $b) echo "$a es menor que $b";
```

else – permite ejecutar una sentencia si cierta condición es cumplida y ejecutar otra si la condición no se cumple.

```
<?php
 if(exp){
  sentencia1
 } else {
   sentencia2
 }
?>
```

Ejemplo:

```
if($a < $b){
 echo "$a es menor que $b";
}else{
 Echo "$a es mayor que $b";
}
```

while – Permite ejecutar la(s) sentencia(s) anidada(s) repetidamente mientras la expresión while se evalue en TRUE.

```
<?php

while( exp){
 sentencia(s)
}

?>
```

Ejemplo

```
$i=1;
while($i<=10){
 echo $i;
 $i=$i+1;
}
```

for – Su sintaxis es la siguiente:

```
<?php

for(exp1; exp2, exp3){
 sentencia (s)
}
?>
```

exp1 es evaluada incondicionalmente al iniciarse el bucle ***for***, posteriormente *exp2* es evaluada al comienzo de cada iteración, si *exp2* es evaluada a TRUE el bucle continua y las sentencias anidadas son ejecutadas; en caso contrario el bucle finaliza su ejecución.
Al final de cada iteración es evaluada *exp3*.

Ejemplo:

```
for( $i = 0; $i < 10; $i++){
  echo “$i”;
}
```

Operaciones básicas con ficheros

Las operaciones de entrada/salida tienen mucha importancia en cualquier lenguaje de programación, ya que generalmente, no tiene sentido una aplicación que no sea capaz de mostrar resultados en pantalla, leer o escribir en un fichero o manipular una base de datos.
El adecuado conocimiento de la manipulación de ficheros es uno de los complementos fundamentales para el diseño de páginas web.
Un fichero es una estructura de datos que se guarda en memoria secundaria y que permite almacenar información. Los ficheros pueden almacenar caracteres, valores numéricos o cualquier otro tipo de información.
Siempre que se vaya a manipular cualquier fichero, han de realizarse las operaciones según el siguiente esquema:

1. Abrir el fichero.
2. Realizar operaciones específicas sobre los datos.
3. Cerrar el fichero.

¿Cómo se abre un fichero?

PHP dispone de la función fopen(), que permite realizar esta operación. La sintaxis de esta función es:
Int fopen(string fichero, string modo [, int ruta_include]);

- El parámetro fichero dispone de varias opciones:

  1. Si empieza por http://..., abre una conexión HTTP al servidor indicado y devuelve un puntero al principio del fichero de página de respuesta.
  2. Si empieza por ftp://.., abre una conexión ftp al servidor indicado y se obtiene un puntero al fichero pedido.
  3. Si comienza por php://stdin, php://stdout o php://stderr, se abrirá el canal correspondiente a la entrada, salida o error.
  4. Si empieza por cualquier otra cadena de caracteres, se intentará abrir el fichero indicado y se devuelve el manejador del fichero.

- El parámetro modo determina la forma de acceso en que se abre el fichero. Un fichero puede ser abierto con diferentes modos: lectura, escritura, lectura y escritura, añadir información,...

  1. r Sólo lectura.
  2. r+ Lectura y escritura.
  3. w Sólo escritura. Borra el contenido anterior.
  4. w+ Lectura y escritura. Borra el contenido anterior.
  5. a Sólo escritura. Conserva el contenido anterior.
  6. a+ Lectura y escritura. Conserva el contenido anterior.

Todas las opciones en las que se abre el fichero en modo de escritura tienen un comportamiento común, de forma que si el fichero existe, lo abre en el modo especificado en caso contrario tratará de ser creado.

- El parámetro *ruta_include* es opcional y sólo puede tomar el valor 1. Se utiliza para indicar al sistema que realice la búsqueda del archivo en los directorios especificados en la directiva *include_path* del fichero *php.ini*.

La función *fopen(),* devuelve un número entero que representa el manejador sobre el archivo abierto a través del cual se realizan las operaciones de acceso a tal fichero.

¿Cómo se recorre un fichero?

Esta es una operación muy sencilla que consiste en ir leyendo los caracteres que se encuentran almacenados en el fichero hasta llegar a una marca especial denominada EOF (End Of File, fín del fichero) que determina el final del mismo.
Los ficheros tienen un cursor interno que indica la posición actual a la que se tiene acceso. Cada vez que se realiza una operación de lectura, dicho cursor se avanza una posición para poder realizar la lectura del siguiente elemento. Cuando se abre el fichero, el cursor se sitúa delante del primer elemento, de modo que la primera posición de lectura recupere el primer elemento. Para comprobar si ha alcanzado el final del fichero, PHP dispone de la función *feof()*. Esta comprueba si la posición actual es la marca de final del fichero.

Su sintaxis es:

Int feof(int fichero)

El parámetro fichero, se corresponde con el manejador del fichero y debe ser el valor devuelto por la función fopen().
PHP dispone de varias funciones para realizar las operaciones de lectura, las diferencias entre ellas se basan en la cantidad de bytes que leen y en su sintaxis.

- fread() Se encarga de leer un número determinado de bytes del fichero y devolverlos en una cadena de caracteres.

  String fread(int fichero, int numero_bytes);

- fgetc() Lee información del fichero caracter a caracter.

  String fgetc(int fichero);

- fgets() Lee una línea entera de un fichero.

  String fgets(int fichero, int numero_bytes);

- fscanf() Lee datos que siguen un determinado formato de entrada.

  fscanf(int fichero, string formato [, string var1,...]);

- file() Permite leer todo el contenido de un fichero y almacenarlo en una matriz.

  Array file(string nombre del fichero, [, int ruta_include]);

¿Cómo se cierra un fichero?

Cerrar es la última operación que se debe realizar al manipular un fichero. La función definida en PHP para esta operación es fclose().

Int fclose(int fichero);

Funciones matemáticas en PHP

En esta sección describiremos algunas de las funciones matemáticas con las que cuenta PHP. Estas funciones están disponibles listas para utilizarlas.

abs(number) Devuelve el valor absoluto de un número.
cos(float arg) obtiene el coseno de arg en radianes.
sin(float arg) obtiene el seno de arg en radianes.
tan(float arg) obtiene la tangente de arg en radianes.
pow(float base, float exp) expresion exponencial. Devuelve la base elevada a la potencia exp.
acos(float arg) devuelve el arco coseno de arg en radianes.
asin(float arg) devuelve el arco seno de arg en radianes.
atan(float arg) obtiene el arco tangente de arg en radianes.

Creación de imágenes con PHP

Con la ayuda de las librerías GD, PHP proporciona además de la capacidad de obtener el tamaño de imágenes PNG, JPEG y GIF la posibilidad de crearlas y manipularlas de forma dinámica.

Las funciones más importantes para la creación de imágenes en PHP son:

int imagecreate(int x_size, int y_size)

Esta función se encarga de crear una imagen de tamaño x_size (ancho) por y_size (alto).

int imageellipse(int image,…)

Dibuja una elipse sobre una imagen.

int imagepng(int image,…)

Genera una imagen png para mostrarla en el navegador o guardarla en un archivo.

int imageline(int image,…)

Dibuja una línea sobre una imagen.

La librería GD además de proporcionar estas funciones ofrece muchas otras relacionadas con la especificación de color, recorte, etc.
Otra de las ventajas importantes de este módulo es que es un software ***open source***, por lo que podemos obtenerlo directamente desde el sitio oficial www.boutell.com y aprovechar las ventajas de tal librería.

Bibliografía

[1] Cay S.Horstmann Gary Comell, Java 2 Fundamentos volumen 1, Prentice Hall.

[2] Douglas Bell Mike Parr, Java para Estudiantes Tercera Edición, Pearson Educación.

[3] Deitel y Deitel, Como Programar en Java, Prentice Hall.

[4] Abraham Gutiérrez Ginés Bravo, PHP 4 a través de ejemplos, Alfa omega Ra-Ma.

[5] Martín Ramos Monso, Programación PHP, Sitios Web Dinámicos e Interactivos, MP Ediciones.

[6] Brent Heslop Larry Budnick, Publicar con HTML en Internet, International Thomson Editores.

[7] Gonzalo Pajares, Jesús M. de la Cruz, José M. Molina, Juan Cuadrado, Alejandro López, Imágenes Digitales Procesamiento Práctico con Java, Alfa Omega Ra-Ma.

[8] J. R. Parker, Algorithms for Image Processing and Computer Vision, Wiley Computer Publishing.

[9] Hearn Baker, Computer Graphics with OpenGL, Prentice Hall.

[10] Dr. Sidnie Feit, TCP/IP, Arquitectura, protocolo e implementación con IPv6 y seguridad de IP, Mc Graw Hill.

[11] Uyless Back, Redes de Computadores, Protocolos, normas e interfaces, Computec Ra-Ma.

[12] Álvaro Gómez Vieites, Manuel Veloso Espiñeira, Redes de Computadoras e Internet. Funcionamiento, servicios ofercidos y alternativas de conexión, Alfa omega Ra-Ma.

[13] Duglas E. Comer, Internetworking with TCP/IP, Principles, Protocols and Architectures, Prentice Hall.

[14] Kris Jamsa, Ken Cope, Programación en Internet, Mc Graw Hill.

[15] Erick J. Braude, Ingeniería de Software una perspectiva orientado a objetos, Alfa omega.

[16] Pressman Roger S., Ingeniería de Software un enfoque práctico, Mc Graw Hill.

[17] Ian Sommerville, Ingeniería de Software, Pearson Educación

[18] Keith Weskamp, Loren Heiny, Gráficas poderosas con turbo C++, megabyte noriega editores.

[19] María del Rosario Muños Hernández, Sistema para el manejo de Graficado bajo ambientes unix. Tes396 2001-1

[20] Marco Antonio Muños Pérez, Sistema Web de Herramientas para la inteligencia de negocios, Tes952 2006-1