ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

ESTUDIO DE TOOLBOOK INSTRUCTOR 2004 E IMPLEMENTACIÓN DE UN CURSO DE ENTRENAMIENTO SOBRE FUNDAMENTOS DE SEGURIDAD EN REDES DE DATOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DORIS MARIBEL REYES ERAZO

DIRECTOR: ING. FERNANDO FLORES
CODIRECTOR: ING. NELSON AVILA

QUITO, MARZO 2006

DECLARACIÓN

Yo, Doris Maribel Reyes Erazo, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Doris Maribel Reyes Erazo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Doris Maribel Reyes Erazo, bajo mi supervisión.

_____

Ing. Fernando Flores

DIRECTOR DEL PROYECTO

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Doris Maribel Reyes Erazo, bajo mi supervisión.

_____

Ing. Nelson Avila

CODIRECTOR DEL PROYECTO

CONTENIDO

ÍNDICE DE TABLAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

ÍNDICE DE FIGURAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

RESUMEN

El propósito del presente proyecto de titulación es el estudio de la herramienta ToolBook Instructor 2004, la cual permite la creación de aplicaciones de entrenamiento multimedia, e implementar un curso de entrenamiento basado en

computador sobre fundamentos de seguridad en redes de datos, que demuestre la funcionalidad de la misma.

El trabajo se ha estructurado en cinco capítulos. En el primero y el segundo se realiza un estudio de ToolBook Instructor 2004 y su lenguaje de programación OpenScript, de forma resumida se expone la funcionalidad y manipulación de sus elementos más relevantes.

En el tercer capítulo se analiza los fundamentos de seguridad en redes de datos, siguiendo el modelo TCP/IP, donde se revisa los principales protocolos y sus mecanismos de seguridad; además, se presenta algunas soluciones y productos comerciales disponibles en el mercado ecuatoriano. Esta información es la que se incluye dentro del contenido del curso desarrollado.

La creación del curso se describe en el cuarto capítulo. Se analizan varias metodologías de desarrollo para entrenamiento basado en computador (CBT) y su generalización al uso en la Web (WBT), se detallan las fases de implementación del curso multimedia y la mejor forma de desarrollarlas. Además se realizan pruebas de operación para evaluar la eficacia del curso.

El capítulo final, presenta conclusiones y recomendaciones alcanzadas a través del proyecto, seguidas por bibliografía y anexos.

PRESENTACIÓN

Actualmente la difusión de medios digitales y la denominada sociedad de la información en la que nos desarrollamos, requiere cada vez mayor cantidad de información soportada en diferentes formatos como: texto, imágenes, sonido y

video animado; lo que ha dado lugar a una nueva tecnología basada en aplicaciones multimedia.

La multimedia en el campo de la educación ha tomado gran fuerza, ya que permite la implementación de múltiples experiencias de aprendizaje las que facilitan el aprendizaje. En conjunto con medios de distribución como CD-ROMs o la Web puede abarcar un diverso grupo de estudiantes en lo que se refiere a edad, lugar de residencia o situación personal, y además permitir que el alumno avance en el estudio de acuerdo a su capacidad, disponibilidad de tiempo e interés.

La tendencia a utilizar medios de aprendizaje electrónicos está motivada en gran parte también por el aspecto económico, debido al ahorro que implica la utilización de éstos medios, que pueden cubrir un amplio grupo de estudiantes sin la necesidad de movilización de los mismos hacia un mismo lugar y al mismo tiempo.

El presente trabajo realiza un estudio de la herramienta de autor ToolBook Instructor y la metodología de desarrollo de aplicaciones de aprendizaje multimedia, de forma clara y concisa presenta sus aspectos más importantes, sirve como guía rápida de consulta tanto en el aspecto teórico como práctico para nuevos desarrolladores en la creación de aplicaciones multimedia interactivas en diferentes ámbitos del conocimiento.

CAPÍTULO 1

TOOLBOOK INSTRUCTOR 2004

INTRODUCCIÓN

ToolBook Instructor es una solución integral para la implementación de cursos altamente interactivos, efectivas simulaciones de aplicaciones de *software* y poderosas evaluaciones.

Instructor está diseñada para diversos tipos de desarrolladores de cursos, los principiantes de forma rápida y fácil lograrán construir un curso interactivo con el uso de la amplia gama de objetos prediseñados del Catálogo los que permiten integrar sonido, animación, video, gráficos y otros efectos especiales; mientras que los más experimentados además de esto podrán crear funcionalidad especializada con el uso de herramientas como *Actions Editor*, el lenguaje de programación *OpenScript*, *Simulation Editor* y otras opciones de diseño.

Instructor permite presentar el curso desarrollado de una manera flexible ya sea por Internet, en una intranet, o como una aplicación que se ejecute desde el disco duro o CD-ROM de un computador.

REQUERIMIENTOS DEL SISTEMA

Para ejecutar Instructor, la configuración del computador debe cumplir como mínimo los siguientes requisitos:

Componente	Mínimo	Recomendado
Computador (procesador)	Pentium 233	Pentium III 500 o superior
RAM (*Random Access Memory*)	64 MB para reproducir, 128 MB para desarrollo	128 MB o más
CD-ROM	De velocidad normal	Doble velocidad o superior
Espacio en disco duro	150 MB	310 MB para todos los componentes opcionales disponibles en el programa
Sistema Operativo	Windows 98 Segunda Edición, Windows Millennium, Windows 2000 Professional Service Pack 3, Windows XP Home, Windows	

	XP Professional	
Tarjeta de video	Adaptador gráfico de video (VGA) capaz de desplegar una resolución de 800x600 *pixels* con 16 bits (65.536 colores) para el equipo de desarrollo	VGA capaz de desplegar una resolución de 1024x768 *pixels* con 16 bits o más (65.536 colores)
Tarjeta de sonido	Opcional	Compatible con Windows

Tabla 1.1: Configuración del sistema recomendada y mínima [1]

CONCEPTOS BÁSICOS

LA FAMILIA DE PRODUCTOS TOOLBOOK

La familia ToolBook se compone de varios productos los cuales están orientados a la creación y distribución de cursos, entre los cuales se tiene:

- ToolBook Assistant
- ToolBook Instructor
- ToolBook Neuron

Toolbook Assistant

Diseñado para fácil uso, permite a usuarios no experimentados crear de manera sencilla cursos de alta calidad y con un contenido que integra medios interactivos gracias al amplio conjunto de objetos básicos de dibujo, gráficos y objetos preprogramados. Tanto el desarrollo del curso como su distribución se realizan sin la necesidad de programación.

Toolbook Instructor

Diseñado para crear aplicaciones de aprendizaje con amplio contenido interactivo y simulaciones avanzadas, que se adaptan a requisitos específicos. El contenido creado con ToolBook Instructor y ToolBook Assistant es compatible; la

personalización de catálogos, herramientas y otras ayudas de productividad para la autoría pueden ser creadas con ToolBook Instructor y usadas en ToolBook Assistant.

Toolbook Neuron

Permite acceder y ejecutar aplicaciones nativas de ToolBook Instructor y Assistant sobre Internet e intranets. Trabaja con el Web *browser* y está disponible en dos versiones: como *plug-in* para Netscape o Internet Explorer y como un control ActiveX[1]. Al navegar a una página que hace referencia a un libro ToolBook remoto, Neuron hará lo necesario para ejecutar el libro, incluyendo la transferencia de medios basados en *clips*[2], libros del sistema *y* DLLs *(Dynamic Link Libraries)* desde el servidor Web al computador.

Para proveer una referencia a un libro ToolBook nativo, se puede dejar que *ToolBook Web Specialist*[3] cree un enlace automáticamente, o escribir un archivo HTML (HyperText Markup Language). El tiempo de descarga de una aplicación ToolBook nativa se puede reducir usando *Impulse*, una herramienta que comprime y divide al libro en segmentos.

TOOLBOOK INSTRUCTOR

Las principales características y ventajas de ToolBook Instructor son las siguientes [3]:

- **Simulación de aplicaciones de *software*:** El editor y grabador de simulaciones permiten una rápida y sencilla creación de simulaciones.
- **Contenido interactivo y dinámico sobre la Web:** Presenta características y funciones diseñadas para tomar ventaja de la Web, además genera automáticamente la aplicación en HTML.

[1] Ver en *1.2.3.3.1*
[2] Cualquier archivo media: sonido, video, animación o imagen fija
[3] Prepara al libro ToolBook (archivo .TBK) para publicarlo en la Web

- **Robusto ambiente de creación de contenido:** *Actions Editor* es una herramienta de programación visual que permite extender la funcionalidad propia del programa, al igual que el lenguaje de programación *OpenScript* y otras herramientas disponibles.
- **Plantillas y *wizards* hacen más fácil el desarrollo:** Incluye una amplia selección de plantillas prediseñadas y un Catálogo con gran variedad de objetos preprogramados.
- **Evaluaciones incorporadas fácilmente:** Tiene más de diez tipos de objetos de evaluación, para probar el nivel de comprensión de los usuarios sobre el contenido del curso.
- **Métodos de presentación:** El contenido desarrollado en ToolBook puede mostrarse sobre Internet o una intranet, en una red de área local (LAN) o en un CD-ROM.

Tipos de aplicaciones

Instructor está orientado principalmente a la creación de cursos educativos y de entrenamiento; sin embargo, gracias a la flexibilidad del programa es posible crear otro tipo de aplicaciones como: simulaciones, demostraciones interactivas, *tours* dirigidos, presentaciones en línea, catálogos, juegos, etc.

Estructura de una aplicación

Antes de iniciar el desarrollo de una aplicación es aconsejable planear su construcción, para de esta forma aprovechar las facilidades de Instructor tanto de desarrollo como de presentación. Las aplicaciones creadas en Instructor se componen de uno o varios archivos llamados libros (*books*), los que deben residir en un directorio central con los subdirectorios para archivos de medios que acompañan estos libros.

Nivel de Autor y Nivel de Lector

Instructor tiene dos niveles de operación: nivel de Autor y nivel de Lector.

- **Nivel de Autor**: Permite crear y modificar aplicaciones usando herramientas de autor y de desarrollo tales como plantillas, catálogo, cuadros de diálogo, y varias paletas. Aquí se define el aspecto y comportamiento de la aplicación.
- **Nivel de Lector:** Este nivel se utiliza para comprobar cómo el usuario final verá la aplicación. A este nivel, los usuarios pueden navegar a páginas particulares, mecanografiar el texto en campos, responder preguntas de objetos interactivos, y a eventos especiales.

Figura 1.1: Nivel de Autor y Nivel de Lector

Libros y Páginas

Una aplicación creada con Instructor está formada de uno o más archivos llamados libros con acceso secuencial o por medio de hiperenlaces. Cada pantalla de un libro es una página separada, la que puede contener diversos elementos (botones, gráficos, texto, etc.). Se puede decir que una página es la unión de un *background* (fondo) y del *foreground* (primer plano de la página). Varias páginas pueden compartir un *background* y un libro puede contener varios *backgrounds*.

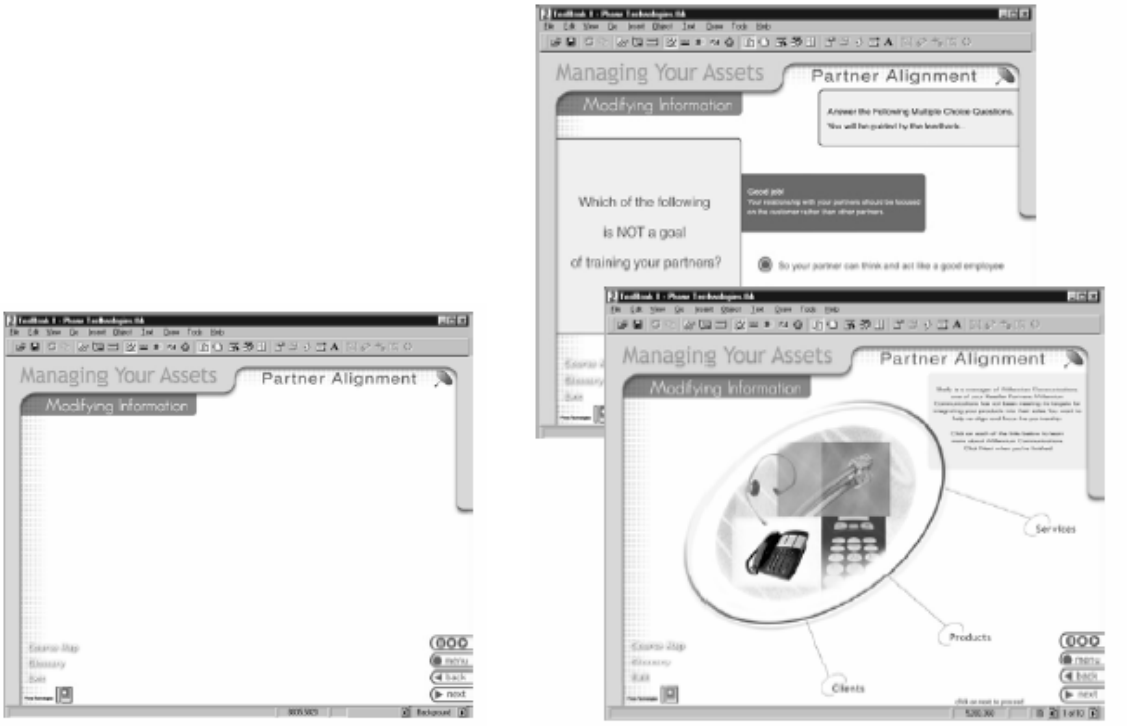


Figura 1.2: *Background* y páginas que comparten un *background*

Es posible crear visores (*viewers)*, ventanas secundarias que aparecen junto a la principal al igual que una ventana *pop-up*[1]. Los visores son objetos y como tales tiene un conjunto de propiedades que establecen su comportamiento y apariencia.

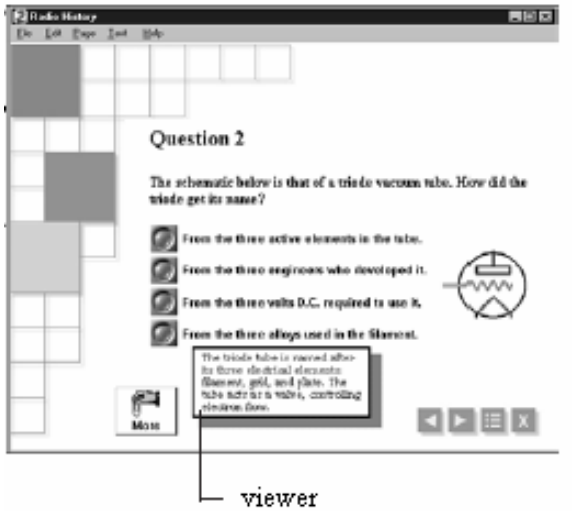


Figura 1.3: Página que usa visores (*viewer*)

Objetos y Propiedades

[1] Ventana flotante desplegada en la pantalla, sobre la ventana principal

Los elementos visuales de una aplicación como botones, texto, e incluso páginas y *backgrounds* son objetos, pueden ser creados usando la paleta de herramientas o arrastrados desde el Catálogo hasta la página. El comportamiento y aspecto de un objeto puede ser preprogramado o definido por medio del cuadro de diálogo *Properties o Extended Properties* (determina capacidades más sofisticadas). Para añadir comportamiento que no se encuentra como propiedad del objeto en los cuadros de diálogo, se puede usar *Actions Editor* u *OpenScript*.

Métodos de presentación

El método de presentación debe ser seleccionado antes de comenzar la aplicación, ya que cada uno de éstos requiere diferente preparación, la Tabla 1.2 muestra las consideraciones a ser tomadas para cada método.

Método de Presentación	Plataforma requerida	Pautas para la preparación
Vía Internet o Intranet como una serie de páginas Web que Incorporan DHTML (*Dynamic* HTML)	Microsoft Internet Explorer 5.5 SP1 o posterior; Nescape Navigator 7.1	• No usar *OpenScript*; en su lugar, usar *Actions Editor*. • Usar medios o archivos auxiliares compatibles con los estándares de Internet. • Usar *ToolBook Web Specialist* para exportar la aplicación.
Vía Internet o Intranet como una aplicación nativa de ToolBook desplegada en un Web browser usando Neuron	Windows 98, NT 4.0, 2000 (o posterior); Internet Explorer 5.5 SP1 Netscape Navigator 7.1	• Usar todas las características de Instructor para la creación de aplicaciones interactivas, incluyendo *OpenScript* y *Actions Editor*. • El usuario debe tener (o poder descargar) Neuron. • Usar *ToolBook Web Specialist* para preparar la aplicación.
Vía una red de área local como una aplicación	Windows 98, ME, NT 4.0, 2000, XP o posterior	• Usar todas las características de Instructor para la creación de aplicaciones interactivas,

ToolBook nativa		incluyendo *OpenScript* y *Actions Editor*. • Si los usuarios instalan la aplicación sobre sus propias computadoras, usar *ToolBook AutoPackager* para crear un programa de instalación.
Vía CD-ROM u otros medios de transferencia	Windows 98, ME, 2000, NT 4.0, XP o un sistema operativo posterior	• Usar todas las características de Instructor para la creación de aplicaciones interactivas, incluyendo *OpenScript* y *Actions Editor*. • Usar *ToolBook AutoPackager* para crear un programa de instalación.

Tabla 1.2: Preparar el proyecto para su distribución basado en el método de presentación [1]

OPCIONES ADICIONALES DE DISEÑO

Actions Editor

La herramienta de programación visual *Actions Editor* permite extender la funcionalidad de los objetos disponibles en el Catálogo sin el uso de *OpenScript*. Se puede crear secuencias de acción que son automáticamente exportadas a HTML, añadiendo acciones las que pueden ser por ejemplo: ejecutar un medio, mostrar una página, sugerir información, o cualquier otra. Una secuencia de acción responde a un evento, tal como un *click* de un botón, el despliegue de una página, o la selección de un ítem de una lista.

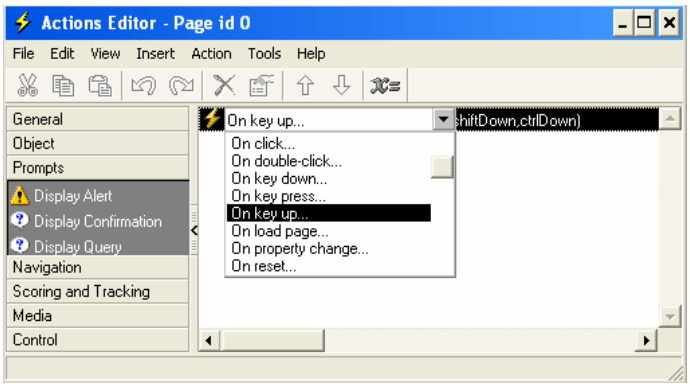


Figura 1.4: Ventana de *Actions Editor*

Openscript[1]

OpenScript es el lenguaje de programación de ToolBook, es de fácil uso debido a su sintaxis (similar al Inglés), su amplio rango de comandos, y su naturaleza orientada a objetos. Incluye comandos que realizan una gran variedad de tareas para definir el aspecto y funcionalidad de un objeto, algunas de estas tareas también pueden ser controladas con el *Action Editor* o desde las propiedades de los objetos del Catálogo.

OpenScript se integra completamente al ambiente ToolBook, se puede escribir un *script*[2] en el editor *ToolBook Script* o ingresar comandos directamente en la *Command window.*

La capacidad de *OpenScript* puede ser ampliada llamando a DLLs, las que son librerías de las funciones disponibles para cualquier programa de Windows como: mostrar ventanas, recibir mensajes, o determinar el estado actual del sistema.

Tecnologías Windows [5]

Instructor amplía su funcionalidad con el uso de ciertas tecnologías Windows, como las que se describen a continuación:

ActiveX

Los controles ActiveX son componentes de *software*, que se puede instalar para agregar cierto tipo de funcionalidad a una aplicación ToolBook o a una aplicación que se planea exportar a la Web. En el cuadro de diálogo *Extensions*, del menú *File* se puede observar una lista de controles ActiveX los cuales típicamente tienen una extensión .ocx (*Ole Control eXtension*) o .dll. Un control ActiveX puede

[1] Se estudiará en el Capítulo 2
[2] Un conjunto de instrucciones combinadas en un archivo, que definen la funcionalidad de los objetos en ToolBook

ser simple (como un calendario) o complejo (como un control Microsoft Web Browser). Algunos controles ActiveX pueden mostrar un comportamiento inesperado al ejecutarse en ambientes ToolBook ya que principalmente son diseñados para ser ejecutados sobre Internet.

OLE

Objetos OLE (*Linking and Embedding*) es una tecnología utilizada para integrar aplicaciones. OLE permite crear un objeto en una aplicación (servidor) y después incorporarlo en otra aplicación (cliente). Los objetos OLE pueden compartir información sobre un objeto OLE *linked* o *embedded* con el servidor de aplicación donde el objeto fue creado. Los servidores OLE típicamente tienen una extensión de archivo .exe. Usando OLE, se puede crear aplicaciones que integran capacidades de diversos programas de Windows. Los objetos OLE se diferencian de los controles ActiveX en que un objeto OLE tiene un servidor de aplicación que proporciona esta funcionalidad. Los objetos OLE y ActiveX son basados en el Microsoft COM (*Component Object Model*).

Automation

Automation es una tecnología que amplía la funcionalidad de Instructor, permite acceder y manipular objetos, propiedades, y métodos de otras aplicaciones Windows (servidor) desde una aplicación ToolBook Instructor (cliente). El cliente al conectarse con el servidor de la aplicación crea una instancia de esa aplicación como objeto. El objeto creado es tratado de la misma forma que cualquier otro objeto. *Automation* requiere una variedad de componentes Microsoft instalados en la computadora del usuario para ser ejecutada.

ADO (Activex Data Object)

ADO es un mecanismo de alto nivel que permite acceso a los datos en bases de datos de ToolBook usando *OpenScript*. Por ejemplo es posible enviar y recibir datos desde una base de datos Microsoft Access.

DDE (Dynamic Data Exchange)

DDE es un protocolo de comunicación de Windows, para intercambiar datos y comandos entre aplicaciones Windows. Por ejemplo, es posible enviar comandos de DDE desde una aplicación Instructor para producir cartas y gráficos en Excel, y además crear un interfaz más amigable desde Instructor.

LA INTERFAZ DE INSTRUCTOR

La interfaz de Instructor fue diseñado para fácil utilización. Como se explicó anteriormente Instructor maneja dos niveles de funcionamiento: nivel de Autor y nivel de Lector. Para cambiar entre el nivel de Autor y el nivel de Lector, presionar F3 o elegir Autor o Lector desde el menú *View*.

Trabajar a Nivel de Autor

En este nivel, se dispone de herramientas para el desarrollo como la barra de menú, barra de herramientas, barra de estado, paletas y el Catálogo. Al iniciar Instructor se ingresa a este nivel.

Es posible esconder y mostrar cada una de las barras y personalizar su apariencia.

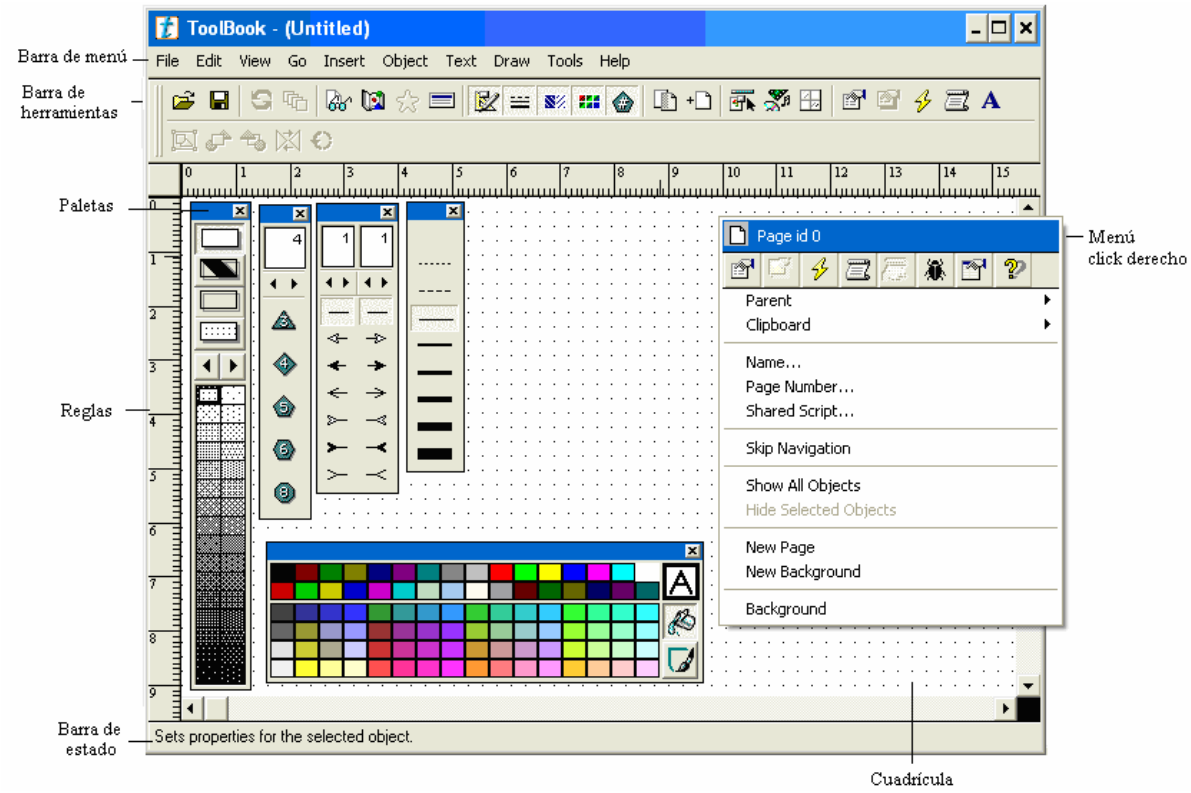


Figura 1.5: Herramientas a nivel de Autor

Barra de Menú (Menu Bar)

Permite ejecutar comandos y acceder a cuadros de diálogo. Su forma de trabajo es similar a cualquier barra del menú de Windows. Simplemente presionar el nombre del menú y escoger un comando *Browser*.

Barra de Herramientas (Tool Bar)

Proporciona una colección de botones que se puede utilizar como accesos directos a los comandos más comunes del menú[1]. Algunas posiciones contienen dos botones: un botón por defecto y un botón oculto que aparecerá cuando el usuario presione CTRL. Para ocultar y mostrar la barra de herramientas: desde el menú *View*, escoger *ToolBar*.

[1] Anexo A: Botones de comando de la barra de herramientas y sus funciones

Barra de estado (Status Bar)

Muestra el nombre de objetos, texto de ayuda para los comandos del menú, y la posición actual del *mouse*. El lado derecho de la barra proporciona una manera rápida de seleccionar la página actual y contiene herramientas que se puede utilizar para navegar entre las páginas. Para mostrar la barra de estado a nivel de Lector, presionar F12. A nivel de Lector, la barra de estado no incluye el indicador de selección de página.

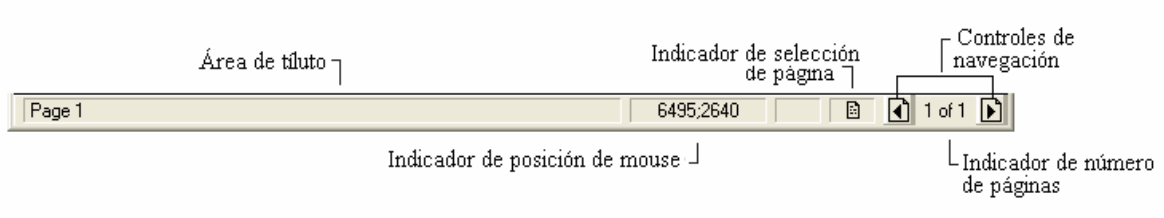


Figura 1.6: Barra de estado a nivel de Autor

Paletas

Son colecciones de herramientas o de botones que se utiliza para dibujar nuevos objetos o cambiar la forma de un objeto, el color u otras propiedades. Instructor incluye seis paletas, que se describen a continuación.

Herramientas (*Tool*): Crea nuevos objetos en una página.

Líneas (*Line*): Fija un acho de línea o borde alrededor de un objeto.

Flechas (*Line Ends*): Agrega flechas al inicio o final de una línea.

Patrones (*Pattern*): Fija un patrón o diseño para rellenar un objeto.

Color (*Color*): Fija el color de un objeto o texto.

Polígono (*Polygon*): Fija el número de lados del polígono a ser dibujado.

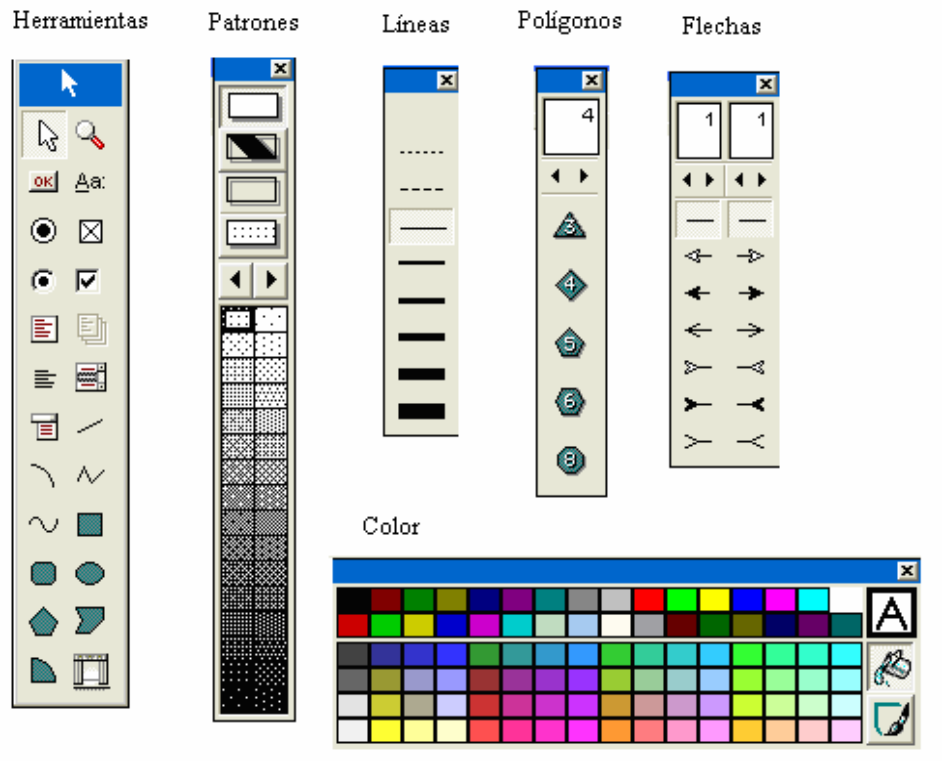


Figura 1.7: Paletas de Instructor

Catálogo

El Catálogo contiene objetos que pueden ser utilizados directamente en una aplicación, desde simple objetos de dibujo a sofisticados objetos interactivos. Contiene gráficos, paneles de navegación, objetos para preguntas, ejecutores de medios y otros objetos preprogramados que se puede utilizar para diseñar aplicaciones. Es posible crear objetos y guardarlos como parte del Catálogo. Para incluir un objeto en la aplicación, simplemente se lo arrastra desde el Catálogo hasta la página o *background*.

Coach

Proporciona avisos o sugerencias para el autor e instrucciones paso a paso mientras se trabaja. Su contenido cambia dinámicamente de acuerdo al objeto y página en la que se está trabajando. Para mostrar el *Coach*: desde el menú *Help*, escoger *Coach*.

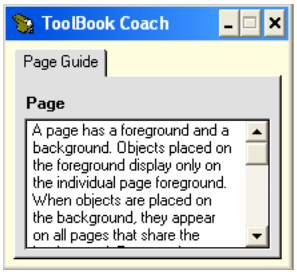


Figura 1.8: *Coach* de ToolBook

Propiedades y Propiedades Extendidas

Permite especificar las propiedades que determinan el aspecto y el comportamiento de un objeto. Para configurar las propiedades de un objeto primero se selecciona el objeto y entonces se abre el cuadro de diálogo *Properties*[1] o *Extended Properties*.

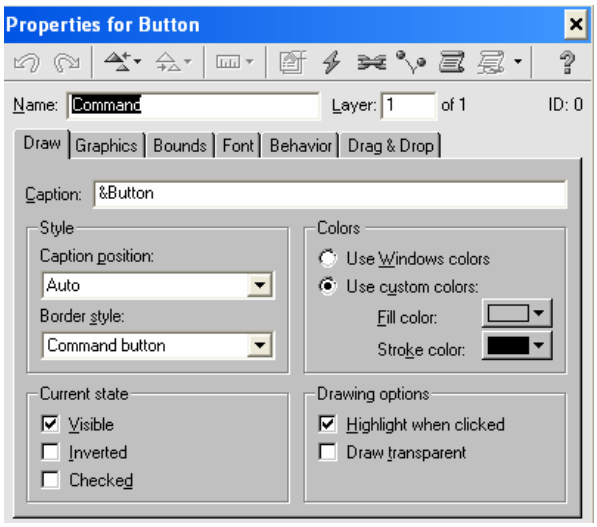


Figura 1.9: Cuadro de diálogo *Properties*

Toolbook Browser

Permiten ver y trabajar con el libro de una manera no lineal organizando el libro por sus propiedades, manipuladores, objetos o páginas. Para abrir uno de los *browsers*: desde el menú *View*, señalar *Browser*, y entonces escoger uno de estos *browsers*:

[1] Anexo B: Barra de herramientas en el cuadro de diálogo *Properties*

- ***Property Browser*:** Muestra y permite la edición de todas las propiedades disponibles para un objeto.
- ***Handler Browser*:** Permite la edición de *scripts* de manipuladores en la jerarquía de paso de mensajes para el objeto actual.
- ***Object Browser*:** Despliega y permite la edición de todos los objetos en la página y el *background* actual.
- ***Page Browser*:** Permite ver y organizar las páginas en un libro. Presenta una imagen en miniatura de cada página así como información sobre cada página.

Reglas (Rulers) y Cuadrícula (Grid)

Muestran el tamaño y posición exactos de objetos. Las reglas aparecen a lo largo de los bordes superior e izquierdo de la ventana principal; las sombras en la regla indican las dimensiones del objeto seleccionado. La cuadrícula es una matriz de puntos espaciados en igual distancia en la ventana principal, ésta nunca aparece a nivel de Lector. Desde el menú *View*, escoger *Rulers*, o presionar CTRL + R.

Right-Click Menus

Aparecen cuando se hace *click* derecho a un objeto, proporcionando un rápido acceso a las configuraciones y a los cuadros de diálogo comunes.[1]

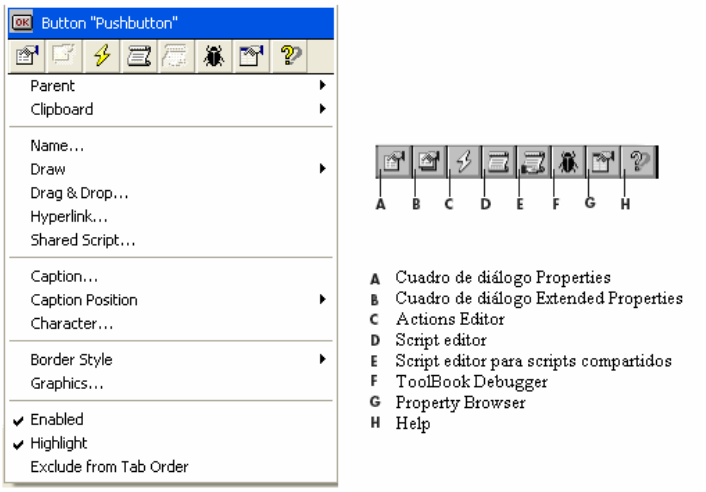


Figura 1.10: Ejemplo de menú *right-click* y la barra de herramientas del menú *right-click*

[1] Anexo C: Tabla de Acceso al menú *right-click*

Por defecto, se puede desplegar el menú *right-click* únicamente a nivel de Autor. Sin embargo, se puede cambiar la configuración por defecto para desplegar los menús *right-click* también a nivel de Lector.

Trabajar a Nivel de Lector

Para cambiar al nivel de Lector desde el nivel de Autor se debe ejecutar o probar una aplicación. Específicamente, Instructor hace los siguientes cambios:

- La barra de menú cambia a una barra de menú a nivel de Lector que no contiene comandos relacionados al desarrollo.
- La barra de herramientas, catálogo, *coach*, reglas, cuadros de diálogo, y cuadrícula desaparecen.
- La barra de estado y los menús *right-click* desaparecen, aunque es posible hacer que éstos sean disponibles.

CONSTRUCCIÓN DE UNA APLICACIÓN

A continuación se resume las formas de creación de libros y páginas, navegación entre páginas y cómo se mostrarán sus objetos.

CREACIÓN DE UN LIBRO

Un libro nuevo puede ser creado de tres diferentes formas:

- Con el uso de plantillas (*templates*).
- Como un libro en blanco.
- Usando *ToolBook Book Specialist.*

Para iniciar un libro nuevo es necesario seleccionar la manera de distribuir el curso: como un archivo ToolBook TBK nativo o como páginas Web (DHTML).

Usando una plantilla

Una plantilla es un libro preconstruido que provee una estructura básica para la aplicación (organización de páginas, gráficos, objetos interactivo y controles de navegación). Se puede utilizar plantillas incluidas con Instructor, o diseñar una propia guardando cualquier libro como plantilla. El uso de plantillas permite ahorrar tiempo y proporcionar un aspecto consistente para cada nueva aplicación.
Se puede escoger una plantilla desde el cuadro de diálogo *New Book* (escoger *New* desde el menú *File*) o desde el cuadro de diálogo *Startup* de ToolBook que aparece cuando se abre Instructor. Cuando selecciona una plantilla y teclea *OK*, Instructor abre una copia de la plantilla con un nombre y la guarda.

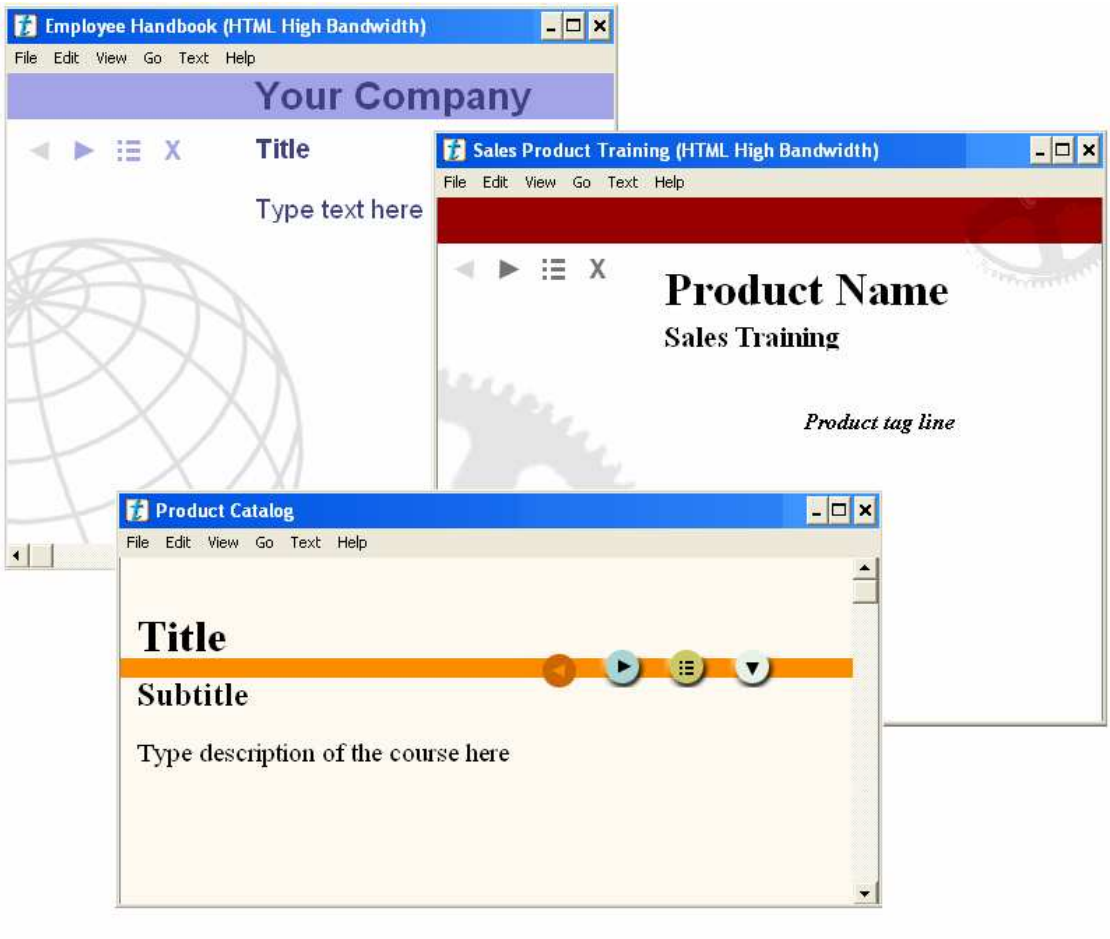


Figura 1.11: Plantillas

Empezar un nuevo, libro en blanco

Es posible empezar con un libro en blanco sin objetos preprogramados o temas gráficos. Para esto desde el menú *File*, escoger *New* y en *Quick Start* seleccionar el ícono *Blank DHTML Book* o *Blank Native ToolBook Book*.

Usando el *Book Specialist*

Permite iniciar rápidamente una aplicación básica ToolBook, recoge información del usuario sobre cómo quiere que el libro aparezca y funcione para construir uno basado en estas especificaciones. Instructor ofrece tres *Book Specialist* para ayudar a construir un nuevo libro:

- ***Full Specialist*:** Recoge especificaciones sobre el método de distribución, estilo de páginas y un diseño preliminar para la organización de páginas.
- ***Lesson Design Specialist:*** Utiliza datos a cerca del estilo de página, aspectos de aprendizaje, puede incluir archivos de texto, gráficos u objetos preprogramados.
- ***Quick Specialist:*** Se basa en los datos ingresados sobre unas pocas configuraciones seleccionadas. Es una versión simplificada de *Full Specialist*.

Para iniciar un libro nuevo usando uno de los *Book Specialists*: desde el menú *File*, escoger *New* y luego *Book Specialist*, seleccionar el ícono *Full Specialist*, *Quick Specialist*, o *Lesson design Specialist* y seguir las instrucciones.

Páginas y Backgrounds

Las páginas y *backgrounds* son objetos Instructor. Se puede añadir una página a un libro escogiendo el comando *New Page* o *Background* desde el menú *Insert*. Mientras que ambos comandos añaden una página al libro, cada uno de ellos da diferentes opciones sobre el tipo de página o *background* que se añade. También es posible crear páginas y *backgrounds* usando *OpenScript*.

Escoger *New Page* permite añadir una página vacía usando el actual *background*, duplicar una página desde un libro, o añadir una página desde una plantilla. Escoger *New Background* permite añadir una página que utiliza un nuevo *background* (con un único número de identificación ID). El *background* puede ser vacío, el de un libro o de una plantilla.

Importar y Exportar

Se puede importar libros ToolBook completos o páginas específicas desde aplicaciones ToolBook a un libro nuevo o existente.

Importar libros

Cuando se importa un libro, Instructor inserta todas sus páginas y *backgrounds* después de la página actual del libro actual, asignando a cada una un nuevo número ID. Para importar un libro escoger *Pages* desde el menú *Insert*, en el cuadro de diálogo *Import* seleccionar *ToolBook files*, aquí seleccionar el libro a importar.

Importar páginas

Se puede importar una página específica o un rango de páginas a un libro nuevo o existente. La página puede ser importada con su *background* o importarse sobre el *backgound* actual. Las páginas son insertadas después de la página actual del libro.

Para importar una o varias páginas escoger *Pages* desde el menú *Insert*, en el cuadro de diálogo *Import* seleccionar *ToolBook files*, aquí seleccionar *Format* y en el cuadro de diálogo *Book Format* escoger *Pages*, finalmente especificar el rango de páginas a ser exportadas. Si se quiere añadir simplemente una página desde otro libro, se puede usar la operación de copiado y pegado. Estos procesos también pueden realizarse con *OpenScript*.

NAVEGACIÓN A TRAVÉS DE UN LIBRO

Instructor presenta varias opciones para navegar entre páginas: usar los menús y barra de estado es decir la navegación incorporada, utilizar los objetos de navegación del Catálogo o con la ayuda del *Actions Editor* asignar navegación a un objeto. Además de la navegación entre páginas es posible crear un enlace a un URL (*Uniform Resource Locator*)[1].

MANEJO DE OBJETOS, TEXTO Y GRÁFICOS

OBJETOS

Todos los elementos visuales de una aplicación son objetos. Instructor contiene una completa colección de objetos en su Catálogo y además permite la creación de nuevos objetos usando la paleta de herramientas. Un objeto tiene propiedades específicas y puede contener *scripts* adjuntos.

A nivel de Lector sólo se observan los objetos de una página unificada, sin embargo, cada página puede estar formada de varias capas de objetos.

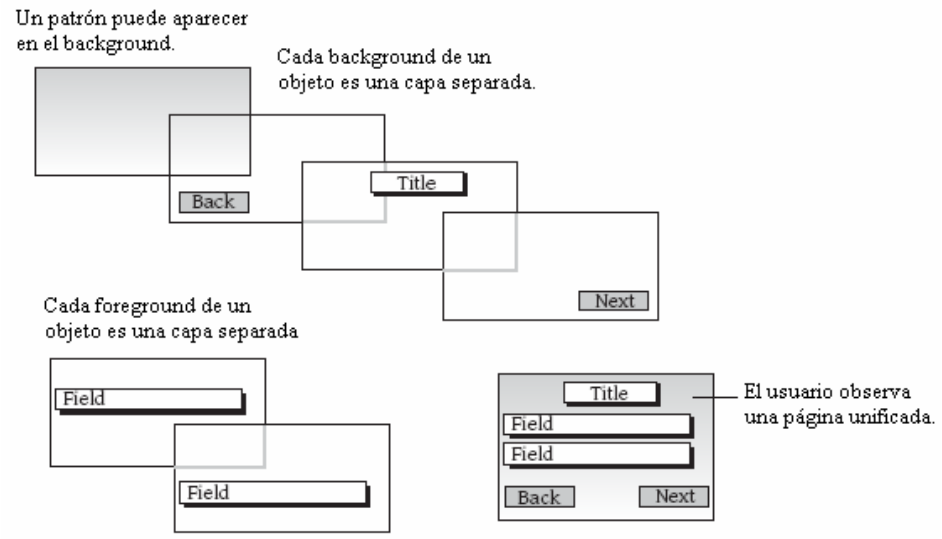


Figura 1.12: Las capas de una página

[1] URL: Dirección global de un recurso en el Internet

Cuando se crea un nuevo objeto, Instructor añade una capa al frente de la página o *background* actual y entonces ubica el nuevo objeto en la capa. Si existen objetos solapados, los objetos en capas superiores aparecen ubicados sobre los objetos de capas inferiores.

Objetos del Catálogo

La ventana del Catálogo contiene botones que muestran los nombres de las categorías de los objetos. Seleccionando el botón de la categoría se observa los objetos disponibles en la categoría.

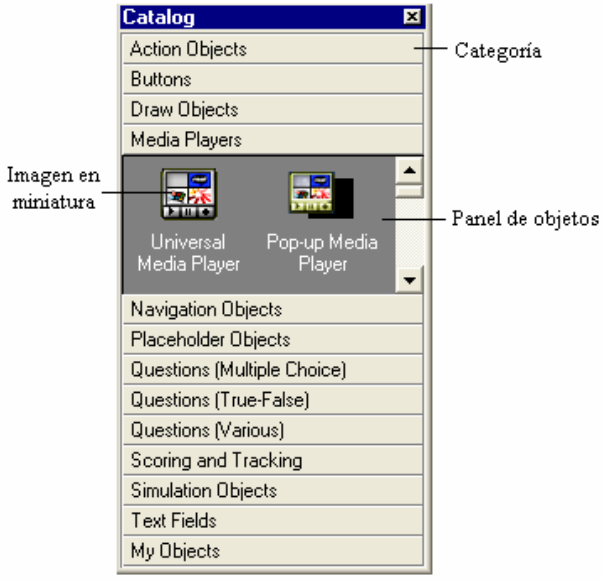


Figura 1.13: El Catálogo de ToolBook Instructor

Tipos de Objetos del Catálogo

A continuación se describe brevemente algunas de las categorías de objetos disponibles en el Catálogo.

- ***Actions Objects*:** Son objetos interactivos que pueden iniciar, activar, o restaurar acciones en una aplicación, los efectos interactivos pueden desencadenarse al teclear un *Action Object* o como consecuencia de un evento.

- ***Scoring y Tracking*:** Son utilizados para propósitos específicos en una lección de una página, por ejemplo: proporcionan el puntaje de una lección, reinician una lección, chequean las respuestas, etc.
- ***Placeholders*:** Añaden características específicas de Internet a una página cuando un libro se exporta a la Web, tales como ejecutar: un libro usando Neuron, un archivo GIF (*Graphics Interchange Format*) animado, o un *applet*[1] de Java.
- ***Media Players*:** Permitir ejecutar archivos de sonido, video, o animación.
- ***Navigation*:** Son botones y campos que permiten a los estudiantes moverse de una página a otra en la aplicación.
- ***Questions*:** Permiten crear preguntas para evaluar el aprendizaje. Todos los objetos de preguntas proporcionar *feedback* y evaluación de las respuestas.

Objetos de la Paleta de Herramientas

Una alternativa al uso de objetos predefinidos del Catálogo es crear nuevos objetos utilizando el *Tool Palette* los cuales tendrán propiedades preprogramadas limitadas. Al igual que cualquier otro objeto se podrá añadir funcionalidad con el *Actions Editor* y *OpenScript*. La forma del nuevo objeto estará definida por la herramienta utilizada en su creación.

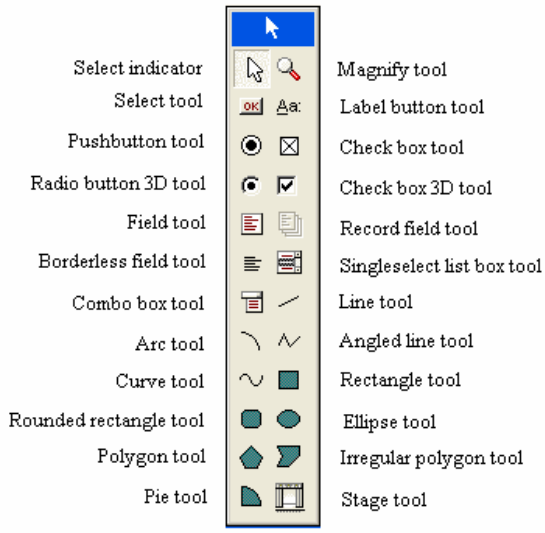


Figura 1.14: Tool Palette de Instructor

[1] Applet: pequeño programa diseñado para ser ejecutado desde otras aplicaciones, como un web browser

Trabajar con objetos

Mover

Para mover un objeto primero se lo debe seleccionar, y luego arrastrarlo a la posición deseada o utilizar las flechas del teclado para moverlo píxel por píxel.

Agrupar

Agrupar objetos permite darles un formato como una sola unidad. Cuando se crea un grupo, Instructor ubica este en la capa más baja ocupada por un objeto en el grupo. Cada grupo tiene su propio número ID, nombre, capa, *script* o secuencia de acción y propiedades. Se puede cambiar las propiedades para un grupo o para un objeto individual del grupo. Además es posible hacer un subgrupo de objetos dentro de un grupo existente. La agrupación se realiza seleccionando los objetos y presionando CTRL + G o escogiendo *Group* desde el menú *Object*.

Creación de Hiperenlaces

Los hiperenlaces permiten al usuario de la aplicación moverse desde una página a cualquier otra dentro o fuera de la aplicación actual. Todos los objetos ubicados dentro de una página tienen capacidades de hiperenlace preconstruidas.

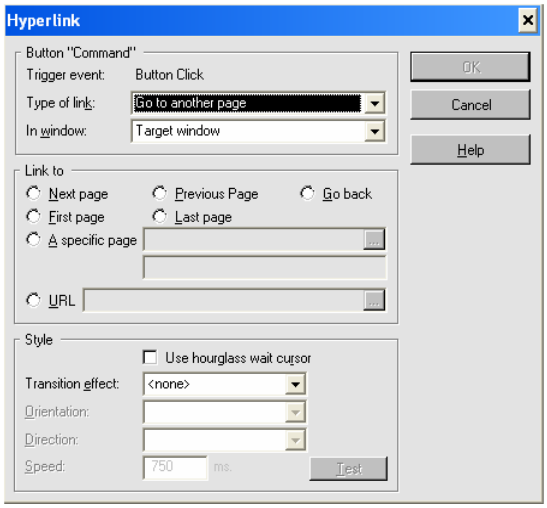


Figura 1.15: El cuadro de diálogo Hiperenlace

Cuando se crea un hiperenlace a otra página, se puede especificar un efecto de transición. El destino de un hiperenlace también puede ser un URL en el Internet.

Establecer propiedades

Mientras Instructor establece algunas propiedades, tales como el número de objeto ID, la mayoría son establecidas por el autor.

En general la mayoría de objetos presentan las siguientes lengüetas en el cuadro de diálogo *Properties*:

- ***Draw*:** Opciones de dibujo (estilo de borde, color y opciones de visibilidad).
- ***Bounds***: Opciones de posición y tamaño del objeto.
- ***Font*:** Tipo de letra, estilo, tamaño, efectos y opciones de formato.
- ***Paragraph*:** Opciones de alineación, interlineado, etc.
- ***Behaviour*:** Opciones de comportamiento en tiempo de ejecución.
- ***Drag & Drop*:** Opciones de apariencia y comportamiento cuando es arrastrado y soltado a nivel de lector.

TEXTO

Para añadir texto dentro de un libro se puede utilizar campos de texto y campos de registro, estos objetos pueden contener información e instrucciones. Un gráfico insertado dentro de un campo es llamado gráfico *inline,* puede ser un bitmap, ícono, o cursor; éste gráfico pasa a formar parte de los recursos del libro.

Campos de texto

Pueden ubicarse en una página o *background*; si se ubica en el *background*, el campo y el texto aparecerán sobre todas las páginas que comparten ese *background*, si se ubica en la página (*foreground*) el texto aparecerá únicamente en dicha página.

Importar: Es posible importar datos desde cualquier aplicación que cree archivos de texto y guarde estos en formato ASCII (*American Standard Code for Information Interchange*) o RTF (*Rich Text Format*). Al importar se mueven los datos desde un documento fuente a un campo de texto. Para importar seleccionar el archivo fuente desde el cuadro de diálogo *Import Text* que aparece al escoger la opción *Text*, desde el menú *Insert* mientras el objeto campo de texto está seleccionado.

Exportar: Cuando se exporta desde un campo de texto se mueve los datos a un archivo de texto. Para exportar se debe seleccionar el campo de texto y seleccionar el comando *Export* desde el menú *Tools*.

Campos de registro

Se ubica en un *background* y puede desplegar diferente texto en cada página que comparte ese *background*.

Importar: Los datos que se importan a un campo de registro deben estar en formato ASCII, este texto puede tener formato de archivo *fixed-field* o *delimited-field*.

- ***Fixed-field:*** Aquí se indica el número de campos de datos de un registro de datos especificando el número de caracteres en cada campo.
- ***Delimited-field:*** Puede contener campos de datos de cualquier longitud separados por un delimitador. Un delimitador es un simple carácter, tal como una coma o un tab, que separa los campos de datos en un registro de datos.

Para importar desde el menú *Insert*, escoger *Pages*, aquí aparece el cuadro de diálogo *Import* donde se selecciona archivos de texto (*.txt). *Click* en *Format* y escoger *Text-Fixed* o *Text-Delimited*, luego seleccione el archivo que se desea importar y *click* en *Ok*.

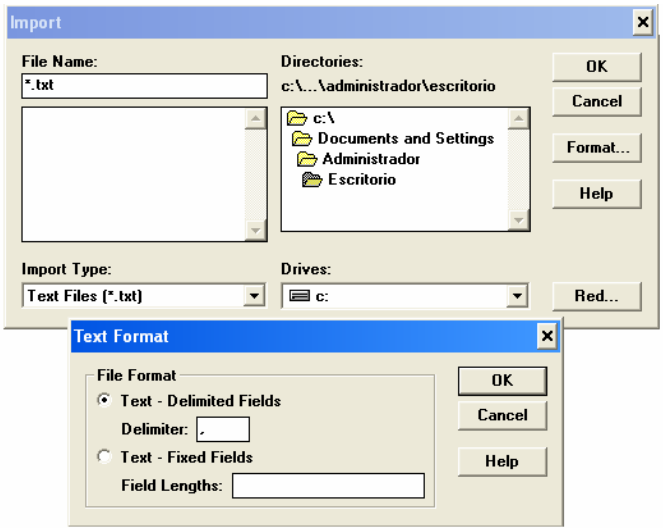


Figura 1.16: Cuadro de diálogo Import

Exportar: Se puede exportar un campo de registro de cualquier tamaño y exportar un número ilimitado de registros de datos desde un libro. Para exportar desde el menú *Tools*, escoger *Record Fields* en *Export*, y aparece un cuadro de diálogo *Export*, aquí especificar una ubicación y un nombre. *Click* en *Format* y en el cuadro de diálogo *Text Format* escoger *Text-Fixed* o *Text-Delimited* y sus respectivas especificaciones, *click* en *Ok* para finalizar.

GRÁFICOS

Un gráfico permite mejorar la apariencia de la aplicación, por ejemplo un botón o un *placeholder* puede mostrar gráficos y almacenarlos como recursos.

Picture object

Instructor almacena objetos *picture* en formato WMF (*Windows Metafile Format*), un formato de archivo para vectores gráficos. Un vector gráfico puede ser una imagen comprimida, una colección de líneas o una imagen *píxel-based* que es redimensionable. Contiene un conjunto mayor de instrucciones gráficas que *paint objects*.

Paint object

Instructor almacena objetos *paint* en formato de archivo *bitmap* (BMP). *Bitmaps* consiste de puntos pequeños que forman una imagen que es no redimensionable. Instructor despliega *paint objects* más rápido, pero requieren mayor espacio de disco que *picture objects*.

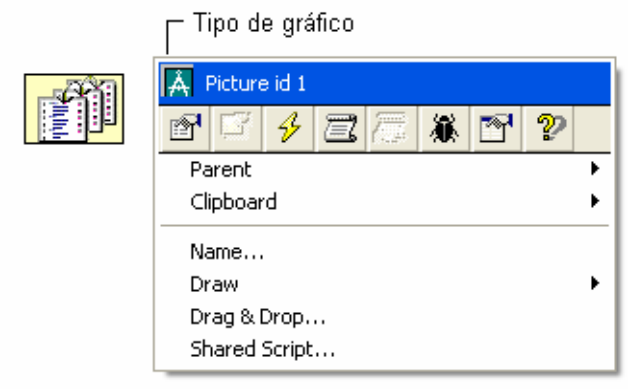


Figura 1.17: Menú *right-click* para un gráfico

Al publicar una aplicación en el Internet los gráficos deben ser compatibles con la *Web*. Los objetos *picture* y objetos de 8 bits son convertidos a archivos GIF. Los objetos *paint* que contienen más de 256 colores se convertirán a archivos JPEG.

Para importar un gráfico como recurso se debe utilizar el cuadro de diálogo *Resourse Manager*. Otra forma rápida de importar un gráfico en el caso que no se desee reutilizarlo es usar el comando *Graphic* en el menú *Insert*.

Para exportar un gráfico seleccionarlo, escoger *Graphic* en *Export* desde el menú *Tools*, en el cuadro de diálogo *Export Graphic* escribir un nombre para el archivo, el formato y el directorio.

CARACTERÍSTICAS INTERACTIVAS Y COMPONENTES MULTIMEDIA

Utilizando las diferentes herramientas que presenta ToolBook Instructor es posible agregar características interactivas y usar multimedia en una aplicación:

OBJETOS INTERACTIVOS

Botones y hotwords

Añadir botones y *hotwords* a una aplicación permite que los usuarios interactúen con el libro, por ejemplo que se ejecute una acción o que el usuario navegue a otra página, etc.

Botones

Un botón de Instructor es un objeto que funciona de la misma manera que un botón de Windows, al seleccionarlo desencadena una acción previamente establecida. La apariencia de un botón puede configurarse mediante sus propiedades, por ejemplo el borde, su estado (activado o desactivado), o la asignación de un gráfico.

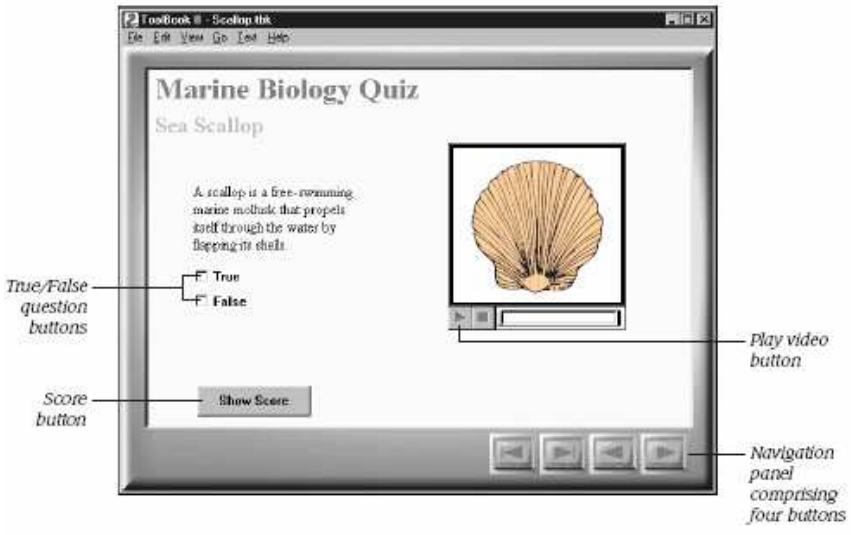


Figura 1.18: Diferentes tipos de botones

Usando la paleta de herramientas se tiene los siguientes tipos de botones:

- ***Pushbutton*:** Inicia una acción del usuario, como navegar de una página a otra.
- ***Label Button*:** Provee acceso rápido a la funcionalidad del botón, presionando ALT + la letra subrayada, también es usado para mostrar información sólo de lectura.

- ***Radio y Radio 3D Buttons*:** Usados en un grupo de alternativas donde solo una puede ser seleccionada.
- ***Check y Check 3D Boxes*:** Permiten que el usuario seleccione o no una opción.

Añadir un gráfico a un botón

Un botón tiene cuatro estados y a cada uno de éstos se puede añadir un gráfico diferente, utilizando la lengüeta *Graphics* del cuadro de diálogo *Properties* u *Openscript*.

Opción del cuadro de diálogo	**Propiedad *OpenScript***	**Descripción**
Normal	normalGraphic	El botón está en estado normal.
Invert	invertGraphic	El botón está siendo presionado.
Disabled	disabledGraphic	El botón está deshabilitado.
Checked (o Rollover)	checkedGraphic	El botón es chequeado, o como gráfico *rollover* cuando el *mouse* se detiene sobre el botón si esta opción es seleccionada en el cuadro de diálogo *Properties for Button.*

Tabla 1.3: Gráficos para cada estado del botón [1]

Hotwords

Un *hotword* está compuesto de una o más palabras en un campo de texto o registro, la misma palabra o frase dentro de un campo no puede pertenecer a más de un *hotword*. Gráficos *inline* pueden también ser *hotwords*.

Un *hotword* funciona como un botón y brinda flexibilidad en la construcción de libros interactivos complejos. A nivel de Lector el puntero cambia a una mano cuando se detiene sobre un *hotword*. Para crear un *hotword* se debe seleccionar el texto o gráfico *inline* deseado, escoger *Create Hotword* desde el menú *Text*, luego ir al menú *Object* y seleccionar *Hyperlink*, aquí especificar el tipo y destino del enlace. También se puede utilizar *OpenScript*.

Para observar a los *hotwords* existentes se debe escoger *Show Hotwords* desde el menú *View*. Si se desea remover un *hotword* se debe escoger *Remove Hotword* desde el menú *Text* o utilizar *OpenScript*, el texto se transforma a texto normal, se descarta su *script* y propiedades de hiperenlace.

List boxes y combo boxes

Los *list boxes* y *combo boxes* permiten mostrar al usuario múltiples opciones, de las cuales podrá hacer una selección y de acuerdo a ésta la aplicación mostrará un contenido específico o llevará al usuario a una página específica.

List boxes

Un *list box* se puede crear usando un campo de texto o registro (si se desea que la lista aparezca en varias páginas con el mismo *background*). Este permite que el usuario haga una selección dentro de la lista de ítems existentes, en el caso de que el tamaño de la lista sea mayor al espacio asignado se puede añadir un *scroll bar*.

Figura 1.19: *List box* con scroll bar

Un *list box* puede crearse arrastrando un objeto desde la categoría *Text Field* del Catálogo, o utilizando la herramienta *list box* desde la paleta de herramientas Instructor. Otra alternativa es utilizar *OpenScript.*

Combo boxes

Un *combo box* tiene tres partes: un *single-line edit box*, un *pushbutton*, y un *drop-down list box*. Al presionar el *pushbutton*, el *drop-dow list* aparece. Los usuarios pueden digitar en el *edit box* o seleccionar un ítem desde la lista *drop-down*. El

edit box muestra el ítem seleccionado actualmente y el *list box* se oculta. Por defecto un *combo box* no es editable, pero esta opción puede ser cambiada.

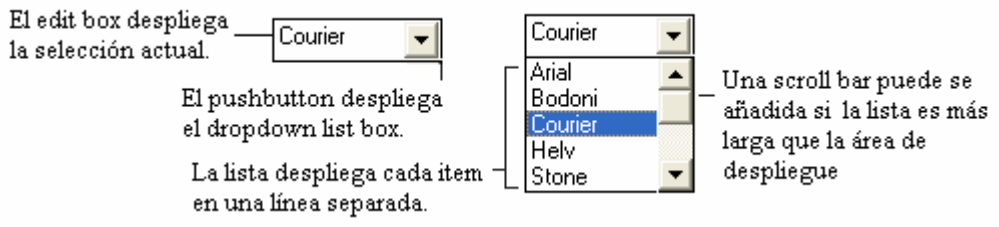


Figura 1.20: Combo box

Al igual que un *list box,* un *combo box* es posible crearlo utilizando el Catálogo, la paleta de herramientas u *OpenScript*.

Question objects

Question objects permiten evaluar los conocimientos que adquiere el estudiante a medida que avanza en el curso, con lo que se logra una aplicación más interactiva.

Entre las propiedades comunes de los *question objects* se tiene:

- Identificar que respuesta es correcta.
- Número de intentos de respuesta a la pregunta.
- Opciones de *feedback*.

Tipos

El Catálogo contiene una amplia variedad de *questions objects*, a continuación se describen algunos de ellos:

- ***Definable Arrange-objects:*** Permite que los estudiantes reconstruyan un objeto formado por piezas ubicadas aleatoriamente en la página, el usuario arrastra las piezas a la posición correcta.
- ***Drag-object:*** Permite definir varias opciones de respuesta y que el usuario arrastre la que crea correcta.

- ***Drop-target*:** Permite definir objetos como potenciales respuestas, el estudiante debe arrastrar las respuestas correctas al *drop-target object* (un gráfico de un contenedor u objetivo).
- **Fill-in-the-blank:** Permite que el usuario ingrese texto como respuesta a una pregunta.
- ***Match-items:*** Permite emparejar un conjunto de objetos a otro conjunto de objetos designados en una página, el estudiante arrastra una flecha desde cada objeto de respuesta a su correspondiente.
- ***Múltiple-choice:*** Permite que el usuario seleccione una o varias respuestas correctas de varias opciones de respuesta, para esto se puede usar un botón, campo, u otro objeto. Es posible habilitar una opción que hace que la respuesta correcta se ubique cada vez en diferente posición.
- ***Order-text:*** Permite que el usuario ordene una lista de ítems o palabras en una sentencia, los ítems son arrastrados a la posición correcta.

Scoring Questions

Scoring es un registro de las preguntas que el estudiante ha respondido correctamente, cada pregunta puede tener diferente puntuación, y cada respuesta de una pregunta puede tener asignado un diferente peso. Además es posible limitar el número de veces que el usuario intenta responder la pregunta.

El puntaje obtenido en una lección puede ser reportado al final del libro, a un archivo *log* (archivo de texto con información de la evaluación), o a un sistema de administración de aprendizaje que mantiene los puntajes de los estudiantes en una base de datos que el administrador puede examinar. Para habilitar *scoring* ir a la lengüeta *Scoring* del cuadro de diálogo *Extended Properties*.

Especificar *feedback*

Feedback es como el objeto responde cuando el estudiante interactúa con un *question object*. *Feedback* puede ser usado para una variedad de propósitos, por ejemplo para mostrar mensajes de ayuda o correcciones al estudiante de acuerdo

a su selección de respuesta, y para guiar al estudiante a través de una tarea o aplicación.

Para *question objects* se tiene varios tipos de *feedback*: ejecutar un *clip* multimedia, desplegar texto, navegar a otra página, navegar a un URL. Se puede combinar estos métodos y usar más de un tipo de *feedback* en respuesta a la selección del usuario.

CREACIÓN DE EFECTOS VISUALES

Un efecto visual puede lograrse con esconder y mostrar objetos, o con animaciones que demuestren cambios en el tiempo o concatenación de eventos. Al crear efectos visuales en una aplicación se logra hacerla más dinámica e interactiva.

Instructor contiene algunas herramientas para la creación de efectos visuales: *Animation Editor*, *Actions Editor* y *OpenScript*.

Animation Editor

Se usa el *ToolBook Animation Editor* para crear animaciones *path-based* o *cel-based* para objetos[1]. Instructor copia todos los objetos de la página (o *background*) al *Animation Editor* cuando se desea crear una animación. El *Animation Editor* presenta el siguiente interfaz y herramientas:

[1] No se pueden animar páginas, *backgrounds* y libros

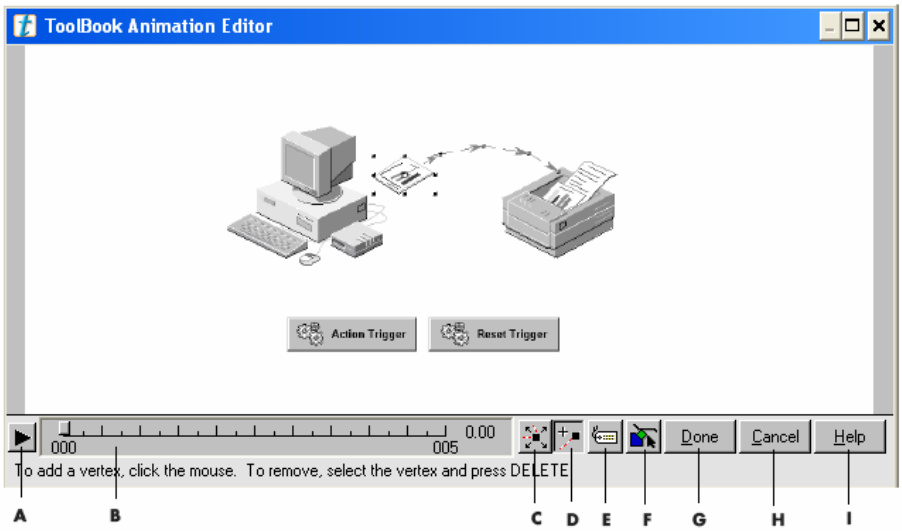


Figura 1.21: ToolBook Animator Editor

A Ejecuta la animación actual.

B Permite mover un objeto a un punto específico en la trayectoria de animación.

C Permite mover segmentos, vértices, o la trayectoria entera.

D Permite añadir nuevos segmentos a la animación.

E Abre el cuadro de diálogo *Animation Settings.*

F Abre el cuadro de diálogo *Select Object.*

G Guarda la animación actual y cierra el *ToolBook Animation Editor.*

H Cancela cambios en la animación actual y cierra el *ToolBook Animation Editor.*

I Abre ayuda en línea.

Animación Path-based

Permite animar objetos dibujando la trayectoria que seguirá. La trayectoria es indicada con el *mouse* y el objeto saltará desde un punto a otro sobre ésta a una velocidad especificada.

Para crear una animación *path-based*, seleccionar el objeto que se desea animar, escoger *Path Animation* desde el menú *Object*. El *Animation Editor* aparece con todos los objetos de la página actual, aquí mover el cursor *Vertex* y presionar el *mouse* para añadir segmentos hasta que se haya completado la trayectoria. *Click* en *Done* para guardar y regresar a la ventana principal.

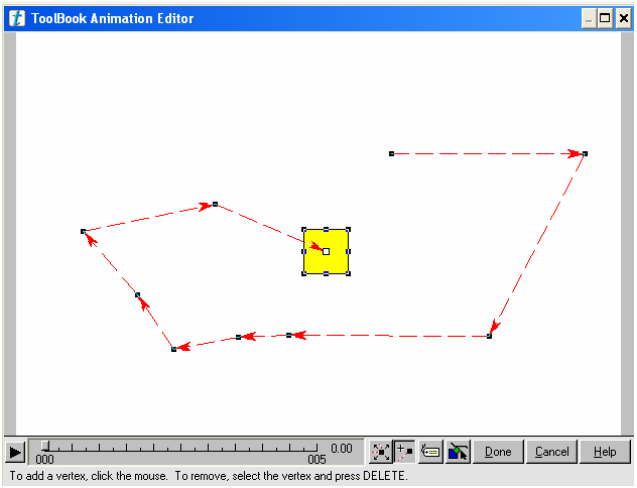


Figura 1.22: Animación Path-based

Animación Cel-based

Permite simular movimiento definiendo vistas individuales de un objeto, llamadas *cels*, que se muestran en rápida sucesión.

Para crear una animación *cel-based*, primero crear los objetos individuales (*cels*) en la ventana principal, la primera *cel* debe estar en la capa más baja y la última *cel* debe estar en la capa más alta. Seleccionar todos los objetos y desde el menú *Object* escoger *Group* para agrupar los objetos, en las propiedades ingresar un nombre al grupo. Seleccionar el grupo y escoger *Path Animation* desde el menú *Object*, aquí se puede marcar una trayectoria como en el caso anterior. *Click* en el botón *Animation Settings*, señalar en el menú *Cel animation* y cerrar. *Click* en *Done* para guardar y salir del *Animation Editor*.

Actions Editor [4]

La herramienta de programación visual *Actions Editor* permite crear secuencias de acción, añadiendo funcionalidad interactiva a los objetos de una aplicación. *Actions Editor* presenta un interfaz familiar con menús y una barra de herramientas.

Se puede crear dos tipos de secuencias de acción: una secuencia de acción para un solo objeto que responde a un evento específico o una secuencia de acción compartida que usan varios objetos y no está enlazada a un evento específico.

Para acceder al *Actions Editor*, seleccionar el objeto y escoger *Actions* desde el menú *Object* o en la barra de herramientas de Instructor. Otra forma de abrir el *Actions Editor* es seleccionar *Actions* desde el menú *right-click*. Además seleccionando el objeto y presionando F5 el *Actions Editor* aparece.

Cada objeto soporta diferentes eventos, el *Actions Editor* modifica su interfaz de acuerdo al objeto seleccionado y las acciones creadas para éste.

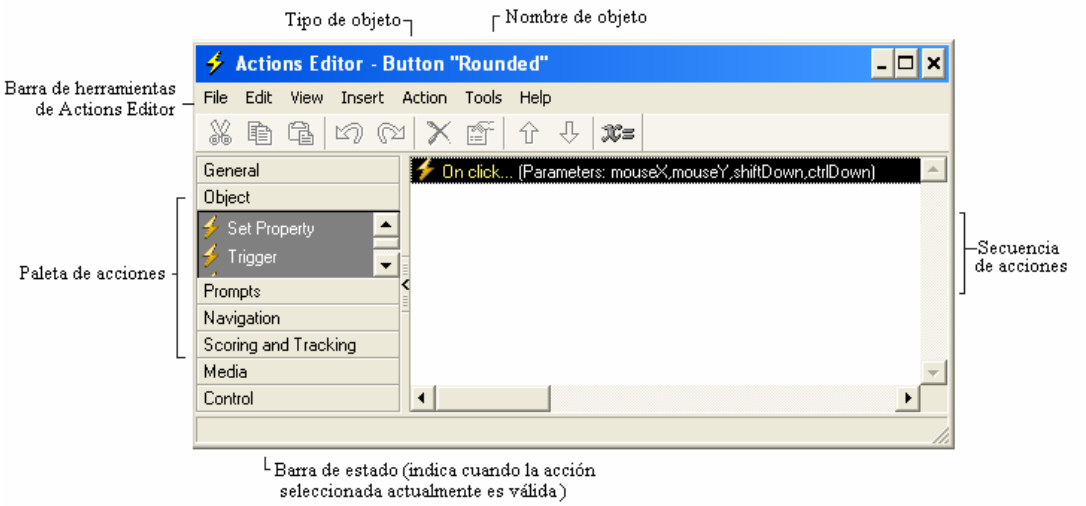


Figura 1.23: Interfaz de *Actions Editor*

Secuencias de Acción

Una secuencia de acción incrementa la interactividad de la aplicación. Se puede crear una secuencia de acción, añadiendo acciones a cualquier objeto de la aplicación en un orden determinado, estas acciones responden a un evento tal como un *click* en un botón. Una secuencia de acción puede permitir que un usuario ingrese datos, cambie las propiedades de un objeto, ejecute medios, y mucho más. Además se puede añadir condiciones, lazos y usar variables en una secuencia de acción.

Las ventajas de usar secuencias de acción creadas en *Actions Editor*, son las siguientes:

- Trabajan en aplicaciones ToolBook nativas y aplicaciones exportadas a DHTML.
- Permite crear comportamiento interactivo más complejo que los objetos del Catálogo: creación de lazos y comportamiento en respuesta a condiciones específicas, manipulación de eventos especiales como cuando se cambian las propiedades de un objeto.
- La aplicación puede exportarse a DHTML sin perder su funcionalidad, mientras que si se usa *OpenScript* no funcionará en el Web cuando se exporta a DHTML usando el *ToolBook Web Specialist*.

Creación de secuencias de acción usando el *Actions Editor*

Cada acción que forma parte de la secuencia es ingresada individualmente, su orden y funcionalidad puede ser alterada en cualquier tiempo, pero al ejecutarse Instructor se basa en el orden que las acciones tienen en la ventana *Actions Editor*.

En una secuencia de acción aparecen *hotspots* (texto en color y subrayados), estos presentan opciones con las que se edita su configuración. Dependiendo del tipo de objeto el *Actions Editor* presenta distintas opciones, entre las más importantes se encuentran las de control que permiten establecer lazos y condiciones para la ejecución de la secuencia de acción.

Secuencias de Acción Compartidas

Una secuencia de acción compartida no es enlazada a un evento específico y no pertenece a un objeto individual. Se puede usar una secuencia de acción compartida para asignar el mismo comportamiento a múltiples objetos. Una secuencia de acción compartida es almacenada bajo un nombre único que se asigna y es disponible para ser ejecutado como parte de cualquier otra secuencia de acción.

Cada una de las secuencias de acción pueden ser usada en varias ocasiones dentro de una aplicación o incluso ser importada a otro libro, su uso es similar a lo que en otros programas serían las funciones o subrutinas. Es posible definir parámetros y variables los que permiten realizar una función, y si es necesario retornan un valor, al igual que una función.

Creación de una secuencia de acción compartida

La creación de secuencias de acción compartidas es similar a crear una secuencia de acción de un objeto. Desde el cuadro de diálogo *Shared Actions* se puede manejar secuencias de acción compartidas, para acceder a este cuadro de diálogo se tiene varias opciones: desde *Object* en el menú principal de Instructor, desde *Actions* en el *Actions Editor*, o con Shift + F5.

MULTIMEDIA

Mostrar en una aplicación texto, imágenes, videos, animaciones y sonido de forma integrada puede hacer que el aprendizaje sea divertido y a la vez más efectivo. Si la aplicación generada en Instructor va a ser exportada a Internet, se debe considerar los formatos de archivos compatibles para su publicación.

Instructor presenta varias herramientas para ejecutar archivos multimedia, una de las principales y más completas es el *Universal Media Player*. [1]

Clip Manager

Se puede usar el cuadro de diálogo *Clip Manager* para crear, seleccionar, editar, duplicar, remover, y ejecutar los clips media en un libro.

Para acceder al *Clip Manager* desde el menú *Object* escoger *Clips*. Otra forma de abrir el *Clip Manager* es en el cuadro de diálogo *Extended Properties* del media

[1] Anexo D: Tipos de *Media Player*

player, seleccionar *Choose a Clip* y en el cuadro de diálogo que aparece, *click* el botón *Clips*.

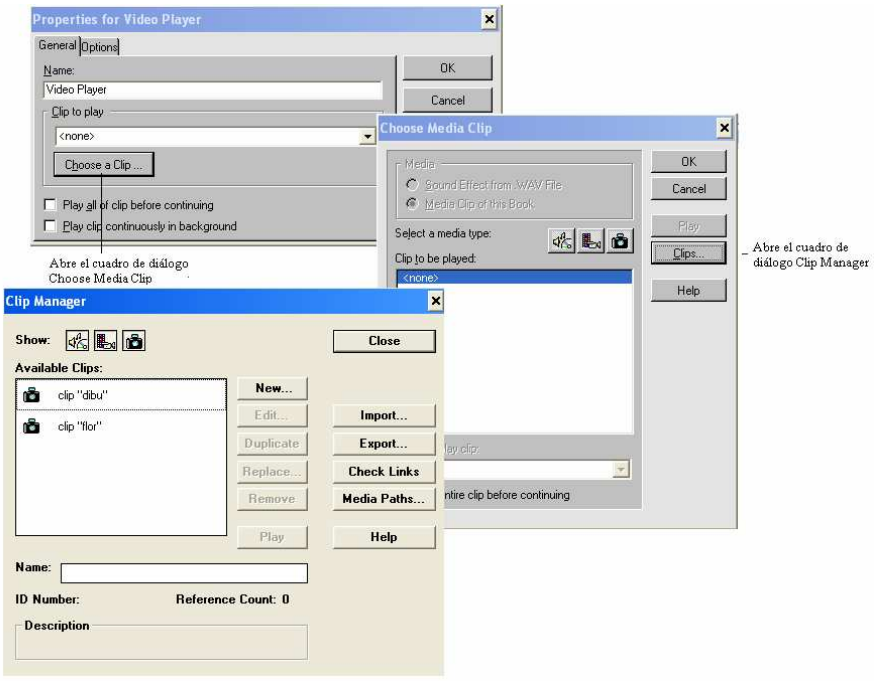


Figura 1.24: Crear un *clip*

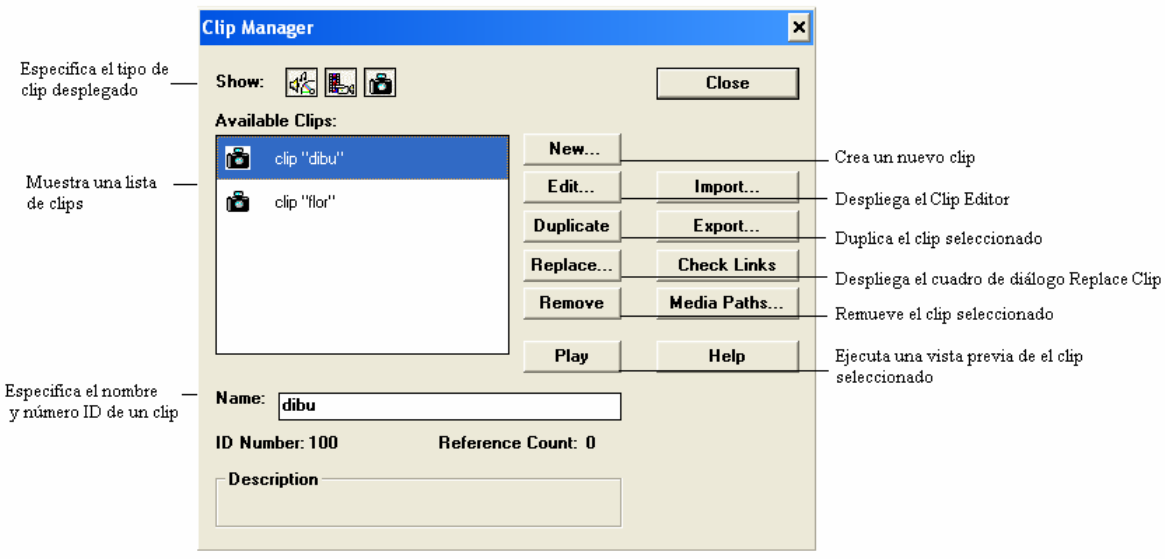


Figura 1.25: El cuadro de diálogo *Clip Manager*

RECURSOS

Los recursos son archivos que pueden ser compartidos y utilizados varias veces dentro de una aplicación ToolBook Instructor sin incrementar el tamaño del archivo[1]. Los recursos de Windows que se pueden usar son: *bitmaps*, cursores, íconos, *color palettes*, y *TrueType fonts*. Para poder utilizar un recurso se debe importarlo al libro y definir éste como una propiedad de un objeto. El *Resource Manager* es una librería común de recursos en el libro, es decir es el encargado de almacenar todos los recursos de una aplicación.

Añadir recursos a un libro

Existen varias maneras:

- Crear un nuevo recurso y añadir éste al libro.
- Importar un archivo como recurso.
- Copiar un recurso desde otro libro.
- Añadir un objeto que incluye un recurso desde el Catálogo. Instructor automáticamente importa el recurso dentro del libro actual.
- Usar *OpenScript* para referirse a un recurso en otro libro. Instructor automáticamente importa el recurso en el libro actual.
- Insertar un gráfico *inline* dentro de un campo o campo de registro.
- Importar una página Instructor o un libro entero que contenga recursos.

CREACIÓN DE SIMULACIONES [4]

Instructor presenta dos poderosos instrumentos para creación de simulaciones de *software*: *Sim AutoBuilder* y *Simulator Editor*, estas herramientas permiten capturar el comportamiento de un programa de *software*, logrando con gran exactitud simular el ambiente real del programa, ésto permite que el estudiante acceda al interfaz real del *software*, gane familiaridad y aprenda a usarlo, pero bajo un ambiente seguro.

[1] Anexo E: Tabla de tipos y propiedades de recursos

Sim AutoBuilder: Permite capturar el comportamiento de una aplicación de *software*, este archivo puede ser abierto y modificado en ToolBook, utilizando los objetos y herramientas que ToolBook presenta.

Simulator Editor: Permite crear los pasos que definen el comportamiento de una simulación, ya sea de un archivo creado en *Sim AutoBuilder* o de una simulación realizada manualmente con los objetos y gráficos de ToolBook.

Para crear simulaciones de forma manual en una página en blanco se ubican los objetos y gráficos que representan interacción en el *software* simulado, manejando de forma correcta las propiedades de los gráficos que pueden ser establecidas con la ayuda del *Simulation Editor*, como si es o no visible en un determinado instante se puede lograr simular ciertas acciones de la aplicación de *software*. Instructor además presenta objetos específicos para crear simulaciones en la categoría del Catálogo *Simulation Objects*, aquí existen objetos que permiten crear menús, cuadros de texto, campos entre otros.

Características de una simulación realizada en ToolBook:

- Puede contener un ilimitado número de pasos.
- Puede extenderse a varias páginas.
- Es posible incluir *feedback* en cualquier lugar de la simulación.
- Programación realizada en el *Actions Editor* puede ser incluida en la simulación.
- La simulación puede distribuirse como aplicación DHTML o ToolBook nativa.
- Se puede hacer un seguimiento de los resultados de la interacción del estudiante con la simulación, utilizando un sistema de administración de aprendizaje.

Para desarrollar una simulación se debe haber definido con anterioridad su propósito, el nivel de aprendizaje que se desea lograr en el estudiante, el tiempo que le tomará al estudiante revisar la simulación. Esto le ayudará a escoger las tareas que se van a simular y el modo de simulación que se va realizar. En

definitiva la planificación le ayudará a orientar su esfuerzo a satisfacer las necesidades del usuario.

MODOS DE SIMULACIÓN

Una simulación creada en Instructor puede ser mostrada en uno o más de estos modos:

- **Modo Práctica:** Permite que el estudiante interactué con el *software*, experimente con botones, menús y otros objetos en el interfaz simulado, es decir se puede probar cada paso por un limitado o ilimitado número de intentos.
- **Modo Evaluación:** Permite escoger una de varias opciones como respuesta por cada paso y asigna un puntaje para cada ingreso del usuario, de esta forma se pone a prueba los conocimientos del estudiante.
- **Modo Demostración:** Permite mostrar cómo completar la tarea, no requiere que el usuario ingrese datos o interactúe con el programa.

SIM AUTOBUILDER

Con el uso de *Sim AutoBuilder* se puede reproducir la aplicación de *software* original, registrar por pasos los eventos realizados en la aplicación como *click* del *mouse*, ingreso de datos y otras interacciones.

Crear una simulación con el Sim AutoBuilder

Primero se debe abrir la aplicación a ser capturada. Luego abrir el *Sim AutoBuilder Recorder* desde Inicio, Programas, ToolBook, Instructor 2004. Aquí aparece el siguiente cuadro de diálogo, donde se escoge el programa de *software*.

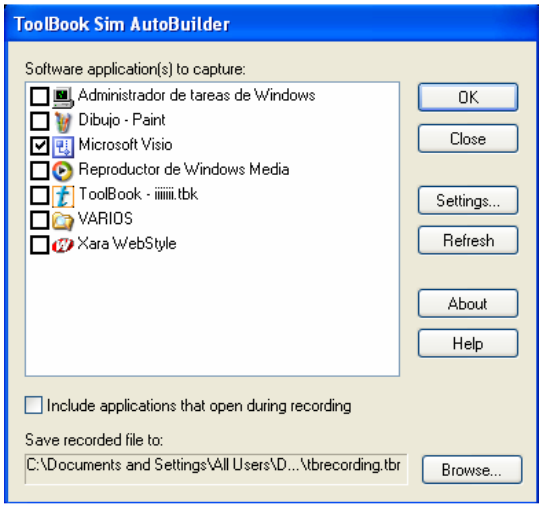


Figura 1.26: El Sim AutoBuilder

Para empezar la grabación presionar la tecla *Print Screen* (o la tecla especificada). Para finalizar la captura de un paso presionar nuevamente la tecla *Print Screen*, generalmente ésto se realiza luego de enviar un comando o instrucción en la aplicación de *software*. Repetir este procedimiento hasta completar los pasos deseados, y para terminar la grabación presionar CTRL + *Print Screen*.

Para visualizar la grabación en ToolBook abrirla desde *Simulation from Recording* desde el menú *Insert*.

Mientras se está grabando, presionar la tecla *Print Screen* después de:

- Navegar a otra página.
- Abrir un cuadro de diálogo.
- Seleccionar un ítem del menú.
- Doble *click* en un ícono.
- Realizando una acción que hace a la aplicación de *software* responder de una cierta manera.

Cada paso registrado debe contener una acción que ejecute una tarea específica, en el caso de que se requieran múltiples acciones, éstas deben estar

relacionadas como por ejemplo el ingreso de texto y presionar la tecla *Ok* para realizar una tarea.

Los eventos tales como teclear un botón del *mouse* (izquierdo, derecho o central), doble *click*, seleccionar una de entre varias opciones en menús, *list boxes* o *combo boxes*, presionar teclas de funciones como F8, tab o Shift, serán registradas por el *software* de captura *Sim AutoBuilder*.

SIMULATION EDITOR

Como se mencionó anteriormente el *Simulation Editor*, permite crear nuevas simulaciones y editar simulaciones existentes.

El *Simulation Editor* muestra todos los pasos de la simulación en su panel superior, y en su panel inferior se muestra las propiedades del ítem seleccionado en la parte superior.

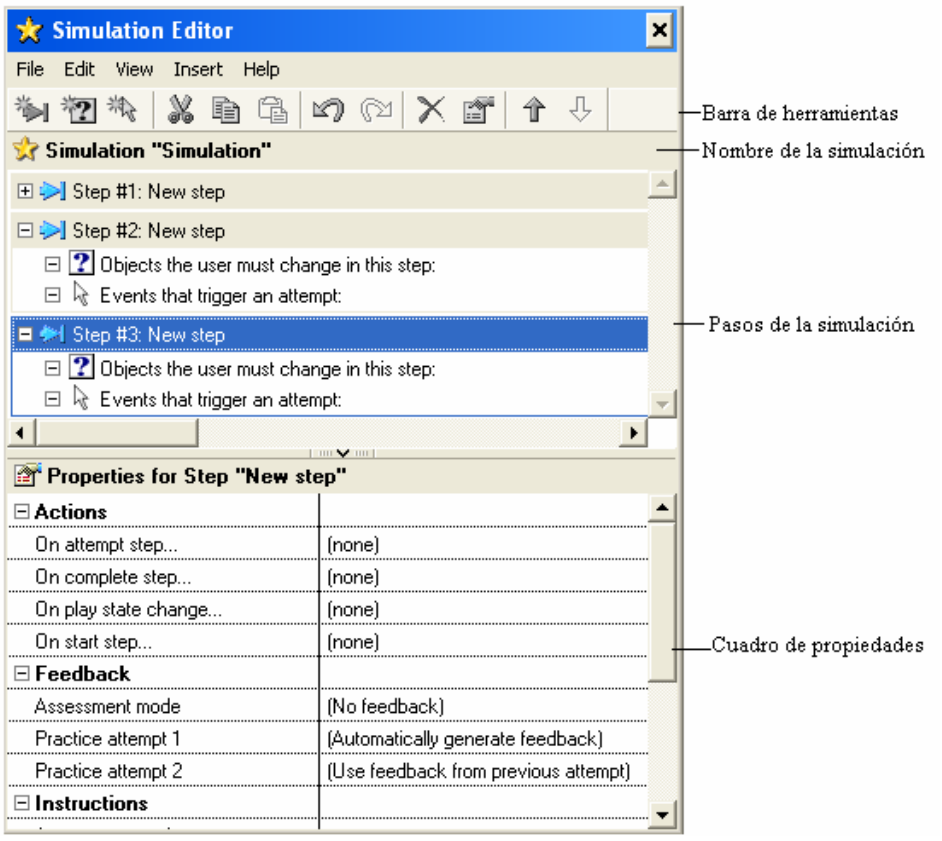


Figura 1.27: El Simulation Editor

Utilizar el Simulation Editor

Para abrir el *Simulation Editor* escoger *New Simulation* desde el menú *Insert*. Para volver a abrirlo seleccionar *Simulation Editor* desde el menú *View*. Un nuevo paso en la simulación puede ser creado presionando el botón *New Step* en la barra de herramientas del *Simution Editor* o escogiendo *New Step* en el menú *Insert*.

Un paso generalmente incluye instrucciones para orientar al estudiante, es posible incluir instrucciones generales para todos los pasos o específicas a un paso individual. Se puede ingresar texto para un paso individual en el cuadro de diálogo *Properties for Step*.

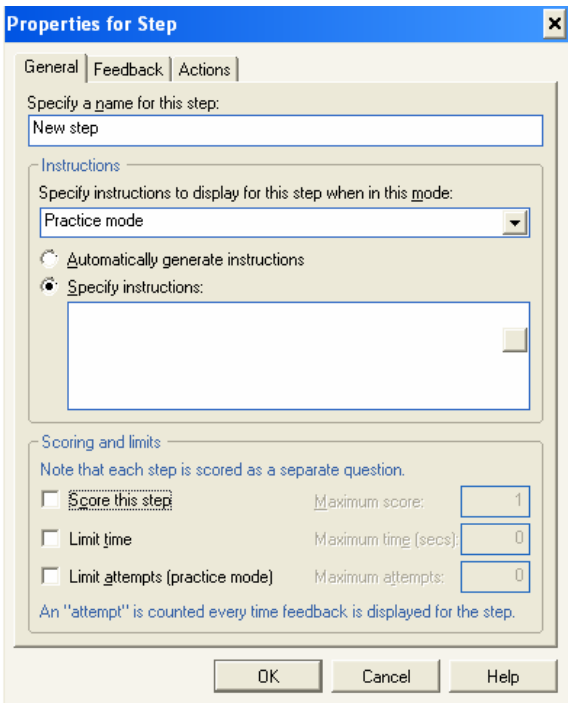


Figura 1.28: Ingresar instrucciones específicas para un paso en la simulación

CAPÍTULO 2

LENGUAJE DE PROGRAMACIÓN OPENSCRIPT

COMPONENTES BÁSICOS

OpenScript es el lenguaje de programación orientado a objetos de ToolBook, permite añadir a la aplicación funcionalidad más avanzada que la que obtiene utilizando las herramientas preconstruidas de ToolBook.

Características y ventajas de OpenScript

- Amplio rango de comandos, lo que permite realizar una variedad de tareas.
- Sintaxis similar al idioma Inglés, facilitando su utilización.
- Naturaleza orientada a objetos.
- Requiere menor tiempo y esfuerzo que otros lenguajes de programación como C o C++.
- Se integra completamente a aplicaciones creadas en ambientes ToolBook.
- Contiene herramientas de depuración de *scripts* como el *ToolBook Debuggger*, para realizar un seguimiento de ejecución de las instrucciones.
- Permite el manejo de eventos, ya que ToolBook al igual que Windows es activado por eventos, cada uno de éstos crea mensajes que son enviados al objeto para ejecutar una acción.

A continuación se presenta una tabla comparativa de *OpenScript* con otros lenguajes de programación:

OpenScript	C	BASIC	Pascal
if/then/else	if/else	If/Then/Else	If/Then/Else
Step	for	For/Next	For/Do
While	while	While/Wend	While/Do
Do/until	do/while	Do/Loop Until	Repeat/Until

conditions/when	Switch/case	Select Case/Case	Case/Of
Increment	+ = inc		
Decrement	– = dec		
send <message>	void function	Sub	procedure
to get <message>	Function	Function	function
System books	Libraries	libraries (or DLLs)	units
System variables	Global variables	global variables	global variables
break	break, return		
Format (number)	printf	print using	

Tabla 2.1: *OpenScript* comparado con C, BASIC, y Pascal [2]

HERRAMIENTAS DE DESARROLLO DE OPENSCRIPT

Script Editor

Se usa para ingresar o editar instrucciones, y chequear la sintaxis del *script*. Para abrirlo utilizar el acceso directo desde la barra de herramientas de ToolBook, desde el menú *right-click* del objeto, o CTRL y doble *click* en el objeto.

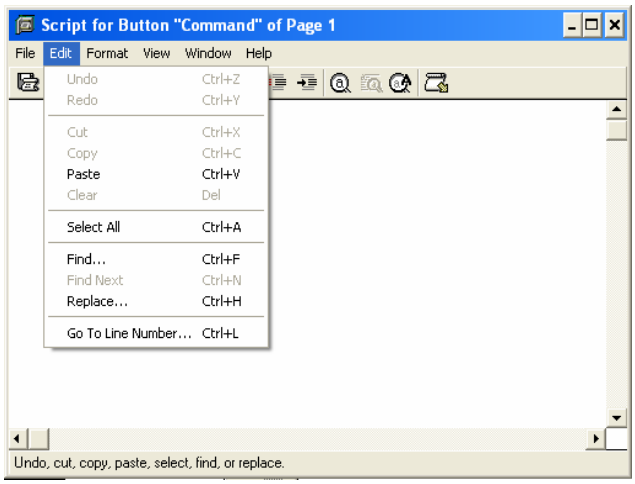


Figura 2.1: *Script editor*

Figura 2.2: Botones de la *toolbar* del *Script editor*

Los *Scripts editors* son no modal, es decir que pueden mantenerse abiertos mientras se realiza otras tareas en la aplicación.

Escribir un *script*

Se utiliza las siguientes normas y comandos clave para editar un *script*:

\ Indica que la instrucción continúa en la siguiente línea.
; Separa instrucciones colocadas en la misma línea.
-- Indica que el texto que sigue es un comentario.
CTRL+D Invoca al *Debugger* (depurador).
CTRL+S Guarda (compila y actualiza) y cierra el *script.*
CTRL+Y Chequea la sintaxis del *script* actual.
CTRL+B Compila, actualiza y guarda el libro.

Command window

Permite ejecutar un segmento de instrucciones del *script* para probar su funcionalidad, obtener información de objetos y variables del sistema. Para acceder al *Command window* escoger *Command* desde el menú *View*, presionar SHIF+F3 o mediante su acceso directo en la barra de herramientas. Para ejecutar un grupo de instrucciones basta con presionar *Enter*.

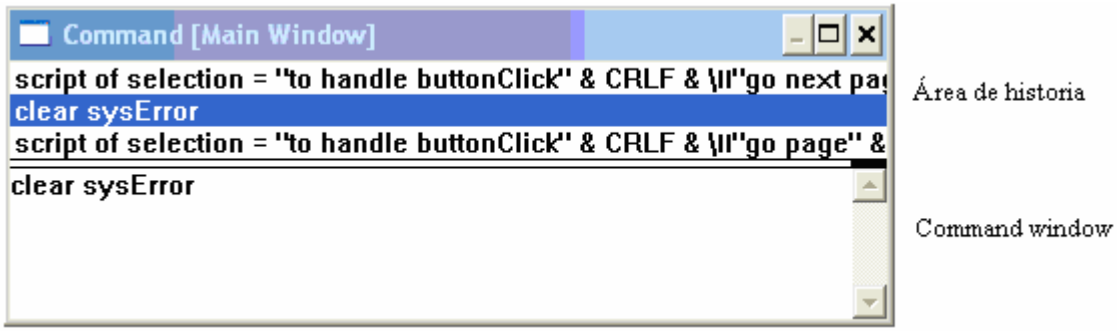


Figura 2.3: *Command window*

La *Command window* tiene un registro de los últimos 20 comandos ingresados en la sesión actual los que se muestran en el área de historia.

Las instrucciones en la *Command window* se ejecutan como si fuesen parte del *script* de la página desplegada en la última *viewer* activa.

Variables en *Command window*

Las variables declaradas en *Command window* son consideradas como locales, sin embargo es posible declararlas como variables del sistema.

Variable	Valor
It	Establecido por el último comando *OpenScript* ejecutado en la *Command window.*
Focus	ID del objeto que tiene el foco al tiempo que la *Command window* fue desplegada.
my, self, target	*uniqueName* de *currentPage* del contexto de ejecución actual.
this page, background, book	*uniqueName* de la página, *background*, o libro desplegado en el contexto de la página actual.
CommandWindow	Texto ingresado en la *Command window.*

Tabla 2.2: Valores de contenedores especiales de *OpenScript* en el *Command window*

ToolBook Debugger Window

Mediante un seguimiento en la ejecución del *script*, identifica errores y permite corregirlos.

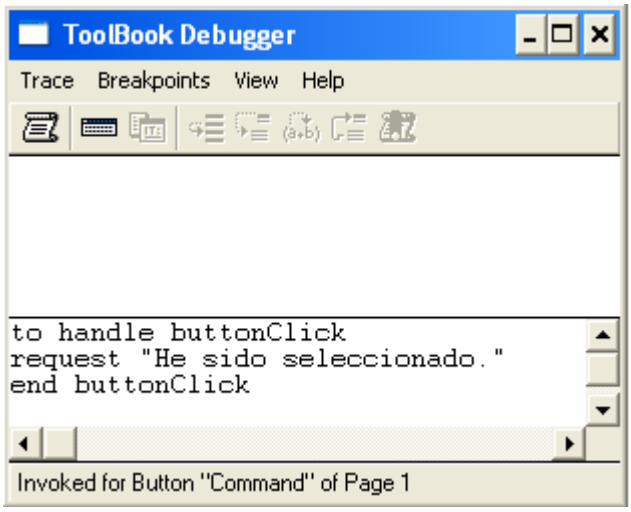


Figura 2.4: ToolBook Debugger windows

SCRIPTS

ToolBook tiene respuestas incorporadas a la mayoría de mensajes generados por eventos. Para controlar la respuesta de un objeto frente a un evento, se puede crear un manipulador (*handler*) para un mensaje. Un manipulador es un conjunto de instrucciones *OpenScript* que definen la respuesta a un mensaje particular.

Un *script* puede tener más de un manipulador cada uno responde a un diferente mensaje.

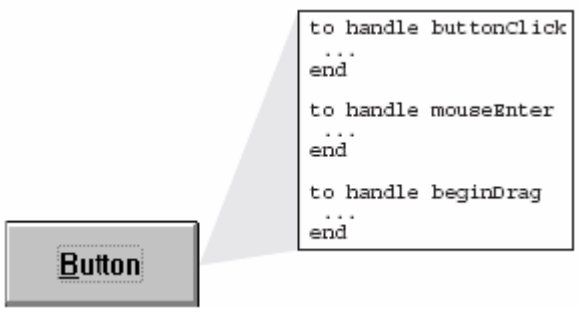


Figura 2.5: Un objeto y su *script*

Al guardar un *script* si no tiene errores automáticamente es compilado, en caso contrario se muestra un mensaje de error. Para ejecutar un *script,* éste debe ser anteriormente compilado.

Cualquier objeto puede tener un *script*, su tamaño máximo es de 64 KB. Un *script* puede contener manipuladores que llaman a otros *scripts* almacenados en otros objetos.

Ejemplo: si se escribe lo siguiente en el *script* de un botón:

```
to handle buttonClick
        request "He sido seleccionado."
end buttonClick
```

Al ejecutar la aplicación si se presiona el botón con el botón izquierdo del *mouse* se presentará el siguiente mensaje:

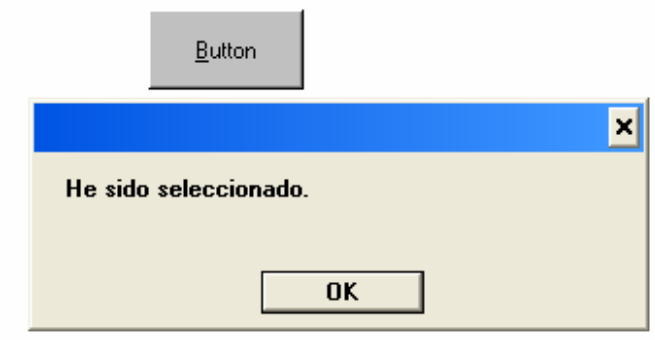


Figura 2.6: Ejemplo de manipuladores

JERARQUÍA DE OBJETOS

El orden en el cual los mensajes pasan de objeto a objeto, es la jerarquía de objetos. Un mensaje pasa desde un objeto a su objeto padre (*parent*), como se muestra en el siguiente gráfico. Dependiendo de los mensajes, éstos ingresan a la jerarquía de objetos a diferentes niveles.

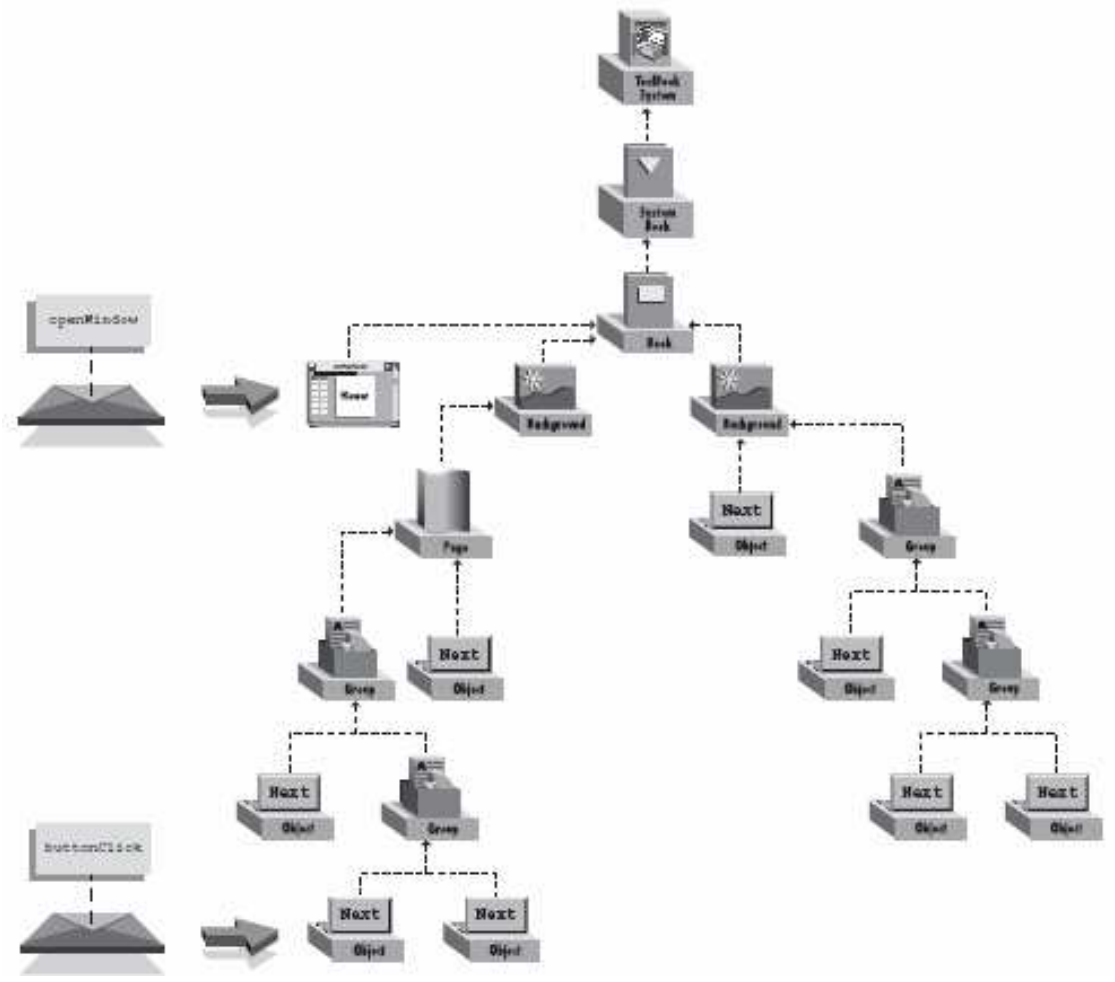


Figura 2.7: Jerarquía de objetos

Cuando un evento ocurre ToolBook envía un mensaje al objeto destino, si el *script* del objeto destino tiene un manipulador para el mensaje, se ejecuta el manipulador y el mensaje no sigue viajando. Si el objeto no tiene un manipulador para el mensaje, sigue pasando basado en la jerarquía de objetos, hasta que encuentra un manipulador o hasta que alcanza el *ToolBook system level*, aquí ToolBook proporciona un comportamiento por defecto.

Para que un mensaje se reenvíe (*forwarding*) al siguiente objeto en la jerarquía, luego de llegar a un objeto que tiene un manipulador para éste, se utiliza el comando *forward* en el manipulador. A continuación se muestra el código:

```
to handle “nombre del evento” [Parámetros]
        <Sentencia de OpenScript>
        forward
end
```

OBJETOS Y PROPIEDADES

Utilizando instrucciones *OpenScript* es posible establecer las propiedades de los objetos de una aplicación.

Referencia a objetos

Se puede referir a un objeto por su tipo y nombre o su número ID:

```
objectCount of page “Principal”[1]
button ID 1 of page ID 0
```

- **Referencia implícita:** Cuando se refiere a un objeto ubicado en la página actual.

[1] Las comillas son indispensables cuando el nombre del objeto incluye espacios o caracteres especiales, pero es recomendable colocarlas para que ToolBook identifique como un nombre de objeto y no trate de determinar si es una variable

- **Referencia explícita:** Si se refiere a un objeto ubicado sobre un *background*, otra página u otro libro. La propiedad *uniqueName* de un objeto contiene ésta referencia.

En el caso de referencia explícita se debe incluir la ubicación del objeto, por ejemplo:

Bounds of field “Nombre” of background ID 1
Text of recordField “Nombre” of page “Inicio”
Script of group “hora” of page 3 of book “aplicación.tbk”

Referencia a la página actual

Se puede utilizar los siguientes términos especiales:

Término	Referencia a	Ejemplo
This	*Background*, página o libro actual	pageCount of this book
my/self	Objeto con el *script* que se ejecuta actualmente	my position = 1500, 3000 move self by 1000, 1000

Target	Objeto que recibió primero el actual mensaje	to handle buttonClick
		  if object of target is "button" then
		  request "You clicked a button."
		  end if
		  if object of target is "field" then
		  request "You clicked a field."
		  end if
		end buttonClick

selection	Objeto u objetos seleccionados actualmente	borderStyle of the selection = shadowed

Tabla 2. 3: Referencia a la página actual

Establecer y obtener valores de propiedades

Existen tres maneras para asignar un valor a una propiedad:

```
fontFace of field "One" = "Times New Roman"
set fontFace of field "One" to "Times New Roman"
put "Times New Roman" into fontFace of field "One"
```

Para obtener el valor de una propiedad se puede utilizar el comando *get* y almacenar el valor en la variable *It*:

```
get uniqueName of field "Empresa"
```

La variable *it* almacenará la referencia explícita del campo.

TIPOS DE MANIPULADORES

- ***to andel:*** Define la respuesta de un objeto a un mensaje particular.
- ***to get:*** Retorna un valor a ser usado en otro manipulador, se usa para crear una función definida por el usuario u obtener el valor de una propiedad definida usando instrucciones *OpenScript*.
- ***to set:*** Define propiedades creadas por el usuario con *OpenScript*.
- ***notifyBefore* or *notifyAfter:*** Define un manipulador que es notificado cuando una página recibe un mensaje particular.

Conjuntamente con los manipuladores es posible utilizar parámetros, los que acompañan a un mensaje para proporcionar información más detallada. Ejemplo:

```
to handle buttonClick location, isShift, isCtrl
        if isShift is true then
                request "Se hizo click mientras la tecla Shift fue presionada"
        end if
end buttonClick
```

En el ejemplo anterior para el mensaje *buttonClick* se tiene dos parámetros *isShift* y *isCtrl,* los cuales indican cuando el usuario mantiene presionada la tecla SHIFT o CTRL cuando se hace *click*.

INGRESAR Y DESPLEGAR DATOS

Existen varias maneras para ingresar y mostrar datos, por ejemplo:

Comando	Función	Ejemplo
Put	Despliega datos en la *Command window.*	put pageCount of this book
Ask	Permite que el usuario ingrese datos en un cuadro de diálogo con botones *OK* y *Cancel*. La respuesta es retornada en la variable *It.*	ask "Enter your name:" request "Welcome to the Tutorial," \ && It
Request	Hacer una pregunta, en una ventana con botones de *Yes* o No. La selección es retornada en la variable *It*.	request "Do you wish to move to the \ next page?" with "Yes" or "No" if It is "Yes" then send next End

Tabla 2. 4: Comandos para ingresar y desplegar datos

Una manera común y fácil de recolectar información desde el usuario es utilizar un cuadro de diálogo (puede contener campos, botones, combos, etc.) o usar *viewers* que actúen como estos.

MANEJO DE MENSAJES, EVENTOS, OBJETOS Y PROPIEDADES

MENSAJES

Toolbook permite manejar dos tipos de mensajes: *built-in* (propios del sistema) y *user-defined* (definidos por el usuario).

Mensajes built-in

ToolBook envía mensajes *built-in* de forma automática como respuesta a un evento. Los manipuladores para un mensaje *built-in* pueden ubicarse en el *script* del objeto destino o en un lugar elevado en la jerarquía de objetos. En el caso de manipuladores que definan comportamiento para un grupo de objetos es recomendable ubicarlo en un lugar superior en la jerarquía de objetos, como en el *script* de la página o del grupo de objetos, esto permite su manipulación sin tener que modificar el *script* de cada objeto.

Si se desea que la respuesta al mismo mensaje sea diferente con cada objeto, se debe colocar el manipulador en el *script* del objeto destino, o en el *script* de la página o el libro y utilizar la propiedad *target* para distinguir que objeto recibe el mensaje utilizando una estructura de control.

```
to handle buttonClick
        conditions
                when object of target is "button"
                        send (name of target)
                when object of target is "hotword"
                        go to page (name of target)
                else
                        forward – En el caso de que el usuario escoja otro objeto
        end conditions
end buttonClick
```

Los mensajes *built-in,* pueden ser cambiados o deshabilitados totalmente, debido a que la mayoría de mensajes se mueven a través de la jerarquía de objetos antes de alcanzar el *ToolBook system* [1].

Tipos de mensajes built-in

[1] Usar *forward to system* para enviar un mensaje directamente a *ToolBook system*, sin seguir la jerarquía de objetos

Mensaje	Descripción	Nivel
Menu-event	Se envía a la página actual, cuando un ítem del menú es seleccionado.	Lector y Autor
Mouse-event	Enviado cuando el botón del *mouse* es presionado o el cursor se mueve sobre un objeto.	Lector
Keyboard-event	Cuando se presiona una tecla.	Lector
Enter/leave-Event	Enviado a la *viewer*, página, *background*, libro o sistema, cuando el usuario abre o cierra una aplicación, libro o *viewer*; navega a otra página; o *click* un menú.	Lector y Autor
	A un botón, *combo box* o campo cuando el objeto recibe o pierde el foco.	Lector
Notification	Notificar al sistema de varios eventos, que afectan el estado de un objeto, como cuando un objeto es: creado, destruido o movido.	Lector y Autor
Drag-and-drop	Ocurre durante la operación *drag-and-drop.*	Lector
Quero	Se envía a un objeto para saber su estado o el valor de una propiedad particular.	Lector
Standard	Únicamente es enviado a *ToolBook system* (no sobre la jerarquía de objetos).	Lector

Tabla 2. 5: Tipos de mensajes *built-in*

Mensajes user-defined

Permite manejar tareas que no son parte de ToolBook, su funcionalidad es similar a llamar un procedimiento o funcionalidad propia en otro lenguaje de programación. Para enviar un mensaje *user-defined* se utiliza el comando *send*. Es necesario proveer un manipulador correspondiente a cada mensaje *user-defined*, en caso contrario ToolBook despliega un mensaje de error.

Los mensajes *user-defined* facilitan la creación de aplicaciones modulares, ya que permiten la creación de manipuladores separados para diferentes tareas, como se observa en el siguiente ejemplo:

```
to handle enterBook
    forward
```

```
-- Los siguientes son todos los mensajes user-defined
        send initMenus
        send sizeAppWindow
        send setSystemDefaults
end
```

En el ejemplo se crea manipuladores separados para inicialización de menús *initMenus*, dimensionar ventanas *sizeAppWindow*, y establecimiento de valores por defecto *setSystemDefaults*, los manipuladores individuales puede ejecutarse automáticamente cuando el libro es abierto, o ser llamados cuando sea necesario con el comando *send*.

Envío de mensajes

Los comandos *send* y *forward* permiten enviar mensajes independientemente de si el evento que usualmente desencadena el mensaje ocurrió.

- ***Send:*** Usado para ejecutar un manipulador en el actual *script* del objeto o enviar un mensaje a cualquier otro objeto, incluyendo objetos inferiores en la jerarquía o en otro libro.
- ***Forward:*** Para el mensaje actual al inmediato objeto superior en la jerarquía.

Un manipulador con una instrucción *send* o *forward* que ejecuta otro manipulador es referido como un manipulador *calling*. El manipulador que es invocado es el manipulador *called*. Cuando ToolBook encuentre una instrucción *send* o *forward* en un *script*, éste suspende el manipulador *calling* hasta que el manipulador *called* se termina.

Es posible enviar el nombre de un mensaje como variable; para forzar a ToolBook a reconocer que se quiere el contenido de la variable enviada y no el nombre de la variable, poner paréntesis alrededor del nombre de la variable que contiene el mensaje a ser enviado. Por ejemplo:

```
Ask “Qué mensaje quiere enviar”
Send (It)
```

Parámetros con mensajes

Los parámetros son utilizados por ToolBook para enviar información adicional sobre el mensaje, es posible conocer su valor añadiendo variables de parámetros a los manipuladores. En el siguiente ejemplo se utiliza una variable de parámetro “*pos*” para indicar la posición donde se hace *click*:

```
to handle buttonClick pos
        request "Usted ha seleccionado" && pos
end
```

Cuando se envía un mensaje con el comando *send*, se puede enviar valores y variables como parámetros para el mensaje; en el manipulador para el mensaje, se debe poner variables de parámetros en el mismo orden, sin importar su nombre ya que éste puede variar:

```
send specialEffects location, color, grayTone
to handle specialEffects place, colorCode, gradation
```

Un parámetro es pasado por valor, es decir una copia del valor en el manipulador *calling*. Los parámetros son variables locales en el manipulador al cual ellos son enviados, si se cambia el valor de un parámetro en el manipulador *called*, este no afecta el valor del correspondiente parámetro en el manipulador *calling*.

Es posible añadir declaraciones de tipos de datos a parámetros en la instrucción *to handle*. Al declarar una variable con un tipo de dato, se requiere menos memoria y ToolBook puede acceder a los datos más rápidamente, esto hace al parámetro más eficiente.

Para referirse a un parámetro se lo hace por su nombre, o utilizando variables especiales:

- *argList* para obtener una lista de los parámetros pasados, y utilizar *item* para referirse a un parámetro:
  put item 3 of argList into field "MyList" of this page
- *argCount* para obtener el número de parámetros pasados.
- *itemCount(argList)* para obtener el número de ítems pasados con un mensaje.

Si un mensaje tiene un parámetro ubicación, el parámetro contendrá una lista de dos números. En este caso, *itemCount(argList)* retornará 2 y el valor de *argCount* es 1.

OBJETOS Y PROPIEDADES DEL SISTEMA

Existen objetos propios del sistema ToolBook como: paletas, barra de estado, barra de herramientas, etc., estos objetos no pueden ser creados o destruidos, sin embargo pueden ser manipulados para cambiar la apariencia y comportamiento de ToolBook.

Los nombres de los tipos de objetos del sistema ToolBook son:

colorTray	*patternPalette*	*statusControls*
commandWindow	*polygonPalette*	*statusIndicators*
lineEndsPalette	*statusBar*	*toolBar*
linePalette	*statusBox*	*toolPalette*

Las propiedades del sistema definen un comportamiento por defecto para el sistema ToolBook completo o identifican los objetos que están actualmente recibiendo mensajes.

Tipos de propiedades del sistema:

Propiedad del sistema	Función
General system properties	Afectan a todo el libro abierto durante la actual instancia de ToolBook.
Event-focus propierties	Propiedades: *target* indica el objeto que recibió el mensaje. *targetWindow* indica la *viewer* que desplegó el objeto. focusWindow indica la *viewer* que es actualmente activa (puede o no ser la misma que *targetWindow*).
Printer system properties	Definen aspectos de impresión del libro.
Internacional system properties	Cambiar automáticamente un libro a convencionalismos locales un país específico.
Startup system properties	Afectan todos los libros en todas las instancias.

Tabla 2.6: Tipos de propiedades del sistema

USER PROPERTIES

El desarrollador de aplicaciones puede establecer una nueva propiedad a un objeto, diferente de las que vienen preconstruidas con cada objeto, estas pueden permitir obtener o definir un valor. Una *user property* puede ser usada como un contenedor global, al igual que una variable del sistema, sin embargo, las *user properties* son almacenadas al cerrar el libro y son asociadas solo con un objeto.

Una *user property* puede crearse con una instrucción de *OpenScript* como la siguiente:

```
userScore of button “Play” = 0
```

Si el nombre de la propiedad no existe, ToolBook crea la propiedad y la asigna el valor establecido. Es importante siempre especificar el objeto al que pertenece la propiedad, de otra forma ToolBook creará una variable local en lugar de una propiedad.

```
get userScore of button "Play"          -- Obtener el valor de la propiedad, pone el
                                           valor en It
```

put userProperties of button "Play"	-- Lista las user properties en la Command window
userScore of button “Play” = null	-- Establece una user property a nulo
clear userScore of button “Play”	-- Borra una user propertiy[1]

Creación de propiedades

Una propiedad puede ser creada con valores determinados por los manipuladores escritos. Los manipuladores son llamados por los comandos *set* o *get* al igual que cuando se establece u obtiene una propiedad *built-in* o *user*. Sin embargo en lugar de establecer u obtener el valor de una propiedad, ToolBook llama a su manipulador, el cual puede fijar valores, modificar objetos o ejecutar cualquier otro proceso que la aplicación requiera.

Para crear una propiedad usando un *script*, escribir un manipulador *to set*. La sintaxis para un manipulador *to set* es:

to set <propertyName> [parameters] to <value>
… -- Instrucciones
end

Cuando se llama el manipulador con un comando *set*, el valor que se asigna a la propiedad es pasado al manipulador *to set* como el parámetro <*value*>. El manipulador *to set* puede ser ubicado en el *script* del objeto o en un lugar superior en la jerarquía de objetos, al igual que cualquier otro manipulador.

Un manipulador *to get* se crea para definir el valor de una propiedad creada con un manipulador *to set*. La sintaxis para un manipulador *to get* es:

to get <Nombre de la propiedad> [parámetros]
--(aquí se calcula el valor)

[1] Guardar el libro con otro nombre para que se reorganice el libro a su mejor tamaño y se recobre espacio utilizado por user properties que han sido borradas

return <value>
end

Como con una propiedad *built-in*, ToolBook invocará el manipulador *to get* para obtener el valor de la propiedad. Si el manipulador *to get* no es creado con un manipulador *to set*, ToolBook retornará *null*.

Manipuladores de notificación

Un manipulador de notificación es uno que se notifica automáticamente cuando el mensaje de este manipulador alcanza la página. Si un objeto contiene un manipulador de notificación, la página envía una copia del mensaje al objeto, sin la necesidad de añadir un manipulador en otro sitio para enviar el mensaje al objeto.

Existen dos tipos de manipuladores de notificación: *notifyBefore* y *notifyAfter*. *NotifyBefore* se ejecuta momentos antes que se envía el mensaje y *notifyAfter* se ejecuta momentos después de que el mensaje es enviado. Sintaxis de un manipulador de notificación:

notifyBefore enterPage
… -- Instrucciones
end

Un manipulador de notificación no puede estar en el *script* de la página, *background*, visores o libros.

INSTRUCCIONES OPENSCRIPT

Una misma tarea puede ser realizada usando un comando o enviando un mensaje:

Usar comandos como *export* o *search* para ejecutar la acción sin desplegar el cuadro de diálogo relacionado, o usar comandos con palabras como *autor* o *reader* si quiere evitar ejecutar manipuladores de mensajes relacionados.

Enviar mensajes como *import* o *saveAs* si se quiere que el usuario realice una acción por ingreso de datos en un cuadro de diálogo. Enviar un mensaje como *reader* o *clear* cuando se quiere que cualquier manipulador para el mensaje se ejecute en adición a la realización de la primera acción. Enviar un mensaje resulta igual que escoger un ítem del menú relacionado.

Estructuras de control

Tipo	Comando	Descripción
Lazos	*Step*	Se repite un número de veces determinado.
	do/until	Se repite hasta que una condición es cumplida; siempre se ejecuta al menos una vez.
	While	Se repite mientras una condición es verdadera.
Establecer Condiciones	*If/then/else*	Ejecuta instrucciones si una condición es verdadera, *else* separación opcional.
	Conditions	Ejecuta una opción desde un conjunto que se excluye mutuamente.
Mensajes de manipuladores	*to andel*	Reacciona a un mensaje del sistema o usuario.
Creación de propiedades u funciones *user-defined*	*to set*	Define un *script* para crear o calcular un valor para una propiedad o función *user-defined.*
	to get	Define un *script* para retornar el valor de una función *user-defined* o calcular propiedades del usuario.

Establecer ventanas Destino	*in <viewer>*	Temporalmente hace una visualización de la ventana destino para la ejecución de *scripts*.
Impresión (*spooling*)	*star spooler* [1]	Enruta al subsecuente comando *print* al dispositivo de impresión actual.
Enlazar DLLs	*linkDLL*	Declara funciones desde una DLL a ser llamada.
Mensajes Windows	*translateWindowwMessage*	Establece manipuladores de la translación de mensajes *Windows-to-OpenScript.*

Tabla 2.7: Estructuras de control con OpenScript [2]

Las estructuras de control terminan con la palabra *end*, con excepción de *do* el cual termina con *until* <*expression*>. Ejemplo:

```
step ctr from 1 to pageCount of this book
        flip
end step
```

Interrupción de estructuras de control

Break permiten interrumpir el flujo de una estructura de control, pueden usarse para abandonar un manipulador sin terminar todas sus instrucciones, o en un lazo para salir tempranamente como cuando un error es detectado.

La instrucción *break to system* detiene la ejecución de todos los manipuladores corriendo actualmente.

```
step i from 1 to 1000
        -- Chequea el espacio disponible en la página
        if percentFreeSpace of this page < 10 then
                break step
```

[1] Programa que controla los trabajos puestos en cola y ejecuta uno a la vez (impresoras)

```
    else
    …       -- Otras Instrucciones
    end if
end step
```

continue permite saltar instrucciones en una estructura de control y continuar directamente a la siguiente iteración.

Funciones

Son procedimientos usados para calcular un valor o desempeñar una operación. ToolBook tiene más de 200 funciones incluidas para varios propósitos. Para calcular el valor a retornar en la función se utilizan parámetros.

OpenScript pasa los parámetros por valor, excepto en arreglos donde se puede usar el término especial *by reference* para pasar los parámetros por referencia. Pasar parámetros por valor aumenta el uso de la memoria ya que se realiza una copia de la variable. Es posible pasar parámetros (que no son arreglos) por referencia, para esto se debe asignar un espacio de memoria y pasar un puntero a la memoria como un parámetro a su función.

Funciones definidas por el usuario

Es una función definida por un manipulador, escrita para calcular un valor o ejecutar un procedimiento y retornar un valor. Para crear una función definida por el usuario, se escribe un manipulador *to get*:

```
to get grossMargin Sales, COGS
    return Sales – COGS – Retorna el valor calculado
end
```

El nombre de una función definida por el usuario no puede contener espacios o un carácter diferente a letras, números, *underscore* (_), o símbolos de prefijo como (@).

El manipulador *to get* puede ubicarse en cualquier lugar de la jerarquía de objetos, sin embargo es recomendable definirlo en el *script* del libro para facilitar su mantenimiento.

Una función definida por el usuario puede ser llamada desde cualquier manipulador al igual que las funciones incluidas. Ejemplo:

```
to handle buttonClick
    local vSales, vCOGS
    vSales = text of field "Total Sales"
    vCOGS = text of field "Unit Cost" + text of field "Overhead"
text of field "Gross Margin" of page 3 = grossMargin(vSales,vCOGS)
end
```

Expresiones

Es una combinación de valores que producen un resultado, los valores son enlazados con operadores que pueden ser palabras o símbolos que indican la operación a realizarse. Ejemplos:

```
--Multiplica dos valores
subTotal = vItems * vQty

--Junta tres grupos de caracteres en uno
greeting = "Dear" && title && lastName

--Divide el valor de dos funciones
average = sum(allItems) / itemCount(allItems)
```

--Compara dos valores
if text of field "Amount" > vLimit then …

Tipos de operadores de OpenScript[1]

- **Aritméticos**: Usados para crear expresiones matemáticas.
- **Lógicos**: Evalúan dos expresiones y retornan *true* o *false*.
- **Caracteres**: Concatena dos *strings* en uno más largo, o extrae un *substring* de uno más largo.
- **Bitwise**: Permite desempeñar operaciones binarias de bits individuales.

DEFINICIÓN DE VARIABLES, VALORES Y TEXTO

VARIABLES

Una variable es un contenedor que almacena datos, se define en un *script*. Las variables definidas en *OpenScript* tienen las siguientes características especiales:

- Pueden ser declaradas con o sin un tipo de dato.
- Si una variable se declara sin un tipo de dato puede contener valores numéricos, lógicos, caracteres y otros.
- No es necesario declarar una variable local antes de asignar un valor a esta, con excepción de arreglos.
- Se definen variables globales o locales usando las palabras claves local o *system*.

El nombre de una variable puede contener únicamente caracteres alfanuméricos, *underscore* y el símbolo (@) como un prefijo; su nombre no puede ser una palabra clave de *OpenScript*.[2]

[1] Anexo F: Tipos de operadores *OpenScript*
[2] *OpenScript* no es sensitivo a la diferencia entre mayúsculas y minúsculas

Para asignar un valor a una variable se puede usar *set*, *put*, o el signo de igual (=); estos comandos tienen el mismo efecto.

Variables locales y del sistema

Una variable local mantiene su valor solo mientras el manipulador en el cual fue declarada está ejecutándose. Una variable del sistema mantiene su valor durante la sesión actual de ToolBook, se usan cuando es necesario que su información esté disponible para más de un manipulador.

Una instrucción local permite declarar variables locales, sin embargo no es necesario su uso, ya que Toolbook declara e inicializa la variable cuando a esta se asigna su primer valor.

Para crear una variable del sistema, declararla con la instrucción *system*.

```
local vTotal            --Use "v" para etiquetar una variable local
system svVarName        --Use "sv" para etiquetar una variable del sistema
```

Las variables *It*, *argList* y *argCount* se reservan para contener resultados de una instrucción o información sobre el manipulador actual.

Tipos de datos para variables

Asignar un tipo de dato al declarar la variable es recomendable para una rápida ejecución, menor uso de memoria y evitar errores de asignación de inapropiados.

```
local LONG I                            --LONG Entero de 32 bits
system INT pagesDisplayed, pagesLeft    --INT Entero de 16 bits
local COLOR oldFillColor                --COLOR Lista de 3 números
                                          positivos en unidades HLS o RGB[1]
```

[1] HLS (*hue*, *lightness*, y *saturation*); RGB (*red*, *green*, y *blue*)

Tipo de dato	Descripción
Int	Entero con signo de 16 bits. Su rango de valores va desde el -32768 hasta el 32767.
Long	Entero con signo de 32 bits. Su rango de valores va desde el -2147483648 hasta el 2147283647.
Real	Números reales de 64 bits.
Word	Entero sin signo de 16 bits. Su rango de valores va desde el 0 hasta 65535.
Dword	Entero sin signo de 32 bits. Su rango de valores va desde el 0 hasta el 4294967295.
String	Cadena de caracteres.
Logical	Contiene los valores lógicos *true* o *false.*
Point	Define una coordenada x,y; donde x e y son enteros.
Color	Define una lista de 3 números no negativos en unidades HLS.
Stack	Define una lista de elementos separados por comas.
Date	Fecha, utiliza el formato definido en el sistema.
Time	Hora, utiliza el formato definido en el sistema.
Page	Referencia explícita a una página.
Background	Referencia explícita a un *background.*
Layer	Referencia explícita a una página o *background.*
Graphic	Referencia explícita a un objeto de una página o *background.*
Field	Referencia explícita a un campo, campo de registro o a un cuadro combinado.
Object	Referencia explícita a un libro, página, *background* o gráfico.
Book	Referencia explícita al camino o *path* de un libro.

Tabla 2. 8: Tipos de datos de OpenScript [2]

Las variables declaradas con un tipo de dato tienen un valor por defecto que en su mayoría es *0*, *false* o *null*, dependiendo del tipo de dato.

ARRAYS

Un *array* (arreglo) es una variable configurada para trabajar al igual que una lista (*array* de una dimensión) o tabla (*array* de dos dimensiones) de elementos individuales. Los *arrays* organizan la información de forma exacta y sistemática. El tamaño de un *array* puede ser fijo o dinámico, pero en un mismo *array* no se puede mezclar los dos tipos de tamaños.

Para fijar el tamaño de un *array* indicar su dimensión en la declaración. Un *array* de tamaño dinámico tiene vacío el campo de números de elementos.

```
local hiScorer[10]                  --Declaración de arrays de tamaño fijo
local logical seats[32][80]
system users[]                      --Declaración de arrays de tamaño dinámico
local playerScores[][]
```

El número de índice de un *array* empieza en uno (1).

El comando *fill* permite asignar un mismo valor a todos los elementos del *array*:

```
local vHiScorers[]
fill vHiScorers with text of field "hiScore" in [textline] order
```

fill con ayuda de *in… order* asigna unidades individuales de texto, tal como líneas (*textline*), ítems (*item*), palabras (*word*) o caracteres (*char*) en el array.

```
sampleText = "Ser o no ser," & CRLF & "es la pregunta"
local myArray[2][4]
fill myArray with sampleText in [textline][word] order
```

La función *dimensions* permite determinar las dimensiones del *array*.

Los *arrays* por defecto son pasados por valor, sin embargo es posible pasarlos por referencia utilizando el termino especial *by reference*, de esta forma el proceso es más rápido, con menos cantidad de memoria y los cambios realizados en el *array* en el manipulador *called* son reflejados al manipulador *calling*.

VALORES LITERALES

Es un valor asignado a una variable o usado en una expresión. Si la variable ha sido declarada con un tipo de dato el valor debe ser correspondiente. Si se asigna un valor a una variable que no tiene definido un tipo de dato, ToolBook determinará el tipo de dato dependiendo de cómo se está usando el valor.

```
vStartingValue = 0
vGreeting = "Dear" && vName
show group "palmTree" at "100,100" --100,100 es un punto en unidades de página
text of field "Total" = sum(100,100) --100,100 es una lista
```

Unidades de página y pixels

Las unidades de página se usan con objetos gráficos para especificar los bordes y vértices, la ubicación de un objeto relativo a la esquina superior izquierda, o el destino y distancia para mover un objeto. Una pulgada contiene 1440 unidades de página. El número de *pixels* por unidades de página depende del dispositivo de visualización. *Pixels* se usan para especificar la ubicación o desplazamiento de visores o paletas.

CONSTANTES

Son palabras que representan un valor que no se puede cambiar, puede ser número o *strings*.

Tipos de constantes

Tipo de constantes	Ejemplos
Color (valores HLS o RGB)	*red, green, blue*
Caracteres de control y puntuación	*quote*, *tab*, *space*, *CRLF*, *EOF*
Keystrokes	*keyA, keyUpArrow, keyLeftButton, keySlash*
Expresiones matemáticas	*Pi*

Tabla 2.9: Tipo de constantes OpenScript [2]

STRINGS

Un *string* es una combinación de caracteres seguidos (alfabéticos, numéricos, puntuación, y control), éstos pueden ser asignados a una variable, a una propiedad de un objeto, o usarse en una expresión.

Para hacer referencia a una parte de un *string* se usa los operadores *char*, *item*, *textline*, o *word*. También se utilizan números o palabras clave como *first*, *last*, *second*,… para indicar la posición de un *string*.

```
char 2 of word 3 of textline 4 of text of field "Note" of page 27 of book "home.tbk"
third character of vAlphabet        --Obtiene "C" desde una variable local
last item of vNames                 --Obtiene el último ítem de una variable local
```

En ToolBook cualquier valor que no tiene un tipo de dato es convertido a *string* cuando se usan operadores.

Funciones usadas con strings

ansiToChar( )	*uppercase( )*	*textlineCount( )*
charCount( )	*lowercase( )*	*wordCount( )*
charToAnsi( )	*offset( )*	
itemCount( )	*textFromPoint( )*	

LISTAS

Las listas son un grupo de ítems separados por comas, son utilizadas para datos estructurados. Una ventaja del uso de listas es almacenar una serie de ítems temporalmente sin crear variables separas para cada ítem.

Para referirse a un elemento de la lista use la palabra clave *item*. El comando *push* añade un ítem a un contenedor y *pop* remueve este.

```
vNames = "Amelia,Molly,Isabel,Roxanne"
get item 2 of vNames                    –Pone el nombre "Molly" en It

local vGrades
push "B" onto vGrades                   --vGrades ahora contiene "B"
push "dog" onto item 5 of vAnimals
```

MANEJO DE DATOS

FORMATO DE DATOS

Dentro de una aplicación la presentación de los datos es importante para darle una apariencia consistente, éstos deben tener un formato similar de acuerdo al tipo de información que representen. Más allá de la presentación, el formato de los datos es de gran importancia al momento de trabajar con éstos, por ejemplo al realizar cálculos matemáticos con ellos o hacer comparaciones entre varios valores.

Formato de datos para la salida

Para desplegar fechas, horas o números con un formato específico primero se debe convertir los datos con el comando *format:*

```
totalValue = "2000000"
format number totalValue as "$0,0.00"
request totalValue – Despliega "$2,000,000.00"
```

En caso de fechas y horas es necesario indicar el formato de origen de los datos, para su correcta interpretación, ésto se realiza con la cláusula *from:*

```
today = "31 December 97"
format date today as "mm/dd/yy" from "d M yy"
request today --Displays "12/31/97"
```

Utilizando los comandos *sysDateFormat* y *sysTimeFormat,* se especifica un formato que va a ser adoptado por los valores de *sysDate* y *sysTime* y las variables declaradas con tipos de datos *date* o *time.*

```
sysTimeFormat = "h24:min"
request sysTime         --Despliega "18:00"
local date today
today = "9/30/97"
sysDateFormat = "M d, y"
request today  --Despliega "September 30, 1997"
```

Formato de datos para cálculos

Antes de realizar cálculos con números, fechas y horas es necesario convertir estos valores a un formato adecuado. En el caso de los números es recomendable formatearlos como *null* para quitar signos o espacios que estos pueden contener.

Para cálculos con fechas y horas se puede usar el formato de *string* de una fecha, o usar el formato *seconds* de esta forma todos los valores ingresados van a tener la misma referencia (1ro de enero de 1970 para fechas y media noche para horas) y realizar cálculos aritméticos va a ser más sencillo.

Formato de datos para comparación

Es importante que números, fechas y horas tengan el mismo formato antes de realizar comparaciones con estos valores, para ésto se puede utilizar la cláusula *as* en una instrucción *if*.

VALIDACIÓN DE DATOS

Los datos ingresados por el usuario no siempre son correctos, para validar sus ingresos utilizamos manipuladores que prueben si existen errores y ayuden al usuario a corregirlos.

Existen pasos generales para la validación de datos: [2]

1. Obtener el valor que el usuario ingresó.
2. Probar que el valor es válido.
3. Responder al usuario. Ejemplo: Existen errores
4. Si el dato es inválido, ubico el foco al campo donde el usuario ingreso el dato, borro el valor inválido, o de otra manera me aseguro que el ingreso inválido se trate con cuidado.

Es mejor chequear los datos inmediatamente luego de que se ingresan. Sin embargo usualmente se valida los datos en los manipuladores *leaveField* o *leaveRecordField*.

Comprobación de datos ingresados por el usuario

En ToolBook se tienen dos funciones generales de validación, *isType* y *isObject*. Ejemplo:

```
ask "Ingrese un número:"
if isType(int,It) is false
        request "Por favor ingrese un valor entero."
end
```

Usar *sysError* para validación

Un error puede ser detectado si se predice el valor que el usuario debe ingresar, si el usuario ingresa un valor no esperado se generará un error. Para que el usuario no observe los errores, *sysSuspend* debe establecerse como *false* antes de chequear los datos y luego volver a su estado *true*.

```
If sysError is not null then
        Request “”      -- mensaje de error
End if
```

Responder al usuario

Para enviar un mensaje de error al usuario, se usa comandos como *request* o *ask*, ejemplo:

```
to handle leaveField
        get text of target
        if It is not null and isType(int,it) is false
                beep 1
                request "Por favor ingrese un valor entero." with OK
        end if
        forward
end leaveField
```

Si se quiere que el usuario continúe sin arreglar los errores, establecer *sysSuspendMessages* a *true* para no desencadenar nuevamente los manipuladores *leaveField* o *leaveRecordField*, luego establecer el foco a otro objeto o a *null* y regresar *sysSuspendMessages* a *false*.

También se puede retornar a un valor definido en caso que el usuario ingrese un dato incorrecto y no escoja corregirlo.

LEER Y ESCRIBIR ARCHIVOS

OpenScript permite manejar archivos de forma directa, de forma simultánea se puede abrir hasta 10 archivos, entre los comandos que presenta para leer o escribir a un archivo se tiene:

createFile	*seekFile*
openFile	*writeFile*
readFile	*closeFile*

Para leer o escribir un archivo primero se debe abrir éste, y al final cerrarlo para guardar cambios en el disco, liberar memoria y otros recursos del sistema.

- *createFile* este comando crea y abre un archivo.
  createFile "myfile.txt"
  Si un archivo con este nombre ya existe ToolBook lo sobreescribe.

- *readFile* permite leer un archivo.
  readFile <file name> for <count> --count: número de caracteres a leer
  readFile <file name> to <delimiter> --delimiter: un caracter hasta el cual quiero leer

- *writeFile* permite actualizar un archivo abierto, por defecto añade los datos al final de el archivo.
  logEntry = name of this book & "," & sysDate & "," & sysTime & CRLF
  writeFile logEntry to fileName

- *seekFile* este comando especifica una posición en el archivo para leer o escribir datos. La variable *It* retorna la posición a la que se ha movido.

El comando *search* busca texto en un libro, empezando con el campo o campo de registro que actualmente tiene el foco y sigue buscando en el orden de capas. Para que siga buscando la siguiente ocurrencia usar *search again*. Por defecto al encontrar el texto deseado ToolBook navega a la página que contiene el texto, si

no se desea ir a la página que contiene el texto se usa *locateOnly* con el comando *search*.

ORDENAR PÁGINAS

Existen dos maneras de ordenar páginas en ToolBook en el comando *Sort Pages* el menú *Tools* o con el comando *sort* de *OpenScript*, con el primero únicamente se ordena las páginas por campos de registro, mientras que con la segunda opción se puede ordenar por otro tipo de expresiones como nombre de página o ID de página. Ejemplo:

```
to handle buttonClick
    sort pages 1 to pageCount of this book by text name of this page
end
```

Se puede especificar un ordenamiento por varios criterios, separados por comas.

EFECTOS ESPECIALES

OpenScript permite incluir efectos especiales a la aplicación. A continuación se verá algunos tipos de efectos.

ANIMACIÓN DE OBJETOS

Existen varias técnicas para animar objetos, como son: cambiar la posición, tamaño o forma de la imagen para aparentar movimiento.

Movimiento de objetos

Con el comando *move* se cambia la ubicación del objeto, a una nueva posición específica. Ejemplo:

```
to handle buttonClick
        send sizeToPage                         –Establece el tamaño de la ventana
        bottomEdge = (item 2 of size of this book) -700
        originalPos = item 2 of my bounds   –Donde el objeto está ahora
        changeInPos = 100                       –Cuanto se mueve cada vez
        do
                move self by 0, changeInPos
                newBounds = my bounds
                f item 2 of newBounds > bottomEdge then
                        changeInPos = -100     --Cambia la direcciónde l objeto
                end if
        until item 2 of newBounds <= originalPos
end buttonClick
```

Cambiar la forma de un objeto

Con la propiedad *bounds* se cambia la forma o el tamaño de un objeto por medio de su recuadro de selección.

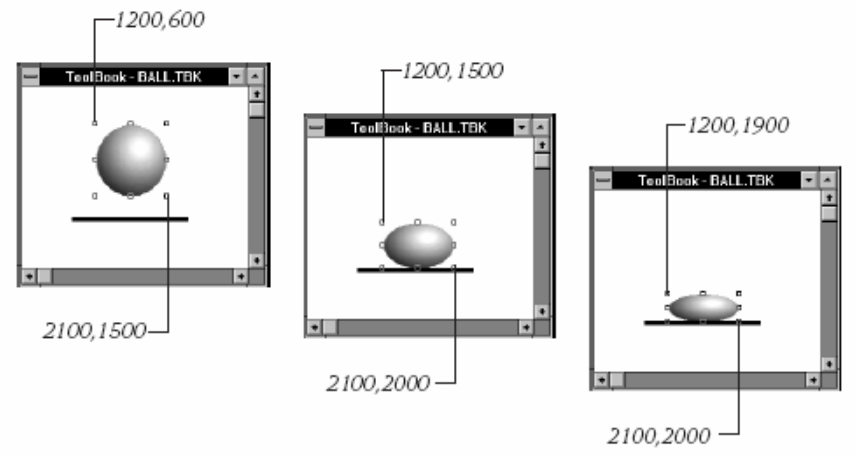


Figura 2.8: Cambiar la forma de un objeto estableciendo su propiedad *bounds*

Cambiar vértices de líneas y curvas

Cambiando los vértices de una línea se puede crear animaciones, como se muestra en el siguiente ejemplo:

```
to handle buttonClick
        zigZag = "400,3000,1200,2000,2000,3000,3300,1800"
        straight = "500,2500,1200,2500,2000,2500,2800,2500"
        move angledLine "ZigZag" to 500,2500
        vertices of angledLine "ZigZag" = zigZag
        pause 50
        vertices of angledLine "ZigZag" = straight
        pause 50
        vertices of angledLine "ZigZag" = zigZag
end
```

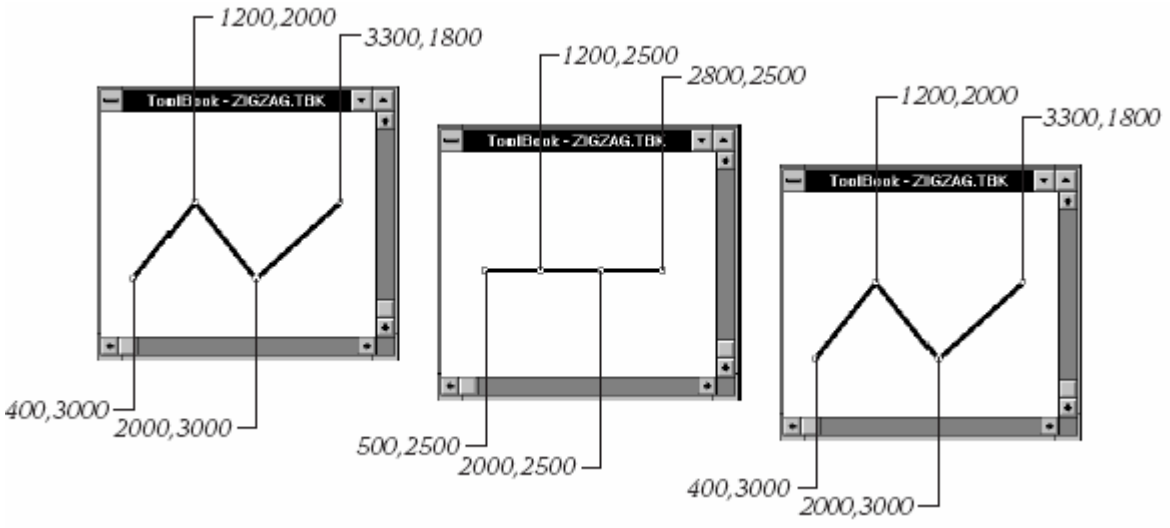


Figura 2.9: Cambiar líneas o curvas estableciendo sus propiedades *vértices*

Inversión de páginas

Moviéndose rápidamente sobre diferentes páginas que contienen diferentes vistas de un mismo gráfico se puede crear visualmente una ilusión de movimiento, para ésto se utiliza el comando *flip.*

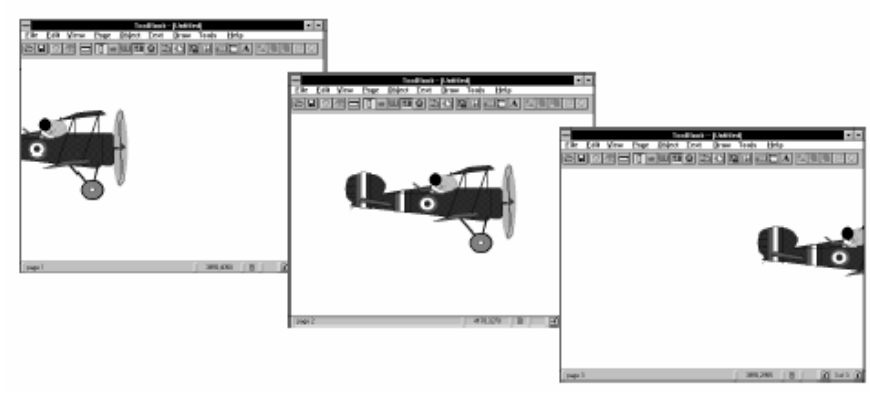


Figura 2. 10: Crear la ilusión de movimiento por inversión (*flipping)* de páginas

```
to handle buttonClick
        flip 3 pages
end
```

Mostrar y esconder objetos

Una animación puede ser creada con una serie de objetos que varíen entre ellos, se muestren y escondan simulando movimiento.

Figura 2.11: Animar un objeto mostrando y escondiendo una serie de objetos

En el gráfico anterior primero se muestra una pila de objetos uno sobre otro, luego se esconde todos excepto el primero y se sigue este proceso en cada nueva imagen construyendo la animación.

AÑADIR COMPORTAMIENTO DRAG-AND-DROP

Este efecto especial puede añadirse a cualquier tarea o acción de la aplicación, se hace un *click* sobre un objeto (llamado objeto fuente), se arrastra la imagen del objeto fuente a otra ubicación y se suelta la imagen sobre otro objeto.

Para añadir comportamiento *drag-and-drop* se debe incluir propiedades y mensajes especiales.[1]

Dentro de una página o visor cualquier objeto como campos, *listboxes*, *combo boxes*, etc. puede ser arrastrado. Sin embargo es imposible arrastrar objetos como páginas, *backgrounds*, libros.

Cuando un usuario arrastra un objeto, ToolBook automáticamente muestra la imagen correcta (*drag* o no *drop*), envía notificaciones que una operación *drag* (arrastre) ha iniciado o terminado, e identifica los objetos fuente y destino. Durante el proceso *drag-and-drop* no se actualizan cambios en la aplicación. Por ejemplo el objeto no se mueve cuando el botón del *mouse* se libera, el programador debe escribir un manipulador que defina si se borra el objeto fuente, se mueve o se cambia su color.

Para definir un comportamiento *drag-and-drop*, se debe:

- Identificar el objeto fuente, al hacer *click* el objeto fuente, ToolBook inicia la operación *drag-and-drop*.
- Identificar al menos un objeto destino donde se libere el objeto fuente, puede ser la misma página.
- Escribir un manipulador para un mensaje *endDrag* que defina que pasa cuando un usuario suelta la imagen arrastrada sobre el objeto destino, si por ejemplo se borra o se mueve el objeto fuente.

---

[1] Anexo G: Elementos *OpenScript* para operaciones *Drag-and-drop*

CREACIÓN DE EFECTOS DE SONIDO

Una forma de incluir sonido a la aplicación es utilizar el comando *beep*, ejemplo:

```
to handle buttonClick
        if pageNumber of this page < pageCount of this book then
                go next page
        else
                beep 1 –Especificar el número de repeticiones de beep
                request "Last page."
        end if
end buttonClick
```

Otra forma de añadir sonido es utilizando archivos de sonido WAV (*wave-audio*), con la función *playSound()*, como en el ejemplo:

```
to handle buttonClick
        get playSound("c:\windows\chimes.wav")
        request It
end
```

La function *playSound()* tiene un segundo parámetro <*wait*>, que determina si ToolBook espera que el sonido termine para continuar con el siguiente comando en el manipulador. Cuando el valor del parámetro es *true* se suspende el manipulador hasta que el sonido se termine. El valor por defecto del parámetro es *false*.

Para detener un sonido mientras se ejecuta, se llama la función *playSound()* con parámetro *null:*

```
to handle buttonClick
        get playSound("toolbook\samples\dbeat1.wav",false)
        pause 2 seconds
        get playSound(null)    --Detiene el archivo de sonido
end
```

CAPÍTULO 3
FUNDAMENTOS DE SEGURIDAD EN REDES DE DATOS

CIFRADO Y AUTENTICACIÓN

INTRODUCCIÓN

El objetivo de las redes de datos como Internet es permitir la comunicación y compartición de información entre usuarios repartidos por todo el mundo. Sin embargo, nadie desea que toda su información esté disponible para todo el mundo, por lo que a la hora de conectarse a una red hay que tomar una serie de medidas con el fin de evitar accesos no deseados. Pero además del acceso a la información confidencial existen otras amenazas que pueden ser incluso más peligrosas; por ejemplo, un ataque que deje fuera de servicio a sistemas críticos de la red pueden ocasionar mayores problemas, más allá de pérdidas económicas puede causar pérdida de credibilidad y baja satisfacción en el usuario.

La necesidad de ambientes computacionales seguros es cada vez más importante, para ésto se debe establecer medidas de seguridad adecuadas las que se anticipen a posibles fallas del sistema y de forma efectiva protejan los activos[1] de la red. Además de las normas de seguridad, también es importante utilizar tecnología adecuada y correctamente configurada para mantener un sistema de datos confiable. No obstante, es imposible llegar a tener una red totalmente libre de riesgos de seguridad.

Objetivo de seguridad en redes

La seguridad en redes es el proceso por el cual los activos de la información digital son protegidos. El objetivo de la seguridad está sintetizado en tres aspectos principales: Confidencialidad, Integridad y Disponibilidad; esto es,

[1] El activo más importante que se posee es la información

“proteger la Confidencialidad, mantener la Integridad, y asegurar la Disponibilidad” [10].

Confidencialidad: La información debe estar protegida de accesos no autorizados, disponible únicamente a personas predefinidas. Para garantizar la confidencialidad de la información generalmente se utiliza métodos de cifrado.

Integridad: La información no debe ser alterada o destruida, debe restringirse permisos de modificación a personal no autorizado. También se debe evitar la modificación de información no autorizada o accidental por parte de usuarios autorizados.

Disponibilidad: La información debe estar disponible para los usuarios autorizados en el momento que ellos la requieran. En caso de presentarse un problema en el *hardware* o *software* que interrumpa el funcionamiento de algún servicio de la red, se refiere a su inmediata y completa recuperación.

Términos importantes

- **Identificación:** Es el proceso de identificar una entidad de otra o determinar la identidad de una entidad con quién se está comunicando.
- **Autenticación:** Verificar que la identidad de una entidad es válida, probar que es quien dice ser. Puede ser a través de un *password*.
- **Contabilidad:** Seguimiento o registro de lo que una entidad o usuario hace en un sistema.
- **Autorización:** Controlar los niveles de acceso y privilegios que una entidad o usuario tienen en un sistema.
- **No repudio (*Nonrepudiation*):** Prevenir que usuarios o entidades nieguen la realización de un evento, como: envío, recibo, acceso o alteración de información o archivos.

Modelo de seguridad en redes

Seguridad por oscuridad

El primer modelo de seguridad que se aplicó es el de seguridad por oscuridad, basado en el desconocimiento u ocultamiento de lo que se desea proteger, funciona mientras realmente permanezca secreto, es decir que en la práctica puede funcionar por un tiempo limitado, porque a largo plazo se va a descubrir y su seguridad posiblemente va a ser violentada.

El perímetro de defensa

Este modelo tradicional de seguridad persigue obtener la seguridad basado en la separación de la red interna hacia fuera. Protege todos los puntos de acceso a la red, lo que es correcto y en la actualidad se mantiene; sin embargo, únicamente como parte de un modelo de seguridad más completo, en el que se analiza además la seguridad en equipos, recursos locales y todos los puntos intermedios de conexión. Los problemas principales de este modelo son: que no brinda seguridad frente a los ataques que se realicen desde la red interna y que no presenta un diferente nivel de protección en caso de que el ataque rompa la barrera de seguridad perimetral.

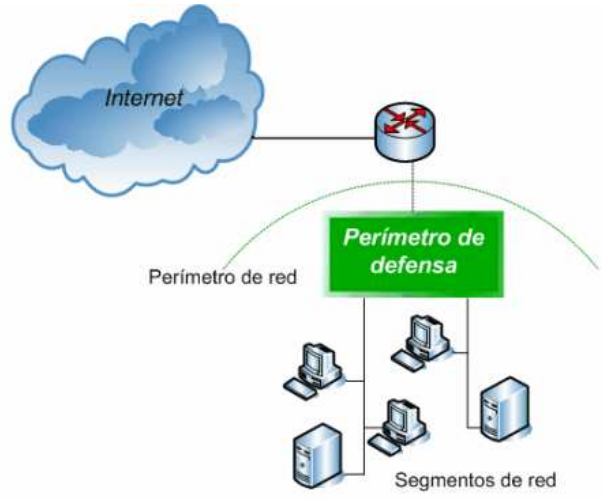


Figura 3.1: Estrategia de perímetro de defensa [9]

Defensa en profundidad

La defensa en profundidad implementa múltiples líneas de protección, subdivide la red en capas de tecnología de seguridad variada, las que se manejan de forma independiente y mutuamente se refuerzan para brindar máxima seguridad.

Poner en práctica el modelo de defensa en profundidad resulta de un análisis profundo y un tanto complejo, a continuación se presentan algunas pautas dadas por IATFF (*Information Assurance Technical Framework Forum*)[1], para su implementación:

- Tomar decisiones de seguridad basadas en análisis de riesgos y ajustadas a los objetivos operacionales de la organización.
- Trazar desde las tres facetas de la defensa en profundidad: personal, operaciones y tecnología. Las mitigaciones tecnológicas no tienen valor sin la gente entrenada para usarlas y los procedimientos operacionales para guiar su aplicación.
- Establecer un programa integral de educación, entrenamiento, experiencias prácticas y conocimientos. Licencias de certificación y profesionalización proveen un grupo técnico validado y reconocido de administradores del sistema.
- Explotar los productos comerciales disponibles.
- Evaluar periódicamente la postura de la IA (*National Information Assurance*)[2] frente a la infraestructura de la información. Herramientas tecnológicas, tales como *scanners* automatizados para redes, pueden ayudar en la valoración de vulnerabilidades.
- Considerar, no solo las acciones de aquellos con intentos hostiles, sino también las acciones accidentales o descuidadas.

[1] Organización patrocinada por la NSA (*Nacional Security Agency*), da soporte al intercambio técnico entre la industria, instituciones académicas y agencias de gobierno de Estados Unidos en asuntos de aseguramiento de información

[2] Entidad del gobierno de Estados Unidos, que pretende resolver las necesidades de pruebas de seguridad de la tecnología de información de consumidores y productores

- Emplear múltiples medios de mitigación de amenazas, el solapamiento de protección intenta contrarrestar anticipadamente eventos de modo que pérdidas o fallas de una sola barrera no comprometan la infraestructura total de la información.
- Asegurarse que solamente el personal digno de confianza tenga acceso físico al sistema. Métodos que proporcionan tal seguridad incluyen apropiadas investigaciones del entorno, permisos de seguridad, credenciales y distintivos.
- Utilizar procedimientos establecidos para reportar la información de incidentes proporcionada por los mecanismos de detección de intrusos a las autoridades y centros de análisis especializados.

Metodología de defensa en profundidad [9]

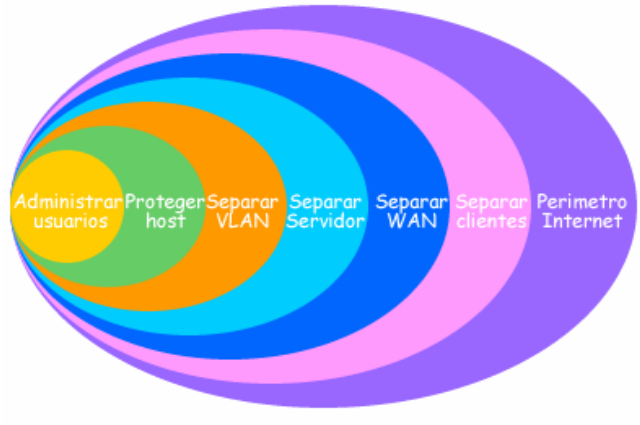


Figura 3.2: Metodología de defensa en profundidad

- **Perímetro de Internet:** El Internet contiene muchos peligros, pero la mayoría de ataques son internos.
- **Separación de usuarios:** Asumir que cualquier usuario o *host* fuera del control de la asociación es inseguro.
- **Separación de redes de área extendida (WAN)**: Establecer criterios para acceso entre *host* y servidores.
- **Separación de servidores:** Ubicar en un lugar de mayor seguridad a los objetivos de valor elevado.
- **Separación de redes de área local virtual (VLAN):** Confiable pero separado, separar usuarios por áreas de trabajo.

- ***Host* Protegidos:** Las características por defecto son el principal objetivo de los atacantes.
- **Administración de usuarios:** La vigilancia y el conocimiento de seguridad de los usuarios puede ser crucial para que todos los otros controles de seguridad sean efectivos.

Vulnerabilidades

Una vulnerabilidad es una debilidad en la seguridad de un sistema, ya sea en su diseño, implementación o administración; puede convertirse en un ataque cuando se descubre y es utilizada con fines maliciosos, provocando graves violaciones a la seguridad del sistema. [6]

Vulnerabilidad física: Es una debilidad en la seguridad física de los equipos de la red.
Vulnerabilidad en *hardware* y *software*: Fallas de diseño de *hardware* y *software*.
Vulnerabilidad en los medios: Es una debilidad en la seguridad de los medios donde se almacena la información.
Interceptación de la información: Interceptar y monitorear información de forma no autorizada.
Vulnerabilidad humana: Negligencia entre los usuarios de la red.

Amenazas

Las amenazas son peligros potenciales que pueden romper las medidas de seguridad, e interrumpir operación, integridad o disponibilidad de la red. Pueden ser accidentales o intencionadas:

- **Accidentales:** Provocados por desastres naturales, o causados por negligencia de los usuarios.
- **Intencionadas:** De forma premeditada se producen ataques a la red, con algún fin malicioso.

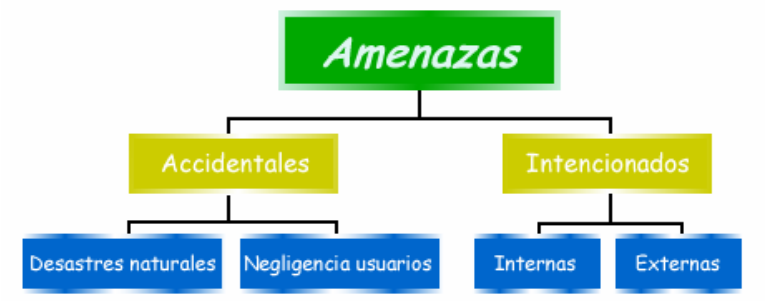


Figura 3.3: Tipos de amenazas [13]

Las amenazas intencionadas o ataques, pueden ser clasificados de la siguiente manera:

Virus: Son programas que no pueden funcionar de forma independiente, están embebidos en otro programa o archivo de datos, son diseñados para copiarse a otros archivos cuando el archivo infectado es abierto o ejecutado. Un antivirus mantiene una base de datos de los virus reconocidos por medio de una “firma” que permite identificar al virus.

Gusanos: Son programas independientes, capaces de reproducirse mientras se propagan a otros *host*. La principal diferencia con los virus es su capacidad de sobrevivir de forma independiente, aunque en la actualidad muchas veces se trata estos dos términos como análogos.

Troyano: Un troyano o Caballo de Troya es un programa que se enmascara como una aplicación legítima, mientras de forma secreta también desempeña otra función. El programa secreto es comúnmente un virus o un programa malicioso que crea una puerta de ingreso (*back door*) al sistema, que en futuro el atacante usará para acceder a la red.

Puerta trasera: Se denomina así a los ataques que fácilmente logran acceder a la red, saltando los mecanismos de control de ingreso. Puede ocurrir usando módems *dial-up* o conexiones externas asíncronas. También puede lograrse con la modificación de un programa que abra una puerta de acceso secreta al sistema.

Bomba lógica: Es un programa malicioso que se activa por un evento específico, como: una condición lógica, un tiempo específico, una fecha, etc.

Rastreo de puertos (*port scanning*): Es un *software* que explora los puertos para determinar que servicios están activos, y utilizar esa información para acceder al sistema.

***Spoofing*:** Un intruso engaña al sistema suplantando la identidad de alguien conocido para acceder al sistema.

- En *spoofing* de dirección IP (*Internet Protocol*) se altera el paquete a nivel TCP (*Transmisión Control Protocol*), el atacante descubre la dirección de un *host* válido y la utiliza como dirección fuente en los paquetes que envía, el *host* destino la reconoce como una dirección legítima y acepta sus paquetes.

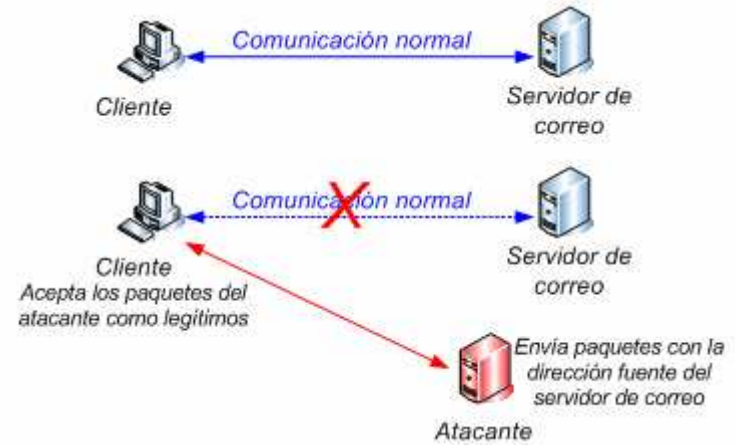


Figura 3.4: *Spoofing* de dirección IP

Hombre en el medio: Un ataque donde se impide la comunicación directa entre dos entidades autorizadas, el hombre en la mitad logra ubicarse entre los dos *host* y redireccionar la comunicación para que los paquetes que se envían desde una fuente al destino primero pasen por él, pudiendo leer éstos e incluso alterarlos y luego reenviarlos.

Figura 3.5: Hombre en el medio

***Pishing*:** Un tipo de ataque de hombre en la mitad que ha ganado mucha popularidad en los últimos años, se trata de crear páginas Web con dominios que varían ligeramente (puede ser una letra, ejemplo www.amazin.com) y con apariencia idéntica al original, así cuando el usuario ingrese creerá que es un sitio correcto (www.amazon.com) e ingresará datos confidenciales, de esta forma el atacante puede robar números de tarjetas de crédito, *passwords*, etc.

Ataques al DNS (*Domain Name System*): se tiene los siguientes tipos de ataques al DNS.

- **Ataque a DNS simple:** Cuando la víctima desea abrir un sitio Web cuya entrada de dirección IP no existe en la tabla ARP (*Address Resolution Protocol*) del cliente, se realiza un pedido al servidor DNS local. El atacante observa esta petición DNS y envía una respuesta falsa a la víctima, la respuesta es aceptada por la víctima y la verdadera respuesta al llegar segunda es descartada.

Figura 3.6: Ataque DNS

- ***Cache poisoning*:** El objetivo de este ataque es redireccionar el tráfico y enviarlo a una página diferente, se logra alterando en la tabla de conversión de URL la dirección IP del servidor DNS.

***Replaying*:** Usar un mensaje interceptado por el atacante en el futuro para tratar de obtener privilegios de otra persona o equipo de la red.

***Eavesdropping*:** Interceptar y leer mensajes intentando ser otra persona o equipo, la información mantiene su integridad pero la privacidad o confidencialidad es comprometida. Una red es más propensa a este ataque cuando incluye componentes inalámbricos y dispositivos de acceso remotos.

Ingeniería social: Usa las habilidades sociales para obtener información, por ejemplo, por medio de una llamada telefónica o un *e-mail* el atacante simula ser una persona encargada de actualizar la base de datos o de las operaciones de mantenimiento, solicitando a los usuarios su nombre y *password*. Este tipo de ataques puede ser muy peligroso, la mejor defensa contra estos es la creación de políticas que atiendan este tipo de ataques y con la educación de los usuarios.

Negación de servicio: Impide que miembros autorizados accedan a un recurso de la red. Ejemplos de ataques de este tipo:

- *Buffer overflow*: Un proceso recibe más datos de los que su memoria puede soportar. El ping de la muerte envía paquetes ICMP (*Internet Control Message Protocol*) causando un desbordamiento y consecuentemente fallas en el sistema.
- SYN *flooding*: El atacante inunda el sistema con requerimientos de conexiones, sesiones TCP inician utilizando espacio en el *buffer* y hacen que el sistema quede fuera de servicio mientras espera una respuesta que nunca llega.
- *Teardrop*: Los campos de longitud y fragmentación en la secuencia de paquetes IP son modificados provocando confusión en el sistema y fallas en su operación.

- *Smurf*: Se envía un paquete ICMP con la dirección de la víctima como fuente y la dirección de *broadcast* de la red como destino, todas las máquinas responden al requerimiento y saturan con sus respuestas a la víctima.
- *Spam*: Es un *mail* no deseado, puede llegar a inundar un sistema y sobrecargar su ancho de banda y espacio de disco, logrando la caída del sistema.

***Sniffing*:** Es un proceso de monitoreo de la red que pretende obtener información como *passwords* o direcciones IP que pueden usarse para atacar a la red.

CIFRADO

Cifrar un mensaje es utilizar un algoritmo para hacerlo incomprensible a cualquier persona no autorizada a recibirlo, únicamente su destinatario va a poseer la información secreta necesaria para restaurar el mensaje a su forma natural texto plano o claro. La información secreta se llama clave o llave (*key*), y su función es similar a una llave de una cerradura que permite que el mensaje se abra y su receptor pueda leerlo. El cifrado de datos permite garantizar la privacidad, la autenticación de las identidades que forman parte de la comunicación, y la integridad del mensaje.

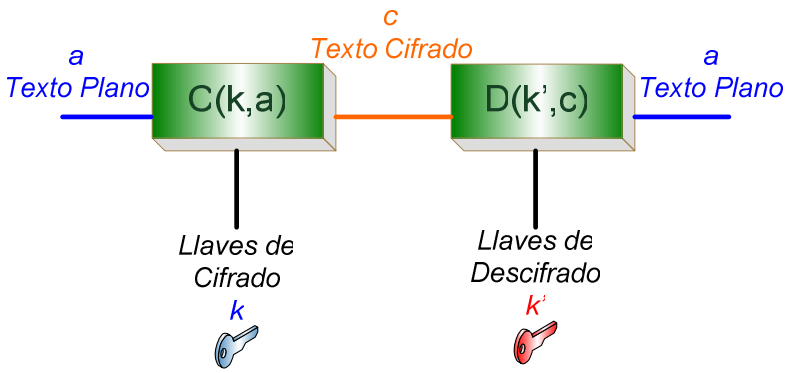


Figura 3.7: Cifrado

Texto plano: Datos en forma original, legible o descifrada.

Llave: Una pieza aleatoria de datos usados en el cifrado y descifrado.

Cifrado: Proceso de tomar el texto plano y usar una llave para convertir éste en un texto cifrado único. Este proceso debe ser reversible.

Texto cifrado: Datos cifrados, en forma no legible.
Descifrado: Tomar el texto cifrado y usar una llave para convertir éste en el texto plano original.

Ruptura de códigos

Existen algunos métodos, a continuación se da una breve idea de estos:

Ataque de texto plano conocido: La llave de cifrado puede descubrirse si se conoce un segmento del texto plano y su correspondiente texto cifrado.
Ataque de determinado texto plano conocido: Basado en la habilidad de conseguir el cifrado de un mensaje en texto plano determinado.
Criptoanálisis: Estudio matemático que en base al texto cifrado trata de obtener el texto plano, sin el uso de la llave de cifrado.
Fuerza bruta: Trata de probar con todas las posibles combinaciones de llaves para romper un código.

Existen dos tipos de cifrado: simétrico y asimétrico.

Cifrado de llave simétrica

También llamado cifrado de llave privada o secreta. Utiliza la misma llave para cifrado y descifrado, o la llave para descifrar es fácilmente calculable en base a la llave de cifrado.

Su principal problema es la seguridad, el envío de la llave sin que sea descubierta, la comunicación entre emisor y destinatario debe estar protegida de ataques de *eavesdropping*. Una ventaja que presenta el cifrado de llave simétrica es su rapidez frente al cifrado de llave asimétrica.

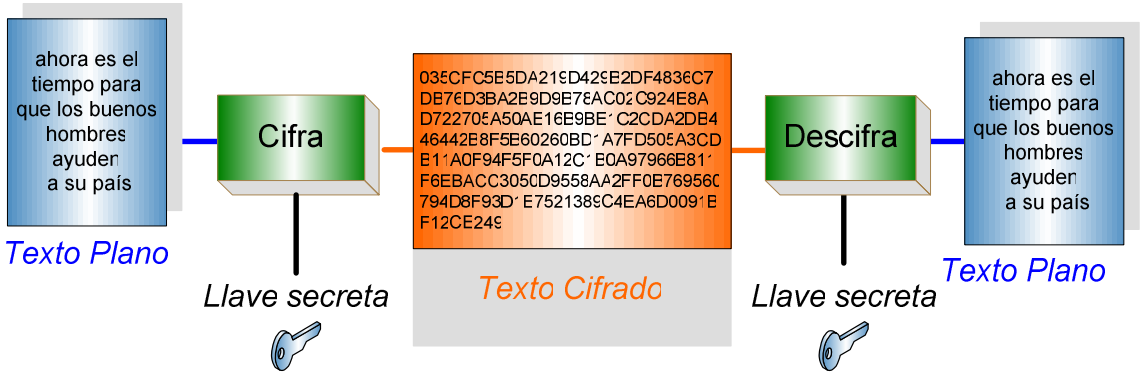


Figura 3.8: Cifrado de llave simétrica [7]

Cifrado de llave simétrica puede agruparse en cifrado continuo y cifrado por bloques.

Cifrado continuo

Cifra un bit o byte a la vez, a cada unidad del texto plano le corresponde una unidad de texto cifrado, el proceso de cifrado se basa en el orden de las unidades precedentes. Actúa a mayor velocidad que el cifrado en bloques y su *hardware* presenta menor complejidad.

Su principal debilidad es que los patrones de repetición que presenta el texto plano pueden reflejarse en el texto cifrado. También es susceptible a ataques de sustitución, donde una fracción de un mensaje antiguo es insertada en un nuevo mensaje.

Cifrado por bloques

Opera sobre un grupo de bits de longitud fija, llamados bloques. Generalmente se utiliza un bloque de 64 bits o de 128 bits. Cada bloque se procesa de forma independiente de los otros. Sin embargo para evitar posibles ataques que descubran la información, se utiliza modos de cifrado. Los modos de cifrado hacen que el texto cifrado de un mismo texto plano no sea siempre el mismo.

Ejemplos

DES (*Data Encription Standar*), 1977: Uno de los algoritmos más antiguos y usados. Creado por IBM y respaldado por la NSA (*National Security Agency*). Tiene un bloque de 64 bits y utiliza una llave de 56 bits, la que en la actualidad puede ser fácilmente descubierta con procesos de fuerza bruta. Para brindar más seguridad se creo una variable de DES llamada 3DES la que utiliza boques de cifrado triples con generalmente dos diferentes llaves, resultando una llave de 112 bits de longitud.

AES (*Advanced Encription Standar*), 2000: Utiliza un algoritmo llamado Rijndael, maneja llaves de 128, 192 y 256 bits. El tamaño de su bloque de datos es de 128 bits.

RC4 (Rivest Cipher 4): Cifrado continuo, con llave de tamaño variable (generalmente de 128 bits). Es usada en Netscape Navigator e Internet Explorer.

Cifrado de llave asimétrica

El cifrado de llave asimétrica pretende resolver los problemas de administración de llaves que se presentan en el cifrado de llave simétrica.

Se manejan dos llaves por cada miembro de la comunicación, una llave pública y una privada, como su nombre lo indica la llave pública es publicada mientras la privada se mantiene en secreto, conocer la llave pública no revela la llave privada. La comunicación se basa únicamente en el uso de la llave pública, la llave privada jamás es transmitida o compartida, eliminando los problemas de seguridad que se presentan al enviar o recibir información secreta. La llave pública es usada para cifrar un mensaje, pero el mensaje sólo puede ser descifrado usando la llave privada, la que estará en posesión del destinatario deseado.

El cifrado de llave pública tiene dos aplicaciones primarias, cifrado (privacidad de la información) y firmas digitales (autenticación).

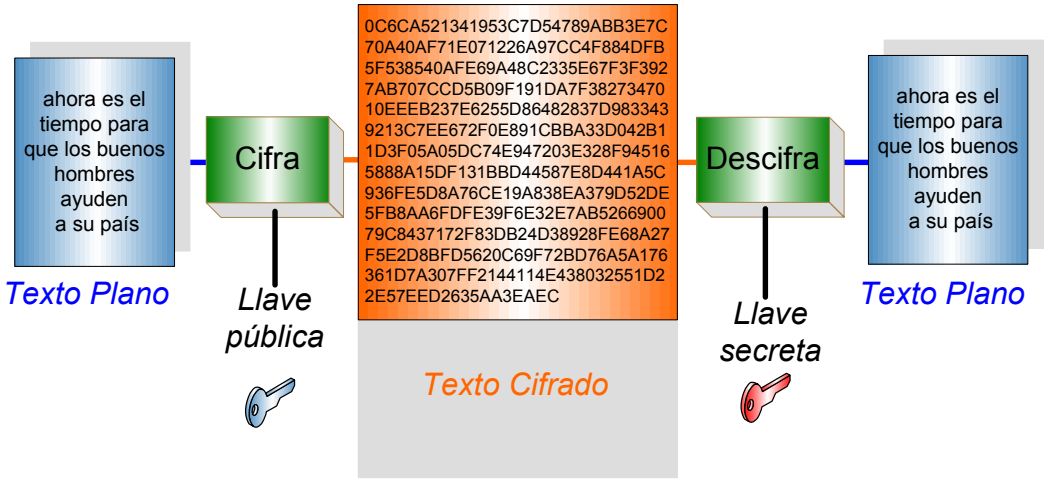


Figura 3.9: Cifrado de llave asimétrica [7]

Ejemplos

Diffie-Hellman, 1976: Whitfield Diffie y Marti Hellman introducen el uso de la llave pública. Puede ser usada para establecer una llave secreta usada por las dos partes en cifrado simétrico. Es usado en los protocolos de administración de llaves en IPSec[1].

RSA (Rivest, Shamir, Adelman): Multiplica largos números primos para generar llaves. Provee capacidad de firmas digitales.

Diferencias entre cifrado de llave simétrica y asimétrica

- EL cifrado de llave asimétrica aumenta la seguridad, ya que no requiere la transmisión de llaves privadas.
- El cifrado de llave asimétrica provee autenticación, mediante firmas digitales. Evita la negación de eventos.
- El cifrado de llave simétrica provee métodos de cifrado más rápidos.
- El cifrado de llave simétrica no permite autenticación.
- Dependiendo del tipo de comunicación se puede escoger entre cifrado de llave simétrica o asimétrica, el cifrado simétrico puede ser eficaz en ambientes de

[1] Ver más adelante 3.3.2

transmisión de llave privada seguros y la cifrado asimétrica en ambientes multiusuario abiertos.

- El cifrado de llave asimétrica es un complemento de la simétrica que brinda mayor seguridad.

AUTENTICACIÓN

Es la forma de garantizar que un usuario o equipo es quien dice ser, o que la información no ha sido manipulada por entidades no autorizadas. Existen tres tipos de técnicas de autenticación de la identidad de usuarios, las que se pueden usar de forma individual o combinada:

1. Algo que solamente el individuo conoce: por ejemplo un *password*, clave secreta, código, etc.
2. Algo que el individuo posee: por ejemplo una llave, una tarjeta.
3. Algo que el individuo es[1]: por ejemplo huellas digitales, voz, iris del ojo.

Firmas digitales[2]

Es un proceso usado para autenticar la identidad del emisor y verificar la integridad del mensaje. La firma digital es enviada junto con el mensaje, se crea cifrando con la llave privada del emisor el valor *hash* del mensaje. El receptor utilizando la llave pública del emisor realiza un proceso de comprobación de la firma.

El valor *hash* es único y se obtiene al realizar la función *hash* del mensaje (texto plano), asegura la integridad del mensaje. MD4, MD5[3] y SHA-1 (*Secure Hash Algoritm-1*) son algunos algoritmos de *hashing*.

[1] Relacionado con autenticación biométrica
[2] El estándar X.509 de la ITU documenta exactamente la certificación de llave pública
[3] Message Digest 5 RFC 1321

Certificados digitales

Un certificado digital permite autenticar una identidad, ligando la entidad con una llave pública. Una tercera entidad en la comunicación entre el emisor y receptor llamada AC (Autoridad Certificadora) la que debe ser bien conocida y confiable, es la que envía el certificado, el cual debe estar firmado digitalmente.

La administración de los certificados digitales presenta ciertas limitaciones, por ejemplo con el manejo de los certificados expirados o revocados.

Figura 3.10: Ejemplo de certificado digital

Autoridad Certificadora

Una organización encargada de emitir, renovar y revocar certificados digitales. Su papel es fundamental en transacciones electrónicas, ya que garantiza que las dos entidades que intercambian información son quién dicen ser.

Algunas de las principales CA son: VeriSing Inc., GTE, AT&T y Microsoft.

La Autoridad Certificadora utiliza una PKI (*Public Key Infraestructure*) jerárquica. Una PKI se encarga de autenticar certificados digitales y CAs. Las principales limitaciones son la integración de organizaciones que utilizan esquemas diferentes y el desarrollo de una infraestructura global.

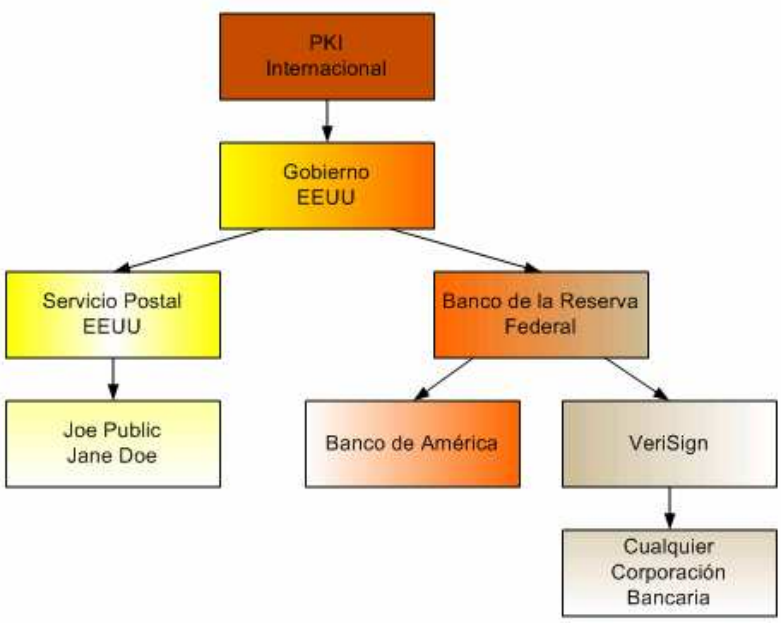


Figura 3.11: PKI Jerárquico Teórico [7]

Otros mecanismos de autenticación

Nombre de usuario y *password*: Es uno de los métodos de autenticación más comunes, el nivel de seguridad que brinde depende en gran parte del manejo que le de el usuario, pero comúnmente es susceptible a ataques.

***Password* de una sola vez OTP (*One-Time Password*):** El sistema de *password* de una sola vez se crea usando algoritmos de *hashing*. Un nuevo *password* se requiere cada vez que un usuario ingresa a la red, El *password* se envía en texto plano sin peligro de ataques de *eavesdropper* ya que luego de que fue usado perderá su validez.

***Token cards* y servidores:** Es otro método de OTP que añade mayor seguridad. Una *token card* es similar a una tarjeta de crédito, es programada para un usuario específico y cada usuario tiene un PIN único que genera un *password* que corresponde estrictamente a la tarjeta. El servidor recibe y verifica el *password* para autenticar al usuario.

Kerberos[1]: Permite la autenticación de requerimientos de recursos de red, sin intercambio de *passwords*. El usuario se autentica localmente y en base a su perfil un servidor emite credenciales que le permiten acceder a determinados servicios. Utiliza el algoritmo DES para cifrado y autenticación.

SEGURIDAD EN LA CAPA ACCESO DE RED

También conocida como capa *host*-red, permite efectuar un enlace físico real con los medios y enviar paquetes IP sobre la red. Esta capa incluye los detalles de la tecnología LAN, WAN y de las dos primeras capas (física y enlace de datos) del modelo OSI (*Open System Interconnection*).

Existen muchos protocolos que operan en esta capa, a continuación se revisará algunos de los más importantes y la seguridad que proveen.

ATM (ASYNCHRONOUS TRANSFER MODE)

ATM es una tecnología de red orientada a conexión, su arquitectura es basada en celdas[2]. Permite transmitir voz, video y datos a través de redes públicas y privadas a alta velocidad (155 Mbps) y con diferentes niveles de calidad de servicio. Soporta *multicast* y conexiones punto a multipunto.

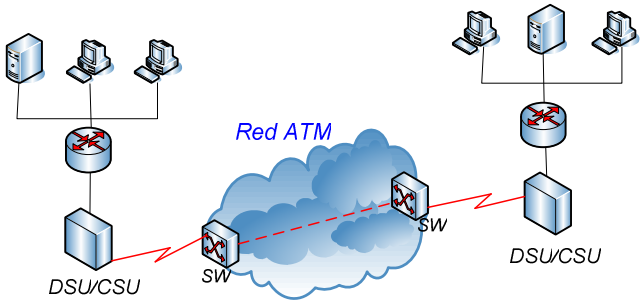


Figura 3.12: Red ATM

[1] Información detallada en el RFC 1510
[2] Paquete de datos de tamaño fijo, en ATM de 53 bytes

DSU/CSU: *Digital Service Unit / Channel Service Unit.*
SW: *Switch* ATM.

En ATM se utilizan SA (Agentes de Seguridad) para proveer y negociar los servicios de seguridad[1].

ATM tiene tres planos de intercambio de datos: usuario, control y administración.

Plano de usuario

En el plano de usuario, se transmite los datos junto con cierta información de control, y se tiene los siguientes servicios de seguridad:

- Autenticación de entidad.
- Autenticación de datos.
- Confidencialidad de datos.
- Integridad de datos.
- Intercambio de llaves.
- Intercambio de certificados y CRL (Lista de Revocación de Certificados).
- Control de acceso.

La autenticación de entidad, se realiza durante el establecimiento de la conexión para esto se utiliza un protocolo de *handshake*[2] que puede ser de dos direcciones para autenticación unilateral y de tres direcciones para autenticación mutua. El protocolo de *handshake* de tres vías provee intercambio de certificados y negociación de servicios de seguridad. En este proceso de autenticación también se controla el acceso ya que aquí se autoriza o niega los requerimientos de conexión.

El intercambio de llaves se realiza por medio del mensaje ConfPar (parámetros confidenciales) donde se transportan las llaves cifradas. La llave *master* es

[1] ATM Forum Security Specification definen la completa funcionalidad de seguridad.
[2] Intercambio de mensajes al inicio de una sesión.

enviada por ambas entidades para acordar parámetros computacionales de generación e intercambio de la llave de confidencialidad e integridad de sesión, la que es obtenida de forma aleatoria. Para garantizar la seguridad luego de un período de establecida la sesión se puede hacer una actualización de llaves de sesión.

La integridad y autenticación de los datos también se establecen al inicio de la conexión con el uso de la llave de integridad de sesión. Esto da protección de ataques de *replaying*.

Plano de control

En el plano de control, se transmiten señales de control de conexión. ATM utiliza un canal común para señalización. La seguridad que se puede proveer a este nivel es:

- Integridad.
- Autenticación.

La integridad y autenticación se realiza de la misma manera que en el plano de usuario.

Plano de administración

En el plano de administración, se realizan funciones de administración y coordinación.

En la transmisión de celdas ATM de forma periódica se envía una celda especial OAM (*Operation, Administration, and Maintenance*) con mensajes de seguridad, los que permiten resincronización de información (cifrado y actualización de llaves).

Seguridad multicast

En una conexión punto a multipunto se establecen los parámetros de seguridad al inicio de la sesión al igual que en una conexión punto a punto. Para evitar el desperdicio de recursos se puede utilizar el protocolo de *hadshake* de tres vías únicamente entre la primera conexión (entidad que inicia la sesión y la que primero responda), donde se generaría la llave *master* y se compartiría a las otras entidades que van a formar parte de la conexión utilizando el protocolo de dos vías. Sólo el que inicia la sesión puede actualizar las llaves de sesión.

VPN (Virtual Private Network)

ATM usa rutas fijas a través de la red llamadas circuitos virtuales, estos se forman en la red ATM por medio de enlaces definidos entre los *switches*. Los agentes de seguridad de cada *switch* pueden negociar el servicio de confidencialidad (cifrado de los datos de la celda) para formar una VPN.

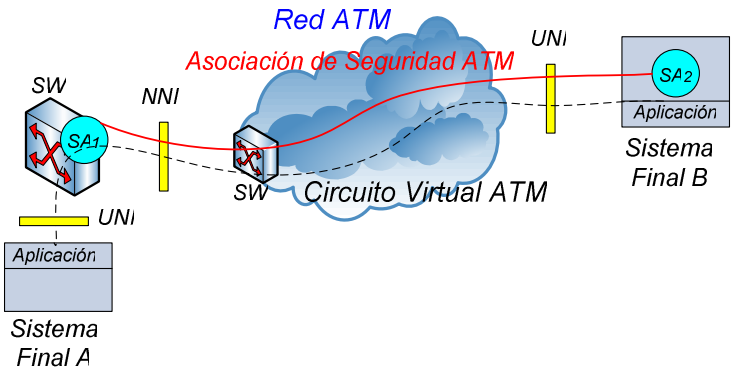


Figura 3.13: Red privada virtual ATM [8]

UNI: *User-to-Network Interface.*
NNI: *Network Network Interface.*

SA1 sirve como Proxy de seguridad al sistema final A porque éste establece una asociación de seguridad con el agente SA2 sobre el sistema final B. La conexión ATM (circuito virtual) es establecida extremo a extremo entre el sistema final A y B.

PPP (POINT TO POINT PROTOCOL)

El protocolo PPP es utilizado en conexiones seriales[1], con comunicación síncrona y asíncrona. Fue diseñado para trabajar con varios protocolos de capa red simultáneamente sobre el mismo enlace. [15]

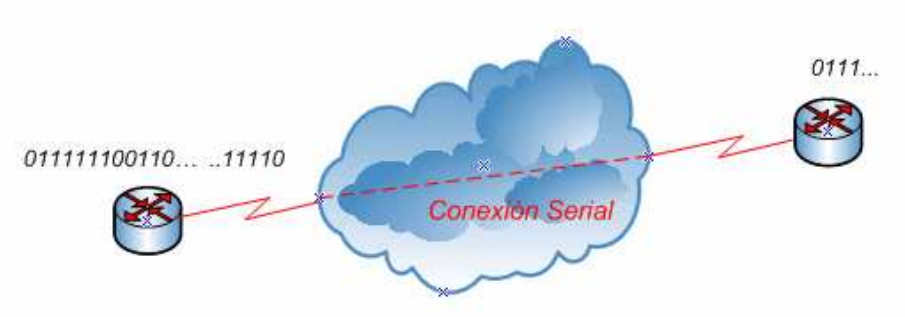


Figura 3.14: Conexión PPP

PPP esta subdividido en dos capas:

LCP (*Link Control Protocol*: Es responsable del establecimiento del enlace, además negocia otros parámetros de control como autenticación y control de errores.

NCP (*Network Control Protocol*): Permite escoger y configurar los distintos protocolos de capa de red. Para cada protocolo de capa red se proporciona un NCP distinto.

Protocolo de capa red	NCP
IP (*Internet Protocol*)	IPCP (*IP Control Protocol*)
IPv6 (*Internet Protocol versión 6*)	IPv6CP (*Ipv6 Control Protocol*)
IPX (*Internetwork Packet Exchange*)	IPXCP (*IPX Control Protocol*)

Tabla 3.1: Protocolos de Control de Red

[1] Los bits de una trama se transmiten uno por uno a lo largo del medio físico.

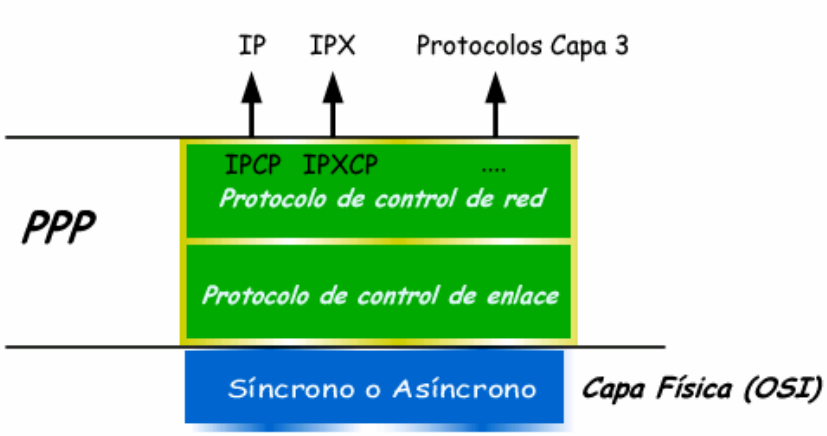


Figura 3.15: Capas de PPP [11]

Autenticación

La autenticación en PPP es una fase opcional que se realiza luego del establecimiento del enlace y del protocolo de autenticación a usarse, por defecto no se requiere autenticación. Existen dos opciones: PAP (*Password Authentication Protocol*) y CHAP (*Challenge Handshake Authentication Protocol*).

PAP

PAP es un protocolo de *handshake* de dos vías, la entidad a ser autenticada envía en conjunto su nombre de usuario y contraseña varias veces hasta que se recibe un mensaje de aceptación de la otra entidad o la conexión se termina. Las contraseñas son enviadas en texto plano lo que hace a este protocolo inseguro, además no ofrece protección más allá de la fase de autenticación.

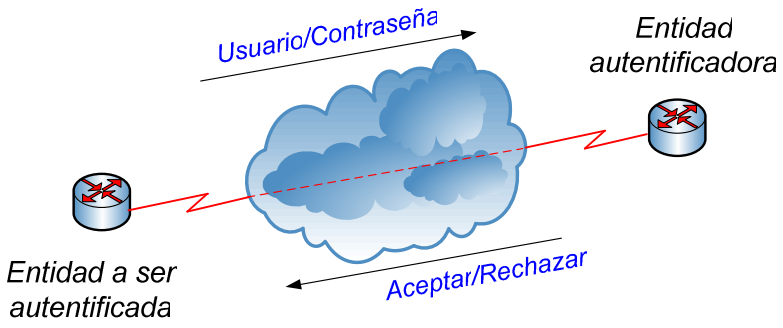


Figura 3.16: PAP

CHAP

CHAP es un protocolo de *handshake* de tres vías, únicamente permite autenticar a la entidad remota; sin embargo, durante la fase de establecimiento del enlace se puede acordar la autenticación de las dos entidades que intervienen en la comunicación. Este protocolo utiliza una contraseña secreta compartida entre las dos entidades.

La entidad autenticadota envía un mensaje de comprobación a la entidad remota, este debe ser impredecible para evitar posibles ataques. La entidad a ser autenticada responde con un mensaje calculado con la función *hash* MD5, obtenido en base a la contraseña y el mensaje de comprobación. Finalmente el autenticador recibe el mensaje, calcula de la misma manera la respuesta y compara los valores, si es correcto se autentica a la entidad remota o de lo contrario se termina la conexión.

CHAP es repetido periódicamente para autenticar continuamente a las entidades, brindando mayor seguridad que PAP. Tanto PAP como CHAP no brindan seguridad en el intercambio de mensajes de datos.

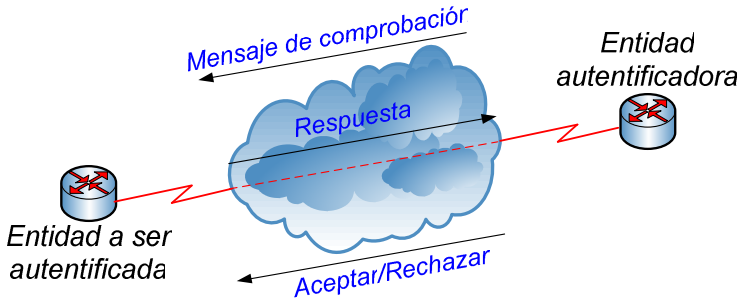


Figura 3.17: CHAP

EAP (Extensible Authentication Protocol)

Este protocolo soporta múltiples mecanismos de autenticación. Con EAP el mecanismo de autenticación se establece en la fase de autenticación, no al momento de establecer el enlace. El autenticador puede requerir mayor

información antes de seleccionar el mecanismo de autenticación. Los mecanismos usados pueden ser: MD5-desafío (similar a CHAP), OTP, *token card*.

ECP (Encryption Control Protocol)

Este protocolo protege la confidencialidad de los datos transportados dentro del datagrama PPP. Utiliza un algoritmo de cifrado simétrica (puede ser DES), el cual es negociado junto con otros parámetros, luego de la fase de autenticación y antes de la transmisión de datos. El proceso de negociación de cifrado no está protegido.

L2TP (LAYER 2 TUNNELING PROTOCOL)[1]

L2TP transporta en un túnel[2] tráfico PPP sobre varias redes como IP, ATM, etc. Opera en la capa enlace de datos del sistema OSI, por lo que su implementación se realiza nodo a nodo en la red.

El protocolo L2TP asegura interoperabilidad entre distintos fabricantes, se basa en protocolos propietarios de Cisco y Microsoft, como L2F (*Layer Two Forwarding*)[3] y PPTP (*Point to Point Tunneling Protocol*)[4] respectivamente.

El túnel se establece entre dos puntos finales L2TP. Varias conexiones lógicas PPP pueden ser establecidas sobre el túnel. L2TP es un protocolo orientado a conexión.

A continuación se muestra un ejemplo de L2TP, Los paquetes IP del emisor se encapsulan en tramas PPP y se envían sobre una conexión PPP al ISP (Proveedor de Servicios de Internet). El ISP como cliente L2TP recibe las tramas PPP, las encapsula en paquetes L2TP y envía al *host* de la red destino que

[1] Información detallada en el RFC 2661
[2] Un camino de comunicación seguro entre dos pares
[3] RFC 2341
[4] RFC 2637

ejecuta el *software* de servidor L2TP. El servidor L2TP remueve la encapsulación (L2TP y PPP) y envía el paquete IP al receptor ubicado en su misma red LAN.

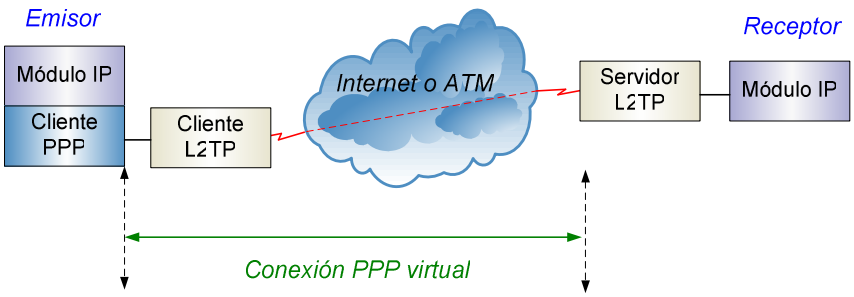


Figura 3.18: Ejemplo de L2TP [8]

L2TP diferencia entre paquetes de datos y paquetes de control (usados para establecer el túnel y la sesión).

Los parámetros de seguridad de L2TP son los de autenticación y cifrado proporcionados por PPP, sin embargo propiamente en L2TP no se utiliza mecanismos de seguridad que protejan a los paquetes de datos y control.

SEGURIDAD EN LA CAPA INTERNET VERSIÓN 4

La capa Internet selecciona la mejor ruta para enviar paquetes y se encarga de la conmutación de los mismos. Su función es similar a la de la capa Red en el modelo OSI.

IP versión 4 es el principal protocolo que funciona en esta capa, enruta paquetes hacia su destino sin importar su contenido y de forma no orientada a conexión. No realiza verificación y corrección de errores, deja esa labor a protocolos de capas superiores.

A continuación se analizan los principales mecanismos de seguridad que se presentan en esta capa, los que especialmente tratan de manejar las vulnerabilidades añadidas al manejo de direcciones IP.

FILTROS DE PAQUETES

También conocidos como *firewalls*[1] de capa de red, su función es impedir el paso de paquetes a menos que cumplan con las reglas de acceso impuestas.

Un *router*[2] con capacidad de configurar ACL (Listas de Control de Acceso) se convierte en un filtro de paquetes, generalmente se ubica en la frontera de la red interna y externa. Permite controlar el flujo de datos entrante y saliente.

Filtros basados en direcciones IP

Se basa en la dirección origen y destino de un paquete para permitir o negar su paso. La Figura 3.19 muestra un ejemplo de reglas de filtraje basadas en direcciones IP.

Regla	Dirección IP Fuente	Dirección IP Destino	Acción
1	176.14.10.*	156.18.5.*	permitir
2	156.18.5.*	176.14.10.*	permitir
3	*.*.*.*	*.*.*.*	negar

El asterisco indica que puede ser cualquier valor entre 0 y 255.

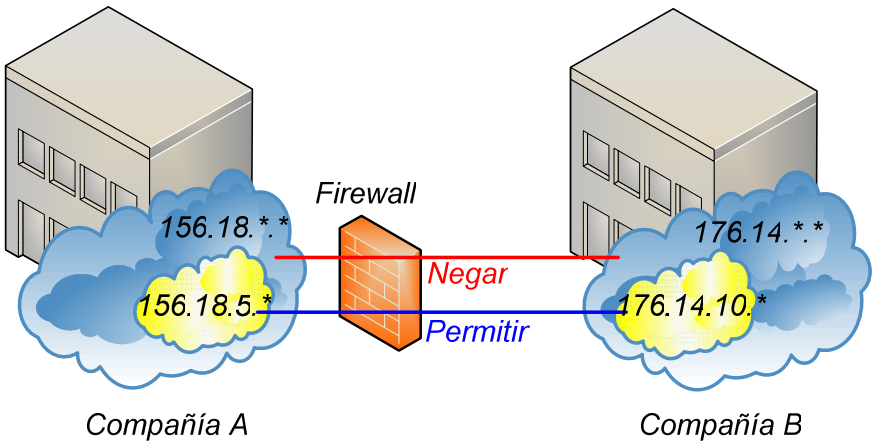


Figura 3.19: *Firewall* de filtraje de paquetes [8]

[1] *Software* o *hardware* que trata de garantizar la seguridad de la red
[2] Dispositivo de capa red, que determina la mejor ruta para el envío de datos

Filtros basados en direcciones IP y números de puerto

Para brindar un mayor control de acceso además de utilizar un filtro de direcciones IP, se añade un parámetro más el número de puerto TCP o UDP[1]. Como se muestra en la Tabla 3.2.

R	Conexión	Tipo	Dirección IP fuente	Dirección IP destino	Puerto Fuente	Puerto destino	Acción
1	entrada	TCP	Externa	interna	>=1024	25	permitir
2	entrada	TCP	Interna	externa	25	>=1024	permitir
3	Salida	TCP	Interna	externa	>=1024	25	permitir
4	Salida	TCP	Externa	Interna	25	>=1024	permitir
5	cualquier	Cualquier	*.*.*.*	*.*.*.*	cualquiera	cualquiera	negar

Tabla 3.2: Reglas de filtraje basadas en direcciones IP y puertos TCP/UDP

Conexión de entrada: Es iniciada desde un cliente sobre un *host* externo.
Conexión de salida: Es iniciada desde un cliente sobre un *host* interno.

La tabla anterior presenta reglas de envío y recepción de mensajes de correo electrónico en la intranet. SMTP (*Simple Mail Transfer Protocol*) suministra servicios de correo electrónico. Un servidor SMTP escucha sobre el puerto 25.

Otras acciones además de permitir o negar una conexión pueden establecerse como autenticación de usuario o cifrado, a continuación se muestra una regla la que permite una conexión FTP (*File Transfer Protocol*) si la autenticación de usuario es exitosa.

Fuente	Destino	Servicio	Acción
..*.*	156.18.5.92	FTP	Autenticación de usuario

Tabla 3.3: Filtraje basado en conexiones

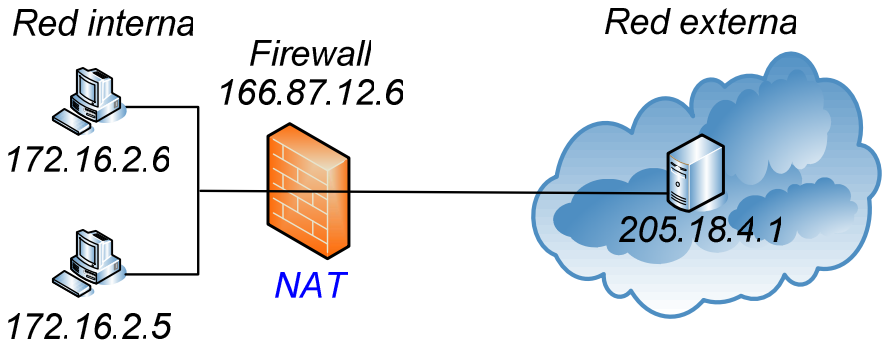
[1] Protocolo de Datagrama de Usuario, de la capa transporte no orientado a conexión. RFC 768

NAT (Network Address Translation)

Figura 3.20: Operación de NAT

Es un mecanismo que permite reemplazar la dirección IP de un paquete por otra. Generalmente se configura en un equipo de frontera entre la red interna y externa como un *firewall* o *router*.

En la Figura 3.21 se observa la operación de NAT:

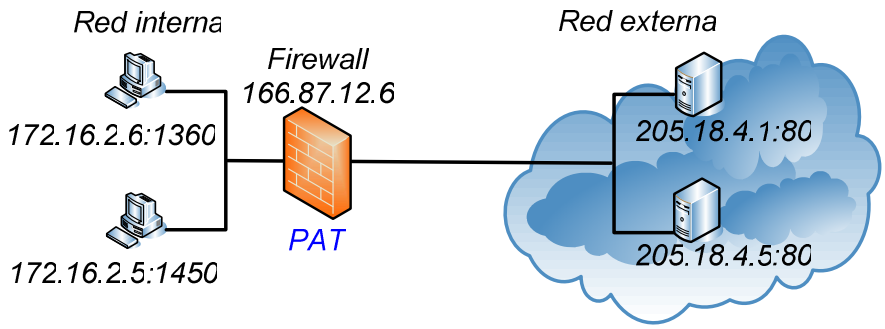


Red interna		NAT	Red externa	
Dirección IP Fuente	Dirección IP Destino		Dirección IP fuente	Dirección IP destino
172.16.2.5	205.18.4.1	→	166.87.12.6	205.18.4.1
205.18.4.1	172.16.2.5	←	205.18.4.1	166.87.12.6

Figura 3.21: Traducción de direcciones de red

La dirección IP fuente de un paquete se traduce de una dirección IP privada a una dirección IP pública, y de la misma forma sucede con los paquetes de respuesta. Este mecanismo permite una conservación de las direcciones de IP las cuales son limitadas y además da cierta seguridad a los dispositivos de la intranet ya que su dirección IP permanece oculta.

PAT (*Port Address Translation*) es una variación de NAT que permite que varías direcciones privadas internas se traduzcan con una solo dirección publica externa, con el uso de puertos diferentes.



Red interna		PAT	Red externa	
Dirección IP fuente	**Dirección IP Destino**		**Dirección IP fuente**	**Dirección IP destino**
172.16.2.6:1360	205.18.4.1:80	→	166.87.12.6:1360	205.18.4.1:80
172.16.2.5:1450	205.18.4.5:80		166.87.12.6:1450	205.18.4.5:80
205.18.4.1:80	172.16.2.6:1360	←	205.18.4.1:80	166.87.12.6:1360
205.18.4.5:80	172.16.2.5:1450		205.18.4.5:80	166.87.12.6:1450

Figura 3.22: Traducción de direcciones de puerto

IPSec (INTERNET PROTOCOL SECURITY)

IPSec es una extensión de IP que asegura su comunicación, para esto utiliza un conjunto de protocolos de seguridad y algoritmos. Provee integridad de datos, autenticación y confidencialidad; así como, asociaciones de seguridad y administración de llaves. Es utilizado para formar VPNs en ambientes de intranets e Internet. [14]

Los protocolos de seguridad de IPSec son AH (*Autentication Header*)[1] y ESP (*Encapsulating Security Payload*)[2], estos protocolos pueden ser usados tanto en IPv4 e IPv6.

[1] RFC 2402
[2] RFC 2406

Protocolos de seguridad

ESP encapsula los datos, pero no proporciona protección a las cabeceras externas, brinda confidencialidad ya que cifra los datos (*payload*), integridad de los datos, servicio *anti-replay* y opcional autenticidad del origen de datos. Para la cifrado utiliza DES o 3DES u otro esquema de cifrado simétrica.

AH protege al datagrama completo, toma a la cabecera dentro de los datos, permite verificar la integridad del datagrama IP, autenticar el origen de los datos y provee protección opcional contra ataques de *replay*. AH puede aplicarse solo o en conjunto con ESP.

Asociaciones de Seguridad

Una SA (Asociación de Seguridad) es una conexión unidireccional que permite negociar los parámetros de seguridad para proteger el tráfico de la red. Sólo se aplica un protocolo de seguridad por cada SA.

Existen dos tipos de SAs, modo transporte y modo túnel:

- Modo transporte es una asociación de seguridad entre dos *host*.
- Modo túnel es un asociación de seguridad donde uno o las dos partes de la comunicación son *gateway* de seguridad (sistema intermedio que actúa como interfaz entre dos redes)

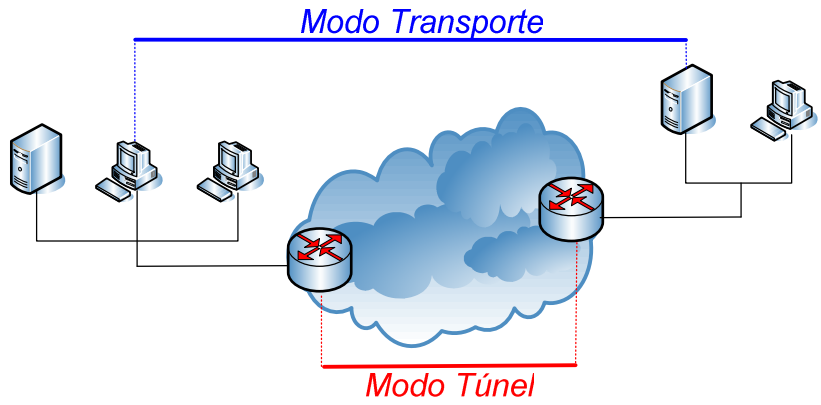


Figura 3.23: Modo transporte y modo túnel

Administración de llaves

IKE (*Internet Key Exchange*)[1] es por defecto el protocolo de administración de llaves, su función principal es establecer y mantener las asociaciones de seguridad. Está basado en ISAKMP (*Internet Security Association and Key Management Protocol*) el cual es un esquema para autenticación e intercambio de llaves[2].

SEGURIDAD PARA DNS

El DNS es un sistema que almacena asociaciones entre direcciones IP y nombres de dominios de *host*. Su seguridad principalmente requiere que el sistema sea diseñado y configurado correctamente.

La mayoría de ataques a DNS tratan de direccionar el tráfico de datos a un sitio incorrecto. Una forma de lograr su objetivo es ingresar al registro DNS y modificarlo, así la respuesta del servidor DNS será una dirección IP incorrecta. Otra forma es suplantar al verdadero servidor DNS, y responder a una solicitud de resolución de dirección IP con un valor falso.

Para prevenir ataques al DNS se presenta un esquema de seguridad donde se provee la autenticación del origen de datos de un registro en el servidor DNS.

NIDS (NETWORK BASED INTRUSION DETECTION SYSTEM)

Un sistema de protección contra intrusos debe proveer los siguientes mecanismos de defensa:

- Detección: Identificar ataques maliciosos sobre recursos de la red y el *host*.
- Prevención: Parar la ejecución del ataque detectado.
- Reacción: Inmunizar el sistema de futuros ataques de fuentes maliciosas.

---

[1] RFC 2409
[2] Ver más adelante 3.4.3

Un IDS está formado por uno o varios sensores y un analizador los cuales pueden estar separados o formar parte de un mismo componente. Los sensores recolectan información desde puntos de entrada a segmentos críticos de red, y la envían al analizador. El analizador examina la información aplicando alguno de los métodos de detección de intrusos. Los métodos de análisis son:

- **Detección de anomalías:** Se busca conocer lo que es normal en un sistema, de esta forma una condición anómala es considerada como una intrusión.
- **Detección de usos indebidos:** Se trata de conocer directamente lo que es anormal y se puede detectar. Se utiliza firmas las cuales son una descripción de ataques conocidos para identificar posibles intrusos.

Un NIDS inspecciona el tráfico entrante o saliente a nivel de red, si detecta una acción maliciosa realiza una acción correctiva propia o notifica al administrador del sistema para que realice la acción correspondiente. Su configuración es realizada por el administrador del sistema en base a las políticas de seguridad establecidas.

Es de crucial importancia que la detección del intrusos se realice tan pronto como sea posible, mejor si es en tiempo real, antes de que los recursos del sistemas sean comprometidos. La protección que brinda un NIDS es tanto para penetraciones externas como internas.

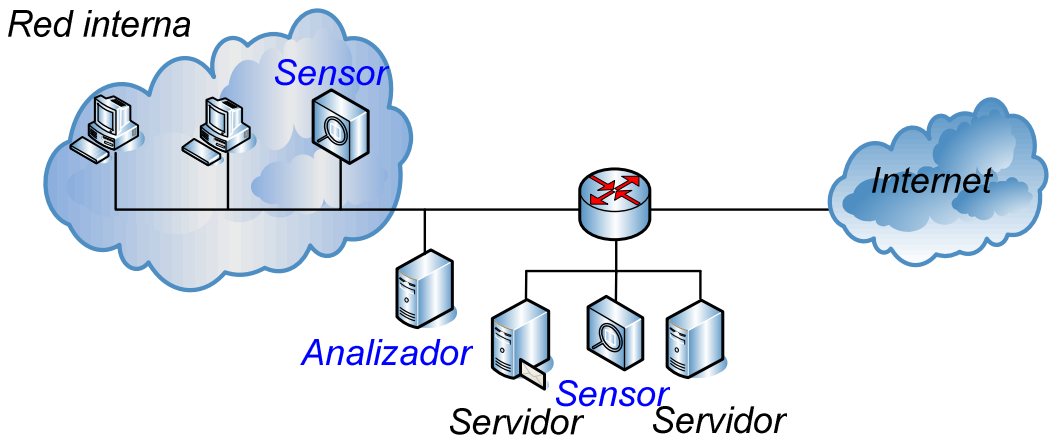


Figura 3.24: Ejemplo de NIDS

Si el ataque logró realizarse, de forma inmediata se debe activar un mecanismo de recuperación que mitigue sus efectos.

SEGURIDAD EN LA CAPA TRANSPORTE

La capa transporte proporciona servicios de transporte de datos desde el *host* origen hacia el *host* destino. Forma una conexión lógica entre los extremos de la comunicación.

A continuación se muestra ciertos mecanismos que proporcionan seguridad a nivel de esta capa.

PROXY A NIVEL DE CIRCUITO

Un *proxy* es un servidor que se ubica entre el cliente de una aplicación y un servidor real, permitiendo su comunicación de manera controlada. Actúa como un "hombre en el medio", de modo que no hay contacto directo entre un cliente y el servidor de la aplicación.

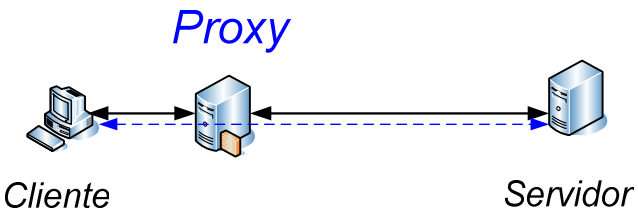


Figura 3.25: Proxy

Un *proxy* a nivel de circuito funciona creando un circuito[1] entre un cliente y un servidor sin interpretación de la naturaleza del requerimiento. El cliente debe ejecutar un *software* de cliente especial para que el *proxy* a nivel de circuito funcione. Uno de los servicios de circuitos más ampliamente usados es SOCKS.

[1] Ruta de comunicación entre dos o más puntos

SOCKS5

SOCKS es un protocolo estándar diseñado para manejar tráfico TCP a través de un servidor *proxy*. Existen dos versiones: SOCKS4 y SOCKS5. La versión SOCKS5 a diferencia de la versión SOCKS4, soporta aplicaciones basadas en UDP e incluye seguridad adicional mediante autenticación. Este protocolo opera entre la capa transporte y aplicación.

SOCK5 proporciona funciones de *firewall* básicas, porque autentica paquetes de entrada-salida y puede proveer NAT.

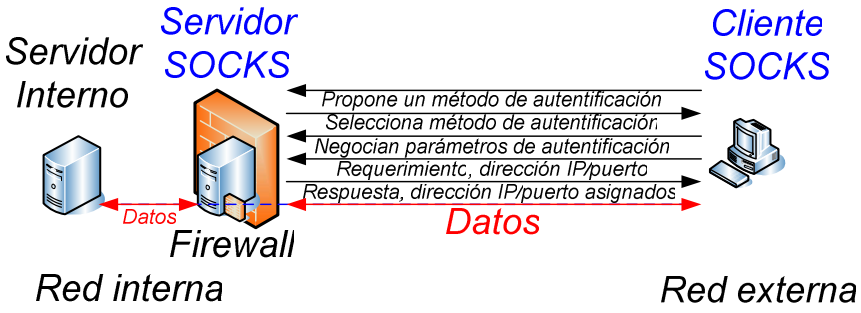


Figura 3.26: Protocolo SOCKS. [8]

En la figura un cliente basado en TCP quiere establecer una conexión a un proceso que se ejecuta sobre un *host* interno que es accesible sólo vía un *firewall*. Primero el cliente debe establecer una conexión TCP al servidor SOCKS sobre el *host firewall*, el servidor SOCKS escucha en el puerto 1080.

- El cliente propone uno o más métodos de autenticación.
- El servidor SOCKS escoge un método y notifica al cliente.
- El servidor y cliente negocian e intercambian parámetros de autenticación.
- El cliente envía al servidor el requerimiento de conexión, en este caso la dirección IP destino y puerto TCP.
- La respuesta del conector contiene el número de puerto que el servidor asignó para conectarse con el *host* objetivo, y la dirección IP asociada.

Luego del proceso de autenticación y establecimiento de conexión exitoso, los datos son transmitidos entre el *host* externo y el *host* interno.

SSL (SECURE SOCKECTS LAYER)

SSL fue desarrollado por Netscape, con el propósito de proveer seguridad cuando se transmite información sobre Internet. Es ampliamente usado, sin embargo no se ha considerado como una propuesta de estándar en IETF (*Internet Engineering Task Force*)[1].

Utiliza cifrado de llave asimétrica y simétrica para establecer y transferir datos en modo seguro sobre una red insegura. Se utiliza para establecer una sesión segura entre el *browser* del cliente y el servidor WWW (*World Wide Web*), usualmente HTTP sobre SSL (HTTPS).

La integridad de la información es establecida por algoritmos de *hashing* y la confidencialidad asegurada con cifrado.

Establecimiento de sesión segura con SSL

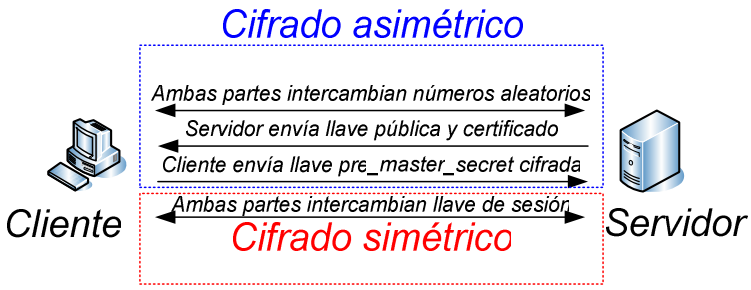


Figura 3.27: Establecimiento de sesión SSL [7]

- Las dos partes de la comunicación intercambian números aleatorios.
- El servidor envía su llave pública con un certificado digital firmado por una CA reconocida, además envía una ID de sesión.

[1] Organización internacional abierta de normalización. Tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, tales como transporte, enrutamiento y seguridad

- El *browser* cliente crea una llave pre_master_secret, la cifra usando la llave pública del servidor y transmite al servidor.
- Las dos partes generan una llave de sesión usando la pre_master_secret y los números aleatorios.

El establecimiento de sesión inicia con cifrado asimétrico, para lograr una conexión segura y autenticar a las partes. Sin embargo, luego de generar e intercambiar la llave de sesión (llave privada), las dos partes cambian a cifrado simétrico. Esto se debe a que con cifrado simétrico se crea mucho menos *overhead* y en consecuencia se logra mayor rendimiento.

TLS (TRANSPORT LAYER SECURITY)[1]

El desarrollo de TLS se basa en el protocolo SSLv3, al ser una versión avanzada proporciona mejores características de seguridad. La arquitectura de los dos protocolos es bastante similar, sin embargo no se asegura la compatibilidad entre estos.

TLS proporciona integridad de datos, confidencialidad de datos, y autenticación de entidades de la comunicación. Se ubica sobre un protocolo de transporte confiable, tal como TCP. Consiste de dos capas, el Protocolo de Registro TLS y el Protocolo de Handshake TLS, como se muestra en la siguiente figura:

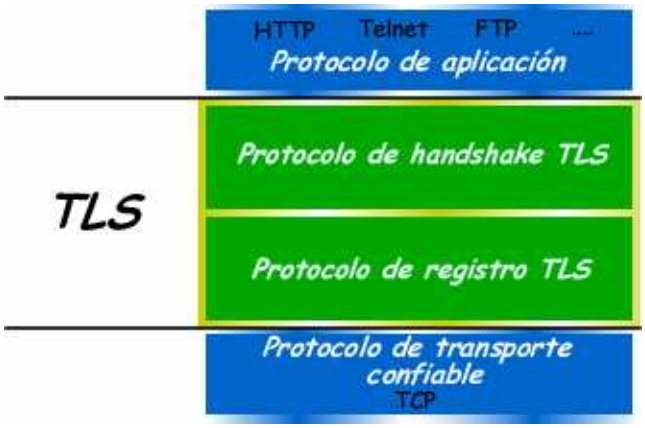


Figura 3.28: Capas TLS

[1] Información detallada en el RFC 2246

Protocolo de Registro TLS

El protocolo de registro TLS permite encapsular protocolos de alto nivel, incluyendo el protocolo de Handshake TLS.

Realiza el siguiente procedimiento con los mensajes a ser transmitidos:

- Fragmenta los datos en bloques manejables.
- Comprime los datos (opcional).
- Aplica un código de autenticación del mensaje.
- Cifra y transmite el resultado.

Protocolo de Handshake TLS

El protocolo de Handshake TLS permite la mutua identificación cliente-servidor y negociación del algoritmo y llaves de cifrado, antes que los datos sean transmitidos.

Consiste de tres protocolos:

- El protocolo de *hanshake*: es el protocolo más importante por el cual el cliente y servidor acuerdan parámetros como: el algoritmo criptográfico, tipo de autenticación y generan llaves secretas.
- El protocolo de cambio de código: mensaje enviado por el cliente y servidor, para notificar que los siguientes mensajes van a ser protegidos bajo los nuevos parámetros de seguridad negociados.
- El protocolo de alerta: permite notificar que la conexión va a ser cerrada, o que ha ocurrido un error en la conexión.

TLS soporta tres tipos de autenticación: autenticación del servidor, autenticación de ambas partes y total anonimato.

ISAKMP[1]

ISAKMP proporciona un *framework* para el establecimiento y administración de asociaciones de seguridad de forma independiente del protocolo de seguridad. Algunos de los protocolos de seguridad usados pueden ser IPsec ESP, IPsec AH o TLS.

Puede implementarse ISAKMP sobre cualquier protocolo de transporte o sobre IP, en el caso de UDP que es un protocolo no confiable, se logra una administración de llaves confiable con el uso de este protocolo.

Su función es similar a SASL(Simple Authentication and Security Layer)[2], el que también permite el uso de diferentes mecanismos de seguridad, aunque ISAKMP es más complejo ya que maneja una asociación de seguridad múltiple.

Negociación

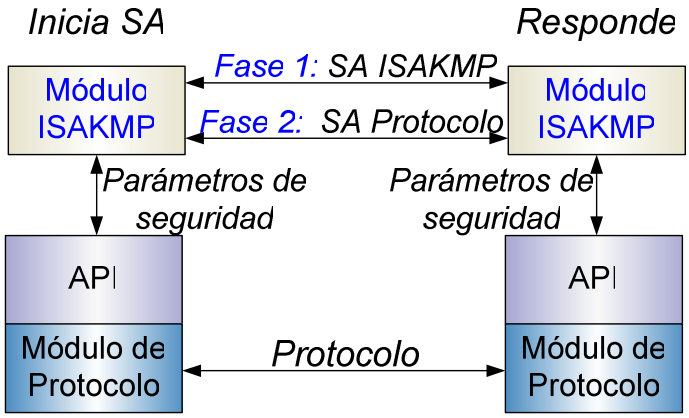


Figura 3.29: ISAKMP [8]

La negociación se realiza en dos fases. En la fase 1 se establece una SA entre las partes de la comunicación, la que define la forma de proteger los futuros mensajes de negociación. La fase 2 establece uno o varios SAs para el protocolo de seguridad que se va a usar. Luego de la fase 2 el protocolo de seguridad puede iniciar su ejecución.

[1] RFC 2408
[2] Mecanismo para añadir autenticación y autorización a protocolos orientados a conexión

El conjunto de atributos y parámetros que definen una SA son llamados dominio de interpretación, estos son:

- Formato de *payload.*
- Tipo de intercambio.
- Convenciones para nombrar a la información importante de seguridad.

SEGURIDAD EN LA CAPA APLICACIÓN

La capa aplicación de TCP/IP equivale a las capas Aplicación, Presentación y Sesión del modelo OSI. Se encarga del manejo de protocolos de alto nivel, entre los principales se tiene SMTP, HTTP, FTP, los que soportan aplicaciones como transferencia de archivos, *e-mail*, conexión remota, etc.

En la siguiente sección se revisa los conceptos generales y mecanismos de seguridad a este nivel.

PROXY DE APLICACIÓN

Un *proxy* de aplicación también conocido como *gateway* de aplicación es un mecanismo instalado sobre un *firewall* que actúa como intermediario entre un cliente y el servidor de la aplicación. Controla el establecimiento de conexiones y el tráfico que pasa entre una red confiable y una no confiable en la capa aplicación. Generalmente también proporciona autenticación del usuario, de la dirección fuente-destino y del protocolo.

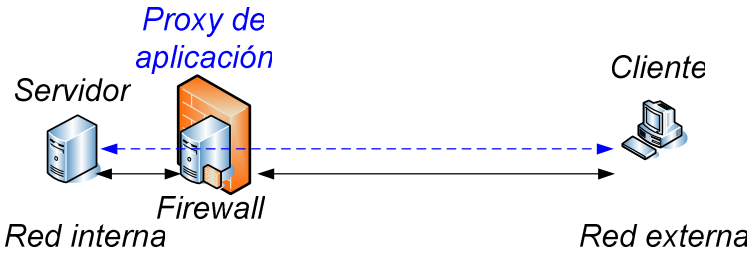


Figura 3.30: Proxy de aplicación

Para cada protocolo que permite atravesar el *firewall* (por ejemplo FTP, HTTP, TELNET[1]) se debe instalar un *proxy* especializado[2]. De esta forma se logra no afectar su desempeño.

Cuando el *proxy* de aplicación recibe una petición de un cliente dependiendo de su configuración puede procesarla y responder al cliente o reenviar la petición a otro servidor. En el último caso recibe la respuesta desde el servidor y la reenvía al cliente.

Los *proxies* de aplicación ofrecen una efectiva protección a la red, al ocultar completamente al servidor de la aplicación de cualquier usuario remoto. Sin embargo la velocidad se ve afectada ya que involucran una gran cantidad de procesamiento de datos.

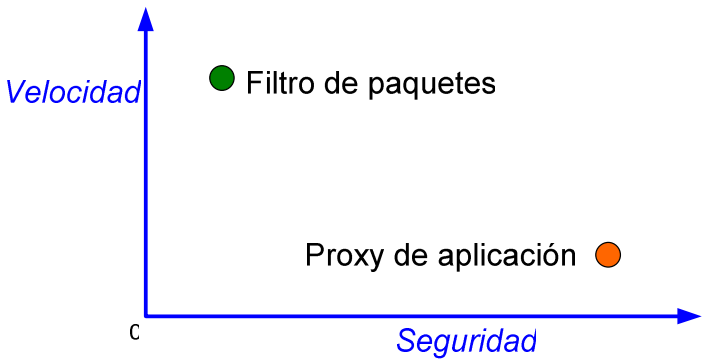


Figura 3.31: Comparación entre Filtro de paquetes y *Proxy* de aplicación

FILTROS DE CONTENIDOS

Los filtros de contenido son programas que se ejecutan sobre el *firewall* y analizan el tráfico en base a reglas a nivel de aplicación.

[1] Protocolo estándar de Internet para servicios de conexión de terminal remoto
[2] Un ejemplo de herramienta para personalizar el firewall es Firewall ToolKit (FWTK) de Trusted Information System http://www.tis.com/research/software/fwtk_over.html

CONTROLES DE ACCESO Y AUTORIZACIÓN

El control de acceso a la red y a sus recursos es el punto principal de seguridad de redes. Los principales protocolos para autenticación y autorización son RADIUS (*Remote Authentication Dial-In User Service*) o TACACS (*Terminal Access Controller Access Control System*), estos se implementan en la mayoría de productos de *firewall*.

RADIUS[1]

RADIUS ofrece un sistema de seguridad distribuida, asegura el acceso remoto a redes y a sus servicios, de accesos no autorizados. Es un protocolo basado en UDP. Está formado principalmente por las siguientes partes: cliente, NAS y servidor RADIUS.

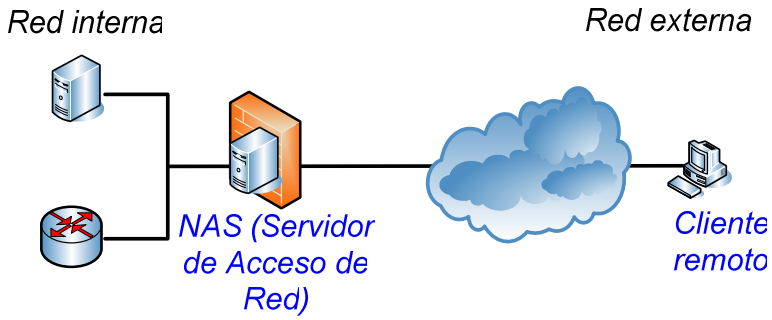


Figura 3.32: Elementos de RADIUS

- **Cliente:** Usuario remoto, que se conecta al NAS para solicitar un servicio como PPP o TELNET.
- **NAS (*Network Access Server*):** Procesa los requerimientos de conexión de los clientes remotos. Envía la información del cliente (usuario, *password*) al servidor RADIUS y actúa sobre la respuesta que se retorne. Proporciona el servicio al cliente si la respuesta es positiva o rechaza este si la respuesta es negativa. Además proporciona información de contabilidad al servidor RADIUS para propósitos de documentación.

[1] RFC 2865 - 2868

- **Servidor RADIUS:** Compara los datos recibidos desde el NAS con los de su base de datos segura para proporcionar autenticación y autorización. Envía una respuesta al NAS. Soporta varios métodos de autenticación como PAP o CHAP.

La seguridad del protocolo se garantiza con el uso de llaves secretas en la comunicación entre el NAS y el servidor RADIUS. El *password* del cliente se cifra para ser enviado entre el NAS y el servidor RADIUS.

TACACS[1]

Es un protocolo que proporciona autenticación de acceso remoto mediante un mecanismo de validación centralizado que implementa AAA (*Authentication, Authorization, and Accounting*). Es un protocolo basado en TCP.

Una versión mejorada de TACACS es TACACS+, la que habilita una función para manejo de *passwords* dinámicos, es decir que van cambiando en el tiempo y no son estáticos como los que se maneja en TACACS. Además permite proporcionar los servicios AAA de forma independiente, manejando una base de datos para cada servicio o en conjunto con los otros servicios disponibles sobre el servidor o la red.

Está compuesto por los siguientes elementos: cliente, NAS y servidor TACACS+. Estos elementos realizan funciones similares a las de RADIUS.

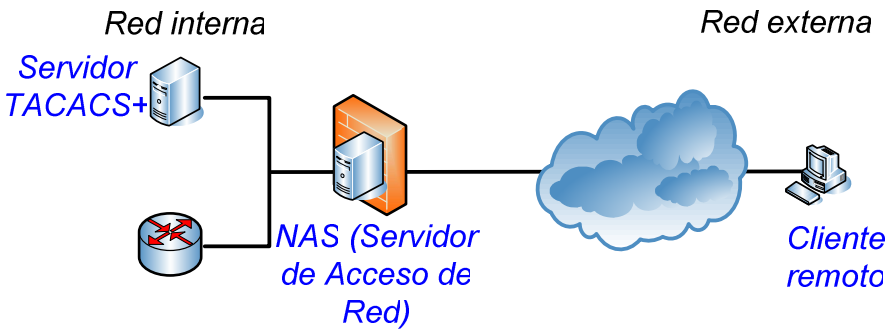


Figura 3.33: Elementos de TACACS+

[1] RFC 1492

- **Cliente:** El usuario remoto que desea acceder a la red.
- **NAS:** Procesa los requerimientos de conexión, obtiene la información del cliente y la transmite al servidor TACACS+.
- **Servidor TACACS+:** Autentica y autoriza el requerimiento de acceso. También recibe información de contabilidad desde el NAS.

Diferencias entre RADIUS y TACACS+

La funcionalidad básica de los dos protocolos RADIUS y TACACS+ es bastante similar, sin embargo existen ciertas diferencias que se muestran en la Tabla 3.4.

	RADIUS	**TACACS+**	
Funcionalidad	Autenticación / Autorización Combinada	AAA separado	TACACS+ permite mayor flexibilidad en su implementación al separar las funciones AAA.
Protocolo de transporte	UDP	TCP	RADIUS simplifica la implementación cliente/servidor, pero hace que el protocolo sea menos robusto.
Confidencialidad	Cifra sólo el *Password*	Cifra todo el paquete	TACACS+ es más seguro, cifra el paquete entero.
Contabilidad	Extensiva	Limitada	Los registros de contabilidad de RADIUS pueden contener mayor información que los de TACACS+.

Tabla 3.4: Comparación TACACS+ y RADIUS [10]

SEGURIDAD EN SISTEMAS OPERATIVOS

Todos los ataques se realizan con la finalidad de afectar a un dispositivo de red, muchas veces ese dispositivo es un *host* o servidor. Es por esto, que la seguridad en un sistema operativo es tan importante como la de cualquier otro control de

seguridad. Sin embargo la complejidad de los sistemas operativos hace imposible lograr una seguridad completa en ellos.

A continuación se analizan algunos mecanismos que se deben considerar para manejar la seguridad de manera general en cualquier sistema operativo:

- Configurar el monitoreo de *logins*, *logins* fallados, y toda la actividad de la red en cada sistema que conforma la red y en especial en sistemas críticos como servidores.
- Revisar periódicamente el archivo de *log*, si es posible diariamente para ver los intentos de acceso a la red y prevenir futuros ataques.
- Respaldar los archivos de *log* constantemente, una copia impresa puede ser una buena opción ya que existe la posibilidad de que los archivos sean alterados.
- Controlar el acceso y definir los niveles de privilegios de uso de aplicaciones en cada sistema.
- Utilizar mecanismos de cifrado sobre información de administración y monitoreo, por ejemplo en la base de control de acceso.
- De forma periódica realizar pruebas que chequeen las vulnerabilidades del sistema, para esto existen productos comerciales que se pueden utilizar.
- Instalar únicamente los componentes, servicios y programas necesarios, constantemente revisar y borrar lo que no se esta utilizando.
- Actualizar las versiones del sistema operativo.

La seguridad integral de la red mejora con la implementación de un modelo de seguridad en profundidad. Este modelo reduce los daños que un atacante puede causar en la red, ya que protege la red y los sistemas con una estrategia multi-nivel, es decir que si un atacante logra atravesar el perímetro de defensa encontrará más obstáculos en el camino que le impedirán ejecutar sus ataques. Además se logra proteger a los sistemas de ataques internos.

Seguridad en Windows

Windows es un sistema operativo muy utilizado, es un componente importante en la seguridad de una red o sistema. Su gran difusión se debe en gran parte a la facilidad de configuración y uso, siendo esto su principal problema de seguridad.

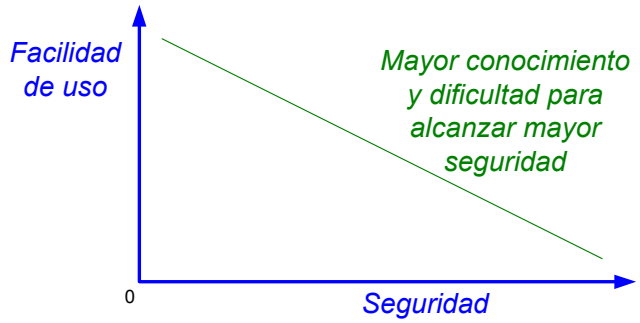


Figura 3.34: Relación entre facilidad de uso y seguridad

Microsoft cada vez ha hecho sus productos más fáciles de instalar, las nuevas versiones de Windows requieren menos conocimientos sobre cómo trabaja un computador para su instalación inicial. Esta ha sido una buena estrategia de negocios para Microsoft, sin embargo no es lo mejor en seguridad para el usuario.

Una instalación donde el usuario se limite a presionar “acepto” y “continuar”, probablemente instalará componentes innecesarios. Estas vulnerabilidades son aprovechadas por *hackers* para efectuar ataques de forma exitosa. Esto no quiere decir que Windows sea un sistema inherentemente inseguro, con una adecuada configuración y conocimiento de su funcionalidad es posible mejorar su seguridad.

Recomendaciones

Microsoft recomienda para mejorar la seguridad básicamente lo siguiente:[1]

1. Usar un *firewall* de Internet.
2. Instalar las actualizaciones, recomendable activar las actualizaciones automáticas.
3. Usar un *software* antivirus actualizado.

[1] Mayor información en www.microsoft.com/security/protect

Seguridad en UNIX y Linux

Los sistemas operativos UNIX y Linux principalmente han sido utilizados sobre servidores y en plataformas de desarrollo de software, pero en la actualidad en especial Linux ha ganado espacio como sistema operativo de estaciones de trabajo.

Estos sistemas operativos presentan características especiales que pueden hacer de estos muy seguros o vulnerables, al igual que Windows su nivel de seguridad dependerá del conocimiento y habilidad del administrador del sistema para configurarlo y operarlo.

Sin embargo, los usuarios de UNIX generalmente son más experimentados y técnicos, lo que hace de su sistema menos vulnerable o por lo menos es más difícil lograr que un ataque sea exitoso.

Un aspecto que hace de este sistema no atractivo para los atacantes es la gran cantidad de versiones disponibles las cuales difieren en su código, es decir que si una falla es detectada en un sistema no necesariamente será peligrosa en todos los sistemas UNIX.

Características

- Código fuente abierto, es decir que el código esta disponible para ser analizado por cualquier persona. En caso de que se descubra una falla esta puede servir para arreglar el código y hacerlo más robusto, o ser utilizado por *hackers* los que aprovechándose de esa vulnerabilidad desencadenen ataques a la seguridad.
- Fácil de obtener, la mayoría de versiones se distribuyen a bajo costo o de forma gratuita lo que hace posible que muchas personas lo usen y experimenten con él, conozcan sus fortalezas y debilidades.
- Información fácilmente obtenible, en el Internet se encuentra con gran facilidad información y herramientas desarrolladas para estos sistemas operativos.

HIDS (HOST BASED INTRUSION DETECTION SYSTEM)

Un HIDS detecta actividades maliciosas dentro de una sola computadora, para esto revisa los archivos de *log* del *host* y del sistema. Dependiendo de su instalación y configuración puede monitorear directamente a un proceso del sistema operativo o a un recurso crítico del sistema que requiera mayor protección. Los agentes son ubicados en el recurso a proteger, estos notifican a un sistema central de administración para su análisis y toma de alguna acción correctiva.

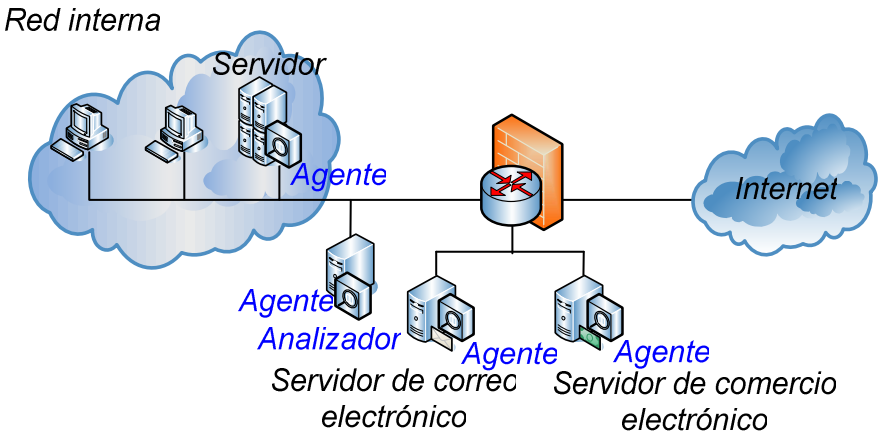


Figura 3.35: Ejemplo de HIDS

En la Figura 3.35 los agentes se ubican sobre los servidores que se quiere proteger de ataques, los agentes reportan el resultado de su monitoreo a un equipo encargado de analizar esta información.

POLÍTICAS DE SEGURIDAD

Como se dijo anteriormente es imposible lograr un sistema cien por ciento seguro, debido principalmente al alto costo que tendría lograr una seguridad total. Sin embargo la implementación de políticas de seguridad adecuadas ayudará en gran mediada a proteger los activos de información de una compañía.

“Política de Seguridad: una declaración de intenciones de alto nivel que cubre la seguridad de los sistemas informáticos y que proporciona las bases para definir y delimitar responsabilidades para las diversas actuaciones técnicas y organizativas que se requerirán…”[1]

Las políticas se manifiestan por medio de normas, reglas y protocolos definidos por los responsables del sistema, las que de forma general regulan lo que está o no permitido en el área de seguridad de la información que maneja una compañía. Sobre todo las políticas son una forma de comunicación con los usuarios, para que ellos conozcan la importancia de la información y todos los recursos que manejan.

Los detalles de la implementación de las políticas son definidos por los procedimientos.

OBJETIVOS DE LAS POLÍTICAS DE SEGURIDAD DE LA INFORMACIÓN

- **Manejo de Riesgos:** Identificar los riesgos que afronta y desarrollar medidas para minimizar el impacto de estos.
- **Asegurar la continuidad de la operación:** Lo que se debe hacer para asegurar la operación de la organización. En caso de incidentes o desastres las acciones necesarias para reanudar las operaciones.
- **Definir responsabilidades, expectativas, y comportamientos aceptables:** Los usuarios deben entender sus responsabilidades y cómo éstas pueden variar dependiendo de las circunstancias.
- **Proteger la organización de responsabilidades**: Demostrar que las acciones de un empleado fueron o no autorizadas por la compañía.
- **Asegurar la integridad de la información y confidencialidad:** Proteger los activos de la información de la compañía.

[1] RFC 1244

POLÍTICAS, ESTÁNDARES, PAUTAS, LINEAMIENTOS Y PROCEDIMIENTOS [9]

Políticas de administración: son las que direccionan a los estándares, pautas, lineamientos, y procedimientos a ser seguidos por la organización. Para una definición más específica se dividen en consultivas, reguladoras e informativas.

- **Consultivas:** No son obligatorias. Recomiendan líneas de acción o acercamientos, pero permiten independiente juicio en casos especiales.
- **Reguladoras:** Pretenden asegurar que una organización implemente procedimientos estándares y las mejores prácticas de su compañía. Se aplican a instituciones como bancos, compañías de seguros, empresas públicas, etc.
- **Informativas:** Proporcionan información y generalmente no requieren ninguna acción por parte de los individuos afectados. Informa a los usuarios de las actividades prohibidas y consecuencias de practicar esas actividades.

Estándares: Son obligatorias y se refieren usualmente a *hardware* y *software* específicos.

Pautas: son sugerencias al personal de una organización sobre cómo asegurar eficazmente sus redes y computadoras

Procedimientos: son obligatorios, pasos detallados a ser seguidos para lograr tareas específicas. Permiten implementar las políticas, los estándares y las pautas.

Lineamientos: son similares a estándares y representan un nivel de implementación de controles de seguridad. El nivel de protección de los lineamientos es obligatorio y puede ser usado para desarrollar estándares de seguridad de sistemas informáticos organizacionales.

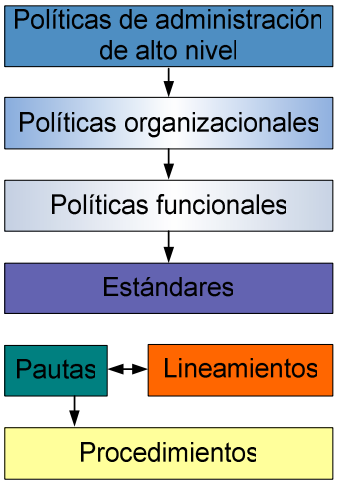


Figura 3.36: Jerarquía de políticas, estándares, lineamientos, pautas y procedimientos

DESARROLLO DE POLÍTICAS DE SEGURIDAD

Elementos a considerar

Las políticas y procedimientos de seguridad deben considerar algunos aspectos como [12]:

- Identificar los activos de la organización.
- Definir los riesgos.
- Definir cómo los activos de información van a ser manejados.
- Definir cómo los activos de información van a ser accesados y que procesos de autenticación van a ser usados.
- Definir claramente y en detalle qué es o no apropiado en el uso de medios electrónicos y servicios propios de la compañía.
- Definir qué tipo de información puede ser accesada y distribuida.
- Definir el tipo de controles.
- Notificar a usuarios de procedimientos de monitoreo y auditoria, acceso de información y consecuencias de incumplimiento.
- Identificar responsables de aplicación de políticas y procedimientos de seguridad.
- Desarrollar pasos a ser tomados frente a incumplimiento de políticas, una brecha de seguridad, o un desastre.

Proceso de creación [7]

A continuación se enumera de forma general los pasos a seguir para la creación de políticas de seguridad:

1. Identificar y priorizar activos.
2. Identificar vulnerabilidades.
3. Identificar amenazas y sus probabilidades.
4. Identificar medidas a tomarse.
5. Desarrollar un análisis costo-beneficio.
6. Desarrollar políticas de seguridad.

IMPLEMENTACIÓN DE POLÍTICAS DE SEGURIDAD [7]

Los pasos para la implementación de políticas de seguridades pueden resumir de la siguiente forma:

1. **Escribir las políticas de seguridad y un manual de procedimientos**
   Una política debe incluir los siguientes elementos:

   Declaración de políticas: Escribir la política y su objetivo de forma general.
   Propósito: Indicar porqué la política es necesaria.
   Alcance: Qué tan lejos llega la política, bajo que circunstancias es aplicable (tiempo, hardware, software, evento, etc.)
   Cumplimiento: incluye una explicación detallada de que constituye o no constituye el cumplimiento de una política.
   Penalizaciones/consecuencias: consecuencias del no cumplimiento de la política claramente definidas.

2. **Desarrollar un programa de educación y conocimiento para los usuarios finales.**
   Los usuarios deben conocer las políticas de seguridad, su responsabilidad dentro de éstas y las consecuencias del no cumplimiento de las mismas. Las

organizaciones deben crear un plan de educación sobre las políticas de seguridad para los usuarios.

3. **Desarrollar un proceso para aplicación de políticas e implementación de procedimientos.**
   Supervisar constantemente que las políticas y procedimientos se están cumpliendo. Debe existir una planificación específica para controlar su implementación y los encargados de hacerlo.

4. **Desarrollar un proceso para actualizaciones y revisiones periódicas de políticas y procedimientos.**
   Revisar periódicamente las políticas y procedimientos, o hacer actualizaciones en caso que una parte del sistema se modifique y sea necesario.

MODELO DE ASEGURAMIENTO CONTINUO

Luego de desarrollar las políticas de seguridad se debe seguir un modelo de aseguramiento continuo el que tiene cuatro pasos:

1. Asegurar.
2. Monitorear.
3. Probar.
4. Mejorar.

El modelo de aseguramiento continuo promueve reexaminar y reaplicar medidas de seguridad actualizadas continuamente.

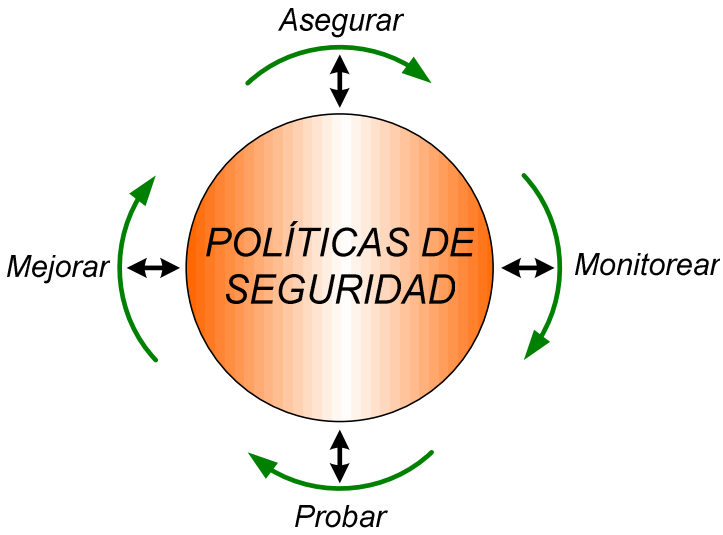


Figura 3.37: Modelo de aseguramiento continuo

“La seguridad es un proceso continuo construido alrededor de políticas de seguridad” [10]

Asegurar: aplicar políticas de seguridad e implementar dispositivos y soluciones de seguridad como: IDS, *firewalls*, VPNs, etc.

Monitorear: descubrir violaciones y ataques contra las políticas de seguridad, verificar que la implementación del paso anterior Asegurar funcione correctamente.

Probar: Comprobar la efectividad de la seguridad implementada.
Existen herramientas de escaneo de vulnerabilidades, para probar periódicamente las medidas de seguridad de la red.

Mejorar: Analizar la información del monitoreo y pruebas para realizar mejoras en la seguridad.

Este proceso es continuo no tiene un fin, todos los pasos deben repetirse permanentemente, en busca de nuevas vulnerabilidades o posibles ataques.

PRODUCTOS COMERCIALES Y SOLUCIONES DISPONIBLES EN EL MERCADO ECUATORIANO

A continuación se muestra las principales características de productos de seguridad disponibles en el mercado ecuatoriano.

AUTENTICACIÓN

CISCO[1]	
Secure ACS (Access Control Server)[2]	• Proporciona el servicio centralizado de AAA a usuarios. • Controla a *gateways* de acceso ubicados en la red. • Basado en Web, tiene un interfaz gráfico. • Soporta más de 400 autenticaciones por segundo. • Soporta cualquier tipo de acceso: inalámbrico, *firewall*, VPN, etc. • Se instala sobre Windows 2000, NT o 2003. • Utiliza TACACS+, RADIUS.

Tabla 3.5: Características de productos de autenticación

FIREWALLS

CISCO	
PIX (Private Internet Exchange)	• *Firewall* dedicado[3], *hardware* y *software* que proporciona filtraje de paquetes. • Tecnologías de servidor *Proxy.* • VPN y detección de intrusos. • Existen varios modelos: PIX 501, 506E, 515E, 525, 535.
Catalyst 6500 Series FWSM (Firewall Services Module)	• Permite integrar seguridad de *firewall* dentro de la infraestructura de red. • Se instala dentro de *switches* Cisco Catalyst Serie 6500 o Router de Internet Cisco 7600.

[1] www.cisco.com
[2] Una solución de hardware de Cisco es Cisco ACS Solution Engine
[3] Dispositivos dedicados completamente a realizar funciones de firewall

3COM[1]	
SuperStack 3	• *Firewall* dedicado. • Inspección de paquetes *stateful*. • VPN basadas en IPSec. • Interfaz gráfico, basado en Web • Ayuda en línea. • Actualizaciones de *software* automáticas y gratuitas. • Certificado ICSA[2].
Netscreen[3]	
Serie NetScreen 200	• Incluye dos productos el NS-204 y en NS-208. • Alta disponibilidad. • Actúa como *firewall* y Túneles de VPN en cada interfaz. • 200 Mbps en VPN, 128.000 sesiones concurrentes, 1.000 túneles VPN, 4.000 políticas.
NetScreen 500	• Alta disponibilidad. • 25 sistemas virtuales. • 700 Mbps f*irewall*, 250 Mbps VPN, 250.000 sesiones concurrentes, 10.000 túneles VPN, 20.000 políticas • Gestión del tráfico. • Filtro de contenido.
NetScreen 50 & NetScreen 25	• NetScreen 25: 100 Mbps en *firewall*, 20 Mbps en VPN, 4.000 sesiones concurrentes, 25 túneles VPN, 500 políticas. • NetScreen 50: 170 Mbps en *firewall*, 50 Mbps en VPN, 8.000 sesiones concurrentes, 100 túneles VPN, 1.000 políticas.
NetScreen 5XT	• Administración controlada por menú del sitio central usando NetScreen-Global PRO o NetScreen-Globales PRO Express. • Administración vía Web con el WebUI (Interfaz de usuario Web) incorporado. • CLI (Interfaz de línea de comandos) accesible por SSH y Telnet • Alertas por *Email*, trampas y alarmas de SNMP. • 70 Mbps de velocidad en *firewall*, 2.000 sesiones concurrentes, 10 túneles VPN, 100 políticas de seguridad.

[1] www.3com.com
[2] Certificación de Laboratorios ICSA (pruebas de un conjunto de estándares de seguridad y funcionamiento)
[3] www.netscreen.com

Microsoft[1]	
ISA (Internet Security and Acceleration) Server 2004	• Firewall basado en servidor[2], mejora la seguridad y el desempeño de la red. • Proporciona una protección avanzada, de fácil uso, y rápida, acceso seguro a todos los tipos de redes. • Servidor de seguridad de Internet: Presenta características de *proxy*, NAT, VPN y *firewall* en un solo paquete, esto hace que sea más fácil de monitorear y administrar que otros dispositivos de *hardware* y/o programas de *software* separados. • Acelerador de Internet: permite un acceso más rápido a sitios Web visitados frecuentemente y reduce el tráfico de Internet.
Check Point [3]	
Check Point Firewall-1	• *Firewall* basado en servidor. • *B*rinda protección perimetral para asegurar los recursos de la red. • Integra protección de capa aplicación y red. • Se instala sobre un sistema operativo servidor como Windows NT/2000/2003, UNIX. • *Stateful inspection.* • Filtro de contenido. • Detección de intrusos. • NAT.
Sygate[4]	
Sygate Personal Firewall	• *Firewall* personal[5] que proporciona características avanzadas. • Sistema de defensa de intrusos bidirecional. • Hace invisible al PC protegido del mundo exterior. • Altamente configurable, aplicaciones basadas en políticas, direcciones IP confiables, puertos y protocolos pueden ser personalizados para soportar y asegurar cualquier requerimiento o configuración de red. • Detección de intrusos basada en *host* y en aplicación en tiempo real. • Se instala sobre Windows 98/ME/NT/2000/XP.

[1] www.microsoft.com
[2] Se ejecutan sobre sistemas operativos de red (UNIX, NT, 2003, o Novell)
[3] www.checkpoint.com
[4] www.sygate.com
[5] Se instalan sobre computadoras de usuarios que no forman parte de una LAN y acceden al Internet por medio de conexiones *dial-up*, DSL o cable módem

McAfee[1]	
McAfee Personal Firewall Plus 6.0.6014	• Efectiva barrera de ataques provenientes de Internet. • Fácil de instalar. • Interfaz de usuario atractivo y sencillo. • Protege las comunicaciones en red, los datos personales y el equipo completo de forma continua. • Controla los accesos a Internet de todas las aplicaciones y los intentos de acceso al PC. • Notifica en todo momento de sus actividades. • Configurable, puede adaptarse a las necesidades de cada usuario. • Se instala sobre Win98/ME/2000/XP.
Symantec[2]	
Norton Personal Firewall 2006	• Protección automática contra *hackers.* • Controla el tráfico que entre y sale de Internet. • Protege la información saliente. • Se instala sobre Windows® XP, Windows 2000.
Zone Labs[3]	
Zone Labs ZoneAlarmInternet Security Suite	• Es una herramienta completa y fácil de usar. • Impide el acceso de piratas informáticos. • Hace al PC invisible para cualquier usuario de Internet. • Protección contra software espía. • Funcionalidad de antivirus. • Seguridad del correo electrónico. • Filtrado de contenidos Web. • Protección contra p*hishing*. • Protección contra el correo no solicitado.
Microsoft	
Microsoft Internet Conection Firewall	• *Firewall* personal que viene incluido en Windows XP, cuando se instale el *service pack* 2 éste se habilitará automáticamente. • Permite restringir que información se comunica entre el Internet y el hogar o pequeña oficina. • También protege a una sola computadora conectada a Internet con cable módem, DSL o *dial-up*.

[1] www.mcafee.com
[2] www.symantec.com
[3] www.zonelabs.com

Armor2net Software[1]	
Armor2net Personal Firewall 3.12	• Proteger al sistema de posibles ataques exteriores y visitas no deseadas, como es el caso de gusanos, troyanos o *spyware*. • Examina cada paquete que llega a al PC, examina todas las conexiones activas y permite cerrar aquellas peligrosas. • Permite establecer qué programas tienen acceso a la conexión a Internet y cuáles no. • Elimina los *spyware* instalados en el sistema, bloquea las ventanas de *pop-up* con diversas opciones de filtrado y bloquea ciertos sitios Web.

Tabla 3.6: Características de productos firewall

ANTIVIRUS

BitDefender[2]	
BitDefender 9 Professional Plus	• Protege de virus, troyanos, gusanos y otros elementos dañinos. • Analiza archivos adjuntos a correos electrónicos o a páginas Web • Añade un *firewall* y además bloquea el acceso a páginas cuyo contenido resulte no deseado. • Fácil de usar, con una interfaz intuitiva, que se actualiza automáticamente vía Internet.
BitDefender 9 Standard	• Ofrece una protección completa para el PC. A diferencia de la versión Profesional, no incluye *firewall* ni control *antidialers*.
Trend Micro [3]	
Trend PC-cillin Internet Security 12.3 build 1028	• Protección automática, busca y elimina de virus mientras se trabaja. • Actualizaciones automáticas para estar al día con las nuevas amenazas de virus. • Brinda una de las mejores protecciones posibles para un PC.
GRISOFT[4]	
AVG Anti-Virus System Free Edition	• El manejo del sistema es ágil y sencillo usar. • Monitorización constante del sistema. • Escanea el correo electrónico.

[1] www.armor2net.com
[2] www.bitdefender.com
[3] www.trendmicro.com
[4] www.griftsoft.com

	• Análisis preprogramados y bajo demanda. • Actualización gratuita de la base de datos de virus. • Desinfección automática de archivos infectados. • Sistema para manejar de forma segura ficheros infectados.
AVG Professional Single Edition 7.0	• Práctico y fácil de usar. • Analizar el sistema completo o seleccionar los directorios que se desea examinar. • Permite programar escaneados automáticos del sistema. • Cuenta con un módulo que vigila el movimiento de archivos dentro del PC y desinfecta archivos infectados de forma automática. • Presenta una zona de cuarentena donde guarda archivos infectados de forma segura y sin peligro para el resto del sistema.
McAfee	
McAfee VirusScan 10.0	• Protección total frente a virus que se pueden contener en archivos, subdirectorios, etc. • Evita posibles infecciones a través de correo electrónico, de descarga de archivos de Internet, de ataques maliciosos a partir de *applets* de java y controles ActiveX, de elementos espía y de programas potencialmente no deseados. • Además trae un potente filtro para Internet que permite bloquear el acceso a sitios Web no deseados. • Protección permanente de forma automática.
F-Secure[1]	
F-Secure Anti-Virus 2005	• Contiene dos de los motores de búsquedas de virus más conocidos para Windows: F-PROT y AVP. • Ofrece funciones completas de red y se actualiza de forma diaria. Proteger la información contra virus, incluyendo los nuevos virus. • Buscador basado en reglas para detectar virus desconocidos y muchas opciones para automatizar la detección de virus.
Symantec	
Norton Internet Security™ 2006	• Una solución integral que protege al PC de virus, *hackers*, *spyware* y *spam*. • Es un conjunto de herramientas que ayudan a mantener la seguridad completa del computador.

[1] www.f-secure.com

	• Las herramientas que conforman esta solución son: Norton AntiVirus, Norton Personal Firewall, Norton Privacy Control, Norton AntiSpam, Norton Parental Control.
Norton Antivirus 2006	• Automáticamente bloquea y remueve virus, caballos de Troya, y gusanos. • Chequea los virus de archivos adjuntos de correos entrantes y salientes. • Automáticamente realiza actualizaciones que protegen contra nuevos ataques. • Protege contra *spyware* y *adware*[1]. • Se instala sobre Windows® XP y Windows 2000.
CA[2]	
eTrust EZ Antivirus 2005 7.0.5.3	• Detecta y elimina todo tipo de virus, incluyendo troyanos y gusanos. • Rápido, eficaz y fácil de usar. • Actualizaciones de software automáticas. • Consola de administración basada en Web. • Capacidades de reportes extendidas.
Fortinet	
FortiGate Antivirus Firewall	• Plataformas que combinan *hardware* y *software* para ofrecer antivirus, filtrado de contenidos Web y de *email*. • *Firewall* de inspección detallada, IPSec VPN, detección y prevención de intrusiones y funciones de perfilado de tráfico. • Detecta y elimina las amenazas que provienen de los contenidos *email* y del tráfico Web en tiempo real, sin reducir el rendimiento de la red. • Elimina las limitaciones impuestas por el uso de múltiples dispositivos, ya que brinda servicios a nivel de aplicación y red en un mismo dispositivo.
Panda[3]	
Panda Titanium Antivirus + Antispyware 2006 5.00	• Protección contra todo tipo de virus, gusanos y troyanos. • Anti-*spyware.* • *Anti-pishing.* • Actualización automática.

[1] Paquetes de avisos publicitarios
[2] www.ca.com
[3] www.pandasoftware.com

Panda Platinium 2006 Internet Security	• Anti-*spam.* • Control de privacidad. • Filtro de contenido Web.

Tabla 3.7: Características de productos antivirus

VPN

Pueden crearse usando productos como *firewalls*, *routers*, concentradores de VPNs, *hardware* de clientes y *software* de VPN, IDS y *software* de administración.

Cisco	
Cisco Concentradores de VPN series 3000	• Esta serie de productos presentan plataforma para VPNs de acceso remoto y clientes de *software*, los cuales soportan varios clientes. • El número de clientes puede variar de 100 hasta 10000 usuarios simultáneos. • Los modelos de esta serie son: Concentrador 3005, 3015, 3030, 3060, 3080.
Cisco VPN cliente	• Aplicación para Windows 95, 98, 2000 y NT. • Confidencialidad (cifrado). • Integridad y Autenticidad (*Hash*). • Autenticación de identidad (firmas digitales y certificados X.509). • Soporta los estándares: IPSec; DES, 3DES, MD-5, SHA-1; IKE con ISAKMP. • Servicios en modo túnel y transporte.
Netscreen	
NetScreen Remote - Cliente VPN	• Solución de VPN para clientes remotos que necesitan tener acceso a la red privada a través de una red pública así como los usuarios finales dentro de un ambiente de la empresa que requieren una conexión segura. • Actualizaciones automatizadas de las políticas. • Compatibilidad extensa.
Check Point	
Check Point VPN-1/FireWall-1 Solution	• Módulo opcional que protege la comunicación sobre el Internet y habilita a una empresa la construcción de VPN fáciles de mantener usando segmentos de redes públicos y privados.

	• Permite comunicaciones cifradas y garantiza la privacidad de datos, integridad y autenticidad. • Soporta algoritmos y protocolos estándar en la industria, como DES, 3DES y IPSec/IKE. • Para organizaciones con despliegue de PKI incluye soporte de certificados digitales.

Tabla 3.8: Características de productos VPN

SISTEMAS DE DETECCIÓN Y PREVENCIÓN DE INTRUSOS

Los dispositivos de detección de intrusos incluyen sensores que pueden instalarse sobre la red, *switch*, *router*, *firewalls*, *host* y sistemas de administración.

Cisco	
Cisco Sensores de aplicaciones de Red	• Los sensores detectan actividad maliciosa o no autorizada en la red. En Cisco se presentan varios modelos: 4215, 4235, 4250, 4250-XL.
SonicWALL[1]	
SonicWALL IPS (Intrusion Prevention Service)	• Tecnología que aumenta las capacidades de prevención de los *firewalls* tradicionales. • Automatización en la actualización de firmas distribuida, disminuye el costo total. • Las políticas globales y de grupo de ataques por prioridad, simplifican el desarrollo y gestión. • La granularidad en la política de inspección, hace que las detecciones sean más exactas, disminuyéndose los falsos positivos. • Solución de alto rendimiento y económica para segmentos de red pequeños/medianos/grandes. • Gestión centralizada y configurable para desarrollo rápido.
3Com	
Tipping Poin Unity One	• Permite ver todo el tráfico de red y rechazar todo tipo de ataques internos y externos. • Puede gestionarse centralmente.

[1] www.sonicwall.com

	• Alta disponibilidad, gran resistencia a condiciones adversas de la red. • Su hardware de propósito único garantiza alto rendimiento y baja latencia en cualquier condición. • Precisión en el reconocimiento de ataques con la combinación de tres métodos de filtrado: vulnerabilidad, anomalías de tráfico y firma. • Fácil de gestionar, asegura el 100% de accesibilidad se combina la gestión centralizada y local.
Snort[1]	
Snort IDS	• Basado en red (NIDS). • Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida. • Todo esto en tiempo real. • Gratuito y funciona bajo plataformas Windows y UNIX/Linux. • Dispone de una gran cantidad de filtros o patrones ya predefinidos. • Actualizaciones constantes a través de los distintos boletines de seguridad. • Implementa un lenguaje de creación de reglas flexible, potente y sencillo. • Puede funcionar como *sniffer*, registro de paquetes o como un IDS normal.
ISS[2]	
ISS Real Secure	• Propone soluciones activas en tiempo real a los múltiples ataques de la red, incluye el cierre de sesión y reconfiguración del *firewall*. • Analiza fuentes y destinos, servicios, tipos de ataques, direcciones IP duplicadas, etc. • Comprueba la eficiencia del *firewall*. • Genera reportes sobre la actividad en la red. • Tiene topologías de red numerosas: Ethernet, Fast Ethernet, Token Ring y FDDI. • Base de datos *on-line* de asistencia. • Plataformas Windows NT y UNIX.

Tabla 3.9: Características de productos de detección y prevención de intrusos

[1] www.snort.org
[2] www.iss.net

CONTROL DE CONTENIDO

SonicWall	
SonicWall (Content Filter Service)	• Su arquitectura de filtrado de contenido utiliza una base de datos ampliable y dinámica para bloquear contenido Web cuestionable e improductivo. • Evita pérdidas de productividad, controlando el acceso a sitios de contenido inadecuado, ilegal o peligroso. • Permite personalizar las características de filtrado de acuerdo a grupos de usuarios de la red, fecha y hora. • El proceso de bloqueo o autorización, sólo tarda una fracción de segundo. • Clasifica y filtra millones de direcciones URL, IP y sitios Web.
Websense[1]	
Websense Enterprise	• Establece políticas flexibles de uso de Internet. • Establece políticas para tipos de archivos y más de 50 protocolos de aplicaciones, según usuarios o grupos definidos. • Usa herramientas de informes poderosos, para analizar y hacer un seguimiento del uso de Internet a través de su organización. • Escala de 50 a 250,000+ usuarios con total confiabilidad y precisión.
Websense Web Security Suite™	• Proporciona una solución de Internet integrada para proteger a las organizaciones de las amenazas basadas en la Web tanto internas como externas.
Websense Web Security Suite - Lockdown Edition™	• Incluye todos los beneficios de Web Security Suite y extiende la identificación de datos y la experiencia de categorización de Websense al escritorio, aprovechando la tecnología probada de filtro Web y de seguridad Web para administrar y ocuparse de las amenazas de seguridad de escritorio.
Aladdin[2]	
Aladdin eSafe	• Anti-*spyware.* • Anti-*spam.* • Filtro de aplicaciones. • Tecnologías de antivirus preactivas.

[1] www.websense.com
[2] www.aladdin.com

MIMEsweeper[1]	
MIMEsweeper Email Managed Service	• Brinda total seguridad de *email*. • Permite personalizar las reglas para mejorar las políticas. • Cifrado punto a punto entre servidores asegurando confidencialidad. • Reporte gráfico y seguimiento de mensajes avanzado. • Defensa integral contra *spam*, virus, *spyware*, *phishing* y DoS.
MIMEsweeper for SMTP 5.2	• Inspecciona el contenido de *email* para detectar todo tipo de ataques. • Tecnología anti-*spam* y anti-virus. • Administración de políticas granular, con plantillas de políticas y *wizards* para hacer un diseño, desarrollo e implementación más fácil. • Configura y administra todos los *gateways* de *email* desde una consola de administración central. • Escalable soporta hasta 100000 usuarios.

Tabla 3.10: Características de productos de control de contenido

ADMINISTRACIÓN

Los sistemas de administración controlan el acceso a los recursos de la red, de acuerdo a las pautas establecidas en la compañía. Algunos dispositivos de red como *routers*, *firewalls*, dispositivos VPN y sensores IDS forman parte del sistema de administración.

Cisco	
CiscoWorks VPN /Security Management Solutions	• Aplicaciones basadas en Web para configurar, monitorear y localizar fallas en *routers* VPNs, *firewalls*, NIDS y HIDS.
IBM[2]	
Tivoli	• El *software* de gestión de sistemas se utiliza para supervisar, controlar y optimizar los recursos informáticos. • Algunas de las opciones de herramientas de administración son las siguientes:

[1] www.mimesweeper.com
[2] www.ibm.com

	Tivoli Configuration Manager: Una solución integrada para supervisar las configuraciones de *hardware* y *software*. **Tivoli Monitoring:** Automatiza la supervisión de los recursos esenciales del sistema. **Tivoli Monitoring for Databases:** Incremente la disponibilidad y el rendimiento de DB2, Oracle, Microsoft® SQL Server e Informix. **Tivoli OMEGAMON XE for Databases:** Una solución global de disponibilidad y rendimiento diseñada para gestionar y ajustar de forma proactiva el entorno de la base de datos de la empresa. **Tivoli Provisioning Manager:** Reduce los costos y mejora la utilización del servidor. **Tivoli Remote Control:** Proporciona control remoto completo y en tiempo real para toda la empresa. **Tivoli System Automation for Multiplatforms:** Aumenta la eficacia de las operaciones del sistema y mejora la disponibilidad de aplicaciones para sistemas basados en AIX y Linux.
Netscreen	
NetScreen Global PRO Security Management	• Administración flexible. • Provee de un detallado reporte de eventos y violaciones de políticas de seguridad. • Vistas de monitoreo adecuadas a las necesidades de los proveedores de servicios y sus clientes. • Arquitectura altamente escalable.
3COM	
3COM Network Supervisor Version 4.0	• Descubrimiento automático de *switches* 3COM, sistemas NBX y estaciones. • Registro inteligente de eventos y alertas. • 1500 nodos. • Envío de alarmas vía e-mail, o mensajes SMS. • Reporte configurable de usuarios. • Soporte de capa 3 y de QoS mejorado.
3COM Network Supervisor Advanced Package V2.0	• Compatibilidad completa con 3COM Network Supervisor. • Administración de VLAN. • Programación de descubrimientos, configuración, respaldos, tareas de restauración. • Actualiza a 3COM Network Supervisor a 2500 nodos.

Tabla 3.11: Características de productos de administración

CAPÍTULO 4

IMPLEMENTACIÓN DEL CURSO DE ENTRENAMIENTO Y PRUEBAS DE OPERACIÓN

PLANIFICACIÓN DE DESARROLLO DEL CURSO

Desarrollar una aplicación multimedia que trabaje del modo en que se planificó y que entregue el contenido de forma exitosa, requiere de la unión de varias tareas en diferentes fases o etapas de un proceso complejo. A éste proceso se le denomina modelo o metodología, el que además de conseguir que el resultado final sea lo más fiable y eficiente posible, dota de un lenguaje común a los miembros del equipo de desarrollo.

A continuación se describen de forma resumida algunas de las metodologías que se consideran más prácticas, las cuales se adaptan al diseño de cursos basados en computador y en Web.

METODOLOGÍA PARA DISEÑO HIPERMEDIAL DE MATERIALES EDUCATIVOS (MEDHIME) [22]

Es una metodología simple e intuitiva para personas con escasos conocimientos informáticos, ofrece etapas que contribuyen a un proceso de análisis y diseño completo, permite que la implementación del material multimedia sea más rápida y sencilla. Está orientada a facilitar el desarrollo de contenido multimedia a docentes con la ayuda de programadores y de diseñadores gráficos.

Se basa en cuatro etapas básicas:

Análisis del Dominio: Se define el qué, para qué, para quiénes y cómo navegar por el material. Es fundamental definir los siguientes aspectos:

- Objetivos: Aquí es donde se define el para qué de la aplicación. Un objetivo bien definido requiere saber lo que se desea que suceda cuando la aplicación se utiliza.
- Público: En este nivel se debe definir el para quiénes. El público está relacionado con los objetivos, se debería tener en cuenta características tales como: edades, gustos, intereses, inclinaciones, etc.
- Contenidos: Implica definir el qué. Esta información deberá ser significativa para el público definido anteriormente y se deberá tener en cuenta lo que a éste le interesa y no lo que le interesa a los desarrolladores.
- Estructura: Involucra la organización de la información, o sea, el cómo. Existen varias maneras de organizar la información dentro de la aplicación: jerárquica, lineal, combinada, etc.

Diseño Conceptual. Provee la organización de los temas, definiendo el orden y relación entre ellos. En ésta etapa se crea una tabla de doble entrada cuyos atributos son:

- Tema: Nombre del módulo o denominación del tema.
- Id-padre: Número que corresponde al tema del cual desciende.
- Id-tema: Número que identifica al tema.
- Dirección: Ruta donde se encuentra almacenada la información.

Diseño Navegacional: Por medio de un sencillo diagrama, se indican los vínculos y relaciones de cada tema, las páginas emergentes y los tipos de menú.

Diseño Comunicacional: En una tabla, a través de gráficos sencillos se explica cómo se pretenden ver las páginas con sus gráficos, textos y animaciones.

Luego de finalizar estas cuatro etapas, las cuales son realizadas generalmente por el docente, los programadores y diseñadores gráficos disponen de la información necesaria para el proceso de implementación de la aplicación multimedia.

PROCESO DE DISEÑO INSTRUCCIONAL MULTIMEDIA [16]

El Proceso de Diseño Instruccional Multimedia está dirigido especialmente a compañías que buscan desarrollar cursos multimedia para capacitación interna. Es útil tanto para diseñadores instruccionales, autores o gerentes de proyecto, que inicien en el desarrollo multimedia, así como también para diseñadores expertos que requieren un proceso sistemático. El objetivo de este proceso es proporcionar tareas específicas que ayuden a reducir el ciclo del tiempo para completar un proyecto de la forma más adecuada.

La Figura 4.1 muestra un esquema de las fases del Proceso de Diseño Instruccional:

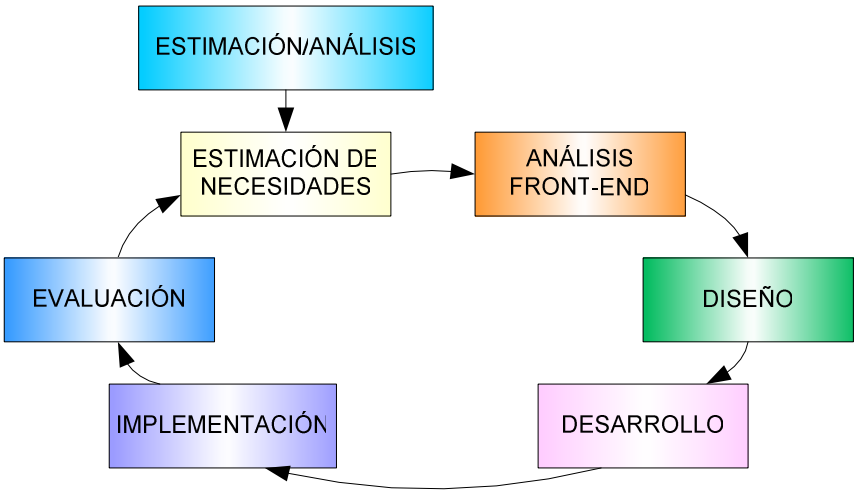


Figura 4.1: Proceso de Diseño Instruccional Multimedia

Estimación de necesidades: Analiza el estado actual y determina el estado deseado, estableciendo prioridades para la acción.

Análisis frond-end: Determina el camino a seguir para llegar al estado deseado. Aquí se detalla la información sobre lo que se va a desarrollar.

- Análisis de audiencia: Identifica quién es el posible público para la solución y sus necesidades de aprendizaje.

- Análisis de tecnología: Identifica la capacidad de la tecnología disponible.
- Análisis situacional: Identifica las variables ambientales u organizacionales que pueden tener un impacto en los objetivos y el diseño multimedia.
- Análisis de tareas: Divide el trabajo en deberes y tareas, así como también determina el conocimiento, habilidades y actitudes que los miembros del equipo de trabajo deben tener.
- Análisis de incidentes críticos: Analiza la lista de tareas y determina las más críticas para un buen desempeño.
- Análisis de objetivos: Escribe objetivos para las tareas de trabajo seleccionadas.
- Análisis de medios: Selección del medio de distribución más adecuado para el curso.
- Análisis de datos existentes: Identifica materiales de entrenamiento disponibles y cuáles necesitan ser desarrollados; básicamente, decidir entre construir o comprar.
- Análisis de costos: Identifica costo y beneficio, y retorno de la inversión.

Diseño: Fase de planeación, donde se crea un documento con las especificaciones para el diseño del curso.

- Horario: Describe el proyecto, con fechas de entrega para cada tarea.
- Equipo de trabajo: Enlista roles y responsabilidades de los miembros del equipo de trabajo.
- Especificaciones de media: Tipos de documentos, estilos de presentación general, texto, gramática, gráficos, fuentes, temas, símbolos de edición, etc.
- Estructura de lección: Describe cómo el contenido es agrupado, ordenado, enlazado o navegado.

Desarrollo e Implementación: Se crea un documento de especificaciones de diseño, en este punto la metodología diverge dependiendo del medio de distribución: basado en computador o en Web. Se escriben los bocetos, se crean, editan y almacenan los videos, sonidos y gráficos; las versiones iniciales del curso son desarrolladas, probadas y revisadas.

Evaluación: Probar la funcionalidad y desempeño, propone dos tipos de evaluaciones:

- Estratégica: Desarrollar y evaluar estrategias para medir la reacción, conocimiento, desempeño y costo del proyecto.
- Táctica: Pruebas de diseño, desarrollo, presentación, validación y confiabilidad.

METODOLOGÍA PARA LA CONSTRUCCIÓN DE CURSOS MULTIMEDIA DE PERURENA [18]

Esta metodología está basada en el modelo de Sistema de Diseño Instruccional estándar (IDS), el que constituye una buena aproximación para desarrollar entrenamiento basado en la tecnología.

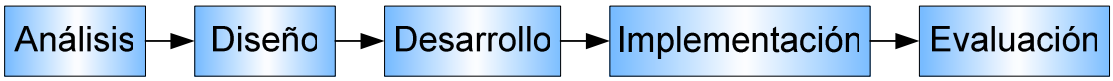


Figura 4.2: Modelo ISD estándar [19]

La metodología está orientada al desarrollo de aplicaciones multimedia para la enseñanza universitaria. Las fases o etapas de desarrollo generales y tareas incluidas en cada una de ellas son:

Planificación del proyecto: Esta fase es fundamental para que el desarrollo de un proyecto multimedia llegue a su fin. Busca definir el proyecto y establecer las prioridades de acción para su realización.

Desarrollo del contenido: En esta etapa hay que adaptar o ajustar el contenido del curso y definir la experiencia de aprendizaje dirigida a cumplir los objetivos. Una vez definido el perfil del usuario se puede seleccionar la mejor manera para entregar esa experiencia.

Diseño de la disposición, arreglo o distribución de los elementos ("*layout*"): En esta fase se debe diseñar una distribución, arreglo u ordenamiento apropiado para el contenido del curso, teniendo en cuenta los elementos fundamentales de un buen diseño, el uso efectivo de metáforas, la definición de estándares para el diseño, paradigmas de navegación, elementos multimedia, confección de plantillas, etc.

Construcción o creación de la aplicación de aprendizaje: En esta fase o etapa del proceso de desarrollo, se crea la aplicación de aprendizaje seleccionando las tecnologías apropiadas y creando los medios necesarios.

Evaluación de la aplicación: Después de crear la aplicación, es necesario evaluarla completamente para asegurar que trabaja y soporta los objetivos de aprendizaje, este proceso se conoce como "*testing*" de la aplicación. Para cerciorarse que la aplicación trabaje hay que revisar el contenido y la funcionalidad.

METODOLOGÍA PARA EL DISEÑO DEL CURSO

En base al análisis de las anteriores metodologías, se observa que todas tienen una estructura común, cubren principalmente las etapas de un proyecto en general: análisis, diseño, desarrollo, implementación y evaluación.

Cabe resaltar que dependiendo del alcance del proyecto, las fases de desarrollo y tareas a ejecutar para diseñar y producir una aplicación multimedia de entrenamiento y/o aprendizaje pueden variar, de forma independiente a la metodología a utilizar.

Para el diseño del curso se seleccionó la Metodología para la Construcción de Cursos Multimedia de Perurena debido a que se orienta al desarrollo de aplicaciones multimedia en el campo de enseñanza universitaria, a continuación

se describe de forma más amplia las tareas a realizarse dentro de cada fase de la metodología.

PLANIFICACIÓN DEL PROYECTO

- **Evaluación de los recursos e identificación de necesidades:** Analizar la situación actual, parte de una necesidad para definir los objetivos del proyecto y los recursos con los que se contará para su desarrollo.
- **Desarrollar un perfil del usuario:** Determinar a quién va a dirigida la solución y sus necesidades de aprendizaje.
- **Consideración de las opciones de distribución:** Analizar los tipos de tecnologías disponibles, sus ventajas y limitaciones.
- **Creación de un plan del proyecto:** Definir tareas específicas a realizar en un tiempo aproximado.

DESARROLLO DEL CONTENIDO

- **Dividir los contenidos en unidades:** Categorizar el contenido dentro de unidades lógicas de aprendizaje.
- **Crear esquemas o diseños de unidad ("*outline*"):** Desarrollar un esquema de qué contenido se presentará en cada módulo o unidad de aprendizaje para modelar y ajustar el contenido del curso. Las unidades de aprendizaje no deben ser largas. Este diseño debe ser tan detallado como sea posible, incluye tópicos y subtópicos, *tests*, sesiones de práctica, exámenes, resúmenes, etc.
- **Elegir una aproximación instruccional:** Analizar y seleccionar las posibles experiencias de aprendizaje orientadas a cumplir un objetivo de aprendizaje específico. Para cada objetivo de aprendizaje, es bueno identificar una experiencia de aprendizaje.

DISEÑO DE LA DISPOSICIÓN, ARREGLO O DISTRIBUCIÓN DE LOS ELEMENTOS ("LAYOUT")

- **Creación de la estructura de navegación:** Crear un mapa con la estructura de navegación que tendrá la aplicación, especificar conexiones y enlaces entre las distintas áreas de su contenido. Es el funcionamiento lógico de la interfaz de usuario.
- **Diseño de la interfaz de usuario:** Determinar la forma de presentar e intercambiar información con el usuario. “La calidad de la interfaz (incluidas herramientas y estructura interna de presentación de la información) son factores clave en sistemas hipermediales.” [17]

CONSTRUCCIÓN O CREACIÓN DE LA APLICACIÓN DE APRENDIZAJE

- **Selección de tecnología:** Escoger la tecnología de desarrollo y analizar los requerimientos para su funcionamiento.
- **Estructura de la aplicación:** Crear la estructura de directorios de la aplicación.
- **Incorporación de la interfaz diseñada:** Adaptar la interfaz diseñada al *software* de autor, crear plantillas de páginas y funcionalidades de navegación comunes.
- **Creación de medios digitales e implementación de su funcionalidad:** Definir formatos de los medios, crearlos y programar su funcionamiento.

EVALUACIÓN DE LA APLICACIÓN [17]

Engloba tres aspectos principales:

- **Comprobación de contenidos:** Comprobación interna, hecha por el equipo de desarrollo, en la cual se comprueba que los textos están libres de errores ortográficos, que los dibujos e imágenes tienen la calidad deseada, que el sistema y esquema de navegación son correctos y que las acciones de los elementos son las diseñadas.

- **Comprobación de compatibilidad:** Se verifica el funcionamiento de la aplicación en diversas plataformas.
- **Comprobación de uso eficaz:** Consiste en entregar un primer prototipo a un grupo de usuarios reales para comprobar si éstos interaccionan correctamente con la aplicación, y si son capaces de percibir claramente todo lo que quiere comunicar.

ESTRUCTURA DEL CONTENIDO

El contenido debe ser agrupado lógicamente y estructurado consistentemente. En conjunto con la interfaz de usuario, esto da a los estudiantes un ambiente efectivo de aprendizaje donde ellos pueden acceder a la información o aprender confortablemente. [21]

La estructura de contenido que se sigue en el desarrollo del curso es la expuesta en el Capítulo 3. Otros detalles de la estructura del contenido se detallan en el punto 4.4 2.

IMPLEMENTACIÓN

Siguiendo las fases de la metodología seleccionada, se procede a la implementación.

PLANIFICACIÓN DEL PROYECTO

Evaluación de los recursos e identificación de necesidades

Situación actual

Actualmente la difusión de medios digitales y la denominada sociedad de la información en la que nos desarrollamos, requiere cada vez mayor cantidad de información soportada en diferentes formatos: texto, imágenes, sonido y video animado; lo que ha dado lugar a una nueva tecnología basada en aplicaciones multimedia.

Por otro lado, medios de comunicación como Internet se usan, en el envío de un simple saludo o en la realización de grandes negocios y transacciones, donde se pone en riesgo el activo más valioso de una persona o empresa que es la información. Esto ha dado mayor relevancia al tema de redes de información y sobre todo al manejo de dicha información de una forma segura.

Dados estos aspectos se ha visto la necesidad de realizar un curso multimedia donde se expongan de manera clara y resumida los aspectos fundamentales de la seguridad en redes.

Objetivos

Objetivo Principal:

Mostrar los conceptos básicos de Seguridad en Redes con el uso de multimedia, para lograr un curso interactivo y atractivo que facilite al usuario el aprendizaje.

Objetivos Específicos:

- Introducir al usuario en conceptos iniciales de seguridad en redes.
- Exponer brevemente los protocolos de seguridad según las capas del modelo TCP/IP.
- Enseñar los aspectos básicos sobre la creación de políticas de seguridad en redes.
- Presentar los productos y soluciones de seguridad disponibles en el mercado ecuatoriano.
- Evaluar el nivel de aprendizaje adquirido con el curso.

- Utilizar en conjunto texto, imágenes, animaciones, audio, y video para tener un mayor impacto en el estudiante.
- Lograr un formato de presentación práctico y de fácil uso.

Desarrollar un perfil del usuario

El curso está orientado a estudiantes con conocimientos previos en redes que deseen iniciarse en el campo de la seguridad.

El desarrollo de la aplicación servirá como material complementario al curso de Seguridad en Redes que forma parte de la malla curricular de la carrera de Ingeniería Electrónica y Redes de Información de la Escuela Politécnica Nacional.

Consideración de las opciones de distribución

Las posibles alternativas de distribución de acuerdo al objetivo del proyecto son: basado en computador y basado en Web.

Basado en computador

La tabla 4.1 muestra las ventajas y limitaciones en el uso de este medio.

Basado en computador	
Ventajas	**Limitaciones**
• Entrega consistente. • Los horarios se adaptan al estudiante. • Capacidad de aprovechar los múltiples sentidos del estudiante. • Periodo de aprendizaje controlado. • Alto grado de interactividad y participación del estudiante. • Se adapta al desempeño del estudiante.	• Puede ser también basada en texto. • Costoso debido al número de miembros requeridos para el equipo de diseño, hardware y software de plataforma de autor, y el costo añadido a los medios. • Largo tiempo de diseño y desarrollo, con un rango de 250 a 750 horas de diseño y desarrollo por hora de instrucción entregada, dependiendo

• Evaluación consistente y mantenimiento de registros. • Oportunidad ilimitada para revisar el contenido. • Aprendizaje reforzado a través de múltiples presentaciones y ejemplos del mismo concepto.	de la complejidad del contenido. Para un primer proyecto se tiene que programar aún más tiempo. • Se requiere habilidades especializadas, de programadores y autores. • Actualización específica de contenido y contexto puede ser limitada. • Un diseño de interfaz pobre dificulta la navegación (por ejemplo un menú con demasiadas capas) o interfiere con el aprendizaje.

Tabla 4.1: Ventajas y limitaciones de medio [16]

Las tareas para el desarrollo del curso, requieren del trabajo coordinado de un equipo, según el miembro del equipo de trabajo sus funciones son: [17]

Gerente de proyectos: Se responsabiliza del diseño y administración del proyecto.

Diseño:

- Proponer una concepción del producto.
- Evaluar con el equipo de diseño, la funcionalidad.
- Definir especificaciones funcionales y/o adaptarlas según se requieran a través del desarrollo.

Administración:

- Programación y asignación de tareas.
- Dirección de reuniones.
- Administrar la consecución de metas.
- Supervisión de los aspectos de la producción (inicio-fin).

Diseñador multimedia:

- Preparar la descripción escrita del proyecto global: contenido, medios e interacción.
- Crear una estructura para el contenido.

- Determinar los elementos de diseño requeridos para apoyar esa estructura.
- Decidir que medios son apropiados para presentar las diferentes partes del contenido.
- Debe ser capaz de analizar el contenido estructuralmente y complementarlo con métodos efectivos de presentación.
- Debe ser experto en diferentes tipos de medios.

Diseñador de interfaces: Crear un *software* que organice el contenido de multimedia y que también permita al usuario acceder o modificar el contenido en que lo presente en pantalla. El diseño de la interfaz se divide en tres áreas:

- Diseño de información.
- Diseño interactivo.
- Diseño de medios.

Diseñadores de contenido: Recopilar, sintetizarla y comunicar la información en forma clara y concisa. Son los encargados de plasmar los textos en la aplicación, controlar su aspecto y correcta visualización.

Especialistas en video: Profesional experimentado con habilidades administrativas en todas las fases de producción (desde la concepción hasta la edición final). Debe estar familiarizado con herramientas y técnicas que se emplean para edición digital en ordenador, debe entender el potencial y limitaciones de los medios y cómo estos afectan a la producción.

Especialistas en sonido: Responsables de localizar, seleccionar música y efectos adecuados en la interacción con el usuario programando sesiones de grabación para digitalizar y editar el material y posteriormente grabarlo como archivo en el ordenador. Debe considerar tamaño, formato y calidad de los archivos de audio.

Programador de multimedia: Integra los elementos del proyecto en un conjunto congruente.

Personal de comprobación: Grupo de personas que no formen parte del equipo de creación, que realizan la labor de comprobación final del funcionamiento de la aplicación.

Basado en Web

La Tabla 4.2 muestra las ventajas y limitaciones en el uso de este medio.

Basado en Web	
Ventajas	**Limitaciones**
• Incluye sesiones de charla donde los participantes y el instructor pueden tener diálogo o interactuar en discusiones en tiempo real. • Incluye buzones de correo electrónicos, donde los participantes pueden poner preguntas al instructor o participantes. • Incluyen referencias y capacidades de almacenaje de datos. • Incluye compartición de archivos y datos. • El material del curso puede incorporar audio, video y gráficos. • El material es fácilmente actualizable.	• Gran preparación del instructor para coordinar y planear los elementos del curso. • Requiere habilidades especializadas de diseño, de programadores y autores. • Seguridad, pruebas y retroalimentación puede ser limitada. • Un pobre diseño de interfaz de usuario puede dificultar la navegación o interferir con el aprendizaje. • Una tasa de compresión y descompresión baja, causa que el audio y video pierda sincronización.

Tabla 4.2: Ventajas y limitaciones de medio [16]

Dadas las ventajas y limitaciones expuestas se concluye que el medio de distribución más adecuado es el basado en computador, a continuación se explica los motivos más relevantes:

- El estudiante tiene acceso total e ilimitado al contenido.
- Permite al alumno avanzar en el curso de acuerdo a su capacidad, disponibilidad de tiempo e interés.

- No es necesario que el estudiante disponga de una conexión al Internet.
- El autor no limita la presentación de contenido como video o audio a las tasas de conexión a Internet que dispondrá el alumno.
- No se requiere de equipo costoso para su distribución en la Web.

El medio de distribución basado en computador requiere un equipo completo de trabajo integrado por varios especialistas, pero debido al carácter del proyecto va a ser desarrollado por una sola persona.

Creación de un plan del proyecto

Siguiendo la metodología, se tienen las siguientes tareas para el desarrollo del proyecto:

Tarea	Tiempo aproximado (semanas)
Planificación del proyecto	2
Evaluación de los recursos e identificación de necesidades.	
Desarrollar un perfil del usuario.	
Consideración de las opciones de distribución.	
Creación de un plan del proyecto.	
Desarrollo del contenido	4[1]
Dividir los contenidos en unidades.	
Elegir una aproximación instruccional.	
Crear esquemas o diseños de unidad ("outline").	
Diseño de la disposición, arreglo o distribución de los elementos ("layout")	2
Creación del sistema de navegación.	
Diseño del interfaz de usuario.	

[1] Incluye el desarrollo del contenido textual de la aplicación realizado en el Capítulo 3

Construcción o creación de la aplicación de aprendizaje	5
Selección de tecnología.	
Estructura de la aplicación.	
Incorporación de la interfaz diseñada.	
Creación de medios digitales e implementación de su funcionalidad.	
Evaluación de la aplicación	2
Comprobación de contenidos.	
Comprobación de compatibilidad.	
Comprobación de uso eficaz.	
Total	15 [1]

Tabla 4.3: Plan de proyecto

DESARROLLO DEL CONTENIDO

Dividir los contenidos en unidades

El contenido en la aplicación sigue la estructura desarrollada en el Capítulo 3, de acuerdo con esta cada subtema corresponde a una unidad o capítulo en el curso. Se definieron 7 capítulos, los cuales son:

Capítulo	Título	Subtemas
1	Cifrado y autenticación	– Cifrado – Autenticación
2	Seguridad en la capa acceso de red	– ATM (*Asynchronous Transfer Mode*) – PPP (*Point to Point Protocol*) – L2TP (*Layer 2 Tunneling Protocol*)
3	Seguridad en la capa Internet	– Filtros de paquetes – IPSec (*Internet Protocol Security*) – Seguridad para DNS – NIDS (*Network Based Intrusion Detection System*)

[1] Este tiempo no incluye el aprendizaje de la herramienta de autor

4	Seguridad en la capa transporte	– Proxy a Nivel de Circuito – SSL (*Secure Sockects Layer*) – TLS (*Transport Layer Security*) – ISAKMP
5	Seguridad en la capa aplicación	– Proxy de aplicación – Filtros de contenidos – Controles de acceso y autorización – Seguridad en sistemas operativos – HIDS (*Host Based Intrusion Detection System*)
6	Políticas de Seguridad	– Objetivos de las políticas de seguridad de la información – Políticas, estándares, pautas, lineamientos y procedimientos – Desarrollo de políticas de seguridad – Implementación de políticas de seguridad – Modelo de aseguramiento continuo
7	Productos comerciales y soluciones disponibles en el mercado ecuatoriano	– Autenticación – Firewalls – Antivirus – VPN – Sistema de detección y prevención de intrusos – Control de contenido – Administración

Tabla 4.4: Capítulos del curso

Además del contenido que se presenta en el capítulo 3, la aplicación presenta mayor cantidad de información como: evaluaciones de cada capítulo, información y enlaces de interés de acuerdo al contenido.

Crear esquemas o diseños de unidad ("outline"):

La Figura 4.3 muestra el esquema general de contenido que tendrá cada unidad o capítulo en la aplicación.

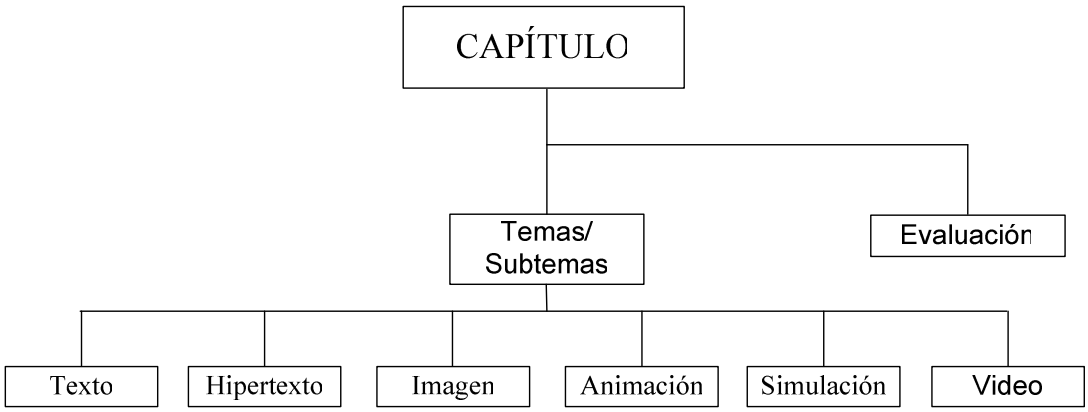


Figura 4.3: Esquema de contenido de cada capítulo

Elegir una aproximación instruccional:

Las aproximaciones instruccionales que se van a utilizar son las siguientes:

Retroalimentación: Aplicación repetida de un conocimiento, para lograr reforzar conceptos importantes en el estudiante.

Evaluaciones: Responder preguntas sobre un tema, para comprobar el nivel de aprendizaje de cada estudiante. En la aplicación por cada capítulo del curso se realizará un examen que consta de ocho preguntas objetivas, basadas en los temas más relevantes del contenido del capítulo.

Apoyo audio-visual: Ver una imagen, video o una secuencia de animación acompañada de sonidos, con lo que se amplía los contenidos textuales y se logra mayor interés por parte del estudiante. La determinación del uso de imágenes o simulaciones en la aplicación se realizará de acuerdo al tema o subtema tratado.

En las aproximaciones instruccionales elegidas para desarrollar el curso se incluyen los siguientes medios:

- **Texto***:* Como una de las vías de información más directas e importantes, este será lo más sintetizado posible. En caso de necesitar insertar mayor información de texto, se recurre a otras técnicas como enlaces a otros documentos, ésto se describe en hipertexto.

- **Hipertexto:** Es una colección de textos organizados para que el usuario no los consulte de forma lineal, sino bajo demanda de sus intereses o necesidades. En la aplicación se presentan datos informativos adicionales dependiendo de cada tema como: RFCs, datos reales o curiosos, y enlaces Web relacionados.
- **Imagen:** Es un medio digital muy importante, se usará en la construcción de la interfaz de la aplicación y en la ilustración de las diferentes escenas de esta.
- **Animaciones:** Es un medio que causa gran impacto en el estudiante, permite describir una idea o un concepto de forma más real que lo que se lograría únicamente con texto o imágenes fijas.
- **Sonido:** Este medio ayudará a enfatizar diversas situaciones y eventos de una aplicación multimedia. En la aplicación se recurre a este medio por ejemplo para indicar cambios a otras unidades del curso.
- **Video:** Es uno de los elementos multimedia más cautivadores, que constituye una potente herramienta para acercar a los usuarios de las aplicaciones informáticas al mundo real, permiten de forma efectiva presentar mensajes o reforzar historias donde los observadores tendrán más posibilidades de retener el contenido de la presentación.

DISEÑO DE LA DISPOSICIÓN, ARREGLO O DISTRIBUCIÓN DE LOS ELEMENTOS ("LAYOUT")

Creación de la estructura de navegación

Se tiene cuatro estructuras fundamentales para proyectos de entrenamiento basado en computador, aunque a menudo se usan de forma combinada. [23]

- **Lineal:** Los usuarios navegan secuencialmente de una porción de información a otra.
- **Jerárquica:** Los usuarios navegan a través de una estructura de árbol a la que da forma la naturaleza lógica del contenido.
- **No lineal:** Los usuarios navegan libremente a través del contenido del proyecto sin sujetarse a rutas predeterminadas

- **Compuesta:** Los usuarios pueden navegar libremente (no linealmente), pero en ocasiones se ven restringidos a presentaciones lineales de información y/o datos críticos que pueden ser mejor ordenados de forma jerárquica.

En la aplicación se utiliza una estructura de navegación compuesta la que se ilustra en la Figura 4.4.

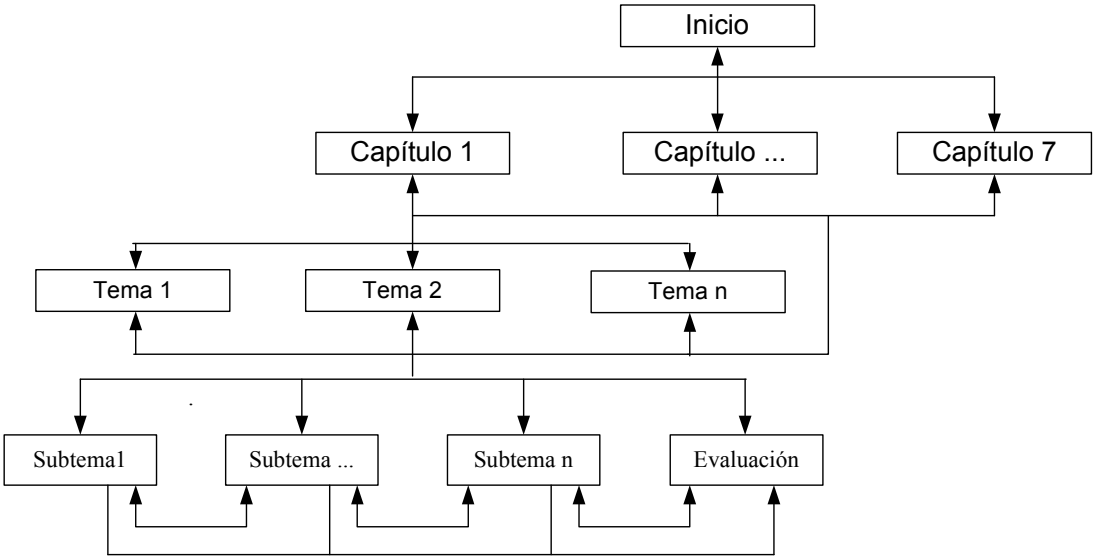


Figura 4.4: Estructura de navegación del curso

La aplicación tendrá una opción de búsqueda que será accesible desde cualquier lugar del curso y permitirá navegar hacia un término específico dentro del contenido.

Diseño de la interfaz de usuario

Esta parte se centra en la forma de diseñar los componentes del software que conforman la interfaz de usuario con el fin de mejorar la comunicación entre el usuario y la máquina. El diálogo usuario-máquina viene facilitado si el usuario tiene acceso a interfaces adecuadas que le ofrezcan los recursos necesarios para navegar y obtener la información que requiere.

Una interfaz debe contar con los siguientes rasgos característicos desde el punto de vista del usuario: [17]

- **Facilidad de manejo:** Permite poder familiarizarse con su funcionamiento básico (contenido y navegación). La mejor interfaz de usuario es la que requiere el menor tiempo de aprendizaje.
- **Originalidad:** Con ello se consigue motivar a la utilización de la aplicación y promover la exploración por los elementos de la interfaz.
- **Homogeneidad:** La interfaz es la herramienta que va a acompañar a lo largo del viaje por la información y por eso tiene que ser lo suficientemente estable u homogénea como para familiarizarse con su funcionamiento en el menor tiempo posible y lo suficientemente versátil para que se amolde a distintas actividades dentro de la misma aplicación.
- **Consistencia:** La interfaz debe ser consistente es decir en situaciones similares se debe emplear la misma secuencia de acciones. Se debe utilizar una terminología idéntica en los mensajes, menús y pantallas de ayuda.
- **Versatilidad:** La versatilidad hará que la interfaz incorpore nuevas funciones específicas para una actividad concreta, manteniendo su estructura básica.
- **Reversibilidad:** La interfaz debe permitir deshacer acciones, debe minimizar la posibilidad de que el usuario pueda cometer errores.
- **Adaptabilidad:** Consiste en ofrecer al usuario la posibilidad de seleccionar las modalidades de navegación en función del tipo de contenidos, de los destinatarios y de los niveles de profundidad previstos.
- **Multimodalidad:** Es una de las características principales de una interfaz de navegación. Se trata de ver la forma de integrar las distintas modalidades de comunicación que pueden ser necesarias para cada concepto.
- **Agilidad:** Una buena interfaz no debe quedarse en una inmejorable presentación física o artística, sino que, sobre todo, tiene que haber un impecable engranaje que se ponga en marcha para cada acción del usuario y que se traduce en un funcionamiento ágil y dinámico del programa.
- **Transparencia:** La interfaz debe ser natural y transparente al usuario, de forma que las posibles actividades implicadas en el aprendizaje de un tema concreto se puedan producir directamente sin mediar procesos incomprensibles o complejos.

- **Interactividad:** El sistema de navegación y la interfaz han de ser interactivos, esto es, deben ofrecer la posibilidad al usuario de que se sienta protagonista dentro del hiperespacio de la información multimedia.

La interfaz del curso de Fundamentos de Seguridad en Redes de Datos, se presenta de una forma amigable para el usuario, permite un fácil manejo y comprensión de su funcionalidad. La Figura 4.5 muestra la interfaz desarrollada, la cual está dividida en tres partes principales: navegación, texto e imagen.

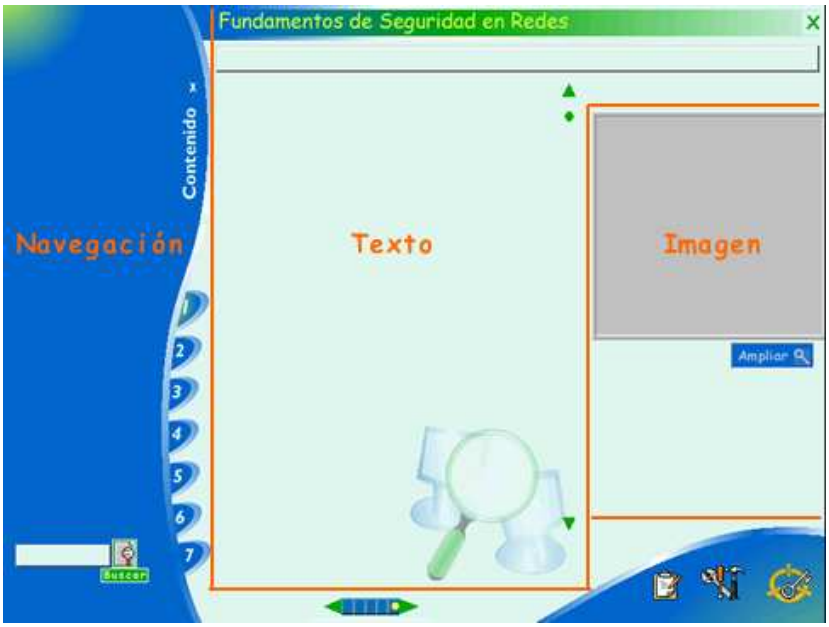


Figura 4 5: Secciones de la interfaz de usuario

Navegación

El sistema de navegación es el canal de comunicación a través del cual se realiza la transferencia de información entre el usuario y la máquina.

Metáforas de interfaces de navegación [17]

El empleo de metáforas en el diseño de la interfaz ayuda a clarificar la naturaleza de los elementos de información que contiene el sistema y consigue que el

usuario capte la manera en la que estos están relacionados. Además, al usuario se le facilitará el acceso a las herramientas que ya le son conocidas, lo cual le permitirá situarse rápidamente en el entorno de trabajo.

Entre las metáforas de la interfaz de navegación hay distintas tipologías cuya aplicación dependerá de factores como: la herramienta de autor, el tipo de destinatario, el tipo de contenido, etc.

Son varias las metáforas que se han utilizado para presentar la información en la pantalla principal: por menús, basados en objetos comunes y espaciales.

- **Metáfora de menús:** Es la más utilizada en aplicaciones multimedia de enseñanza asistida por ordenador. Hay distintas modalidades de menús:
  Los menús cerrados: Están basados en una estructura modular muy útil para evitar la desorientación, mas, al ser una navegación secuencial no permite saltos de forma libre, ya que requiere retroceder al menú inicial para comenzar otras bifurcaciones.
  Los menús abiertos o de selección libre: Requieren descomponer el contenido en pequeñas unidades de información. En esta modalidad se facilita la búsqueda de datos concretos, pero no ayuda a crear una estructura general del contenido.
  El menú mixto: Pretende establecer un hilo conductor básico y, al mismo tiempo, dar una libertad de exploración para profundizar en aspectos concretos. El problema detectado es que al salir de la estructura lineal, es difícil regresar el punto de bifurcación y se llega a perder el objetivo de la búsqueda original.

- **Metáfora basada en aspectos comunes:** Esta metáfora utiliza la familiaridad de objetos como puede ser un archivador de fichas, un libro, un escritorio, una biblioteca, una ventana (*windows*), etc., para presentar la información.
  Suelen combinarse varios de estos objetos para construir una interfaz: archivador de fichas para el menú principal, escritorio para las funcionalidades y ventanas para la navegación por contenidos.

- **Metáfora Espacial:** Esta metáfora utiliza la realidad como modelo. Así suele haber espacios en dos o tres dimensiones en los que se sitúa las distintas opciones para la navegación por la aplicación. Estos espacios pueden ser: paisajes, edificios, escenarios, etc.; en definitiva escenarios que simulan la realidad.

No hay metáfora ideal, lo mejor es combinar las posibilidades que ofrecen cada una de ellas y sobretodo dar la opción al usuario de que realice su selección.

En la aplicación se utilizó una combinación de las metáforas: basada en aspectos comunes y en menús, en parte del menú de contenido (ya que se asemeja a un archivador de fichas para separar el contenido de cada capítulo) y en la opción de herramientas del menú principal (donde se muestra un grupo de opciones que el usuario puede seleccionar) respectivamente.

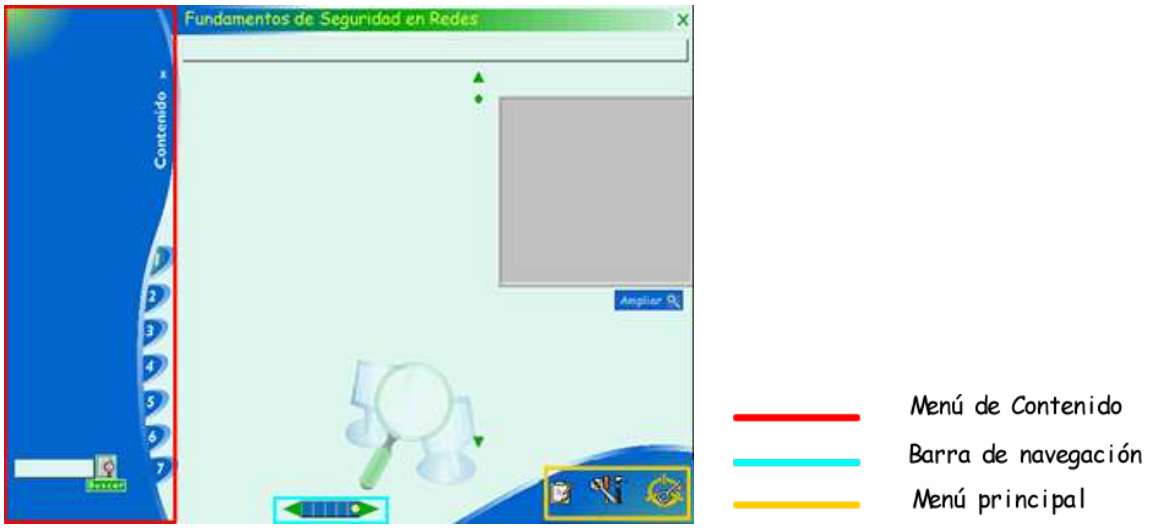


Figura 4.6: Elementos de navegación de la interfaz de usuario

- Menú de contenido

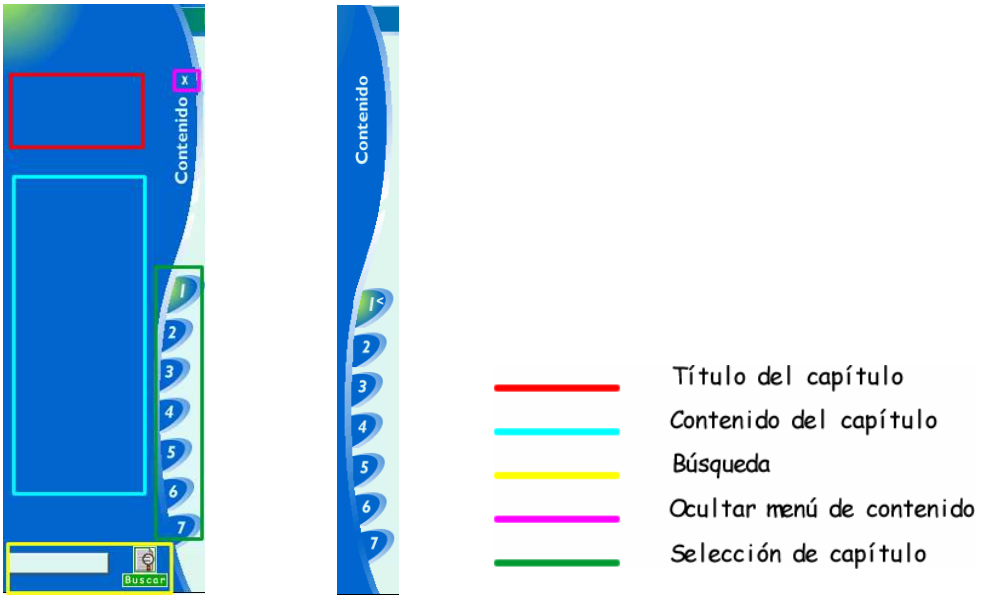


Figura 4.7: Elementos del menú de contenido

- Barra de navegación

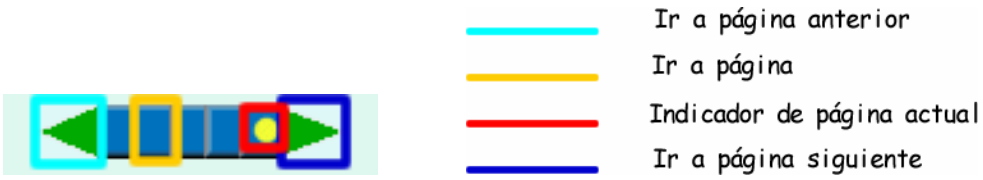


Figura 4.8: Elementos de la barra de navegación

- Menú principal

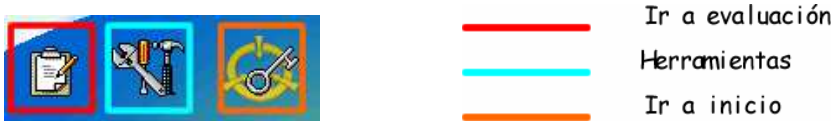


Figura 4.9: Elementos del menú principal

Ventana de evaluación que se presenta al presionar el botón Ir a evaluación:

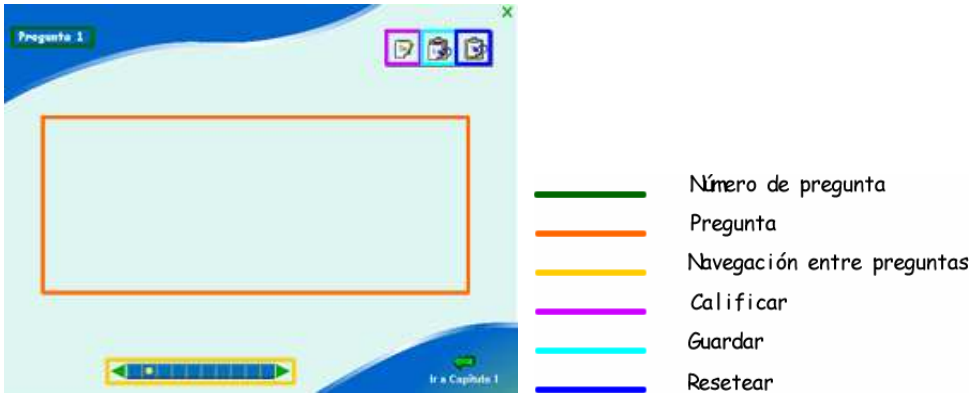


Figura 4.10: Elementos de la ventana de evaluación

Texto

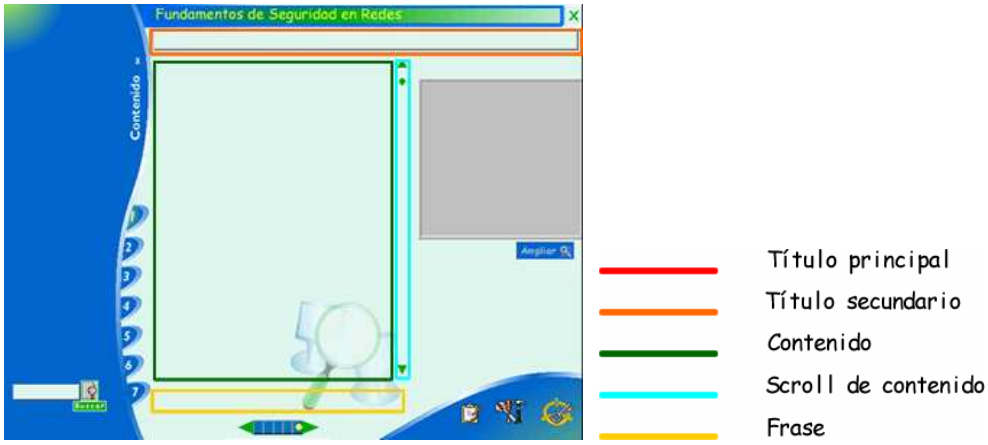


Figura 4.11: Elementos del texto de la interfaz de usuario

Imagen

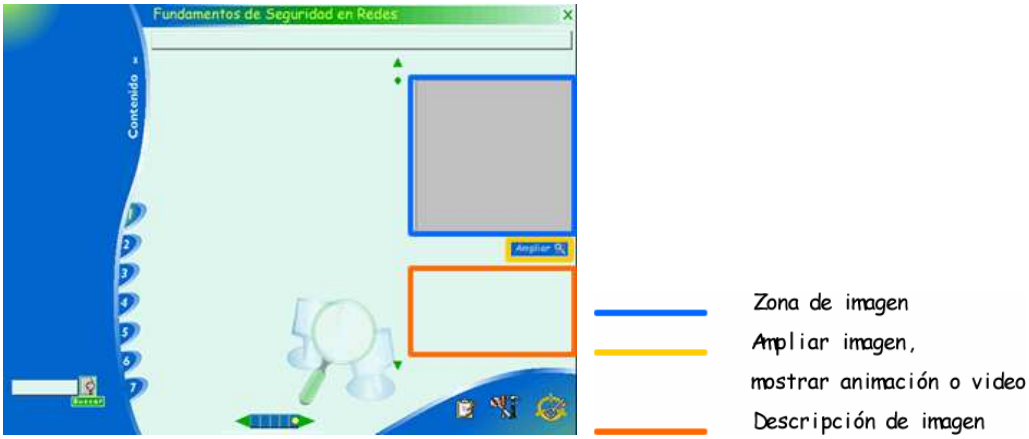


Figura 4.12: Elementos de la sección de imagen de la interfaz de usuario

Ventana de animaciones que se presenta al presionar el botón Ampliar.

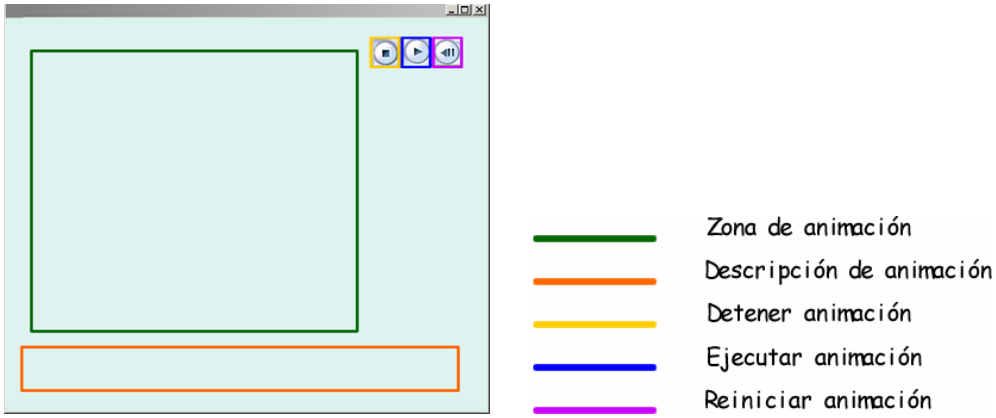


Figura 4.13: Elementos de la ventana de animación

CONSTRUCCIÓN O CREACIÓN DE LA APLICACIÓN DE APRENDIZAJE

Selección de tecnología

Como herramienta de desarrollo de autor se selecciona a ToolBook Instructor 2004[1]. Los requisitos mínimos y recomendados para el funcionamiento de ToolBook Instructor 2004 tanto como para autor o lector se analizaron en el Capítulo 1 en el punto 1.1.1.

A continuación se presenta una tabla con los recursos de hardware disponibles para el desarrollo.

Hardware para el desarrollo	
Procesador	2.4 GHz
RAM	512 MB
Espacio de disco duro	5 GB

Tabla 4.5: *Hardware* de desarrollo

La aplicación a desarrollarse será presentada en lenguaje ToolBook nativo, con lo que será posible usar todas las características de Instructor para la creación de aplicaciones interactivas, incluyendo el lenguaje de programación *OpenScript* y la herramienta de programación visual *Actions Editor*.

[1] Mayor información sobre ToolBook Instructor 2004 en el Capítulo 1

Estructura de la aplicación

La creación de la estructura de directorios es un paso fundamental en el desarrollo del proyecto, ya que permite manejar de forma más organizada los recursos que se utilizarán en la creación de la aplicación y ayuda a la distribución del trabajo final a los usuarios.

En este punto se debe tomar en cuenta las siguientes recomendaciones:

- Nombrar los archivos de medios siguiendo un estándar que permita identificarlos de forma fácil y rápida, acorde a la estructura del proyecto.
- Almacenar los archivos en directorios o carpetas con nombres lógicos.
- Controlar las versiones de sus archivos (seguir la pista de los cambios de edición), especialmente en proyectos largos.
- Si en un grupo de archivos se encuentra trabajando más de una persona, debe asegurarse de que siempre se sabe cuál es la última versión y quién la tiene.
- Almacenar todas las versiones que se han realizado de la aplicación, por si en un momento se desea utilizar una de estas.

Una aplicación ToolBook se compone de archivos llamados libros, los que residen en un directorio central que además tiene subdirectorios para los archivos de medios que se incorporan a estos libros.

La Figura 4.14 muestra un diagrama con la estructura de directorios de la aplicación:

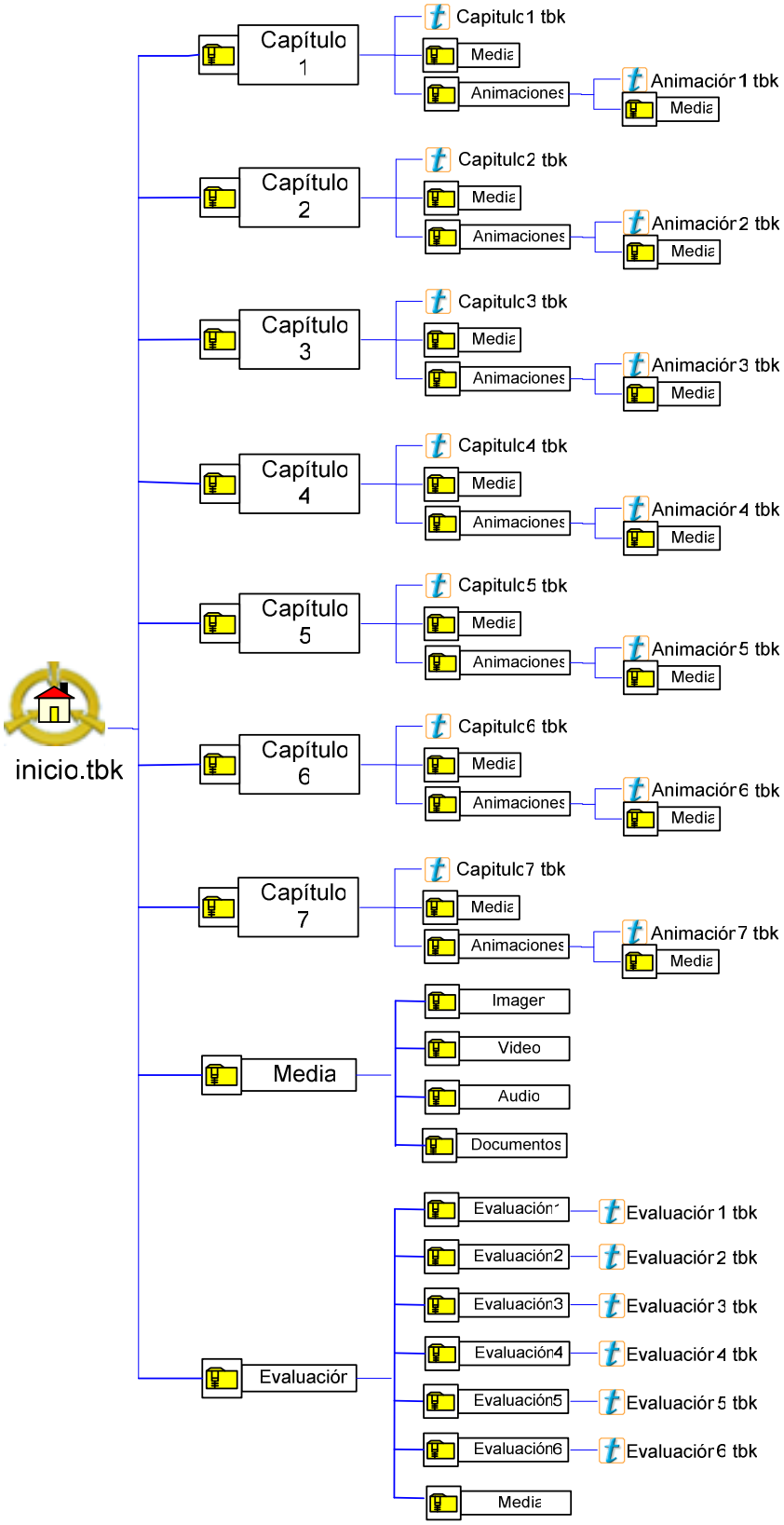


Figura 4.14: Estructura de directorios de la aplicación

Incorporación de la interfaz diseñada

En este paso se realizaron las siguientes actividades:

Crear un libro matriz

Crear un libro nuevo e incorporar los objetos que forman parte del diseño de la interfaz. Los elementos comunes a varias páginas o libros se agrupan en un fondo o *background*, en la aplicación se creó el siguiente *background* para las páginas de contenido de cada capítulo.

Figura 4.15: *Background* de páginas de contenido de capítulos

Utilización de los libros matrices, para formar la estructura de la aplicación

Una vez que se ha logrado desarrollar un libro modelo con las funcionalidades básicas, se procederá a guardarlo como un libro plantilla de ToolBook y se utilizará en la creación de los otros libros que forman parte de la estructura de la aplicación.

A continuación se construyó la estructura navegación general, se creó enlaces entre los diferentes libros creados, desde inicio.tbk a los libros de cada capítulo (capítulo1.tbk) y desde cada libro de capítulo con el respectivo libro de evaluación (evaluación1.tbk) y animación (animación1.tbk)

Creación de medios digitales e implementación de su funcionalidad

Para crear los medios digitales se deben tomar en cuenta pautas de diseño de aplicaciones multimedia, se enumera algunas de las más importantes: [20]

Texto Se puede hablar de cuatro reglas fundamentales para incluir texto en una pantalla de ordenador:

1. Textos breves: No se debe incluir cantidades de texto exageradas por cada página.
2. Utilización de hipervínculos de texto: Para interconectar distintos textos o hacer aparecer texto en una ventana flotante.
3. Uso de términos adecuados: Usar términos de amplia aceptación para conseguir una audiencia lo más amplia posible para la aplicación. En cuanto a las leyendas del sistema de navegación éstas deben ser términos aceptados por la industria del software a los cuales el público está más acostumbrado (menú, barra de herramientas, mapa, ayuda, etc.).
4. Inteligibilidad: La selección de las fuentes que deben figurar en una aplicación multimedia puede resultar algo difícil desde el punto de vista del diseño. A continuación se enumeran ciertos consejos**:**
    - Utilizar fuentes legibles.
    - Utilizar el menor número de tipos de letras, variando el peso, tamaño, color, efecto de las mismas dependiendo de la importancia del mensaje que desea transmitir.
    - Considerar el interlineado, que las líneas demasiado pegadas resultan difíciles de leer.
    - Ajustar los espacios entre caracteres.
    - Reducir al máximo el número de líneas de un bloque de texto centrado.
    - Utilice efectos que hagan del texto más llamativo al usuario.
    - Utilice palabras o frases significativas para los vínculos y elementos de los menús.
    - Usar fuentes comunes (que se puedan encontrar instaladas en la mayoría de ordenadores) para evitar la descolocación del texto en las pantallas.

Hipertexto Para el uso de hipertexto se debe seguir algunas consideraciones:

- No abusar del uso de hipertexto.
- Usar colores o imágenes que permitan al usuario identificar el tipo de información que se va a mostrar.
- Evitar encadenamientos largos de consulta, el usuario siempre espera encontrar la información que le falta apenas activado el enlace.

Imagen

- Utilizar colores acordes a los utilizados en la interfaz diseñada.
- Las imágenes deben ser legibles, de otra forma no se justifica su presencia.
- Escoger formatos de imágenes que sean compatibles con la herramienta de autor.
- El número de imágenes, debe estar en función de la capacidad de almacenamiento y del resto de material a incluir.
- Utilizar gradientes para dar un efecto de iluminación que es atractivo al usuario

Animaciones

- Deber ser consistente al tema que se está tratando.
- No mostrar demasiados elementos o presentarlos de una manera muy rápida.

Video

- Seleccionar el formato adecuado acorde a la herramienta de desarrollo y el equipo que dispondrá el estudiante.
- Debe ser cuidadosamente planificado y reproducido de acuerdo al objetivo que se persigue con el mismo.

Audio

- Incluir en la aplicación la opción de deshabilitar el audio.
- Utilizar sonidos que reflejen el ritmo del proyecto y que puedan afectar a la actitud de la audiencia.
- No usar sonidos que distraigan la atención del usuario o impidan su concentración.

Los formatos seleccionados en la aplicación, son los siguientes:

Medio	Formato	Ubicación en la interfaz
Texto	Verdana 10, negro	Contenido
	Comic Sans MS 12, azul, negrita	Títulos
	Arial Narow 7, blanco	Menú de contenido
	Verdana 10, verde, cursiva	Frase
Hipertexto	Verdana 10, azul, cursiva	Hiperenlaces en contenido
Imagen	Gif, jpg	Zona de imagen
Animaciones	Path-based	Ventana de animación
	Cell-based	
	Gif	
Audio	Wav	Botones de navegación
Video	Avi	Ventana de animación

Tabla 4.6: Formato de medios

Para identificar la información que muestra un hipertexto, se tiene los siguientes formatos:

Objeto	Evento	Descripción
	*Mouseover*	Carga imagen en la zona de imagen.
	Clic	Abre ventana de animación o video.
	*Click*	Enlaces a información adicional.
	*Click*	Enlace a datos interesantes.

	*Click*	Enlace a sitios Web.
	*Click*	Enlace a aplicaciones.

Tabla 4.7: Formato de hipertexto

Para la creación e incorporación de funcionalidad de los medios, se utilizaron herramientas propias de ToolBook como: *IconEdit*, *BitEdit*, *Animation Editor*, *WaveEditor*, *Actions Editor*, *Openscript* y los objetos del catálogo. Además se utilizaron otras herramientas independientes que permiten la creación de medios como: Firewoks MX 2004 y Visio 2003.

Luego de crear los medios se sigue los procesos de incorporación de los mismos, explicados en el Capítulo 1. La Figura 4.16 muestra una página del libro donde ya se ha incorporado los medios digitales:[1]

Figura 4.16: Página del curso desarrollado

[1] Anexo H: Ejemplos de creación de medios

Para la creación de evaluaciones se utilizaron toda la gama de objetos preprogramados del catálogo del ToolBook, [1]

PRUEBAS DE OPERACIÓN

EVALUACIÓN DE LA APLICACIÓN

Comprobación de contenidos:

Luego de realizar varias pruebas internas del contenido a cada capítulo de la aplicación, se llegó a una versión de máxima calidad libre de los errores que se describen en la Tabla 4.8.

Capítulo	Errores ortográficos	Calidad de Imágenes	Calidad de animaciones y video	Sistema de navegación	Acciones de elementos
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓

Figura 4.17: Comprobación de contenidos

Comprobación de compatibilidad:

En este punto se debe comprobar los siguientes aspectos:

- Probar las diferentes resoluciones gráficas para la aplicación:

[1] Anexo I: Ejemplos de preguntas de las evaluaciones

La aplicación trabaja de forma óptima en las siguientes resoluciones de pantalla:
800 x 600 pixels
1024 x 768 pixels

- Profundidad de color adecuada a la aplicación y cómo resulta la aplicación con otras profundidades.

  La calidad del color a la que se desarrolló es de 16 bits, se recomienda utilizar una calidad igual o superior.

- Audición de todos los sonidos de la aplicación y comprobación del software y hardware mínimo necesario para su audición. Un aspecto importante del sonido es su sincronización con la imagen.

  Se requiere únicamente que la tarjeta de sonido sea compatible con Windows y que se disponga de parlantes.

- Visualización de la correcta ejecución de animaciones y videos, comprobación de software y hardware mínimos.

  Memoria RAM de 128 MB o superior.
  Sistema operativo: Windows 2000, 2003 o XP (recomendado).
  Se requiere de la instalación previa de Neuron para su funcionamiento.[1]

Esta comprobación lleva un paso previo al inicio del diseño de cada uno de estos aspectos, pues un error en este paso puede llevar a tener que reformar toda la aplicación.

[1] El instalador de Neuron se incluye en el CD del curso o puede ser descargado desde www.sumtotalsystems.com

Comprobación de uso eficaz:

Se deben comprobar los siguientes aspectos:

- Si la aplicación en sí despierta interés al usuario.
- Si la interfaz cumple las características de su diseño y el usuario comprende todas las funcionalidades.
- Si el sistema de navegación es de fácil manejo y el usuario comprende todas sus funcionalidades.
- Si el tiempo de la aplicación, de sus animaciones y videos es correcto, y no aburre al usuario.
- Si no existen puntos de colisión o congestión en los cuales el usuario tenga dudas de que opción tomar.

Finalmente, realizar encuestas a los usuarios de prueba para que informen de su opinión sobre la aplicación y si ha respondido a sus expectativas. [24]

Los resultados de las encuestas realizadas a usuarios que cumplen con el perfil y sus opiniones sobre la aplicación ayudaron a identificar y corregir pequeñas fallas, permitiendo llegar a la presentación de la versión final de la aplicación. [1]

[1] Anexo J: Encuesta de comprobación del uso eficaz

CAPÍTULO 5
CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- El uso de multimedia ha tomado fuerza principalmente en las disciplinas de entrenamiento y aprendizaje, se integra a ambientes basados en computador y Web; permitiendo implementar múltiples experiencias de aprendizaje que facilitan la educación.

- La aplicación de tecnologías multimedia en el proceso de enseñanza, cambia el rol de los miembros que intervienen en éste, obliga al estudiante a tener un papel más activo, mientras que el profesor coordina un trabajo grupal de responsabilidades compartidas.

- En el ámbito universitario la aplicación de tecnologías de enseñanza basadas en computador o en Web seguirá en aumento, ya que permiten abarcar una sociedad muy diversificada en lo que refiere a la edad, lugar de residencia y situación personal de los alumnos. Abriendo nuevas oportunidades de capacitación a personas que por problemas de tiempo y/o distancia no pueden acceder a actividades formativas. Además permiten que el estudiante avance en el proceso educativo a su ritmo y necesidad.

- CBTs y WBTs son altamente útiles en la formación corporativa, ya que la mayoría de empresas buscan ser competitivas mediante la continua capacitación de sus empleados, y la utilización de las nuevas tecnologías es muy conveniente para obtener mejores resultados en la enseñanza.

- La tendencia a utilizar medios de aprendizaje electrónicos, en lugar de los tradicionales (aula e instructor), está motivada en gran parte por el aspecto económico; considerando el ahorro que implica desarrollar un curso de entrenamiento basado en computador o Web, dirigido a una audiencia

significativa, frente a los excesivos costos que representa enviar a grandes grupos a sitios remotos por largos periodos de tiempo para acceder a un curso tradicional.

- La selección de la herramienta de autor dependerá del tipo de proyecto multimedia a realizar y de las necesidades que éste va a cumplir. Entre las herramientas más destacadas están ToolBook Instructor, Authorware, Director y Scala Multimedia.

- Para la creación de cursos asistidos por computador ToolBook Instructor 2004 es una de las mejores herramientas disponibles, ya que permite añadir a la aplicación el mayor grado de interactividad, dispone de una amplia gama de recursos preprogramados y brinda flexibilidad para su personalización con herramientas avanzadas como *Actions Editor* y *OpenScript*.

- Las metáforas de trabajo en las que se basan las herramientas de autor ayudan a mejorar el entendimiento del proyecto a desarrollar. Por ejemplo, ToolBook Instructor utiliza la metáfora de libros (considera a la aplicación como un libro, dentro de éste se ubican las páginas o pantallas y en éstas los recursos multimedia), figura con la que el desarrollador se identifica rápidamente debido al común uso de libros; mientras Macromedia Director utiliza la metáfora de película (la aplicación es considerada una película, donde el director es el creador de la misma, está compuesta de escenas o *frames* y los elementos que forman parte de cada escena son denominados actores o miembros de reparto) la que en ciertos casos resulta abstracta.

- ToolBook Instructor permite crear aplicaciones interactivas de forma sencilla sin la necesidad de programar en su lenguaje *OpenScript*, a diferencia de otras herramientas de autor como Director, donde para habilitar la posibilidad de que el usuario intervenga de forma activa en la película se requiere el uso de su lenguaje de programación *Lingo*.

- En una fase inicial del desarrollo de cursos multimedia es fundamental definir el método de distribución, Instructor habilita la posibilidad de presentación sobre la Web, en una intranet, en una red de área local o en un CD-ROM, pero es importante destacar la flexibilidad que brinda Instructor para migrar desde un método de distribución a otro.

- El método de distribución del presente proyecto es CD-ROM, pero debido a las facilidades presentadas por Instructor y la forma de desarrollo del proyecto es fácilmente adaptable para funcionar sobre una intranet o Internet. En el caso de implementarlo en la Web se debe considerar la publicación del curso en un servidor de Internet *"hosting"* y que el usuario disponga de una conexión al Internet.

- ToolBook Instructor permite extender su funcionalidad con la incorporación de componentes de *software* adicionales como controles ActiveX y DLLS, los que pueden además ser manipulados a través de secuencias de acción y *OpenScript*.

- El conocimiento de la herramienta de autor y sobre todo la experiencia en la creación de CBTs tiene vital importancia al momento de desarrollar un curso multimedia, su relación es directa con el tiempo de desarrollo y es determinante en la calidad de la aplicación final.

- La correcta selección y seguimiento de una metodología de diseño instruccional multimedia, permite un desarrollo ordenado y consistente de la aplicación, lo cual garantiza obtener el resultado deseado. Sin embargo, cabe aclarar que dependiendo del alcance de la aplicación, algunas fases de desarrollo o tareas de ejecución dentro de la metodología pueden variar.

- Dentro del proceso de enseñanza es más importante la información que se desea transmitir, que los medios de distribución utilizados (Internet, CD-ROM, etc.); ya que la tecnología no es sino la herramienta que facilita el acceso al contenido.

- El aprendizaje a través de computador o Web requiere que el estudiante esté realmente motivado y predispuesto al esfuerzo que supone asumir la responsabilidad del aprendizaje.

- Independientemente del tipo de aplicación instruccional multimedia, el proceso de desarrollo básico es el mismo. Primero, establecer una estructura de desarrollo; luego crear los elementos de medios que estarán dentro de esta estructura; entonces revisar y probar el producto; y por último, implementar el producto final.

- Un paso esencial dentro de cualquier metodología de desarrollo multimedia es el análisis de la audiencia, éste es la base para la definición de las etapas posteriores en la metodología, puesto que son los usuarios finales los que determinarán la eficacia de la aplicación.

- La funcionalidad de la interfaz de usuario está limitada por la imaginación del diseñador, habilidad del programador y la tecnología disponible para el desarrollo.

- La característica de interactividad le da un valor agregado a las aplicaciones instruccionales multimedia, ya que permite que el usuario se sienta protagonista de su propio aprendizaje, al controlar y manejar la aplicación.

- Las evaluaciones y pruebas de operación son cruciales en la creación de cursos multimedia, éstas se deben realizar periódicamente, tanto dentro del equipo de desarrollo, como también a usuarios reales; para corregir tempranamente errores y garantizar la eficacia del producto final. Esto puede lograrse realizando encuestas, donde de forma directa se obtiene la opinión de los usuarios.

- Los cursos de entrenamiento multimedia requieren un constante mantenimiento y actualización; este proceso en CBTs requiere “quemar”

nuevos CD-ROMs con el costo y tiempo que esto conlleva, mientras que en WBTs el proceso de actualización se realiza en un menor tiempo.

- Una posibilidad que se abre al llevar un curso basado en computador a la Web es la capacidad de habilitar la comunicación del estudiante con el instructor u otros estudiantes en tiempo real mediante sesiones de charla o foros.

RECOMENDACIONES

- La mayoría de herramientas de autor tratan de facilitar la creación de aplicaciones al permitir su desarrollo sin la necesidad de programación, puesto que el tiempo de aprendizaje de un lenguaje de programación inicialmente supone pérdidas de productividad. Sin embargo, al momento de seleccionar una herramienta de autor es conveniente que ésta incluya un buen lenguaje de programación, el que permita tener la mayor flexibilidad posible. Lo cual es garantizado al utilizar ToolBook Instructor y su lenguaje de programación *OpenScript*.

- En desarrollo de aplicaciones CBT se debe considerar el tamaño de disco duro disponible, ya que es conveniente guardar respaldos de los archivos de medios y de versiones compiladas de la aplicación. Se recomienda considerar diez veces el tamaño que se prevé tendrá el curso finalizado.

- Es recomendable que el equipo de desarrollo de autor disponga de los mejores recursos de *hardware* y además cuente con las últimas versiones disponibles de *software* de producción.

- Una técnica recomendada en especial para desarrolladores no experimentados, es analizar las ventajas y limitaciones en la estrategia de diseño de productos similares, para extraer ideas que ayuden a mejorar la aplicación a desarrollar. Sin embargo, es importante recalcar que la creatividad e innovación están ligadas a un buen desarrollo multimedia.

- Se recomienda tomar en cuenta los objetivos que se desea alcanzar con el curso de aprendizaje y su medio de distribución, durante el proceso de selección de formatos y creación de medios a incorporarse.

- Crear una estructura de directorios adecuada y respetarla durante el proceso de desarrollo, permitirá mantener una correcta organización de los archivos y recursos que éstos necesitan para su funcionamiento.

- En Instructor se recomienda guardar de forma periódica un libro de respaldo bajo un nombre nuevo en otro directorio y luego volver a guardarlo con su nombre original en el directorio inicial; el libro de respaldo podrá utilizarse en caso de producirse fallas en la primera versión, mientras que el original conseguirá con este procedimiento reducir su tamaño, lo cual también ayudará a prepararlo para su entrega final.

- En el desarrollo de aplicaciones multimedia se genera gran cantidad de archivos temporales que podrían afectar el desempeño del equipo de trabajo, se recomienda continuamente realizar una depuración de estos con herramientas de *software* especializadas que pueden ser las incorporadas al sistema operativo.

- Se recomienda ser cuidadosos al añadir controles ActiveX en la aplicación desarrollada en ToolBook, ya que están orientados a trabajar en Web y podrían presentar funcionalidad inesperada cuando se ejecutan en otros ambientes de trabajo.

- Cuando se trabaja con objetos agrupados se debe poner atención especial en la manipulación de sus propiedades, tomando en cuenta que el grupo tendrá propiedades independientes a las de los objetos que forman parte de él.

- Al importar páginas o *backgrouds* se debe primero hacer visibles todos los objetos de las mismas, para que mantenga la funcionalidad original y el resultado sea el esperado.

- Instructor almacena como recursos a todos los objetos que se incluyan al libro, si los objetos son eliminados, es recomendable verificar en el *Resource Manager* si se mantiene un registro de éstos y si es necesario removerlos manualmente.

- Aprovechando las facilidades que presta ToolBook Instructor, a futuro podría implementarse otras aplicaciones que estudien diferentes temáticas. El presente proyecto contempla los aspectos básicos de Seguridad en Redes, dentro de este mismo campo se podría profundizar en el estudio de un tema específico, como por ejemplo: crear simulaciones de algoritmos de cifrado, simulaciones de configuración de equipos, manuales interactivos de manejo de aplicaciones, etc.

ANEXO A: Botones de comando de la barra de herramientas y sus funciones[1]

Botón de la barra de herramientas por defecto **Botón de la barra de herramientas cuando CTRL es presionado**	
 Abre otra aplicación ToolBook.	 Ejecuta otra aplicación o instancia de Instructor.
 Guarda el libro.	 Guarda el libro bajo un nombre diferente.
 Deshace la última acción.	
 Duplica el objeto seleccionado.	
 Muestra u oculta el *Object Browser.*	 Muestra u oculta el *Page Browser.*
 Muestra u oculta el Catálogo.	
 Muestra u oculta la ventana *Command.*	
 Muestra u oculta la paleta de herramientas.	
 Muestra u oculta la paleta línea.	 Muestra u oculta la paleta flechas.
 Muestra u oculta la paleta de patrones.	
 Muestra u oculta la bandeja de color.	
 Muestra u oculta la paleta polígono.	
 Muestra el background actual.	
 Añade una nueva página al libro.	
 Abre el cuadro de diálogo *Resource Manager.*	

[1] ToolBook Instructor 2004 User Guide, página 57

 Abre el cuadro de diálogo *Clip Manager.*	
 Abre el cuadro de diálogo *Viewers.*	
 Abre el cuadro de diálogo *Properties.*	
 Abre el cuadro de diálogo *Extended Properties.*	
 Abre el *Actions Editor.*	
 Abre el Editor de *scripts.*	 Abre el Editor de *scripts* para compartir *scripts.*
 Abre la lengüeta *Font* del cuadro de diálogo *Properties.*	 Abre la lengüeta *Paragraph* del cuadro de diálogo *Properties.*
 Agrupa o desagrupa objetos.	
 Lleva al objeto al frente.	 Mueve el objeto más cercano.
 Envía a atrás.	 Mueve el objeto más lejano.
 Gira horizontalmente.	 Gira verticalmente.
 Rota hacia la izquierda.	 Rota hacia la derecha.

ANEXO B: Barra de herramientas en el cuadro de diálogo Properties

El cuadro de diálogo *Properties* contiene una barra de herramientas con accesos a otros cuadros de diálogo:

Los botones en la barra de herramientas están disponibles dependiendo de las características del objeto y especificaciones de cada *tab*.

A ***Undo*** Revierte el último comando.

B ***Redo*** Reaplica el último comando.

C ***Edit Parent*** Cambia las propiedades desplegadas al *parent* (padre) del objeto seleccionado actualmente.

D ***Edit Child*** Cambia las propiedades desplegadas al *child* (hijo) del objeto seleccionado actualmente.

E ***Set Units*** Permite escoger con que unidades el objeto es medido (*pixels*, unidades de página, pulgadas, o centímetros).

F ***Extended Properties*** Abre el cuadro de diálogo *Extended Properties*.

G ***Actions*** Abre el *Actions Editor*.

H **Hyperlink** Abre el cuadro de diálogo *Hyperlinks*.

I **Path Animation** Abre el ToolBook *Animation Editor*.

J **Edit Script** Abre el editor de *script*.

K **Edit Shared Script** Abre el editor de *script* compartido.

L **Select Shared Script** Permite escoger un *script* compartido para editar o crear un nuevo *script* compartido.

M **Help** Abre el tópico de ayuda en línea para el *tab* desplegado actualmente en el cuadro de diálogo *Properties*.

ANEXO C: Tabla de Acceso al menú right-click

Para desplegar un menú *right-click* de un objeto: *click* el botón derecho del *mouse* de acuerdo a la tabla siguiente[1]:

Tipo de objeto	**Donde hacer *click***
Página	*Click* derecho en un punto vacío de la página, o CTRL.+ *click* derecho en cualquier parte del *background*.
Background	CTRL + *click* derecho en la página, o *click* derecho en un punto vacío en el *background*.
Viewer	CTRL + SHIFT + *click* derecho en cualquier lugar de la *viewer*.
Libro	SHIFT + *click* derecho en cualquier lugar de la página.
Otros objetos	*Click* derecho en el objeto.

[1] ToolBook Instructor 2004 User Guide, página 79

ANEXO D: Tipos de Media Player

A continuación se muestra las características principales de los objetos Media Player de Instructor[1]:

Media placer	Soporta	Ventajas	Limitaciones
Universal Media Player	La mayoría de los formatos actualmente disponibles de video digital, audio y de medios streaming.	Trabaja igual cuando es exportado a DHTML como en tiempo de ejecución en ToolBook Programable usando *Action Editor* Tiene un interfaz de autor consistente, sencillo, el cual lo hace fácil de aprender y usar.	No muestra gráficos sin movimiento.
Media players	Audio específico, video digital y archivos de imagen fija, como se enumeran en la descripción del Catálogo.	Permite mayor control sobre la apariencia del objeto en la página en tiempo de ejecución de Instructor.	Ejecuta ciertos tipos de archivos media, usuarios deben cargar el controlador del propio *software* en el computador Permite programación limitada *Actions Editor*.
Video players	Archivos AVI (video digital).	Permite mayor control sobre la apariencia del objeto en la página en tiempo de ejecución de ToolBook.	Ejecuta ciertos tipos de archivos media, usuarios deben cargar el controlador propio del software en la computadora Permite programación limitada con *Actions Editor*.

[1] ToolBook Instructor 2004 User Guide, página 329

Inline clip players	Audio específico, video digital y archivos de imagen fijos, como se enumeran en la descripción del Catálogo.	Puede ser insertado en un campo de texto de manera que el botón se desplace con el texto, similar a un *hotword*.	Ejecuta cierto tipo de archivos media, usuarios deben instalar el controlador propio del *software* en el computador Permite programación limitada con *Actions Editor*.

El Universal Media Player

El Universal Media Player se encuentra en la categoría Media Players del Catálogo, soporta una amplia variedad de formatos, entre los cuales se tiene[1]:

Si este player es disponible...	Estos formatos de archivos se ejecutarán...
ToolBook (no requiere player)	Archivos de video: *.avi Archivos de sonido: *.wav Archivos MIDI: *.mid, *.midi,*.rmi
RealPlayer	Archivos RealMedi: *.ra, *.rm, *.rmj Archivos SMIL: *.smi, *.smil Archivos RealText: *.rt Archivos RealPix: *.rp Archivos MP3: *.mp3, *.dat Archivos MP3 Playlist: *.m3u, *.pls Meta archivos RealMedia: *.ram, *.rmm, *.rpm Archivos RealText3D: *.r3t Archivos de audio: *.aa
Shockware/Flash Player	Archivos Shockware y Flash: *.swf
Windows Media Player	Archivos Windows Media: *.asf, *.asx, *.wm, *.wma, *.wmv, *.wax Archivos MP3: *.mp3, *.mpa, *.m3u Archivos MPEG: *.mpg, *.mpeg, *.m1v, *.mp2 Archivos QuickTime: *.mov, *.qt Archivos Audio: *.aif, *.aifc, *.aiff

[1] ToolBook Instructor 2004 User Guide, página 331

ANEXO E: Tabla de tipos y propiedades de recursos[1]

Tipo de recurso	Extensión de archivo	Programas usados para creación
Bitmap	.bmp, .dib .gif, .jpg	ToolBook *BitEdit* u otro programa de Windows.
Cursor	.cur	ToolBook *Icon/Cursor Editor* u otro programa de Windows.
Font	.ttf	Fuentes TrueType.
Icono	.ico	ToolBook *Icon/Cursor Editor* u otro programa de Windows.
Barra Menú	.mnu	ToolBook *Menu Bar Editor*.
Paleta	.pal	ToolBook *PalEdit* u otro programa de Windows.
Script compartido	-	Editor de *script* de Instructor.

[1] ToolBook Instructor 2004 User Guide, página 329

ANEXO F: Tipos de operadores OpenScript

Operators for arithmetic expressions

Operator	Meaning
+	Addition. For example: `It = 25.3 + vFirst  --Puts result into It`
–	Subtraction. For example: `It = 25 - 4  --Puts 21 into It`
*	Multiplication. For example: `It = 25 * 4  --Puts 100 into It`
/	Division. For example: `It = 25 / 4  --Puts 6.25 into It`
div	Division (returns only the integer). For example: `It = 25 div 4.5  --Puts 5 into It`
–	Negation. For example: `It = -4 + 25  --Puts 21 into It`
^	Exponentiation. For example: `It = 2 ^ 4  --Puts 16 into It`
mod	Modulo (returns the remainder). For example: `It = 16 mod 5.0  --Puts 1 into It`

Operators for logical comparisons

Operator	Meaning

= or *is*	Equal. *True* if both expressions have same value. For example:
	`It = (25=4)  --Puts false into It`
	`if color is blue then`
	`  color = red`
	`end`

<	Less than. *True* if expression on left is less than expression on right. For example:
	`if balance < lowLimit then`
	`  text of field "odWarning" = "Overdraft"`
	`end`

<=	Less than or equal to. *True* if expression on left is less than or equal to expression on right. For example:
	`if text of field "Date" <= sysDate as date`
	`  request "Too early to process"`
	`end`

>	Greater than. *True* if expression on left is greater than expression on right. For example: `get 25 > 4  --Puts true into It`

>=	Greater than or equal to. *True* if expression on left is greater than or equal to expression on right. For example:
	`if vFirst >= vLast`
	`  request "Last item is less than first."`
	`end`

<> or *is not*	Not equal to. *True* if two expressions are different. For example:
	`if svText <> text of field "Text"`
	`  text of field "Text" = svText`
	`end`

and	Boolean And. *True* if both expressions are true. For example:
	`if text of field "input" > "01/01/94" as date and \`
	`  text of field "refAmount" < 25`
	`  request "The expression is true."`
	`end`

or	Boolean Or. *True* if either expression is true. For example:
	`if interest > 10.5 or interest < 9.5 then`
	`  request "Interest rate is out of range."`
	`end`

contains	String search. *True* if expression on right is found in expression on left. For example:
	`if text of field "emperors" contains "Roman"`
	`  go to book "c:\lessons\roman.tbk"`
	`end`

is in	String search. *True* if expression on left is found in expression on right. For example:
	`if nameOnly(vName) is in text of field \`
	`  "chooseIndex"`
	`  request "That index is already open."`
	`end`

is not in	String search. *True* if expression on left is not found in expression on right. For example:
	`if "Nero" is not in vAnswerText`
	`  request "Please try again."`
	`end`

not	Not. *True* if expression on right is false; *false* if expression on right is true. For example:
	`--File name is in title bar`
	`if not captionShown of mainWindow`
	`  request "Do you want to personalize this book?" \`
	`  with "Yes" and "No"`
	`  if It is "Yes"`
	`  send personalize -User-defined message`
	`  end if`
	`end if`

Operators for string evaluations

Operator	Meaning
&	Concatenates two expressions with no space between them. For example: `push (bookPath & "mysysbook.sbk") onto sysBooks`
&&	Concatenates two expressions with a space between them. For example: `request "The" && "End" --Displays "The End"`

char(s) or *character(s)*	One or more contiguous characters of text, including letters, numbers, spaces, punctuation marks, tabs, and other special characters. For example:
	`if char 1 of userResponse is "Y" then`
	`  send exit`
	`end`
	`vFirst3 = chars 1 to 3 of fullName`

word or *words*	One or more contiguous sequences of printable characters separated on each side by a space or a nonprintable character, or by the beginning or end of the expression that includes the word. For example: `surname = last word of text of field "FullName"`

item or *items*	One or more contiguous values in a comma-separated list. For example:
	`xPosition = item 1 of mousePosition of this window`
	`yPosition = item 2 of mousePosition of this window`

textline or *textlines*	A string of zero or more characters that ends with a carriage return/linefeed (CRLF), or two or more strings separated with CRLFs. For example:
	`allLines = text of field "listbox1"`
	`step ctr from 1 to textLineCount(allLines)`
	`  request textline ctr of allLines`
	`end`

Operators for bitwise operations

Operator	Meaning

bitAnd	Performs a bitwise AND operation on two numbers. For example:
	`--Displays 2 (binary 11 AND 10 yields 10)`
	`request 3 bitAnd 2`

bitOr	Performs a bitwise inclusive OR operation on two numbers. For example:
	`--Displays 3 (binary 01 OR 10 yields 11)`
	`request 1 bitOr 2`

bitXOr	Performs a bitwise exclusive OR operation on two numbers. For example:
	`--Displays 1 (binary 11 XOR 10 yields 01)`
	`request 3 bitXOr 2`

bitNot	Converts all bits to their opposite values (performs a one's complement of a number). For example:
	`--Displays -2 (binary 00000001 NOT yields 11111110)`
	`request bitNot 1`

bitShiftRight	Performs a bitwise shift to the right on a number by a specified number of bits. For example:
	`--Puts 18 into It (144/(2^3))`
	`get 144 bitShiftRight 3  --(10010000 yields 10010)`

bitShiftLeft	Performs a bitwise shift to the left on a number by a specified number of bits. For example:
	`--Puts 32 into It (4*(2^3))`
	`get 4 bitShiftLeft 3  --(100 becomes 100000)`

ANEXO G: Elementos OpenScript para operaciones drag-and-drop[1]

Propiedades

Propiedad	Valor
defaultAllowDrag	Especifica cuando un objeto puede ser arrastrado.
DragImage	Imagen (ícono, cursor, o bitmap) desplegado durante una operación *drag* cuando el objeto bajo el cursor puede aceptar un *drop*.
noDropImage	Imagen (icono, cursor, o bitmap) desplegada durante una operación *drag-and-drop* cuando el objeto bajo el cursor no puede aceptar un *drop*.
defaultAllowDrop	Especificar cuando un objeto que es el destino de un *drop* puede aceptar el *drop*.

Comandos

Comando	Descripción
Drag	Inicia comportamiento *drag-and-drop* desde un *script* como si el usuario ha iniciado este con el *mouse*. El *drag* puede ser limitado dentro de un área rectangular específica.

Mensajes de eventos

Mensaje	Descripción
BeginDrag	Enviado a el objeto arrastrado cuando el evento *drag* empieza.
EndDrag	Enviado al objeto arrastrado cuando el botón del *mouse* es liberado; señala el final de un evento *drag*.
EnterDrop	Enviado a un objeto si su propiedad *defaultAllowDrop* es *true* o el manipulador *allowDrop* retorna *true* cuando el cursor del *mouse* ingresa a sus límites durante el modo *drag*.
LeaveDrop	Enviado al objeto destino cuando un objeto arrastrado es liberado sobre sus límites.

[1] Instructor 2004 Programming in OpenScript, página 254-256

stillOverDrop	Enviado al objeto destino continuamente durante un evento *drag* cuando este está ubicado debajo de la imagen arrastrada; similar a *buttonStillDown*.
AllowDrag	Enviado a el objeto fuente para determinar si el objeto puede ser arrastrado; si el manipulador *to get allowDrag* para este mensaje no es encontrado, ToolBook chequea la propiedad *defaultAllowDrag* de el objeto fuente.
allowDrop	Enviado al objeto destino para determinar si el objeto acepta el *drop*; si un manipulador *to get allowDrop* no es encontrado, ToolBook chequea la propiedad *defaultAllowDrop* del objeto.

Propiedades de objetos

Objeto	Propiedades drag-and-drop
Todos los objetos *draw* y *hotwords*	*defaulAllowDrag, defaultAllowDrop, dragImage, noDropImage*
Grupo	*dragImage, noDropImage*
Página	*DefaultAllowDrop*
Background	Ninguna
Libro	Ninguna
Viewer	Ninguna

El manipulador *to get* puede ayudar a determinar si un objeto puede ser arrastrado o si permite un drop.

Es posible añadir comportamiento drag-and-drop a un grupo de objetos, para esto se debe usar el comando *drag* de OpenScript, ya que como grupo no se tienen las propiedades *defaultAllowDrag* o *defaultAllowDrop.*

A continuación se muestra la secuencia de mensajes enviados por una operación drag-and-drop:

Orden en el cual mensajes *drag-and-drop* son enviados[1]

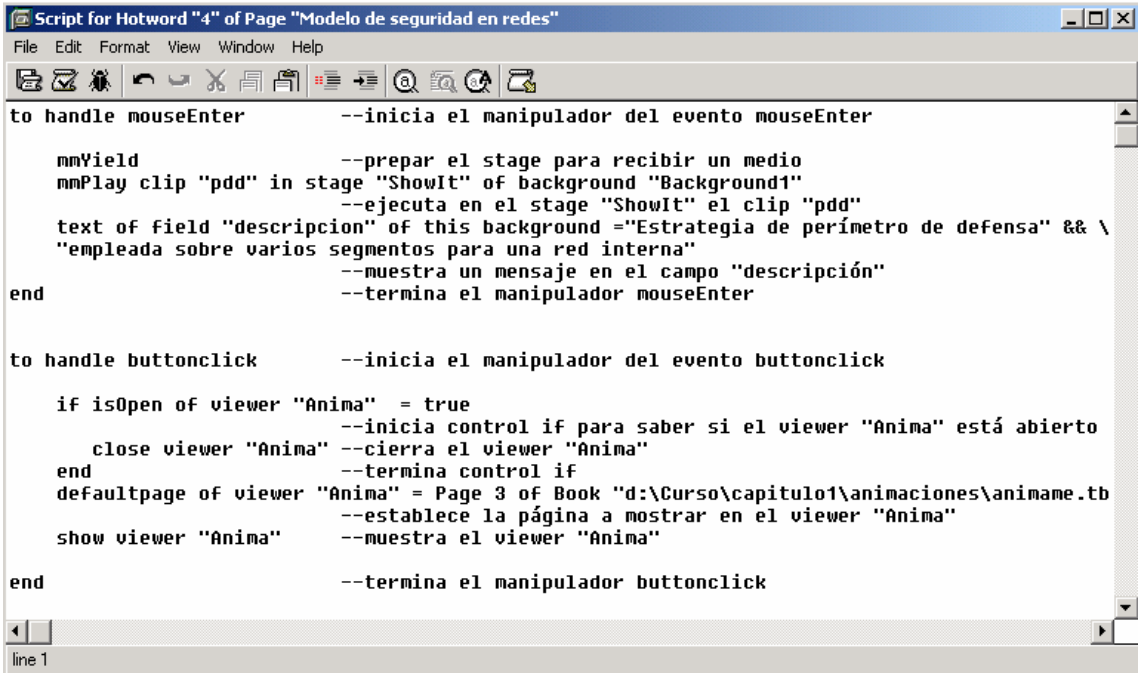
Evento	Mensaje	Destino
Usuario hace *click* a un objeto.	*AllowDrag*	Objeto fuente
Usuario ha hecho *click* el objeto con *true* como retorno de *allowDrag* o *defaultAllowDrag* establecido a *true.*	*BeginDrag*	Objeto fuente
El cursor del *mouse* ingresa a los límites del objeto destino.	*allowDrop,* entonces *enterDrop*	Objeto bajo la imagen arrastrada
El cursor del *mouse* está sobre el objeto destino.	*StillOverDrop*	Objeto bajo la imagen arrastrada
El cursor del *mouse* deja el objeto destino.	*LeaveDrop*	Objeto bajo la imagen arrastrada
El botón del *mouse* es liberado sobre el objeto destino.	*objectDropped* *endDrag*	Objeto destino Objeto fuente

[1] Instructor 2004 Programming in OpenScript, página 261

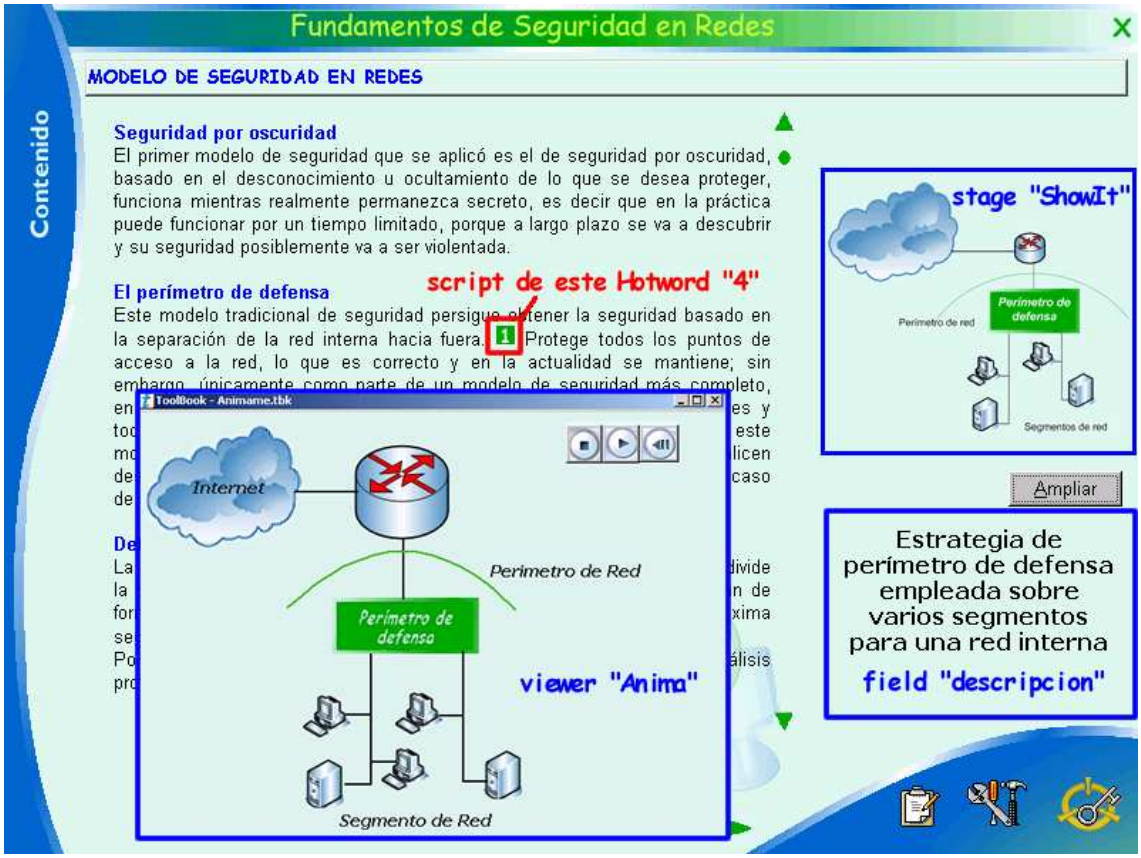
ANEXO H: Ejemplos de creación de medios

Creación de hiperenlaces con OpenScript

A continuación se muestra un *script* para el *hotword* “4”, donde programa el evento MouseEnter y ButtonClick para mostrar un a imagen en la zona de imagen y llamar a la ventana de animación.

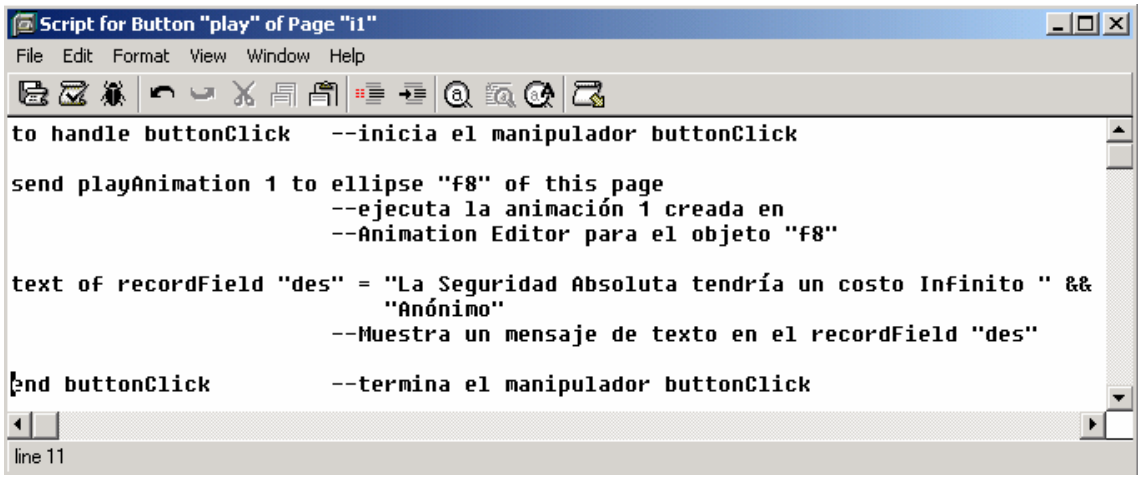


Elementos al que hace referencia el script del hotword “4”

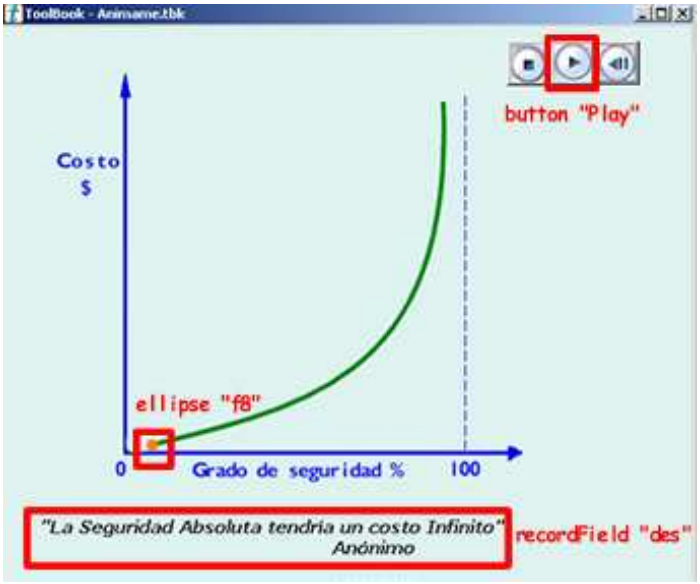


Creación de animaciones, utilizando Animation Editor y OpenScript

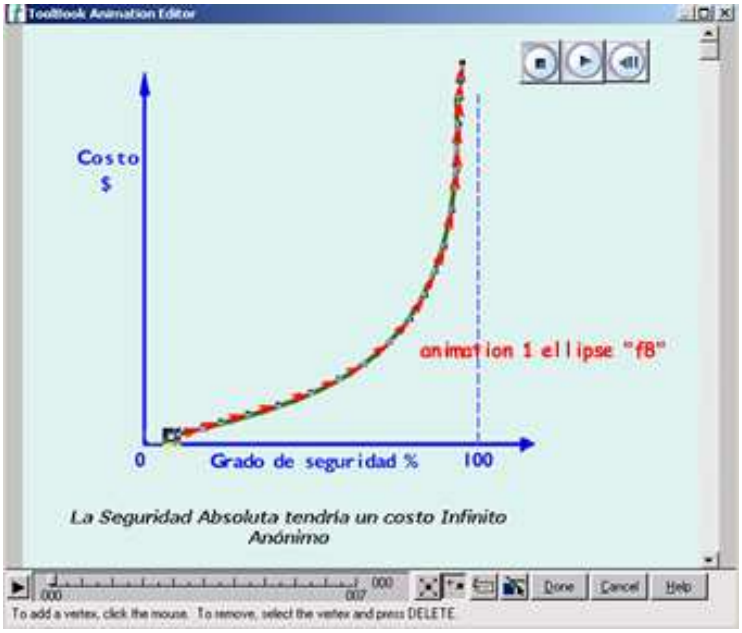
El siguiente *script* del botón “play”, maneja el evento *buttonClick* para ejecutar una animación creada en el *Animación Editor*.



Elementos a los que hace referencia el *script* del botón “play”

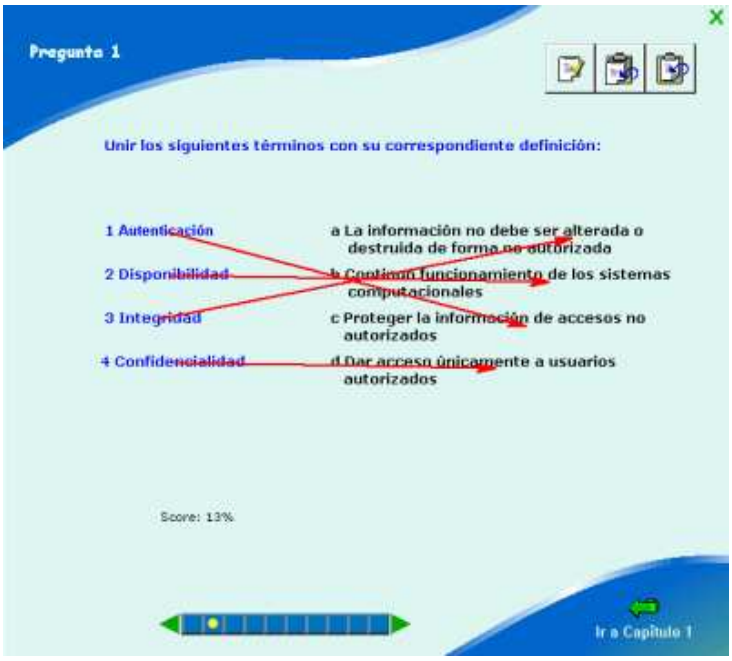


La animación del objeto ellipse “f8” es del tipo *Path Animation* y fue creada en el *Animation Editor*.

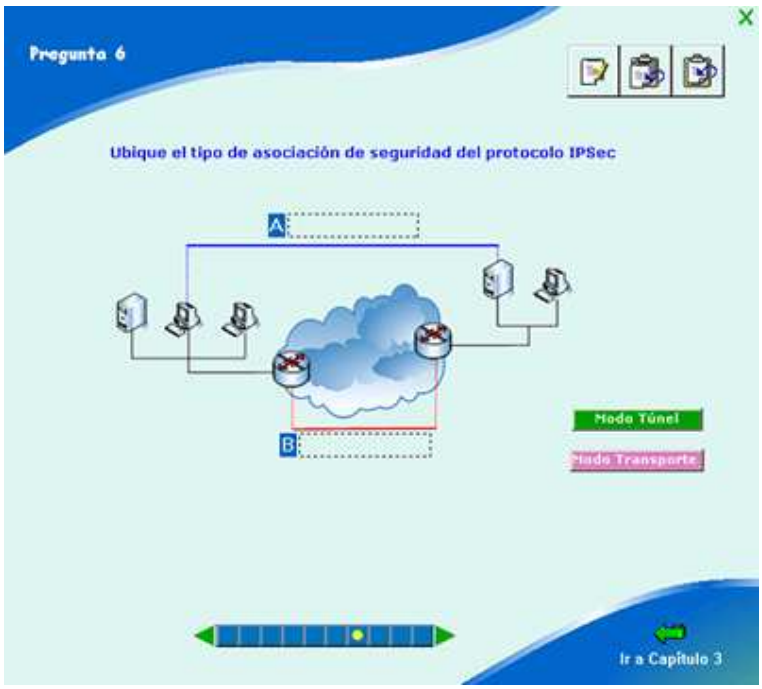


ANEXO I: Ejemplos de preguntas de las evaluaciones

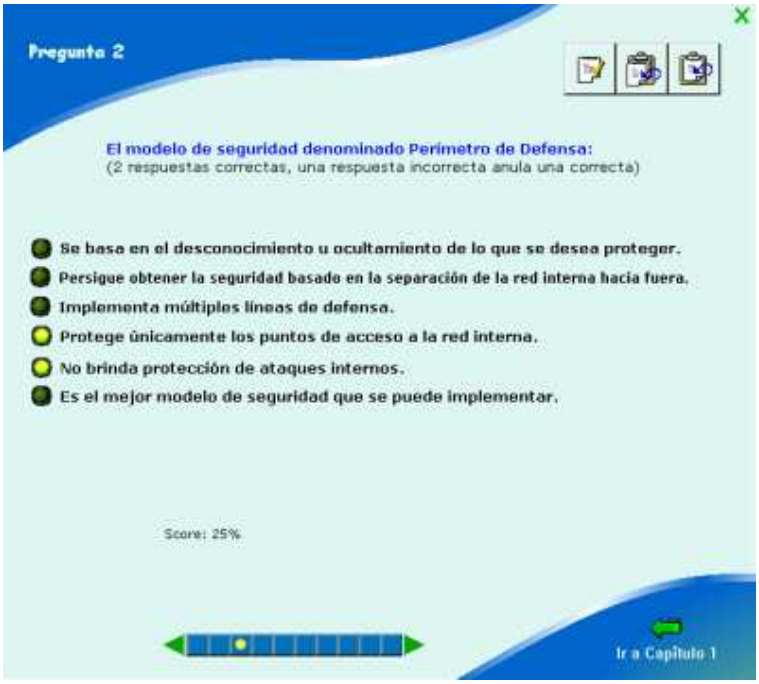
Emparejamiento



Arrastrar y soltar



Opción múltiple



Completar

ANEXO J: Encuesta de comprobación del uso eficaz

Por favor, marque las casillas que mejor representen su opinión sobre las características de la aplicación multimedia:
1 = malo, 2 = regular, 3 = bueno, 4 = muy bueno y 5 excelente

	Preguntas	**1**	**2**	**3**	**4**	**5**
1. Estructura de la aplicación						
a	Organización estructural: La distribución de los elementos estructurales de la aplicación (barras de navegación, contenido, botones, etc.)					
b	Densidad estructural: La cantidad de elementos estructurales que se utilizan en la aplicación					
c	Consistencia de la estructura: Uniformidad de la distribución de los elementos estructurales a lo largo de la aplicación					
2. Operación de la aplicación						
a	Navegabilidad: Recorrido que se hace por el contenido de la aplicación					
b	Interactividad: Relación mutua entre el usuario y la aplicación					
c	Accesibilidad: Manejo de las acciones que solicita la aplicación					
d	Sistema de indicación: Identificación de las figuras, tablas, hipertextos, zonas activas y el tipo de acción que se debe ejecutar					
e	Desempeño del sistema: La velocidad de funcionamiento de la aplicación considerando el tipo de tarea que se exige					
f	Fiabilidad del sistema: La operación de la aplicación					
g	Consistencia de la operación: Estándar en la ejecución de tareas (navegar por la aplicación, hacer *click* en botones, seleccionar operaciones, etc.) a lo largo de la aplicación					
3. Información al usuario						
a	Sistema de ayuda: Resolución de las dudas del usuario					
b	Feedback (retroalimentación): Información al usuario sobre las tareas en ejecución dentro de la aplicación					
c	Búsqueda de información: Localización de los datos que busca el usuario					
4. Apariencia						
a	La presentación del contenido (el tipo y tamaño de la fuente, el uso del color, disposición de los elementos según su significado, etc.)					
5. Intuición						
	Forma intuitiva de aprender los procedimientos de navegación por la aplicación o ejecución de tarea asignadas					

6. Contenido						
a	Organización del contenido: La distribución del contenido de la aplicación (textos, imágenes, evaluaciones, etc.)					
b	Densidad del contenido: Cantidad de información que se presenta en la aplicación					
c	Fiabilidad del contenido: Confiabilidad de la información que se presenta en la aplicación					
d	Comprensión del contenido: Facilidad de entendimiento de la información que se presenta en la aplicación					

7. Experiencia del usuario						
	Experiencia para manejar la aplicación					

1 = necesaria, 2= considerable, 3 = poca, 4 = mínima y 5 = no necesaria

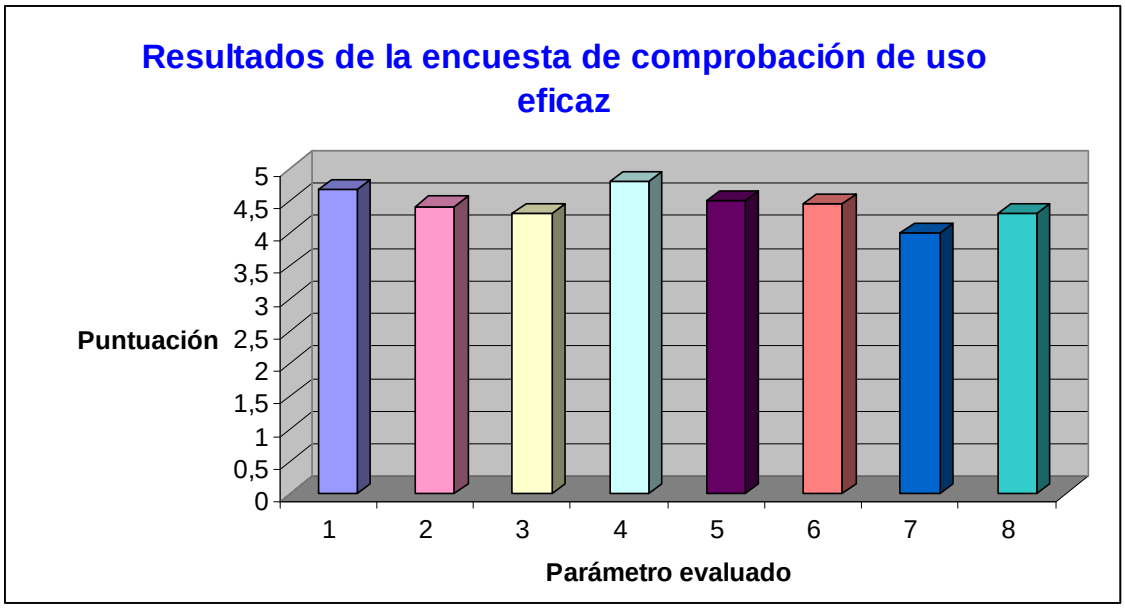
8. Opinión general sobre la aplicación					

1 = malo, 2 = regular, 3 = bueno, 4 = muy bueno y 5 = excelente

El curso desarrollado esta enfocado a ser una herramienta complementaria para los estudiantes de la materia de Seguridad en Redes de la carrera de Ingeniería Electrónica de Redes de Información de la EPN.

La encuesta fue realizada a un grupo de 10 personas, grupo considerado representativo tomando en cuenta un universo promedio de 20 estudiantes que se inscriben en la materia semestralmente.

De la encuesta se obtuvieron los siguientes resultados correspondientes a cada grupo de parámetros evaluados, en promedio la aplicación final tiene una valoración que va entre 4 a 4,5 lo que corresponde a un uso eficaz muy bueno - excelente, por lo que se puede concluir que los objetivos planteados han sido cumplidos.



1. Estructura de la aplicación
2. Operación de la aplicación
3. Información al usuario
4. Apariencia
5. Intuición
6. Contenido
7. Experiencia del usuario
8. Opinión general sobre la aplicación

REFERENCIAS BIBLIOGRÁFICAS

1. SUMTOTAL SYSTEMS, ToolBook Instructor 2004 User Guide. Click2learn. Washington. 2004

2. SUMTOTAL SYSTEMS, ToolBook Instructor 2004 Programming in OpenScript. Click2learn. Washington. 2004

3. SUMTOTAL SYSTEMS, ToolBook Instructor 2004. http://www.sumtotalsystem.com

4. TOM L. Hall, The Actions Editor, HTTP Post, and Simulations in ToolBook Instructor. TCC Publishing Inc. 2004

5. SUMTOTAL SYSTEMS, ToolBook Instructor 2004 Online Help

6. AVILA, Nelson, Folleto de la Materia de Seguridad en Redes, Ingeniería Electrónica y Redes de Información. Quito, Marzo – Agosto 2004

7. CANAVAN, John, Fundamentals of Network Security. Artech House. Boston - London. 2001

8. HASSLER, Vesna, Security Fundamentals for E-Commerce. Artech House. Boston - London 2001

9. COLE, Eric; KRUTZ, Ronald; CONLEY James, Network Security Bible. Wiley Publishing. Indiana. 2005

10. CISCO SYSTEMS, Fundamentals Network Security v1.1 CBT. 2003

11. CISCO SYSTEMS, Cisco Certified Network Associate CCNA v3.1 CBT. 2003

12. BARMAN, Scott, Writing Information Security Policies. 2004

13. BORGHELLO, Cristian, Tesis de Licenciatura en Sistemas – Universidad Tecnológica Nacional - Seguridad Informática sus Implicancias e Implementación. Argentina. 2001

14. McNAB, Chris, Network Security Assessment. O'Reilly. United States of America. 2004

15. PHALTANKAR K., Practical Guide for Implementing Secure Intranets and Extranets. Artech House. Boston. 2000

16. LEE, William; OWENS, Diana, Multimedia Based Instructional Design. Pfeiffer. 2da Edición. Estados Unidos. 2004

17. COLMENAR, Antonio; CASTRO, Manuel-Alonso; SAN CRISTOBAL, Elio; PÉREZ, Vicente; LOSADA, Pablo; AEDO, Ignacio; DÍAZ, Paloma, Diseño y desarrollo multimedia. Herramientas de autor. RA-MA. España. 2005

18. PERURENA, Lilliam, Sistema de herramientas para la construcción y administración de cursos multimedia. 2003 http://www.sav.us.es/pixelbit/articulos/n21/n21art/art2105.htm

19. CHAPMAN, BL., Accelerating the Design Process: A Tool for Instructional Designers, Allen Communication, A Times Mirror Company. USA. 1998

20. PORTALZONDA, Método abreviado de evaluación de sitios Web. 2006 www.portalzonda.com.ar/americosirvente/ASirvente%20Labor/MeDHiME/DOCUMNTOS/Evaluación%20web.mht

21. SANTISO, María; GONZÁLEZ, Begoña, Diseño multimedia en e-learning para el ámbito universitario. 2005 www.nosolousabilidad.com/articulos/multimedia_elearning.htm

22. SIRVENTE, Francisco, MEDHIME, un puente de comunicación entre programadores y docentes para producir materiales educativos navegables. 2004 www.portalzonda.com.ar/medhime

23. MARQUÉS, Pere, Diseño y desarrollo multimedia. 2006 dewey.uab.es/pmarques/ud.htm

24. BORGES, Hernane. Análisis experimental de los criterios de evaluación de usabilidad de aplicaciones multimedia en entornos de educación y formación a distancia www.tdx.cesca.es/TESIS_UPC/AVAILABLE/ TDX-0716102-102210//12ApendiceA.pdf

OTRAS:

- AUSTRALIAN TOOLKOOK USER GROUP, ToolBook Tips & Techniques. 2006 www.atug.com/toolbookmag/OUTPUT/TOCONV21.htm

- WELLER, Wellwe´s ToolBook Samples. 2006 www.mweller.com/toolbook/default.htm

- CHEAH, ToolBook Developer's Network (TBDN). 2006 www.webcbt.ch/toolbook/

- UIVERSIDAD PEDAGÓGICA RAÚL GÓMEZ GARCÍA, ToolBook. 2006 www.edusol.rimed.cu/