ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN GENERADOR DE CÓDIGO HTML"

PROYECTO DEL TÓPICO DE GRADUACIÓN
"ANÁLISIS Y DISEÑO DE APLICACIONES ORIENTADAS A OBJETO"

Previo a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

Presentado por:

NYREE TORRES GALLARDO
RAÚL PÉREZ TIXE
FREDI ERAS LÓPEZ

GUAYAQUIL - ECUADOR
1997

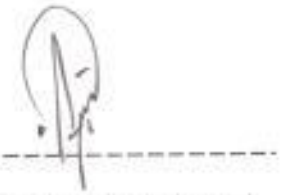
AGRADECIMIENTO

A Dios, a nuestros padres, a nuestra familia en general, a nuestros profesores y amigos, por haber colaborado de una u otra forma en nuestra formación.

Un agradecimiento especial a Karina, Mario, Nelson y Carlos, así como al Ing. Johnny Macías, profesor de nuestro tópico de graduación.

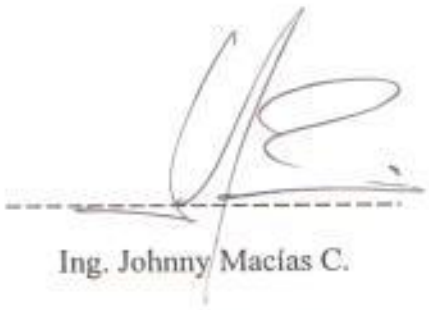
DEDICATORIA

A NUESTRA FAMILA



Ing. Armando Altamirano

Presidente



Ing. Johnny Macías C.

Profesor de Tópico

Dr. Enrique Peláez

Miembro del Tribunal

Ing. Carlos Valero

Miembro del Tribunal

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en este proyecto, nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL."

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)

Mélida Nyree Torres Gallardo

Raúl Felipe Pérez Tixe

Fredi Gonzalo Eras López

RESUMEN

Durante años anteriores el gran problema para el usuario común del *World Wide Web* ha sido la creación de sus propios documentos *HTML* pueden ser creados usando cualquier editor de texto (*vi* o *Emacs* para máquinas *Unix*). Sin embargo el manejo de estos editores requiere el conocimiento del lenguaje *HTML*.

Hoy en día existen en el mercado muchos editores y generadores de código *HTML*, algunos de los cuales inclusive presentan una interface *WYSIWYG* (lo que ve es lo que obtiene) con lo cual la persona que está creando su documento *HTML* tendrá una idea de cómo se verá su documento al examinarlo con un *browser*. Debido a que no existe aún un estándar , *cada browser* interpreta el código *HTML* siguiendo sus propias reglas. Por eso es recomendable abrir el documento *HTML* con diferentes *browsers* y comprobar que por lo menos las características más importantes del documento se conserven .

Nuestro grupo de tópico presenta la propuesta del análisis, diseño e implementación de un generador básico de código *HTML*. El objetivo no es el de desarrollar un generador de código que compita con los existentes en el mercado. Nuestro interés se centra en utilizar una metodología diferente en el análisis y el

diseño, así como en conseguir que el usuario de este generador de código no precise de ningún conocimiento del lenguaje HTML para crear sus páginas.

Esto lo conseguimos utilizando cajas de diálogo, en las cuales el usuario tan sólo elige opciones o ingresa información relacionada con la apariencia que desee darle a la página.

ÍNDICE GENERAL

INTRODUCCIÓN

El objetivo principal de nuestro tópico es aplicar el análisis y diseño orientado a objetos en el desarrollo de aplicaciones, para lo cual se ha escogido el lenguaje *Visual C++*.

El objetivo planteado por nuestro grupo fue diseñar e implementar un generador de código *HTML* que tome como base los estándares de *HTML 3.0.*.

Para explicar en forma detallada el desarrollo del generador de código *HTML*, se ha dividido este documento en tres capítulos bien diferenciados y clasificados de la siguiente manera:

CAPÍTULO I

En este capítulo procedemos a explicar las características generales del generador de código *HTML*, tales como sus antecedentes, objetivos, funcionalidad y metodología aplicada en la implementación.

CAPÍTULO II

En este capítulo se da una información completa del funcionamiento del generador de código *HTML*, así como los requisitos de hardware y software que este generador requiere para su normal funcionamiento, además se da una explicación de su manejo.

CAPÍTULO III

En este capítulo se detallan el análisis, diseño e implementación del generador de documentos *HTML*; proporcionando de esta manera los diagramas de estructura de objetos y comportamiento de objetos del sistema. De la misma manera se hace una amplia exposición de todas las clases con sus respectivas estructuras de datos y las operaciones que manipulan dichos datos.

CAPÍTULO I

CARACTERÍSTICAS DEL GENERADOR DE CÓDIGO HTML

1.1. ANTECEDENTES

En épocas pasadas pocas eran las personas y empresas que tenían acceso a *Internet*, no sólo por el desconocimiento del mismo, sino por limitaciones de *hardware* y *software*. Hoy en día, cuando la industria ha crecido notablemente, el mercado oferta todo tipo de equipos y software a precios accesibles para el público en general. Esto, sumado al desarrollo en las comunicaciones y a la necesidad del ser humano de romper las barreras del conocimiento, ha ocasionado la generación y crecimiento de empresas que ofertan bienes y servicios ante una demanda creciente.

El tema de moda desde hace tiempo es *INTERNET*. Todos quieren acceder a este medio, debido a que presenta información mundial actualizada. Por esta

razón diariamente aparecen en el mercado nuevos programas, muchos de ellos gratis en Internet, que permiten el desarrollo de documentos *HTML*, por medio de los cuales se puede exponer ante el mundo información ya sea de carácter personal , social, comercial, etc..

Durante años anteriores el gran problema para el usuario común *del World Wide Web* ha sido la creación de sus propios documentos *HTML* los cuales se encuentran en un formato de texto plano (*ASCII*) y pueden ser creados usando cualquier editor de texto (*vi* o *Emacs* para máquinas *Unix*).

Con el devenir de los años este problema ha empezado ha disminuir debido al surgimiento de una gran variedad de *browsers* que van desde aquellos que incluyen rudimentarios editores *HTML* en un ambiente *WYSIWYG* tales como WWW para *X Windows Systems* y el *Web browser* de *CERN para NeXT computers*, hasta aquellos que se aproximan en su totalidad al ambiente *WYSIWYG* como el *Hot Metal* para *Sun Sparc Stations* , el Editor *HTML* para *Macintosh* o el *Nestscape Composer* .

En la actualidad se cuenta con editores y *wizards* (asistentes) de *HTML* que brindan una interface amigable al usuario. Entre los más importantes podemos mencionar a continuación:

- *Dida,*
- *HotDog,*
- *Internet Assistant for Microsoft Word,*
- *Internet Assistant for Microsoft Excel,*
- *Netscape Navigator Gold y*
- *ANT_HTML*

Sin embargo casi podemos asegurar que esta lista crecerá en poco tiempo, puesto que la industria de desarrollo de software para acceder a INTERNET, ya sea para ofertar o demandar información, está en constante crecimiento.

1.2. OBJETIVOS DEL SISTEMA

Para el desarrollo del generador de código *HTML* se han planteado los siguientes objetivos:

- Emplear en el análisis y diseño del sistema, las técnicas y metodología de orientación a objetos.
- En la fase de la implementación emplear un lenguaje de programación orientado a objetos : *Visual C ++*.
- Proporcionar una interface totalmente amigable al usuario de tal manera que éste pueda crear sus documentos *HTML* con relativa facilidad.
- Proporcionarle al usuario la capacidad de creación de varios documentos *HTML* a la vez.
- Generación del código *HTML* en base a los principios básicos del lenguaje *HTML* versión 3.0.

1.3. FUNCIONALIDAD DEL SISTEMA

El generador de código *HTML*, desde el punto de vista funcional, cumple con todos los requisitos previamente establecidos; dicha funcionalidad procedemos a explicar a continuación:

- El Sistema presenta una pantalla común de *Windows 95*, con un menú principal y una barra de herramientas con sus respectivos atributos que le permiten al usuario crear el documento *HTML*.
- La barra de herramientas incluye tanto botones estándar como botones propios del sistema; entre los botones estándar se encuentran aquellos que son utilizados para crear un documento nuevo, cerrar un documento, guardar un documento y además el botón *Acerca de*. Los botones considerados parte del Sistema son aquellos que son utilizados para darle formato al documento *HTML* entre los cuales se encuentran: *Título, Párrafo, Imagen, Ancla, Enlace, Línea, Enter, Eliminar y Propiedades*.
- Para cada botón de la barra de herramientas se encuentra asociada una ventana que permite ya sea asignar o modificar las propiedades del objeto seleccionado.
- El sistema permite la creación de múltiples documentos, esto es, el usuario puede trabajar en varios documentos sin necesidad de abrir una aplicación nueva para cada uno.

1.4. METODOLOGÍA

El análisis y diseño del sistema ha sido desarrollado en base a la metodología orientada a objetos y su implementación fue desarrollada en un lenguaje de programación orientada a objetos, específicamente Visual C++.

El sistema utiliza tanto clases definidas por el grupo de desarrollo como las propias generadas por Visual C++ tales como CObject, CDialog, CView, CDocument, entre otras. Este lenguaje ofrece un conjunto de librerías entre las cuales se utilizó la librería *MFC*, la cual permite el uso de la aplicación *MDI*.

En la barra de herramientas, así como en las opciones del menú, se presenta los objetos que el usuario puede incluir en su documento HTML, entre los cuales se encuentran: título, párrafo, imagen, línea horizontal, espaciamiento vertical (ENTER) y ancla. El usuario también puede manipular estos objetos eligiendo las opciones de Propiedades y Eliminar, presentadas también en la barra de herramientas y en las opciones del menú.

La inclusión de objetos en el documento se realiza a través de ventanas de diálogo. En estas ventana se presentan opciones para configurar los atributos del objeto de tal modo que se presente en la pantalla de la forma deseada. El

usuario tan sólo elige intuitivamente las opciones de la ventana de diálogo que lograrán el efecto esperado.

La característica de nuestro generador de código *HTML* es que ingresa el texto por medio de una ventana de diálogo y lo grafica en la vista, a diferencia de otros generadores, en los cuales el texto es ingresado directamente en la vista.

Para la fase de implementación y funcionamiento del sistema se ha requerido tanto en hardware como en software de lo siguiente:

Plataforma de Hardware:

- Computador IBM o compatible.
- Procesador Pentium, 100 MHz o superior.
- 16 Mb en RAM.
- Disco duro de 1 Gb.
- Monitor SVGA de 14" con capacidad de 256 colores o superior.

Plataforma de Software:

- Sistema operativo *Windows 95.*
- Browser: *Netscape Navigator Gold 3.0.*.

CAPÍTULO II

DISEÑO, ANÁLISIS E IMPLEMENTACIÓN DEL SISTEMA "GENERADOR DE CÓDIGO HTML"

2.1. ANÁLISIS DE LA ESTRUCTURA DE OBJETOS DEL SISTEMA

2.1.1. DESCRIPCIÓN DE LOS TIPOS DE OBJETOS Y OBJETOS DETERMINADOS

Clase: CEditorDoc
Clase base: Cdocument

Por cada documento abierto en nuestra aplicación MDI, se crea una instancia de esta clase. CEditorDoc almacena en una lista los punteros hacia los objetos *HTML* insertados en la vista asociada al documento.

Clase: CEditorView
Clase base: CScrollView

Al crearse un documento en la aplicación *MDI*, automáticamente se le asocia una vista, por medio de la cual se puede presentar información al usuario del sistema.

Clase: CElemento

Clase base: CObject

Esta clase contiene los atributos de los diferentes elementos *HTML* (heading, link, anchor, etc.). Existen algunos atributos comunes a todos los elementos y otros específicos.

Clase: ancla
Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto ancla.

Clase: titulo
Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto titulo.

Clase: parrafo
Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto párrafo.

Clase: enlace
Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto enlace.

Clase: linea
Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto linea.

Clase: imagen

Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto imagen.

Clase: pagina
Clase base: CDialog

Esta clase tiene asociada una ventana de diálogo que sirve de interface entre el sistema y el usuario para ingresar o modificar los atributos del objeto página.

Clase: BinTree

Esta clase contiene toda la información sobre la estructura del lenguaje html. Contiene una referencia hacia un objeto CHtml.

Clase: CLista

Esta clase se la utiliza para almacenar información de los atributos de los elementos html.

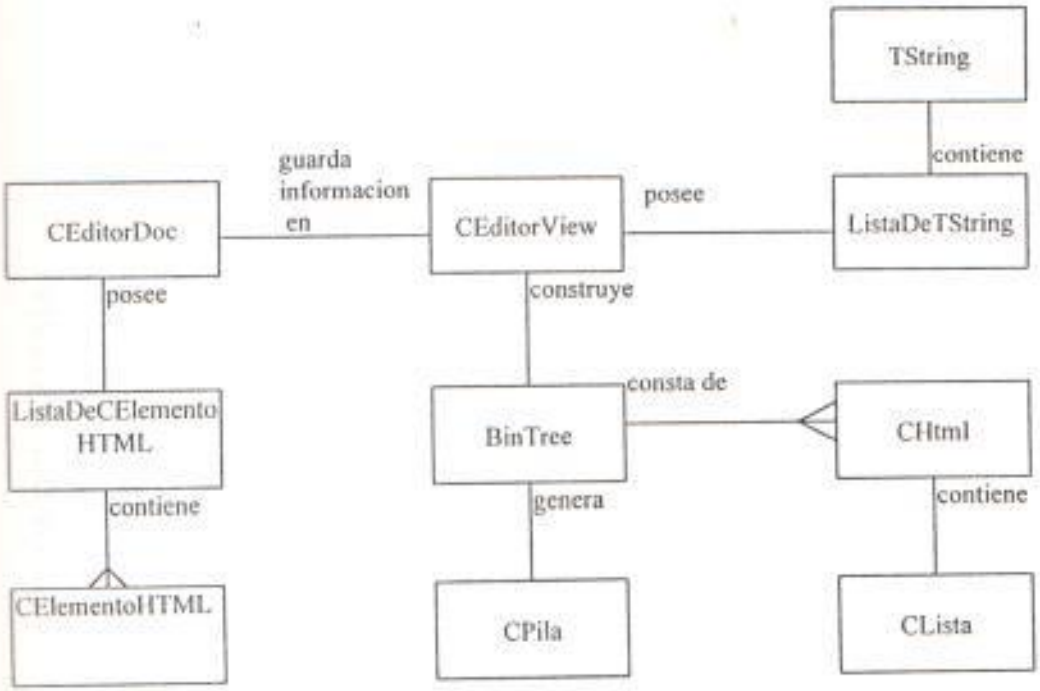
Clase: CPila

Esta clase se utiliza como un recurso para recorrer el árbol Binario (BinTree), además para almacenar el recorrido a través del árbol con el cual se genera el código en lenguaje html.

Clase: CHtml

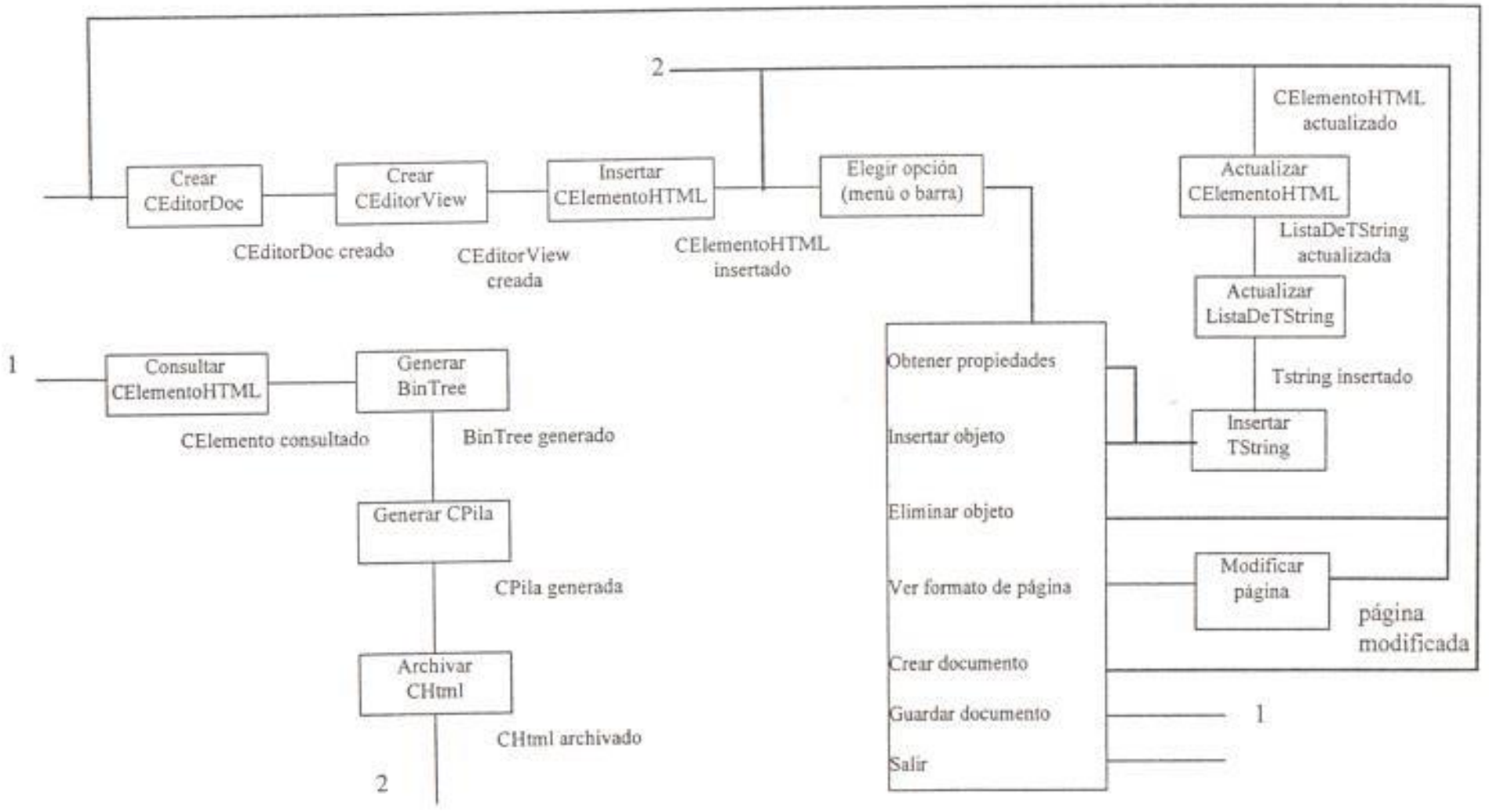
Es la clase base de los elementos html, contiene información común para los elementos html derivados de esta clase.

2.1.2. DIAGRAMA DE LA ESTRUCTURA DE OBJETOS DEL SISTEMA



2.2. ANÁLISIS DEL COMPORTAMIENTO DE OBJETOS DEL SISTEMA

2.2.1. DIAGRAMA DEL COMPORTAMIENTO DE OBJETOS DEL SISTEMA



2.3. DISEÑO DE LA ESTRUCTURA Y COMPORTAMIENTO DEL SISTEMA

2.3.1. CLASES Y SUBCLASES IMPLEMENTADAS CON SUS RESPECTIVAS OPERACIONES Y DATOS

ancla

Clase base:
CDialog

Atributos:
Públicos:
CString m_texto

Operaciones:
Públicas:
ancla( )

Protegidas:
void OnOK( )
BOOL OnInitialDialog( )
void OnCancel( )

EditorDoc

Clase Base:
CDocument

Atributos:
Privados:
CObList lista

Operaciones:
Protegidas:
CEditorDoc( )
virtual ~CEditorDoc

Públicas:
CObList* GetList( )

CEditorView

Clase Base:
CScrollView

Atributos:
Protegidos:
POSITION m_posicion
CObList * m_lista
COLORREF bordeimagen
COLORREF texto, penlace, activo, seleccionado, fondo
CString ptitulo
int YGSM, XGSM
int m_YPos, m_XPos
int n_YPos, nXPos
CString m_ElementoSeleccionadoEnToolbar;

Operaciones:
Públicas:
void ObtenerPosicion( )
void PropiedadesElemento( )
void InsertarElemento( )
virtual void OnDraw(CDC* pDC)
CEditorDoc* GetDocument( )
virtual ~CEditorView( )

Protegidas:
CEditorView( )
virtual void OnInitialUpdate( )
void OnInsertarAncla( )
void OnInsertarEnlace( )
void OnInsertarEnter( )
void OnInsertarImagen( )
void OnInsertarLinea( )
void OnInsertarParrafo( )
void OnInsertarTitulo( )
void OnEdicionEliminar( )
void OnUpdateEdicionEliminar(CCmdUI* pCmdUI)
void OnEdicionPropiedades( )

void OnUpdateEdicionPropiedades(CCmdUI* pCmdUI)
int OnCreate(LPCREATESTRUCT lpCreateStruct)
void OnLButtonDown(UINT nFlags, CPoint point)
void OnLButtonDblClk(UINT nFlags, CPoint point)
void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
void OnFormatoPagina( )

CElemento

Clase base:
CObject

Atributos:
Públicos:
int bordeimagen
CString lineatipoancho
CString tituloheading
CString letra
BOOL blink
BOOL italicas
BOOL subrayado
BOOL negritas
CString tipo
CString texto
CString enlace
int alineacion
int ancho
int alto

Operaciones:
Públicas:
CElemento(CString elemento)
virtual ~CElemento( )

enlace

Clase base:
CDialog

Atributos:
Públicos:
CObList lista
BOOL aceptar
CListBox m_enlaceancla
BOOL m_negritas
BOOL m_italicas
BOOL m_subrayado
BOOL m_blink
int m_izquierda
CString m_enlace
CString m_tamano
CString m_texto

Operaciones:
Públicas:
enlace(CWnd* pParent = NULL)
virtual ~enlace( )

Protegidas:
virtual BOOL OnInitDialog( )
void OnOk( )
virtual void OnCancel( )
void OnRadioIzquierda( )
void OnRadioCentro( )
void OnRadioDerecha( )
void OnCloseupCombo1( )
void OnButtonExaminar( )
void OnUpdateEdit1( )
void OnUpdateEditEnlace( )
void OnDblclkList1( )

imagen

Clase base:
CDialog

Atributos:
Públicos:
CObList lista
BOOL aceptar
CListBox m_imagenancla
CString m_texto
int m_ancho
CString m_enlace
int m_izquierda
int m_grosor
int m_alto

Operaciones:
Públicas:
imagen(CWnd* pParent = NULL)

Protegidas:
void OnButtonExaminar( )
OnCancel( )
void OnOk( )
void OnRadioCentro( )
void OnRadioDerecha( )
void OnRadioIzquierda( )
void OnButtonTamanooriginal( )
void OnButtonExaminar2( )
BOOL OnInitDialog( )
void OnUpdateEditImagen( )
void OnUpdateEditAlto( )
void OnUpdateEditAncho( )
void OnUpdateEditEnlace( )
void OnUpdateEditGrosor( )
void OnChangeEditAlto( )
void OnDblclkList1( )

linea

Clase Base:
CDialog

Atributos:
Públicos:
BOOL aceptar
int m_alto
int m_ancho
CString m_combo_ancho
int m_alineacion

Operaciones:
Públicas:
linea(CWnd* pParent = NULL)

Protegidas:
void OnRadio1( )
void OnRadio2( )
void OnRadio3( )
BOOL OnInitDialog( )
void OnOk( )
virtual void OnCancel( )
void OnChangeEdit1( )
void OnChangeEdit2( )
void OnUpdateEdit1( )
void OnUpdateEdit2( )
void OnCloseupComboAncho( )

listaanclas

Clase base:
CObject

Atributos:
Públicos:
CString textoancla

Operaciones:

Públicas:
listaanclas(CString texto)
virtual ~listaanclas();

pagina

Clase base:
CDialog

Atributos:
Públicos:
BOOL aceptar
COLORREF u_texto,u_enlace,u_activo,u_seleccionado,u_fondo
COLORREF m_texto,m_enlace,m_activo,m_seleccionado,m_fondo
CString m_titulo
int m_set

Operaciones:
Públicas:
pagina(CWnd* pParent = NULL)
void chequear();

Protegidas:
void OnButtonTexto( )
void OnRadioColorUsuario( )
void OnRadioColoresAplicacion( )
BOOL OnInitDialog( )
void OnPaint( )
void OnButtonEnlace( )
void OnButtonActivo( )
void OnButtonSeleccionado( )
void OnButtonFondo( )
void OnOk( )

parrafo

Clase base:
CDialog

Atributos:
Públicos:
BOOL aceptar
CString m_tipo
BOOL m_negritas
BOOL m_italicas
BOOL m_subrayado
CString m_tamano
int m_izquierda;
BOOL m_blink
CString m_texto

Operaciones:
Públicas:
parrafo(CWnd* pParent = NULL)

Protegidas:
virtual BOOL OnInitDialog( )
void OnCloseupComboTamano( )
void OnRadioCentro( )
void OnRadioDerecha( )
void OnRadioIzquierda( )
void OnOk( )
void OnCancel( )
void OnUpdateEditTexto( )

titulo

Clase base:
CDialog

Atributos:
Públicos:
CObList lista

Operaciones:
Públicas:
titulo(CWnd* pParent = NULL)
void agregarlista(LPCTSTR string)
BOOL aceptar
CString m_tipo
CListBox m_tituloancla
int m_izquierda
CString m_enlace
CString m_tamano
CString m_titulo
BOOL m_negritas
BOOL m_italicas
BOOL m_subrayado
BOOL m_blink
CString m_texto

Protegidas:
virtual BOOL OnInitDialog( )
void OnCloseupComboTitulo( )
void OnRadioCentro( )
void OnRadioDerecha( )
void OnRadioIzq( )
virtual void OnCancel( )
void OnButtonAceptar( )
void OnButtonExaminar( )
void OnChangeEdit1( )
void OnChangeEditEnlace( )
void OnUpdateEditEnlace( )
void OnUpdateEdit1( )
void OnDblclkAnclaTitulo( )

CLnodo

Atributos:
Públicos:
CString info1
CString info2
CLnodo* sig

Operaciones:
Públicas:
CLnodo()
CLnodo(CString a,CString b)
~CLnodo()

CLista

Atributos:
Privados:
CLnodo* root

Operaciones:
Públicas:
CLista()
void InsDown(CString a, CString b)
int setlist(CString a, CString b)
CString get (CString a)
CLnodo* vacia()
CString Atributos()
virtual ~CLista()

NPila

Atributos:
Públicos:
BinNode* html
NPila* sig

Operaciones:
Públicas:
NPila()
~NPila()

CPila

Atributos:
Privados:
NPila* root

Públicos:
CPila()

Operaciones::
Públicos:
int meter(BinNode* a)
BinNode* sacar(
BinNode* mirar()
BOOL vacia()
virtual ~CPila()

BinNode

Atributos:
Privados
CHtml* ele
BinNode* left
BinNode* right

BinTree

Privados
BinNode* root
BinNode* p

Operaciones:
Públicas:
BinNode* InsertD (CHtml* elemento)
BinNode* InsertDAqui(BinNode* node, CHtml* elemento)
BinNode* InsertIAqui(BinNode* node, CHtml* elemento)
BinNode* InsertNodeDerAqui (BinNode* node,BinNode* aqui)
BinNode* InsertNodeIzqAqui (BinNode* node,BinNode* aqui)
Bool InsertI (CHtml* elemento)
BinNode* Find(CHtml* elemento)
void DeleteTree(BinNode* n)
CPila* html()
int InfoN(BinNode* pnd)
BOOL EsHijoDer(BinNode* hijo,BinNode* padre)
BOOL EsHijoIzq(BinNode* hijo,BinNode* padre)

CHtml

Attributes:
Privados:
BOOL Visitado

Públicos:
BOOL m_Tag_Abierto
CString Tag
CLista* atrib
CString Texto

Operaciones:
Públicas:
CHtml()
CString TAG()
void SetText(CString txt)
void GetText(CString txt)
void SetTagAbierto()
BOOL TagAbierto()
CString codigo_html()
BOOL FueVisitado()
void SetVisita()
CString tag_fin()
virtual ~CHtml()

Protegidas:
CString SetAlign(int align)
CString SetType(int type)

Clases Derivadas de CHtml:
CTipoDocumento
CEncabezamiento
CTitulo
CCuerpo
CBreak
CParrafo
CBold
CItalicas
CFixedWidth
CUnderline

CBlink
CFonts
CHypertextAnchor
CHypertextAnchorName
CHeading
CDivisionDocumento
CImg
CHorizontalRule

2.3.2. BREVE DESCRIPCIÓN DE LAS OPERACIONES Y ATRIBUTOS DE LAS CLASES

ancla

Atributo:
CString m_texto

Esta variable es de tipo *string*, el mismo que identifica al ancla en el documento.

Operaciones:
Públicas:
ancla( )
Constructor de la clase.

Protegidas:
void OnOK( )
Esta función verifica si la caja de edición está vacía, con lo cual se controla que el usuario no cree un *tag* no válido.

BOOL OnInitialDialog( )
Función que inicializa la ventana de diálogo asociada con esta clase. Actualiza los valores relacionados con la ventana.

void OnCancel( )
Destruye la ventana de diálogo.

EditorDoc

Atributos:
Privados:
CObList lista
Se crea una instancia de la clase CObList, la misma que será utilizada para almacenar punteros a instancias de la clase Celemento.

Operaciones:
Protegidas:
CEditorDoc( )
virtual ~CEditorDoc

Son el constructor y destructor de la clase CeditDoc.

Públicas:
CObList* GetList( )

CEditorView

Atributos:
Protegidos:
POSITION m_posicion
Variable utilizada para almacenar la posición actual dentro de la lista de elementos declarada en CEditorDoc.

CObList * m_lista
Puntero hacia la lista declarada en la clase CEditorDoc.

COLORREF bordeimagen
Color que tomará el borde alrededor de la imagen.

COLORREF texto, penlace, activo, seleccionado, fondo
Colores que poseerán el texto normal, el texto para enlace, el texto de enlace activo, el texto de enlace seleccionado, y el fondo del documento.

CString ptitulo
Contiene el título del documento *HTML*.

int YGSM, XGSM
int m_YPos, m_XPos
int n_YPos, nXPos
Variables utilizdas para almacenar valores globales utilizados por diferentes funciones de esta clase.

CString m_ElementoSeleccionadoEnToolbar;
String que identifica al elemento seleccionado en el toolbar, cuyo valor pueden ser: Titulo, Párrafo, Imagen, Ancla, Linea, Enter, Enlace, Eliminar o Propiedades.

Operaciones:
Protegidas:
CEditorView( )

Constructor de esta clase.

Públicas:
void ObtenerPosicion( )
Obtiene la posición del elemento en la lista implementada en la clase CeditorDoc. Esta posición es obtenida en base a la ubicación del ratón en el área cliente una vez que se ha pulsado el botón izquierdo .

void PropiedadesElemento( )
En base a una posición determinada, despliega una caja de diálogo acorde con el tipo de elemento *HTML*, en la cual muestra las propiedades del objeto seleccionado.

void InsertarElemento( )
Inserta una elemento en la lista implementada en la clase CEditorDoc.

virtual void OnDraw(CDC* pDC)
Función que se encarga de dibujar los objetos en la vista.

CEditorDoc* GetDocument( )
Función por medio de la cual obtenemos un puntero hacia el documento asociado a la vista.

virtual ~CEditorView( )
Destructor de la clase.

Protegidas:
virtual void OnInitialUpdate( )
Esta función inicializa la vista, asignando el tamaño del área cliente.

void OnInsertarAncla( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Ancla" y llama a la función InsertarElemento( ).

void OnInsertarEnlace( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Enlace" y llama a la función InsertarElemento( ).

void OnInsertarEnter( )

Da a m_ElementoSeleccionadoEnToolbar el valor de "Enter" y llama a la función InsertarElemento( ).

void OnInsertarImagen( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Imagen" y llama a la función InsertarElemento( ).

void OnInsertarLinea( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Linea" y llama a la función InsertarElemento( ).

void OnInsertarParrafo( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Párrafo" y llama a la función InsertarElemento( ).

void OnInsertarTitulo( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Titulo" y llama a la función InsertarElemento( ).

void OnEdicionEliminar( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Eliminar" y llama a la función EliminarElemento( ).

void OnUpdateEdicionEliminar(CCmdUI* pCmdUI)
Verifica si la lista está vacia. Si lo está, entonces deshabilita el botón de Eliminar.

void OnEdicionPropiedades( )
Da a m_ElementoSeleccionadoEnToolbar el valor de "Propiedades" y llama a la función PropiedadesElemento( ).

void OnUpdateEdicionPropiedades(CCmdUI* pCmdUI)
Verifica si la lista está vacia. Si lo está, entonces deshabilita el botón de Eliminar.

int OnCreate(LPCREATESTRUCT lpCreateStruct)

void OnLButtonDown(UINT nFlags, CPoint point)
Llama a la función ObtenerPosicion( ) y luego a la función OnDraw( ).

void OnLButtonDblClk(UINT nFlags, CPoint point)
Llama a la función ObtenerPosicion( ) y luego a la función PropiedadesElemento( ).

void OnFormatoPagina( )
Llama a una ventana de diálogo donde se puede elegir el color que tendrá el texto normal, el utilizado para enlace activo, seleccionado, no seleccionado, y el color del fondo de la vista.

CElemento

Públicas:
int bordeimagen
Variable que almacena el ancho del borde que rodea a la imagen.

CString lineatipoancho
Variable que almacena la referencia utilizada para fijar el ancho de la linea. Puede ser : "pixels" o "% de ventanta".

CString tituloheading
Variable que almacena el tipo de heading que un titulo puede tener. Puede ser cualquiera de los siguiientes: "Heading1", "Heading2", "Heading3", "Heading4", "Heading5", "Heading6"..

CString letra
Variable que indica el tamaño de la lettra seleccionado para el texto. Los valores que puede almacenar son los siguientes: +3,+2,+1,0+-1-2.

BOOL blink
Variable que indica si el texto tiene la propiedad de parpadeo.

BOOL italicas
Variable que indica si el texto tiene la propiedad de itálicas.

BOOL subrayado
Variable que indica si el texto tiene la propiedad de subrayado.

BOOL negritas
Variable que almacena si el texto tiene la propiedad de negritas.

CString tipo

Variable que almacena el tipo de elemento *HTML*.

CString texto
Variable que almacena texto.

CString enlace
Variable que almacena el path del documento con el cual se creará el enlace.

int alineacion
Determina la alineación del elemento *HTML*. Cero significa alineación a la izquierda, uno, alineación en el centro y dos alineación a la derecha.

int ancho
Variable que contiene el ancho de un elemento *HTML*.

int alto
Variable que contiene el alto de un elemento.

Operaciones:
Públicas:
CElemento(CString elemento)
virtual ~CElemento( )
Constructor y destructor de la clase.

enlace

Atributos:
Públicos:
CObList lista
Se crea una instancia de CObList, necesaria para ingresar los elementos al list-box.

BOOL aceptar
Booleana que indica si al salir de la ventana de diálogo se presionó Enter o Cancelar.

CListBox m_enlaceancla
Variable tipo control de CListBox.

CString letra
Variable que indica el tamaño de la letra seleccionado para el texto. Los valores que puede almacenar son los siguientes: +3,+2,+1,0+-1-2.

BOOL blink
Variable que indica si el texto tiene la propiedad de parpadeo.

BOOL italicas
Variable que indica si el texto tiene la propiedad de itálicas.

BOOL subrayado
Variable que indica si el texto tiene la propiedad de subrayado.

BOOL negritas
Variable que indica si el texto tiene la propiedad de negritas.

CString tipo
Variable que almacena el tipo de elemento *HTML*.

CString texto
Variable que almacena texto.

CString enlace
Variable que almacena el path del documento con el cual se creará el enlace.

int alineacion
Determina la alineación del elemento *HTML*. Cero significa alineación a la izquierda, uno, alineación en el centro y dos alineación a la derecha.

Operaciones:
Públicas:
enlace(CWnd* pParent = NULL)
Constructor de la clase.

Protegidas:
void OnButtonExaminar( )

Llama a una ventana de diálogo modal del fichero de archivos, en la cual se puede seleccionar el archivo con el cual se quiere crear el enlace.

void OnDblclkList1( )
Copia el elemento seleccionado en el list-box a la caja de edición que almacena el path del archivo al cual se hará el enlace.

imagen

Atributos:
Públicas:
CObList lista
Se crea una instancia de CObList, necesaria para ingresar los elementos al list-box.

BOOL aceptar
Booleana que indica si al salir de la ventana de diálogo se presionó Enter o Cancelar.

CListBox m_imagenancla
Variable tipo control de CListBox.

CString texto
Variable que almacena texto.

CString enlace
Variable que almacena el path del documento con el cual se creará el enlace.

int alineacion
Determina la alineación del elemento *HTML*. Cero significa alineación a la izquierda, uno, alineación en el centro y dos alineación a la derecha.

int m_grosor
Variable entera que almacena el grosor del borde alrededor de la imagen.

int m_alto
Variable entera que almacena el alto de la imagen.

int m_ancho
Variable entera que almacena el ancho de la imagen.

Operaciones:
Públicas:
imagen(CWnd* pParent = NULL)
Constructor de la clase.

Protegidas:

void OnButtonExaminar( )
Llama a una ventana de diálogo modal del fichero de archivos, en la cual se puede seleccionar el archivo que contiene la imagen que se desea insertar.

void OnButtonExaminar2( )
Llama a una ventana de diálogo modal del fichero de archivos, en la cual se puede seleccionar el archivo con el cual se quiere crear el enlace.

void OnDblclkList1( )
Copia el elemento seleccionado en el list-box a la caja de edición que almacena el path del archivo al cual se hará el enlace.

linea

Atributos:
Públicos:
BOOL aceptar
Booleana que indica si al salir de la ventana de diálogo se presionó Enter o Cancelar.

int m_alto
Variable entera que almacena el alto de la linea.

int m_ancho
Variable entera que almacena el ancho de la linea.

CString m_combo_ancho
Variable que indica la referencia utilizada para determinar el ancho de la linea. Puede ser: "pixels" o "% de ventana".

int m_alineacion
Determina la alineación del elemento *HTML*. Cero significa alineación a la izquierda, uno, alineación en el centro y dos alineación a la derecha.

Operaciones:
Públicas:
linea(CWnd* pParent = NULL)
Constructor de la clase.

Protegidas:

BOOL OnInitDialog( )
Función de inicialización de la ventana de diálogo asociada a esta clase.

void OnChangeEdit1( )
Actualiza el control edit1.

void OnChangeEdit2( )
Actualiza el control edit2.

listaanclas

Atributos:
Públicos:
CString textoancla
Variable que almacena el texto del item de un list-box.

Operaciones:
Públicas:
listaanclas(CString texto)
virtual ~listaanclas();
Constructor y destructor de la clase.

pagina

Atributos:
Públicos:
BOOL aceptar
Variable que indica si al salir de la ventana de diálogo se presionó Enter o Cancel.

COLORREF u_texto,u_enlace,u_activo,u_seleccionado,u_fondo
Variables que almacenan los colores activos para el texto normal, texto de enlace, texto de enlace seleccionado, texto de enlace activo y fondo de pantalla.

COLORREF
m_texto,m_enlace,m_activo,m_seleccionado,m_fondo
Variables que almacenan los colores seleccionados por el usuario para el texto normal, texto de enlace, texto de enlace seleccionado, texto de enlace activo y fondo de pantalla.

CString m_titulo
Variable que contiene el titulo del documento *HTML*.

int m_set
Variable utilizada para indicar al usuario si los colores activos son los personalizados o los del sistema.

Operaciones:
Públicas:

pagina(CWnd* pParent = NULL)
Constructor de la clase.

void chequear();
Examina si los colores activos son iguales a los del sistema. Da valores a la variable m_set.

Protegidas:
void OnButtonTexto( )
Crea una ventana modal en la cual el usuario puede elegir el color del texto.

void OnRadioColorUsuario( )
Asigna como activos los colores seleccionados por el usuario.

void OnRadioColoresAplicacion( )
Asigna como activos los colores del sistema.

BOOL OnInitDialog( )
Inicializa la ventana de diálogo asociada a la clase.

void OnPaint( )
Dibuja en la ventana de diálogo.

void OnButtonEnlace( )
Crea una ventana modal en la cual el usuario puede elegir el color del enlace.

void OnButtonActivo( )
Crea una ventana modal en la cual el usuario puede elegir el color del texto del enlace activo.

void OnButtonSeleccionado( )
Crea una ventana modal en la cual el usuario puede elegir el color del texto del enlace seleccionado.

void OnButtonFondo( )
Crea una ventana modal en la cual el usuario puede elegir el color del fondo de la pantalla.

parrafo

Atributos:

Públicos:
BOOL aceptar
Booleana que indica si al salir de la ventana de diálogo se presionó Enter o Cancelar.

CString letra
Variable que indica el tamaño de la letra seleccionado para el texto. Los valores que puede almacenar son los siguientes: +3,+2,+1,0+-1-2.

BOOL blink
Variable que indica si el texto tiene la propiedad de parpadeo.

BOOL italicas
Variable que indica si el texto tiene la propiedad de itálicas.

BOOL subrayado
Variable que indica si el texto tiene la propiedad de subrayado.

BOOL negritas
Variable que indica si el texto tiene la propiedad de negritas.

CString tipo
Variable que almacena el tipo de elemento *HTML*.

CString texto
Variable que almacena texto.

CString enlace
Variable que almacena el path del documento con el cual se creará el enlace.

int alineacion
Determina la alineación del elemento *HTML*. Cero significa alineación a la izquierda, uno alineación en el centro y dos alineación a la derecha.

Operaciones:
Públicas:

parrafo(CWnd* pParent = NULL)
Constructor de la clase.

Protegidas:
virtual BOOL OnInitDialog( )
Inicializa la ventana de diálogo asociada a esta clase.

titulo

Atributos:

Públicos:
CObList lista
Instancia de la clase CObList.

CListBox m_tituloancla
Variable tipo control del list-box.

CString m_titulo
Variable que indica que tipo de heading se escogió.

BOOL aceptar
Booleana que indica si al salir de la ventana de diálogo se presionó Enter o Cancelar.

CString letra
Variable que indica el tamaño de la letra seleccionado para el texto. Los valores que puede almacenar son los siguientes: +3,+2,+1,0+-1-2.

BOOL blink
Variable que indica si el texto tiene la propiedad de parpadeo.

BOOL italicas
Variable que indica si el texto tiene la propiedad de itálicas.

BOOL subrayado
Variable que indica si el texto tiene la propiedad de subrayado.

BOOL negritas
Variable que indica si el texto tiene la propiedad de negritas.

CString tipo
Variable que almacena el tipo de elemento *HTML*.

CString texto
Variable que almacena texto.

CString enlace
Variable que almacena el path del documento con el cual se creará el enlace.

int alineacion
Determina la alineación del elemento *HTML*. Cero significa alineación a la izquierda, uno, alineación en el centro y dos alineación a la derecha.

Operaciones:

Públicas:
titulo(CWnd* pParent = NULL)
Constructor de la clase.

void agregarlista(LPCTSTR string)
Agrega un elemento a la lista en el list-box de anclas.

Protegidas:
virtual BOOL OnInitDialog( )
Función que inicializa la ventana de diálogo asociada a la clase.

void OnButtonExaminar( )
Función que crea una ventana modal en la cual el usuario puede elegir el archivo con el cual quiere crear el enlace.

void OnDblclkAnclaTitulo( )
Copia el item seleccionado en el list-box en la caja de edición que almacena el path del archivo con el cual se quiere crear el enlace.

CLnodo

Atributos:
Públicos:

CLnodo

Atributos:
Públicos:
CString info1
Es el Tipo de atributo del elemento
CString info2
El el valor del atributo asociado al elemento
CLnodo* sig
Permite realizar el enlace entre instancias de CLnodo (nodos de la lista)

Operaciones:
Públicos:
CLnodo()
El constructor de la clase.
CLnodo(CString a,CString b)
Es el constructor de la clase explícito
~CLnodo()
Es el destructor de la clase.

CLista

Atributos:
Privados:
CLnodo* root
El puntero a la lista

Públicos:

CLista()

void InsDown(CString a, CString b)
Permite ingresar un nuevo CLnodo en la lista

int setlist(CString a, CString b)
Permite asignar los valores de los atributos

CString get (CString a)

Devuelve el valor del atributo solicitado

CLnodo* vacia()
A través de la cual consultamos el estado de la lista.

CString Atributos()
Obtiene el texto html de los atributos, de los elementos html.

virtual ~CLista()
Permite realizar la eliminación de la lista.

NPila

Atributos:
Públicos:
BinNode* html
El un puntero hacia los elemento CHtml

NPila* sig
Es el puntero que permite conectar los CNodos de la pila (enlace con el siguiente nodo).

NPila()
El es constructor del nodo de la pila.

~NPila()
Es el destructor para el nodo de la pila.

CPila

Atributos:

Privados:
NPila* root
Es el puntero hacia la pila

Públicos:
constructor

CPila()
Es el constructor de la clase pila.

Operaciones:
Públicos:
BOOL meter(BinNode* a)
Permite ingresar un nuevo elemento BinNode en la pila.

BinNode* sacar()
Extrae el elemento superior de la pila.

BinNode* mirar()
Permite obtener el elemento de la cima de la pila sin extraerlo.

BinNode* cima()
Devuelve el valor del elemento cima de la pila.

BOOL vacia()
Permite examinar si la pila está vacia.

virtual ~CPila()
Permite destruir la pila.

BinNode

Atributos:
Públicos:
CHtml* ele
Contiene el puntero hacia el elemento Html (objeto Chtml).
BinNode* left
Es el puntero hacia el nodo izquierdo del árbol.
BinNode* right
Es el puntero hacia el nodo derecho del árbol.

BinTree

Atributos:
Públicos:
BinNode* root
El puntero hacia la raiz del árbol binario (BinTree).

BinNode* p
El puntero hacia un nodo del árbol binario (BinTree).

Operaciones:
Publicas:
BinNode* InsertD (CHtml* elemento)
Permite insertar por la derecha un nuevo elemento Chtml en el árbol.

BinNode* InsertDAqui(BinNode* node, CHtml* elemento)
Inserta por la derecha un nuevo elemento Chtml en el árbol en el nodo del árbol especificado.

BinNode* InsertIAqui(BinNode* node, CHtml* elemento)
Inserta por la izquierda un nuevo elemento Chtml en el árbol en el nodo del árbol.

BinNode* InsertNodeDerAqui (BinNode* node,BinNode* aqui)
Inserta por la derecha un nodo en el árbol en el nodo del árbol especificado.

BinNode* InsertNodeIzqAqui (BinNode* node,BinNode* aqui)
Inserta por la izquierda un nodo en el árbol en el nodo del árbol especificado.

Bool InsertI (CHtml* elemento)
Permite insertar un elemento por la izquierda.

BinNode* Find(CHtml* elemento)
Permite localizar el nodo del árbol (BinNode) se encuentra referenciado el elemento de la clase CHtml.

void DeleteTree(BinNode* n)
Permite eliminar el árbol.
CPila* html()

int InfoN(BinNode* pnd)
Permite obtener información del nodo del árbol.

BOOL EsHijoDer(BinNode* hijo,BinNode* padre)
Permite conocer si el tipo de nodo consultado es un hijo derecho con respecto a su nodo padre.

BOOL EsHijoIzq(BinNode* hijo,BinNode* padre)
Permite conocer si el tipo de nodo consultado es un hijo izquierdo derecho con respecto a su nodo padre.

CHtml

Atributos:
Privados:
BOOL Visitado
Sirve para conocer si el elemento ha sido visitado alguna vez, durante su recorrido, utilizado para generar el código html.

Públicos:
BOOL m_Tag_Abierto
Permite especificar si el elemento específico no necesita cerrar su *tag*.

CString Tag
Contiene el *tag* que identifica a cada elemento html.

CLista* atrib
El puntero a una lista que contiene los atributos asociados a cada elemento.

CString Texto
Es el texto asociado con el elemento Html.

Operaciones:
Públicos:
CHtml()
El constructor de la clase

CString TAG()
Permite obtener el *tag* que identifica con cada elemento.

void SetText(CString txt)
Permite asignar el texto de cada elemento.

void GetText(CString txt)
Permite obtener el texto asociado al elemento.

void SetTagAbierto()
Permite especificar si el *tag* no se debe cerrar.

BOOL TagAbierto()
Obtenemos información sobre si el *tag* es de tipo abierto.

CString codigo_html()
Obtiene el lenguaje html asociado el elemento html , sin cerrar el *tag*.

BOOL FueVisitado()
Obtiene información sobre si la visita al elemento Chtml fue realizada o no cuando se recorre el árbol.

void SetVisita()
Permite especificar que un nodo ha sido visitado.

CString tag_fin()
Obtiene el código html respecto al *tag* de cierre.

virtual ~CHtml()
Permite eliminar elementos.

Protegidas:
CString SetAlign(int align)
Permite asignar el atributo de align para los elementos que usen esta propiedad.

CString SetType(int type)
Permite asignar el atributo de Type para los elementos que usen esta propiedad.

CAPÍTULO III

MANUAL DEL USUARIO

3.1. REQUERIMIENTOS DEL SISTEMA IMPLEMENTADO

Para el correcto funcionamiento del Sistema Implementado se requiere de ciertos requisitos tanto en el ambiente Hardware como en el ambiente de Software los cuales los detallamos a continuación:

Ambiente Hardware:

- Computador IBM o compatible.
- Procesador Pentium, 100 MHz o superior.
- 16 Mb en Ram.
- Disco Duro de 1 Gb.
- Monitor SVGA de 14" con capacidad de 256 colores o superior.

Ambiente Software:

- Sistema operativo Windows 95.
- Browser: Netscape Navigator Gold 3.0.
- Browser: Internet Explorer.

• 3.2. GUÍA DE USO DEL SISTEMA IMPLEMENTADO

En esta sección explicaremos paso a paso el uso del Sistema Implementado; para lograr que el usuario entienda rápidamente el Sistema empezaremos describiendo cada uno de los objetos visuales que presenta el Sistema y cómo el usuario debe interactuar con ellos.

Para que usted tenga una idea general del Sistema, éste tiene la apariencia de un editor de textos pero el ingreso de los datos es a través de Ventanas de diálogo lo que permite ser diferente de un editor común; el menú principal básicamente presenta los siguientes objetos:

- Una barra de herramientas con sus respectivos botones.
- Una barra de menú y con sus elementos correspondientes.
- Cajas de diálogo con sus respectivos atributos.

3.2.1. Descripción de la interface del Sistema

Pantalla principal

El Sistema implementado opera bajo una serie de órdenes generadas a través de opciones que pertenecen a la barra del Menú principal, Barra de herramientas y Ventanas de diálogo, que el

Sistema posee para la generación del código HTML correspondiente.

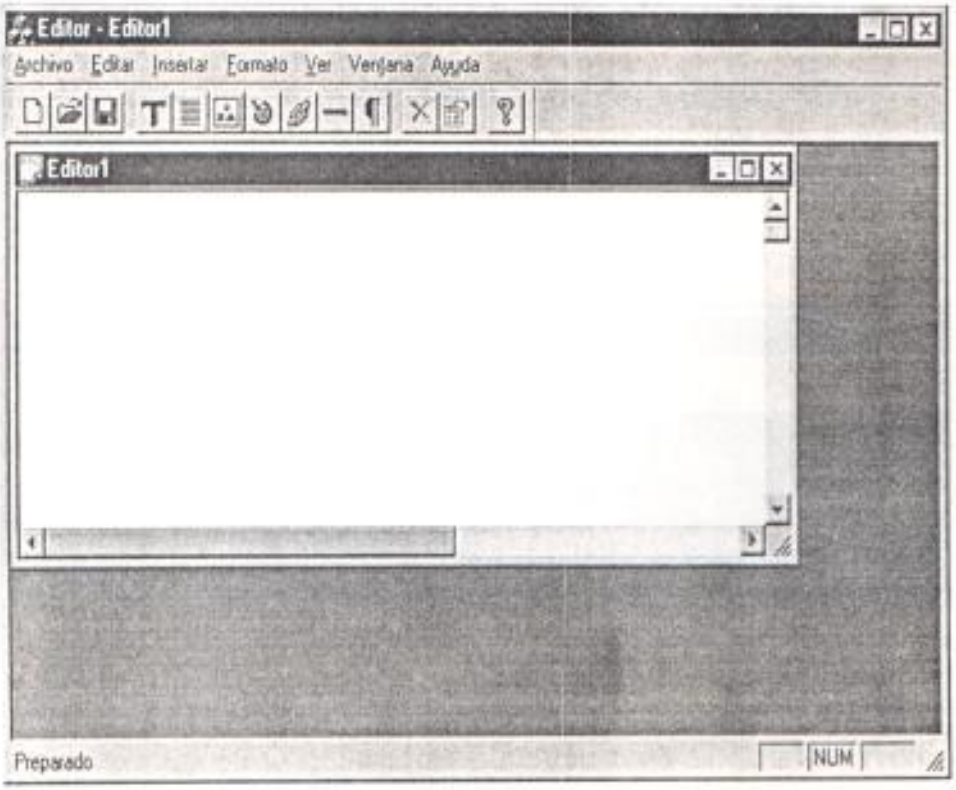


Figura 1: Pantalla principal del sistema

La pantalla presentada en la figura 1 es la primera que aparece cuando arranca el sistema implementado; esta pantalla consta de una barra de título, una barra de menús, una barra de herramientas, una ventana que representa a la vista del documento y una barra de estados.

A continuación se describen cada uno de los elementos que componen la pantalla principal:

Barra de título

Se encuentra en la parte superior de la pantalla; tiene características propias dadas por Windows 95 y se encuentra resaltada con la palabra Editor.

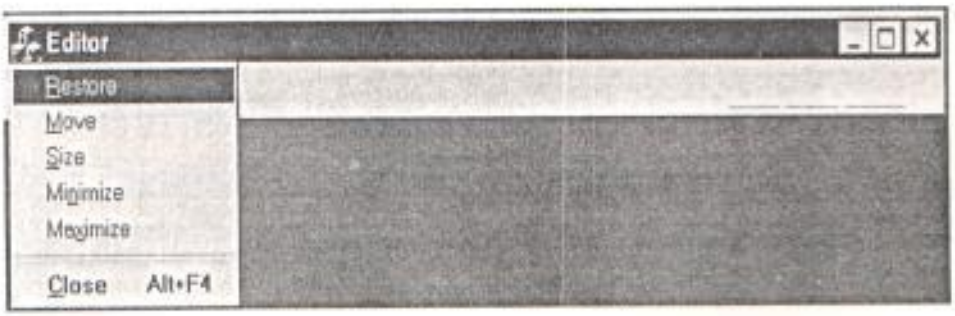


Figura 2: Barra de título del sistema

Barra de menú

La Barra de menú se encuentra en la parte superior de la pantalla después de la Barra de título; consta de siete elementos; cada uno de los cuales encierra un conjunto de opciones.

Figura 3: Barra del menú del sistema

Estos elementos son:

- Archivo
- Editar
- Insertar
- Propiedades
- Ver
- Ventana
- Ayuda

Barra de herramientas

La barra de herramientas se encuentra en la parte superior de la pantalla después de la barra de menú, consta de 13 botones gráficos.

Figura 4: Barra de herramientas del sistema

Estos botones son:

- **Nuevo:** Corresponde al primer botón empezando por la izquierda, sirve para crear un nuevo documento.

- **Abrir:** Corresponde al segundo botón empezando por la izquierda, sirve para abrir un documento ya existente.
- **Guardar:** Corresponde al tercer botón empezando por la izquierda, sirve para guardar un documento activo.
- **Título:** Corresponde al cuarto botón empezando por la izquierda, sirve para insertar un título en el documento.
- **Párrafo:** Corresponde al quinto botón empezando por la izquierda, sirve para insertar un párrafo en el documento.
- **Imagen:** Corresponde al sexto botón empezando por la izquierda, sirve para insertar una imagen en el documento.
- **Ancla:** Corresponde al séptimo botón empezando por la izquierda, sirve para insertar un ancla en el documento.
- **Enlace:** Corresponde al octavo botón empezando por la izquierda, sirve para crear un enlace en el documento.
- **Línea Horizontal:** Corresponde al noveno botón empezando por la izquierda, sirve para insertar una línea horizontal en el documento.
- **Enter:** Corresponde al décimo botón empezando por la izquierda, sirve para insertar un espacio vertical (ENTER) en el documento.

- **Eliminar:** Corresponde al onceavo botón empezando por la izquierda, sirve para Eliminar un objeto HTML del documento.
- **Propiedades:** Corresponde al décimo segundo botón empezando por la izquierda, sirve para mostrar las propiedades de un objeto HTML, para su correspondiente su modificación.
- **Acerca de:** Corresponde al décimo tercero botón empezando por la izquierda, sirve para mostrar información del programa.

Ventana de la Vista del documento

La Ventana de la vista se encuentra después de la Barra de herramientas; ocupa la mayor parte de la pantalla, es utilizada para mostrar los diferentes estilos que el usuario le da a su documento. Además cuenta con atributos propios de una ventana de Windows.

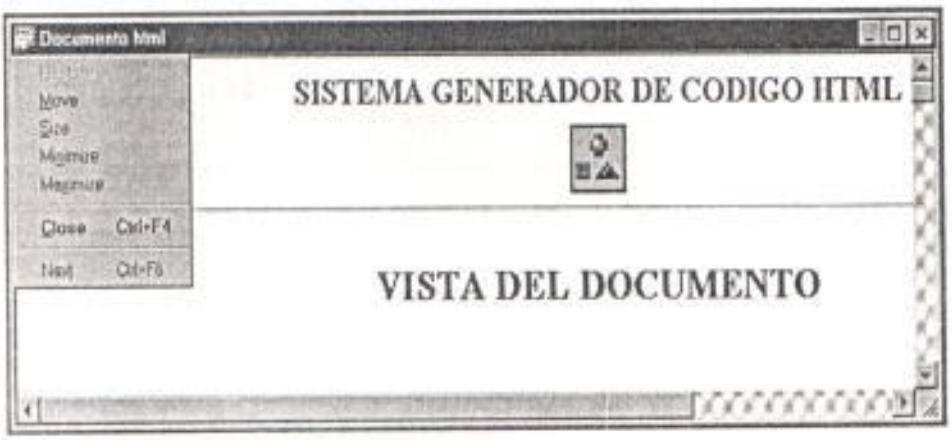


Figura 5: Ventana de vista del documento

Barra de estado

La barra de estado se encuentra en la parte inferior de la pantalla del Sistema, ésta es utilizada para mostrar las cadenas de peticiones cada vez que un elemento del menú o algún botón de la barra de herramientas es iluminado.

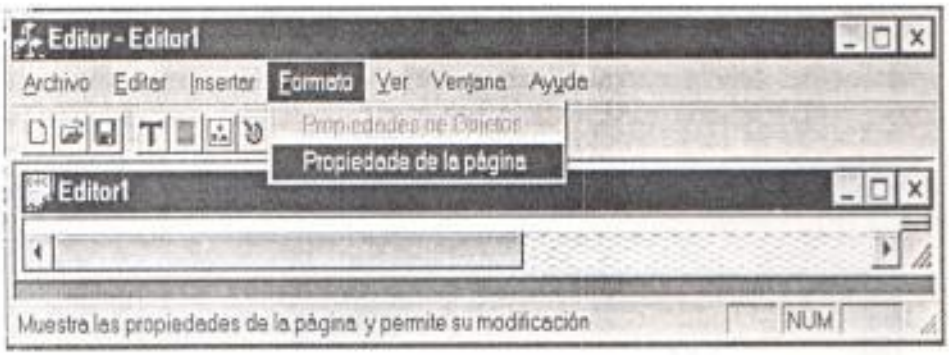


Figura 6: Barra de estado

3.2.2. Descripción de opciones de lo elementos que conforman la Barra de menú

El menú principal del Sistema consta de siete elementos totalmente funcionales, cada elemento agrupa un conjunto de opciones; cada opción es una orden que el Sistema debe interpretar y responder en un instante determinado.

Opción Archivo

El menú Archivo se encuentra en la parte superior izquierda de la pantalla, ocupando el primer lugar por izquierda de la barra de menú; consta de siete opciones todas ellas relacionadas entre sí. Tal como se ilustra en la figura 7 este menú consta de seis opciones tales como Nuevo, Abrir, Cerrar, Guardar, Guardar como, Imprimir, Presentación preliminar, Configurar impresora, Archivo reciente y Salir.

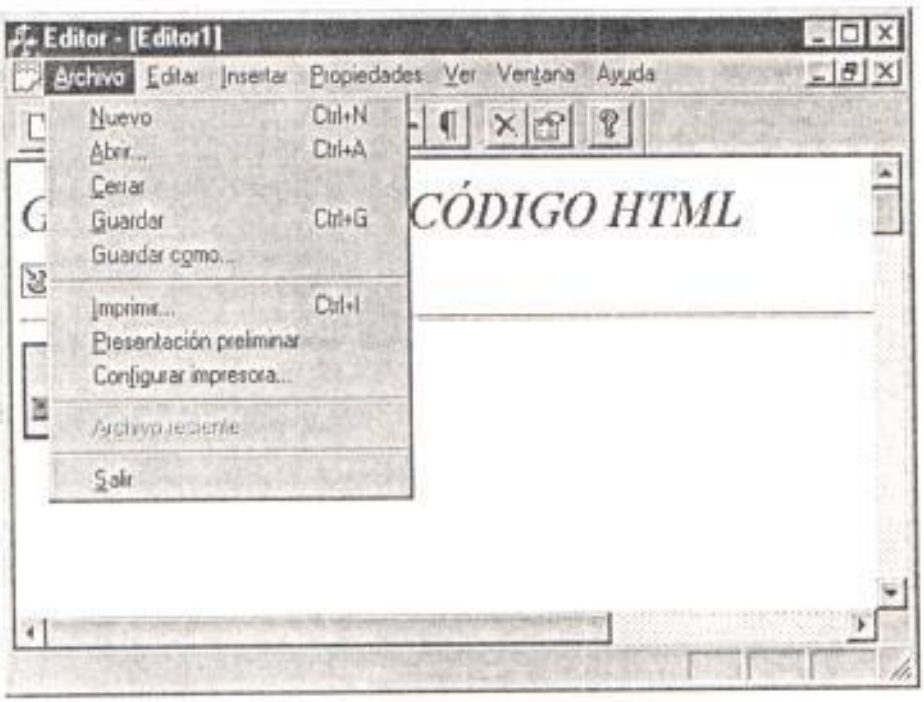


Figura 7: Opción Archivo

Las opciones que este menú posee son similares a las del menú Archivo de un editor; las características de estas opciones las describimos a continuación:

Nuevo: Permite la creación de un nuevo documento HTML.

Abrir: Permite abrir un documento HTML existente.

Cerrar: Cierra el documento HTML activo.

Guardar: Tiene la función de guardar un documento activo.

Guardar como: La función de esta opción es guardar el documento activo con un nuevo nombre.

Imprimir: Esta opción tiene la función de imprimir un documento activo.

Presentación preliminar: Tiene la función de mostrar páginas completas de un documento HTML.

Configurar impresora: Permite cambiar las opciones de la impresora y de la impresión.

Archivo reciente: La función de esta opción es abrir el documento HTML que fue último en ser cerrado.

Salir: Sale de la aplicación, le pide que guarde los documentos.

Opción Editar

Ocupa el segundo lugar empezando por la izquierda en la barra de menú, cuenta con una sola opción, y tiene la apariencia de la figura 8.

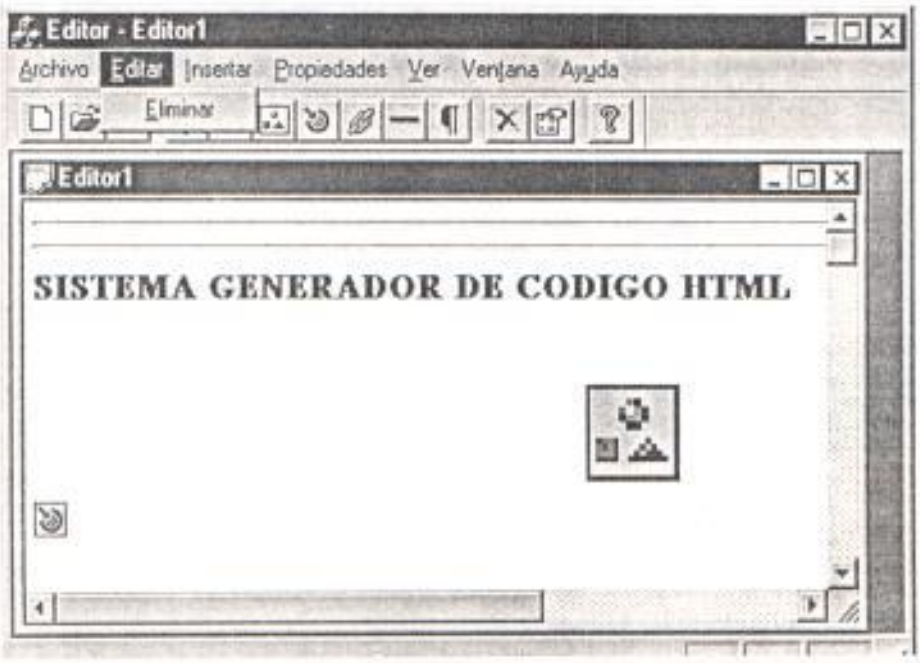


Figura 8: Opción Editar

La opción que posee el menú de Editar es:

Eliminar: Elimina un elemento HTML del documento.

Opción Insertar

Este elemento del menú se encuentra en la tercera posición empezando por la izquierda de la barra de menú. Cuenta con las opciones de Título, Párrafo, Imagen, Ancla, Enlace, Línea, Enter como se puede apreciar en la figura 9.

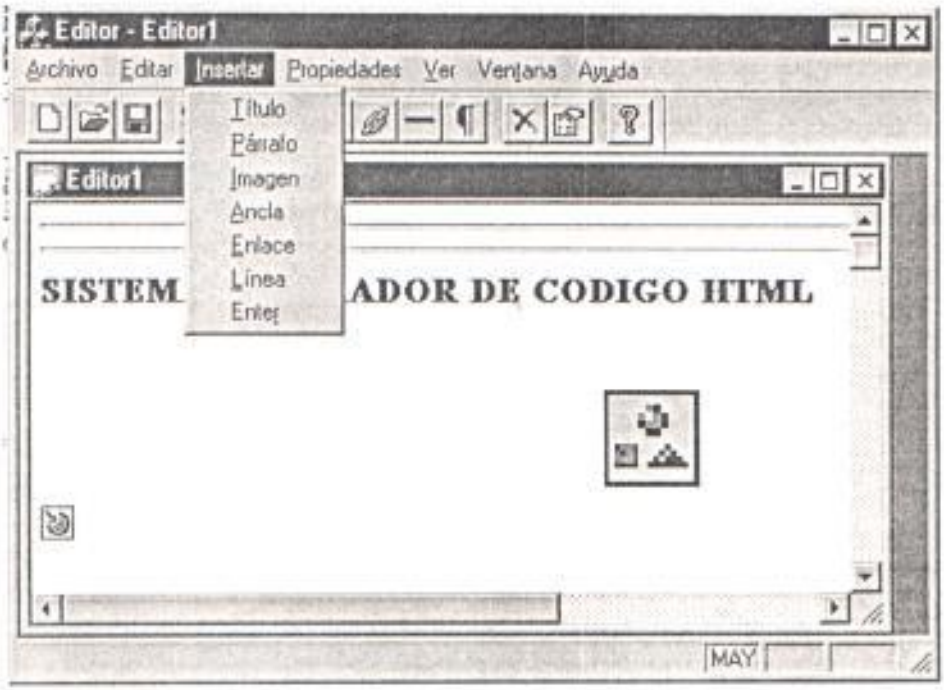


Figura 9: Opción Insertar

Todas las opciones que se encuentran bajo este menú son órdenes que tienen que ver con el manejo de objetos HTML. Estas opciones las describimos a continuación:

Título: Inserta un título en el documento HTML.

Párrafo: Permite la inserción de un párrafo en el documento HTML.

Imagen: Tiene la función de insertar un imagen en el documento HTML.

Ancla: Permite insertar un ancla en el documento HTML.

Enlace: Tiene la función de crear un enlace en el documento HTML.

Línea: Tiene la función de insertar una línea horizontal en el documento HTML.

Enter: Permite insertar un espacio vertical en el documento.

Opción Propiedades

Este elemento del menú se encuentra en la tercera posición empezando por la izquierda de la barra de menú. Cuenta con las opciones siguientes: Objetos y Página, tal como se observa en la figura 10.

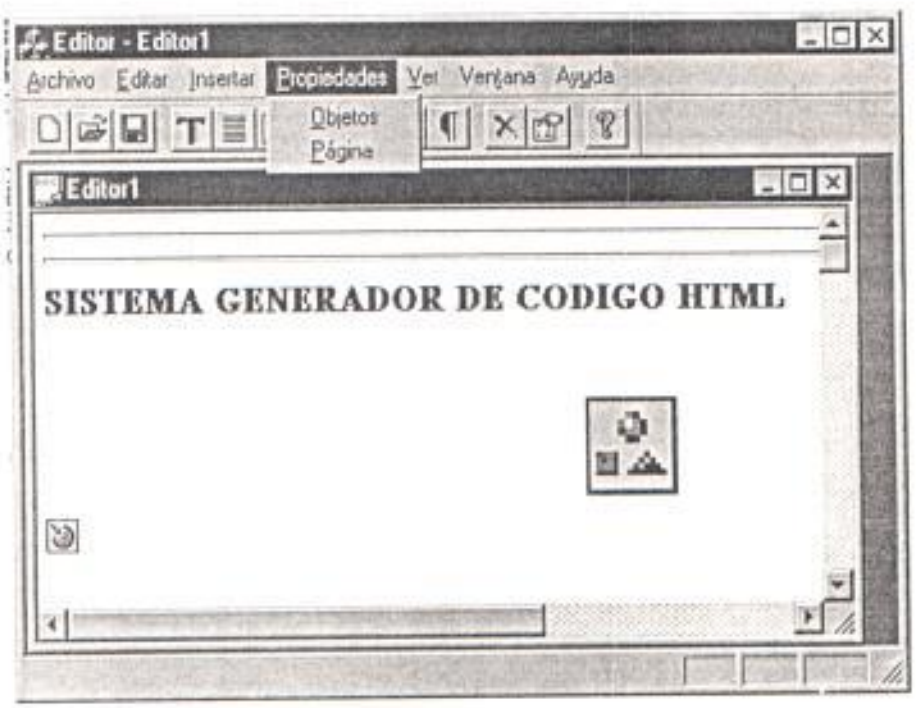


Figura 10: Opción Propiedades

Las opciones pertenecientes a este menú se describen a continuación:

Objetos: Permite mostrar las propiedades un objeto HTML; estas propiedades pueden ser modificadas.

Página: Muestra las propiedades de la página y permite su modificación.

Opción Ver

El menú Ver es el quinto elemento a partir de la izquierda que compone la barra de herramientas. Estos elementos son la Barra de menús y la Barra de estado.

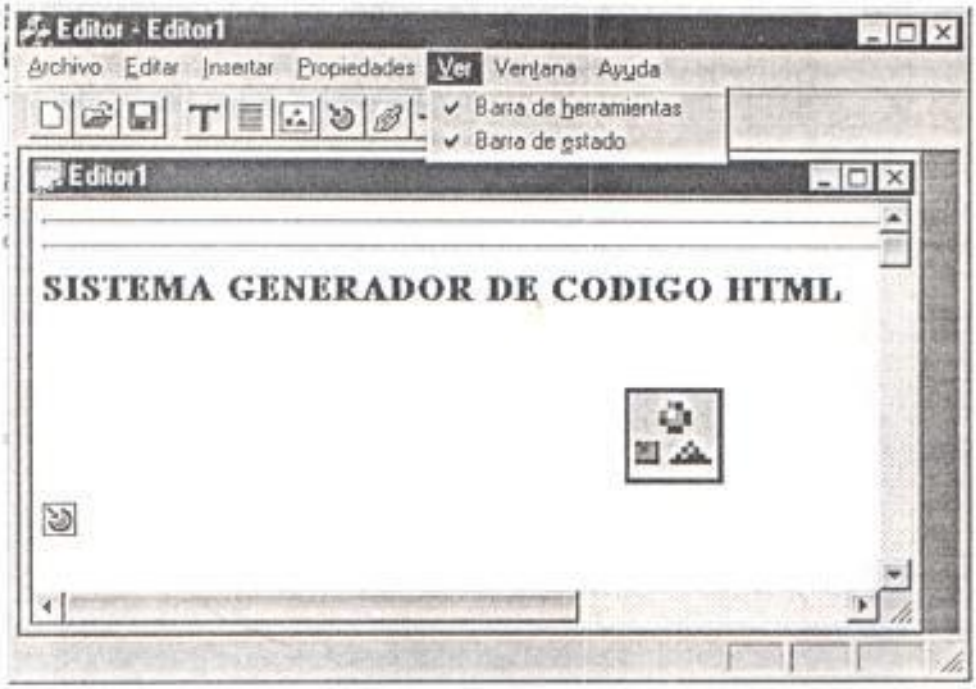


Figura 11: Opción Ver

La descripción de las opciones pertenecientes a este menú las damos a continuación:

Barra de herramientas: Muestra u oculta la barra de herramientas.

Barra de Estado: Muestra u oculta la barra de estado.

Opción Ventana

Es el sexto elemento empezando por la izquierda de la barra de menú; consta de tres opciones y un campo dedicado para almacenar los nombres de los documentos activos estos son:

Cascada, Mosaico, Organizar íconos y el campo dedicado para los documentos que se encuentren activos.

Figura 12: Opción Ventana

La descripción de cada una de las opciones pertenecientes a este menú las damos a continuación:

Cascada: Organiza las ventanas de modo que se superpongan.

Mosaico: Organiza las ventanas como mosaicos que no se superponen.

Organizar íconos: Organiza los íconos en la parte inferior de la ventana.

Opción Ayuda

Corresponde al séptimo elemento empezando por la izquierda de la barra de menú, solamente consta de una sola opción; esta opción es "Acerca de Editor".

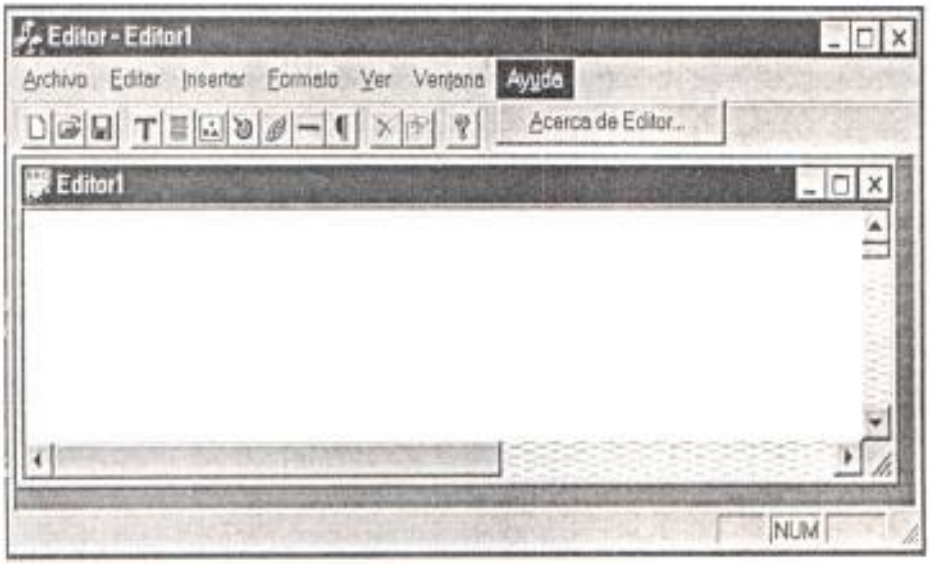


Figura 13: Opción Ayuda

A continuación describiremos la función de la única opción con que cuenta este elemento del menú:

Acerca de Editor: Muestra información del programa, número de versión y copyright.

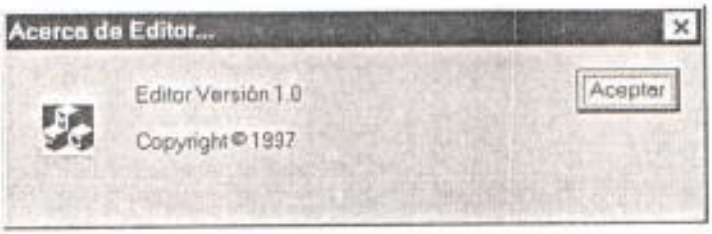


Figura 14: Ventana de Ayuda

3.2.3. Interactuando con el menú principal

La manera de ejecutar las opciones(órdenes interpretadas por el Sistema), requeridas por los usuarios se procede de varias maneras, las cuales son mencionadas a continuación:

- Para ciertas opciones del menú pulse la combinación de teclas correspondientes que aparecen a la derecha de la opción.
- Pulse la tecla ALT para ubicarse en el menú, luego con las teclas de control de movimiento del cursor seleccione la opción deseada y pulse ENTER para ejecutar la opción.
- Posicione el ratón sobre el botón deseado de la Barra de herramientas y dar un pulso sobre el botón izquierdo del ratón.
- Apunte con el ratón a una opción del menú principal, y pulsar el botón izquierdo del ratón. Para cancelar la opción elegida, apuntar a cualquier parte fuera del menú presentado, y pulsar el botón izquierdo del ratón.

3.2.4. Creación de un documento HTML

Para la creación de un documento puede utilizar tanto la barra de herramientas flotante como la opción propuesta por el menú principal.

Utilizando la barra de herramientas flotante:

- Arrastre el ratón hasta el botón ubicado en la primera posición empezando por el lado izquierdo de la barra de herramientas y haga clic sobre el botón izquierdo del ratón. El Sistema le responderá abriendo una ventana de edición similar a la ventana presentada en la figura 1.

Si se usa el menú principal, esta orden la puede ejecutar de varias maneras:

- Presionando las combinaciones de teclas CTRL + N.
- Utilizando la tecla ALT para ubicarse en el menú principal, luego desplazarse mediante el uso de las teclas de movimiento del cursor hasta la opción Nuevo que se encuentra en el menú Archivo y pulse la tecla ENTER.

- Arrastrando el ratón hacia el menú Archivo; una vez que la opción Nuevo es resaltada presione el clic izquierdo del ratón.

3.2.5. Abriendo un documento HTML existente

1. Las diferentes maneras para abrir un documento son:
   - Arrastre el ratón sobre el segundo botón empezando por la izquierda de la barra de herramientas y pulse el botón izquierdo del ratón.
   - Apunte el ratón sobre opción "Abrir" de la Barra de menú y pulsar el botón izquierdo del ratón.
   - Presione la combinación de teclas CTRL + A.
   - Pulse la tecla ALT para que la barra de menú sea activada y con los controles del movimiento del cursor avance hasta el menú Archivo y descienda hasta resaltar la palabra "Abrir" para luego pulsar ENTER.

2. El Sistema desplegará la siguiente Ventana de diálogo con todos los directorios y archivo existentes.

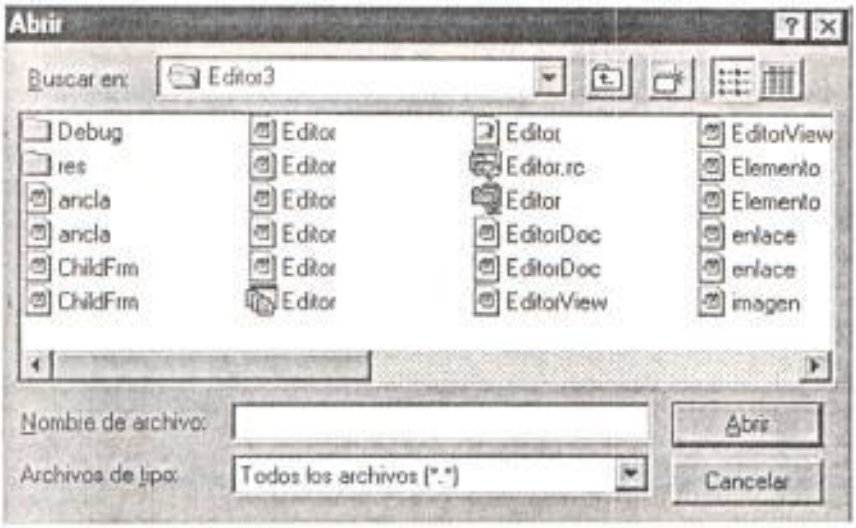


Figura 15: Ventana de diálogo para la apertura de un documento HTML

3. Arrastre el ratón hasta ubicarse sobre el nombre del archivo que desee elegir y pulse el botón izquierdo del ratón.

4. Arrastre el ratón hasta ubicarlo sobre el botón "Abrir" y pulse el botón izquierdo del ratón. El Sistema responderá abriendo el documento seleccionado.

3.2.6 Insertando un título en el documento HTML

1. Las diferentes formas para insertar un Título en el documento son:

   - Arrastre el ratón sobre el cuarto botón empezando por la izquierda de la barra de herramientas y pulse el botón izquierdo del ratón.

- Apunte el ratón sobre opción "Título" de la Barra de menú y pulsar el botón izquierdo del ratón.
- Pulse la tecla ALT para que la barra de menú sea activada y con los controles del movimiento del cursor avance hasta el menú Archivo y descienda hasta resaltar la palabra "Título" para luego pulsar ENTER.

2. El Sistema desplegará la siguiente Ventana de diálogo con todos los atributos que un Título puede tener.

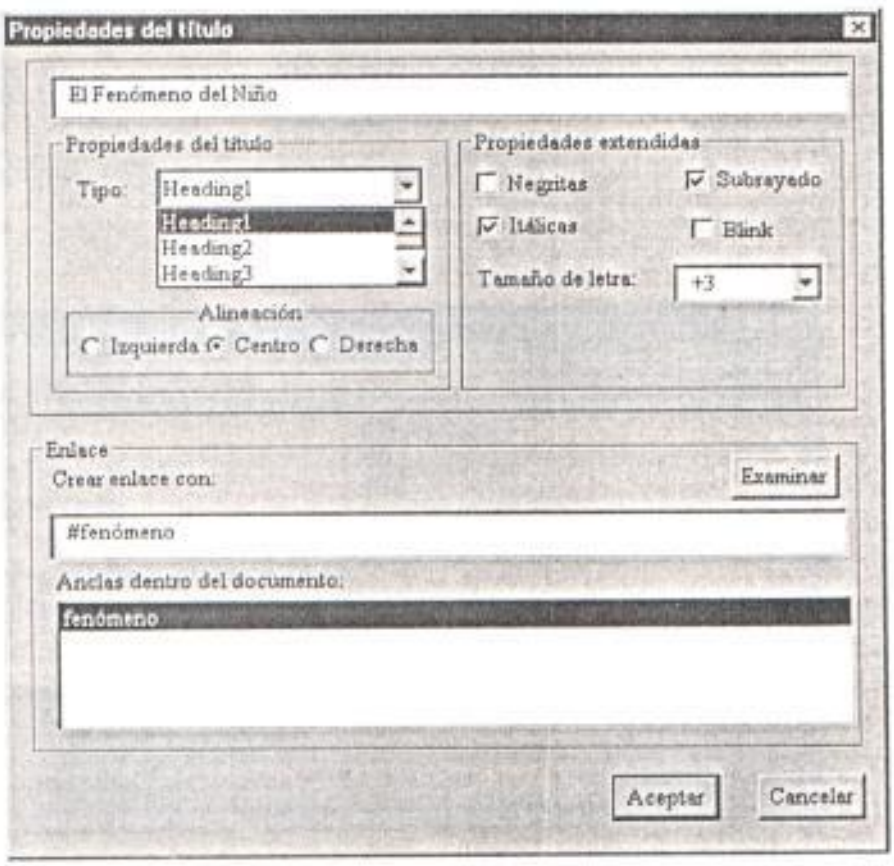


Figura 16: Ventana de diálogo para la inserción de un título en el documento HTML

3. Ingrese el título para su documento y asigne las propiedades correspondientes.
4. Ingrese el ancla correspondiente para crear el enlace dentro del documento o el path que tenga la dirección del documento que servirá de enlace para el Título.
5. Para ubicar el path correspondiente arrastramos el ratón hasta el botón "Examinar" y sobre él pulsamos el botón izquierdo del ratón.
6. El Sistema desplegará una ventana similar a la que se muestra cuando se "Abre" un documento.

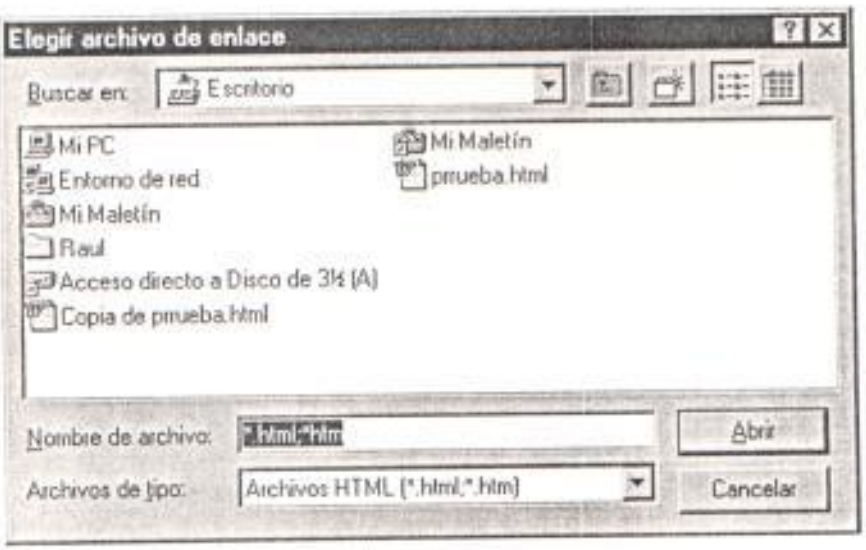


Figura 17: Ventana de diálogo pra elegir un archivo de enlace

7. Una vez asignado todos los atributos correspondientes, arrastre el ratón hasta ubicarlo sobre el botón "Aceptar" y pulse el botón izquierdo del ratón. El Sistema responderá por desplegar los cambios que el usuario ha hecho.

3.2.7. Insertando un párrafo en el documento HTML

Para el ingreso de un párrafo al documento HTML el Sistema le proporciona una Ventana de diálogo, esta ventana puede ser activada de dos maneras; mediante uso de la barra de herramientas o activando la barra de menú.

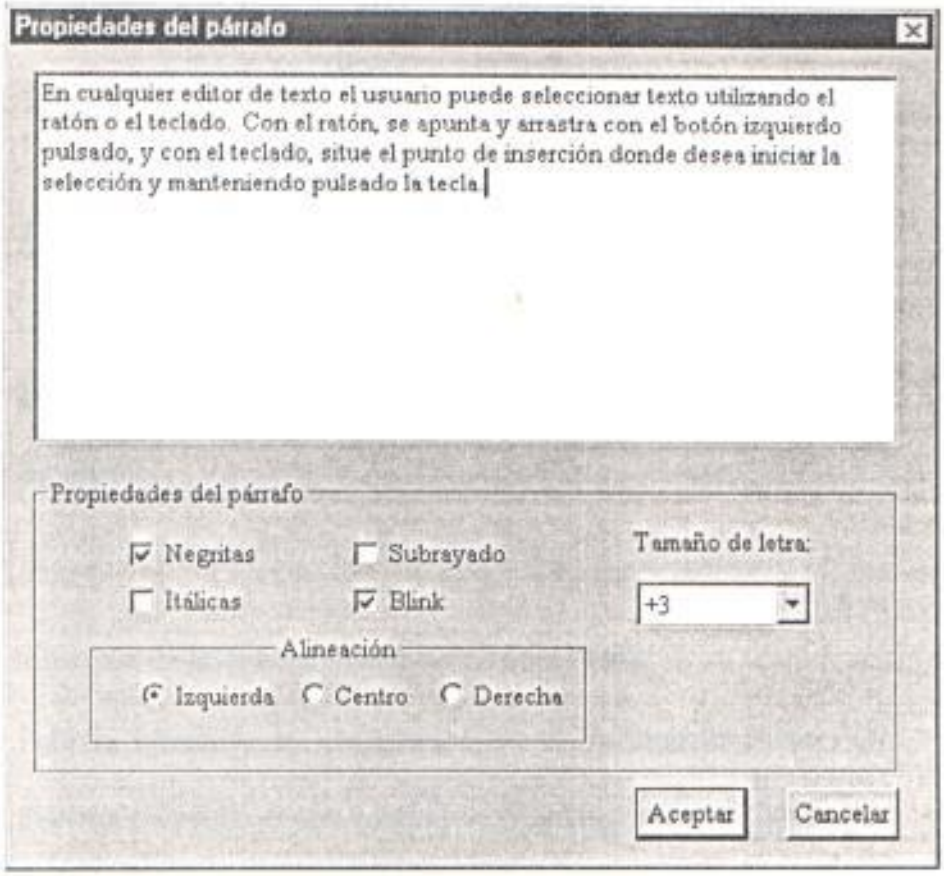


Figura 18: Ventana de diálogo para la inserción de un párrafo en el documento HTML

El párrafo ingresado puede ser alterado de acuerdo a las propiedades que el Sistema proporciona; esta propiedad es:

Propiedad del párrafo: En esta sección el Sistema permite asignarle párrafo propiedades de tipo de letra tales como Negritas, Itálicas, Subrayado, Blink.

El Sistema también permite cambiar el tamaño de los caracteres que conforman el párrafo para cual propone varias alternativas.

De la misma manera el Sistema permite centrar el párrafo para lo cual propone alternativas tales como izquierda, centro y derecha que permiten ubicar el texto en determinada posición del documento.

3.2.8. Inserción de imágenes en el documento HTML

Para la inserción de una imagen en el documento HTML el Sistema proporciona una ventana de diálogo a través de la cual el usuario puede elegir las propiedades de una imagen. Para que la Ventana de diálogo sea activada el usuario puede seguir cualquiera de los dos procedimientos:

Activar la barra de menú o seleccionar el botón correspondiente en la barra de herramientas.

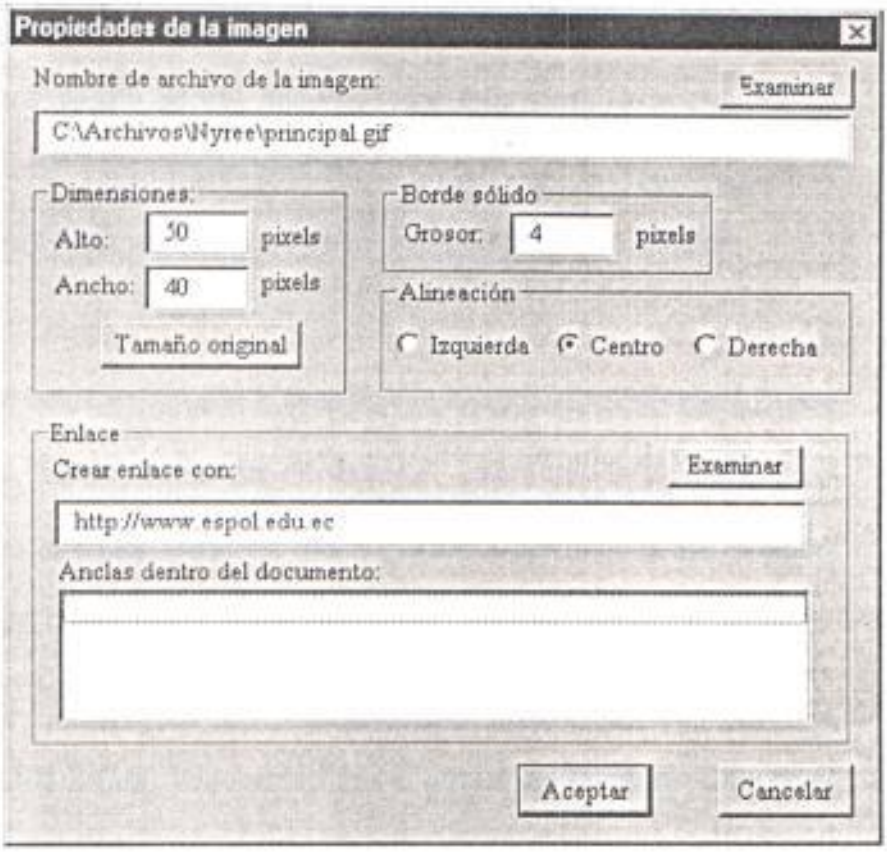


Figura 19: Ventana de diálogo para la inserción de una imagen en el documento HTML

La dirección donde se encuentra la imagen que se quiere insertar debe ser ingresada en el campo "Nombre de archivo de la imagen" si esta es conocida por el usuario, en caso contrario pulse el botón examinar para escoger el directorio correspondiente.

El Sistema tiene la capacidad para asignar las siguientes propiedades a una imagen:

Dimensiones: Esta propiedad permite establecer tanto el ancho como el alto de una imagen; todas estas medidas son dadas en pixeles. Para establecer el tamaño original de la imagen pulsar el botón izquierdo del ratón sobre el botón "Tamaño original".

Enlace: Esta sección le permite a la imagen crear un enlace ya sea con "anclas dentro del documento" con otros documentos, pero para ello debe pulsar el botón izquierdo del ratón sobre el botón "examinar".

Borde sólido: Esa sección fija el grosor del borde de la imagen, sus unidades están dadas en pixeles.

Alineación: Esta sección permite ubicar la imagen en el documento de acuerdo a la necesidad del usuario; presenta las opciones de alinear la imagen ala izquierda, centro, o derecha.

3.2.9. Inserción de una ancla en el documento HTML

La orden que permite desplegar la Ventana de diálogo para el ingreso del nombre del ancla puede ser ejecutada a través del menú principal o de la Barra de herramientas.

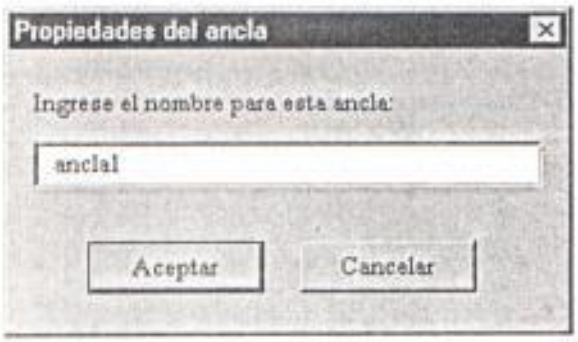


Figura 20: Ventana de diálogo pra la inserción de un ancla en el documento HTML

Para crear una ancla en el documento HTML ingrese un nombre en la caja de texto correspondiente al ítem "Ingrese el nombre para esta ancla", luego pulse el botón izquierdo del ratón cuando este posicionado sobre el botón Aceptar.

3.2.10. Inserción de un enlace en el documento HTML

Para insertar un enlace en el documento HTML ubíquese con el ratón ya sea sobre el botón de enlace de la barra de herramientas o sobre la opción Enlace que se encuentra en el menú de Insertar. El sistema responderá desplegando la siguiente Ventana de diálogo:

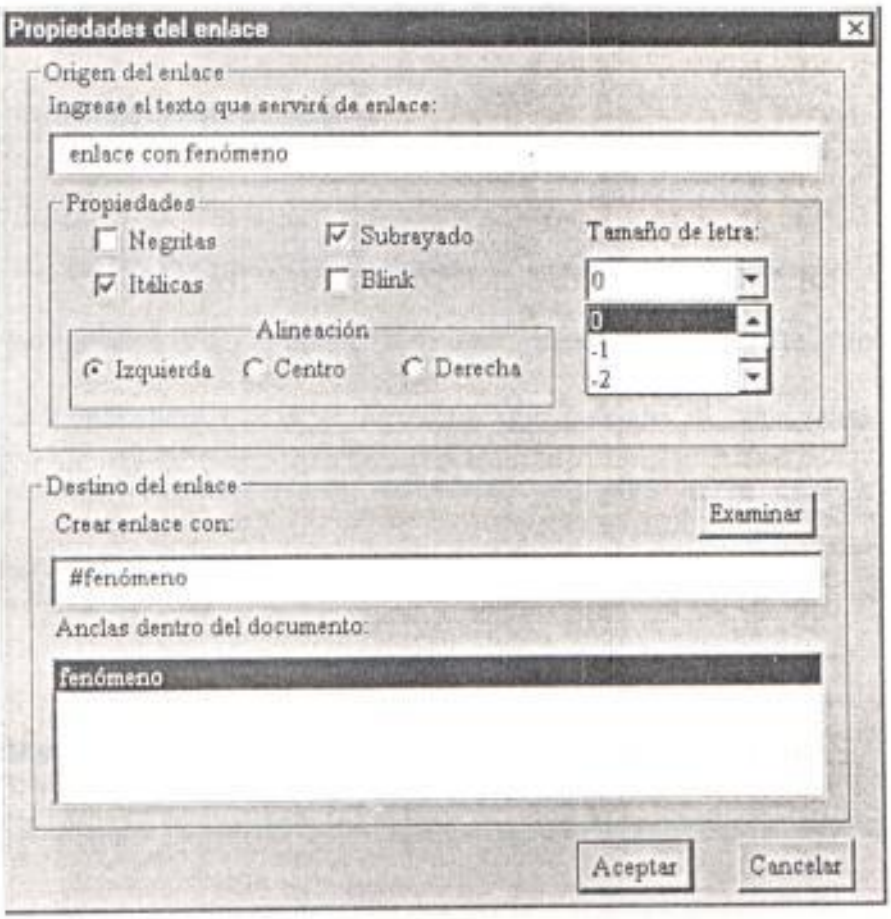


Figura 21: Ventana de diálogo para la inserción de un enlace en el documento HTML

Ingrese el texto que luego le servirá como enlace, bajo el ítem "Ingrese el texto que servirá de enlace"; este texto tiene propiedades de negritas, itálicas, subrayado y blink, así también puede cambiar el tamaño de la letra del texto ingresado, seteando a los valores fijos(0,-1,-2) que aparecen bajo el ítem "Tamaño de letra". De la misma forma puede alinear el texto de acuerdo a las

propiedades de alineación que el sistema propone estas son izquierda, centro, derecha.

Ingrese el destino del enlace bajo el ítem "Crear enlace con"; posicionese sobre el botón "Examinar" y pulse el botón izquierdo del ratón para buscar el directorio que contiene el documento destino. También puede seleccionar cualquier ancla que se encuentra bajo el ítem "Anclas dentro del documento".

3.2.11. Inserción de una Línea Horizontal en el documento HTML

Seleccione ya sea del menú principal o de la barra de herramientas la opción o botón en su orden que permite el desplegamiento de la Ventana de diálogo que nos permite el ingreso de la línea horizontal al documento HTML.

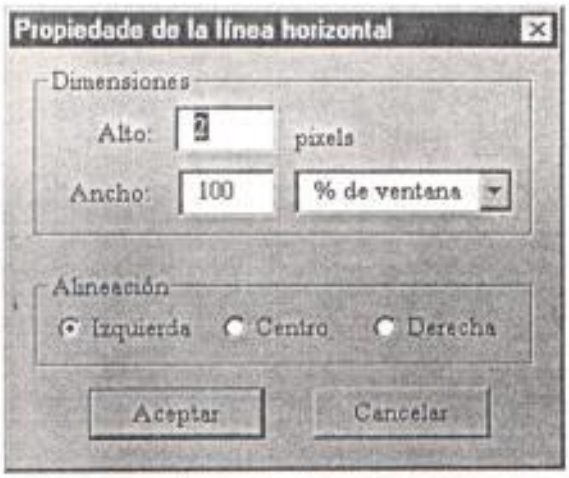


Figura 22: Ventana de diálogo para la inserción de una línea horizontal en el documento HTML

Una vez que la Ventana de diálogo ha sido desplegada ingrese valores para el alto y el ancho con sus unidades respectivas junto a los ítems "Alto:" y "Ancho:". También puede asignarle propiedades de alineación tales como izquierda, centro, derecha y luego ubíquese sobre el botón Aceptar y pulse el botón izquierdo del ratón.

3.2.12. Inserción de Espacios Verticales en el documento HTML

Para la inserción de uno o varios espacios en el sentido vertical del documento HTML se procede de la siguiente manera:

Seleccione del documento HTML el objeto sobre el cual se desea insertar uno o varios espacios y mediante uso del menú principal o

de la barra de herramientas active la opción que permite la generación del espacio vertical.

3.2.13. Cambiando las propiedades de la página en un documento HTML

Cambiar las propiedades de una página significa cambiar el color de un texto normal , texto de enlace, texto de enlace activo, texto de enlace seleccionado y el fondo de la página. Para que el Sistema ejecute estas acciones debe seguir los siguientes pasos:

1. Mediante uso del teclado o el ratón seleccione la opción "Propiedades de la Página" del menú Formato.

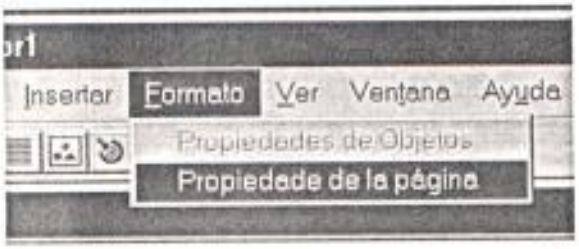


Figura 23: Propiedades de la página

2. A continuación se presenta una ventana de diálogo conteniendo todos los atributos que el Sistema le puede dar a una página; elija y fije cualquiera de las opciones presentes.

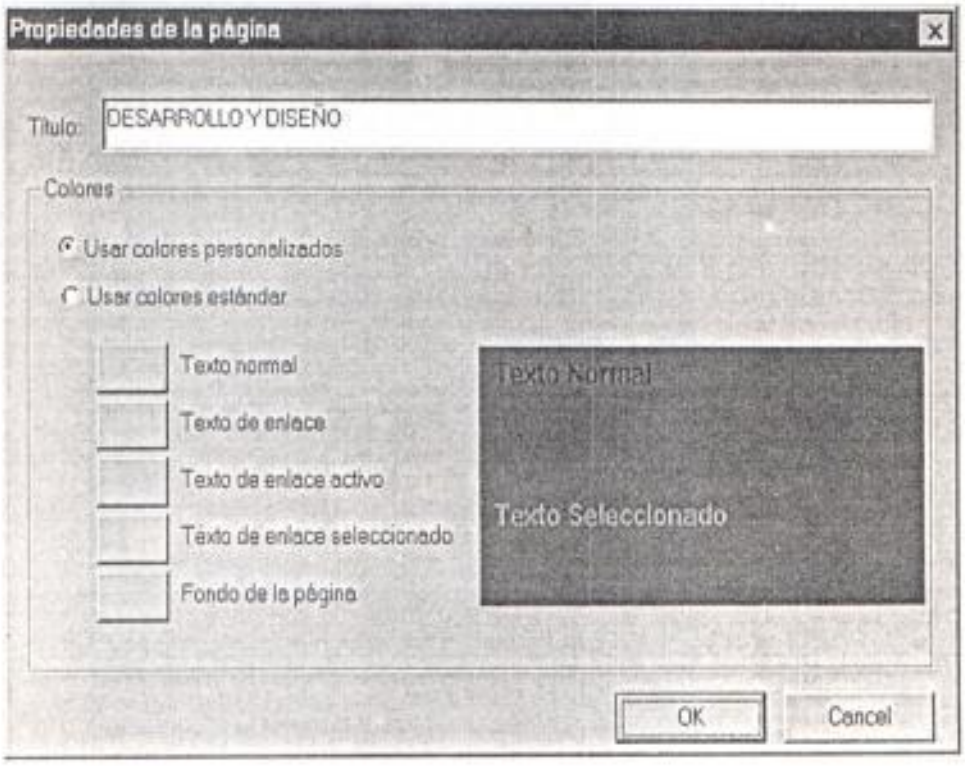


Figura 24: Asignación de propiedades de página en un documento HTML

3. Posiciónese sobre cualquiera de los botones correspondiente al atributo que desea cambiar y pulse el botón izquierdo del ratón.

4. El Sistema desplegará una ventana con los colores básicos: elija cualquiera de ellos y presione el botón OK.

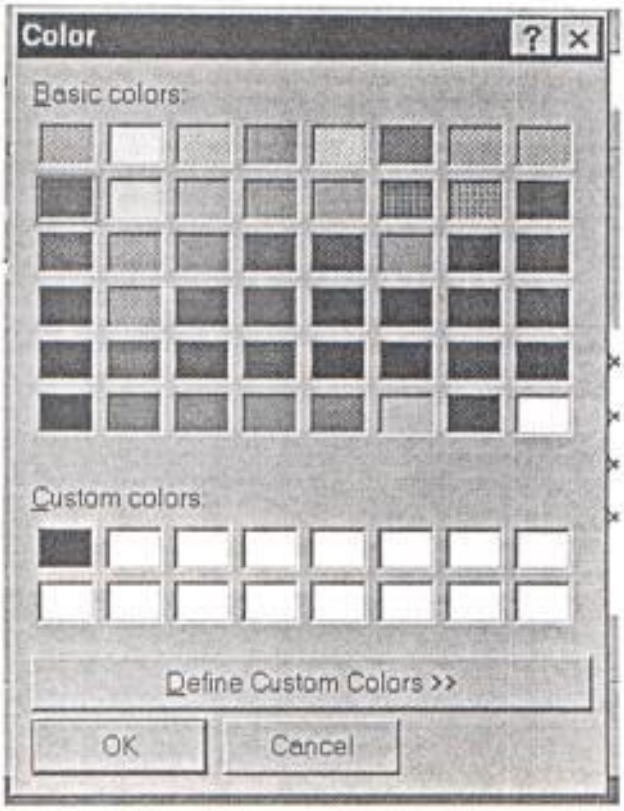


Figura 25: Asignación de un color básico

5. Para definir un color customizado posiciónese sobre el botón "Define Custom Color" y pulse el botón izquierdo del ratón.

6. El Sistema desplegará una ventanas con todos los colores posibles; elija cualquiera de ellos dando un clic de ratón sobre el botón "OK".

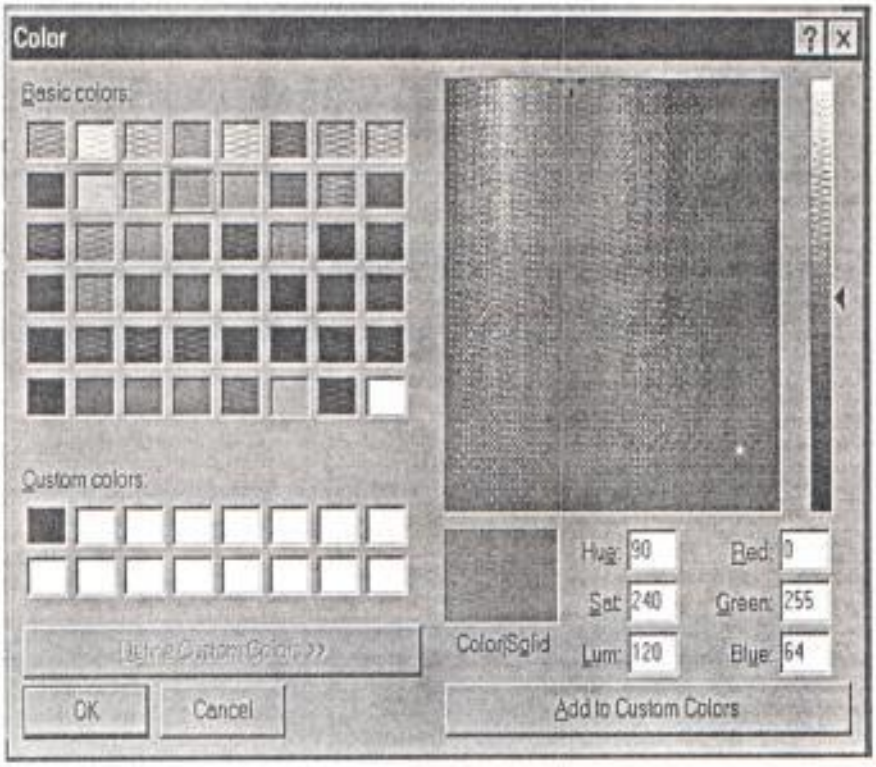


Figura 26: Asignación de un color personalizado

7. Una vez elegido el color, el Sistema se fija en la opción "Usar colores personalizados"; dar un clic izquierdo de ratón sobre el botón OK para efectuar los cambios correspondientes. Si no elige los colores disponibles el Sistema se fijará en "Usar colores estándar".

3.2.14. Eliminar elementos HTML del documento HTML

Para eliminar cualquier elemento HTML del documento primero debe ser marcado y luego eliminarlo mediante activación de Eliminar del menú Editar o ubicándose sobre el botón de la barra de herramientas y presionando el botón izquierdo del ratón.

El siguiente mensaje es presentado cuando se elimina cualquier objeto HTML.

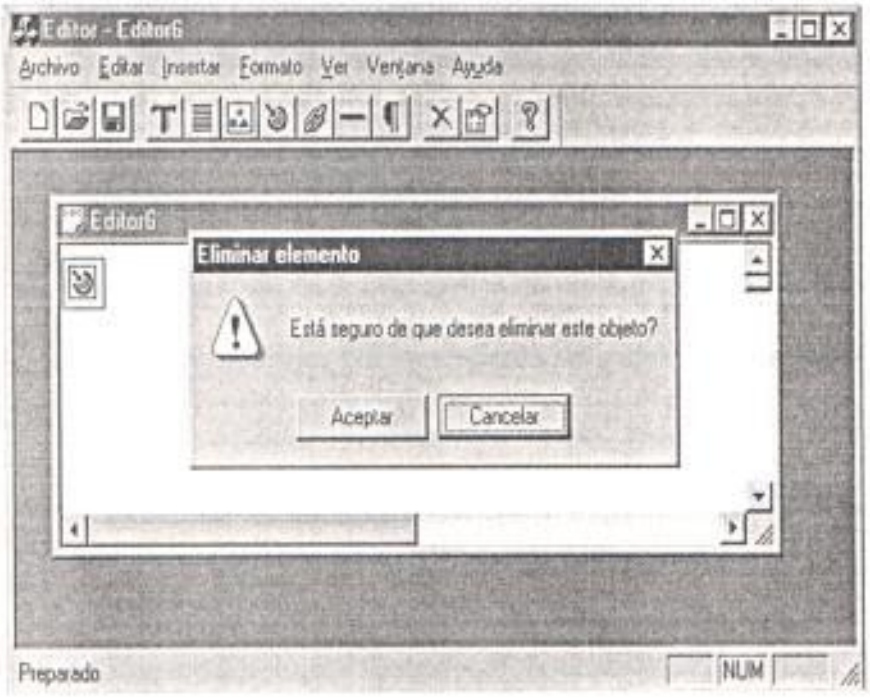


Figura 27: Mensaje de confirmación para la eliminación de un objeto del documento HTML

3.2.15. Guardando un documento HTML

1. Guarde su documento HTML escogiendo cualquiera de los siguientes procedimientos:
   - Arrastre el ratón sobre el tercer botón empezando por la izquierda de la barra de herramientas y pulse el botón izquierdo del ratón.
   - Apunte el ratón sobre la opción "Guardar como" de la Barra de menú y pulsar el botón izquierdo del ratón.
   - Apunte el ratón sobre la opción "Guardar" de la Barra de menú y pulsar el botón izquierdo del ratón.
   - Presione la combinación de teclas CTRL + G.
   - Pulse la tecla ALT para que la barra de menú sea activada y con los controles del movimiento del cursor avance hasta el menú Archivo y descienda hasta resaltar la palabra "Guardar" o "Guardar como" para luego pulsar ENTER.

2. El Sistema desplegará la siguiente Ventana de diálogo con todos los directorios y archivo grabados anteriormente.

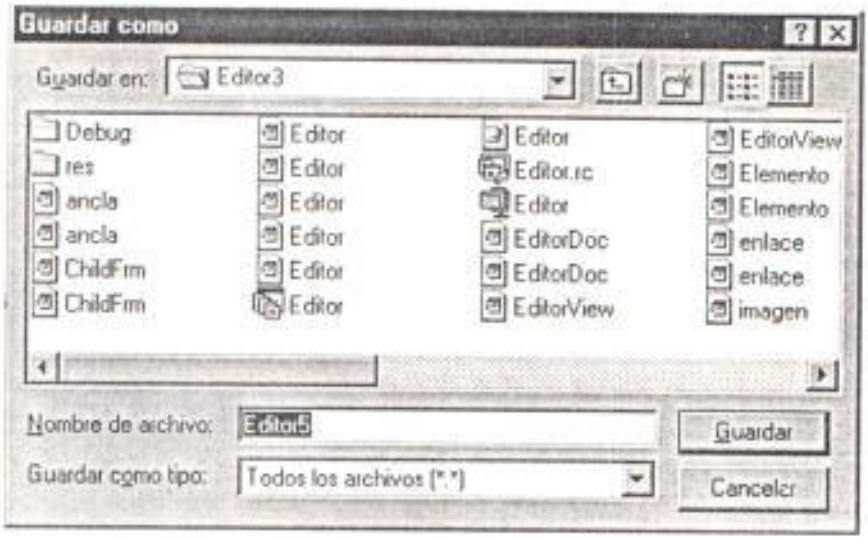


Figura 28: Ventana de diálogo para guardar un documento HTML

3. Elija el directorio correspondiente y escriba el nombre del archivo para su documento, junto al campo marcado como "Nombre de archivo".

4. Arrastre el ratón hasta ubicarlo sobre el botón "Guardar" y pulse el botón izquierdo del ratón. El Sistema responderá asignando un espacio de memoria al nombre del documento ingresado.

3.2.16.Cerrando un documento HTML

1. Las diferentes formas en que un documento puede ser cerrado son:

- Apunte el ratón sobre opción "Cerrar" de la Barra de menú y pulsar el botón izquierdo del ratón.
- Pulse la tecla ALT para que la barra de menú sea activada y con los controles del movimiento del cursor avance hasta el menú Archivo y descienda hasta resaltar la palabra "Cerrar" para luego pulsar ENTER.

2. El Sistema responderá presentando una segunda pantalla permitiendo que el usuario se mantenga en el Sistema.

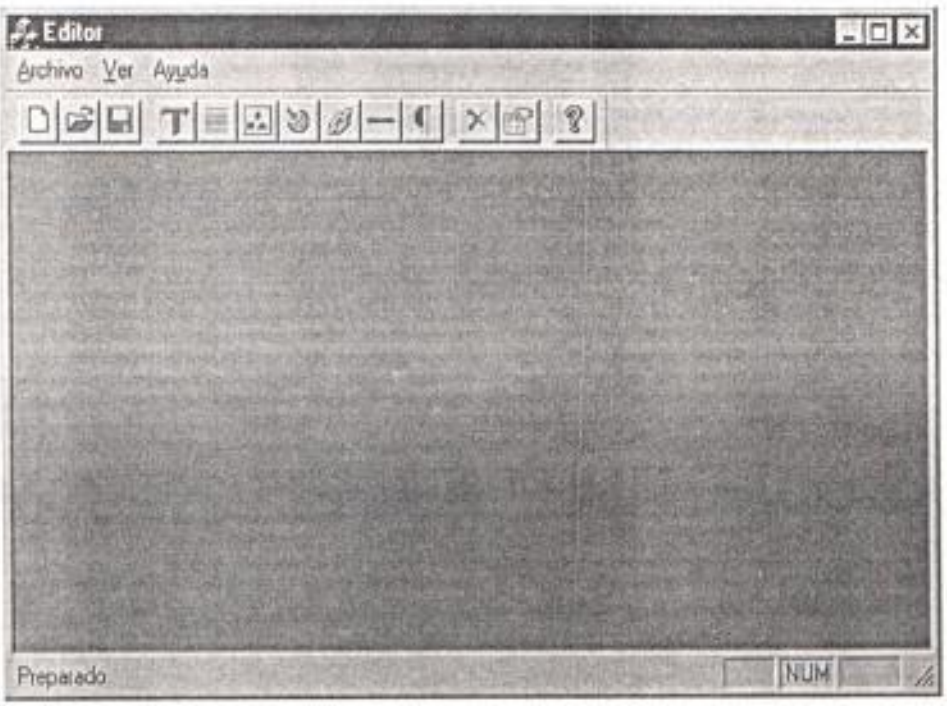


Figura 29: Estado del menú principal cuando se ha finalizado la edición de un documento HTML

3. En esta pantalla puede crear un nuevo documento, abrir un documento o salir del Sistema.

3.2.17.Saliendo del Sistema

Para salir del Sistema se puede seguir cualquiera de los siguientes pasos:

- Apunte el ratón sobre opción "Salir" de la Barra de menú y pulsar el botón izquierdo del ratón.
- Pulse la tecla ALT para que la barra de menú sea activada y con los controles del movimiento del cursor avance hasta el menú Archivo y descienda hasta resaltar la palabra "Salir" para luego pulsar ENTER.

CONCLUSIONES

1. Debido a la complejidad en la implementación del ambiente WYSIWYG, la mayoría de los editores HTML que tienen estas propiedades no son tan óptimos.

2. El Uso de la metodología Orientada a Objetos en el análisis y diseño permite la simplificación de recursos a utilizar.

3. El método de implementación Orientada a Objetos permite la simplificación de código y permite la reutlización de funciones a gran escala.

4. La metodología orientada a objetos nos permite hacer grandes cambios funcionales sin que su diseño e implementación se vea afectado en grandes proporciones.

5. El esfuerzo hecho en el análisis y diseño del Sistema implementado fue recompensado durante la etapa de la implementación; eso es lo interesante de la metodología Orientada a Objetos.

BIBLIOGRAFÍA

1. CEVALLOS, FCO. JAVIER. **Visual C++ Aplicaciones para Windows.** RA-MA, S.A. España. 1ra. edición. 1997, pp. 3 - 791.

2. GRAHAM, IAN S. **HTML SOURCE BOOK.** Jhon Wiley & Sons, Inc. Canadá. 2da. edición. 1992, pp.125 - 418.

3. JOYANES AGUILAR, LUIS. **C++ a su alcance. Un enfoque orientado a objetos.** McGRAW-HILL, S.A. España. 1ra. edición. 1994, pp. 185 - 851.

4. KRUGLINSKI, DAVID J. **Progrese con VISUAL C++.** McGRAW-HILL, S.A. España. 1ra. edición. 1993, pp. 3 - 625.

5. MICROSOFT. **Reference Volumen I.** Microsoft Corporation. EEUU. Versión 1.0. 1992, pp. 11 - 1175