UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

FACULTAD DE INGENIERÍAS

CARRERA DE INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del Título de:

Ingeniera en Sistemas

TITULO DEL TEMA:

"Kiosko Multimedia para Consulta y Emisión de Certificados Académicos de la Universidad Politécnica Salesiana"

AUTORA:

NATALY MONSERRATH MOLINA TOLEDO

DIRECTORA:

Ing. Bertha Katerine Tacuri Capelo.

CUENCA – ECUADOR

CERTIFICACIÓN

Certifico que el presente trabajo de tesis previo a la obtención del título de Ingeniero de Sistemas fue desarrollado por Nataly Monserrath Molina Toledo bajo mi supervisión.

……..……..

Ing. Bertha Katerine Tacuri Capelo

DIRECTORA DE TESIS

DECLARACIÓN

Yo, Nataly Monserrath Molina Toledo, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado por ningún grado o calificación profesional y que he consultado las referencias bibliográficas que se incluyen en este documento

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la Normativa Vigente.

……

Nataly Monserrath Molina Toledo

AUTORA

Dedicatoria

Lo más importante para un ser humano va mucho
más allá de los bienes materiales e intelectuales
que bombardean a la sociedad actual
los recuerdos que cada uno tiene del transcurso de su vida
se han forjado y alimentado de acuerdo al contexto en
el que han crecido, el país, la ciudad, el barrio,
el lugar donde realizaron sus estudios,
el clima, las costumbres, la alimentación,
y muchos más factores que marcan la personalidad
y la vida misma del ser humano;
todos estos factores intervienen en la formación completa de las personas.
Pero lo realmente importante,
más que cualquiera de los factores antes mencionados,
tendrían validez, sin la presencia de otros seres humanos,
sin poder compartir estas costumbres, este clima, esta
bella ciudad, este hermoso país, las aulas de clase…
nada tendría sentido si no existieran esas personas, las
que están con nosotros compartiendo su tiempo
luchando junto a nosotros, sufriendo con nosotros,
planteando soluciones, buscando recursos,
en fin… un sin número de acciones
todas ellas con el único fin de vernos felices
ningún logro fuera tan gratificante, sin tener a nadie
que se sienta orgulloso de nosotros, sin tener
a quien dar un buen ejemplo,
sin tener a nadie que nos brinde un abrazo de felicitación
por todas estas razones me siento completamente
feliz de poder dedicar un logro tan importante como éste a MI FAMILIA
mis padres: FRANCISCO y NUBE
y mis queridos hermanos: PACO, LISSETH y DENISSE
por ser parte de mi vida…

Agradecimiento

Nunca es demasiado tarde
Y tampoco es demasiado pronto
para mostrar gratitud,
Siempre es el momento adecuado;
No hay un método correcto
Ni un medio apropiado,
para decir gracias;
no existen restricciones de edad,
sexo, religión…,
no hay motivo alguno que impida
que las personas manifiesten sentimientos de gratitud.
Sin embargo, muchas veces lo olvidamos.

Por esta razón,
Y para no olvidar a nadie
Quiero agradecer inmensamente,
a ***todas*** *las personas que física o espiritualmente*
han colaborado en todo este proceso,
que me han contagiado su entusiasmo y buenas energías,
A todas las personas que han creído en mí en todo momento
Y me han transmitido seguridad y confianza,
Y me han mostrado su apoyo desinteresado,
A todos aquellos, que me han proporcionado los medios físicos
Necesarios para la realización de este proyecto,
Y de cualquier forma
Han formado parte de este proceso
Y han contribuido para su finalización.

Por su apoyo, GRACIAS!

ÍNDICE DE CONTENIDO

INTRODUCCIÓN 9
OBJETIVOS 10
CAPITULO I 11
DISEÑO 11
1.1 REQUERIMIENTOS Y ALCANCE 12
1.2 CONSIDERACIONES DE SEGURIDAD 16
1.3 DIAGRAMAS DE CASOS DE USO 19
1.4 DATOS DE ENTRADA Y SALIDA 26
1.5 DIAGRAMAS DE CLASES 31
1.6 DIAGRAMA DE SECUENCIAS 33
1.7 DIAGRAMAS DE ESTADOS 34
1.8 DIAGRAMA DE COLABORACIÓN 36
1.9 DIAGRAMAS DE ACTIVIDAD 37
1.10 DIAGRAMAS DE PROCESOS 41
1.11 DIAGRAMA ENTIDAD RELACIÓN DE BASE DE DATOS 43
1.12 DICCIONARIO DE DATOS 44
1.13 ACCESO A TABLAS EXTERNAS 50
CAPITULO II 51
PHP 51
2.1. INTRODUCCIÓN 52
2.2. LA SINTAXIS DE PHP 54
2.3. VARIABLES EN PHP 55
2.3.1. Variables Objeto 56
2.3.2. Variables asignadas por referencia 57
2.3.3. Cambio del Tipo de las Variables en PHP 58
2.3.4. Cambio Forzado de Tipos de Variable 58
2.3.5. Los cambios forzados permitidos 59
2.4. VARIABLES DE SISTEMA EN PHP 59
2.5. VARIABLES SUPERGLOBALES 61
2.6. TABLAS O ARRAYS EN PHP 62
2.7. TRABAJO CON TABLAS O ARRAYS EN PHP 66

2.7.1. La función array_slice() 66
2.7.2. La función array_shift() 67
2.7.3. La función unset() 68
2.7.4. Aumentar el tamaño de un Arraycon la función array_push() 68
2.7.5. La Función array_merge() 69
2.8. CADENAS 70
2.9. CONTROL DEL FLUJO EN PHP CONDICIÓN IF 73
2.10. LOS CONTROLADORES DE FLUJO O BUCLES 75
2.10.1. Bucle While 75
2.10.2. Bucle for 77
2.10.3. Bucle foreach 78
2.10.4. Sentencias de Salida de Bucle break y continue 79
2.11. FUNCIONES EN PHP 79
2.12. PASO DE PARÁMETROS A FUNCIONES 83
2.12.1. Paso de Parámetros por Valor 84
2.12.2. Paso de Parámetros por Referencia 84
2.12.3. Paso de Parámetros por Defecto 85
2.13. PASO DE VARIABLES POR LA URL 85
2.14. PROCESAMIENTO DE VARIABLES DE FORMULARIO 88
2.15. AUTO LLAMADA DE PÁGINAS 91
2.15.1. Autollamadas para un Formulario 92
2.15.2. Autollamadas para Paso de Variables por URL 92
2.16. PROGRAMACIÓN ORIENTADA A OBJETOS 94
2.16.1. Las clases 94
2.16.2. Utilizar la clase 95
2.16.3. La variable $this 96
2.16.4. Constructores en PHP 96
2.16.5. Herencia en PHP 97
2.16.6. Redefinir métodos en clases extendidas 98
2.17. CONEXIÓN A LA BASE DE DATOS ORACLE 100
2.17.1. Abrir una Conexión Oracle 100
2.17.2. Definir la Instrucción SQL 100
2.17.3. Ejecutar la instrucción SQL 101
2.17.4. Acceder a los Registros de una Consulta 101
2.17.5. Recuperar el valor de cada Campo de la Consulta 102

2.17.6. Cerrar la Conexión 103
CAPITULO III 104
ADOBE FLASH CS4 104
3.1 TIPOS DE GRÁFICOS MANEJADOS: 105
3.1.1 Vectoriales 105
3.1.2 Mapa de bits 106
3.2 ENTORNO DE ADOBE FLASH 106
3.2.1 La Línea de Tiempo 107
3.2.2 Gestor de Capas 108
3.2.3 El Escenario 109
3.2.4 Panel de Herramientas 110
3.2.5 Herramientas Avanzadas 111
3.2.6 Herramientas Opcionales 112
3.2.7 Los Objetos de Iniciación Gráficos 114
3.2.8 Panel de Información 114
3.2.9 Los Grupos 116
3.3 LAS CAPAS 117
3.3.1 Trabajar con Capas 119
3.3.2 Opciones Avanzadas 120
3.3.3 Reorganizar Capas 122
3.3.4 Tipos de Capas 122
3.4 LOS SÍMBOLOS 123
3.4.1 Como crear un Símbolo 123
3.4.2 Las Bibliotecas 124
3.4.3 Diferencia entre Símbolo e Instancia 125
3.4.4 Modificar una instancia 126
3.4.5 Panel de Propiedades de una Instancia 126
3.5 GRÁFICOS 130
3.5.1 Tipos de Gráficos 131
3.5.2 Creando Gráficos y Comprobando sus Propiedades 132
3.5.3 Introducir un mapa de bits 133
3.5.4 Introducir un archivo vectorial 134
3.5.5 Exportar un archivo flash como mapa de bits 135
3.5.6 Exportar un objeto flash como animación 135
3.6 CLIPS DE PELÍCULA 136

3.6.1 Comprobar las propiedades de un Clip 137
3.6.2 Como crear un Clip de Película 139
3.6.3 Importar y Exportar Movie Clips de la Biblioteca 140
3.7 BOTONES 141
3.7.1 Creación de un Botón 142
3.7.2 Formas en los Botones 143
3.7.3 Incluir un clip en un Botón 144
3.7.4 Incluir Bitmaps en Botones 144
3.7.5 Acciones en Botones 145
3.8 ANIMACIONES DE MOVIMIENTO 146
3.8.1 La interpolación de movimiento 147
3.8.2 Editor de Movimiento 148
3.8.3 Interpolación Clásica 150
3.8.4 Diferencias entre Interpolación Clásica y de movimiento 151
3.8.5 Animación de Texto 152
3.8.6 Animación de Líneas 153
3.8.7 Efecto Brillo 153
3.8.8 Efecto Tinta 153
3.8.9 Efecto Alfa 154
3.9 GENERACIÓN Y PUBLICACIÓN DE PELÍCULAS 154
CAPITULO IV 158
ACTIONSCRIPT 3 158
4.1 EL PANEL DE ACCIONES 160
4.2 DESCRIPCIÓN DE LOS TIPOS DE DATOS 162
4.2.1 Tipo de datos Boolean 162
4.2.2 Tipo de datos int 162
4.2.3 Tipo de datos Null 163
4.2.4 Tipo de datos Number 163
4.3 CONVERSIONES DE TIPOS 165
4.4 SINTAXIS 166
4.2.1 Operadores Aritméticos 170
4.2.2 Operadores de Asignación 171
4.2.3 Operadores de Comparación 171
4.2.4 Operadores lógicos. 172
4.5 CONDICIONALES 172

4.6 BUCLES 174
4.6.1 Bucle for 174
4.6.2 Bucle for..in 174
4.6.3 Bucle for each..in 175
4.6.4 Bucle while 176
4.6.5 Bucle do..while 176
4.7 PROGRAMACIÓN ORIENTADA A OBJETOS 176
4.8 GESTIÓN DE EVENTOS 181
4.9 LOS OBJETOS DESDE EL CÓDIGO 185
4.9.1 Objeto "Button" (Botón) 186
4.9.2 Objeto "MovieClip" (Clip de Película) 186
4.9.3 Objeto "DisplayObject" (Objeto de visualización) 187
4.9.4 Objeto "Mouse" (Ratón) 187
4.9.5 Objetos "Loader" y "URLLoader" 187
4.10 EVENTOS EN BOTONES 187
4.11 NAVEGACIÓN MEDIANTE BOTONES 188
4.12 CONTENEDORES Y LISTAS DE VISUALIZACIÓN 190
4.13 FORMULARIOS 192
4.14 CARGA EXTERNA DE OBJETOS DE VISUALIZACIÓN 198
4.16.1 Supervisión del progreso de carga 199
4.16.2 Especificación del contexto de carga 200
CAPÍTULO V 203
AMFPHP 203
5.1 WEB SERVICES 204
5.1.1. Beneficios de un Web Service 205
5.1.2. Componentes de un Web Service 206
5.1.3. XML (Extensible Markup Language) 206
5.1.4. WSDL (Web Services Description Language) 207
5.1.5. SOAP (Simple Object Access Protocol) 208
5.2 PUBLICAR UN MÉTODO PHP COMO WEB SERVICE MEDIANTE AMFPHP 210
5.3 ACCEDER A UN MÉTODO PHP DESDE ACTIONSCRIPT 3 214
CONCLUSIONES 218
RECOMENDACIONES 220
BIBLIOGRAFÍA: 221

ANEXOS 223

ÍNDICE DE FIGURAS

Figura 2.2.1: Ejemplo de Script PHP 54
Figura 3.1.1 Entorno de Adobe Flash CS4 107
Figura 4.1.1: Editor de Código ActionScript en Flash 160
Figura 4.1.2: Visor de Errores de Sintaxis 161
Figura 4.1.3: Editor de Código 162
Figura 4.4.6.1: Listado de Palabras Clave en ActionScript 3 169
Figura 4.9.1: Estructura Jerárquica de los Objetos de Visualización 186
Figura 4.11.1: Diseño de una Interfaz con Botones de Navegación 188
Figura 4.11.2: Diseño de una Interfaz con Botones de Navegación 189
Figura 4.13.1: Paleta de Controles de Formulario 192
Figura 4.13.1: Panel de Propiedades de los Controles de Formulario 193
Figura 4.13.2: Inspector de Componentes 194
Figura 4.13.3: Editor de Valores de ComboBox 196
Figura 4.13.4: Inspector de Componentes (propiedades de un List) 197
Figura 4.16.1: Proceso de Carga de un Archivo Externo 199
Figura 5.1: Arquitectura de Un Servicio Web 205
Figura 5.2.1: Arquitectura de Un Servicio Web 211
Figura 5.2.1: Explorador de Servicios Web de AMFPHP 214

ÍNDICE DE TABLAS

Tabla 2.3.1: Variables Numéricos en PHP 55
Tabla 2.3.2: Variables Alfanuméricas en PHP 56
Tabla 2.3.3: Variables Tipo Tabla en PHP 56
Tabla 2.4.1: Variables de Sistema en PHP..... 60
Tabla 2.6.1: Funciones Importantes de la Clase Array 65
Tabla 2.8.1: Caracteres de Escape Especiales en PHP..... 72
Tabla 2.8.2: Caracteres de Escape Comunes 72
Tabla 4.4.8.1 Operadores Aritméticos 170
Tabla 4.4.8.2 Operadores de Asignación 171
Tabla 4.4.8.2 Operadores de Comparación..... 171
Tabla 4.4.8.2 Operadores de Lógicos 172
Tabla 4.4.8.2 Atributos de Clase..... 178

INTRODUCCIÓN

A pesar de que la Universidad Politécnica Salesiana cuenta con implementos tecnológicos muy avanzados (faltos en otras universidades), infraestructura e instalaciones aptas para los estudiantes y métodos de enseñanza actuales, aún existen algunos procesos que no son del todo digitalizados, es decir, existen ciertos trámites que se deben realizar de manera manual, como es el caso de la emisión de certificados que causan un poco de tedio a los estudiantes, debido al respectivo trámite que consigo lleva el hecho de solicitar un certificado (algunos no saben cómo realizar la solicitud para el certificado respectivo).

Con el desarrollo de este sistema se podrá dar un paso más a la digitalización de procesos en la Universidad Politécnica Salesiana, lo que permitirá integrar la información con el objetivo de satisfacer las necesidades de la Universidad y a su vez servirá como ejemplo para los estudiantes de la misma, para continuar con este tipo de proyectos originales, innovadores y sobre todo con un carácter utilitario que permiten facilitar los trámites, generando ahorro de recursos sobre todo en la medida del tiempo.

Este proyecto consiste en desarrollar una herramienta multimedia útil para la vida académica de estudiantes. Mediante la generación de un software, se permitirá tener acceso a información selecta para realizar ciertos procesos de trámite, esta información resulta de un estudio apropiado de las necesidades de cada trámite. A través de una interfaz amigable y fácil de usar, se presentará un menú y se podrá elegir una categoría según sea el caso

OBJETIVOS

Objetivo General:

- Diseñar e Implementar un Software Interactivo y Práctico en el que se pueda integrar información de estudiantes, y que permita a los mismos realizar procesos en un período de tiempo más corto que en el que usualmente lo realizan.

Objetivos Específicos:

- Investigar cuáles son los certificados que pueden solicitar los estudiantes de la Universidad Politécnica Salesiana y cómo se realiza el proceso de emisión de los mismos, para lograr que estos trámites que son necesarios en la vida académica, se realicen en un tiempo menor del que usualmente tardan.
- Realizar un análisis sobre el tipo de certificados que pueden ser implementados y con esta información poder implementar una ayuda para agilizar estos procesos.
- Investigar cuáles son las solicitudes que pueden realizar los estudiantes de la Universidad Politécnica, y analizar cuáles de ellas son viables para ser implementadas, y se realicen en un tiempo menor del que usualmente tardan.
- Investigar datos sobre los "Kioscos Multimedia", para poder identificar las ventajas y desventajas de su implementación, para estar alerta a cualquier complicación que pueda surgir en el desarrollo del proyecto.
- Realizar una investigación sobre herramientas de diseño gráfico (Adobe flash). para crear un Software Dinámico, Amigable, y Agradable a la vista, de fácil acceso y que no cause tedio al momento de utilizarlo.
- Conseguir que el Software Desarrollado ayude a satisfacer las necesidades identificadas como resultado de las investigaciones realizadas sobre los futuros usuarios del sistema.
- Evitar la aglomeración de gente en la secretaría de la Universidad.

CAPITULO I

DISEÑO

1.1 REQUERIMIENTOS Y ALCANCE

En cuanto a la parte de Certificados

- Brindará a los estudiantes la posibilidad de generar cualquiera de los siguientes certificados:

    - Certificado de Inscripción
    - Certificado de Matrícula
    - Certificado de Asistencia a Clases

Estos 3 certificados fueron selectos luego de un diálogo con los responsables de Secretaría General de la Universidad Politécnica Salesiana, en el cual se llegó a la conclusión de que son de factible generación, en el análisis también se pudo notar que los demás certificados obligatoriamente tendrán que seguirse realizando a mano debido a que requieren gran cantidad de información que no se encuentra disponible en la base de datos del Sistema Nacional Académico. Por esta razón es imposible su generación automatizada, este es el caso por ejemplo del certificado de Haber Aprobado el 80% de las Materias del Pensum, que implica varios datos no disponibles, esto puede ser verificado si se pregunta en Secretaría el proceso de como éste se lleva a cabo.

El módulo de Certificados dentro del Kiosko Multimedia pondrá a disposición las siguientes funcionalidades:

- El acceso al Kiosko estará definido por el ingreso de la cédula del estudiante, que será verificada por sede y campus; de ser correcta, se concederá el acceso.

- El estudiante podrá seleccionar su carrera de un listado en el que se exponen todas las carreras que el estudiante haya cursado o esté cursando. La carrera que seleccione será la cual defina los datos del certificado.

- En cuanto al pago del valor o costo del certificado, este se lo realizará de la forma hasta ahora habitual, es decir mediante el ingreso de la factura del respectivo derecho de certificación que deberá haberse pagado previamente.

- Existirá un navegador de periodos lectivos por los cuales haya cursado el alumno la carrera (seleccionada previamente), y a partir del cual podrá generar el certificado.

- Se le permitirá hacer una vista previa del certificado antes de imprimirlo.

- El Sistema tendrá la capacidad de comprobar la validez de la factura, que esté vigente, que incluya el derecho de certificación respectivo, y que la fecha de emisión no sea superior a la fecha de inauguración del Kiosko Multimedia.

- Cada vez que un estudiante solicita un certificado al Kiosko, este guardará un histórico con hora y fecha de cuando se haya emitido el mismo, para poder llevar un control interno para validación de re-uso de facturas.

- En caso de impresión errónea, se ha creado una opción mediante la cual se podrá realizar una reimpresión del certificado. Es requerida una contraseña de reimpresión, la cual únicamente las secretarias del campus conocerán, y que por ende deberán ingresar únicamente ellas. Esta contraseña es modificable desde la interfaz de administración del Kiosko.

Para la generación del PDF del certificado se hará un llamado a uno de los 3 respectivos reportes del Sistema Nacional Académico, y se descargará el archivo para ser previsualizado o impreso.

En cuanto a la parte de Solicitudes

Se brindará al estudiante la posibilidad de generar cualquier tipo de solicitud que él desee.

El módulo de Solicitudes dentro del Kiosko Multimedia pondrá a disposición las siguientes funcionalidades:

- El acceso al Kiosko está definido por el ingreso de la cédula del estudiante, que será verificada y validada por sede y por campus, de ser correcta, se concederá el acceso.

- El estudiante podrá seleccionar su carrera de un listado en el que se exponen todas las carreras que haya cursado o esté cursando.

- Se le permitirá al estudiante seleccionar la carrera a la que dirige la solicitud, para el caso en que un estudiante desee realizar homologaciones.

- Se tendrá un menú con las solicitudes disponibles para que el estudiante elija la que requiera, y proceda a llenar sus campos.

- Mediante una interfaz gráfica con teclado táctil al estudiante se le permite digitar los valores editables de cada campo de la solicitud, el teclado cuenta con las teclas de un teclado normal, permitiendo realizar todos los cambios necesarios en la solicitud antes de la impresión.

- También se contará con la opción de VISTA PREVIA que permite comprobar que los datos de la solicitud son correctos para luego proceder a la impresión.

- Antes de imprimir se validará el correcto ingreso de los campos de la solicitud y se procederá a solicitar el pago según la tarifa respectiva de la hoja valorada requerida. El pago de las solicitudes se realizará con la

ayuda de un monedero (la funcionalidad del monedero y su implementación, no está contemplada dentro del desarrollo de esta tesis)

- En caso de impresión errónea, se ha creado una opción mediante la cual se podrá realizar una reimpresión de la solicitud al igual que en el caso de los certificados, es requerida una contraseña de reimpresión, la cual únicamente las secretarias del campus conocerán, y que por ende deberán ingresar únicamente ellas. Esta contraseña es modificable desde la interfaz de administración del Kiosko.

En cuanto al Módulo de Administración del Kiosko Multimedia

Se tendrán las siguientes funcionalidades:

Para el caso de Certificados:

- Se dará la posibilidad de listar los certificados emitidos hasta el momento en el Kiosko, listado que podrá ser filtrado por fechas, sede y campus.

- Se podrá cambiar la contraseña de reimpresión de los certificados.

- Se podrá elegir la impresora destinada a la impresión de los certificados así como la bandeja de la misma que contiene el formato de las hojas que éstos requieren (además de las hojas firmadas).

Para el caso de las Solicitudes:

- Se tendrá una interfaz completa tanto de edición como de creación de solicitudes.

- Como se dijo en el punto anterior se podrá crear solicitudes, habrá una zona para definir la redacción de la misma, y otra con un listado de campos válidos de solicitud, dentro de estos campos estarán los editables por el estudiante, es

decir los que el estudiante digita al momento de realizar su solicitud, y los campos fijos cuyo valor se recuperará según la carrera, campus y sede.

- Se podrá eliminar las solicitudes que se consideren como descontinuadas.
- Se podrá modificar las solicitudes, según se requiera y en cualquier momento.
- Se podrá crear o eliminar campos de solicitud, es decir se podrán definir nuevos campos pero únicamente campos editables por el estudiante, como por ejemplo justificación, nombre materia, etc.
- El módulo de administración permitirá configurar la Sede en la que está ubicado así como también el Campus. Esto será necesario para llenar los campos respectivos en las solicitudes.
- Se permitirá elegir la impresora destinada a la impresión de los certificados así como la bandeja de la misma que contiene las hojas valoradas.
- Se podrá cambiar la contraseña de reimpresión de las solicitudes.

En el caso de las solicitudes no habrá ningún problema si se desea crear o adicionar más de ellas, debido a que no requieren información de difícil acceso desde el Sistema Nacional Académico.

1.2 CONSIDERACIONES DE SEGURIDAD

El certificado obtenido, para que tenga validez legal necesita tener las firmas originales que certifiquen que el documento es válido, y esto ha dado paso a algunas interrogantes y posibles limitantes.

Con el desarrollo del kiosco se tratará de que los procesos de emisión de certificados se realicen con una intervención humana mínima; estos percances han hecho buscar alternativas para que las personas que intervienen en el proceso

(Secretarias) no sientan amenazada su integridad, a la vez que el sistema cumpla con la característica de operativo. Debido a esto, se señalan algunas consideraciones de seguridad:

Dentro de la propuesta inicial se pretende que los certificados puedan ser impresos y que la firma que éste requiera ya iba a encontrarse en el papel a la hora de imprimir, pero en secretaría, no están de acuerdo con firmar hojas en blanco, puesto que existen riesgos como:

- Hurto las hojas de la impresora y libre uso de las mismas.
- Al momento de imprimir salgan hojas pegadas y el estudiante obtiene una hoja extra en blanco y firmada.

Para poder solucionar estos problemas se ha realizado una investigación sobre el tema y se ha planteado soluciones que se describen a continuación:

Para el problema de hurto:

- El kiosco estará ubicado en un lugar interno de la universidad, posiblemente dentro del edificio principal frente a los cubículos de secretaría, es decir no estará a la intemperie en el patio o en un pasillo al que se accede todo el tiempo; estará disponible en horas de oficina nada más por lo que estará en constante supervisión.

- El enclosure, o contenedor del monitor y la impresora debido a su función de proteger equipos que son sensibles al clima y a la manipulación de las personas, siempre son de un material resistente como madera o aluminio.

- Además la impresora estará ubicada en el interior de secretaría y el estudiante tendrá acceso únicamente al monitor; cuando el estudiante imprima su certificado este se obtendrá de las manos de la secretaria.

Hojas pegadas:

Para evitar que al momento de imprimir el certificado se vayan pegadas dos hojas es decir que una extra vaya firmada en blanco:

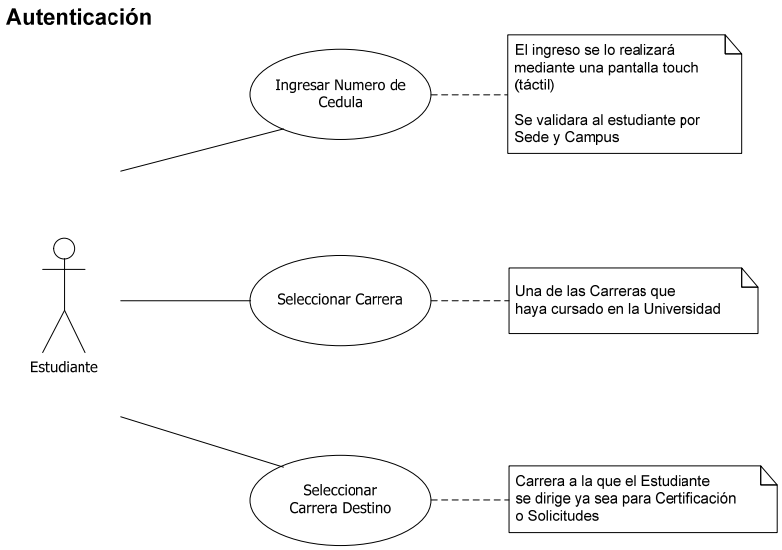
- El papel a usar será un papel un poco más grueso que el normal (75gr.), puede ser de 120gr. o un poco más, con esto el riesgo de que se peguen disminuye.
- Las hojas que podrían estar pegadas, al momento de realizar las firmas, se va a separar obligatoriamente.
- Las hojas estarán membretadas y numeradas y dentro del sistema se asociará el certificado impreso con la persona que lo solicitó, por ejemplo, en el caso que vayan hojas pegadas, se tendrá conocimiento de la persona a la que se le atribuyó el certificado número 15 y si el siguiente certificado que se imprime está en la hoja 17, se conoce que hubieron hojas pegadas y que fue el estudiante al que adquirió la solicitud 15 el que obtuvo la hoja extra, en este caso previamente los estudiantes serán notificados de que si ocurre un caso de hojas pegadas, éste debe entregarlo a secretaría, caso contrario, cualquier acción realizada con esta hoja en blanco, será atribuida a éste estudiante.
- Antes de que la secretaria firme las hojas se debe acordar para que trámites es aplicable el certificado, y por ende para que certificados tendrá validez la firma.
- Para que el certificado tenga validez, se requerirá de la firma de los implicados (secretaria, director de carrera, etc.)
- En la parte inferior de la hoja se indica para que trámites es aplicable el certificado, y se especifica que para que el certificado tenga validez se requiere de la firma.

Además con la investigación realizada también se pudo conocer que existe un sistema análogo en la ESPOL, (Escuela Superior Politécnica del Litoral), denominado POLIMÁTICO, que ha funcionado de esta forma sin tener inconvenientes.

1.3 DIAGRAMAS DE CASOS DE USO

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Autenticación
Fecha: 09 de noviembre de 2009 Página 1 de 1

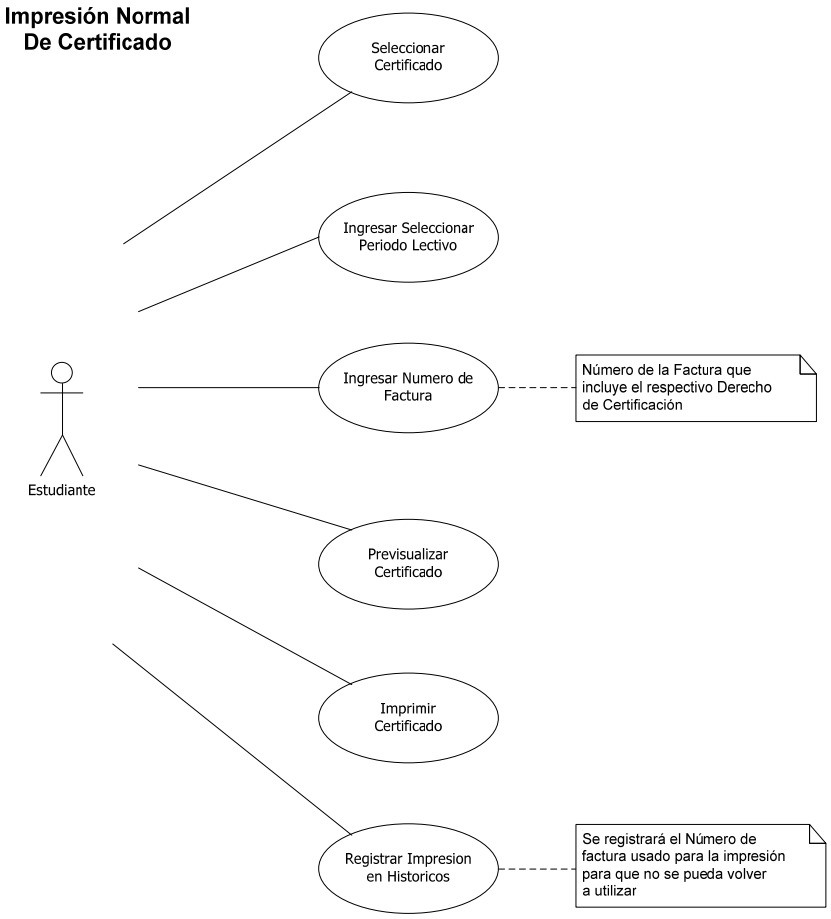


Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Certificados
Fecha: 09 de noviembre de 2009 Página 1 de 2



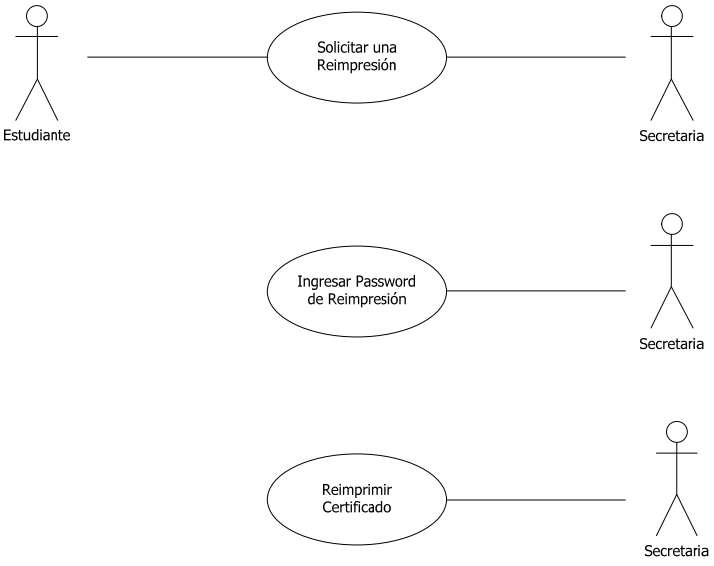
Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Certificados
Fecha: 09 de noviembre de 2009 Página 2 de 2

Error en Impresión de Certificado



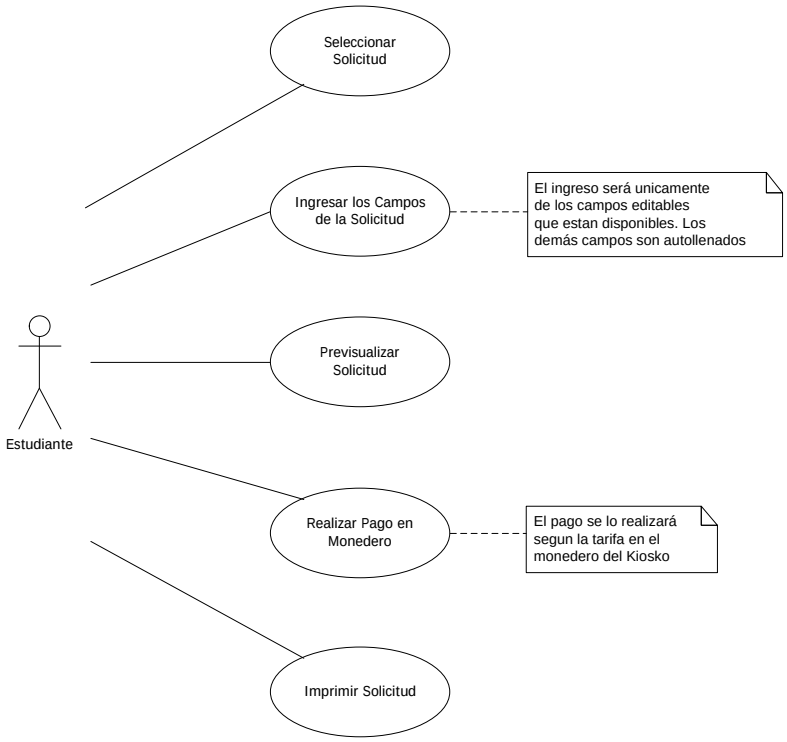
Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 09 de noviembre de 2009 Página 1 de 2

Impresión Normal De Solicitud



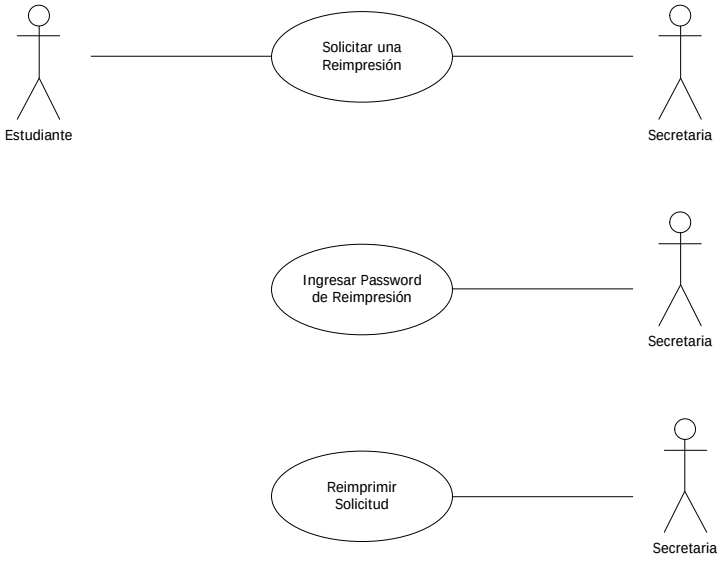
Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 09 de noviembre de 2009 Página 2 de 2

Error en Impresión de Solicitud



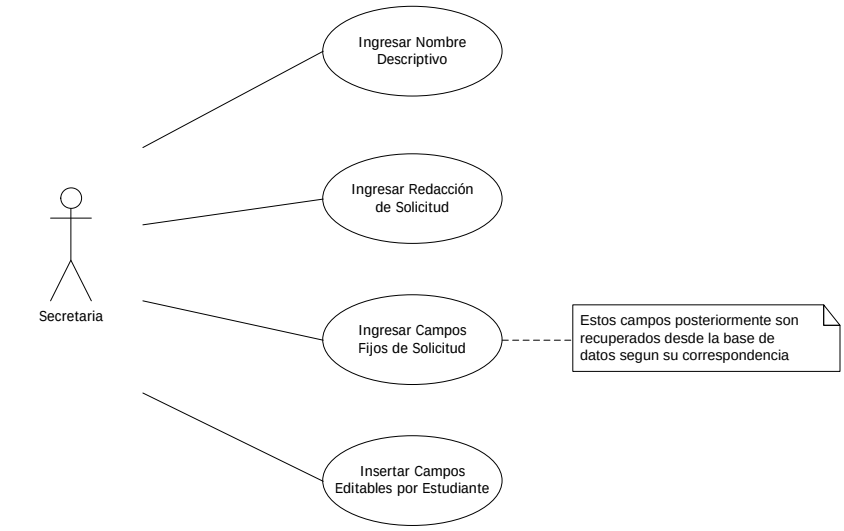
Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Administración
Fecha: 09 de noviembre de 2009 Página 1 de 2

Agregar / Modificar Solicitud



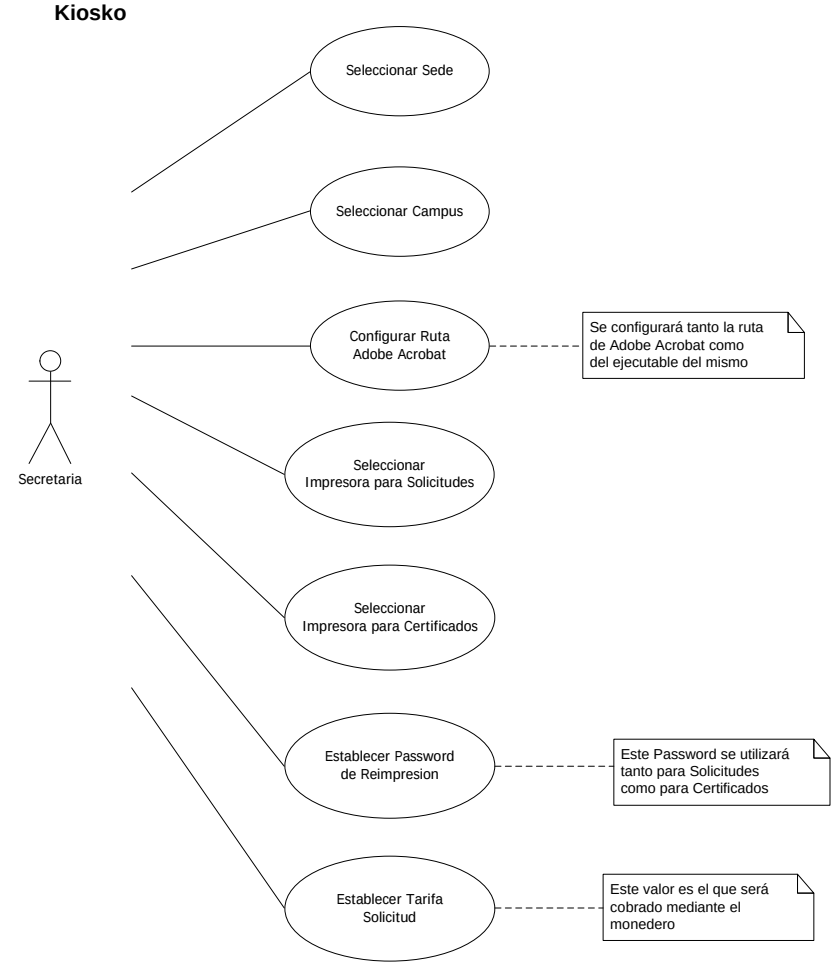
Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Sistema de Gestión del Talento Humano
Documentación del Análisis

Subsistema: Kiosko Multimedia
Módulo: Administración
Fecha: 09 de noviembre de 2009 Página 2 de 2

Configuración del Kiosko



Realizado por: Nataly Molina Fecha Aprobación: 09 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

1.4 DATOS DE ENTRADA Y SALIDA

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Autenticación
Fecha: 11 de noviembre de 2009 Página 1 de 1

Datos de Entrada

- Número de Cédula del Estudiante

Datos de Salida

- Listado de las Carreras que el alumno ha cursado o cursa

Realizado por: Nataly Molina Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Certificados
Fecha: 11 de noviembre de 2009

Datos de Entrada

- Nombre del Certificado
- Periodo Lectivo
- Numero de Factura
- Contraseña de Reimpresión

Datos de Salida

- Datos del Certificado Generado

Realizado por: Nataly Molina
Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi
f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 11 de noviembre de 2009 Página 1 de 1

Datos de Entrada

- Carrera Destinataria
- Nombre de la Solicitud
- Campos de Solicitud
- Contraseña de Reimpresión

Datos de Salida

- Datos de la Solicitud Generada

Realizado por: Nataly Molina Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Administración
Fecha: 11 de noviembre de 2009

Ingreso de Solicitudes

Datos de Entrada

- Nombre Descriptivo de la Solicitud
- Redacción de la Solicitud
- Campos Fijos
- Campos Editables

Realizado por: Nataly Molina — Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi — f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Administración
Fecha: 11 de noviembre de 2009

Configuración del Kiosko

Datos de Entrada

- Nombre de la Sede del Kiosko
- Redacción del Campus del Kiosko
- Ruta Ejecutable Adobe Acrobat
- Nombre de Impresora para Certificados
- Nombre de Impresora para Solicitudes
- Password de Reimpresión
- Costo de cada Solicitud

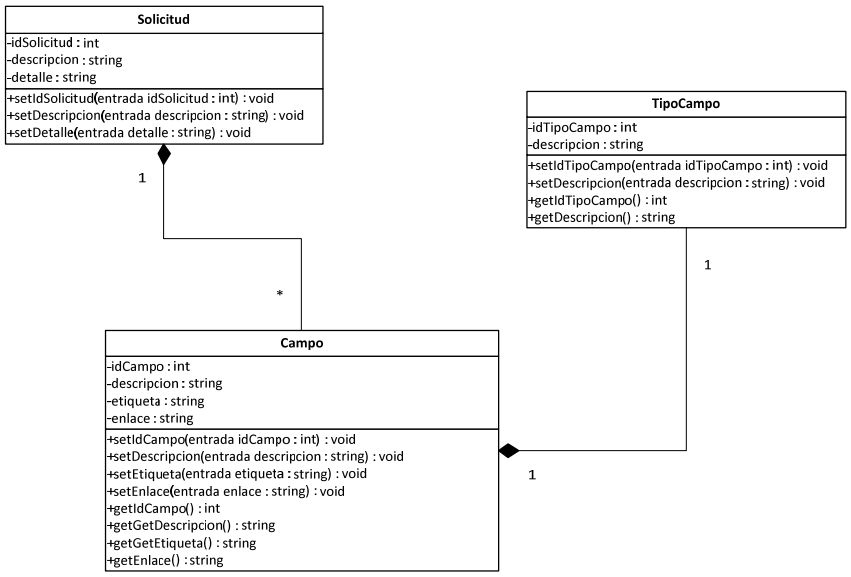
Realizado por: Nataly Molina — Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi — f._____

1.5 DIAGRAMAS DE CLASES

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 11 de noviembre de 2009 Página 1 de 1

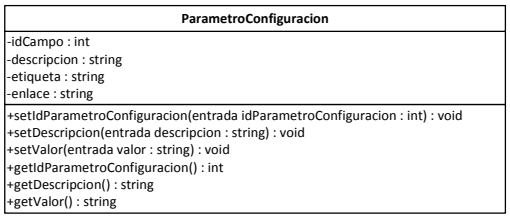


Realizado por: Nataly Molina Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Administracion
Fecha: 11 de noviembre de 2009 Página 1 de 1



Realizado por: Nataly Molina Fecha Aprobación: 12 de Noviembre del 2009

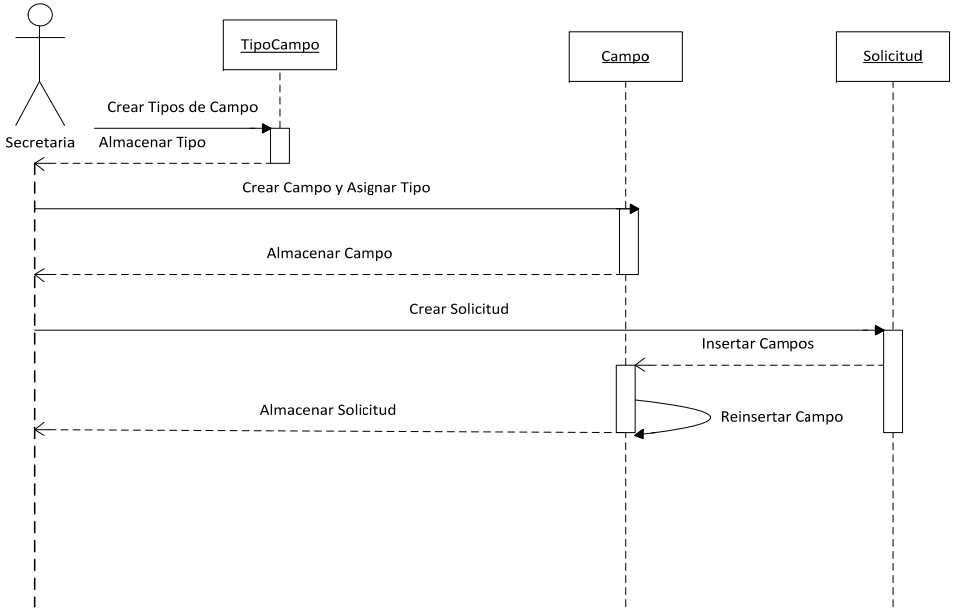
Aprobado por: Ing. Cristian Timbi f._____

1.6 DIAGRAMA DE SECUENCIAS

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 11 de noviembre de 2009

Creación / Modificación de Solicitud



Realizado por: Nataly Molina

Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi

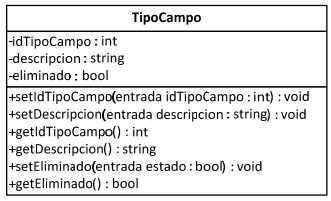
f._____

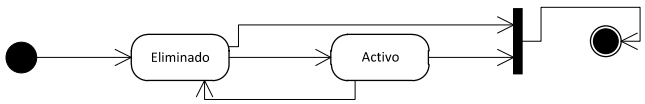
1.7 DIAGRAMAS DE ESTADOS

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

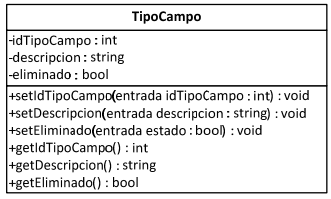
Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 16 de noviembre de 2009 Página 1 de 2

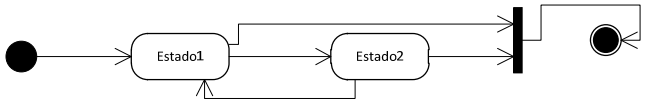
Clase Tipo Campo





Clase TipoCampo





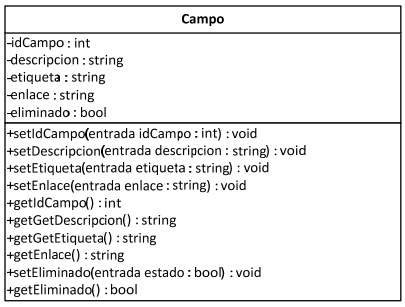
Realizado por: Nataly Molina Fecha Aprobación: 18 de Noviembre del 2009

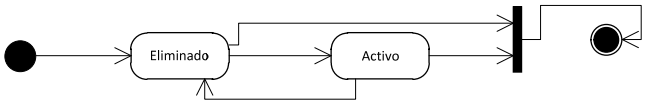
Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 16 de noviembre de 2009 Página 2 de 2

Clase Campo





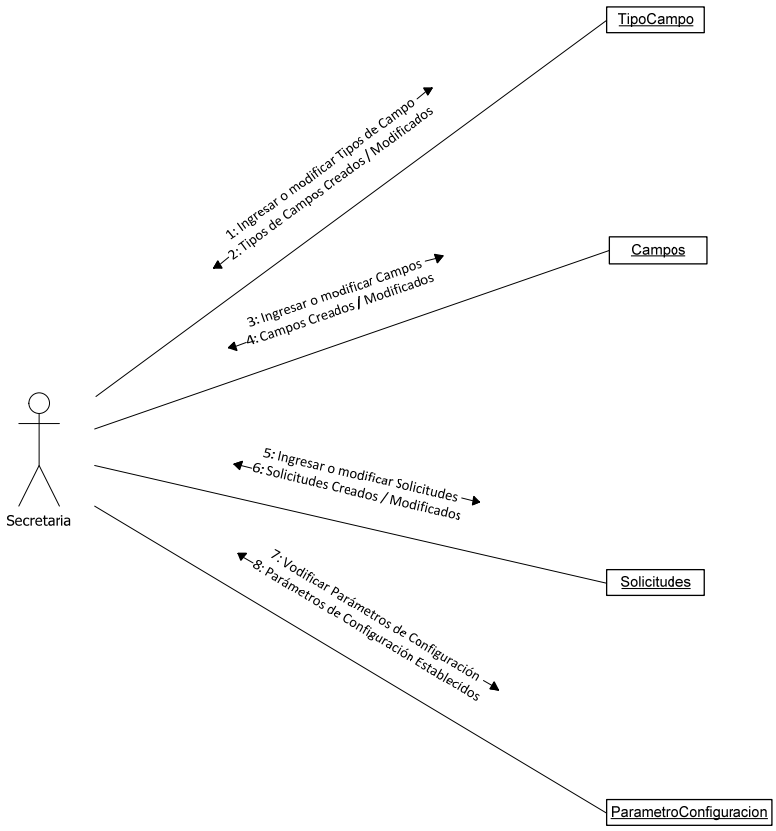
Realizado por: Nataly Molina Fecha Aprobación: 18 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

1.8 DIAGRAMA DE COLABORACIÓN

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Global
Fecha: 16 de noviembre de 2009 Página 1 de 1



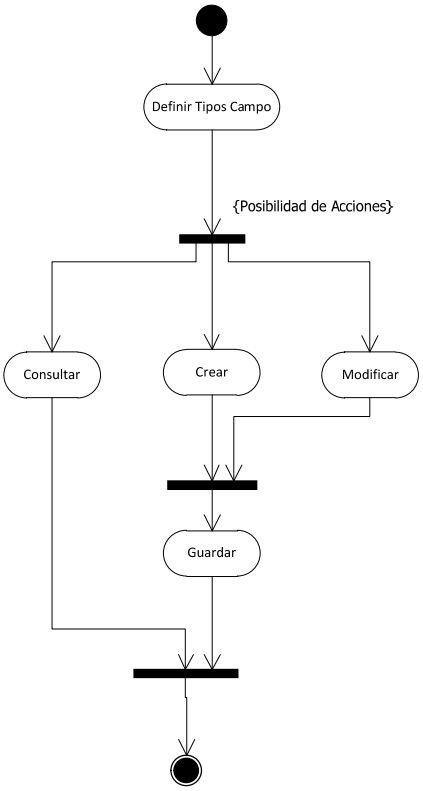
Realizado por: Nataly Molina Fecha Aprobación: 18 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

1.9 DIAGRAMAS DE ACTIVIDAD

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 20 de noviembre de 2009 Página 1 de 4

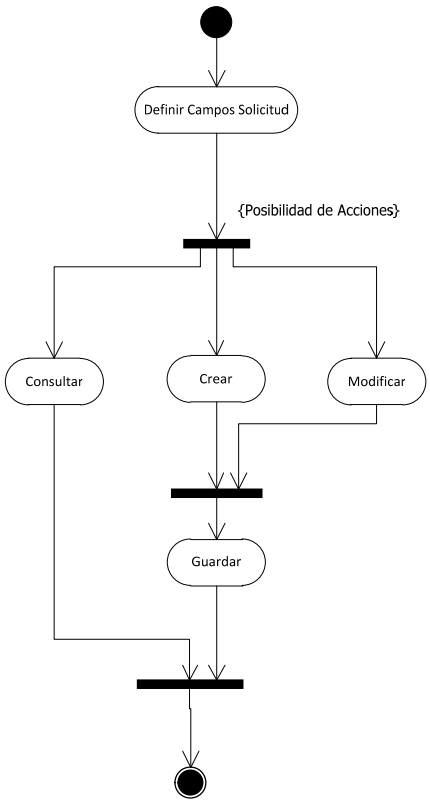


Realizado por: Nataly Molina Fecha Aprobación: 23 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 20 de noviembre de 2009 Página 2 de 3

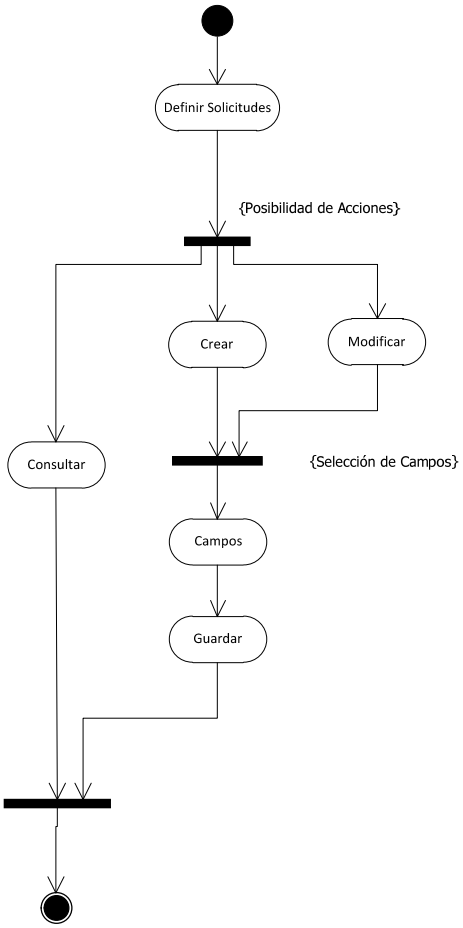


Realizado por: Nataly Molina Fecha Aprobación: 23 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 20 de noviembre de 2009 Página 3 de 3

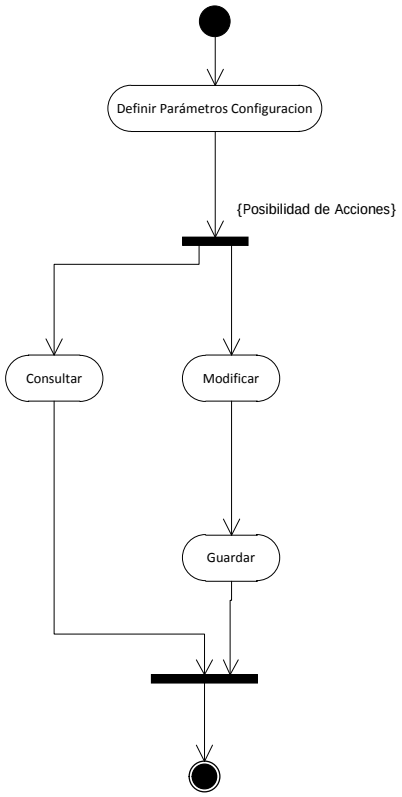


Realizado por: Nataly Molina Fecha Aprobación: 23 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Administración
Fecha: 20 de noviembre de 2009 Página 1 de 1



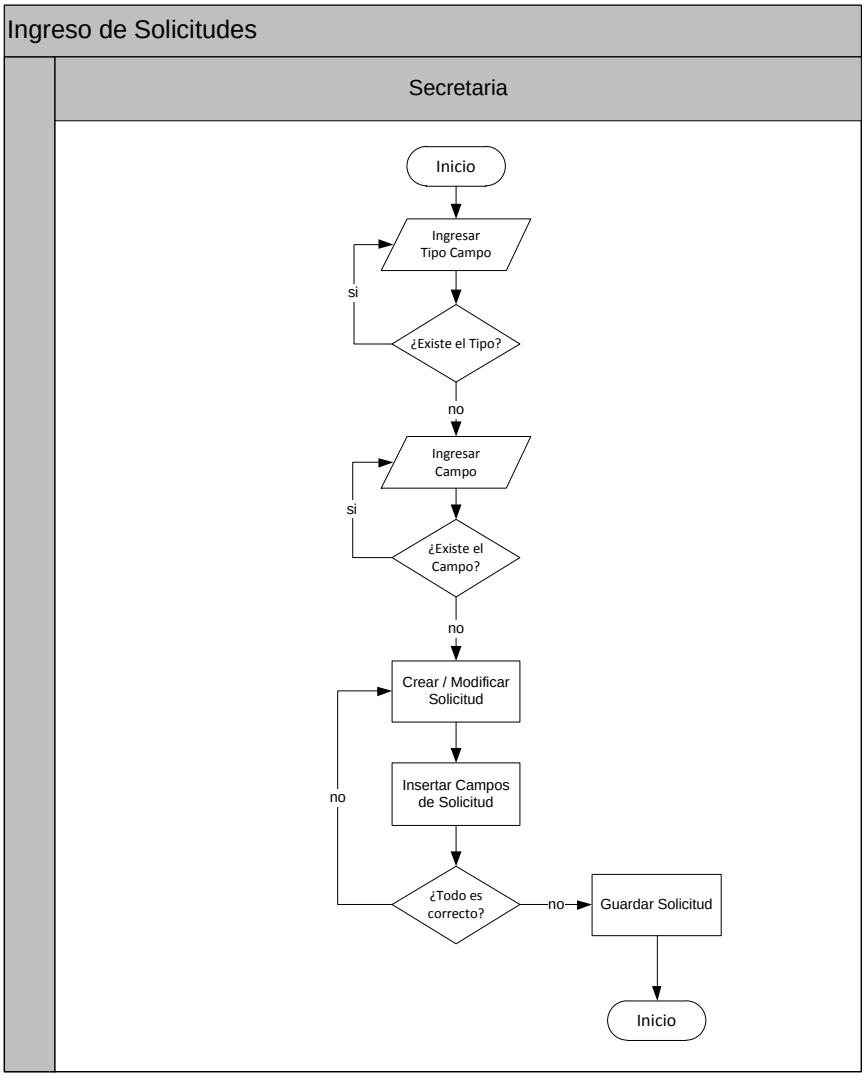
Realizado por: Nataly Molina Fecha Aprobación: 23 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

1.10 DIAGRAMAS DE PROCESOS

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Solicitudes
Fecha: 20 de noviembre de 2009 Página 1 de 1

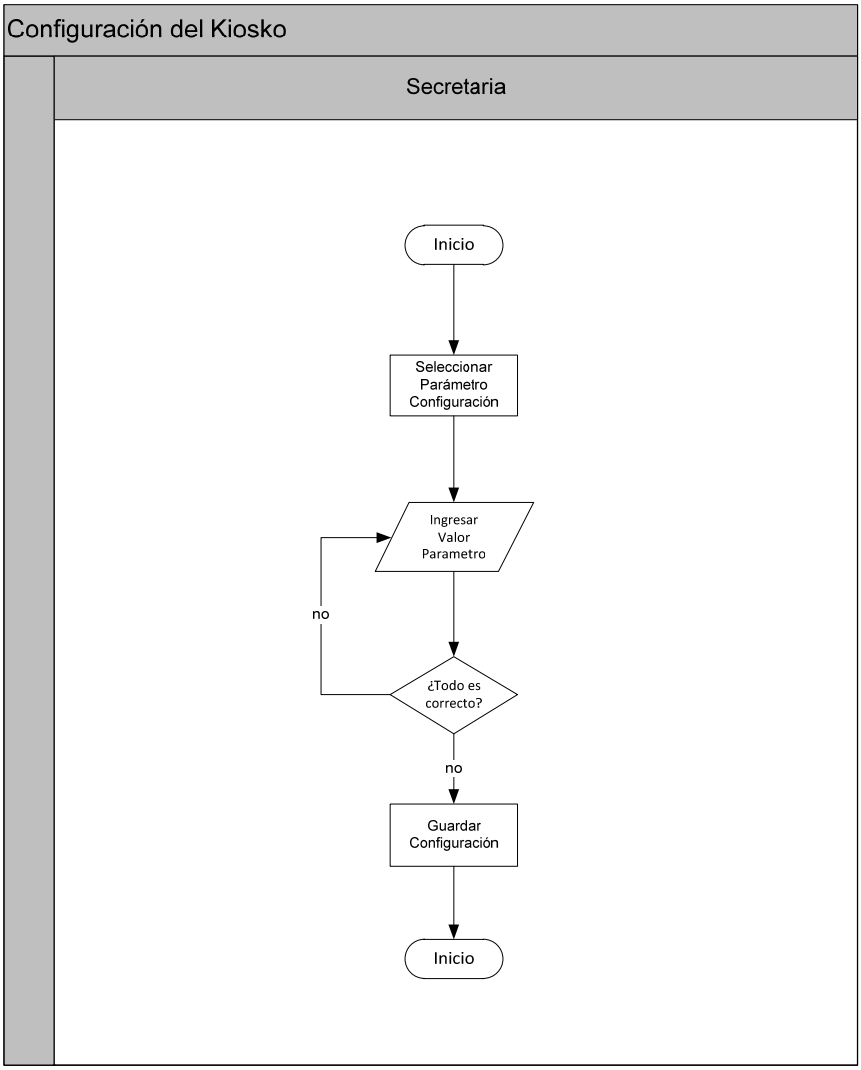


Realizado por: Nataly Molina Fecha Aprobación: 23 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi f._____

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Administración
Fecha: 20 de noviembre de 2009



Realizado por: Nataly Molina

Fecha Aprobación: 23 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi

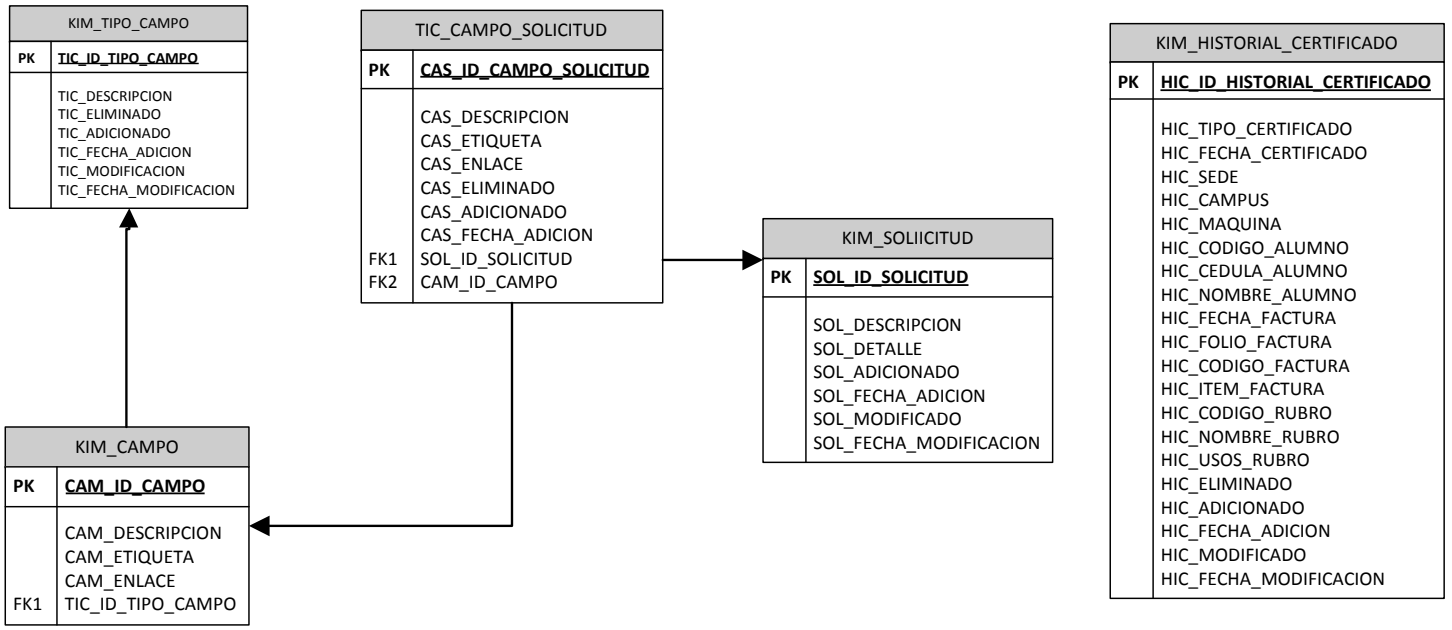
f._____

1.11 DIAGRAMA ENTIDAD RELACIÓN DE BASE DE DATOS

Universidad Politécnica Salesiana
Kiosko Multimedia
Documentación del Diseño

Subsistema: Kiosko Multimedia
Módulo: Global
Fecha: 11 de noviembre de 2009

Página 1 de 1



Realizado por: Nataly Molina

Fecha Aprobación: 12 de Noviembre del 2009

Aprobado por: Ing. Cristian Timbi

f._____

1.12 DICCIONARIO DE DATOS

Nombre de la Tabla: KIM_CAMPO

Descripción de la Tabla: Almacena los campos de las solicitudes, ya sean campos editables o fijos

PK	Columna	Tipo de Dato	Valor Defecto	Nulos	Único	Restricción de Valores	Clave Foránea	Tabla Referencia	Comentarios
#	TIC_ID_TIPO_CAMPO	NUMBER(7,0)		No					Código del Tipo de Campo
	TIC_DESCRIPCION	VARCHAR2(60)		No					Nombre descriptivo del Tipo de Campo
	TIC_ELIMINADO	VARCHAR2(1)	'N'	No		S='Si', N='No'			Campo de Auditoría: Control de eliminación lógica
	TIC_ADICIONADO	VARCHAR2(30)	USER	No					Campo de Auditoría: Usuario que adicionó el registro
	TIC_FECHA_ADICION	DATE	SYSDATE	No					Campo de Auditoría: Fecha en la que se agregó el registro
	TIC_MODIFICADO	VARCHAR2(30)		Si					Campo de Auditoría: Usuario que modificó el registro
	TIC_FECHA_MODIFICACION	DATE		Si					Campo de Auditoría: Fecha en la que se modificó el registro

Nombre de la Tabla: KIM_CAMPO

Descripción de la Tabla: Almacena los campos de las solicitudes, ya sean campos editables o fijos

PK	Columna	Tipo de Dato	Valor Defecto	Nulos	Único	Restricción de Valores	Clave Foránea	Tabla Referencia	Comentarios
#	CAM_ID_CAMPO	NUMBER(7,0)		No					Código del Campo
	TIC_ID_TIPO_CAMPO	NUMBER(7,0)		No			FK1	KIM_TIPO_CAMPO	Tipo del Campo (Editable / Fijo)
	CAM_DESCRIPCION	VARCHAR2(60)		No					Nombre descriptivo del Campo
	CAM_ETIQUETA	VARCHAR2(50)		No					Identificador Visual del Campo en la Redacción de la Solicitud
	CAM_ENLACE	VARCHAR2(200)		No					Enlace desde el Campo hacia una Columna de la Base de Datos
	CAM_ELIMINADO	VARCHAR2(1)	'N'	No		S='Si', N='No'			Campo de Auditoría: Control de eliminación lógica
	CAM_ADICIONADO	VARCHAR2(30)	USER	No					Campo de Auditoría: Usuario que adicionó el registro
	CAM_FECHA_ADICION	DATE	SYSDATE	No					Campo de Auditoría: Fecha en la que se agregó el registro
	CAM_MODIFICADO	VARCHAR2(30)		Si					Campo de Auditoría: Usuario que modificó el registro
	CAM_FECHA_MODIFICACION	DATE		Si					Campo de Auditoría: Fecha en la que se modificó el registro

Nombre de la Tabla: KIM_CAMPO_SOLICITUD

Descripción de la Tabla: Almacena campos asignados a una determinada solicitud

PK	Columna	Tipo de Dato	Valor Defecto	Nulos	Único	Restricción de Valores	Clave Foránea	Tabla Referencia	Comentarios
#	CAS_ID_CAMPO_SOLICITUD	NUMBER(7,0)		No					Código de Campo/Solicitud
#	SOL_ID_SOLICITUD	NUMBER(7,0)		No			FK1	KIM_SOLICITUD	Código de la Solicitud
#	CAM_ID_CAMPO	NUMBER(7,0)		No			FK2	KIM_CAMPO	Código del Campo
	CAS_ELIMINADO	VARCHAR2(1)	'N'	No		S='Si', N='No'			Campo de Auditoría: Control de eliminación lógica
	CAS_ADICIONADO	VARCHAR2(30)	USER	No					Campo de Auditoría: Usuario que adicionó el registro
	CAS_FECHA_ADICION	DATE	SYSDATE	No					Campo de Auditoría: Fecha en la que se agregó el registro
	CAS_MODIFICADO	VARCHAR2(30)		Yes					Campo de Auditoría: Usuario que modificó el registro
	CAS_FECHA_MODIFICACION	DATE		Yes					Campo de Auditoría: Fecha en la que se modificó el registro

Nombre de la Tabla: KIM_SOLICITUD

Descripción de la Tabla: Almacena las solicitudes, su descripción y redacción

PK	Columna	Tipo de Dato	Valor Defecto	Nulos	Único	Restricción de Valores	Clave Foránea	Tabla Referencia	Comentarios
	SOL_ID_SOLICITUD	NUMBER(7,0)		No					Código de la Solicitud
	SOL_DESCRIPCION	VARCHAR2(60)		No					Nombre de la Solicitud
	SOL_DETALLE	VARCHAR2(2000)		No					Redacción de la Solicitud
	SOL_ELIMINADO	VARCHAR2(1)	'N'	No		S='Si', N='No'			Campo de Auditoría: Control de eliminación lógica
	SOL_ADICIONADO	VARCHAR2(30)	USER	No					Campo de Auditoría: Usuario que adicionó el registro
	SOL_FECHA_ADICION	DATE	SYSDATE	No					Campo de Auditoría: Fecha en la que se agregó el registro
	SOL_MODIFICADO	VARCHAR2(30)		Si					Campo de Auditoría: Usuario que modificó el registro
	SOL_FECHA_MODIFICACION	DATE		Si					Campo de Auditoría: Fecha en la que se modificó el registro

Nombre de la Tabla: KIM_HISTORIAL_CERTIFICADOS

Descripción de la Tabla: Almacena el historial de los certificados emitidos

PK	Columna	Tipo de Dato	Valor Defecto	Nulos	Restricción de Valores	Comentarios
	HIC_ID_HISTORIAL_CERTIFICADO	NUMBER(7,0)		No		Código del Historial Certificado
	HIC_TIPO_CERTIFICADO	VARCHAR2(100)		No		Nombre del Certificado Impreso
	HIC_FECHA_CERTIFICADO	DATE	SYSDATE	No		Fecha en la que se emitió el certificado
	HIC_SEDE	VARCHAR2(100)		No		Nombre de la Sede en la que se emitió el certificado
	HIC_CAMPUS	VARCHAR2(100)		No		Nombre del Campus en el que se emitió el certificado
	HIC_MAQUINA	VARCHAR2(100)		No		Nombre del equipo en que se emitió el certificado
	HIC_CODIGO_ALUMNO	NUMBER(10,0)		No		Código del Alumno (El que corresponde a SIGAC.CLIENTE_LOCAL.CLLC_CDG)
	HIC_CEDULA_ALUMNO	VARCHAR2(16)		No		Documento de Identificación utilizado al momento de la emision
	HIC_NOMBRE_ALUMNO	VARCHAR2(255)		No		Nombre Completo del Alumno que emitió la solicitud
	HIC_FECHA_FACTURA	DATE		No		Fecha en la que se emitió la factura utilizada para imprimir el certificado
	HIC_FOLIO_FACTURA	NUMBER(8,0)		No		Número de Folio de la factura utilizada para imprimir el certificado
	HIC_CODIGO_FACTURA	NUMBER(8,0)		No		Código de la factura utilizada para imprimir el certificado
	HIC_ITEM_FACTURA	NUMBER(4,0)		No		Numero de Item del derecho de certificación en la factura utilizada para imprimir el certificado
	HIC_CODIGO_RUBRO	VARCHAR2(8)		No		Código del Derecho de Certificación
	HIC_NOMBRE_RUBRO	VARCHAR2(120)		No		Nombre del Derecho de Certificación utilizado
	HIC_USOS_RUBRO	NUMBER		No		Numero de sus restantes del Derecho de certificación incluido en la factura (Para casos en los que se compren 2 o más)
	HIC_ELIMINADO	VARCHAR2(1)	'N'	No	S='Si', N='No'	Campo de Auditoría: Control de eliminación lógica
	HIC_ADICIONADO	VARCHAR2(30)	USER	No		Campo de Auditoría: Usuario que adicionó el registro
	HIC_FECHA_ADICION	DATE	SYSDATE	No		Campo de Auditoría: Fecha en la que se agregó el registro
	HIC_MODIFICADO	VARCHAR2(30)		Si		Campo de Auditoría: Usuario que modificó el registro

	HIC_FECHA_MODIFICACION	DATE		Si		Campo de Auditoría: Fecha en la que se modificó el registro

1.13 ACCESO A TABLAS EXTERNAS

Esquema: SIGAC

Esquema	Nombre de Tabla	Privilegios Activos
SIGAC	CLIENTE_LOCAL	SELECT

Esquema: EASI

Esquema	Nombre de Tabla	Privilegios Activos
EASI	D_FACTURA	SELECT
EASI	D_FACTURA1	SELECT
EASI	D_LISTA_PRECIOS	SELECT
EASI	E_FACTURA	SELECT
EASI	E_NVENTA	SELECT
EASI	SERVICIO	SELECT

Esquema: GHADMIN

Esquema	Nombre de Tabla	Privilegios Activos
GHADMIN	RH_PERSON	SELECT
GHADMIN	RH_PROFESION	SELECT

Esquema: SNA

Esquema	Nombre de Tabla	Privilegios Activos
SNA	SNA_ALUMNO	SELECT
SNA	SNA_CAMPUS	SELECT
SNA	SNA_CARRERA	SELECT
SNA	SNA_CARRERA_CAMPUS	SELECT
SNA	SNA_FACULTAD	SELECT
SNA	SNA_INSCRIPCION_ACADEMICO	SELECT
SNA	SNA_MATERIA	SELECT
SNA	SNA_MATRICULA	SELECT
SNA	SNA_MOD_PRO_ACA	SELECT
SNA	SNA_MOD_PRO_CAR_CAM	SELECT
SNA	SNA_MODALIDAD	SELECT
SNA	SNA_PERIODO_LECTIVO	SELECT
SNA	SNA_PROYECTO_ACADEMICO	SELECT
SNA	SNA_PROYECTO_CARRERA_CAMPUS	SELECT
SNA	SNA_SECRETARIO	SELECT
SNA	SNA_SEDE	SELECT

CAPITULO II

PHP

2.1. INTRODUCCIÓN

PHP es el lenguaje de lado servidor más extendido en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente reciente, aunque con la rapidez con la que evoluciona Internet parezca que ha existido toda la vida. Es un lenguaje que ha tenido una gran aceptación en la comunidad de desarrolladores, debido a la potencia y simplicidad que lo caracterizan, así como al soporte generalizado en la mayoría de los servidores de hosting.

PHP permite embeber sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz, combinando lo que ya se sabe del desarrollo HTML. Es decir, con PHP se escriben scripts dentro del código HTML. Por otra parte, y es aquí donde reside su mayor interés con respecto a los lenguajes pensados para los CGI, PHP ofrece un sinfín de funciones para la explotación de bases de datos de una manera llana, sin complicaciones.

PHP ofrece muchas ventajas pero a la vez un inconveniente.

Entre las Ventajas

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).

- Posee una amplia documentación en su página oficial[1], entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño siguiendo el Modelo Vista Controlador (o MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes (ver más abajo Frameworks en PHP).

Inconvenientes

- La ofuscación de código es la única forma de ocultar las fuentes.

[1]Documentación de PHP disponible en: http://www.php.net/manual/es/

2.2. LA SINTAXIS DE PHP

PHP se escribe dentro de la propia página web, junto con el código HTML y, como para cualquier otro tipo de lenguaje incluido en un código HTML, en PHP se necesita especificar cuáles son las partes constitutivas del código escritas en este lenguaje. Esto se hace, como en otros casos, delimitando el código por etiquetas. Se puede utilizar distintos modelos de etiquetas en función de las preferencias y costumbres. Hay que tener sin embargo en cuenta que no necesariamente todas están configuradas inicialmente y que otras, como es el caso de <% y %> sólo están disponibles a partir de una determinada versión (3.0.4.).

Estos modos de abrir y cerrar las etiquetas son:

```
<?       y     ?>
<%       y     %>
<?php   y     ?>
<script language="php">
```

Figura 1.2.1: Ejemplo de Script PHP

El modo de funcionamiento de una página PHP, a grandes rasgos, no difiere del clásico para una página dinámica de lado servidor: El servidor va a reconocer la extensión correspondiente a la página PHP (phtml, php, php4,...) y antes de enviarla al navegador va a encargarse de interpretar y ejecutar todo aquello que se encuentre entre las etiquetas correspondientes al lenguaje PHP. El resto, lo enviara sin más ya que, asumirá que se trata de código HTML absolutamente comprensible por el navegador.

Otra característica general de los scripts en PHP es la forma de separar las distintas instrucciones. Para hacerlo, hay que acabar cada instrucción con un punto y coma ";". Para la última expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario.

Se incluye también en este capítulo la sintaxis de comentarios. Un comentario, es una frase o palabra que se incluye en el código para comprenderlo

más fácilmente al volverlo a leer un tiempo después y que, por supuesto, el ordenador tiene que ignorar ya que no va dirigido a él sino al programador. Los comentarios tienen una gran utilidad ya que es muy fácil olvidarse del funcionamiento de un script programado un tiempo atrás y resulta muy útil si se quiere hacer rápidamente comprensible el código a otra persona.

La forma de incluir estos comentarios es variable dependiendo si se quiere escribir una línea o más. El siguiente es un ejemplo de script:

```
<?
$mensaje="Tengo hambre!!"; //Comentario de una línea
echo $mensaje; #Este comentario también es de una línea
/*En este caso
el comentario ocupa
varias líneas */?>
```

Si se usa doble barra (//) o el símbolo # se pueden introducir comentarios de una línea. Mediante /* y */ se crean comentarios multilínea.

2.3. VARIABLES EN PHP

En PHP tanto las variables son definidas anteponiendo el símbolo dólar ($) al nombre de la variable.

Dependiendo de la información que contenga, una variable puede ser considerada de uno u otro tipo:

Variables numéricas Almacenan cifras		
Enteros	$entero=2002;	Números sin decimales
Real	$real=3.14159;	Números con o sin decimal

Tabla 1.3.1: Variables Numéricos en PHP

Variables alfanuméricas Almacenan textos compuestos de números y/o cifras		
Cadenas	Almacenan variables alfanuméricas	$cadena="Hola amigo";

Tabla 2.3.2: Variables Alfanuméricas en PHP

Tablas Almacenan series de informaciones numéricas y/o alfanuméricas		
Arrays	Son las variables que guardan las tablas	$sentido[1]="ver"; $sentido[2]="tocar"; $sentido[3]="oir"; $sentido[4]="gusto"; $sentido[5]="oler";

Tabla 3.3.3: Variables Tipo Tabla en PHP

2.3.1. Variables Objeto

Se trata de conjuntos de variables y funciones asociadas. Presentan una complejidad mayor que las variables vistas hasta ahora pero su utilidad es más que interesante.

A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando se define una variable asignándole un valor, el ordenador le atribuye un tipo. Si por ejemplo se define una variable entre comillas, la variable será considerada de tipo cadena:

```
$variable="5"; //esto es una cadena
```

Sin embargo si se pide en un script realizar una operación matemática con esta variable, no se obtendrá un mensaje de error sino que la variable cadena será asimilada a numérica:

```
<?
$cadena="5"; //esto es una cadena
$entero=3; //esto es un entero
echo $cadena+$entero ?>
```

El script dará como resultado "8". La variable cadena ha sido asimilada en entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, se puede operar entre variables tipo entero y real. PHP se encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa.

Sin embargo, en contraste, hay que tener cuidado en no cambiar mayúsculas por minúsculas ya que, en este sentido, PHP es sensible. Conviene por lo tanto trabajar ya sea siempre en mayúsculas o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar.

2.3.2. Variables asignadas por referencia

En PHP también se pueden asignar variables por referencia. En ese caso no se les asigna un valor, sino otra variable, de tal modo que las dos variables comparten espacio en memoria para el mismo dato.

La notación para asignar por referencia es colocar un "&" antes del nombre de la variable.

```
<?php
$foo = 'Bob'; // Asigna el valor 'Bob' a $foo
$bar = &$foo; // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo; // $foo también se modifica.
echo $bar;
?>
```

Esto dará como resultado la visualización dos veces del string "Mi nombre es Bob". Algo como:

```
Mi nombre es BobMi nombre es Bob
```

2.3.3. Cambio del Tipo de las Variables en PHP

PHP no requiere que se le indique el tipo que va a contener una variable, sino que lo deduce del valor que se asigne a la variable. Asimismo, se encarga de actualizar automáticamente el tipo de la variable cada vez que le asigna un nuevo valor.

Por ello, para cambiar el tipo de una variable simplemente se le asigna un valor con un nuevo tipo.

Nota: Se excluyen en este caso el cambio de variables a tipo Array porque la sintaxis puede resultar ambigua al expresar ese código, es decir, puede darse el caso de que una línea de código pueda significar dos cosas.

```
$a = "1";
//$a es una cadena
$a[0] = "f";
//¿Se está editando el índice de la cadena o forzando a Array?
```

2.3.4. Cambio Forzado de Tipos de Variable

En cualquier caso, se puede forzar una variable para que cambie de tipo con la función setType().

```
setType($variable,"nuevo_tipo");
```

La función setType() actualiza el tipo de $variable a "nuevo_tipo" y devuelve un booleano indicando si hubo éxito o no en la conversión.

Entre "nuevo_tipo" se tiene:

```
"integer"
"double"
"string"
"Array"
"object"
```

También se puede hacer que una variable se comporte como un tipo determinado forzándola, de la misma manera a como se hace en el lenguaje C.

```
$variable = "23";
$variable = (int) $variable;
```

2.3.5. Los cambios forzados permitidos

(int), (integer) - fuerza a entero (integer)
(real), (double), (float) - fuerza a doble (double)
(string) - fuerza a cadena (string)
(Array) - fuerza a Array (Array)
(object) - fuerza a objeto (object)

2.4. VARIABLES DE SISTEMA EN PHP

Dada su naturaleza de lenguaje de lado servidor, PHP es capaz de dar acceso a toda una serie de variables que informan sobre el servidor y sobre el cliente. La información de estas variables es atribuida por el servidor y en ningún caso deja posible modificar sus valores directamente mediante el script. Para hacerlo es necesario influir directamente sobre la propiedad que definen.

Existen multitud de variables de este tipo, algunas sin utilidad aparente y otras realmente interesantes y con una aplicación directa para un sitio web. Aquí se enumeran algunas de estas variables y la información que aportan:

Variable	Descripción
$HTTP_USER_AGENT	Informa principalmente sobre el sistema operativo y tipo y versión de navegador utilizado por el internauta. Su principal utilidad radica en que, a partir de esta información, se puede redireccionar a los usuarios hacia páginas optimizadas para su navegador o realizar cualquier otro tipo de acción en el contexto de un navegador determinado.
$HTTP_ACCEPT_LANGUAGE	Devuelve la o las abreviaciones de la lengua considerada como principal por el navegador. Esta lengua o lenguas principales pueden ser elegidas en el menú de opciones del navegador. Esta es útil para enviar al internauta a las páginas escritas en su lengua, si es que existen.

$HTTP_REFERER	Indica la URL desde la cual el internauta ha tenido acceso a la página. Muy interesante para generar botones de "Atrás" dinámicos o para crear propios sistemas de estadísticas de visitas.
$PHP_SELF	Devuelve una cadena con la URL del script que está siendo ejecutado. Muy interesante para crear botones para recargar la página.
$HTTP_GET_VARS	Se trata de un Array que almacena los nombres y contenidos de las variables enviadas al script por URL o por formularios GET
$HTTP_POST_VARS	Se trata de un Array que almacena los nombres y contenidos de las variables enviadas al script por medio de un formulario POST
$PHP_AUTH_USER	Almacena la variable usuario cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_PW resulta ideal para controlar el acceso a las páginas internas del sitio.
$PHP_AUTH_PW	Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_USER resulta ideal para controlar el acceso a las páginas internas del sitio.
$REMOTE_ADDR	Muestra la dirección IP del visitante.
$DOCUMENT_ROOT	Devuelve el path físico en el que se encuentra alojada la página en el servidor.
$PHPSESSID	Guarda el identificador de sesión del usuario.

Tabla 4.4.1: Variables de Sistema en PHP

No todas estas variables están disponibles en la totalidad de servidores o en determinadas versiones de un mismo servidor. Además, algunas de ellas han de ser previamente activadas o definidas por medio de algún acontecimiento. Así, por ejemplo, la variable $HTTP_REFERER no estará definida a menos que el internauta acceda al script a partir de un enlace desde otra página.

2.5. VARIABLES SUPERGLOBALES

A partir de PHP 4.1.0, se dispone de un conjunto de variables de tipo Array que mantienen información del sistema, llamadas superglobales porque se definen automáticamente en un ámbito global.

Estas variables hacen referencia a las mismas que se accedían antes por medio de los Array del tipo $HTTP_*_VARS.

La lista de estas variables, extraída directamente de la documentación de PHP es la siguiente:

- **$GLOBALS**
  Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las llaves de esta matriz son los nombres de las variables globales. $GLOBALS existe desde PHP 3.
- **$_SERVER**
  Variables definidas por el servidor web o directamente relacionadas con el entorno en donde el script se está ejecutando. Análoga a la antigua matriz $HTTP_SERVER_VARS (la cual está todavía disponible, aunque no se use).
- **$_GET**
  Variables proporcionadas al script por medio de HTTP GET. Análoga a la antigua matriz $HTTP_GET_VARS (la cual está todavía disponible, aunque no se use).
- **$_POST**
  Variables proporcionadas al script por medio de HTTP POST. Análoga a la

antigua matriz $HTTP_POST_VARS (la cual está todavía disponible, aunque no se use).

- **$_COOKIE**
  Variables proporcionadas al script por medio de HTTP cookies. Análoga a la antigua matriz $HTTP_COOKIE_VARS (la cual está todavía disponible, aunque no se use).
- **$_FILES**
  Variablesproporcionadas al script por medio de la subida de ficheros víaHTTP. Análoga a la antigua matriz $HTTP_POST_FILES (la cual está todavía disponible, aunque no se use).
- **$_ENV**
  Variables proporcionadas al script por medio del entorno. Análoga a la antigua matriz $HTTP_ENV_VARS (la cual está todavía disponible, aunque no se use).
- **$_REQUEST**
  Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario y por lo tanto no se puede confiar en ellas. La presencia y el orden en que aparecen las variables en esta matriz es definido por la directiva de configuración variables_order. Esta matriz no tiene un análogo en versiones anteriores a PHP 4.1.0.
- **$_SESSION**
  Variables registradas en la sesión del script. Análoga a la antigua matriz $HTTP_SESSION_VARS (la cual está todavía disponible, aunque no se use).

2.6. TABLAS O ARRAYS EN PHP

Un tipo de variable que puede ser relativamente complicado de asimilar con respecto a la mayoría son los Array. Un Array es una variable que está compuesta de varios elementos cada uno de ellos catalogado dentro de ella misma por medio de una clave.

Un ejemplo de un Array llamado sentido con los distintos sentidos del ser humano:

```
 $sentido[1]="ver";
$sentido[2]="tocar";
$sentido[3]="oir";
$sentido[4]="gustar";
$sentido[5]="oler";
```

En este caso este Array cataloga sus elementos, comúnmente llamados valores, por números. Los números del 1 al 5 son por lo tanto las claves y los sentidos son los valores asociados. Nada impide emplear nombres (cadenas) para clasificarlos. Lo único que se debe hacer es entrecomillarlos:

```
<?
$moneda["espana"]="Peseta";
$moneda["francia"]="Franco";
$moneda["usa"]="Dolar";

?>
```

Otra forma de definir idénticamente este mismo Array y que puede ayudar para la creación de Array más complejos es la siguiente sintaxis:

```
<?
$moneda=Array("espana"=> "Peseta","francia" => "Franco","usa" => "Dolar");
?>
```

Una forma muy práctica de almacenar datos es mediante la creación de Array multidimensionales (tablas). Por ejemplo: Se requiere almacenar dentro de una misma tabla el nombre, moneda y lengua hablada en cada país. Para hacerlo se puede emplear un Array llamado país que vendrá definido por estas tres características (claves). Para crearlo, hay que escribir una expresión del mismo tipo que la vista anteriormente en la que se incluirá un Array dentro del otro. Este proceso de incluir una instrucción dentro de otra se llama anidar y es muy corriente en programación:

```
<?
$pais=Array
(
"espana" =>Array
  (
  "nombre"=>"España",
  "lengua"=>"Castellano",
  "moneda"=>"Peseta"
  ),
"francia" =>Array
  (
  "nombre"=>"Francia",
  "lengua"=>"Francés",
  "moneda"=>"Franco"
  )
);
echo $pais["espana"]["moneda"] //Saca en pantalla: "Peseta"
?>
```

Como puede verse, en esta secuencia de script, no se ha introducido punto y coma ";" al final de cada línea. Esto es simplemente debido a que lo que está escrito puede ser considerado como una sola instrucción. En realidad, es el programador quien decidirá cortarla en varias líneas para, así, facilitar su lectura. La verdadera instrucción acabaría una vez definido completamente el Array y es precisamente ahí donde ha de colocarse el único punto y coma. Por otra parte, se puede notar cómo se ha jugado con el tabulador para separar unas líneas más que otras del principio. Esto también se lo hace por cuestiones de claridad, ya que permite ver qué partes del código están incluidas dentro de otras. Es importante acostumbrarse a escribir de esta forma del mismo modo que a introducir los comentarios ya que la claridad de los scripts es fundamental a la hora de depurarlos. Un poco de esfuerzo a la hora de crearlos puede ahorrar muchas horas a la hora de corregirlos o modificarlos meses más tarde.

Pasando ya al comentario del programa, como se puede apreciar, éste permite almacenar tablas y, a partir de una simple petición, visualizar un determinado valor en pantalla.

Lo que es interesante es que la utilidad de los Arrays no acaba aquí, sino que también se pueden utilizar toda una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, contar el número de elementos que

componen el Array además de poder moverse por dentro de él hacia delante o atrás.

La siguiente tabla ilustra las funciones contenidas dentro de la Clase Array[2], que pueden utilizarse según sea el caso u objetivo:

Función	Descripción
array_values (mi_array)	Lista los valores contenidos en mi_array
asort(mi_array) y arsort(mi_array)	Ordena por orden alfabético directo o inverso en función de los valores
count(mi_array)	Devuelve el número de elementos del Array
ksort(mi_array) y krsort(mi_array)	Ordena por orden alfabético directo o inverso en función de las claves
list ($variable1, $variable2...)=mi_array	Asigna cada una variable a cada uno de los valores del Array
next(mi_array), prev(mi_array), reset(mi_array) y end(mi_array)	Permite moverse por dentro del Array con un puntero hacia delante, atrás y al principio y al final.
each(mi_array)	Devuelve el valor y la clave del elemento en el que uno se encuentra y mueve al puntero al siguiente elemento.

Tabla 5.6.1: Funciones Importantes de la Clase Array

De gran utilidad es también el bucle **foreach**[3] que recorre de forma secuencial el Array de principio a fin.

[2]Documentación de la Clase Array, disponible en: http://www.php.net/manual/es/ref.array.php
[3]Documentación del bucle foreach, disponible en: http://www.desarrolloweb.com/articulos/315.php?manual=12

2.7. TRABAJO CON TABLAS O ARRAYS EN PHP

A continuación se exponen algunas de las funciones típicas del trabajo con Arrays a través de una pequeña explicación y un ejemplo de uso.

2.7.1. La función array_slice()

Para disminuir el número de casillas de un arreglo se tienen varias funciones. Entre ellas, array_slice() la se utiliza cuando se quiere recortar algunas casillas del arreglo, sabiendo los índices de las casillas que se desea conservar.

Recibe tres parámetros. El Array, el índice del primer elemento y el número de elementos a tomar, siendo este último parámetro opcional.

En el ejemplo siguiente,hay un Array con cuatro nombres propios. En la primera ejecución de array_slice() , se desea tomar todos los elementos desde el índice 0 (el principio) hasta un número total de 3 elementos. El segundo array_slice() indica que se tomen todos los elementos a partir del índice 1 (segunda casilla).

```
<?
$entrada = Array ("Miguel", "Pepe", "Juan", "Julio", "Pablo");

//modifico el tamaño
$salida = array_slice ($entrada, 0, 3);
//Se muestra el Array
foreach ($salida as $actual)  echo $actual . "<br>";
echo "<p>";

//modifico otra vez
$salida = array_slice ($salida, 1);
//Se muestra el Array
foreach ($salida as $actual)
   echo $actual . "<br>";
?>
```

Tendrá como salida:

```
Miguel
Pepe
Juan

Pepe
Juan
```

2.7.2. La función array_shift()

Esta función extrae el primer elemento del Array y lo devuelve. Además, acorta la longitud del Array eliminando el elemento que estaba en la primera casilla. Siempre hace lo mismo, por tanto, no recibirá más que el Array al que se desea eliminar la primera posición.

En el código siguiente se tiene el mismo vector con nombres propios y se ejecuta dos veces la función array_shift() eliminando un elemento en cada ocasión. Se imprimen los valores devueltos por la función y los elementos del Array resultante de eliminar la primera casilla.

```
<?
$entrada = Array ("Miguel", "Pepe", "Juan", "Julio", "Pablo");

//quito la primera casilla
$salida = array_shift ($entrada);
//Se muestra el Array
echo "La función devuelve: " . $salida . "<br>";
foreach ($entrada as $actual)   echo $actual . "<br>";
echo "<p>";

//quito la primera casilla, que ahora sería la segunda del Array original
$salida = array_shift ($entrada);
echo "La función devuelve: " . $salida . "<br>";
//Se muestra el Array
foreach ($entrada as $actual)  echo $actual . "<br>";
?>
```

Da como resultado:

La función devuelve: Miguel
Pepe
Juan
Julio
Pablo

La función devuelve: Pepe
Juan
Julio
Pablo

2.7.3. La función unset()

Se utiliza para destruir una variable dada. En el caso de los arreglos, se puede utilizar para eliminar una casilla de un Array asociativo (los que no tienen índices numéricos sino que su índice es una cadena de caracteres).

Obsérvese el siguiente código para conocer cómo definir un Array asociativo y eliminar luego una de sus casillas.

```
<?
$estadios_futbol = Array("Barcelona"=> "Nou Camp","Real Madrid" => "Santiago
Bernabeu","Valencia" => "Mestalla","Real Sociedad" => "Anoeta");

//se muestra los estadios
foreach ($estadios_futbol as $indice=>$actual)  echo $indice . " -- " . $actual . "<br>";
echo "<p>";

//Se elimina el estadio asociado al real madrid
unset ($estadios_futbol["Real Madrid"]);

//se muestra los estadios otra vez
foreach ($estadios_futbol as $indice=>$actual) echo $indice . " -- " . $actual . "<br>";
?>
```

La salida será la siguiente:

```
Barcelona -- Nou Camp
Real Madrid -- Santiago Bernabeu
Valencia -- Mestalla
Real Sociedad -- Anoeta

Barcelona -- Nou Camp
Valencia -- Mestalla
Real Sociedad -- Anoeta
```

2.7.4. Aumentar el tamaño de un Arraycon la función array_push()

Inserta al final del Array una serie de casillas que se le indiquen por parámetro. Por tanto, el número de casillas del Array aumentará en tantos elementos como se hayan indicado en el parámetro de la función. Devuelve el número de casillas del Array resultante.

El siguiente código crea un arreglo y añade luego tres nuevos valores.

```
<?
$tabla = Array ("Lagartija", "Araña", "Perro", "Gato", "Ratón");

//se aumenta el tamaño del Array
array_push($tabla, "Gorrión", "Paloma", "Oso");

foreach ($tabla as $actual)  echo $actual . "<br>";
?>
```

Da como resultado esta salida:

```
Lagartija
Araña
Perro
Gato
Ratón
Gorrión
Paloma
Oso
```

2.7.5. La Función array_merge()

Ahora se muestra como unir dos Arrays utilizando la función array_merge(). A ésta se le pasan dos o más Arrays por parámetro y devuelve un arreglo con todos los campos de los vectores pasados.

Este código de ejemplo crea tres Arrays y luego los une con la función array_merge()

```
<?
$tabla = Array ("Lagartija", "Araña", "Perro", "Gato", "Ratón");
$tabla2 = Array ("12","34","45","52","12");
$tabla3 = Array ("Sauce","Pino","Naranjo","Chopo","Perro","34");

//Se aumenta el tamaño del Array
$resultado = array_merge($tabla, $tabla2, $tabla3);

foreach ($resultado as $actual)
echo $actual . "<br>";
?>
```

Da como resultado:

```
Lagartija
Araña
Perro
Gato
Ratón
12
34
45
52
12
Sauce
Pino
Naranjo
Chopo
Perro
34
```

Una última cosa. También pueden introducirse nuevas casillas en un arreglo por los métodos habituales de asignar las nuevas posiciones en el Array a las casillas que se necesiten.

En Arrays normales se haría así:

```
$tabla = Array ("Sauce","Pino","Naranjo");
$tabla[3]="Algarrobo";
```

En Arrays asociativos:

```
$estadios_futbol = Array("Valencia" => "Mestalla","Real Sociedad" => "Anoeta");
$estadios_futbol["Barcelona"]= "Nou Camp";
```

2.8. CADENAS

Una de las variables más corrientes a las que tener que hacer frente en la mayoría de los scripts son las cadenas, que no son más que información de carácter no numérico (textos, por ejemplo).

Para asignar a una variable un contenido de este tipo, ha de escribirse entre comillas dando lugar a declaraciones de este tipo:

```
$cadena="Esta es la información de mi variable"
```

Si se quiere ver en pantalla el valor de una variable o bien un mensaje cualquiera se utilizará la instrucción echo como ya se ha visto anteriormente:

```
echo $cadena //sacaría "Esta es la información de mi variable"
```

```
echo "Esta es la información de mi variable" //daría el mismo resultado
```

Se puede yuxtaponer o concatenar varias cadenas poniendo para ello un punto entre ellas:

```
<?
$cadena1="Perro";
$cadena2=" muerde";
$cadena3=$cadena1.$cadena2;
echo $cadena3 //El resultado es: "Perro muerde"
?>
```

También es posible introducir variables dentro de la cadena lo cual puede ayudar mucho en el desarrollo de los scripts. Lo que se imprime no es el nombre, sino el valor de la variable:

```
<?
$a=55;
$mensaje="Tengo $a años";
echo $mensaje //El resultado es: "Tengo 55 años"
?>
```

La pregunta que ahora puede surgir es, ¿Cómo hacer para que en vez del valor "55" salga el texto "$a"? En otras palabras, cómo se hace para que el símbolo $ no defina una variable sino que sea tomado tal cual. Esta pregunta es tanto más interesante cuanto que en algunos de scripts este símbolo debe ser utilizado por una

simple razón comercial (pago en dólares por ejemplo) y de escribirse tal cual, el ordenador va a pensar que lo que viene detrás es una variable siendo que no lo es.

Pues bien, para meter éste y otros caracteres utilizados por el lenguaje dentro de las cadenas y no confundirlos, lo que hay que hacer es escribir una contra barra delante:

Carácter	Efecto en la cadena
\$	Escribe dólar en la cadena
\"	Escribe comillas en la cadena
\\	Escribe contra barra en la cadena
\8/2	Escribe 8/2 y no 4 en la cadena

Tabla 6.8.1: Caracteres de Escape Especiales en PHP

Además, existen otras utilidades de esta contra barra que permiten introducir en el documento HTML determinados eventos:

Carácter	Efecto en la cadena
\t	Introduce una tabulación en el texto
\n	Cambia de línea
\r	Retorno de carro

Tabla 7.8.2: Caracteres de Escape Comunes

Estos cambios de línea y tabulaciones tienen únicamente efecto en el código y no en el texto ejecutado por el navegador. En otras palabras, si se desea que el texto ejecutado cambie de línea ha de introducirse un echo "<br>" y no echo "\n" ya que este último sólo cambia de línea en el archivo HTML creado y enviado al navegador cuando la página es ejecutada en el servidor. La diferencia entre estos dos elementos

puede ser fácilmente comprendida mirando el código fuente producido al ejecutar este script:

```
<HTML>
<HEAD>
<TITLE>cambiolinea.php</TITLE>
</HEAD>
<BODY>
<?
echo "Hola, \n sigo en la misma línea ejecutada pero no en código fuente.<br>Ahora cambio
de línea ejecutada pero continuo en la misma en el código fuente."
?>
</BODY>
</HTML>
```

El código fuente que se observaría~~seria~~ sería el siguiente:

```
<HTML>
<HEAD>
<TITLE>cambiolinea.php</TITLE>
</HEAD>
<BODY>
Hola,
 sigo en la misma línea ejecutada pero no en código fuente.<br>Ahora cambio de línea
ejecutada pero continuo en la misma en el código fuente.</BODY>
</HTML>
```

2.9. CONTROL DEL FLUJO EN PHP CONDICIÓN IF

La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución.

Como ejemplo, podría hacerse alusión a un script que ejecute una secuencia diferente en función del día de la semana en el que uno se encuentre. Este tipo de acciones pueden ser llevadas a cabo gracias a una paleta de instrucciones presentes en la mayoría de los lenguajes.

Cuando se quiere que el programa, llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución

difieren, se utilizará el conjunto de instrucciones if, else y elseif. La estructura de base de este tipo de instrucciones es la siguiente:

```
if (condición){
   Instrucción 1;
   Instrucción 2;
   ...
}
else {
   Instrucción A;
   Instrucción B;
   ...
}
```

Llegado a este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (else), las instrucciones A y B serán llevadas a cabo.

Esta estructura de base puede complicarse un poco más si se tiene en cuenta que pueden haber muchas más posibilidades por darse. Es por ello que otras condiciones pueden plantearse dentro de la condición principal. Se habla por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
if (condición1){
   Instrucción 1;
   Instrucción 2;
   ...
}
else{
   if (condición2){
      Instrucción A;
      Instrucción B;
      ...
   }
   else {
      Instrucción X
      ...
   }
}
```

De este modo podrían introducirse tantas condiciones como se desee dentro de una condición principal.

2.10. LOS CONTROLADORES DE FLUJO O BUCLES

Los ordenadores, como cualquier máquina, están diseñados para realizar tareas repetitivas. Es por ello que los programas pueden aprovecharse de este principio para realizar una determinada secuencia de instrucciones un cierto número de veces. Para ello, se utilizan las estructuras llamadas en bucle que ayudan a, usando unas pocas líneas, realizar una tarea incluida dentro del bucle un cierto número de veces definido por los programadores. PHP propone varios tipos de bucle cada uno con características específicas:

2.10.1. Bucle While

Sin duda el bucle más utilizado y el más sencillo. Se utiliza para ejecutar las instrucciones contenidas en su interior siempre y cuando la condición definida sea verdadera. La estructura sintáctica es la siguiente.

```
while (condición)
{
  instruccion1;
  instruccion2;
  ...
}
```

Un ejemplo sencillo es este bucle que aumenta el tamaño de la fuente en una unidad a cada nueva vuelta por el bucle:

```
<?
$size=1;
While ($size<=6)
{
  echo"<font size=$size>Tamaño $size</font><br>\n";
  $size++;
}
?>
```

A modo de explicación, antes de nada, ha de definirse el valor de la variable a evaluar en la condición. Algo absolutamente obvio pero fácil de olvidar. En este caso se le ha atribuido un valor de 1 que corresponde a la letra más pequeña.

El paso siguiente es crear el bucle en el que se impone la condición de que la variable no exceda el valor de 6.

La instrucción a ejecutar será imprimir en el documento un código HTML en el que la etiqueta font y el mensaje que contiene varían a medida que $size cambia su valor.

El siguiente paso es incrementar en una unidad el valor de $size. Esto se puede hacer con una expresión como la mostrada en el bucle ($size++) que en realidad es sinónima de:

```
$size=$size+1
```

Otro ejemplo del bucle While

El bucle while se suele utiliza cuando no se sabe exactamente cuántas iteraciones se deben realizar antes de acabar. Ahora se utilizará en otro ejemplo, en el que hay que recorrer una cadena hasta encontrar un carácter dado. Si lo encuentra, escribir su posición. Si no, escribir que no se ha encontrado.

Nota: Para hacer este ejercicio se necesita conocer la función de cadena strlen(), que obtiene la longitud de la cadena que se le pase por parámetro.

int strlen (string cad)
Devuelve un entero igual a la longitud de la cadena.

```
<?
$cadena = "hola a todo el mundo";

//recorre la cadena hasta encontrar una "m"
$i=0;
while ($cadena[$i]!="m" && $i< strlen($cadena)){
   $i++;
}

if ($i==strlen($cadena)) echo "No se encuentra...";
else echo "Está en la posición $i";
?>
```

En este ejemplo se define una cadena con el valor "hola a todo el mundo". Posteriormente se recorre esa cadena hasta el final de la cadena o hasta encontrar el caracter "m", utilizando una variable $i que lleva la cuenta de los caracteres recorridos.

Al final del bucle while, si se salió porque se encontró el caracter "m", la variable $i valdrá un número menor que la longitud de la cadena. Si se salió por llegar al final de la cadena, la variable $i valdrá lo mismo que la longitud en caracteres de esa cadena. En el condicional simplemente se comprueba si $i vale o no lo mismo que la longitud de la cadena, mostrando los mensajes adecuados en cada caso.

2.10.1. Bucle do/while

Este tipo de bucle no difiere en exceso del anterior. La sintaxis es la siguiente:

```
do
{
  instruccion1;
  instruccion2;
  ...
}
while (condición)
```

La diferencia con respecto a los bucles while es que este tipo de bucle evalúa la condición al final con lo que, incluso siendo falsa desde el principio, éste se ejecuta al menos una vez.

2.10.2. Bucle for

PHP está provisto de otros tipos de bucle que también resultan muy prácticos en determinadas situaciones. El más popular de ellos es el bucle for que, como para los casos anteriores, se encarga de ejecutar las instrucciones entre llaves. La diferencia con los anteriores radica en cómo se plantea la condición de finalización del bucle. Para aclarar su funcionamiento se va a expresar el ejemplo de bucle while visto en el capítulo anterior en forma de bucle for:

```
<?
For ($size=1;$size<=6;$size++)
{
  echo"<font size=$size>Tamaño $size</font><br>\n";
}
?>
```

Las expresiones dentro del paréntesis definen respectivamente:

- Inicialización de la variable. Válida para la primera vuelta del bucle.
- Condición de evaluación a cada vuelta. Si es cierta, el bucle continúa.
- Acción a realizar al final de cada vuelta de bucle.

2.10.3. Bucle foreach

Este bucle, implementado en las versiones de PHP4 o superior, ayuda a recorrer los valores de un Array lo cual puede resultar muy útil por ejemplo para efectuar una lectura rápida del mismo. Si se toma en cuenta que un Array es una variable que guarda un conjunto de elementos (valores) catalogados por claves.

La estructura general es la siguiente:

```
Foreach ($Array as $clave=>$valor)
{
  instruccion1;
  instruccion2;
  …;
}
```

Un ejemplo práctico es la lectura de un Array lo cual podría hacerse del siguiente modo:

```
<?
$moneda=Array("España"=> "Peseta","Francia" => "Franco","USA" => "Dolar");
Foreach ($moneda as $clave=>$valor)
{
  echo "Pais: $clave Moneda: $valor<br>";
}?>
```

Este script se encargaría de mostrar por pantalla el contenido del Array$moneda. No resultaría mala idea crear una función propia basada en este bucle que permitiera visualizar Arraysmono dimensionales y almacenarla en la librería. Esta función podría ser definida de esta forma:

```
Function mostrar_array ($Array)
{
Foreach ($Array as $clave=>$valor)
{echo "$clave=>$valor<br>";}
}
```

2.10.4. Sentencias de Salida de Bucle break y continue

Estas dos instrucciones se introducen dentro de la estructura y sirven respectivamente para escapar del bucle o saltar a la iteración siguiente. Pueden resultar muy prácticas en algunas situaciones.

2.11. FUNCIONES EN PHP

Una función podría ser definida como un conjunto de instrucciones que explotan ciertas variables para realizar una tarea más o menos elemental.

PHP basa su eficacia principalmente en este tipo de elemento. Una gran librería que crece constantemente, a medida que nuevas versiones van surgiendo, es complementada con las funciones de propia cosecha dando como resultado un sinfín de recursos que son aplicados por una simple llamada.

Las funciones integradas en PHP son muy fáciles de utilizar. Tan sólo ha de realizarse la llamada de la forma apropiada y especificar los parámetros y/o variables necesarios para que la función realice su tarea.

Lo que puede parecer ligeramente más complicado, pero que resulta sin lugar a dudas muy práctico, es crear funciones propias. De una forma general, podría crearse propias funciones para conectarse a una base de datos o crear los encabezados o

etiquetas meta de un documento HTML. Para una aplicación de comercio electrónico podrían crearse por ejemplo funciones de cambio de una moneda a otra o de cálculo de los impuestos a añadir al precio de artículo. En definitiva, es interesante crear funciones para la mayoría de acciones más o menos sistemáticas que se realizan en los programas. Aquí se daráun ejemplo de creación de una función que, llamada al comienzo del script, crea el encabezado del documento HTML y coloca el título que deseado a la página:

```
<?
function hacer_encabezado($titulo)
{
$encabezado="<html>\n<head>\n\t<title>$titulo</title>\n</head>\n";
echo $encabezado;
}
?>
```

Esta función podría ser llamada al principio de todas las páginas de la siguiente forma:

```
$titulo="Mi web";
hacer_encabezado($titulo);
```

De esta forma se automatiza el proceso de creación del documento. Podría por ejemplo incluirse en la función otras variables que ayudasen a construir las etiquetas meta y de esta forma, con un esfuerzo mínimo, se crearían los encabezados personalizados para cada una de las páginas. De este mismo modo es posible crear cierres de documento o formatos diversos para textos como si se tratase de hojas de estilo que tendrían la ventaja de ser reconocidas por todos los navegadores.

Por supuesto, la función ha de ser definida dentro del script ya que no se encuentra integrada en PHP sino que la ha creado el programador. Esto en realidad no pone ningún impedimento ya que puede ser incluida desde un archivo en el que se irán almacenando las definiciones de las funciones que se vayan creando o recopilando.

Estos archivos en los que se guardan las funciones se llaman librerías. La forma de incluirlos en el script es a partir de la instrucción require o include:

```
require("libreria.php") o include("libreria.php")
```

En resumen, el código quedaría así:

Se tendría un archivo libreria.php como sigue:

```
<?
//función de encabezado y colocación del título
Function hacer_encabezado($titulo)
{
$encabezado="<html>\n<head>\n\t<title>$titulo</title>\n</head>\n";
echo $encabezado;
}
?>
```

Por otra parte se tendría en el script principal página.php (por ejemplo):

```
<?
include("libreria.php");
$titulo="Mi Web";
hacer_encabezado($titulo);
?>
<body>
El cuerpo de la página
</body>
</html>
```

Pueden meterse todas las funciones que se vayan encontrando dentro de un mismo archivo pero resulta muchísimo más ventajoso ir clasificándolas en distintos archivos por temática: Funciones de conexión a bases de datos, funciones comerciales, funciones generales, etc. Esto ayudara a poder localizarlas antes para corregirlas o modificarlas, esto permite también cargar únicamente el tipo de función que requerido para el script sin recargar éste en exceso además de que permite utilizar un determinado tipo de librería para varios sitios webs distintos.

También puede resultar muy práctico el utilizar una nomenclatura sistemática a la hora de nombrarlas: Las funciones comerciales podrían ser llamadas com_loquesea, las de bases de datos bd_loquesea, las de tratamiento de archivos file_loquesea. Esto permitirá reconocerlas enseguida cuando se lea el script sin tener que recurrir a una oxidada memoria para descubrir su utilidad.

No obstante, antes de lanzarse a crear una función propia, merece la pena echar un vistazo a la documentación para ver si dicha función ya existe o poder aprovechar alguna de las existentes para aligerar el trabajo. Así, por ejemplo, existe una función llamada header que crea un encabezado HTML configurable lo cual evita tener que crearla uno mismo.

Ejemplo de función

El siguiente es un ejemplo de creación de funciones en PHP. Se trata de hacer una función que recibe un texto y lo escribe en la página con cada carácter separado por "-". Es decir, si recibe "hola" debe escribir "h-o-l-a" en la página web. La manera de realizar esta función será recorrer el string, caracter a caracter, para imprimir cada uno de los caracteres, seguido del signo "-". Se recorre el string con un bucle for, desde el caracter 0 hasta el número de caracteres total de la cadena. El número de caracteres de una cadena se obtiene con la función predefinida en PHP strlen(), que recibe el string entre paréntesis y devuelve el número de los caracteres que tenga.

```
<html>
<head>
   <title>funcion 1</title>
</head>
<body>
<?
function escribe_separa($cadena){
   for ($i=0;$i<strlen($cadena);$i++){
      echo $cadena[$i];
      if ($i<strlen($cadena)-1)
         echo "-";
   }
}
```

```
escribe_separa ("hola");
echo "<p>";
escribe_separa ("Texto más largo, a ver lo que hace");
?>
</body>
</html>
```

La función creada se llama escribe_separa y recibe como parámetro la cadena que hay que escribir con el separador "-". El bucle for sirve para recorrer la cadena, desde el primer al último caracter. Luego, dentro del bucle, se imprime cada caracter separado del signo "-". El if que hay dentro del bucle for comprueba que el actual no sea el último caracter, porque en ese caso no habría que escribir el signo "-" (se quiere conseguir "h-o-l-a" y si no estuviera el if se obtendría "h-o-l-a-").

En el código mostrado se hacen un par de llamadas a la función para ver el resultado obtenido con diferentes cadenas como parámetro.

2.12. PASO DE PARÁMETROS A FUNCIONES

A continuación se van a explicar algunos detalles adicionales sobre la definición y uso de funciones, para ampliar el concepto de las funciones.

Los parámetros son los datos que reciben las funciones y que utilizan para realizar las operaciones de la función. Una función puede recibir cualquier número de parámetros, incluso ninguno. A la hora de definir la función, en la cabecera, se definen los parámetros que va a recibir.

```
function f1 ($parametro1, $parámetro2)
```

Así se define una función llamada f1 que recibe dos parámetros. Como se puede observar, no se tiene que definir el tipo de datos de cada parámetro.

Los parámetros tienen validez durante la ejecución de la función, es decir, tienen un ámbito local a la función donde se están recibiendo. Cuando la función se termina, los parámetros dejan de existir.

2.12.1. Paso de Parámetros por Valor

El paso de parámetros en PHP se realiza por valor. "Por valor" es una manera típica de pasar parámetros en funciones, quiere decir que el cambio de un dato de un parámetro no actualiza el dato de la variable que se pasó a la función. Por ejemplo, cuando se invoca una función pasando una variable como parámetro, a pesar de que se cambie el valor del parámetro dentro de la función, la variable original no se ve afectada por ese cambio. Puede que se vea mejor con un ejemplo:

```
function porvalor ($parametro1){
$parametro1="hola";
echo "<br>" . $parametro1; //imprime "hola"
}

$mivariable = "esto no cambia";
porvalor ($mivariable);
echo "<br>" . $mivariable; //imprime "esto no cambia"
```

Esta página tendrá como resultado:

```
hola
esto no cambia
```

2.12.2. Paso de Parámetros por Referencia

En contraposición al paso de parámetros por valor, está el paso de parámetros por referencia. En este último caso, el cambio del valor de un parámetro dentro de una función sí afecta al valor de la variable original.

Se pueden pasar los parámetros por referencia si, en la declaración de la función, se coloca un "&" antes del parámetro.

```
<?
function porreferencia(&$cadena)
{
$cadena = 'Si cambia';
}
$str = 'Esto es una cadena';
porreferencia ($str);
echo $str; // Imprime 'Si cambia'
?>
```

Este script mostrará por pantalla 'Si cambia'.

2.12.3. Paso de Parámetros por Defecto

Se pueden definir valores por defecto para los parámetros. Los valores por defecto sirven para que los parámetros contengan un dato predefinido, con el que se inicializarán si no se le pasa ningún valor en la llamada de la función. Los valores por defecto se definen asignando un dato al parámetro al declararlo en la función.

```
function pordefecto ($parametro1="pepe";$parametro2=3)
```

Para la definición de función anterior, $parametro1 tiene como valor por defecto "pepe", mientras que $parametro2 tiene 3 como valor por defecto.

Si se llama a la función sin indicar valores a los parámetros, estos tomarán los valores asignados por defecto:

```
pordefecto () // $parametro1 vale "pepe" y $parametro2 vale 3
```

Si se llamaa la función indicando un valor, este será tenido en cuenta para el primer parámetro.

```
pordefecto ("hola") // $parametro1 vale "hola" y $parametro2 vale 3
```

Atención, es una obligación declarar todos los parámetros con valores por defecto al final.

2.13. PASO DE VARIABLES POR LA URL

Para pasar las variables de una página a otra se lo puede hacer introduciendo dicha variable dentro del enlace hipertexto de la página destino. La sintaxis sería la siguiente:

```
<a href="destino.php?variable1=valor1&variable2=valor2&...">Mi enlace</a>
```

Se puede observar que estas variables no poseen el símbolo $ delante. Esto es debido a que en realidad este modo de pasar variables no es específico de PHP sino que es utilizado por otros lenguajes.

Ahora la variable pertenece también al entorno de la página destino.php y está lista para su explotación.

Nota: No siempre se definen automáticamente las variables recibidas por parámetro en las páginas web, depende de una variable de configuración de PHP: register_globals, que tiene que estar activada para que así sea.

Para aclarar posibles dudas, ver esto en forma de ejemplo. Se tienen dos páginas, origen.html (no es necesario darle extensión PHP puesto que no hay ningún tipo de código) y destino.php:

Origen.html

```
<HTML>
<HEAD>
<TITLE>origen.html</TITLE>
</HEAD>
<BODY>
<a href="destino.php?saludo=hola&texto=Esto es una variable texto">Paso variables saludo
y texto a la página destino.php</a>
</BODY>
</HTML>
```

Destino.php

```
<HTML>
<HEAD>
<TITLE>destino.php</TITLE>
</HEAD>
<BODY>
<?
echo "Variable \$saludo: $saludo <br>\n";
echo "Variable \$texto: $texto <br>\n"
?>
</BODY>
</HTML>
```

$HTTP_GET_VARS

Recordando que es posible recopilar en una variable tipo Array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema $HTTP_GET_VARS, que es un Array asociativo. Utilizándolo quedaría así:

```
<?
echo "Variable \$saludo: $HTTP_GET_VARS["saludo"] <br>\n";
echo "Variable \$texto: $HTTP_GET_VARS["texto"] <br>\n"
?>
```

> **Nota:** Aunque se pueden recoger variables con este Array asociativo o utilizar directamente las variables que se definen en la página, resulta más seguro utilizar $HTTP_GET_VARS por dos razones, la primera que así hay seguridad de que esa variable viene realmente de la URL y la segunda, que así el código será más claro cuando se vuelva a leer, porque quedará especificado que esa variable está siendo recibida por la URL.

$_GET

A partir de la versión 4.1.0 de PHP se ha introducido el Array asociativo $_GET, que es idéntico a $HTTP_GET_VARS, aunque un poco más corto de escribir.

Caracteres especiales en URL y su codificación con PHP

Hay algunos caracteres raros que no se pueden pasar, tal cual, por la URL. Por ejemplo, una URL no puede contener espacios en blanco, por lo que si se intenta enviar una variable por URL con un valor que tiene un espacio en blanco, dará problemas. Por ejemplo, el signo "*" no puede figurar tampoco en una URL. Así pues, hay que hacer algo para convertir esos caracteres, de modo que no den problemas en la URL.

La solución en PHP es sencilla, simplemente se debe codificar la variable que tiene caracteres conflictivos a formato URL. Para ello es conveniente utilizar la función urlenconde(), que viene en la librería de funciones de PHP.

2.14. PROCESAMIENTO DE VARIABLES DE FORMULARIO

Este tipo de transferencia es de gran utilidad ya que permite interaccionar directamente con el usuario.

El proceso es similar al explicado para las URLs. Primeramente, se presenta una primera página con el formulario clásico a rellenar y las variables son recogidas en una segunda página que las procesa:

Nota:

No siempre se definen automáticamente las variables recibidas por el formulario en las páginas web, depende de una variable de configuración de PHP: register_globals, que tiene que estar activada para que así sea.

```
<HTML>
<HEAD>
<TITLE>formulario.html</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="destino2.php">
Nombre<br><INPUT TYPE="TEXT" NAME="nombre"><br>
Apellidos<br><INPUT TYPE="TEXT" NAME="apellidos"><br>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>destino2.php</TITLE>
</HEAD>
<BODY>
<?
echo "Variable \$nombre: $nombre <br>\n";
echo "Variable \$apellidos: $apellidos <br>\n"
?>
</BODY></HTML>
```

$HTTP_POST_VARS

Recordar que es posible recopilar en una variable tipo Array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema $HTTP_POST_VARS.

```
echo "Variable \$nombre: " . $HTTP_POST_VARS["nombre"] . "<br>\n";
```

> **Nota:** Aunque pueda recoger variables con este Array asociativo o utilizar directamente las variables que se definen en la página, resulta más seguro utilizar $HTTP_POST_VARS por dos razones, la primera que así habrá de asegurarse que esa variable viene realmente de un formulario y la segunda, que así el código será más claro cuando se lo vuelva a leer, porque quedará especificado que esa variable se la estárecibiendo por un formulario.

$_POST

A partir de PHP 4.1.0 se pueden recoger las variables de formulario utilizando también el Array asociativo $_POST, que es el mismo que $HTTP_POST_VARS, pero más corto de escribir.

Ejemplo de restricción de acceso por edad

Para continuar aportando ejemplos al uso de formularios se realizará una página que muestra solicita la edad del visitante y, dependiendo de dicha edad,

permita o no visualizar el contenido de la web. A los mayores de 18 años se les permite ver la página y a los menores no.

El ejemplo es muy sencillo y no valdría tal cual está para utilizarlo a modo de una verdadera restricción de acceso. Únicamente sirve para saber cómo obtener datos de un formulario y como tratarlos para realizar una u otra acción, dependiendo de su valor.

La página del formulario, edad.php tendría esta forma:

```
<html>
<head>
<title>Restringir por edad</title>
</head>
<body>
<form action="edad2.php" method="post">
Escribe tu edad: <input type="text" name="edad" size="2">
<input type="submit" value="Entrar">
</form>
</body>
</html>
```

Esta es una página sin ningún código PHP, simplemente tiene un formulario. Fijarse en el action del formulario, que está dirigido hacia una página llamada edad2.php, que es la que recibirá el dato de la(e) no entiendo mostrará un contenido u otro dependiendo de ese valor. Su código es el siguiente:

```
<html>
<head>
  <title>Restringir por edad</title>
</head>
<body>
<?
$edad = $_POST["edad"];
echo "Tu edad: $edad<p>";
if ($edad < 18) {
  echo "No puede entrar";
}else{
  echo "Bienvenido";
}
?>
</body>
</html>
```

Simplemente se recibe la edad, utilizando el Array $_POST. Luego se muestra la edad y se ejecuta una expresión condicional en función de que la edad sea menor que 18. En caso positivo (edad menor que 18), se muestra un mensaje que informa de que no se deja acceder al página. En caso negativo (mayor o igual a 18) se muestra un mensaje de bienvenida.

2.15. AUTO LLAMADA DE PÁGINAS

Al incluir un formulario en una página se debe indicar, a través del atributo action, el nombre del archivo PHP al que se enviarán los datos escritos en el formulario. De este modo, para un esquema de envío de datos por formulario, pueden participar dos páginas: una que contiene el formulario y otra que recibe los datos de dicho formulario.

Lo mismo ocurre cuando se envían variables por una URL. Se tiene una página que contendrá el enlace y otra página que recibirá y tratará esos datos para mostrar unos resultados.

Ahora se va a ver cómo enviar y recibir datos de un formulario con una única página. Asimismo, se verá como en la misma página es posible tener enlaces con paso de variables por URL y además, se puede recoger y tratar esos datos con la misma página. A este efecto se le puede llamar "autollamada de páginas", también se le suele llamar como "Formularios reentrantes" o términos similares. Es muy interesante conocer el modo de funcionamiento de estos scripts, porque serán muy habituales en las páginas PHP y ayudan mucho a tener los códigos ordenados.

En ambos casos, para formularios o envío de datos por la URL, se debe seguir un esquema como este:

- Comprobar si se reciben datos por URL o por formulario
- Si no se reciben datos
- Mostrar el formulario o los enlaces que pasan variables.
- Si se reciben datos
- Entonces procesar el formulario o las variables de la URL

2.15.1. Autollamadas para un Formulario

Véase a continuación como sería el código de un formulario reentrante.

```
<html>
<head>
   <title>Me llamo a mi mismo...</title>
</head>

<body>
<?
if (!$_POST){
?>
   <form action="auto-llamada.php" method="post">
   Nombre: <input type="text" name="nombre" size="30">
   <br>
   Empresa: <input type="text" name="empresa" size="30">
   <br>
   Telefono: <input type="text" name="telefono" size=14 value="+34 " >
   <br>
   <input type="submit" value="Enviar">
   </form>
<?
}else{
   echo "<br>Su nombre: " . $_POST["nombre"];
   echo "<br>Su empresa: " . $_POST["empresa"];
   echo "<br>Su Teléfono: " . $_POST["telefono"];
}
?>
</body>
</html>
```

En el ejemplo, el primer paso es conocer si se están recibiendo o no datos por un formulario. Para ello se comprueba con un enunciado if si existe o no una variable $_POST.

En concreto if (!$_POST) querría decir algo como "Si no existen datos venidos de un formulario". En caso de que no existan, muestra el formulario. En caso de que sí existan, recoge los datos y los imprimo en la página.

2.15.2. Autollamadas para Paso de Variables por URL

La idea es la misma. Comprobar con un enunciado if si se reciben o no datos desde una URL. Si se ve el código a continuación, se trata de una página que muestra

una serie de enlaces para ver las tablas de multiplicar del 1 hasta el 10. Cada uno de los enlaces muestra una tabla de multiplicar. Pulsando el primer enlace se puede ver la tabla del 1, pulsando el segundo la tabla del 2, etc. Recordar que la página se llama a sí misma.

```
<html>
<head>    <title>Tablas de multiplicar</title>
</head>
<body>
<?
if (!$_GET){
   for ($i=1;$i<=10;$i++){
      echo "<br><a href='ver_tabla.php?tabla=$i'>Ver la tabla del $i</a>\n";
   }
} else {
   $tabla=$_GET["tabla"];
?>
   <table align=center border=1 cellpadding="1">
<?
   for ($i=0;$i<=10;$i++){
      echo "<tr><td>$tabla X $i</td><td>=</td><td>" . $tabla * $i . "</td></tr>\n";
   }
?>
   </table>
<?
}
?>
</body>
</html>
```

Este código es un poco más complicado, porque hace un poco más de cosas que el anterior, pero para el asunto en cuestión que es la autollamada de páginas, todo sigue igual de simple.

Hay que fijarse en el if que comprueba si se reciben o no datos por URL: if (!$_GET), que querría decir algo como "Si no se reciben variables por la URL".

En caso positivo (no se reciben datos por URL) se muestran los enlaces para ver cada una de las tablas y en caso de que sí se reciban datos, se muestra la tabla de multiplicar del número que se está recibiendo en la URL.

2.16. PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando clases, que son un conjunto de datos y funcionalidades. Las clases son definiciones, a partir de las que se crean objetos. Los objetos son ejemplares de una clase determinada y como tal, disponen de los datos y funcionalidades definidos en la clase.

La programación orientada a objetos permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la programación orientada a objetos como paradigma para modelizar los sistemas. Prueba de ello es la nueva versión de PHP (5), que implanta la programación de objetos como metodología de desarrollo. También Microsoft ha dado un vuelco hacia la programación orientada a objetos, ya que .NET dispone de varios lenguajes para programar y todos orientados a objetos.

2.16.1. Las clases

Una clase es un conjunto de variables, llamados atributos, y funciones, llamadas métodos, que trabajan sobre esas variables. Las clases son, al fin y al cabo, una definición: una especificación de propiedades y funcionalidades de elementos que van a participar en los programas.
Por ejemplo, la clase "Caja" tendría como atributos características como las dimensiones, color, contenido y cosas semejantes. Las funciones o métodos que se podrían incorporar a la clase "caja" son las funcionalidades que se desea que realice la caja, como introduce(), muestra_contenido(), comprueba_si_cabe(), vaciate()...

Las clases en PHP se definen de la siguiente manera:

```
<?
class Caja{
   var $alto;
   var $ancho;
   var $largo;
   var $contenido;
   var $color;
```

```
function introduce($cosa){
   $this->contenido = $cosa;
}

function muestra_contenido(){
   echo $this->contenido;
} }?>
```

En este ejemplo se ha creado la clase Caja, indicando como atributos el ancho, alto y largo de la caja, así como el color y el contenido. Se han creado, para empezar, un par de métodos, uno para introducir un elemento en la caja y otro para mostrar el contenido. Hay que notar que los atributos se definen declarando unas variables al principio de la clase. Los métodos se definen declarando funciones dentro de la clase. La variable $this, utilizada dentro de los métodos se explicará más adelante.

2.16.2. Utilizar la clase

Las clases solamente son definiciones. Si se quiere utilizar la clase hay que crear un ejemplar de dicha clase, lo que corrientemente se le llama instanciar un objeto de una clase.

```
$micaja = new Caja;
```

Con esto se ha creado, o mejor dicho, instanciado, un objeto de la clase Caja llamado $micaja.

```
$micaja->introduce("algo");
$micaja->muestra_contenido();
```

Con estas dos sentencias se introduce "algo" en la caja y luego se muestra ese contendido en el texto de la página. Nótese que los métodos de un objeto se llaman utilizando el código "->".

```
nombre_del_objeto->nombre_de_metodo()
```

Para acceder a los atributos de una clase también se accede con el código "->". De esta forma:

```
nombre_del_objeto->nombre_del_atributo
```

2.16.3. La variable $this

Dentro de un método, la variable $this hace referencia al objeto sobre el que se invoca el método. En la invocación $micaja->introduce("algo") se está llamando al método introduce sobre el objeto $micaja. Cuando se está ejecutando ese método, se vuelca el valor que recibe por parámetro en el atributo contenido. En ese caso $this->contenido hace referencia al atributo contenido del objeto $micaja, que es sobre el que se invocaba el método.

2.16.4. Constructores en PHP

Los constructores son funciones, o métodos, que se encargan de realizar las tareas de inicialización de los objetos al ser instanciados. Es decir, cuando se crean los objetos a partir de las clases, se llama a un constructor que se encarga de inicializar los atributos del objeto y realizar cualquier otra tarea de inicialización que sea necesaria.

No es obligatorio disponer de un constructor, pero resultan muy útiles y su uso es muy habitual. En el ejemplo de la caja, comentado en la sección anterior, lo normal sería inicializar las variables como color o las relacionadas con las dimensiones y, además, indicar que el contenido de la caja está vacío. Si no hay un constructor no se inicializan ninguno de los atributos de los objetos.

El constructor se define dentro de la propia clase, como si fuera otro método. El único detalle es que el constructor debe tener el mismo nombre que la clase. Hay que tomar en cuenta que PHP diferencia entre mayúsculas y minúsculas.

Para la clase Caja definida anteriormente, se podría declarar este constructor:

```
function Caja($alto=1,$ancho=1,$largo=1,$color="negro"){
   $this->alto=$alto;
   $this->ancho=$ancho;
   $this->largo=$largo;
   $this->color=$color;
   $this->contenido="";
}
```

En este constructor se recibe por parámetro todos los atributos que hay que definir en una caja.

Es muy útil definir unos valores por defecto en los parámetros que recibe el constructor, igualando el parámetro a un valor dentro de la declaración de parámetros de la función constructora, pues así, aunque se llame al constructor sin proporcionar parámetros, se inicializará con los valores por defecto que se hayan definido.

Es importante señalar que en los constructores no se tienen por qué recibir todos los valores para inicializar el objeto. Hay algunos valores que pueden inicializarse a vacío o a cualquier otro valor fijo, como en este caso el contenido de la caja, que inicialmente se supone que estará vacía.

2.16.5. Herencia en PHP

La programación orientada a objetos tiene un mecanismo llamado herencia por el que se pueden definir clases a partir de otras clases. Las clases realizadas a partir de otra clase o mejor dicho, que extienden a otra clase, se llaman clases extendidas o clases derivadas.

Las clases extendidas heredan todos los atributos y métodos de la clase base. Además, pueden tener tantos atributos y métodos nuevos como se desee.

Para ampliar el ejemplo que se ha venido desarrollando, la clase Caja, se va a crear una clase extendida llamada Caja_tematica. Esta clase hereda de caja, pero además tiene un "tema", que es la descripción del tipo de cosas que se insertará en la caja. Con esto se puede tener varias cajas, cada una con cosas de un tema concreto.

```
class Caja_tematica extends Caja{
  var $tema;
  function define_tema($nuevo_tema){
  $this->tema = $nuevo_tema;
  } }
```

En esta clase se hereda de Caja, con lo que se tiene a disposición todos los atributos y métodos de la clase base. Además, se ha definido un nuevo atributo, llamado $tema, y un método, llamado define_tema(), que recibe el tema con el que se desea etiquetar la caja.

Se podría utilizar la clase Caja_tematica de manera similar a como lo se hacía con la clase Caja original.

```
$micaja_tematica = new Caja_tematica();
$micaja_tematica->define_tema("Cables y contectores");
$micaja_tematica->introduce("Cable de red");
$micaja_tematica->introduce("Conector RJ45");
$micaja_tematica->muestra_contenido();
```

En este caso, el resultado que se obtiene es parecido al que se obtiene para la clase base. Sin embargo, cuando se muestra el contenido de una caja, lo más interesante sería que se indicara también el tipo de objetos que contiene la caja temática. Para ello, se tendría que redefinir el método muestra_contenido().

2.16.6. Redefinir métodos en clases extendidas

Redefinir métodos significa volver a codificarlos, es decir, volver a escribir su código para la clase extendida. En este caso, habrá de redefinirse el método muestra_contenido() para que muestre también el tema de la caja.

Para redefinir un método, lo único que hay que hacer es volverlo a escribir dentro de la clase extendida.

```
function muestra_contenido(){
  echo "Contenido de la caja de <b>" . $this->tema . "</b>: " . $this->contenido;
}
```

En este ejemplo se ha codificado de nuevo el método entero para mostrar los datos completos.

En algunas ocasiones es muy útil apoyarse en la definición de un método de la clase base para realizar las acciones de la clase extendida. Por ejemplo, hay que definir un constructor para la clase Caja_tematica, en el que también se inicialice el tema de la caja. Como ya existe un método constructor en la clase base, no merece la pena reescribir el código de éste, lo mejor es llamar al constructor que se había definido en la clase Caja original, con lo que se inicializarán todos los datos de la clase base, y luego realizar la inicialización para los atributos de la propia clase extendida.

Para llamar a un método de la clase padre dentro del código de un método que se está redefiniendo, se utiliza una sintaxis como esta:

```
function Caja_tematica($alto=1,$ancho=1,$largo=1,$color="negro",$tema="Sin
clasificación"){
  parent::Caja($alto,$ancho,$largo,$color);
  $this->tema=$tema;
}
```

Aquí se ve la redefinición del constructor, de la clase Caja, para la clase Caja_tematica. El constructor hace primero una llamada al constructor de la clase base, a través de una referencia a "parent". Luego inicializa el valor del atributo $tema, que es específico de la Caja_tematica.

En la misma línea de trabajo, se puede redefinir el método muestra_contenido() apoyándose en el que fue declarado en la clase base. El código quedaría como sigue:

```
function muestra_contenido(){
  echo "Contenido de la caja de <b>" . $this->tema . "</b>: ";
  parent::muestra_contenido();
}
```

2.17. CONEXIÓN A LA BASE DE DATOS ORACLE

En PHP, se han implementado funciones para poder acceder a la Base de Datos Oracle, en cualquiera de sus versiones, a continuación se listan 7 funciones clave para la implementación rápida y sencilla de una interfaz de comunicación.

Para poder utilizar dichas funciones la extensión OCI8 debe estar habilitada y configurada debidamente en el servidor web

Para poder consultar o manipular la información mediante una conexión a Oracle, el proceso se divide en 6 pasos, los cuales se detallan a continuación:

2.17.1. Abrir una Conexión Oracle

Para ello es necesario definir una variable que representará la conexión en sí, para ello se procede a llamar a la función de la biblioteca OCI8, pasando los parámetros necesarios de la siguiente manera:

```
$v_servidor = "localhost/orcl";
$v_usuario = "mi_usuario";
$v_clave = "mi_clave";
$conexion = oci_connect($this->v_usuario, $this->v_clave, $this->v_servidor);
```

En este caso la función oci_connect requiere 3 parámetros, el primero es el nombre del usuario, el segundo es el password y le tercero un String que contiene el nombre del servidor y la cadena de conexión concatenados pero separados por un “/”, en caso de ser correcta la conexión se generará una instancia y posteriormente será asignada a la variable $conexion

2.17.2. Definir la Instrucción SQL

En este paso el objetivo es plantear la instrucción sql a ejecutar, ya sea esta de DML o DDL, para ello se utilizará una variable de tipo cadena. Luego de haber definido el

SQL a ejecutar, se debe proceder a validarlo mediante el parser de Oracle con la función oci_parse, como en el siguiente ejemplo:

```
$v_sql = "select * from solicitudes";
$v_consulta = oci_parse($conexion, $v_sql);
```

La función oci_parse recibe 2 parámetros, el primero es el objeto conexión, creado en el paso 1, y el segundo parámetro es la instrucción SQL a ejecutar. Esta función retorna un objeto que representa la instrucción SQL validada para poder ser ejecutada, este es almacenado en la variable $v_consulta.

2.17.3. Ejecutar la instrucción SQL

Una vez validada la instrucción SQL, el siguiente paso es ejecutarla, para ello basta hacer un llamado a la función oci_execute de la siguiente forma:

```
oci_execute($v_consulta);
```

Luego de haber llamado a la función oci_execute, el valor de $v_consulta, es sobrescrito con un juego de registros al que se accederá en el siguiente paso

2.17.4. Acceder a los Registros de una Consulta

Dado que ahora la variable $v_consulta es un Array con los registros generados por la consulta, se necesita un bucle para poder acceder a ellos, pero adicionalmente se requiere una función capaz de recuperar cada registro uno a uno en cada interacción del bucle, para ello se plantea el siguiente código:

```
while (($a_resultados = oci_fetch_array($v_consulta, OCI_BOTH)))
{
}
```

En este fragmento de código se puede notar la llamada a la función oci_fetch_array, que recibe 2 parámetros, el primero es el juego de registros origen, es decir, para este caso la variable $v_consulta, el segundo parámetro es muy importante, ya que define aspectos de la información que se extraerá del juego de registros, los valores que pueden utilizarse para este parámetro son:

OCI_ASSOC: Retorna un Array Asociativo, es decir un Array que trabaja con Keys para acceder a sus elementos
OCI_NUM: Retorna un Array normal con el valor de cada campo de la consulta, para acceder a cada uno se utilizará su índice respectivo
OCI_BOTH: Retorna un Array que combina tanto índices como llaves para poder acceder a los campos de la consulta.
OCI_RETURN_NULLS: Funciona igual que OCI_NUM, pero en caso de no existir registros genera valores nulos para cada campo y evita un valor nulo entero para el registro recuperado.

Continuando con la descripción del código, la función oci_fetch_array generará un Array con las características definidas según el uso de alguna de las constantes definidas mencionadas y dicho Array será almacenado en la variable $a_resultados.

2.17.5. Recuperar el valor de cada Campo de la Consulta

Ahora el objetivo es recuperar el valor de cada campo, para ello con el siguiente código, en el que se ha utilizado la constante OCI_BOTH como segundo parámetro de la función, por lo tanto se podrá navegar por los campos ya sea mediante el índice del campo o mediante su llave:

```
while (($a_resultados = oci_fetch_array($v_consulta, OCI_BOTH)))
{
        echo "Codigo: ".$a_resultados[0];
        echo "Descripcion: ".$a_resultados['SOL_DESCRIPCION'];
        echo "Detalle: ".$a_resultados['SOL_DETALLE'];
}
```

2.17.6. Cerrar la Conexión

El último paso es cerrar la conexión luego de haber realizado todo lo deseado con la base de datos, para ellos se utilizará la instrucción:

```
oci_close($conexion);
```

Esta función requiereuna variable que represente una conexión hacia Oracle.

El código completo para la consulta que se ha estado explicando sería el siguiente el siguiente:

```
$v_servidor = "localhost/orcl";
$v_usuario = "mi_usuario";
$v_clave = "mi_clave";
$conexion = oci_connect($v_usuario, $v_clave, $v_servidor);

$v_sql = "select * from kim_solicitud";
$v_consulta = oci_parse($conexion, $v_sql);

oci_execute($v_consulta);

while (($a_resultados = oci_fetch_array($v_consulta, OCI_BOTH)))
{
        echo "Codigo: ".$a_resultados[0];
        echo "Descripcion: ".$a_resultados['SOL_DESCRIPCION'];
        echo "Detalle: ".$a_resultados['SOL_DETALLE'];
}
```

CAPITULO III

ADOBE FLASH CS4

Flash es una tecnología que permite la creación de animaciones vectoriales. Está orientada a aportar vistosidad a un sitio web al mismo tiempo que permite interaccionar con usuarios. No se trata de la única alternativa de diseño vectorial aplicada al Web pero, se trata de la más utilizada y más completa.

El beneficio del uso de gráficos vectoriales radica en que éstos permiten generar animaciones de poco peso, es decir, que tardan poco tiempo en ser cargadas por el navegador, y no pierden calidad al ser renderizados a tamaños diferentes.

Para poder comprender mejor las ventajas del uso de animaciones vectoriales es necesario tener presentes los siguientes conceptos:

3.1 TIPOS DE GRÁFICOS MANEJADOS:

Flash maneja varios formatos tanto para los gráficos, clips de película o demás elementos de las animaciones generadas, pero son 2 los de mayor realce. Flash aprovecha las posibilidades que ofrece el trabajar con gráficos vectoriales, fácilmente redimensionables y alterables por medio de funciones, y de un almacenamiento inteligente de las imágenes y sonidos empleados en sus animaciones por medio de bibliotecas, para optimizar el tamaño de los archivos que contienen las animaciones.

3.1.1 Vectoriales

Una imagen vectorial es una imagen digital formada por objetos geométricos independientes (segmentos, polígonos, arcos, etc.), cada uno de ellos definido por distintos atributos matemáticos de forma, de posición, de color, etc., es decir que la imagen es representada a partir de líneas o vectores. La calidad de este tipo de gráficos no depende del zoom o del tipo de resolución con el cual se esté mirando; por mucho que se amplíe, el gráfico no se pixela, debido a que el ordenador traza automáticamente las líneas para ese nivel de acercamiento; esto quiere decir que se les puede aplicar zoom de forma ilimitada sin que sus bordes se vean afectados.

3.1.2 Mapa de bits

Este tipo de gráficos se asemejan a una cuadrícula en la cual cada uno de los cuadrados (píxeles) muestra un color determinado. La información de estos gráficos es guardada individualmente para cada píxel y es definida por las coordenadas y color de dicho píxel. Este tipo de gráficos dependen de la variación del tamaño y resolución, pudiendo perder calidad al modificar sucesivamente sus dimensiones, puesto que llega un momento en que el zoom produce la distorsión de la misma lo cual revela que la imagen está compuesta por píxeles.

3.2 ENTORNO DE ADOBE FLASH

Dentro del entorno de Flash se pueden distinguir cuatro partes principales:

- **La Línea de Tiempo**

  Sección donde se organiza la sucesión de imágenes que dan lugar a una animación. Esta sección queda reservada a la gestión de los fotogramas, las capas y los fotogramas se tratarán más adelante.

- **Gestor de Capas**

  Donde se organizan las capas, que a su vez contendrán a los fotogramas

- **El escenario**

  Espacio en el cual se llevan a cabo todas las tareas de edición de gráficos.

- **Panel de herramientas**

  Espacio donde se encuentran las herramientas de edición gráfica.

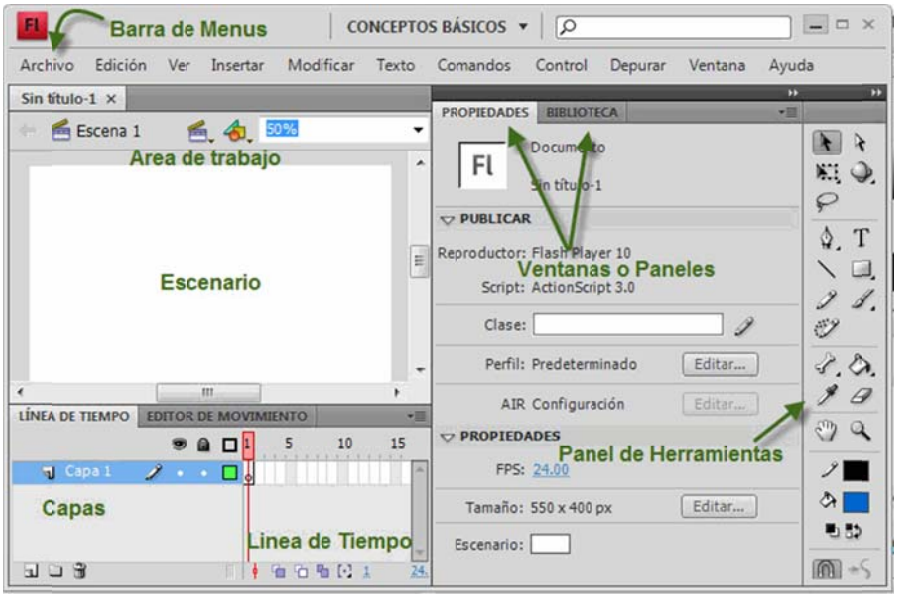
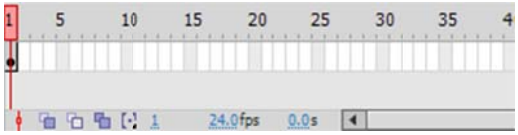


Figura 2.1.1 Entorno de Adobe Flash CS4

3.2.1 La Línea de Tiempo

La Línea de Tiempo es una forma de ver los fotogramas de modo simplificado. Tiene 2 partes:



- Los Fotogramas (frames) están delimitados por líneas verticales (formando rectángulos)
- Los Números de Fotograma permiten saber qué número tiene asignado cada fotograma, cuánto dura o cuándo aparecerá en la película.

Además, en la parte inferior hay herramientas para trabajar con Papel cebolla e información sobre el Número de Fotograma actual (1 en la imagen), la Velocidad de los Fotogramas (24.0 en la imagen) y el Tiempo de película transcurrido (0.0s en la imagen).

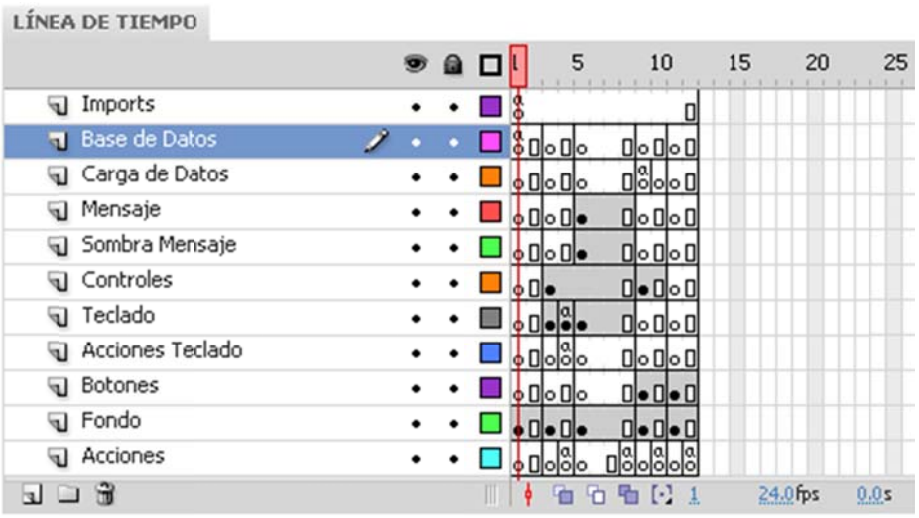
A nivel conceptual, la Línea de Tiempo representa la sucesión de Fotogramas en el tiempo. Es decir, la película Flash no será nada más que los fotogramas que aparecen en la Línea de tiempo uno detrás de otro, en el orden que establece la misma Línea de tiempo.

3.2.2 Gestor de Capas

El concepto de Capa es fundamental para manejar Flash de forma eficiente. Dada la importancia de éstas, se le dedicará un tema completo.

Una Capa se puede definir como una película independiente de un único nivel. Es decir, una capa contiene su propia Línea de Tiempo (con infinitos fotogramas).

Los objetos que estén en una determinada capa comparten fotogramas y por tanto, pueden "mezclarse" entre sí. Esto es interesante a menudo, pero otras veces es conveniente separar los objetos de modo que no interfieran entre sí. Para ello, se debe crear tantas capas como sea necesario. El uso de múltiples capas además, da lugar a películas bien ordenadas y de fácil manejo (es conveniente colocar el código ActionScript en una capa independiente llamada "acciones", por ejemplo).



3.2.3 El Escenario

Es el espacio sobre el cual se dibujan y colocan los diferentes elementos de la película. El escenario tiene propiedades muy importantes, ya que coinciden con las Propiedades del documento. Para acceder a ellas, habrá de hacerse clic con el botón derecho sobre cualquier parte del escenario en la que no haya ningún objeto y después sobre Propiedades del documento, de inmediato se muestra un panel donde se pueden editar las siguientes opciones:

Dimensiones**:** Determinan el tamaño de la película. El tamaño mínimo es de 1 x 1 px (píxeles) y el máximo de 2880 x 2880 px.

Coincidir**:** Provocan que el tamaño de la película coincida con el botón seleccionado (tamaño por defecto de la Impresora, Contenidos existentes o los elegidos como Predeterminados)

Color de Fondo: El color aquí seleccionado será el color de fondo de toda la película

Velocidad de Fotogramas: Número de fotogramas por segundo que aparecerán en la película.

Unidades de Regla: Unidad que se empleará para medir las cantidades.

Transformar en predeterminado: Permite almacenar las propiedades del documento actual y aplicarlas a todos los documentos nuevos que se creen desde ese instante en adelante.

Estas propiedades podrán ser alteradas desde este panel cuando se desee.

3.2.4 Panel de Herramientas

Como en cualquier programa de diseño, es el lugar donde están todas las opcionalidades de dibujo o edición, entre las más importantes están:

Herramienta Selección (flecha): Es la herramienta más usada de todas. Su uso principal es para seleccionar objetos, permite seleccionar los bordes de los objetos (con doble clic) y los rellenos (con un sólo clic)

Herramienta Texto: Crea un texto en el lugar en el que se haga clic.

Herramienta Línea: Permite crear líneas rectas de un modo rápido. Las líneas se crean como en cualquier programa de dibujo, se hace clic y se arrastra hasta donde se desee que llegue la línea recta. Una vez creada se la puede modificar situando el cursor encima de los extremos para estirarlos y en cualquier otra parte cercana a la recta para curvarla

Herramienta de forma: Permite dibujar formas predefinidas como las que aparecen en la imagen, la última herramienta utilizada es la que aparece como icono en la barra de herramientas. Manteniendo pulsado el icono se puede acceder a las diferentes herramientas de forma. Por ejemplo

La herramienta Óvalo permite dibujar círculos o elipses de manera rápida y sencilla.

 Herramienta Lápiz: Es la primera herramienta de dibujo propiamente dicho. Permite dibujar líneas con la forma que se decida, modificando la forma de éstas según se prefiera. El color que aplicará esta herramienta se puede modificar, bien desde el Panel Mezclador de Colores o bien desde el subpanel Colores que hay en la Barra de Herramientas.

 Herramienta Pincel: Su funcionalidad es parecida a la del lápiz, pero por defecto su trazo es más grueso e irregular. Se suele emplear para aplicar rellenos.

 Herramienta Cubo de pintura: Permite aplicar rellenos a los objetos que se hayan creado. Al contrario que muchos otros programas de dibujo, no permite aplicar rellenos si la zona no está delimitada por un borde. El color que aplica esta herramienta se puede modificar, bien desde el Panel Color o bien desde el subpanel Relleno que hay en la Barra de Herramientas.

 Herramienta Borrador: Su funcionamiento es análogo a la Herramienta Pincel. Pero su función es la de eliminar todo aquello que "dibuje".

3.2.5 Herramientas Avanzadas

 Herramienta Lazo: Su función es complementaria a la de la herramienta Flecha, pues puede seleccionar casi cualquier cosa, sin importar la forma, (la Herramienta Flecha sólo puede seleccionar objetos o zonas rectangulares o cuadradas). En contrapartida, la Herramienta Lazo no puede seleccionar rellenos u objetos (a menos que se haga la selección a mano).
Al seleccionar esta herramienta, en el Panel de Herramientas aparecen estos botones:

 Herramienta Varita Mágica: Permite hacer selecciones según los colores de los objetos.

 Para seleccionar Polígono.

 Herramienta Pluma: Crea polígonos (y por tanto rectas, rectángulos...) de un modo sencillo. Su empleo consiste en hacer clic en los lugares que se desee definir como vértices de los polígonos, asegurando una gran precisión, para crear curvas, hay que señalar los puntos que la delimitan y posteriormente trazar las tangentes a ellas.

 Herramienta Subelección: Esta Herramienta complementa a la Herramienta Pluma, ya que permite mover o ajustar los vértices que componen los objetos creados con dicha herramienta.

 Herramienta Bote de Tinta: Se emplea para cambiar rápidamente el color de un trazo. Se aplica sobre objetos, si tienen borde, cambia al color mostrado de dicho borde, por el mostrado en el Panel Mezclador de Colores (que coincide con el subpanel Colores que hay en la Barra de Herramientas**.**)

3.2.6 Herramientas Opcionales

Algunas Herramientas poseen unas opciones especiales que facilitan y potencian su uso. Para acceder a estas utilidades, hay que hacer clic en la línea o en el objeto dibujado y aparecerá un submenú como el siguiente:

 Ajustar a Objetos: Se usa para obligar a los objetos a "encajar" unos con otros, es decir, para que en caso de ser posible, sus bordes se superpongan, dando la sensación de estar "unidos".

 Suavizar: Convierte los trazos rectos en líneas menos rígidas.

 Enderezar: Realiza la labor inversa. Convierte los trazos redondeados en más rectilíneos.

 Texto:Flash fue concebido para crear animaciones gráficas, de modo que tratará cualquier texto como un objeto, listo para ser animado. Esto permitirá posteriormente animar textos y crear animaciones con muy poco esfuerzo. Flash distingue entre 3 tipos de texto, texto estático o normal, texto dinámico y texto de entrada (para que el

usuario introduzca sus datos), también se puede crear texto que soporte formato HTML etc...

Tipos de Texto

Flash distingue entre diversos tipos de texto y les da un tratamiento especial, según el tipo que sean. El tipo de texto se puede modificar desde el Panel Propiedades haciendo clic sobre la pestaña Tipo de texto:

Texto Estático

El Texto Estático no presenta ningún cambio a lo largo de la animación. Es importante que no se confunda la palabra "estático" con que el texto no se mueva. El texto estático puede estar animado (girar, cambiar de color...) y sin embargo ser estático.

Texto Dinámico

El Texto Dinámico en contraposición al estático puede cambiar su contenido (además de estar animado). Su uso es bastante más complejo que el del Texto Estático, ya que cada recuadro de texto Dinámico puede ser una variable modificable mediante ActionScript, esto quiere decir que los valores y propiedades de este tipo de textos se pueden modificar mediante programación.

Texto de Entrada

El Texto de Entrada o Introducción de texto tiene básicamente las mismas propiedades que el Texto Dinámico, junto con algunas propias de un tipo de texto orientado a la introducción de datos por parte de usuario, como por ejemplo el número máximo de líneas que puede introducir en ese campo de texto o si se quiere que lo que el usuario escriba en dicho campo aparezca como asteriscos (para las contraseñas).

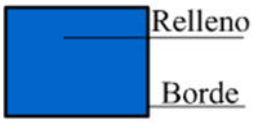
3.2.7 Los Objetos de Iniciación Gráficos

Independientemente de si se está trabajando en una animación, en una página web, en un catálogo para un DVD o en cualquier otra cosa, tendremos que trabajar con objetos. A grandes rasgos, podremos considerar un objeto a todo aquello que aparezca en nuestra película y sea visible, de modo que podamos trabajar con él, por ejemplo, cualquier imagen que creemos o importemos, un botón, un dibujo creado por nosotros mismos etc...

Los objetos así considerados tienen 2 partes fundamentales:

El Borde: Consiste en una delgada línea que separa el objeto del exterior, del escenario. Puede existir o no, según nos convenga. Cuando creamos un objeto, el borde se crea siempre y su color será el indicado en el Color de Trazo (dentro del Panel Mezclador de Colores). Si queremos dibujar creando Bordes deberemos emplear las Herramientas Lápiz, Línea o Pluma y si queremos que nuestro dibujo no tenga borde, bastará con seleccionar el borde y suprimirlo (ver siguiente punto).

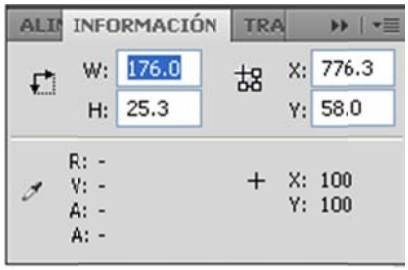
El Relleno: El relleno no es más que el propio objeto sin borde. Es, por tanto, la parte interna del objeto. Su existencia también es arbitraria, ya que podemos crear un objeto cuyo color de relleno sea transparente, como ya se vio en el tema de Dibujar, y por tanto, parecerá que dicho objeto no tiene relleno, aunque en realidad sí exista pero sea de color transparente. Para dibujar Rellenos (sin borde) podemos usar herramientas tales como el Pincel o el Cubo de Pintura.



3.2.8 Panel de Información

Además de controlar la posición de los objetos desde el Panel Alineamiento, también podemos hacerlo, de un modo más exacto (más matemático) desde otro panel, el Panel Información.

A este Panel se puede acceder desde el Menú Ventana → Información. Las posibilidades de este Panel son limitadas, pero es útil si buscamos exactitud en las medidas o no nos fiamos de las distribuciones de objetos que crea Flash.



Medidas del Objeto: Aquí se introduce un número que representa el **tamaño** del objeto en la medida seleccionada en las Propiedades del documento. **An**: hace referencia al ancho y **Al**: a la altura.

Situación del objeto: Desde aquí se controla la posición del objeto en el escenario. La X y la Y representan el eje de coordenadas (La X es el eje Horizontal y la Y el eje vertical). Las medidas también van en función de las medidas elegidas para la película.

Color Actual: Indica el color actual en función de la cantidad de Rojo (R), Verde (V), Azul (A) y efecto Alfa (Alfa) que contenga.

Este indicador puede ser engañoso, el motivo es que indica el color que tiene el objeto por el que en ese momento pasamos el cursor del ratón. Por tanto, se puede seleccionar un objeto (haciendo clic en él) y ver en el Panel Información su tamaño y su posición, pero al desplazar el ratón, el valor del color cambiará y ya no indicará el color del objeto seleccionado, sino el del objeto por el que pase el cursor.

Posición del Cursor: Indica la posición del cursor. Es útil por si queremos que suceda algo en la película al pasar el cursor justo por una posición determinada o para situar partes del objeto en lugares específicos.

3.2.9 Los Grupos

Un Grupo no es más que un conjunto de objetos. Para crear un grupo, basta seleccionar los objetos que queremos que formen parte de un grupo y después hacer clic en el Menú Modificar → Agrupar (Ctrl + G).

Tras hacer esto se observa que desaparecen las texturas que indicaban que los objetos estaban seleccionados y se puede apreciar que el grupo pasa a ser un "todo", ya que resulta imposible seleccionar a uno de sus miembros sin que se seleccionen a su vez los demás. Además, aparece el rectángulo azul (por defecto) que rodea al grupo, definiéndolo como tal.

Crear grupos es muy útil, ya que permiten, tratar al conjunto de objetos como un todo y por tanto, se puede aplicar efectos al conjunto, ahorrando al usuario, la labor de hacerlo de objeto en objeto.

Al crear un grupo, se da propiedades comunes a un conjunto de objetos y, sin perderlos. El grupo puede deshacerse en cualquier momento, mediante el Menú Modificar → Desagrupar.

Además, Flash permite modificar los elementos de un grupo sin tener que desagruparlo. Para ello, se debe seleccionar el Grupo de elementos y hacer clic en el Menú Edición → EditarSeleccionado. Se puede editar los objetos que componen el grupo por separado teniendo en cuenta que, como es lógico, los cambios realizados afectarán al grupo además de al elemento en cuestión

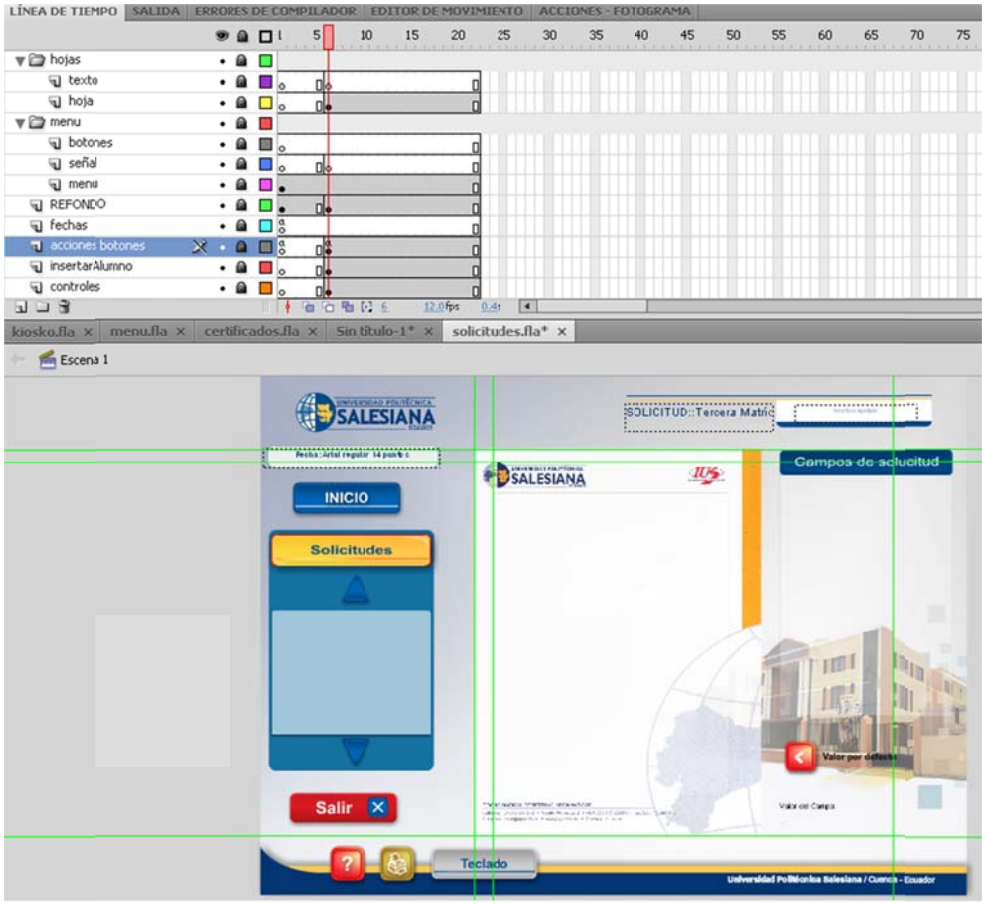
3.3 LAS CAPAS

Cada capa es, un nivel en el que se puede dibujar, insertar sonidos, textos... con independencia del resto de capas. Hay que tener en cuenta que todas las capas comparten la misma Línea de Tiempo y por tanto, sus distintos fotogramas se reproducirán simultáneamente.

Se puede aclarar esto con un ejemplo:
Se supone que se tiene 2 capas. En una de ellas los fotogramas del 1 al 10 contienen el dibujo de una portería de fútbol. En la otra los fotogramas del 1 al 5 contienen el dibujo de un portero (del 5 en adelante están vacíos).

Esta película mostrará inicialmente (durante el tiempo que duren los primeros 5 fotogramas) la portería con el portero, para después (durante los fotogramas del 5 al 10) mostrar la portería sin portero.

De este modo la portería es independiente del portero, y se puede tratar estos objetos con total libertad, ya que no interfieren entre ellos para nada.



Otra razón para separar los objetos en capas, es que Flash obliga al usuario a colocar cada animación distinta en una capa. De lo contrario, todos los objetos que se encuentren en dicha capa formarán parte de la animación. Si se quiere que un objeto no forme parte de una animación, se debe quitarlo de la capa en la que se produce dicha animación.

Siguiendo con el ejemplo del portero, si se quiere crear un movimiento que haga que el portero se desplace hacia un lado no hay ningún inconveniente, pero si la portería estuviera en la misma capa que el portero, entonces ambos objetos se moverían hacia dicho lado, con lo que resultaría imposible que sólo se moviera el portero. La solución es separar los objetos en 2 capas.

Las capas además, tienen otras utilidades, permiten ordenar la película de forma lógica, y ayudan en la edición de dibujos (evitando que se "fundan" en uno sólo, o

bloqueando el resto de capas de modo que sólo se pueda seleccionar la capa que nos interese).

Otro motivo es para organizar mejor el contenido. Igual que se creaba una capa para los elementos de audio, se crean capas para otros elementos, como el código ActionScript.

3.3.1 Trabajar con Capas

La vista estándar de una capa es la que muestra la imagen. A continuación, se describe para qué sirven los distintos botones y cómo usarlos.

- Nueva capa: Como su nombre indica, sirve para Insertar una nueva capa en la escena actual. Crea capas normales (en el siguiente punto se verán los distintos tipos de capas).
- Crear carpeta: Sirve para crear carpetas, que ayudarán a organizar las capas.
- Borrar Capa: Borra la capa seleccionada.

Cambiar Nombre: Para cambiar el nombre a una capa, basta con hacer doble clic en el nombre actual.

Propiedades de Capa: Si se hace doble clic en el icono junto al nombre de la capa, se puede acceder a un panel con las propiedades de la capa en la que se haya hecho clic. Se pueden modificar todas las opciones comentadas anteriormente y alguna más de menor importancia.

Aquí se pueden cambiar diferentes opciones sobre la capa, como su nombre o su color. También existe la posibilidad de bloquearla u ocultarla.

3.3.2 Opciones Avanzadas

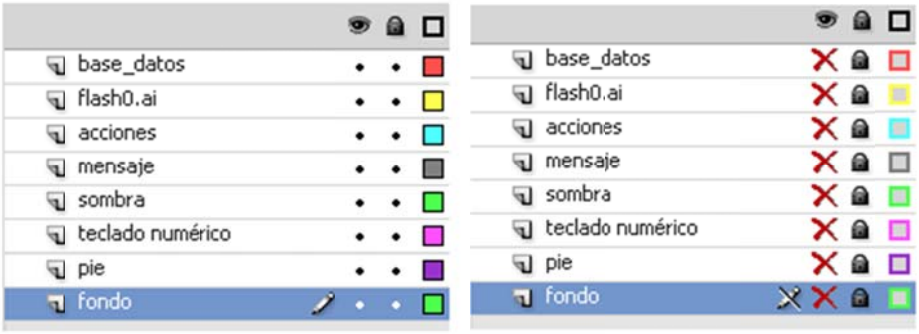
- Mostrar / Ocultar Capas: Este botón permite ver u ocultar todas las capas de la película. Es muy útil cuando existen muchas capas y se desea ver solamente una de ellas ya que permite ocultar todas a la vez, para después mostrar sólo la actual. Para activar la vista de una capa en concreto (o para ocultarla) basta con hacer clic en la capa correspondiente en el punto (o en la cruz) que se encuentra bajo el icono "Mostrar / Ocultar capas".
- Bloquear Capas: Bloquea la edición de todas las capas, de modo que no pueden ser modificadas hasta que estén desbloqueardas. Para bloquear o desbloquear una capa concreta, se procede como en el punto anterior, clic en el punto o icono "Cerrojo" situados en la capa actual bajo el icono "Bloquear Capas".

  Bloquear una capa es muy útil cuando existen varios objetos juntos y en capas distintas y se desea tener la seguridad de no modificar "sin querer" alguno de ellos. Tras bloquear su capa se puede trabajar con la seguridad de no modificar dicho objeto, ni siquiera puede ser seleccionarlo, de modo que se puede editar con mayor facilidad el objeto que se desee.

- **Mostrar/Ocultar capas como contornos**: Este botón muestra/oculta los contenidos de todas las capas como si sólo estuviesen formados por bordes. De este modo y ante un conjunto numeroso de objetos, se puede distinguir a todos de forma fácil y se puede ver en qué capa está cada uno de ellos.

También se puede activar o desactivar para cada capa de un modo similar a los anteriores botones.

A continuación se muestran estas opciones activadas y desactivadas.



En la primera imagen la capa actual no tiene ninguno de los botones activados, se puede observar que en la columna Mostrar Capas aparece un punto. Este punto significa que no está activada esta opción, lo mismo sucede con el botón Bloquear capas. En la columna Mostrar capas como contornos aparece un cuadrado con relleno, lo que simboliza que los objetos se mostrarán completos y no sólo sus contornos.

En la segunda imagen aparece una cruz situada bajo la columna Mostrar Capas, lo que indica que dicha capa no es visible en el escenario. Aparece un cerrojo bajo la columna "bloquear capas", lo que simboliza que la capa está bloqueada. Y en la columna "Mostrar capas como contornos" NO aparece relleno. La capa se está mostrando en este modo y no se podrá ver los rellenos hasta deseleccionar esta opción.

Además, el color de los contornos será diferente para cada capa, para poder distinguirlas mejor. El color del contorno, coincidirá con el indicado en cada capa.

3.3.3 Reorganizar Capas

Como ya se ha comentado, las distintas capas tienen muchas cosas en común unas con otras. Lo primero y principal es la Línea de tiempo, todas las capas de una misma escena comparten la misma línea de tiempo y por tanto, los objetos de todos los fotogramas 1 de todas las capas se verán al mismo tiempo en la película superpuestos unos sobre otros. ¿Y qué objeto está delante de los demás? Pues este criterio viene dado por la colocación de las Capas en la película. Los objetos que se mostrarán delante de todos los demás serán aquellos que se encuentren en la capa situada más arriba.

3.3.4 Tipos de Capas

Existen varios tipos de capas:

- Capas normales: Son las capas por defecto de Flash y tienen todas las propiedades descritas en los puntos anteriores. Son las más usadas y se emplean para todo, por ejemplo, para colocar objetos, sonidos, acciones, ayudas...

- Capas animación movimiento: Son las capas que contienen una animación por interpolación de movimiento.

- / Capas Guía: Son capas especiales de contenido específico. Se emplean en las animaciones de movimiento de objetos y su único fin es marcar la trayectoria que debe seguir dicho objeto. Debido a que su misión es representar la trayectoria de un objeto animado, su contenido suele ser una línea (recta, curva o con cualquier forma).

Es importante recordar que el contenido de las Capas Guía no se verá en la película final. Su efecto hará que el objeto se desplace de un extremo de la línea al otro siguiendo esa ruta.

3.4 LOS SÍMBOLOS

Los Símbolos provienen de objetos que se han creado. Estos objetos al ser transformados en símbolos, son incluidos en una biblioteca en el momento en que son creados, lo que permite que sean utilizados en varias ocasiones, ya sea en la misma o en otra película. Los símbolos resultan fundamentales al momento de crear nuestras animaciones.

3.4.1 Como crear un Símbolo

La acción de crear un nuevo símbolo es una de las más usadas en Flash ya que es uno de los primeros pasos para crear una animación, como se verá más adelante.

El procedimiento es el siguiente:

1. Seleccionamos el o los objetos que se desee convertir en símbolo. Lo más habitual es partir de una forma.

2. Se abre la ventana Convertir en símbolo, accediendo al menú Insertar → Nuevo Símbolo, desde el menú contextual eligiendo Convertir en símbolo, o directamente con las teclas Ctrl + F8 o F8.

3. Una vez hecho esto aparecerá una ventana como la mostrada en la imagen. Se introduce el nombre del símbolo a crear, y que permitirá identificarlo en la biblioteca, lo que se hará imprescindible cuando se tengan muchos símbolos.

4. Se procede a seleccionar el tipo de símbolo (desplegable Tipo) al que se quiera convertir en objeto. Se puede elegir entre Clip de Película, Botón y Gráfico. Sus características y las diferencias entre ellos se verán en temas posteriores. Lo más habitual es Clips de película para los objetos que se quieren mostrar en el escenario, y Botón si se sesea que actúe como tal.

5. Bastará con pulsar Aceptar para tener el símbolo creado.

3.4.2 Las Bibliotecas

En Flash CS4 se pueden encontrar dos tipos de bibliotecas, las bibliotecas comunes y de ejemplos y aquellas asociadas a las películas creadas. Todas ellas están a la disposición del usuario para utilizar los símbolos que contienen.

Para acceder a las bibliotecas comunes que ofrece Flash simplemente se debe ir al menú Ventana → Bibliotecas Comunes y seleccionar alguna de las que se están disponibles. Las hay de todo tipo de símbolos: botones, clips o gráficos.

Para acceder a la biblioteca de símbolos de la película que se está creando, de nuevo se recurre a la Barra de Menús, Ventana → Biblioteca. En esta biblioteca aparecerán todos los símbolos que creados hasta el momento.

Se puede comprobar el nuevo símbolo creado, accediendo a la biblioteca como se acaba de indicar.

Los símbolos contenidos en las bibliotecas están identificados por su nombre y por un icono que representa el tipo de símbolo que representan:

 Clip  Botón  Gráfico

Para utilizar un símbolo de una biblioteca basta con pulsar en el nombre de dicho símbolo y arrastrarlo a cualquier lugar del área de trabajo.

3.4.3 Diferencia entre Símbolo e Instancia

Cuando se crea un símbolo, Flash lo almacena en una biblioteca. Pues bien, cada vez que se utiliza ese objeto en una película, éste se convierte en una instancia del símbolo.

Por tanto, se puede crear muchas instancias de un símbolo, pero a una instancia solo le corresponderá un símbolo.

Aunque parece que sean lo mismo, la importancia de esta distinción es que cuando se utiliza un símbolo creado previamente en una película y éste sea modificado, se alterará únicamente ésa instancia, mientras que el objeto seguirá intacto, tal y como era en el momento de su creación, de manera que se pueda utilizar en otro momento. En cambio, si es modificado el símbolo de la biblioteca, se modificarán todas sus instancias.

3.4.4 Modificar una instancia

Se ha visto que es posible modificar una instancia de un símbolo sin modificar el símbolo original en cuestión. Sin embargo, al no tratarse de un gráfico vectorial, no se pueden modificar las instancias con las herramientas de dibujo de Flash CS4, pero sí mediante el Panel de Propiedades, que permite la manipulación "externa" de la instancia.

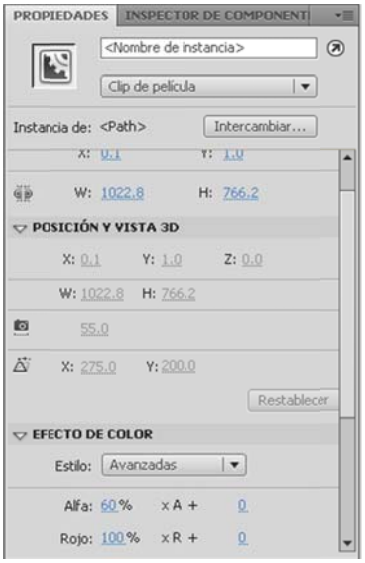
Así, este panel, que como se ha visto resulta sumamente útil, no permite modificar la estructura básica de la instancia, pero sí otras propiedades, esto es, se puede hacer que la instancia tenga más brillo, pero no transformar una estrella en un círculo). Esos cambios deben hacerse directamente sobre el símbolo. Aunque sí se puede crear un símbolo a partir de una instancia, lo que desvinculará la instancia del símbolo original.

3.4.5 Panel de Propiedades de una Instancia

Para acceder al panel de propiedades de instancia, se debe seleccionar en primer lugar la instancia que se quiere modificar y posteriormente abrir el panel Propiedades.

Si se selecciona un objeto Flash que no se trate de un símbolo, el Panel Propiedades mostrará las propiedades del objeto en cuestión, pero no las características propias de los símbolos (cambios de color, intercambios etc...)

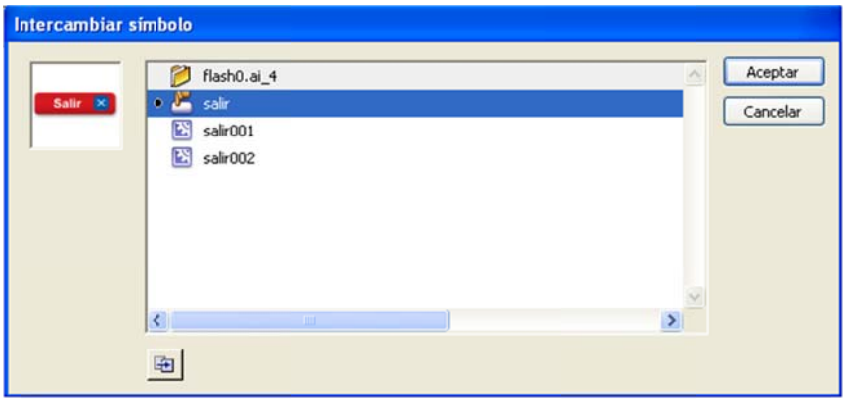
En el momento que se seleccione un símbolo aparecerán una serie de propiedades y opciones que se comentan a continuación:



- Nombre y su icono correspondiente: El nombre de la instancia permite identificarla durante la película, y se verá más adelante que resulta imprescindible para acceder mediante ActionScript. El nombre, se introduce en el recuadro donde pone por defecto <Nombre de instancia>, y debe de ser único.
- Tipo de la instancia. Por defecto se muestra el tipo al que pertenecía el símbolo original pero éste puede ser cambiado para que cambie su comportamiento, aunque pueda seguir manteniendo su estructura inicial (en la imagen es Botón).
- Símbolo de la Instancia seleccionada (Instancia de:). Esta opción muestra el símbolo raíz del que proviene la instancia que está siendo modificada.
- Intercambiar: Su función consiste en cambiar el símbolo de la instancia por cualquier otro que exista en la Biblioteca, por lo que la instancia tomará el aspecto del nuevo símbolo. Puede parecer simple, pero durante el desarrollo de un trabajo profesional rápidamente surge la necesidad de probar situaciones y los diseños gráficos definitivos no suelen estar disponibles hasta bien avanzado el proyecto. Gracias a esta opción es posible trabajar tranquilamente con un "boceto" y sustituirlo de un modo efectivo (el nuevo símbolo hereda las propiedades del antiguo símbolo, incluido el nombre de

instancia, las acciones que le afectarán, efectos gráficos etc...) cuando llegue el momento.

En la imagen se puede observar el panel Intercambiar Símbolo.



Este panel además, incorpora el botón Duplicar Símbolo  . Es muy útil para hacer pruebas con un símbolo sin que éste se pierda. Se lo duplica y se trabaja tranquilamente con la copia.

3.4.6 Efectos sobre Instancias

Para acceder a los efectos aplicables sobre una instancia determinada, se acude nuevamente al Panel Propiedades y desde aquí se accede a todos los efectos que Flash proporciona.

Hay varios tipos de efectos. Si el símbolo se acaba de crear o si no tiene efecto asignado aparecerá en la pestaña Ninguno.

A continuación se muestran los tipos de efectos. Para ello se parte de la siguiente imagen original:

Brillo. Se puede modificar su valor desde -100% al 100%, esto es, completamente oscuro (negro) y completamente brillante (blanco). Se puede mover la barra deslizante o introducir su valor directamente en la casilla.

Efecto Brillo del 50 %

Tinta. Esta opción permite cambiar el color de la instancia, pero como se explicó, no es posible modificar la instancia internamente, al variar el color en la pestaña Tinta o bien mediante los valores RGB (cantidad de rojo, verde y azul), se cambiará el color de toda la instancia como si la estuviéramos tiñendo o poniendo una capa imaginaria de un color determinado. El grosor o intensidad de esta "capa" la podemos modificar en porcentaje mediante la primera pestaña que aparece a la derecha.

Efecto Tinta del 50 % con el color verde (0 255 0)

Alfa. Representa el grado de visibilidad o transparencia que se tendrá de la instancia en cuestión. También se puede modificar mediante valor directo o con la barra deslizante y es muy útil para animaciones de aparición y desaparición de objetos. Si aplicamos un efecto alfa sobre una instancia que está encima de otro objeto, el objeto que antes estaba tapado se podrá ver a través de la instancia.

Efecto Alfa del 65 % boton

Avanzado. Aquí se puede aplicar todos los efectos anteriores al mismo tiempo de manera más precisa, con la ventaja de que se puede poner un poco de cada uno, dando lugar a efectos de gran vistosidad.

A medida que se modifiquen los efectos sobre las instancias, se puede identificar el resultado sobre el propio escenario.

3.5 GRÁFICOS

Los Gráficos son símbolos que permiten representar objetos estáticos y animaciones sencillas.

En caso de que se utilice un símbolo gráfico para realizar una animación, se debe tener en cuenta que ésta estará ligada a la línea de tiempo de la película en la que se encuentre. Es decir, la animación se reproducirá siempre y cuando la película original también se esté reproduciendo. Esto hace que, pese a tener su propia línea de tiempo, no puedan contener sonidos, controles ni otros símbolos gráficos.

Así pues, normalmente se utilizarán los gráficos para imágenes estáticas o para cuando sea conveniente que una animación se reproduzca sólo cuando determinado frame de la línea de tiempo de la película esté en marcha, ya que para los casos comentados anteriormente en los que un gráfico no es útil, Flash ofrece otro tipo de símbolos como se verá en temas posteriores.

3.5.1 Tipos de Gráficos

Los gráficos pueden ser:

a) Estáticos: estos gráficos se mantienen sin cambios cuando pasa el tiempo. Éstos son los típicos en los fondos y en los objetos que no desempeñan ninguna función especial. Su tamaño y por tanto, el tiempo de carga de este tipo de gráficos, aunque siempre dependerá de la resolución, de sus dimensiones y de la forma en la que estén creados *, será en general reducido.

Gráfico estático

b) **Animaciones**: este tipo de gráfico varía su forma, posición u otras propiedades a medida que va pasando el tiempo. Puesto que para realizar la animación se deben usar varios gráficos más además del original o bien realizar determinadas acciones que modifiquen el estado inicial, el tamaño de esta clase de gráficos, para las mismas dimensiones y forma de creación, será mucho mayor que uno estático.

Por esto, aunque las animaciones dan a la web un aspecto más bonito y espectacular tienen dos inconvenientes:

1. Si se trata de un Mapa de Bits la web puede llegar a tener un tamaño excesivamente grande.

2. Aunque no se traten de mapas de bits, por ejemplo, si son animaciones típicas de Flash, cuyo tamaño no es excesivo, el hecho de poner muchas animaciones puede llegar a "marear" un poco al visitante del sitio y desviar su atención de lo que realmente importa, su contenido.

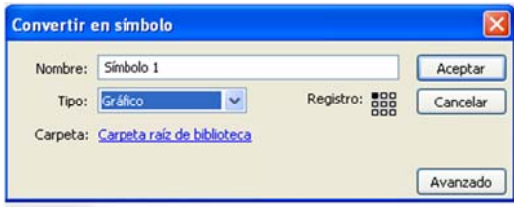
(*) Los tipos de gráfico anteriores pueden ser, a su vez de dos tipos, según la forma en la que estén creados: Gráfico Vectorial o Mapa de Bits.

3.5.2 Creando Gráficos y Comprobando sus Propiedades

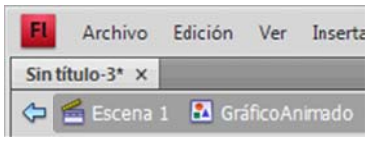
Para crear un gráfico en Flash se, abre una nueva película Flash (Archivo→ Nuevo, Archivo de Flash (AS 3.0)).

Ahora se crea el objeto que se quiere convertir en un símbolo Gráfico. Se dibuja, por ejemplo, un óvalo en cualquier lugar del área de trabajo con la herramienta Óvalo de la barra de herramientas de dibujo y se le da un color de relleno que será lo que después se va a animar.

Ya se ha creado el objeto, ahora se debe convertirlo en un símbolo gráfico. Como se describió anteriormente, dándole el nombre GráficoAnimado y seleccionando el Tipo Gráfico:



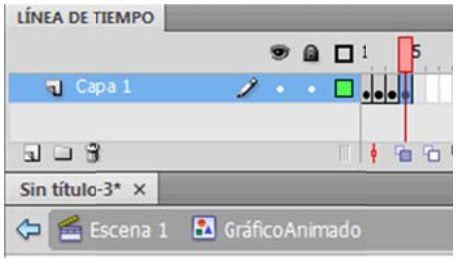
Ahora se va a crear la animación del gráfico. Para ello se selecciona el gráfico, se pulsa el botón derecho del ratón. Se desplegará un menú, en el que se escoge la opción Edición para modificar el gráfico y acceder a su línea de tiempos. En la imagen inferior, se puede apreciar que se encuentra en "Escena1 - Gráfico Animado" y, por tanto se encuentra dentro del gráfico (y la línea de tiempos que se aprecia es la del gráfico, y no la de la película principal)



Se crean a continuación nuevos fotogramas clave seleccionando uno a uno los frames número 2, 3 y 4 y pulsando F6 al momento de seleccionarlos.

Se debe pulsar sobre el frame 2 y se cambia el color de fondo al óvalo. Se procede de igual manera en los dos siguientes frames.

La línea de tiempo debería tener este aspecto:



Se debe pulsar donde pone Escena 1 justo encima del escenario y de este modo se regresa al nivel inicial (Película principal) y se puede ver el gráfico "desde fuera".

Se ha creado el gráfico animado.

Para probar la animación se debe pulsar Control + Intro.

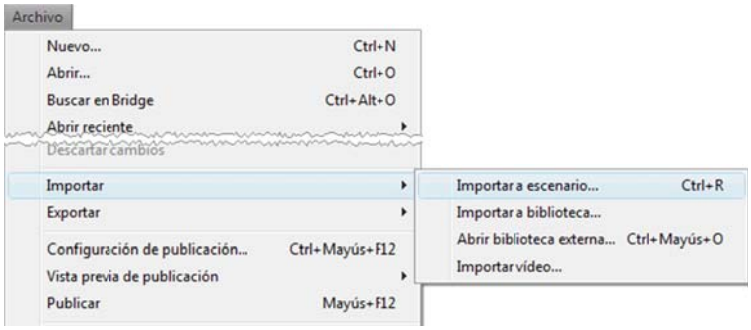
3.5.3 Introducir un mapa de bits

Los **gráficos de tipo Mapa de Bits** pueden crearse con muchos programas. Si se desea que en las películas hayan mapas de bits lo más seguro es que se prefiera

crearlos y hacerlos más espectaculares en otro programa más apropiado que Flash para el manejo de Bitmaps: Photoshop, Fireworks, GIMP, etc...

Flash CS4 permite importar mapas de bits de otros programas, cuando han sido guardados en formatos gráfico GIF, JPG, TIFF y muchos más. También permite modificarlos en cierto modo. Pudiendo cambiarle el tamaño y convertirlo en un símbolo para aprovechar las opciones que de Flash aunque, teniendo en cuenta qué es un bitmap, no puede ser modificarlo "internamente" pero puede usarse como un símbolo más.

Para importar un archivo de Mapa de Bits al escenario se debe hacer clic en el menú Archivo → Importar → Importar a escenario.



Se abrirá el cuadro de diálogo de Importar, allí se debe seleccionar el formato de imagen que se desee importar seleccionándolo en el desplegable Tipo. Luego se debe navegar por las carpetas hasta encontrarlo. Luego se pulsa el botón Abrir.

La imagen se incluirá en el escenario y estará lista para trabajar con ella.

3.5.4 Introducir un archivo vectorial

Al igual que los mapas de bits, hay otros programas que trabajan con gráficos vectoriales como también hace Flash CS4.

Si se desea traer un archivo vectorial creado en otro programa, por ejemplo Freehand o Illustrator, simplemente se accede al menú Archivo → Importar →

Importar a escenario. A continuación, se selecciona el tipo de archivocorrespondiente al gráfico vectorial que se desee importar. Por ejemplo AI de Illustrator.

Se pulsa Abrir, y ya se cuenta con el archivo vectorial.
Este archivo sí puede ser modificado internamente ya que Flash es capaz de hacer gráficos de este tipo.
Concretamente, Illustrator pertenece también a Adobe, igual que Flash, con lo que la compatibilidad en este caso es total.

3.5.5 Exportar un archivo flash como mapa de bits

La interfaz de dibujo de Flash, como se ha visto, resulta muy cómoda en determinadas ocasiones para realizar dibujos. Así, se podría utilizar Flash CS4 para crear un dibujo y después utilizar éste en otros programas o para cualquier otro uso.

Esto es posible con Flash, ya que permite exportar un objeto de flash como un bitmap. Eso sí, se debe saber, que la mayoría de mapas de bits no permiten animaciones, por esto el objeto flash que se exporte no debería contener animación ya que ésta no se guardará.
Para realizarlo, se selecciona el objeto a exportar y se accede al menú Archivo → Exportar → Exportar Imagen... Luego se introduce en el campo Nombre el nombre para el nuevo bitmap.
Se selecciona el tipo de mapa de bits en que se desee convertir el objeto y luego se pulsa Guardar.
Ahora ya se puede usar el objeto Flash como un bitmap.

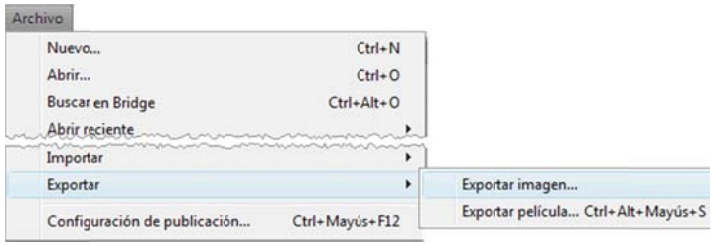
3.5.6 Exportar un objeto flash como animación

Hay tipos de archivo de mapa de bits que soportan animaciones, como los GIF. Con Flash también existe la posibilidad de crear una animación y exportarla como un GIF animado. Sin embargo, como se vió anteriormente, el GIF animado consiste en una secuencia de imágenes mostradas secuencialmente y es por esto que para exportar un

objeto Flash CS4 como GIF animado es necesario que todos los fotogramas de esta animación sean clave, ya que el GIF no lo reconocerá en caso contrario y no se verá el efecto deseado.

Para exportar un símbolo y guardarlo como una imagen primero debe ser seleccionarlo con la herramienta Selección.

Una vez seleccionado se debe hacer clic en el menú Archivo → Exportar → Exportar imagen... y se abrirá un cuadro de diálogo.



En este cuadro de diálogo se debe introducir el nombre del archivo que ha sido creado y seleccionar en el desplegable Tipo el formato de imagen con el que se desea guardarlo.

Una vez rellenados todos los campos y elegida la carpeta donde se guardará el archivo se debe pulsar el botón Guardar y el archivo de imagen se creará y estará listo si se desea incluirlo en una página web estática o modificarlo con cualquier programa de imagen.

3.6 CLIPS DE PELÍCULA

Un Clip de Película, es una película en sí misma. Se referiere a ellas como clips cuando son incluidas en otra película, formando un símbolo. Por tanto, cualquier clip siempre podrá estar compuesto por otros clips insertados en él, que a su vez estén formados por otros, etc.

Al igual que los otros tipos de símbolos de Flash, los clips de película tienen su propia línea de tiempo. Sin embargo, esta línea temporal no está ligada a la línea de tiempo del documento que lo contiene, de tal forma que su ejecución es

independiente, y en un fotograma de la película principal se puede estar reproduciendo repetidamente un clip.

Este tipo de símbolos puede contener cualquier otro tipo de símbolo: gráfico, clip o botón, así como cualquier objeto creado con Flash, ya que un clip es realmente una película.

Otra de las ventajas de los Clips se identifica cuando se realizan películas de gran complejidad y tamaño, en la que intervienen un número muy elevado de fotogramas, debido a que en la vista general del documento, sólo se ve un fotograma por clip, el cual puede estar compuesto por muchos frames, lo que permitirá tener una mejor visión de cómo se desarrolla la animación, y una línea de tiempo más clara y "limpia"

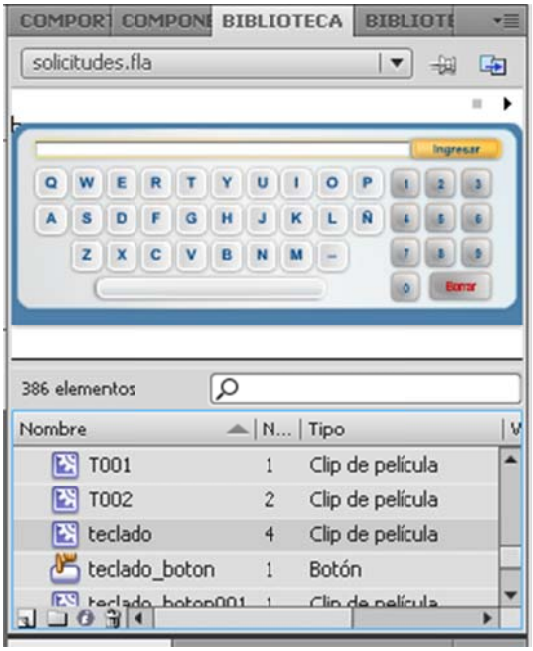
Los Clips son una de las herramientas que dan mayor potencia a Flash CS4, permitiéndo crear películas de gran complejidad y multiplicar los efectos visuales, ya que se pueden crear múltiples movimientos independientes entre sí y crear conexiones entre los diferentes Clips de un documento.

Todas aquellas cosas que no se puede hacer con un símbolo de tipo Gráfico, es posible hacerlo con un Clip, además de poder realizar también todo aquello que permitía dicho símbolo. Por esto, normalmente se utilizan los clips para cualquier tipo de animación debido a su gran flexibilidad, dejando los gráficos sólo para imágenes estáticas.

3.6.1 Comprobar las propiedades de un Clip

Se abre una nueva película Flash (Archivo → Nuevo, Archivo de Flash (AS 3.0)). Después se importa una imagen cualquiera o bien se crea una. Se convierte ésta en Símbolo (botón derecho, Convertir en símbolo...) y se selecciona en Tipo "Clip de Película"

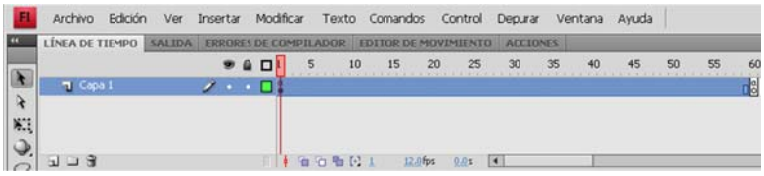
Se procede a arrastrar al escenario el clip de película.



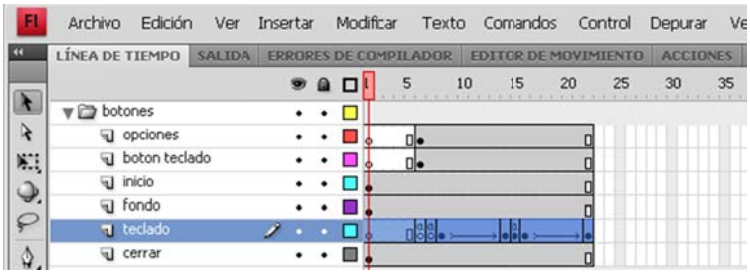
Ahora el Clip está insertado en el nuevo documento Flash. Es decir, existen dos películas, cada una con su línea de tiempo.

Ahora se ve que son independientes, siguiendo un proceso similar al que se utilizó con los símbolos de tipo Gráfico.

Se observa la línea de tiempo de la película principal y se comprueba que sólo tiene un fotograma.



Se edita el Clip que se insertó en el documento, haciendo doble clic sobre él y se examina su línea de tiempo. Luego aparecerá un único MovieClip. Se puede realizar algo similar a lo realizado con los gráficos. Si se crea una animación de movimiento, como se verá más adelante, quedaría así:



Como se ve, la duración del clip insertado es mucho mayor que la película nueva que lo contiene. Si el símbolo fuera un gráfico ya se ha visto que al reproducir la película no ocurriría nada, porque sólo se reproduciría el primer fotograma de su línea de tiempo. En cambio, al tratarse de un clip, comienza a reproducirse al pasar por el primer fotograma, y como la línea de tiempo es independiente, sigue reproduciéndose aunque la línea de tiempo principal haya acabado.

3.6.2 Como crear un Clip de Película

Se usa normalmente Clips para hacer animaciones. Se puede crear un símbolo Flash de la nada, igual que se crea un nuevo archivo, de forma que se quede en la biblioteca y se pueda editar cuando sea conveniente. Esto puede ser interesante en los clips, ya que a diferencia de los gráficos, su finalidad suele ser el movimiento y, en animaciones complejas, en ocasiones se les asignan acciones especiales en las cuales puede que no sea necesario crearlo en ese momento o convenga dejar el clip vacío.

Por esto, es interesante aprender cómo crear un símbolo, en este caso un clip, de la nada para después modificarlo.

Para insertar un clip vacío se debe hacer clic en Insertar → Nuevo símbolo y se abrirá el cuadro de diálogo de Crear un nuevo símbolo.

Se debe dar un Nombre para identificarlo más tarde en la Biblioteca y seleccionar la opción Clip de Película en el desplegable Tipo.

A partir de este momento se tendrá un nuevo clip (vacío) al cual podremos acceder desde la Biblioteca (menú Ventana → Biblioteca), si se hace clic derecho sobre él y se selecciona Edición, se puede editar y trabajar con él.

3.6.3 Importar y Exportar Movie Clips de la Biblioteca

Como para todos los símbolos los Clips se almacenan en la biblioteca del documento cuando son creados. Esto es muy importante en muchos casos ya que habitualmente los clips son muy reutilizables. Para importar clips de película se debe abrir primero la biblioteca en la que está contenido.
Se ha visto en el tema de Símbolos, dos tipos de bibliotecas: las que están asociadas a documentos u otras películas y las que proporciona Flash CS4. Pues bien, no sólo se pueden utilizar símbolos del mismo documento en el que se encuentra el usuario sino que existe la posibilidad de Importarlos de otros documentos del disco duro, lo que, puede resultar de gran utilidad. Obviamente la exportación mediante biblioteca se hace automáticamente ya que Flash deja los objetos creados en la biblioteca para que puedan ser reutilizados.

Para importar un Clip de un archivo del disco duro se debe ir al menú Archivo → Importar → Abrir biblioteca externa..., seleccionar el Archivo Flash (.fla) del que se desee importar sus símbolos de biblioteca y pulsar Abrir.

Aparecerá la biblioteca con la lista de los símbolos correspondientes a los gráficos, botones y clips del documento en cuestión.
Es importante destacar que cuando se inserta un clip de una biblioteca, se insertarán a su vez todos los símbolos que contenga, incluidos los clips.

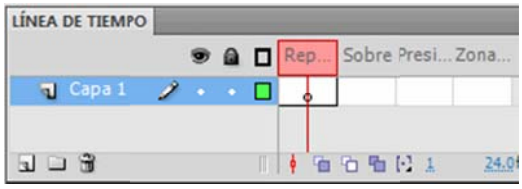
3.7 BOTONES

Los símbolos de tipo Botón son los que aportan la mayor parte de la interactividad de las películas Flash con aquel que la está visualizando. Un botón, en Flash, es igual que cualquier botón de cualquier entorno informático, sea web o cualquier otro.

Son elementos que se prestan a que el usuario los presione, desencadenando al hacerlo una serie de acciones. También es habitual ver cómo este tipo de elementos reaccionan cuando se les pasa el ratón por encima o cuando están pulsados.

Para conseguir los efectos interactivos que se han mencionado, en otros lenguajes orientados a la web, hay que crear programas relativamente grandes. Esto es un inconveniente ya que el uso de los botones es una práctica muy habitual en el diseño en Internet. Sin embargo, en Flash no ocurre así. Su interfaz está diseñada de manera especial para la creación de botones, lo que permite crear todos estos efectos de una manera muy sencilla.

Al igual que los otros símbolos de Flash CS4, los botones tienen su propia línea de tiempo. Esta es independiente pero, sin embargo, está formada únicamente por cuatro fotogramas, uno para cada estado posible del botón.



- Reposo. Aspecto por defecto del botón, es decir, cuando el puntero del ratón no está situado sobre él.
- Sobre. Aspecto del botón cuando se sitúa el puntero sobre él.
- Presionado. Apariencia que se desea que tenga el botón mientras esté pulsado.

- Zona activa. Aquí se debe indicar el área real en la que se desea que actúe el botón. Esto es importante sobre todo en botones compuestos sólo por texto como se verá más adelante.

Parece que la limitación de fotogramas podría implicar una limitación en la capacidad de espectacularidad y utilidad de estos símbolos, pero no es así.

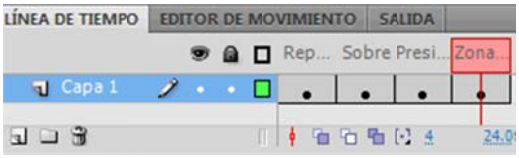
Los botones pueden contener a su vez otros símbolos, como clips o gráficos (también de tipo Bitmap). La unión de las posibilidades de todos los símbolos pueden dotar a los botones de efectos asombrosos.

3.7.1 Creación de un Botón

En la creación de un botón se consideran dos fases. En la primera se convierte el objeto a símbolo de tipo botón y posteriormente se verá cómo completarlo internamente, lo que ayudará a entender mejor dicha estructura.
Se crea el objeto que representará el aspecto por defecto del botón con las herramientas que ofrece Flash CS4.
Se selecciona el objeto y se accede al menú Insertar → Convertir en Símbolo, se da el TipoBotón y se asigna un nombre al nuevo símbolo.

De esta forma se ha transformado el objeto para que se comporte como un botón. Ahora hay que completarlo internamente.
Para determinar cómo debe reaccionar el botón en función de las acciones del ratón, es necesario editarlo, haciendo clic con el botón derecho del ratón sobre el nuevo botón y seleccionando la opción Editar.

Cuando se tiene delante la línea de tiempo del botón, se debe seleccionar cada uno de los frames (sobre, reposo, presionado y zona activa) y pulsar F6 para crear un fotograma clave en cada uno de ellos.



Ahora se puede modificar el aspecto inicial del botón para cada posición del cursor y marcar el área de acción del botón (fotograma Hit) en la que simplemente se puede dejar la misma figura que la inicial o bien dibujar con las herramientas de dibujos de Flash una nueva figura, en cuya superficie "se sentirá aludido" el botón

Si una vez creado el botón se desea observar sus distintos estados y todavía no se ha terminado la película entera y por tanto no se desea tener que reproducirla toda se puede hacerlo accediendo a la Biblioteca de la película y seleccionando el botón creado. Para ver lo que comentábamos bastará con pulsar el icono situado a la derecha de la vista previa del símbolo.

3.7.2 Formas en los Botones

Los botones son símbolos que pueden tener multitud de formas. Si bien lo más habitual es ver botones rectangulares, cuadrados y circulares, cuya creación es inmediata, también hay otros muchos tipos de botones que, pese a ser menos utilizados, es muy habitual verlos en multitud de páginas web.

Entre estos están los creados mediante formas poligonales, aquellos que están formados por texto únicamente, dibujos con diferentes motivos, etc. Es interesante su uso para dar más vistosidad ya que algunos resultan más expresivos, y en esto Flash ayuda mucho, debido a la relativa sencillez de creación de botones que sus herramientas de dibujo ofrecen.

Hay varias formas de botón también muy extendidas, como el botón con relieve sencillo o los botones en forma de píldora. Puesto que existen muchas formas de conseguir estos efectos.

3.7.3 Incluir un clip en un Botón

La inclusión de clips de película en los botones puede dotar a éstos de más vistosidad.

Es habitual colocar un clip en el fotograma Sobre para indicar algún tipo de información extra o una animación para ir más allá de un cambio de color.
También es común ver un clip de película actuando como un botón. Esto caso se puede hacer por ejemplo poniendo el clip en el fotograma Reposo.

3.7.4 Incluir Bitmaps en Botones

Además de clips, los botones también pueden contener símbolos de tipo Gráfico.
Si se consideran las limitaciones sobre los mapas de bits puede parecer poco interesante hacer uso de ellos en la creación de botones, pero no es así.
Se pueden ejecutar las siguientes acciones:

1. Incluir en cada uno de los fotogramas del botón un bitmap distinto, obteniendo un efecto como el que se consigue con lenguajes como javascript.
2. Aprovechar las propiedades de los Gráficos en Flash. Para esto, se debería importar primero el Bitmap y después convertirlo a símbolo botón. Posteriormente será editado y, después de insertar cada fotograma clave, se procede a convertir su contenido a símbolo Gráfico. Luego se pueden aplicar efectos de las instancias en Flash (Alfa, Tinta, Brillo).

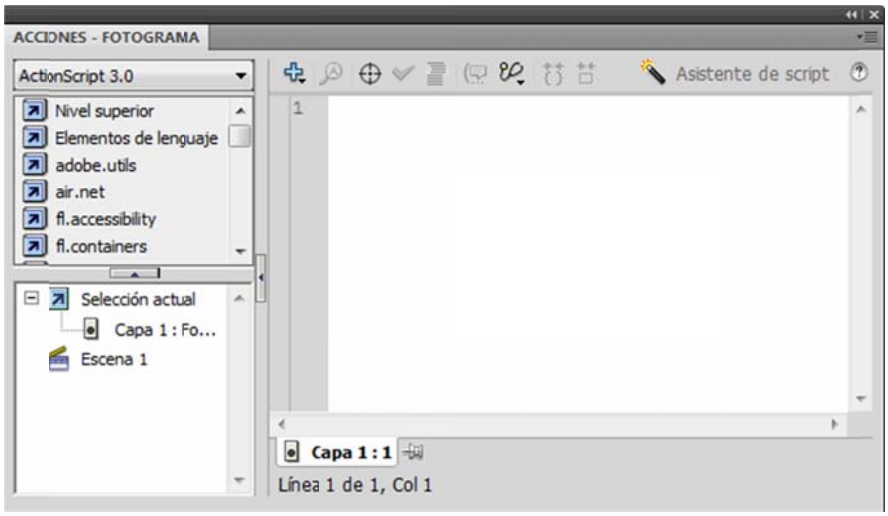
3.7.5 Acciones en Botones

Existen varias acciones que se pueden aplicar tanto a los botones, como a otros elementos de Flash CS4 pero se va a comentar las más comunes.

Se crea o añade el botón, se le da un nombre de instancia. El nombre que le demos es muy importante, porque permitirá acceder a él desde el código.

El código puede crearse en la misma capa, pero es recomendable crear una capa exclusivamente para el código para una mejor organización.

Se abre el panelAcciones (menú Ventana → Acciones). Se mostrará un área en blanco en la que se puede escribir:



Se escribe el siguiente código para asociar acciones al botón:

```
miBoton.addEventListener('click', accionesMiBoton);
functionaccionesMiBoton(event):void{
        //Acciones
}
```

Dónde:

miBoton hace referencia al nombre de la instancia del botón.

accionesMiBoton contiene las acciones a realizar.

Si se cuenta con varios botones, a cada uno se lo referencia por su nombre de instancia que es único. También es necesario dar un nombre único a

accionesMiBoton para cada uno, caso contrario, todos realizarían las mismas acciones.

Luego sustituimos //Acciones por lo que se quiere que haga. Las más comunes:

1. **Abrir una página web.** Con lo que se abre una página cualquiera de internet (o una película Flash),

   Para ello ActionScript cuenta con la instrucciónnavigateToURL(new URLRequest("http://www.mipagina.es"), "_blank").

   "http://www.mipagina.es" se refiere a la página que se desea abrir, y "_blank" indica que se abrirá en una página nueva.

2. **Controlar una película en curso.** Si se está reproduciendo una película Flash y se desea que el usuario la detenga, la ponga en marcha, avance, retroceda ...

   Para ello se puede emplear las acciones:

   - stop(); para detener.
   - play(); para reproducir.
   - gotoAndPlay(numeroFotograma); para ir a un fotograma determinado.

3.8 ANIMACIONES DE MOVIMIENTO

Flash es un programa básicamente orientado a la animación. Para ir creando animaciones cada vez más complicadas se necesita, mucha práctica, aparte de conocer bien las herramientas.

En el tema de la animación, Flash ofrece unas facilidades muy grandes, consiguiendo efectos que normalmente requieren ciertos conocimientos y espacio de almacenamiento para ser creados, como por ejemplo los GIF animados o lenguajes de programación como JavaScript, de una manera muy sencilla, sin necesidad de excesivos conocimientos y ocupando muy poco espacio en disco.

En Flash CS4 ha habido un cambio importante en las animaciones, lo que hasta ahora se llamaba interpolación de movimiento, pasa a llamarse interpolación clásica y la interpolación de movimiento actual es totalmente nueva, más potente y versátil.

3.8.1 La interpolación de movimiento

Es la acción básica de las animaciones en Flash. Permite desplazar un símbolo Flash de un lugar a otro del escenario, siendo necesarios únicamente dos fotogramas, lo que optimiza mucho el rendimiento de la película.
Es importante destacar que para que una Interpolación de movimiento se ejecute correctamente aquellos objetos que intervengan deberán haber sido previamenteconvertidos a símbolos (gráficos, clips de película, textos y botones son algunos de los símbolos que se pueden interpolar).
También se debe tener cuidado al realizar una interpolación con dos símbolos que se encuentren en la misma capa, ya que el motor de animación los agrupará como uno sólo y el resultado no será el esperado. Por esto es conveniente separar en distintas capas:

- Los objetos fijos y los que estarán animados.
- Objetos que vayan a ser animados con direcciones o formas distintas.

Una interpolación de movimiento, es el desplazamiento de un símbolo de uno a otro punto del escenario. El hecho de que sólo se necesiten dos fotogramas es debido a que Flash, únicamente con la posición inicial y final, "intuye" una trayectoria en línea recta y la representa (también se pueden realizar movimientos no rectilíneos).
Para crearla se debe hacer clic derecho sobre el fotograma que contiene los elementos y elegir Crear interpolación de movimiento. Por defecto, se añadirán unos cuantos fotogramas, rellenos de un color azulado.
Luego se debe ir al fotograma final, o crear uno clave donde se desee. Y se procede a desplazar el símbolo. A continuación aparece una línea punteada, por defecto recta, que representa el trazado de la animación.

Esto indica que la animación cambiará la posición del símbolo del fotograma 1 hasta la posición del mismo símbolo en el fotograma 24, utilizando precisamente 24 fotogramas. El número de fotogramas que se usen en la interpolación indicará las subetapas de que constará la animación. Cuantas más subetapas más sensación de "continuidad" (menos saltos bruscos) pero a la vez menos velocidad en el movimiento.

La velocidad en el movimiento de las películas se puede cambiar modificando su parámetro en la línea de tiempo, 

pero esto no cambiará lo que se comentó anteriormente respecto al número de fotogramas.

La velocidad está expresada en Fotogramas Por Segundo (fps) y se puede modificar haciendo doble clic en el lugar indicado de la línea de tiempo. A mayor valor más velocidad, pero se deben poner siempre suficientes fotogramas para que se desarrolle la animación como sea requerida.

El trazado recto generado por defecto puede ser modificarlo directamente haciendo clic y arrastrándolo, una vez seleccionada previamente la herramienta Selección.

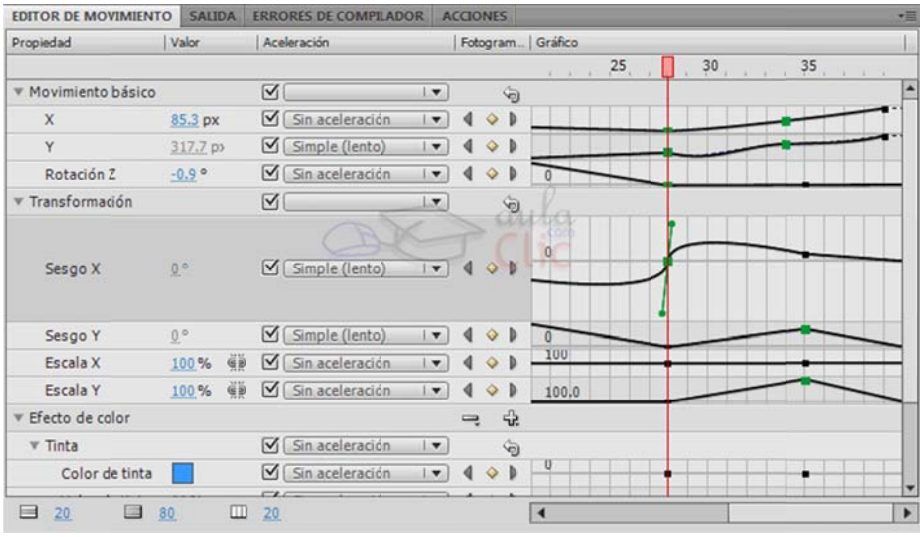
En cualquier fotograma de la interpolación se puede cambiar la posición del símbolo (o cualquier otra propiedad), creando ahí un fotograma clave de propiedad, que se representa por un pequeño rombo en la línea de tiempo.

La interpolación de movimiento permite modificar muchos parámetros del movimiento mediante el Editor de movimiento, que veremos a continuación

3.8.2 Editor de Movimiento

En la versión de Flash, CS4, se encuentra un nuevo panel llamado Editor de movimiento. Para acceder a él basta tener seleccionada una interpolación de movimiento y hacer clic en la pestaña Editor de movimiento que aparece junto a la línea de tiempo. Como cualquier panel, también es accesible desde el menú Ventanas.

Este panel permite controlar multitud de propiedades y efectos que afectan a una animación con total precisión, fotograma a fotograma.



A la izquierda está una columna con las propiedades que son modificables, divididas en Movimiento básico, Transformación, Efectos de Color, Filtros y Suavizados.

Junto a estas propiedades, aparece una columna con los valores que toma esa propiedad en el momento seleccionado de la línea de tiempo.

En la siguiente columna podemos establecer si el valor se aplica con aceleración o no.

En la columna Fotogramas, se puede recorrer o eliminar los distintos fotogramas clave. También los controles - y + que permiten añadir efectos.

Y a la derecha del todo se encuentra la gráfica. Se puede ver que cada propiedad tiene una gráfica específica, que indica los fotogramas en horizontal y los valores de la propiedad en vertical. Si se hace clic sobre una propiedad, se ve que su gráfica se expande para editarla con facilidad. En la gráfica se identifican los fotogramas clave marcados como un cuadrado negro, o verde cuando está seleccionado. Estirando de ellos, o de la línea de la gráfica es posible alterar los valores.

En la gráfica, se aprecia que los puntos suelen formar un vértice. Una opción es poder transformarlos en puntos suavizados (desde el menú contextual del fotograma), creando una curva Bezier, lo que formará transiciones más suaves entre los picos de valor. Esto no es aplicable a las propiedades X,Y, Z.

3.8.3 Interpolación Clásica

Una interpolación clásica, igual que una interpolación de movimiento, es el desplazamiento de un símbolo de uno a otro punto del escenario, muchos de los conceptos vistos en las interpolaciones de movimiento son los mismos para las interpolaciones clásicas. Por ejemplo, las animaciones también se realizan sobre símbolos y deben estar en una capa. Los Fotogramas Por Segundo (fps) tienen el mismo significado.

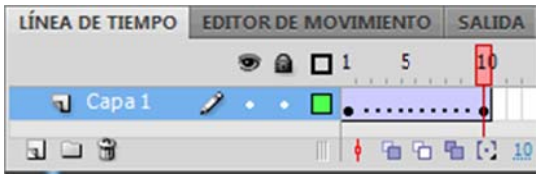
Para crear una interpolación clásica hay que hacer clic derecho sobre el fotograma que contiene los elementos y elegir Crear interpolación clásica.

Cuando se realiza la interpolación correctamente se observa un aspecto como este en la línea de tiempo.

La animación va desde el fotograma 1 hasta el fotograma 20. Aparece una flecha que no aparece en la interpolación de movimiento y el icono que hay a la derecha del nombre de la capa es distinto.

Al realizar una interpolación clásica el fotograma inicial y final deberán ser diferentes, en caso contrario no se creará ningún tipo de animación.

Si el objeto con el que se desea hacer la interpolación clásica no está convertido, la línea del tiempo se mostrará de la siguiente manera.



y la animación no funcionará.

También es posible realizar la interpolación de otra forma, sin convertir previamente el objeto a símbolo, ya que Flash lo convierte a símbolo automáticamente si no es realizado por el usuario, dándole el nombre "Animar" más un número. Esto es conveniente en películas grandes, debido a la gran cantidad de símbolos que pueden aparecer y la confusión que crean muchos símbolos con nombres parecidos.

Para crear una interpolación de este tipo, basta con tener un fotograma clave. Se hace clic con el botón derecho sobre el fotograma en la línea de tiempo, y se selecciona Crear Interpolación Clásica. Ahora, se crea un nuevo fotograma clave donde se desea que finalice la interpolación, y se modifica los símbolos en los fotogramas clave.

Se puede ver que si se selecciona uno intermedio, se muestran los símbolos en su transición al fotograma final. Se puede decidir cómo mostrar el símbolo en ese fotograma, por ejemplo moviéndolo. Al hacerlo automáticamente se crea un fotograma clave. Esto hace que el movimiento ya no sea recto, y pueda ser en zig-zag.

Si se realiza esto varias veces sobre varios fotogramas se obtendrán varias trayectorias consecutivas más.

3.8.4 Diferencias entre Interpolación Clásica y de movimiento

Las interpolaciones de movimiento son más fáciles de utilizar y más potentes, no obstante las interpolaciones clásicas tienen características que pueden hacerlas más interesantes para determinados usuarios.

Estas son algunas de las diferencias entre los dos tipos de interpolaciones:

- Las interpolaciones de movimiento incluyen el trazado del movimiento, mientras que en una animación clásica no existe el trazado, a menos que lo creemos expresamente.
- Sólo se permiten realizar interpolaciones con símbolos, si aplicamos una interpolación de movimiento a un objeto que no es un símbolo, Flash lo convertirá en un clip de película, mientras que si se trata de una interpolación clásica lo convertirá en un símbolo gráfico.

- En las interpolaciones clásicas cuando cambia una propiedad se crea un fotograma clave y cambia la instancia del objeto, mientras que en las interpolaciones de movimiento sólo hay una instancia de objeto y al cambiar una propiedad se crea un fotograma clave de propiedad.
- Las interpolaciones de movimiento pueden trabajar con texto sin tener que convertirlo en símbolo, como ocurre en las clásicas.
- En un grupo de interpolación de movimiento no está permitido usar scripts de fotograma, mientras que sí es posible en las clásicas.
- Los grupos de interpolaciones de movimiento se pueden cambiar de tamaño en la línea de tiempo. Se tratan como un objeto único. Las interpolaciones clásicas están formadas por grupos de fotogramas que se pueden seleccionar de forma independiente.
- Las interpolaciones de movimiento sólo pueden aplicar un efecto de color por interpolación, mientras que las clásicas pueden aplicar más de uno.
- Los objetos 3D sólo pueden animarse en interpolaciones de movimiento, no en clásicas.
- Sólo las interpolaciones de movimiento se pueden guardar como configuraciones predefinidas de movimiento.

3.8.5 Animación de Texto

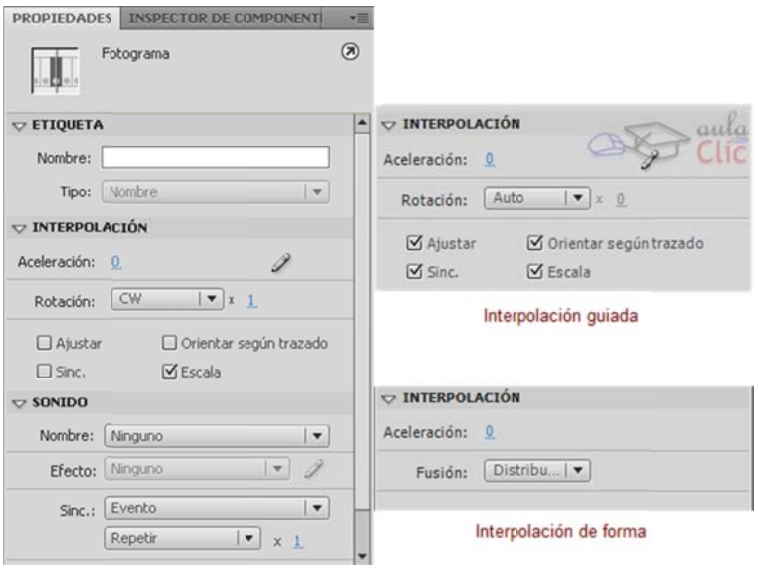
Para comunicar algún mensaje, no basta con imágenes o iconos, y es aquí donde el texto cobra gran importancia. Pero, se debe tener cuidado con la animación de los textos, puesto que resulta bastante complicado leer un texto que se desplaza o cambia de tamaño.

Por ello, un texto animado debería estar sólo en las presentaciones o bien formar parte de una animación corta y, no debería estar reproduciéndose infinitamente.

Una de las opciones más utilizadas es separar las letras de los textos y animarlas independientemente

3.8.6 Animación de Líneas

Una buena animación no tiene que porqué estar compuesta sólo por textos o imágenes espectaculares. En ocasiones conviene darle a la película un aire más sencillo o añadir determinados efectos que la hagan vistosa sin necesidad de cargar mucho la película visualmente, y en cuanto a tamaño de archivo se refiere.
Esto se consigue animando líneas y haciendo que se muevan por el escenario. Esta técnica permite dar dinamismo a la animación o crear formas distintas a lo largo de su recorrido.



3.8.7 Efecto Brillo

El efecto de brillo, como su propio nombre indica, sirve para simular el hecho de que algún objeto se encienda o brille, o por el contrario se apague o pierda brillo.

3.8.8 Efecto Tinta

El efecto de tinta, tiene un amplio marco de posibilidades de uso. Es muy usado en textos y en botones, por ejemplo haciendo que cambien progresivamente de color al pasarles el ratón por encima o simplemente efectos de cambio de color en presentaciones.

El tintar colores supone un toque alegre y muy vistoso en las presentaciones y si se usan varios efectos, combinados adecuadamente, dota de un ritmo rápido a la animación, como una explosión de color que sorprenda al receptor de la película.

3.8.9 Efecto Alfa

Este es probablemente el efecto más utilizado debido a la versatilidad del hecho de controlar el grado de visibilidad de los objetos. Se puede, por ejemplo, simular un foco apuntando a un escenario o, lo más común, hacer aparecer objetos de la nada y también hacer que se desvanezcan poco a poco.

3.9 GENERACIÓN Y PUBLICACIÓN DE PELÍCULAS

Antes de publicar la película, sobre todo si se la va a publicar en una página web, donde el tamaño de descarga es de vital importancia, se debe considerar los siguientes aspectos:

CONSIDERACIONES EN EL DIBUJO:

- Aunque los degradados queden muy vistosos, también requieren más memoria, por lo que se debe evitar su uso excesivo.
- La herramienta Pincel gasta más memoria que el resto de herramientas de dibujo.
- Se ha visto que la animación de líneas es bastante útil. Sin embargo el uso de líneas que no sean las definidas por defecto, hará que el tamaño de la descarga aumente.
- Dibujar las curvas con el menor número de nodos posible.

CONSIDERACIONES EN LA ORGANIZACIÓN:

- Agrupar los objetos que estén relacionados, con el comando Modificar → Agrupar.

- Si se ha creado un objeto que va a aparecer varias veces, se debería convertirlo a símbolo, ya que como se ha descrito anteriormente, Flash lo colocará en la biblioteca y cada vez que quiera mostrarlo, hará referencia a una única posición de memoria.
- Minimizar el uso de los mapas de bits en nuestra película.

CONSIDERACIONES EN LOS TEXTOS:

- Cuando se abre el menú de tipos de letras, las tres primeras son siempre "_sans", "_serif" y "_typewriter". Están colocadas ahí para resaltar que estas fuentes ocupan un mínimo de memoria, por lo que se recomienda su uso.

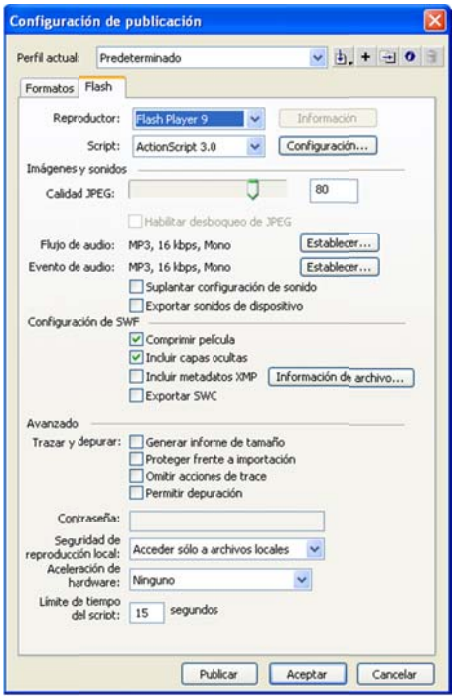
CONSIDERACIONES EN LA ANIMACIÓN:

- Utilizar lo más que se pueda las interpolaciones de movimiento y las guías para reducir el número de fotogramas clave y el tamaño de la película.
- Evitar el uso de la interpolación por forma para animaciones de cambio de color, cuando sea posible.
- Independientemente de la optimización que hagamos, a veces no se puede evitar que el tamaño de la película aumente. Es recomendable entonces hacer un preloader (precarga) cuando la película que se desee publicar sea de tamaño superior a unos 80KB.

En cuanto a la ventana de publicación se detalla lo siguiente:

- Para poder distribuir películas creadas en Flash que la gente pueda ver, son necesarias dos cosas: crear un archivo SWF y que el que la quiera visualizar tenga instalado el Reproductor de Flash.
- Flash ofrece varias opciones y funcionalidades para la creación de un archivo SWF. Estas opciones se pueden ver en el panel de Configuración de Publicación, al que se puede acceder mediante el menú Archivo → Configuración de Publicación (Pestaña Flash).

- Estas opciones son:



- Reproductor: Si se desea publicar la película para que sea vista con versiones anteriores de Flash, se debe seleccionar aquí la versión deseada.
- Versión de ActionScript: El uso de ActionScript 3 permitirá usar las novedades relativas a objetos, clases etc... Si se ha introducido código ActionScript se debe respetar la versión elegida al crear el archivo, si no se pueden producir errores.
- Calidad JPEG: Si en el panel de propiedades del mapa de bits no se ha indicado una compresión concreta, aquí se puede determinar su grado de compresión, que determinará a su vez el espacio ocupado en memoria por este tipo de imágenes. A mayor compresión, menos espacio en memoria ocupará la imagen, pero también su calidad será menor.
- Establecer Flujo de Audio o Evento de Audio: Esta opción permite acceder al Panel "Configuración de Sonido" desde donde se puede configurar, para cada tipo de sonidos, sus características.
- Suplantar configuración de sonido: Con esto se suplantarán los niveles de compresión seleccionados para cada archivo de sonido de nuestro documento.
- Comprimir película: Comprime la película al máximo posible.

- Generar Informe de tamaño: Esta opción se ha usado en el apartado anterior. Si se la activa, se creará un archivo de texto con una relación detallada del tamaño del documento.
- Proteger Frente a Importación: Activando está casilla hará que cuando otro usuario (o nosotros mismos) quiera importarla no pueda o tenga que introducir una contraseña si se ha escogido alguna.
- Omitir acciones de trace: Las acciones de traza se emplean para comprobar el correcto funcionamiento de la película durante la creación de esta (durante las pruebas). También se consideran trazas los comentarios que insertemos en el código ActionScript. Si se activa esta señal, la película creada no los incluirá, ocupará menos tamaño y se ahorrará tiempo innecesario. Es recomendable cuando se publique la película de un modo definitivo.
- Permitir depuración: Permite que se pueda depurar el archivo SWF. También exige la introducción de una contraseña ya que se debe tener permiso del creador para Importar el archivo y depurarlo.

CAPITULO IV

ACTIONSCRIPT 3

ActionScript es un lenguaje de programación orientado a objetos (OOP), utilizado en especial en aplicaciones web animadas, es el lenguaje de programación que utiliza Flash. Permite realizar con Flash CS4 todo lo que se desee, puesto que tiene el control absoluto de todo lo que rodea a una película Flash.

Características generales

- ActionScript está basado en la especificación ECMA-262, al igual que otros lenguajes como Javascript.
- Es un lenguaje de script, es decir que no hará falta crear un programa completo para conseguir resultados, normalmente la aplicación de fragmentos de código ActionScript a los objetos existentes en las películas nos permiten alcanzar nuestros objetivos.
- ActionScript 3 es un lenguaje de programación orientado a objetos. Tiene similitudes, por tanto, con lenguajes tales como los usados en el Microsoft Visual Basic, en el Borland Delphi etc... y aunque, evidentemente, no tiene la potencia de estos lenguajes, cada versión se acerca más. Así, la versión 3.0 utilizada en Flash CS4 es mucho más potente y mucho más "orientada a objetos" que su anterior versión 2.0.
- La sintaxis ActionScript presenta muchos parecidos con el Javascript o PHP.
- En la mayor parte de las ocasiones, será necesario "programar". Flash CS4 pone a disposición una biblioteca de funciones, clases y métodos ya implementadas que realizan lo que se quiere, bastará con colocarlas en el lugar adecuado.

4.1 EL PANEL DE ACCIONES

En Flash CS4, el Panel Acciones sirve para programar scripts con ActionScript, por tanto lo que aquí se introduzca le afectará de menor o mayor medida. Hay que tener claro desde un principio que el Panel Acciones puede hacer referencia a Fotogramas u objetos, de modo que el código ActionScript introducido afectará tan sólo a aquello a lo que referencia el Panel. Por ejemplo, en la imagen inferior, se puede distinguir que el Panel Acciones hace referencia al Fotograma 1 de la Capa 1 

(en el nombre de la pestaña de la zona de la derecha y en la zona izquierda en el apartado Selección actual).

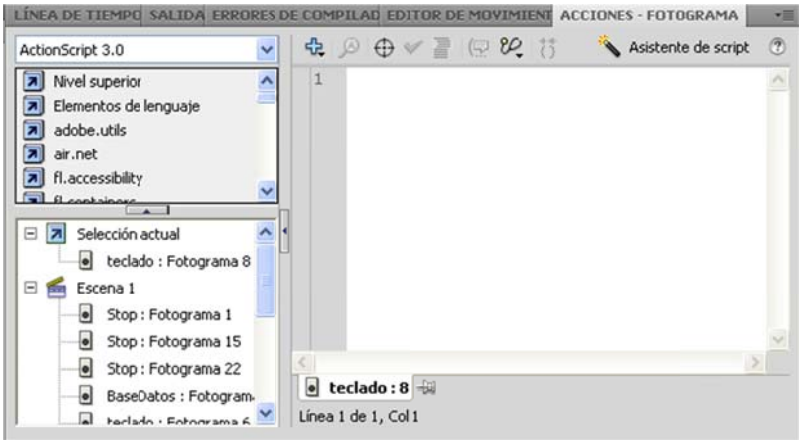


Figura 3.1.1: Editor de Código ActionScript en Flash

El Panel Acciones se divide en 2 partes, a la izquierda hay una ayuda facilitada por Flash que da acceso de un modo rápido y muy cómodo a todas las acciones, objetos, propiedades etc., que Flash tiene predefinidos. Estos elementos están divididos en carpetas, que contienen a su vez más carpetas clasificando de un modo eficaz todo lo que Flash pone a disposición. Para insertarlos en nuestro script bastará con un doble clic sobre el elemento elegido.

En la parte derecha está el espacio para colocar nuestro script, el código de ActionScript. El código se lo puede insertar en cualquier fotograma clave, aunque lo más "limpio" es crear una capa para el código.

El Panel Acciones de Flash CS4, no tiene únicamente un modo de edición. Se puede utilizar el botón 

, en el que en vez de escribir directamente, se pueden seleccionar los distintos elementos desde listas.

En la parte superior están herramientas que ayudarán con el tratamiento del código:

- 
Buscar: Busca un texto en el código. Útil, por ejemplo, si quiere buscar en todos los sitios que usan un objeto.
- 
Revisar sintaxis. Comprobará errores en la sintaxis, normalmente que se hayan olvidado cerrar paréntesis o corchetes. Si encuentra alguno, mostrará un mensaje como el siguiente:

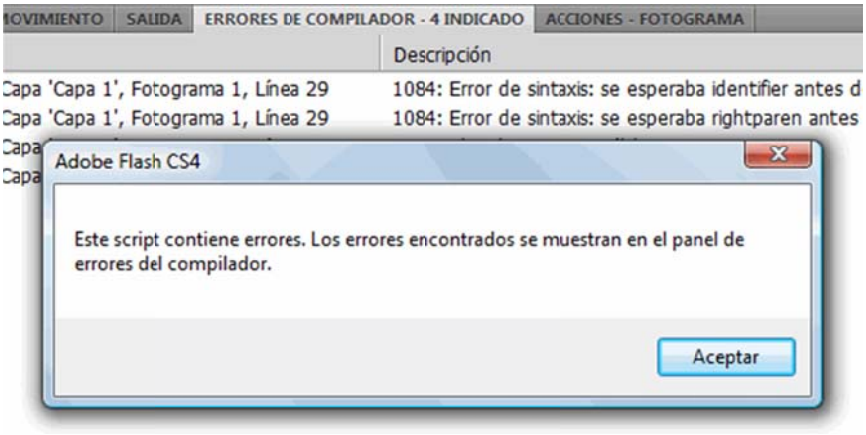


Figura 4.1.2: Visor de Errores de Sintaxis

En el panel ERRORES DE COMPILADOR se mostrarán los errores indicando la capa, fotograma y línea. Habrá la posibilidad ir al lugar del error haciendo doble clic.

- 
Formato automático. Al escribir en cualquier lenguaje, es muy importante hacerlo ordenadamente y con el formato adecuado. Este botón lo hace automáticamente, siempre que no haya errores de sintaxis.
- 
Cuando hay mucho código, resultará más cómodo si se contrae ciertas partes. Con estos botones, se puede de izquierda a derecha contraer el espacio entre llaves, contraer la selección o expandir todo.
- 
Cuando se está programando, es frecuente comentar un fragmento de código para que no se ejecute o que se quiera añadir comentarios descriptivos. Con estos botones, se puede comentar el texto seleccionado, o descomentarlo.

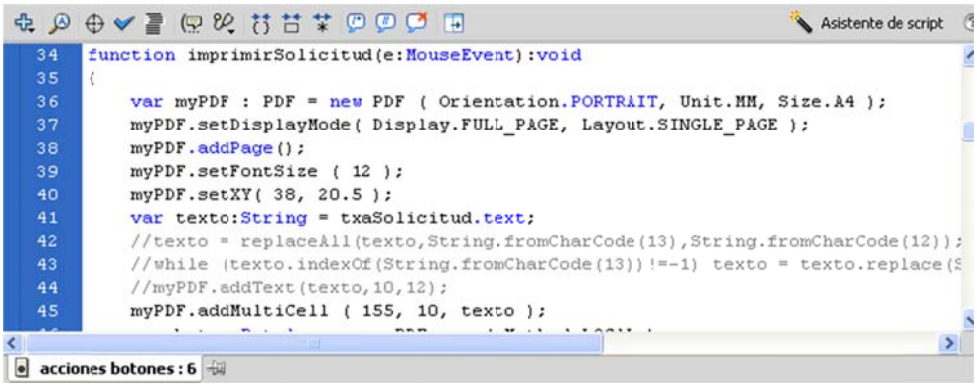


Figura 5.1.3: Editor de Código

Aunque la sintaxis sea correcta, puede que al probar la película se sigan produciendo errores (errores de compilación). Por ejemplo porque se intenta a una propiedad de un objeto que no existe, o se cometió un error al escribir el nombre de una variable.

Estos errores también aparecerán en el panel Errores de compilador. En este caso hay que fijarse en el número de línea del error, ya que al hacer doble clic, a veces no va al lugar correcto si el código tiene comentarios.

4.2 DESCRIPCIÓN DE LOS TIPOS DE DATOS

Los tipos de datos simples son Boolean, int, Null, Number, String, uint y void. Las clases principales de ActionScript también definen los tipos de datos complejos siguientes: Object, Array, Date, Error, Function, RegExp, XML y XMLList.

4.2.1 Tipo de datos Boolean

El tipo de datos Boolean consta de dos valores: true y false. Ningún otro valor es válido para variables de tipo Boolean. El valor predeterminado de una variable booleana declarada pero no inicializada es false.

4.2.2 Tipo de datos int

El tipo de datos int se almacena internamente como un entero de 32 bits y consta del conjunto de enteros entre -2.147.483.648 (-2^{31}) a 2.147.483,647 ($2^{31} - 1$),

ambos incluidos. Las versiones anteriores de ActionScript sólo ofrecían el tipo de datos Number, que se usaba tanto para enteros como para números de coma flotante. En ActionScript 3.0 se tiene acceso a tipos de bajo nivel para enteros de 32 bits con o sin signo.

4.2.3 Tipo de datos Null

El tipo de datos Null (nulo) tiene un único valor: null. Éste es el valor predeterminado para el tipo de datos String y para todas las clases que definen tipos de datos complejos, incluida la clase Object. Ninguno de los demás tipos de datos simples, como Boolean, Number, int y uint, contienen el valor null. Flash Player y Adobe AIR convertirán el valor nullen el valor predeterminado adecuado si intenta asignar nulla variables de tipo Boolean, Number, int o uint. Este tipo de datos no se puede utilizar como una anotación de tipo.

4.2.4 Tipo de datos Number

En ActionScript 3.0, el tipo de datos Number representa enteros, enteros sin signo y números de coma flotante. Sin embargo, para maximizar el rendimiento se recomienda utilizar el tipo de datos Number únicamente para valores enteros que ocupen más que los 32 bits que pueden almacenar los tipos de datos inty uinto para números de coma flotante. Para almacenar un número de coma flotante se debe incluir una coma decimal en el número. Si se omite un separador decimal, el número se almacenará como un entero.

4.2.5 Tipo de datos String

El tipo de datos String representa una secuencia de caracteres de 16 bits. Las cadenas se almacenan internamente como caracteres Unicode empleando el formato UTF-16. Las cadenas son valores inmutables, igual que en el lenguaje de programación Java. Una operación sobre un valor de cadena (String) devuelve una nueva instancia de la cadena. El valor predeterminado de una variable declarada con el tipo de datos String es null. El valor null no es lo mismo que la cadena vacía (""), aunque ambos representan la ausencia de caracteres.

4.2.6 Tipo de datos uint

El tipo de datos uint se almacena internamente como un entero sin signo de 32 bits y consta del conjunto de enteros entre 0 y 4.294.967.295 (2^{32}- 1), ambos incluidos. El tipo de datos uint debe utilizarse en circunstancias especiales que requieran enteros no negativos. Por ejemplo, se debe utilizar el tipo de datos uint para representar valores de colores de píxeles, ya que el tipo de datos int tiene un bit de signo interno que no es apropiado para procesar valores de colores. Para valores enteros más grandes que el valor uint máximo, se debe utilizar el tipo de datos Number, que puede procesar valores enteros de 53 bits. El valor predeterminado para variables con tipo de datos uint es 0.

4.2.7 Tipo de datos Void

El tipo de datos void tiene un único valor: undefined. En las versiones anteriores de ActionScript, undefined era el valor predeterminado de las instancias de la clase Object. En ActionScript 3.0, el valor predeterminado de las instancias de Object es null. Si se intenta asignar el valor undefineda una instancia de la clase Object, Flash Player o Adobe AIR convertirán el valor en null. Sólo se puede asignar un valor undefined a variables que no tienen tipo. Las variables sin tipo son variables que no tienen anotación de tipo o utilizan el símbolo de asterisco (*) como anotación de tipo. Sólo se puede usar void como anotación de tipo devuelto.

4.2.8 Tipo de datos Object

El tipo de datos Object (objeto) se define mediante la clase Object. La clase Object constituye la clase base para todas las definiciones de clase en ActionScript. La versión del tipo de datos Object en ActionScript 3.0 difiere de la de versiones anteriores en tres aspectos. En primer lugar, el tipo de datos Object ya no es el tipo de datos predeterminado que se asigna a las variables sin anotación de tipo. En segundo lugar, el tipo de datos Object ya no incluye el valor undefined que se utilizaba como valor predeterminado de las instancias de Object. Por último, en ActionScript 3.0, el valor predeterminado de las instancias de la clase Object es null.

4.3 CONVERSIONES DE TIPOS

Se dice que se produce una conversión de tipo cuando se transforma un valor en otro valor con un tipo de datos distinto. Las conversiones de tipo pueden ser implícitas o explícitas.

4.3.1 Conversiones implícitas

Las conversiones implícitas se realizan en tiempo de ejecución en algunos contextos:

- En sentencias de asignación
- Cuando se pasan valores como argumentos de función
- Cuando se devuelven valores desde funciones
- En expresiones que utilizan determinados operadores, como el operador suma (+)

Para tipos definidos por el usuario, las conversiones implícitas se realizan correctamente cuando el valor que se va a convertir es una instancia de la clase de destino o una clase derivada de la clase de destino. Si una conversión implícita no se realiza correctamente, se producirá un error. Por ejemplo, el código siguiente contiene una conversión implícita correcta y otra incorrecta:

```
class A {}
class B extends A {}
var objA:A = new A();
var objB:B = new B();
var arr:Array = new Array();
objA = objB; // Conversion succeeds.
objB = arr; // Conversion fails.
```

Para tipos simples, las conversiones implícitas se realizan llamando a los mismos algoritmos internos de conversión que utilizan las funciones de conversión explícita. En las secciones siguientes se describen en mayor detalle estas conversiones de tipos simples.

4.3.2 Conversiones explícitas

Resulta útil usar conversiones explícitas cuando se compila en modo estricto, ya que a veces no se desea que una discordancia de tipos genere un error en tiempo de compilación. Esto puede ocurrir, por ejemplo, cuando se sabe que la coerción convertirá los valores correctamente en tiempo de ejecución. Por ejemplo, al trabajar con datos recibidos desde un formulario, puede ser interesante basarse en la coerción para convertir determinados valores de cadena en valores numéricos. El código siguiente genera un error de tiempo de compilación aunque se ejecuta correctamente en modo estándar:

```
var quantityField:String = "3"; var quantity:int = quantityField; // compile time error in strict mode
```

Si se desea seguir utilizando el modo estricto pero se quiere convertir la cadena en un entero, se puede utilizar la conversión explícita de la manera siguiente:

```
var quantityField:String = "3"; var quantity:int = int(quantityField); // Explicit conversion succeeds.
```

4.4 SINTAXIS

La sintaxis de un lenguaje define un conjunto de reglas que deben cumplirse al escribir código ejecutable.

4.4.1 Distinción entre mayúsculas y minúsculas

El lenguaje ActionScript 3.0 distingue mayúsculas de minúsculas. Los identificadores que sólo se diferencien en mayúsculas o minúsculas se considerarán identificadores distintos. Por ejemplo, el código siguiente crea dos variables distintas:

```
var num1:int; var Num1:int;
```

4.4.2 Sintaxis con punto

El operador de punto (.) permite acceder a las propiedades y los métodos de un objeto. La sintaxis con punto permite hacer referencia a una propiedad o un método de clase mediante un nombre de instancia, seguido del operador punto y el nombre de la propiedad o el método. Por ejemplo, considere la siguiente definición de clase:

```
class
DotExam
ple {
    public var prop1:String;
    public function
method1():void {} }
```

La sintaxis con punto permite acceder a la propiedad prop1 y al método method1() utilizando el nombre de la instancia creada en el código siguiente:

```
var myDotEx:DotExample = new DotExample(); myDotEx.prop1 = "hello";
myDotEx.method1();
```

4.4.3 Signos de punto y coma

Se puede utilizar el signo de punto y coma (;) para finalizar una sentencia. Como alternativa, si se omite el signo de punto y coma, el compilador dará por hecho que cada línea de código representa a una sentencia independiente. Como muchos programadores están acostumbrados a utilizar el signo de punto y coma para indicar el final de una sentencia, el código puede ser más legible si se usan siempre signos de punto y coma para finalizar las sentencias.

El uso del punto y coma para terminar una sentencia permite colocar más de una sentencia en una misma línea, pero esto hará que el código resulte más difícil de leer.

4.4.4 Paréntesis

Los paréntesis (()) se pueden utilizar de tres modos diferentes en ActionScript 3.0. En primer lugar, se pueden utilizar para cambiar el orden de las operaciones de una expresión. Las operaciones agrupadas entre paréntesis siempre se ejecutan primero. Por ejemplo, se utilizan paréntesis para modificar el orden de las operaciones en el código siguiente:

```
trace(2 + 3 * 4); // 14 trace((2 + 3) * 4); // 20
```

En segundo lugar, se pueden utilizar paréntesis con el operador coma (,) para evaluar una serie de expresiones y devolver el resultado de la expresión final, como se indica en el siguiente ejemplo:

```
var a:int = 2; var b:int = 3; trace((a++, b++, a+b)); // 7
```

Por último, se pueden utilizar paréntesis para pasar uno o más parámetros a funciones o métodos, como se indica en el siguiente ejemplo, que pasa un valor String a la función

```
trace():
trace("hello"); // hello
```

4.4.5 Comentarios

El código de ActionScript 3.0 admite dos tipos de comentarios: comentarios de una sola línea y comentarios multilínea. Estos mecanismos para escribir comentarios son similares a los equivalentes de C++ y Java. El compilador omitirá el texto marcado como un comentario.

Los comentarios de una sola línea empiezan por dos caracteres de barra diagonal (//) y continúan hasta el final de la línea. Por ejemplo, el código siguiente contiene un comentario de una sola línea:

```
var someNumber:Number = 3; // a single line comment
```

Los comentarios multilínea empiezan con una barra diagonal y un asterisco (/*) y terminan con un asterisco y una barra diagonal (*/).

4.4.6 Palabras clave y palabras reservadas

Las palabras reservadas son aquellas que no se pueden utilizar como identificadores en el código porque su uso está reservado para ActionScript. Incluyen las palabras clave léxicas, que son eliminadas del espacio de nombres del programa por el compilador. El compilador notificará un error si se utiliza una palabra clave léxica como un identificador. En la tabla siguiente se muestran las palabras clave léxicas de ActionScript 3.0.

as	break	case	catch
clase	const	continue	default
delete	do	else	extends
false	finally	for	function
if	implements	import	in
instanceof	interface	internal	is
native	new	null	package
private	protected	public	return
super	switch	this	throw
to	true	try	typeof
use	var	void	while
with			

Figura 6.4.6.1: Listado de Palabras Clave en ActionScript 3

4.4.7 Constantes

ActionScript 3.0 admite la sentencia const, que se puede utilizar para crear constantes. Las constantes son propiedades con un valor fijo que no se puede

modificar. Se puede asignar un valor a una constante una sola vez y la asignación debe realizarse cerca de la declaración de la constante.

Por ejemplo, si se declara una constante como un miembro de una clase, se puede asignar un valor a esa constante únicamente como parte de la declaración o dentro del constructor de la clase.

4.4.8 Operadores

Los operadores son funciones especiales que se aplican a uno o más operandos y devuelven un valor. Un operando es un valor (generalmente un literal, una variable o una expresión) que se usa como entrada de un operador. Por ejemplo, en el código siguiente, los operadores de suma (+) y multiplicación (*) se usan con tres operandos literales (2, 3 y 4) para devolver un valor. A continuación, se exponen en tablas los operadores de los que dispone Action Script 3 categorizados según su uso:

4.2.1 Operadores Aritméticos

Son los operadores empleados en operaciones matemáticas.

Operador	Descripción	Ejemplo
+	Suma	5 + 5 = 10
-	Resta	5 - 5 = 0
*	Multiplicación	5 * 5 = 25
/	División	5 / 5 = 1
%	Resto o Módulo	10%8 = 2
++	Incremento. Suma 1 al valor	valor++ equivaldría a valor = valor + 1
--	Decremento. Resta 1 al valor	valor-- equivaldría a valor = valor - 1

Tabla 8.4.8.1 Operadores Aritméticos

4.2.2 Operadores de Asignación

Asignan el valor de una variable.

Operador	Descripción	Ejemplo
=	Asigna a la variable de la izquierda el valor de la derecha	variable vale 3; variable = 5; variable vale 5;
+=	Suma con asignación. Le añade a la variable el valor de la derecha.	variable vale 3; variable += 5; variable vale 8;
-=	Resta con asignación. Le resta el valor de la derecha.	variable vale 3; variable -= 5; variable vale -2;
*=	Multiplicación con asignación.	variable vale 3; variable *= 5; variable vale 15;
/=	División con asignación	variable vale 15; variable /= 5; variable vale 3;

Tabla 9.4.8.2 Operadores de Asignación

4.2.3 Operadores de Comparación

Empleados en expresiones decondicionales, devuelven un valor lógico, verdadero (TRUE o 1) si la comparación es cierta, o falso (FALSE o 0) si no lo es.

Operador	Descripción	Ejemplo
>	Mayor que	6 > 5 es verdadero.
<	Menor que	6 < 5 es falso.
>=	Mayor o igual que	6 >= 5 es verdadero.
<=	Menor o igual que	6 >= 6 es verdadero.
==	Igual	'hola' == 'hola' es verdadero.
!=	Distinto	'hola' != 'hola' es verdadero.

Tabla 10.4.8.2 Operadores de Comparación

4.2.4 Operadores lógicos.

Evalúan valores lógicos. Normalmente se emplean para comparar dos expresiones con operadores relacionales, y devuelve verdadero o falso.

Operador	Descripción	Ejemplo
&&	And (Y) Devuelve verdadero si los dos valores son verdaderos	(6 > 5) && (1==1) devuelve verdadero (6 > 5) && (1==0) devuelve falso
||	Or (O) Devuelve verdadero si alguno de los valores es verdadero	(6 > 5) || (1==1) devuelve verdadero (6 > 5) || (1==0) devuelve verdadero (6 > 6) || (1==0) devuelve falso
!	Not (Negado) Devuelve verdadero si el valor era falso, y al revés.	!(9 > 2) devuelve falso !(9 ==9) devuelve falso

Tabla 11.4.8.2 Operadores de Lógicos

4.5 CONDICIONALES

ActionScript 3.0 proporciona tres sentencias condicionales básicas que se pueden usar para controlar el flujo del programa.

4.5.1 Condicional IF

La sentencia condicional if..else permite comprobar una condición y ejecutar un bloque de código si dicha condición existe, o ejecutar un bloque de código alternativo si la condición no existe. Por ejemplo, el siguiente fragmento de código comprueba si el valor de x es superior a 20 y genera una función trace() en caso afirmativo o genera una función trace() diferente en caso negativo:

```
if (x > 20) {
trace("x is > 20"); } else {
trace("x is <= 20"); }
```

Si no desea ejecutar un bloque de código alternativo, se puede utilizar la sentencia if

sin la sentencia else.

if..else if

Puede comprobar varias condiciones utilizando la sentencia condicional if..else if. Por ejemplo, el siguiente fragmento de código no sólo comprueba si el valor de x es superior a 20, sino que también comprueba si el valor de x es negativo:

```
if (x > 20) {
trace("x is > 20"); } else if (x < 0) {
trace("x is negative"); }
```

Si una sentencia if o else va seguida de una sola sentencia, no es necesario escribir dicha sentencia entre llaves. Por ejemplo, en el código siguiente no se usan llaves:

```
if (x > 0)
trace("x is positive"); else if (x < 0)
trace("x is negative"); else
trace("x is 0");
```

4.5.2 Condicional Switch

La sentencia switch resulta útil si hay varios hilos de ejecución que dependen de la misma expresión de condición. La funcionalidad que proporciona es similar a una serie larga de sentencias if..else if, pero su lectura resulta un tanto más sencilla. En lugar de probar una condición para un valor booleano, la sentencia switch evalúa una expresión y utiliza el resultado para determinar el bloque de código que debe ejecutarse. Los bloques de código empiezan por una sentencia case y terminan con una sentencia break. Por ejemplo, la siguiente sentencia switch imprime el día de la semana en función del número de día devuelto por el método Date.getDay():

```
var someDate:Date = new Date(); var dayNum:uint = someDate.getDay(); switch(dayNum)
{
case 0: trace("Sunday"); break;
case 1: trace("Monday"); break;
case 2: trace("Tuesday"); break;
case 3: trace("Wednesday"); break;
case 4: trace("Thursday"); break;
case 5: trace("Friday"); break;
case 6: trace("Saturday"); break;
default: trace("Out of range"); break;
}
```

4.6 BUCLES

Las sentencias de bucle permiten ejecutar un bloque específico de código repetidamente utilizando una serie de valores o variables. Adobe recomienda escribir siempre el bloque de código entre llaves ({}). Aunque puede omitir las llaves si el bloque de código sólo contiene una sentencia, no es recomendable que lo haga por la misma razón expuesta para las condicionales: aumenta la posibilidad de que las sentencias añadidas posteriormente se excluyan inadvertidamente del bloque de código. Si posteriormente se añade una sentencia que se desea incluir en el bloque de código, pero no se añaden las llaves necesarias, la sentencia no se ejecutará como parte del bucle.

4.6.1 Bucle for

El bucle for permite repetir una variable para un rango de valores específico. Debe proporcionar tres expresiones en una sentencia for: una variable que se establece con un valor inicial, una sentencia condicional que determina cuándo termina la reproducción en bucle y una expresión que cambia el valor de la variable con cada bucle. Por ejemplo, el siguiente código realiza cinco bucles. El valor de la variable icomienza en 0 y termina en 4, mientras que la salida son los números 0 a 4, cada uno de ellos en su propia línea.

```
var i:int; for (i = 0; i < 5; i++) {
trace(i); }
```

4.6.2 Bucle for..in

El bucle for..inrecorre las propiedades de un objeto o los elementos de un conjunto. Por ejemplo, se puede utilizar un bucle for..inpara recorrer las propiedades de un objeto genérico (las propiedades de un objeto no se guardan en ningún orden concreto, por lo que pueden aparecer en un orden aparentemente impredecible):

```
var myObj:Object = {x:20, y:30}; for (var i:String in myObj) {
trace(i + ": " + myObj[i]); } // output: // x: 20 // y: 30
```

También se pueden recorrer los elementos de un conjunto:

```
var myArray:Array = ["one", "two", "three"]; for (var i:String in myArray) {
trace(myArray[i]); } // output: // one // two // three
```

Lo que no se puede hacer es repetir las propiedades de un objeto si se trata de una instancia de una clase definida por el usuario, a no ser que la clase sea una clase dinámica. Incluso con instancias de clases dinámicas, sólo se pueden repetir las propiedades que se añadan dinámicamente.

4.6.3 Bucle for each..in

El bucle for each..in recorre los elementos de una colección, que puede estar formada por las etiquetas de un objeto XML o XMLList, los valores de las propiedades de un objeto o los elementos de un conjunto. Por ejemplo, como muestra el fragmento de código siguiente, el bucle for each..in se puede utilizar para recorrer las propiedades de un objeto genérico, pero al contrario de lo que ocurre con el bucle for..in, la variable de iteración de los bucles for each..in contiene el valor contenido por la propiedad en lugar del nombre de la misma:

```
var myObj:Object = {x:20, y:30}; for each (var num in myObj) {
trace(num); } // output: // 20 // 30
```

Se puede recorrer un objeto XML o XMLList, como se indica en el siguiente ejemplo:

```
var myXML:XML =
<users><fname>Jane</fname><fname>Susan</fname><fname>John</fname>
</users>;
for each (var item in myXML.fname) {
trace(item); } /* output Jane Susan John */
```

También se pueden recorrer los elementos de un conjunto, como se indica en este ejemplo:

```
var myArray:Array = ["one", "two", "three"]; for each (var item in myArray) {
trace(item); } // output: // one // two // three
```

No se pueden recorrer las propiedades de un objeto si el objeto es una instancia de una clase cerrada. Tampoco se pueden recorrer las propiedades fijas (propiedades

definidas como parte de una definición de clase), ni siquiera para las instancias de clases dinámicas.

4.6.4 Bucle while

El bucle while es como una sentencia if que se repite con tal de que la condición sea true. Por ejemplo, el código siguiente produce el mismo resultado que el ejemplo del bucle for:

```
var i:int = 0; while (i < 5) {
trace(i);
i++; }
```

Una desventaja que presenta el uso de los bucles while frente a los bucles for es que es más probable escribir un bucle infinito con bucles while. El código de ejemplo de bucle for no se compila si se omite la expresión que aumenta la variable de contador, mientras que el ejemplo de bucle while sí se compila si se omite dicho paso. Sin la expresión que incrementa i, el bucle se convierte en un bucle infinito.

4.6.5 Bucle do..while

El bucle do..while es un bucle while que garantiza que el bloque de código se ejecuta al menos una vez, ya que la condición se comprueba después de que se ejecute el bloque de código. El código siguiente muestra un ejemplo simple de un bucle do..while que genera una salida aunque no se cumple la condición:

```
var i:int = 5; do {
trace(i);
i++; } while (i < 5); // output: 5
```

4.7 PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos es una forma de organizar el código de un programa agrupándolo en objetos, que son elementos individuales que contienen información (valores de datos) y funcionalidad. La utilización de un enfoque orientado a objetos para organizar un programa permite agrupar partes específicas de la información (por ejemplo, información de una canción como el título de álbum, el

título de la pista o el nombre del artista) junto con funcionalidad o acciones comunes asociadas con dicha información (como "añadir pista a la lista de reproducción" o "reproducir todas las canciones de este artista"). Estos elementos se combinan en un solo elemento, denominado objeto (por ejemplo, un objeto "Album" o "MusicTrack"). Poder agrupar estos valores y funciones proporciona varias ventajas, como la capacidad de hacer un seguimiento de una sola variable en lugar de tener que controlar varias variables, agrupar funcionalidad relacionada y poder estructurar programas de maneras que reflejen mejor el mundo real.

A continuación se expone como maneja ActionScript 3 estos paradigmas

4.7.1 Clases

Una clase es una representación abstracta de un objeto. Una clase almacena información sobre los tipos de datos que un objeto puede contener y los comportamientos que un objeto puede exhibir. La utilidad de esta abstracción puede no ser apreciable al escribir scripts sencillos que sólo contienen unos pocos objetos que interactúan entre sí. Sin embargo, a medida que el ámbito de un programa crece y aumenta el número de objetos que hay que administrar, las clases pueden ayudar a obtener mayor control sobre la creación y la interacción mutua de los objetos.

Definiciones de clase

En las definiciones de clase de ActionScript 3.0 se utiliza una sintaxis similar a la utilizada en las definiciones de clase de ActionScript 2.0. La sintaxis correcta de una definición de clase requiere la palabra clave class seguida del nombre de la clase. El cuerpo de la clase, que se escribe entre llaves ({}), sigue al nombre de la clase. Por ejemplo, el código siguiente crea una clase denominada Shape que contiene una variable denominada visible:

```
Public class Shape
{
        var visible:Boolean = true;
}
```

Atributos de clase

ActionScript 3.0 permite modificar las definiciones de clase mediante uno de los cuatro atributos siguientes: Todos estos atributos, salvo internal, deben ser incluidos explícitamente para obtener el comportamiento asociado. Por ejemplo, si no se incluye el atributo dynamic al definir una clase, no se podrá añadir propiedades a una instancia de clase en tiempo de ejecución.

Atributo	Definición
dynamic	Permite añadir propiedades a instancias en tiempo de ejecución.
final	No debe ser ampliada por otra clase.
internal(valor predeterminado)	Visible para referencias dentro del paquete actual.
public	Visible para referencias en todas partes.

Tabla 12.4.8.2 Atributos de Clase

Un atributo se asigna explícitamente colocándolo al principio de la definición de clase, como se muestra en el código siguiente:

dynamic class Shape {}

Cuerpo de la clase

El cuerpo de la clase, que se escribe entre llaves, se usa para definir las variables, constantes y métodos de la clase. En el siguiente ejemplo se muestra la declaración para la clase Accessibility en la API de Adobe Flash Player:

```
public final class Accessibility
{
        public static function get active():Boolean;
        public static function updateProperties():void;
}
```

También se puede definir un espacio de nombres dentro de un cuerpo de clase. En

el siguiente ejemplo se muestra cómo se puede definir un espacio de nombres en el cuerpo de una clase y utilizarse como atributo de un método en dicha clase:

```
public class SampleClass
 {
        public namespace sampleNamespace;
        sampleNamespace function doSomething():void;
}
```

Métodos

Los métodos son funciones que forman parte de una definición de clase. Cuando se crea una instancia de la clase, se vincula un método a esa instancia. A diferencia de una función declarada fuera de una clase, un método sólo puede utilizarse desde la instancia a la que está asociado.

Los métodos se definen con la palabra clave function. Al igual que sucede con cualquier propiedad de clase, puede aplicar cualquiera de sus atributos a los métodos, incluyendo private, protected, public, internal, static o un espacio de nombres personalizado. Puede utilizar una sentencia de función como la siguiente:

```
public function sampleFunction():String {}
```

También se puede utilizar una variable a la que se asigna una expresión de función, de la manera siguiente:

```
public var sampleFunction:Function = function () {}
```

Constructores

Los métodos constructores, que a veces se llaman simplemente constructores, son funciones que comparten el nombre con la clase en la que se definen. Todo el código que se incluya en un método constructor se ejecutará siempre que una instancia de la clase se cree con la palabra clave new. Por ejemplo, el código siguiente define una clase simple denominada Example que contiene una sola propiedad denominada status. El valor inicial de la variable status se establece en la función constructora.

```
Class Example
{
        public var status:String;
        public function Example()
        {
            status = "initialized";

        }

}
var myExample:Example = new Example(); trace(myExample.status); // output: initialized
```

4.7.2 Herencia

La herencia es una forma de reutilización de código que permite a los programadores desarrollar clases nuevas basadas en clases existentes. Las clases existentes se suelen denominar clases base o superclases, y las clases nuevas se denominan subclases. Una ventaja clave de la herencia es que permite reutilizar código de una clase base manteniendo intacto el código existente. Además, la herencia no requiere realizar ningún cambio en la interacción entre otras clases y la clase base. En lugar de modificar una clase existente que puede haber sido probada minuciosamente o que ya se está utilizando, la herencia permite tratar esa clase como un módulo integrado que se puede ampliar con propiedades o métodos adicionales. Se utiliza la palabra clave extends para indicar que una clase hereda de otra clase.

La herencia también permite beneficiarse del polimorfismo en el código. El polimorfismo es la capacidad de utilizar un solo nombre de método para un método que se comporta de distinta manera cuando se aplica a distintos tipos de datos. Un ejemplo sencillo es una clase base denominada Shape con dos subclases denominadas Circle y Square. La clase Shape define un método denominado area() que devuelve el área de la forma. Si se implementa el polimorfismo, se puede llamar al método area()en objetos de tipo Circle y Square, y se harán automáticamente los cálculos correctos. La herencia activa el polimorfismo al permitir que las subclases hereden y redefinan (sustituyan) los métodos de la clase base. En el siguiente ejemplo, se redefine el método area() mediante las clases Circle y Square:

```
class Shape
{
	public function area():Number
	{
		return NaN;
	}
 }

class Circle extends Shape
{
	private var radius:Number = 1; override public function area():Number
	{
		return (Math.PI * (radius * radius));
	}
}

class Square extends Shape
{
	private var side:Number = 1; override public function area():Number
	{
	return (side * side);
	}
}

var cir:Circle = new Circle();
trace(cir.area()); // output: 3.141592653589793
var sq:Square = new Square();
trace(sq.area()); // output: 1
```

Como cada clase define un tipo de datos, el uso de la herencia crea una relación especial entre una clase base y una clase que la amplía. Una subclase posee todas las propiedades de su clase base, lo que significa que siempre se puede sustituir una instancia de una subclase como una instancia de la clase base. Por ejemplo, si un método define un parámetro de tipo Shape, se puede pasar un argumento de tipo Circle, ya que Circle amplía Shape, como se indica a continuación:

```
function draw(shapeToDraw:Shape) {}
var myCircle:Circle = new Circle(); draw(myCircle);
```

4.8 GESTIÓN DE EVENTOS

Un sistema de gestión de eventos permite a los programadores responder a la entrada del usuario y a los eventos del sistema de forma cómoda. El modelo de

eventos de ActionScript 3.0 no sólo resulta cómodo, sino que también cumple los estándares y está perfectamente integrado en la lista de visualización de Adobe® Flash® Player y Adobe® AIR™. El nuevo modelo de eventos está basado en la especificación de eventos DOM (modelo de objetos de documento) de nivel 3, una arquitectura de gestión de eventos estándar, y constituye una herramienta de gestión de eventos intuitiva y de grandes prestaciones para los programadores de ActionScript.

4.8.1 Fundamentos de la gestión de eventos

Los eventos se pueden considerar como sucesos de cualquier tipo en el archivo SWF que resultan de interés para el programador. Por ejemplo, la mayor parte de los archivos SWF permiten algún tipo de interacción con el usuario, ya sea algo tan sencillo como responder a un clic del ratón o algo mucho más complejo, como aceptar y procesar datos escritos en un formulario. Toda interacción de este tipo entre el usuario y el archivo SWF se considera un evento. Los eventos también pueden producirse sin interacción directa con el usuario, como cuando se terminan de cargar datos desde un servidor o se activa una cámara conectada.

En ActionScript 3.0, cada evento se representa mediante un objeto de evento, que es una instancia de la clase Event o de alguna de sus subclases. Los objetos de evento no sólo almacenan información sobre un evento específico, sino que también contienen métodos que facilitan la manipulación de los objetos de evento. Por ejemplo, cuando Flash Player
o AIR detectan un clic de ratón, crean un objeto de evento (una instancia de la clase MouseEvent) para representar ese evento de clic de ratón en concreto.

Tras crear un objeto de evento, Flash Player o AIR lo distribuyen, lo que significa que el objeto de evento se transmite al objeto que es el destino del evento. El objeto que actúa como destino del objeto de evento distribuido se denomina destino del evento. Por ejemplo, cuando se activa una cámara conectada, Flash Player distribuye un objeto de evento directamente al destino del evento que, en este caso, es el objeto que representa la cámara. No obstante, si el destino del evento está en la lista de

visualización, el objeto de evento se hace pasar por la jerarquía de la lista de visualización hasta alcanzar el destino del evento. En algunos casos, el objeto de evento se "propaga" de vuelta por la jerarquía de la lista de visualización usando la misma ruta. Este recorrido por la jerarquía de la lista de visualización se denomina flujo del evento.

Se pueden usar detectores de eventos en el código para detectar los objetos de evento. Los detectores de eventos son las funciones o métodos que se escriben para responder a los distintos eventos. Para garantizar que el programa responde a los eventos, es necesario añadir detectores de eventos al destino del evento o a cualquier objeto de la lista de visualización que forme parte del flujo del evento de un objeto de evento.

Cuando se escribe código para un detector de eventos, se suele seguir esta estructura básica (los elementos en negrita son marcadores de posición que se sustituyen en cada caso específico):

```
function eventResponse(eventObject:EventType):void
{
        // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

Este código realiza dos acciones. En primer lugar, define una función, que es la forma de especificar las operaciones que se llevarán a cabo como respuesta al evento. A continuación, llama al método addEventListener()del objeto de origen, básicamente "suscribiendo" la función al evento especificado de modo que se lleven a cabo las acciones de la función cuando ocurra el evento. Cuando el evento se produce finalmente, el destino de evento comprueba la lista de todas las funciones y métodos registrados como detectores de eventos. A continuación los llama de uno en uno, pasando el objeto de evento como parámetro.

Es necesario modificar cuatro elementos del código para crear un detector de eventos personalizado. En primer lugar se debe cambiar el nombre de la función por el nombre que se desee usar (es necesario realizar este cambio en dos lugares, donde en el código aparece **eventResponse**). Seguidamente, hay que especificar el nombre de

clase adecuado para el objeto de evento distribuido por el evento que se desea detectar (**EventType** en el código) y hay que indicar la constante apropiada para el evento específico (**EVENT_NAME** en el listado). En tercer lugar, es necesario llamar al método addEventListener()en el objeto que distribuirá el evento (**eventTarget** en este código). Opcionalmente, se puede cambiar el nombre de la variable utilizada como parámetro de la función (**eventObject** en el código).

4.8.2 Tareas comunes de la gestión de eventos

A continuación se enumeran diversas tareas comunes de la gestión de eventos que se describirán en este capítulo:

- Escribir código para responder a eventos
- Impedir que el código responda a eventos
- Trabajo con objetos de eventos
- Trabajo con el flujo del evento:
    - Identificar la información del flujo del evento
    - Detener el flujo del evento
- Impedir los comportamientos predeterminados
- Distribuir eventos desde las clases
- Creación de un tipo de evento personalizado

4.8.3 Conceptos y términos importantes con Eventos

La siguiente lista de referencia contiene términos importantes que se utilizan en este capítulo:

- Comportamiento predeterminado: algunos eventos incluyen un comportamiento que ocurre normalmente junto con el evento, denominado comportamiento predeterminado. Por ejemplo, cuando un usuario escribe texto en un campo de texto, se activa un evento de introducción de texto. El comportamiento predeterminado de ese evento es mostrar el carácter que se ha escrito en el campo de texto, si bien se

puede sustituir ese comportamiento predeterminado (si, por alguna razón, no se desea que se muestre el carácter escrito).

- Distribuir: notificar a los detectores de eventos que se ha producido un evento.
- Evento: algo que le sucede a un objeto y que dicho objeto puede comunicar a otros.
- Flujo del evento: cuando los eventos se producen en un objeto de la lista de visualización (un objeto que se muestra en pantalla), todos los objetos que contienen ese objeto reciben la notificación del evento y, a su vez, notifican a sus detectores de eventos. El proceso comienza en el escenario y avanza a través de la lista de visualización hasta el objeto en el que se ha producido el evento para después volver de nuevo hasta el escenario. Este proceso se conoce como el flujo del evento.
- Objeto de evento: objeto que contiene información acerca de un evento en concreto y que se envía a todos los detectores cuando se distribuye un evento.
- Destino del evento: objeto que realmente distribuye el evento. Por ejemplo, si el usuario hace clic en un botón que se encuentra dentro de un objeto Sprite que, a su vez, está en el escenario, todos esos objetos distribuirán eventos, pero el destino del evento es el objeto en el que se produjo el evento; en este caso, el botón sobre el que se ha hecho clic.
- Detector: función u objeto que se ha registrado a sí mismo con un objeto para indicar que debe recibir una notificación cuando se produzca un evento específico.

4.9 LOS OBJETOS DESDE EL CÓDIGO

Los Objetos, como ya se vio, son instancias de una determinada clase. Esto es, son representantes de una clase ya definida. Cada objeto tiene las propiedades y métodos propios de la clase, y normalmente son independientes unos de otros. Así, son objetos, por ejemplo, un botón, un clip de película, un gráfico o un sonido... es decir, que prácticamente todo es un objeto en Flash CS4.

Antes de empezar a exponer los objetos accesibles desde el código fuente es recomendable tener en cuenta que parten de la siguiente estructura jerárquica:

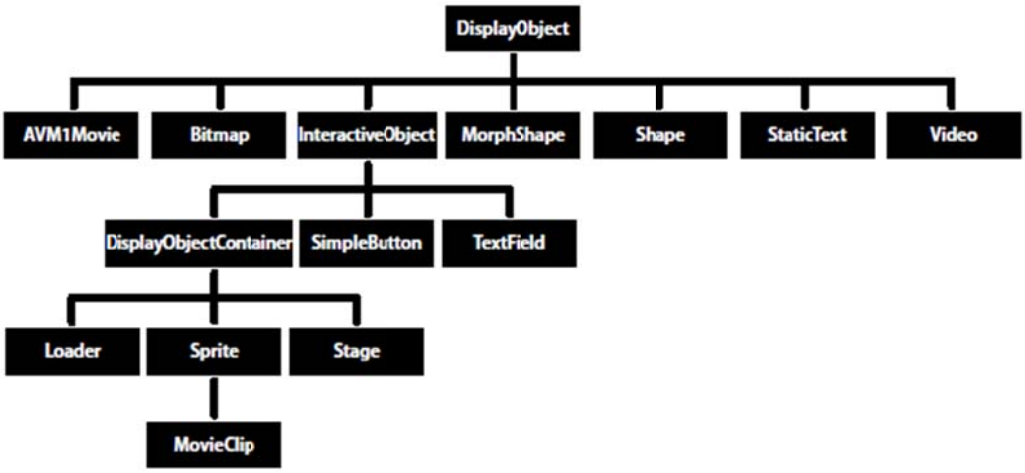


Figura 7.9.1: Estructura Jerárquica de los Objetos de Visualización

4.9.1 Objeto "Button" (Botón)

Los objetos de tipo Botón es un tipo de MovieClip especialmente pensado para que el usuario interactúe con él, permitiendo diferenciar entre sus estados, y crear una apariencia para cada uno.

Cuando se desee que una imagen que ya diseñada se comporte como un botón, bastará convertirla a botón y ya se la podrá usar los eventos típicos de un botón.

4.9.2 Objeto "MovieClip" (Clip de Película)

Cuando necesitemos crear una película Flash dentro de otra película, pero no se busque tener 2 ficheros separados ni molestarse en cargar una película u otra, se puede crear un objeto movieclip. Entre sus propiedades especiales destaca que los objetos "clip de película" tienen, internamente, una línea de tiempos que corre independiente de la línea de tiempos de la película principal de Flash, lo que permite crear animaciones tan complejas e independientes como se requiera.

4.9.3 Objeto "DisplayObject" (Objeto de visualización)

Esta clase es la clase padre de todos los objetos que se pueden ver en la película flash, como los Clips de película y botones, y define las propiedades y métodos comunes para todos ellos.

4.9.4 Objeto "Mouse" (Ratón)

El objeto mouse es uno de los objetos de Flash que ya está definido por Flash, pues hace referencia al ratón de Windows (al que manejará el usuario que vea la película flash). Con él se podrá acceder a las propiedades del ratón de Windows, tipo de cursos, efectos asociados, detección de su posición etc.

Vale la pena insistir en que su manejo no es análogo al de otros objetos como el botón, pues se pueden crear tantos botones como desee y hacer con ellos lo que decida, pero el objeto Mouse es único y actúa sobre el ratón del PC. Se puede decir que es un objeto "externo" que permite que otras partes del Sistema Operativo interactúen con la película Flash. Por tanto, es muy potente.

4.9.5 Objetos "Loader" y "URLLoader"

Los objetos Loader permiten cargar archivos para mostrarlos (imágenes, archivos swf, etc...) en la película, mientras que los objetos URLLoader permiten cargar información de archivos (archivos de texto, XML, páginas web...).

4.10 EVENTOS EN BOTONES

Los Botones tienen mucha utilidad siempre que se quiera que la película interactúe con el usuario. Dado que esto va a ser prácticamente siempre, es conveniente estudiar y entender bien algunos códigos típicos que habrá que usar para conseguir los propósitos.
Para tener el código organizado, es mejor crear una nueva capa e insertarlo ahí.

El siguiente es un ejemplo de asignación de evento al botón:

```
import flash.events.MouseEvent;
miBoton.addEventListener(MouseEvent.MOUSE_CLICK, funcionAlHacerClick);
function funcionAlHacerClick(event:MouseEvent):void
{
  gotoAndPlay(15)
}
```

Esta acción provoca que al hacer clic en el botón se redireccione el foco de visualización directamente al Fotograma número 15 de la película.

Utiliza la sentencia import para especificar el nombre completo de la clase, de modo que el compilador de ActionScript sepa dónde encontrarlo. La clase MouseEvent debe importarse en este caso, para ello:

```
import flash.events.MouseEvent;
```

4.11 NAVEGACIÓN MEDIANTE BOTONES

Uno de los elementos que más ayudarán a la navegación son los botones.En la imagense puede notar como mediante el uso los botones se puede crear la navegación entre las diferentes secciones.

Figura 8.11.1: Diseño de una Interfaz con Botones de Navegación

Así que el primer paso, después de haber creado la interfaz de la película en una capa, será crear e insertar los botones en una nueva capa para trabajar con mayor facilidad.

Para asignarle una acción a un botón es necesario darle un nombre de instancia. Para ello se escribirá el nombre que lo identifique en el Inspector de Propiedades, en este caso lo se lo ha llamadobtn_ingresar.

Figura 9.11.2: Diseño de una Interfaz con Botones de Navegación

Luego, es recomendable crear otra capa independiente para poder insertar las acciones que se necesitarán los botones, resta abrir el Panel Acciones y añadir el código que deberá realizar el botón.

Ésta es la parte más importante pues se deberá decidir a qué evento responderá el botón. Existen varios eventos que son capturados en Flash, entre los más importantes:

- MouseEvent.MOUSE_DOWN: ejecuta la acción al presionarse el botón.
- MouseEvent.MOUSE_UP: ejecuta la acción al soltarse el botón (después de haberlo presionado).
- MouseEvent.MOUSE_OVER: ejecuta la acción al desplazar el cursor dentro del botón
- MouseEvent.MOUSE_OUT: ejecuta la acción al desplazar el cursor fuera del botón.

Nota: ActionScript diferencia entre mayúsculas y minúsculas, por lo que si se escribe, por ejemplo, mouse_upno será reconocido.

El Código para agregar el evento sería:

```
btn_ingresar.addEventListener(MouseEvent.MOUSE_UP, nombreFuncion);

functionnombreFuncion(e:MouseEvent):void
{
//código de la función
}
```

4.12 CONTENEDORES Y LISTAS DE VISUALIZACIÓN

Con respecto a los elementos que se ven en la película con ActionScript 3, hay que tener claros estos conceptos:

- Los objetos que se pueden ver son llamados objetos visibles o de visualización, y todos heredan de la claseDisplayObjectysiempre han de estar dentro de un contenedor para que se vean.
- Los objetos están agrupados dentro de un contenedor, que hace de elemento padre. A su vez, dentro de un contenedor se pueden tener otros contenedores con sus respectivos elementos. Los contenedores pertenecen a la clase DisplayObjectContainer, y aunque pueda parecer lioso, a su vez un contenedor es un objeto de visualización, y se puede tratar como tal.
- La lista de visualización es cómo están ordenados los objetos dentro del contenedor, y establece el orden de apilamiento de los objetos.

4.12.1 Los contenedores:

En la película flash se pueden tener cuatro tipos de contenedores:

- La escena (stage). Es el contenedor general de la película. Todo lo que se ve, está dentro de la escena.

- Loader. Permite cargar un archivo externo en él. Tema que se tratará más adelante
- MovieClip. Aunque normalmente no se lo trata como tal, un MovieClip contiene un archivo SWF con una línea de tiempo propia. Por ejemplo, dentro de él se puede acceder a los distintos símbolos que lo forman.
- Sprite. Es como una carpeta, a la que se le puede ir añadiendo y quitando objetos. Se pueden crear tantos sprites como desee.

Se pueden añadir objetos a un contenedor que pasaran a ser elementos hijos del contenedor. Para ello se puede utilizar uno de los métodos de los contenedores:

```
nombreContenedor.addChild(nombreObjeto1);
nombreContenedor.addChild(nombreObjeto2);
```

4.12.2 Listas de visualización:

Los elementos añadidos a un contenedor forman su lista de visualización. La posición dentro de esta lista establece el orden de apilamiento. Es decir, los objetos con un índice menor se verán por debajo de los elementos con un índice mayor.

La lista de visualización puede alterarse mediante los métodos implementados por la clase padre DisplayObjectContainer, entre los mas usados están.

- numChildren - Esta propiedad devuelve el número de elementos de la lista. En el ejemplo anterior, nombreContenedor.numChildren devuelve 2.
- getChildIndex(objeto) - Permite conocer el índice de un elemento.
- addChild(objeto) - Añade el elemento al final de la lista, encima del resto.
- addChildAt(objeto, índice) - Añade un elemento y permite indicar en qué posición colocarlo. Por ejemplo, si se tiene en la lista el objeto3, y se desea añadir el objeto7 justo antes que este para que quede debajo, se puede emplear: addChildAt(objeto7, getChildIndex(objeto3)); para saber el índice del objeto3 y colocar ahí el objeto7, desplazando el resto hacia el final.
- setChildIndex(objeto, índice) - Permite cambiar el orden de un objeto dentro de la lista.

- getChildByName(nombre_instancia) - Permite obtener un objeto conociendo su nombre de instancia.
- getChildAt(índice) - Permite obtener un objeto conociendo su índice.
- contains(objeto) - Devuelve verdadero si el objeto ya está en la lista.
- removeChild(objeto) - Quita el objeto indicado de la lista.
- removeChildAt(objeto) - Quita de la lista el objeto con el índice indicado.

Nota: Antes de quitar un objeto de la lista, es recomendable borrar sus eventos si los tiene, ya que esto puede producir errores. Además, si no se lo quita, el objeto sigue ocupando memoria. Para borrar un evento, habrá de utilizarse el método removeEventListener, con los mismos parámetros que se empleó en addEventListener. Por ejemplo:

```
objeto.removeEventListener(Event.ENTER_FRAME, nombreFuncion);
```

4.13 FORMULARIOS

Al utilizar formularios se podrá utilizar muchos elementos. Pero los principales serán siempre los mismos: cajas de texto y botones.

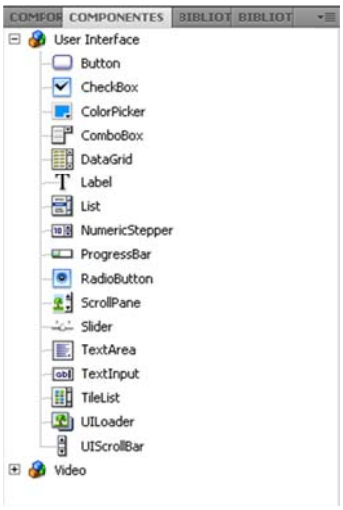


Figura 10.13.1: Paleta de Controles de Formulario

De vez en cuando utilizará otros elementos como los radioButtons, checkBoxes, comboBoxes o listBoxes. Flash ofrece estos objetos como componentes. Para acceder a ellos se puede que abrir el Panel Componentes desde Ventana → Componentes.

Una vez abierto el panel seleccionarUser Interface para desplegar sus elementos y poder ver todos los componentes disponibles.

Incluso para la introducción de texto en formularios es aconsejable el uso de componentes, pues se verá que poseen propiedades que las simples cajas de texto no tienen.

Para utilizar alguno de estos componentes basta con arrastrarlo del panel al escenario, o arrastrarlo a la biblioteca para utilizarlo más tarde.

En cualquier caso, cuando se haya añadido el componente a la película deberádársele un nombre de instancia para poder acceder a él desde el código y configurar sus opciones en el Panel Propiedades y en el panel Inspector de Componentes:

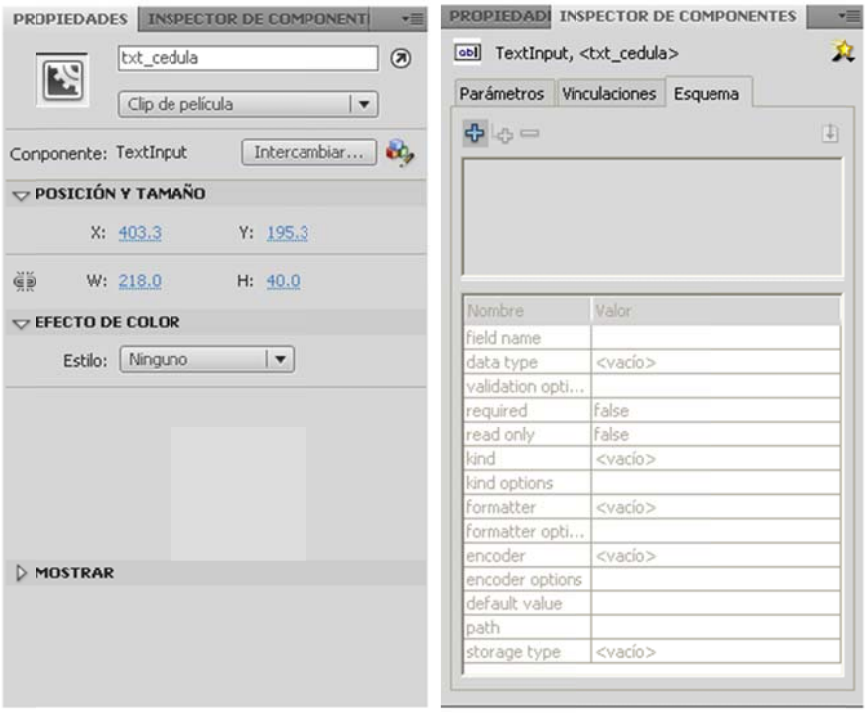


Figura 11.13.1: Panel de Propiedades de los Controles de Formulario

A continuación se exponencuáles son las opciones para los diferentes componentes:

- TextInput (Introducción de texto):
    - editable: true o false. Permite que el texto se pueda editar o no.
    - password: true o false. Oculta el contenido del texto mostrándose un asterisco por carácter.
    - text: Indica el texto inicial de la caja.

- TextArea (Área de texto):

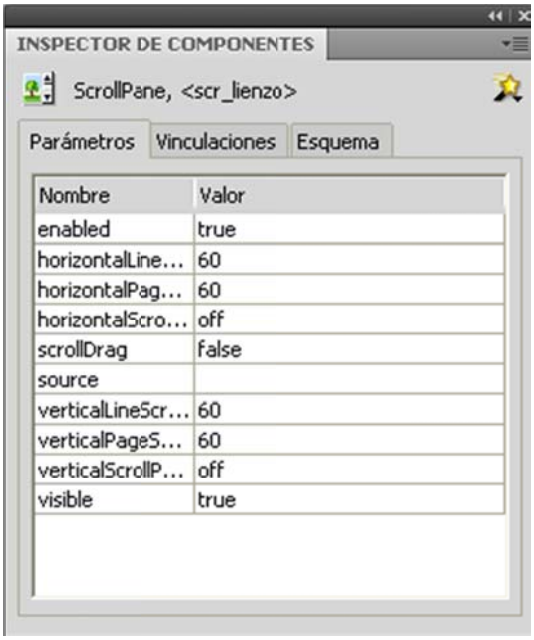


Figura 12.13.2: Inspector de Componentes

    - editable: true o false. Permite que el texto se pueda editar o no.
    - html: true o false. Permite que se acepte contenido HTML dentro de la caja de texto. Propiedad muy útil para incluir enlaces en el texto.
    - text: Indica el texto inicial de la caja.
    - wordWrap: true o false. Permite que se pueda realizar un desplazamiento del texto de arriba abajo. En caso de que no se permita (false) cuando el texto sobre pase el área del componente aparecerá una barra de desplazamiento que permitirá mover el texto de izquierda a derecha.

- Button (Botón):
    - icon: Añade un icono al botón. Para insertar un icono se debe crear un gráfico o clip de película y guardarlo en la Biblioteca. Una vez allí se lo puede seleccionar y haciendo clic derecho sobre él, asignarse suVinculación. Habrá de marcarse la casilla Exportar para ActionScript en el cuadro de diálogo que aparecerá y le damos un nombre en Identificador. Este nombre es el que deberemos escribir en el campo icon del componente botón.
    - label: Texto que se leerá en el botón.
    - labelPlacement: left, right, top o bottom. Indica la posición de la etiqueta de texto en caso de que se utilice junto a un icono. Respectivamente, izquierda, derecha, arriba y abajo.
    - selected: true o false. Indica si el botón se encuentra seleccionado.
    - toggle: true o false. Cuando se encuentra a true hace que el botón pueda tomar dos posiciones, presionado y no presionado.

- RadioButton (Botón de opción):
    - data: Especifica los datos que se asociarán al RadioButton. La propiedad data puede ser cualquier tipo de datos. Podemos acceder a esta propiedad a través de código para ver que contiene.
    - groupName: Nombre del grupo. En un grupo de botones de opción sólo uno de ello puede estar seleccionado. Definiremos este grupo mediante esta propiedad. Todos los botones que tengan el mismo nombre en groupName pertenecerán al mismo grupo.
    - label: Texto que se leerá al lado del botón.
    - labelPlacement: left, right, top o bottom. Indica la posición de la etiqueta de texto respecto al botón. Respectivamente, izquierda, derecha, arriba y abajo.
    - selected: true o false. Indica si el botón se haya seleccionado o no. De nuevo, en un mismo grupo sólo un botón de opción puede estar seleccionado.

- CheckBox (Casilla de verificación):
    - label: Texto que se leerá al lado de la casilla.
    - labelPlacement: left, right, top o bottom. Indica la posición de la etiqueta de texto respecto a la casilla. Respectivamente, izquierda, derecha, arriba y abajo.

- selected: true o false. Indica si la casilla de verificación se haya seleccionada.

- ComboBox (Lista desplegable):
  - data: Matriz donde determinaremos el valor qué devolverá el componente al seleccionar determinada posición.
  - editable: true o false. Permite la edición del campo. Mediante ActionScript podemos hacer que se añadan nuevos elementos a la lista.
  - labels: Matriz donde se determinará el nombre de los elementos de la lista. Estos elementos se corresponderán uno a uno a los valores de la matriz introducida en data. Para ambas propiedades se abrirá el siguiente cuadro de diálogo:

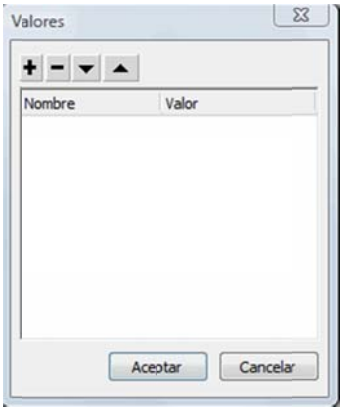


Figura 13.13.3: Editor de Valores de ComboBox

Desde aquí se puede añadir o quitar elementos utilizando los botones y . O alterar el orden de éstos subiéndolos o bajándolos en la lista con los botones y .

- rowCount: Número máximo de elementos visibles en la lista. Si este número es superado por los elementos se añadirá una barra de desplazamiento.

- List (Lista):

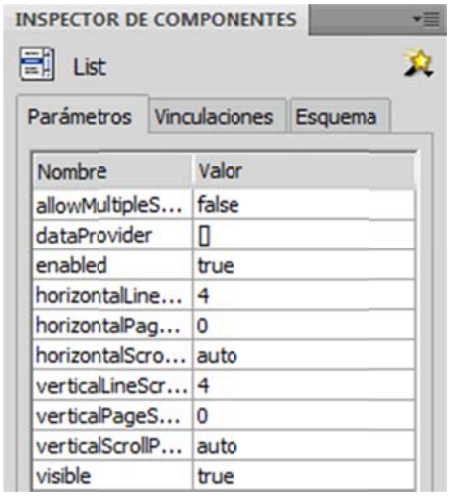


Figura 14.13.4: Inspector de Componentes (propiedades de un List)

- data: Matriz donde se determina el valor qué devolverá el componente al seleccionar determinada posición.
- labels: Matriz donde se establece el nombre de los elementos de la lista. Estos elementos se corresponderán uno a uno a los valores de la matriz introducida en data. Para ambas propiedades se abrirá el mismo cuadro de diálogo visto para el ComboBox.
- multipleSelection: true o false. Permite la selección múltiple de elementos manteniendo la tecla Ctrl presionada. También es posible seleccionar un rango de elementos seleccionando uno de ellos y pulsando la tecla Shift mientras se selecciona otro diferente, todos los elementos intermedios resultarán también seleccionados.
- rowHeight: Número máximo de filas visibles en la lista. Si este número es superado por los elementos se añadirá una barra de desplazamiento.

Nota: Todas las propiedades mencionadas para cada uno de los componentes son accesibles a través de ActionScript escribiendo el nombre de instancia del componente seguido de un punto y el nombre de la propiedad:

```
miBoton.label  = "Haz clic aquí";
```

4.14 CARGA EXTERNA DE OBJETOS DE VISUALIZACIÓN

Los objetos Loader se usan para cargar archivos SWF y archivos de gráficos en una aplicación. La clase Loader es una subclase de la clase DisplayObjectContainer. Un objeto Loader sólo puede contener un objeto de visualización secundario en su lista de visualización: el objeto de visualización que representa el archivo SWF o archivo de gráficos que se carga. Cuando se añade un objeto Loader a la lista de visualización, como en el código siguiente, también se añade a la lista de visualización el objeto de visualización secundario cargado, una vez que se ha cargado:

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

Cuando se carga el archivo SWF o la imagen, se puede mover el objeto de visualización cargado a otro contenedor de objeto de visualización, como el objeto DisplayObjectContainer container que se muestra en este ejemplo:

```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
container.addChild(pictLdr.content);
}
```

4.16.1 Supervisión del progreso de carga

Cuando se empieza a cargar el archivo, se crea un objeto LoaderInfo. Un objeto LoaderInfo proporciona información tal como el progreso de carga, los URL del cargador y el contenido cargado, el número total de bytes del medio, y la anchura y la altura nominal del medio. Un objeto LoaderInfo también distribuye eventos para supervisar el progreso de la carga.

El siguiente diagrama muestra los diferentes usos del objeto LoaderInfo para la instancia de la clase principal del archivo SWF, para un objeto Loader y para un objeto cargado por el objeto Loader:

Figura 15.16.1: Proceso de Carga de un Archivo Externo

Se puede acceder al objeto LoaderInfo como una propiedad tanto del objeto Loader como del objeto de visualización cargado. En cuanto comienza la carga, se puede acceder al objeto LoaderInfo a través de la propiedad contentLoaderInfo del objeto Loader. Cuando finaliza la carga del objeto de visualización, también es posible acceder al objeto LoaderInfo como una propiedad del objeto de visualización cargado, a través de la propiedad loaderInfo del objeto de visualización. La propiedad loaderInfo del objeto de visualización hace referencia al mismo objeto

LoaderInfo al que se refiere la propiedad contentLoaderInfo del objeto Loader. Dicho de otro modo, un objeto LoaderInfo se comparte entre un objeto cargado y el objeto Loader que lo ha cargado (es decir, entre el contenido cargado y el cargador). Para acceder a las propiedades del contenido cargado, es necesario añadir un detector de eventos al objeto LoaderInfo, como se muestra en el siguiente código:

```
import flash.display.Loader;
import flash.display.Sprite;
import flash.events.Event;
var ldr:Loader = new Loader();
var urlReq:URLRequest = new URLRequest("Circle.swf");
ldr.load(urlReq);
ldr.contentLoaderInfo.addEventListener(Event.COMPLETE, loaded);
addChild(ldr);
function loaded(event:Event):void
{
        var content:Sprite = event.target.content;
        content.scaleX = 2;
}
```

4.16.2 Especificación del contexto de carga

Cuando se carga un archivo externo en Flash Player o AIR con el método load() o loadBytes(), se puede especificar opcionalmente un parámetro context. Este parámetro es un objeto LoaderContext.
La clase LoaderContext incluye tres propiedades que permiten definir el contexto de uso del contenido cargado:

• checkPolicyFile: utilice esta propiedad sólo si carga un archivo de imagen (no un archivo SWF). Si establece esta propiedad en true, Loader comprueba el servidor de origen de un archivo de política. Sólo es necesaria en el contenido procedente de dominios ajenos al del archivo SWF que contiene el objeto Loader. Si el servidor concede permisos al dominio de Loader, el código ActionScript de los archivos SWF del dominio de Loader puede acceder a los datos de la imagen cargada y, por lo tanto, se puede usar el comando BitmapData.draw() para acceder a los datos de la imagen cargada.

Tenga en cuenta que un archivo SWF de otros dominios ajenos al del objeto Loader puede llamar a Security.allowDomain() para permitir el uso de un dominio específico.

• securityDomain: utilice esta propiedad sólo si carga un archivo SWF (no una imagen). Esta propiedad se especifica en un archivo SWF de un dominio ajeno al del archivo que contiene el objeto Loader. Si se especifica esta opción, Flash Player comprueba la existencia de un archivo de política y, en caso de que exista uno, los archivos SWF de los dominios permitidos en el archivo de política entre dominios pueden reutilizar los scripts del contenido SWF cargado. Se puede especificar flash.system.SecurityDomain.currentDomain como este parámetro.

• applicationDomain: utilice esta propiedad solamente si carga un archivo SWF escrito en ActionScript 3.0 (no una imagen ni un archivo SWF escritos en ActionScript 1.0 ó 2.0). Al cargar el archivo, se puede especificar que se incluya en el mismo dominio de aplicación del objeto Loader; para ello, hay que establecer el parámetro applicationDomain en flash.system.ApplicationDomain.currentDomain. Al colocar el archivo SWF cargado en el mismo dominio de aplicación, se puede acceder a sus clases directamente. Esto puede ser muy útil si se carga un archivo SWF que contiene medios incorporados, a los que se puede tener acceso a través de sus nombres de clase asociados.

A continuación se muestra un ejemplo de comprobación de un archivo de política durante la carga de un mapa de bits de otro dominio:

```
var context:LoaderContext = new LoaderContext();
context.checkPolicyFile = true;
var urlReq:URLRequest = new
URLRequest("http://www.[your_domain_here].com/photo11.jpg");
var ldr:Loader = new Loader();
ldr.load(urlReq, context);
```

A continuación se muestra un ejemplo de comprobación de un archivo de política durante la carga de un archivo SWF de otro dominio, con el fin de colocar el archivo en el mismo entorno limitado de seguridad que el objeto Loader. Además, el código

añade las clases del archivo SWF cargado al mismo dominio de aplicación que el del objeto Loader:

```
var context:LoaderContext = new LoaderContext();
context.securityDomain = SecurityDomain.currentDomain;
context.applicationDomain = ApplicationDomain.currentDomain;
var urlReq:URLRequest = new
URLRequest("http://www.[your_domain_here].com/library.swf");
var ldr:Loader = new Loader();
ldr.load(urlReq, context);
```

CAPÍTULO V

AMFPHP

AMFPHP es una implementación en PHP, gratuita y de código abierto del AMF (Action Message Format), el cual es un formato binario basado en SOAP. AMF permite la serialización binaria de objetos y tipos nativos de ActionScript 2 y ActionScript 3 para ser enviados a servicios del lado del servidor. AMFPHP permite que aplicaciones realizadas en Flash, Flex y AIR puedan comunicarse directamente con objetos de clases de PHP.

Prácticamente, la forma en la que funciona se resume en que se pueden crear clases en PHP, acceder a los métodos de esas clases por medio de Flash, Flex o AIR y obtener los datos retornados por esos métodos. Por ejemplo, se podría crear una clase para conectarse a una base de datos en Oracle. Esta clase podría contener un método para insertar datos a la base, otro método para generar consultas y obtener en una aplicación de Flash, Flex o AIR los registros regresados por la consulta.

AMFPHP hace uso de servicios web para la intercomunicación con Flash, a continuación estos servicios serán detallados

5.1 WEB SERVICES

Un Servicio Web (Web Service), es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, generando así formas de comunicar varios lenguajes e incluso plataformas, tendiendo a un entorno donde el principal objetivo es la interoperatibilidad.

Un Servicio Web plantea una arquitectura como la expuesta en el siguiente gráfico

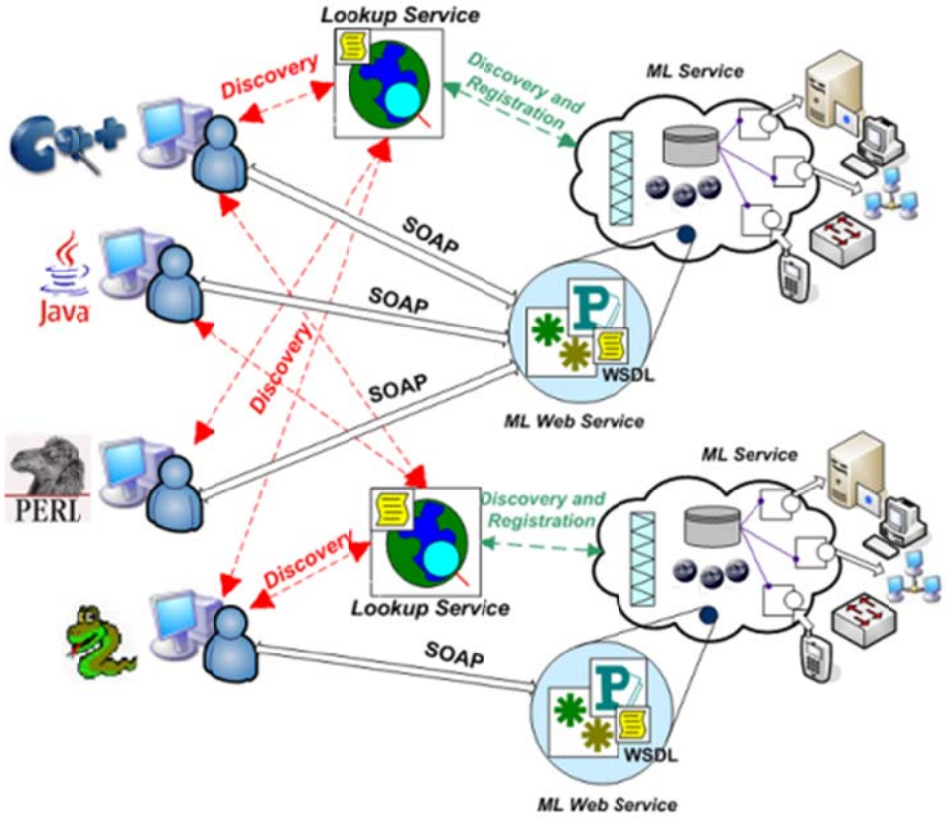


Figura 16.1: Arquitectura de Un Servicio Web

5.1.1. Beneficios de un Web Service

Las ventajas que ofrecen los web services, se orientan a la intercomunicación, más comúnmente, a continuación se citan algunos aspectos más específicos:

- Permiten que software de diferentes compañías puedan ser combinados fácilmente para proveer servicios integrados sin importar su ubicación geográficos.

- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas la W3C[4], por tanto son estándares totalmente públicos.

- Al tomar como plataforma base el protocolo HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

[4]W3C: World Wide Web Consortium

5.1.2. Componentes de un Web Service

El caso más común de los servicios web, casi siempre ha englobado a 4 protocolos, que al funcionar y comunicarse entre sí constituyen un servicio web.

Los 4 protocolos son los siguientes:

- XML
- WSDL
- UDDI
- SOAP

Estos protocolos se detallan a continuación

5.1.3. XML (Extensible Markup Language)

El propósito principal del lenguaje XML es el de facilitar la transferencia de datos a través de diferentes plataformas, especialmente las conectadas a Internet. Todo esto se da ya que XML es un lenguaje que es ampliamente utilizado para definición de valores, estructuras y propiedades en general.

XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no se aplica únicamente en Internet, sino que se utiliza como un estándar para el intercambio de información estructurada entre diferentes plataformas.

Entre los beneficios q ofrece XML se pueden citar los siguientes

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.

5.1.4. WSDL (Web Services Description Language)

Un documento WSDL no es más que un documento XML que describe ciertas características propias de un servicio web, así como su localización y aquellos parámetros y métodos que soporta.

WSDL se usa a menudo en combinación con SOAP y XML. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar que funciones están disponibles en el servidor.

Un documento WSDL define un servicio web utilizando elementos XML, como:

Los puertos de WSDL <portType>

Es el elemento XML de WSDL que define el servicio web, así como las operaciones posibles mediante dicho servicio y los mensajes vinculados. <portType> cumple una función análoga a la de una función de biblioteca en programación clásica o a la de una clase en programación orientada a objetos.

Los mensajes WSDL <message>

Este elemento define los datos que participan en una operación. Cada mensaje puede tener una o varias partes, y cada parte puede considerarse como si fuera los parámetros que se pasan en la llamada a una función en programación clásica o un método en programación orientada a objetos.

Los tipos de datos en WSDL <types>

Mediante este elemento se definen los tipos de datos utilizados en el servicio web. Para ello, WSDL utiliza XML.

Los vínculos en WSDL <binding>

Los enlaces o vínculos de WSDL permiten la definición de los formatos de mensaje y de protocolo para los servicios web.

La estructura de un WDSL podria ser semejante a la siguiente:

<definitions>
<types>

Los tipos de datos...

</types>
<message>

Las definiciones del mensaje...

</message>
<portType>

Las definiciones de operación ...

</portType>
<binding>

Las definiciones de protocolo...

</binding>
</definitions>

5.1.5. SOAP (Simple Object Access Protocol)

Es un protocoloestándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Con este protocolo se pueden realizar RPC (remote procedure calls), es decir, se podría bien desde el cliente o servidor realizar peticiones mediante http a un servidor web. Los mensajes deben tener un formato determinado empleando XML para encapsular los parámetros de la petición.

SOAP ofrece muchos beneficios y utilidades, entre ellos están las siguientes:

- Aprovecha los estándares existentes en la industria: SOAP extiende de estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se mencionó SOAP no define un medio de trasporte para el envío de mensajes, los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.

- Permite la interoperabilidad entre múltiples entornos: SOAP está desarrollado en base a estándares existentes, por lo que las aplicaciones que se ejecuten en plataformas con estos estándares, pueden comunicarse a través de mensajes DOAP con otras aplicaciones y en otras plataformas.

- No se encuentra fuertemente asociado a ningún protocolo de transporte: Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

- No está asociado con ningún lenguaje: SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.

5.1.6. UDDI (Universal Description, Discovery And Integration)

UDDI son las siglas del catálogo de negocios de Internet. El registro en el catálogo se hace en XML.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

Los datos manejados por UDDI se dividen en tres categorías:

- Páginas Blancas: Con información general sobre una empresa (nombre, descripción, información de contacto, dirección y teléfono).

- Páginas Amarillas es muy similar a su equivalente telefónico, e incluyen categorías de catalogación industrial tradicionales, ubicación geográfica, etc. Mediante el uso de códigos y claves predeterminadas, los negocios se pueden registrar y así facilitar a otros servicios la búsqueda usando estos índices de clasificación.
- Páginas Verdes: Con información técnica sobre un servicio web. Generalmente esto incluye un apuntador a la especificación externa y una dirección en la que invocar el servicio.

- **ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION.**Tiene 140 países miembros se encarga de desarrollar estándares para mejorar el comercio internacional

5.2 PUBLICAR UN MÉTODO PHP COMO WEB SERVICE MEDIANTE AMFPHP

Para publicar un método PHP como un Web Service, se debe definir un llamado archivo de servicios, que AMFPHP interpretará, cuyos métodos internamente definidos él automáticamente convertirá en Web Service.

Lo primero en este caso es configurar la utilidad “Browser” de AMFPHP para poder probar el archivo de definiciones, para ello se puede acceder a la misma desde cualquier navegador web utilizando la dirección:

http://localhost/amfphp/browser/

Aparecerá la primera vez una pantalla como la siguiente:

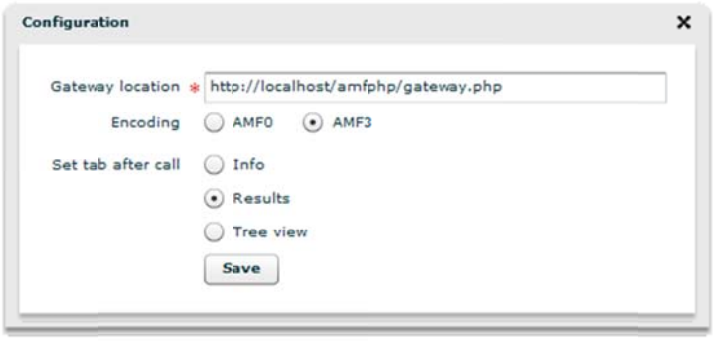


Figura 17.2.1: Arquitectura de Un Servicio Web

Desde esta pantalla se debe configurar los parámetros de funcionamiento del Gateway o portal de comunicación de AMFPHP

El archivo de definiciones debe Contener una clase, con el mismo nombre del archivo mismo, y cada método que se defina en ella será visible como si fuese un servicio web, todo esto sucede a que los procesos de publicación de AMFPHP son totalmente transparentes al usuario, es decir, todo lo que son definiciones WSDL, XML, etc. son automáticamente generadas y estructuradas para evitar tareas adicionales a los programadores

Todos los archivos de definiciones deben estar ubicados en la ruta “\htdocs\amfphp\services\”

Dada la situación anteriormente mencionada todo método definido en la clase o archivo de definiciones, será visible de inmediato en el Browser de AMFPHP

Ahora, mediante un ejemplo se explicará cómo realizar la implementación de la intercomunicación, para ello el archivo de definiciones llamado PruebaAmfPhp, contendrá una clase con el mismo nombre e implementará 2 funciones, la primera obtiene un Array de registros desde una base de datos, y la retornará hacia ActionScript 3 en forma de objeto igualmente Array.

El archivo de definiciones sería el siguiente:

```
<?php
require_once ($_SERVER['DOCUMENT_ROOT'].'/kiosko/clases/bd/Conexion.php');

class PruebaAmfPhp
{
   /**
    * Este metodo sirve para recuperar el listado de todas las solicitudes ordenadas
alfabeticamente
    * @returns Regresa el arreglo de todas las solicitudes.
*/
function getRsetSolicitudes()
   {
$o_conexion = new Conexion();
$o_conexion->abrirConexion();
$v_sql = "select * from KIM_SOLICITUD order by SOL_DESCRIPCION";
$v_consulta = oci_parse($o_conexion->o_enlace, $v_sql);
oci_execute($v_consulta);
while ($row = oci_fetch_array($v_consulta,OCI_RETURN_NULLS))
      {
         $data_array[] = $row;
      }
return($data_array);
}

/**
    * Este método sirve para recuperar el listado de todas las solicitudes ordenadas
    * Alfabéticamente
    * @returns Regresa el arreglo de todas las solicitudes.
    */
function getSolicitudPorCodigo ($id)
   {
      $o_conexion = new Conexion();
      $o_conexion->abrirConexion();
$v_sql = "select * from KIM_SOLICITUD where sol_id_solicitud=".$id;
$v_consulta = oci_parse($o_conexion->o_enlace, $v_sql);
oci_execute($v_consulta);
while ($row = oci_fetch_array($v_consulta,OCI_RETURN_NULLS))
      {
         $data_array[] = $row;
      }
```

```
return($data_array);
}
/**
    * Este metodo sirve para recuperar máximo codigo de Solicitud
    * @returns Regresa el valor máximo de codigo de Solicitud
*/
function getMaxId()
   {
      $n_max = 0;
$o_conexion = new Conexion();
$o_conexion->abrirConexion();
$v_sql = "select max(sol_id_solicitud) id from kim_solicitud";
$v_consulta = oci_parse($o_conexion->o_enlace, $v_sql);
oci_execute($v_consulta);
while (($a_resultado = oci_fetch_array($v_consulta, OCI_BOTH)))
{
         $n_max = $a_resultado['ID'];
      }
      $o_conexion->cerrarConexion();
return $n_max;
   }
}
?>
```

En este archivo puede notarse primeramente la posibilidad de hacer inclusiones de clases externas, mediante instrucciones del tipo require_once, en este caso se está llamando a una clase que implementa la conexión a Oracle. Esta es una buena práctica, en especial es un punto a favor si se desea manejar una arquitectura a 3 o más capas.

```
require_once ($_SERVER['DOCUMENT_ROOT'].'/kiosko/clases/bd/Conexion.php');
```

También hay que recalcar que un método implementado puede devolver ya sean valores de variables primitivas o de forma más flexible un Array que combine navegación por índices o por llaves como es el caso del método getRsetSolicitudes que una vez que sea captado en ActionScript 3 mantendrá igualmente dichas ventajas, en el caso del método getMaxId, lo que se retorna es un valor numérico que en ActionScript 3 será tratado como int

Otro aspecto importante es el hecho de comentar el código, muchas veces es una tarea obviada pero en este caso es altamente importante ya que será el descriptivo que muestre el browser de AMFPHP al momento de permitir pruebas con los métodos definidos en cualquiera de los archivos de definición.

Una vez terminado el archivo de definiciones con el código mostrado anteriormente su visualización en el browser de AMFPHP sería como se puede observar en la siguiente imagen:

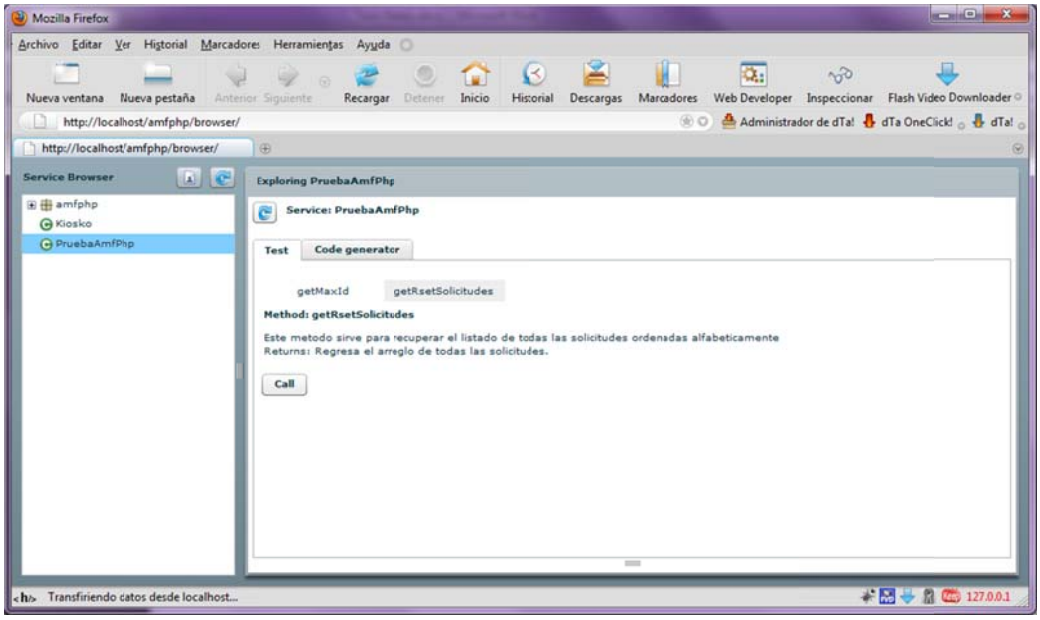


Figura 18.2.1: Explorador de Servicios Web de AMFPHP

5.3 ACCEDER A UN MÉTODO PHP DESDE ACTIONSCRIPT 3

Para poder acceder a un método PHP, éste debe haberse publicado en un archivo de definiciones, para que en ActionScript 3 esté disponible. Hecho esto ahora resta combinar el uso de las clases flash.net.NetConnection y flash.net.Responder para generar una conexión al Gateway de AMFPHP y un receptor respectivamente, toda esta puede ser fácilmente simplificada mediante el código siguiente, que implementa toda la parte de la petición y recepción de información:

```
package clases
{
	import flash.net.NetConnection;
	import flash.net.Responder;

	public class ConexionAmfphp
	{
```

```
                private var ga_resultado:Array = new Array();
                private var go_gateway:String="http://localhost/amfphp/gateway.php";
                private var go_conexion:NetConnection = new NetConnection();
                private var go_respuesta:Responder;
                public function ejecutarConsulta(
                        lv_metodo:String,
                        lf_funcion:Function,
                        lf_error:Function,...la_args):void
                {
                        var la_argumentos:Array = new Array;
                        abrirEnlace();
                        go_respuesta = new Responder(lf_funcion, lf_error);
                        la_argumentos.push(lv_metodo);
                        la_argumentos.push(go_respuesta);
                        for (var i:uint=0; i<la_args.length; i++)
                        {
                                la_argumentos.push(la_args[i]);
                        }
                        var lf_consulta:Function = go_conexion.call;
                        lf_consulta.apply(go_conexion,la_argumentos);
                }
                public function abrirEnlace():void
                {
                        go_conexion.connect(go_gateway);
                }
                public function cerrarEnlace():void
                {
                        go_conexion.close();
                }
        }
}
```

Esta clase es hasta cierto punto automática, pero hay una línea muy importante, `private var go_gateway:String="http://localhost/amfphp/gateway.php";` que indica la dirección http del Gateway de AMFPHP que deberá ser modificada según el caso.

En la clase hay 3 métodos, el método que más interesa es ejecutarConsulta, que será el que realice la petición en sí hacia el Gateway, los otros 2 métodos son abrirEnlace y cerrarEnlace, que son comunes en una conexión.

Ahora, a continuación se expone el código necesario para poder utilizar la clase ConexionAmfphp para recuperar información desde el Gateway de AMFPHP, que as su vez consulta datos a una base de datos Oracle.

```
import clases.ConexionAmfphp;
```

```
var o_con_amfphp:ConexionAmfphp = new ConexionAmfphp();

o_con_amfphp.ejecutarConsulta("PruebaAmfPhp.getRsetSolicitudes",respuestaSolicitudes,e
rrorAmfPhp);

o_con_amfphp.ejecutarConsulta("PruebaAmfPhp.getMaxId",respuestaMaximoId,errorAmfP
hp);

function respuestaSolicitudes(resultado:Array):void
{
	for (var i:int=0;i<resultado.length;i++)
	{
		trace("Código de Solicitud: "+resultado[i]["SOL_ID_SOLICITUD"]);
		trace("Descripción de Solicitud: "+resultado[i]["SOL_DESCRIPCION"]);
	}
	o_con_amfphp.ejecutarConsulta("PruebaAmfPhp.getSolicitudPorCodigo",respuesta
SolicitudCodigo,errorAmfPhp,1);
}

function respuestaSolicitudCodigo(resultado:Array):void
{
	for (var i:int=0;i<resultado.length;i++)
	{
		trace("Código de Solicitud: "+resultado[i]["SOL_ID_SOLICITUD"]);
		trace("Descripción de Solicitud: "+resultado[i]["SOL_DESCRIPCION"]);
	}
}

function respuestaMaximoId(resultado:int):void
{
	trace("El maximo codigo es: "+resultado)
}

function errorAmfPhp(error:Object):void
{
	trace("Error: " + error.description);
}
```

Solo ejecutando el código anterior se podrá notar un pequeño problema, que si se piensa mejor puede utilizarse como ventaja, consiste en el hecho de que cada llamada a un método publicado en el archivo de definiciones del Gateway de AMFPHP se la realiza de forma asíncrona, es decir puede darse un caso en el que sin que termine de captarse el resultado de una consulta el resultado de otra ya se haya captado. Si se ejecutase el código anterior, sería el caso del método respuestaMaximoId cuyo resultado llega primero que el de las otras consultas. Ahora nótese que el resultado del métodorespuestaSolicitudCodigo llega siempre e inequívocamente luego de que haya llegado el resultado de respuestaSolicitudes esto

se da porque la llamada al método respuestaSolicitudCodigo se la hace dentro del capturador de resultados del método respuestaSolicitudes. Es así como podría utilizarse esta situación como ventaja, para renderizar los objetos de flash solo si se ha cargado ya la información solicitada y evitar así una visión ralentizada de las animaciones.

CONCLUSIONES

Al culminar el presente proyecto de tesis se llego a las siguientes conclusiones:

- Se consiguió Diseñar y Desarrollar un Software Interactivo y Práctico en el que se pueda integrar información de estudiantes, y que permita a los mismos realizar procesos en un período de tiempo más corto que en el que usualmente lo realizan.

- Luego de un diálogo con los responsables de Secretaría General de la Universidad Politécnica Salesiana, se determinó que para muchos de los Certificados Académicos su automatización aun no es posible, debido a que los datos de algunos Periodos Lectivos no se han ingresado en el Sistema Nacional Académico todavía, por ello su generación requiere obligatoriamente intervención por parte de las secretarias y por ende deberán continuar por ahora generándose de forma manual luego de analizar la información disponible en papel.

- Se llegó a la resolución de implementar la automatización de los certificados:

    - Certificado de Asistencia a Clases
    - Certificado de Matrícula
    - Certificado de Inscripción

  Dado que para la generación de dichos certificados existe el 100% de la información necesaria.

- Dada la reducción del alcance originalmente planteado causada por la imposibilidad de generar la totalidad de Certificados Académicos, se llegó a un acuerdo de compensar el cambio con la “Implementación de un Software de Generación de Solicitudes para los Estudiantes”, el mismo que funciona en conjunto con el Módulo de Certificados. En el Módulo de solicitudes, se da la

posibilidad de generar cualquier tipo de Solicitud Académica. Permite también agregar, eliminar o modificar solicitudes.

- En la actualidad los Kioskos Multimedia son una herramienta de apoyo para los usuarios en distintas áreas debido a que gracias a una interfaz amigable y de fácil de usar han facilitado el desarrollo de procesos en tiempo reducido.

  Una aplicación de esto es evidente por ejemplo en la Escuela Politécnica del Litoral, cuyo nombre es "Polimático" y que ha simplificado la gestión de los trámites académicos de los estudiantes. Además el uso de herramientas de este tipo mejora la imagen de las Instituciones que la implementan.

- Se consiguió una familiarización con las aplicaciones necesarias para crear un Software Dinámico, Amigable, y Agradable a la vista. Se decidió utilizar Adobe Flash, principalmente por su capacidad de manejo vectorial, importante al momento de cambios de resolución, Para el diseño e implementación del ambiente de administración web, se utilizó Adobe Dreamweaver, por su facilidad de uso, y su posible complementación con otras herramientas.

RECOMENDACIONES

- Para la realización de proyectos de tesis similares a la presente es recomendable que en lo posible se solicite asesoría de personas especializadas en la rama de Diseño Web y Animación.

- Antes de empezar con el Diseño e Implementación analizar cuáles son los programas más adecuados y de fácil manejo.

- Disponer de "La Internet" como fuente de información en todo momento, razón por la cual en el caso de Tesis similares a ésta se recomienda las direcciones citadas en la sección "Bibliografía".

- En el caso la creación y edición de la Interfaz Web, es recomendable utilizar Adobe DreamWeaver como editor HTML, dado que gracias a sus constantes actualizaciones, se mantiene a la par de la tecnología en cuanto a manejo de estándares HTML, CSS, etc.

- En cuanto al diseño de animaciones, es muy conveniente que no se utilicen Imágenes en formato Bitmap, y se haga uso al contrario de únicamente Imágenes Vectorizadas, dado que al producirse un cambio de resolución no se pierde nada de calidad.

- En caso de utilizarse Flash para animaciones, pensar también en el lenguaje que lo complementará para el acceso a las bases de datos requeridas.

- Es muy conveniente el uso de Servicios Web, dado que Flash carece de capacidades de accesos a Bases de Datos por esta razón habrá de utilizarse un lenguaje de programación alterno, y para mantener la interoperatibilidad entre los mismos sería una solución muy apreciable la traducción de las comunicaciones mediante servicios web.

BIBLIOGRAFÍA:

Libros:

- Uso de ADOBE® DREAMWEAVER® CS4, escrito por Adobe Corporation. Publicado en el Año 2008

- ADOBE® FLASH® CS4, escrito por Adobe Corporation. Publicado en el Año 2008

- Programación con ADOBE ® ACTIONSCRIPT® 3.0, escrito por Adobe Corporation. Publicado en el Año 2008

- Web Database Applications with PHP, escrito por Hugh E. Williams, publicado por O'Reilly en el 2008

Referencias Electrónicas:

- http://www.planet-source-code.com/URLSEO/vb/scripts/ShowCode!asp/txtCodeId!1851/lngWid!8/anyname.htm
- API de Action Script, disponible en: http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3
- http://www.kirupa.com/forum/showthread.php?p=1923917
- http://amfphp.sourceforge.net/docs/
- http://www.cristalab.com/tips/lenguajes-funcionales-y-la-clase-array-de-actionscript-3-c45347l/
- http://www.actionscript.org/forums/showthread.php3?t=158039
- http://www.robvanderwoude.com/printfiles.php
- http://www.programacionweb.net/cursos/curso.php?num=10
- http://www.conceptopixel.com/aprende-a-usar-adobe-flash-como-en-el-kinder

- http://www.webestilo.com/flash
- http://www.webestilo.com/dreamweaver
- http://banners.galiciacity.net/html/conceptos.html
- http://www.carsoft.com.ar/pro1.htm
- http://cdec.unican.es/libro/elementos_avanzados.htm
- http://cdec.unican.es/libro/Estructura_HTML.htm
- http://cdec.unican.es/libro/estructura_texto.htm
- http://cdec.unican.es/libro/nociones_basicas_de_html.htm
- http://www.ceniainternet.cu/servicios/web.html
- http://www.fortunecity.es/imaginapoder/nada/617/PT111.htm
- http://www.infopeople.com/aaii/diseno/index.html
- http://www.learnthenet.com/spanish/html/14wbpganat.htm
- http://www.redestudiantilpr.net/construirweb.htm
- http://www.supaginaweb.com/ques2.htm
- http://www.uam.es/departamentos/medicina/anesnet/redeshtml/estructura%20protocolotexto.htm
- http://www.uned.es/psico-doctorado-interuniversitario/Web/estructuraweb.htm
- http://www.unitec.edu.co/biblioteca/internet
- http://www.webaprendiz.com/estudio/manual/html/estructura/estruct1.htm
- http://webpersonal.uma.es/de/MGJ/estructura.htm
- http://www.xfere.net/cursosonline/protocolos.html
- http://www.cybercursos.net/cursos-online/protocolos.htm
- http://dc.inictel.gob.pe/telecom/Cursos_Actualizacion/act_curso_16.htm
- http://www.forest.ula.ve/~mana/cursos/redes/protocolos.html
- http://personales.com/mexico/mexico/Redes/page5.html
- http://pyspanishdoc.sourceforge.net/lib/internet.html
- http://www.rcp.net.pe/rcp/internet/protocolos1.shtml
- http://structio.sourceforge.net/guias/AA_Linux_colegio/redes-protocolos-e-internet.html

ANEXOS

UNIVERSIDAD POLITÉCNICA SALESIANA

Esta encuesta tiene el propósito de conocer la frecuencia con la que un estudiante realiza trámites como solicitudes o certificados en la Universidad Politécnica Salesiana.

Nombre: _____ fecha:_____

Carrera: _____

1. **Cuantos años estudia en la UPS**

   _____

2. **¿Cuántas veces ha solicitado un certificado?**

   _____

3. **¿Cuántas veces ha realizado una solicitud formal a una autoridad de la UPS?**

   _____

4. **¿Qué tarea le resulta más complicada tediosa: realizar una solicitud o pedir un certificado? ¿Por qué?**

   _____

   _____

   _____

5. **Le gustaría que se automatice la terea de:**

   Solicitar certificados: _____

   Realizar Solicitudes: _____

ANEXOS

Los resultados obtenidos de las encuestas aplicadas a 88 estudiantes de las carreras de Ingeniería de Sistemas, Ingeniería Electrónica, e Ingeniería Industrial de primero a tercer año son los siguientes:

	Por Certificados	Por Solicitudes	En Blanco	Ninguna	Ambos
Pregunta 2	183	0	0	0	
Pregunta 3	0	270	0	0	
Pregunta 4	7	67	1	13	
Pregunta 5	85	59	0	1	53

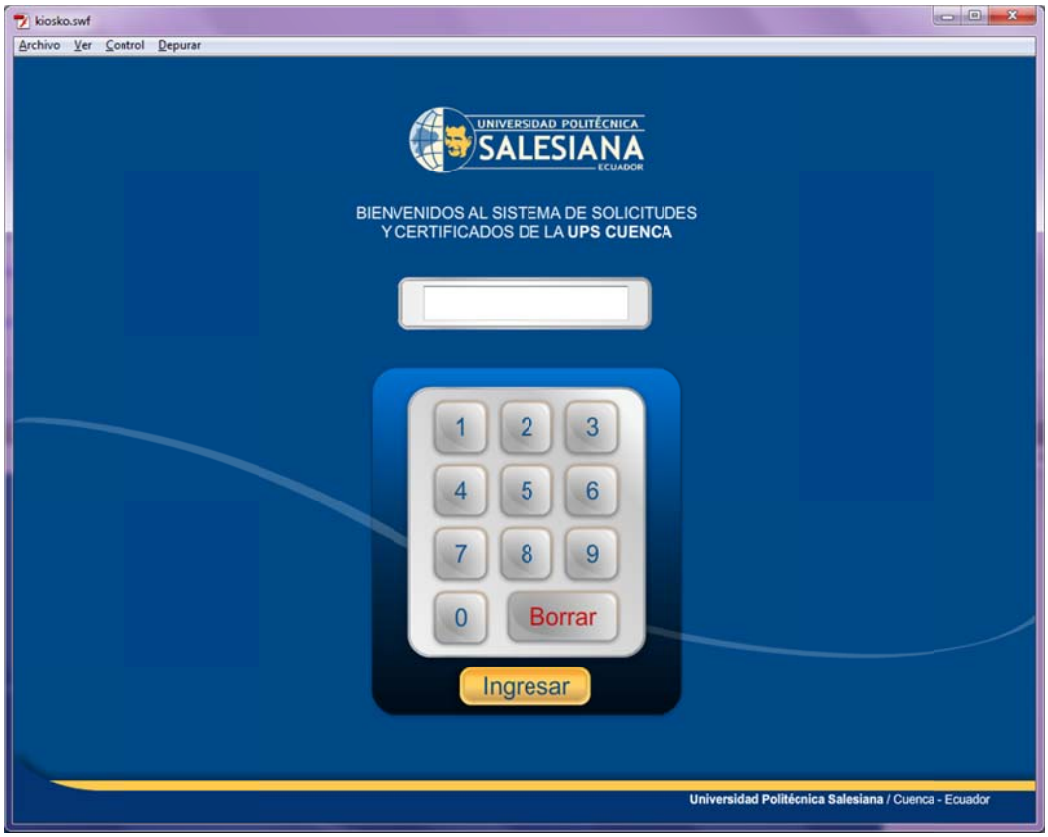
ANEXOS

MANUAL DE USUARIO

"KIOSKO MULTIMEDIA PARA EMISIÓN DE CERTIFICADOS Y SOLICITUDES ACADÉMICAS"

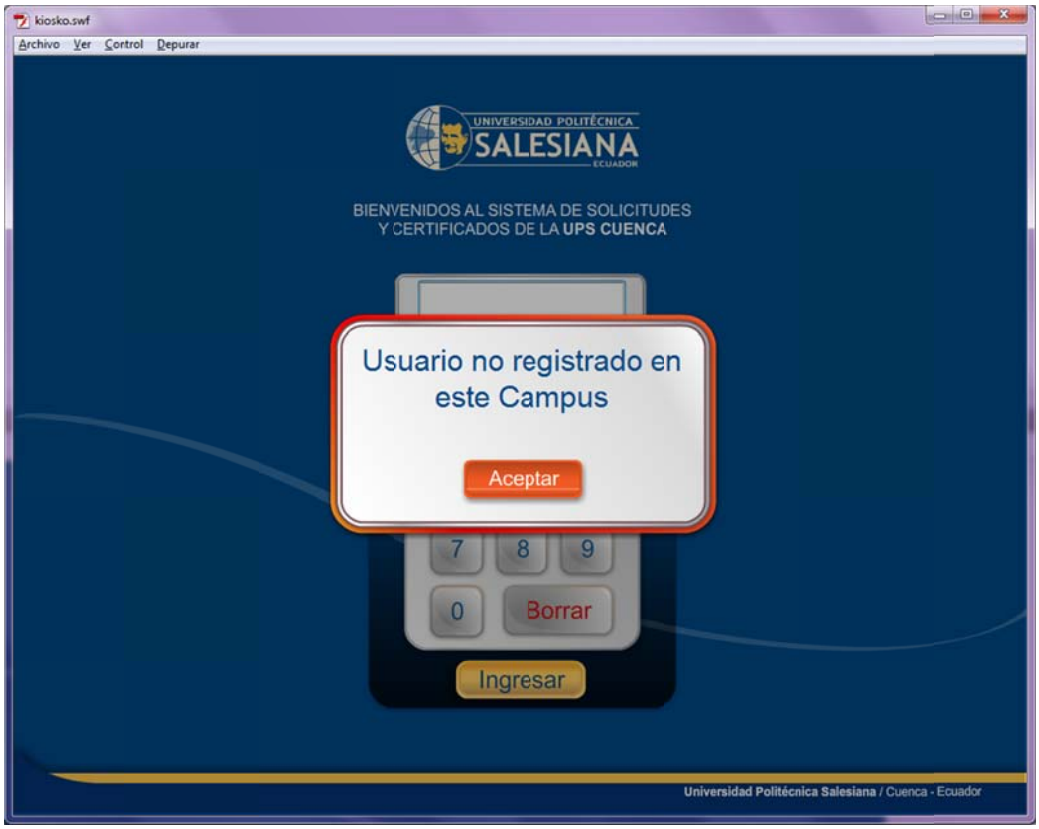
Esta aplicación permite a los estudiantes de la Universidad Politécnica Salesiana solicitar CERTIFICADOS y SOLICITUDES académicas, a continuación se describen cada una de las pantallas con las que los usuarios se encontrarán dentro de la aplicación según las elecciones que realice.

Al inicio de la aplicación el usuario se encuentra con una pantalla introductoria en la que se le solicita digitar el número del identificador: cédula o pasaporte que lo identifique dentro de la institución.



En esta pantalla se visualiza únicamente un teclado numérico digital, que permite al usuario introducir su número de identificador, si existe un error de digitación, el usuario puede recurrir al botón BORRAR, para corregir el error, caso contrario, procede a pulsar el botón INGRESAR, para continuar con el proceso.

Si el número de identificador no es correcto se genera un error y se visualiza el siguiente mensaje:



En este caso, el usuario debe pulsar el botón aceptar y digitar nuevamente el númerode identificador.

Cuando el identificador es correcto se habilita la siguiente pantalla en la que se generan dos menús:

SELECCIONE SU CARRERA, que permite escoger al usuario la carrera desde la cual desea realizar las acciones dentro de la aplicación; cabe recalcar que las carreras

disponibles para el usuario en este menú corresponden a las carreras en las que el estudiante se ha inscrito en cualquier periodo dentro de la universidad;

SELECCIONE LA CARRERA DESTINATARIA, que habilita todas las carreras con las que cuenta la universidad permitiendo al usuario escoger hacia cual quiere realizar la acción, por defecto se habilita la carrera que escogió el usuario en el primer menú, pero esta puede cambiarse, recorriendo las opciones con las flechas que se encuentran a la derecha del menú.

Existen dos tipos de flechas; las que se encuentran a los extremos superior e inferior, permiten al usuario dirigirse directamente al principio y al final de la lista respectiva, mientras que las flechas que se encuentran en el centro, permiten al usuario recorrer un puesto a la vez, dentro de la lista correspondiente.

Existe la opción atrás, que regresa a la pantalla inicial, en caso de que se desee salir de la aplicación.

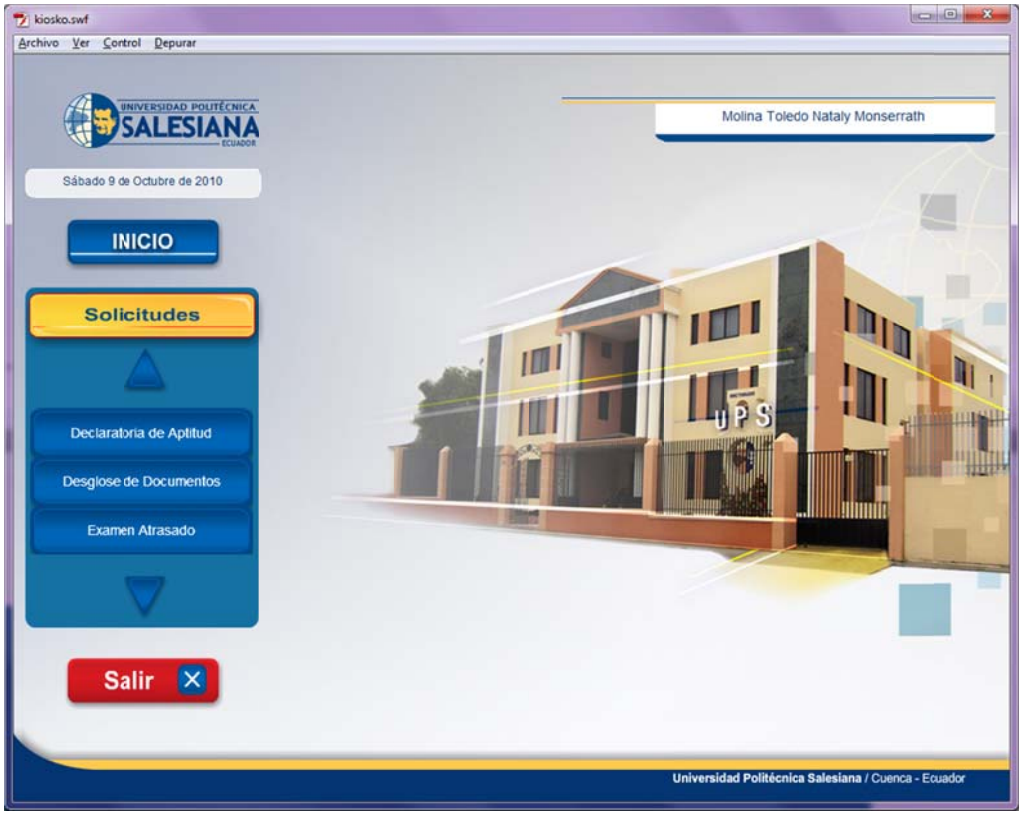
Luego de seleccionar las opciones de carrera origen y carrera destinataria, el usuario debe pulsar SIGUIENTE para continuar con el proceso y a continuación se habilitará la siguiente pantalla que es la que muestra el menú con las acciones que se pueden realizar dentro de la aplicación.

Esta pantalla contiene las opciones de la aplicación: SOLICITUDES y CERTIFICADOS y según sean los requerimientos del usuario, éste escogerá una de las dos.

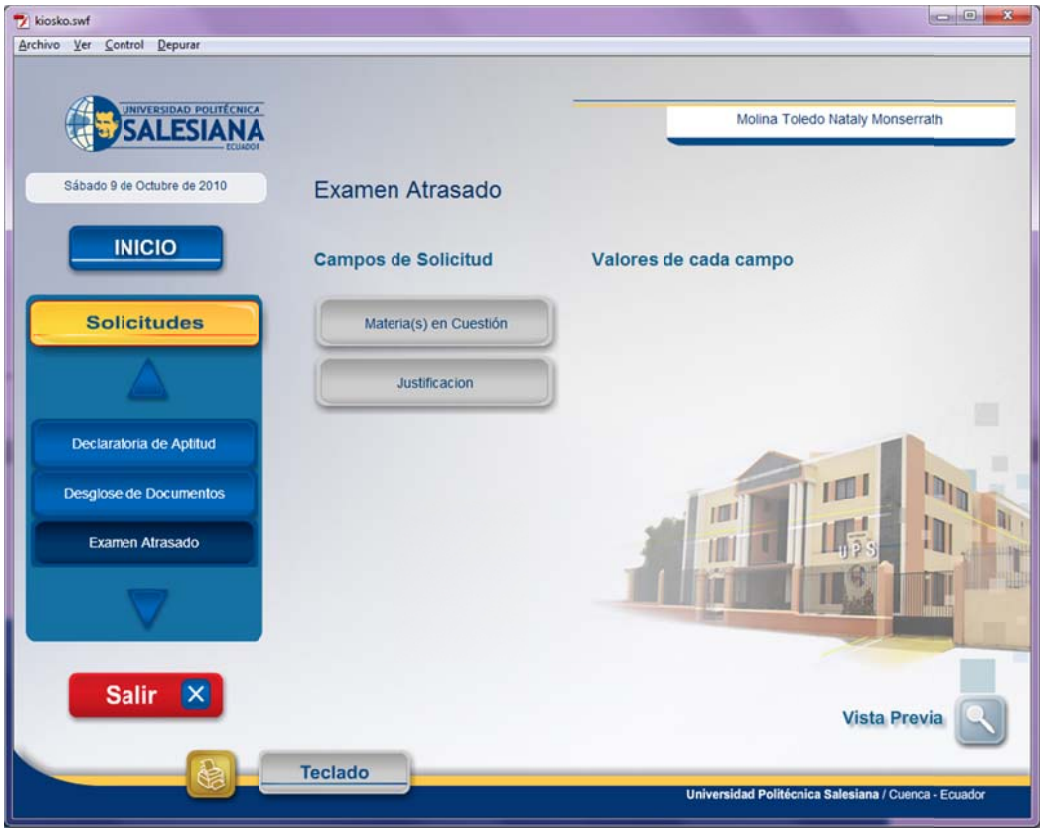
El usuario puede también salir de la aplicación pulsando el botón SALIR, o puede cambiar la carrera origen o destino con la ayuda del botón ATRÁS, mismo que lo llevará a la pantalla anterior para cambiar las opciones antes establecidas.

MENÚ SOLICITUDES:

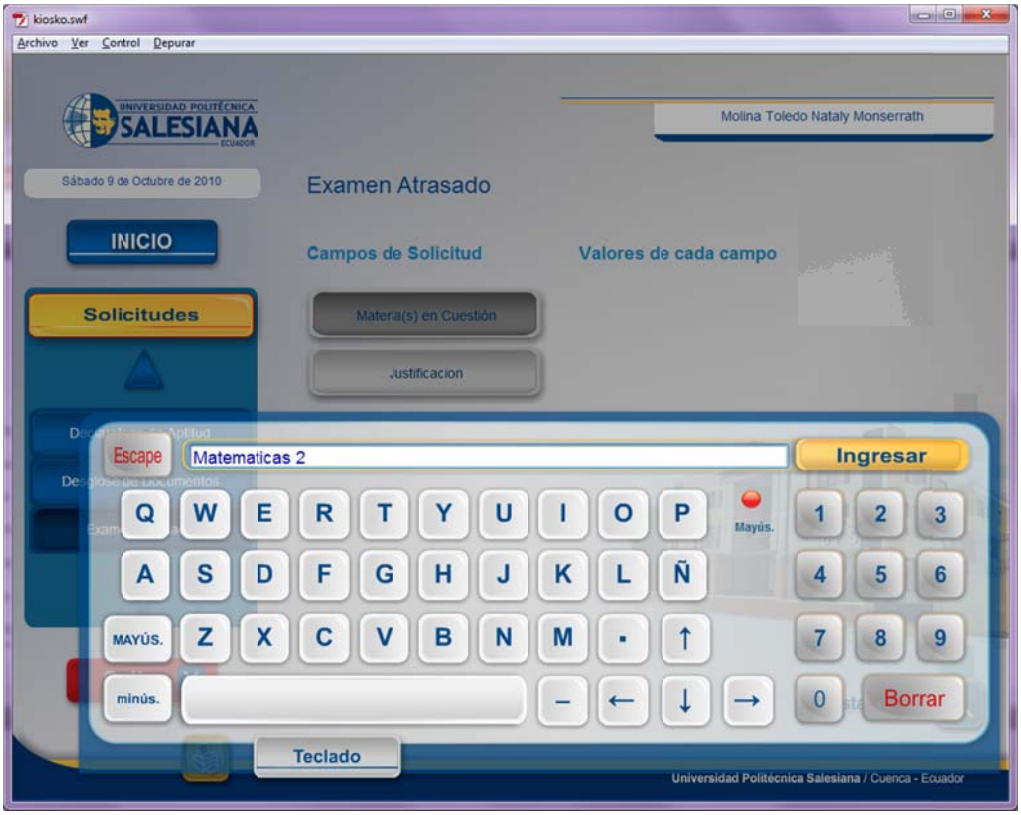
En el caso de que el usuario escoja la primera opción: SOLICITUDES, se muestra la siguiente pantalla:



En esta pantalla se carga un menú con las solicitudes disponibles, y el usuario procede a escoger la que requiera.



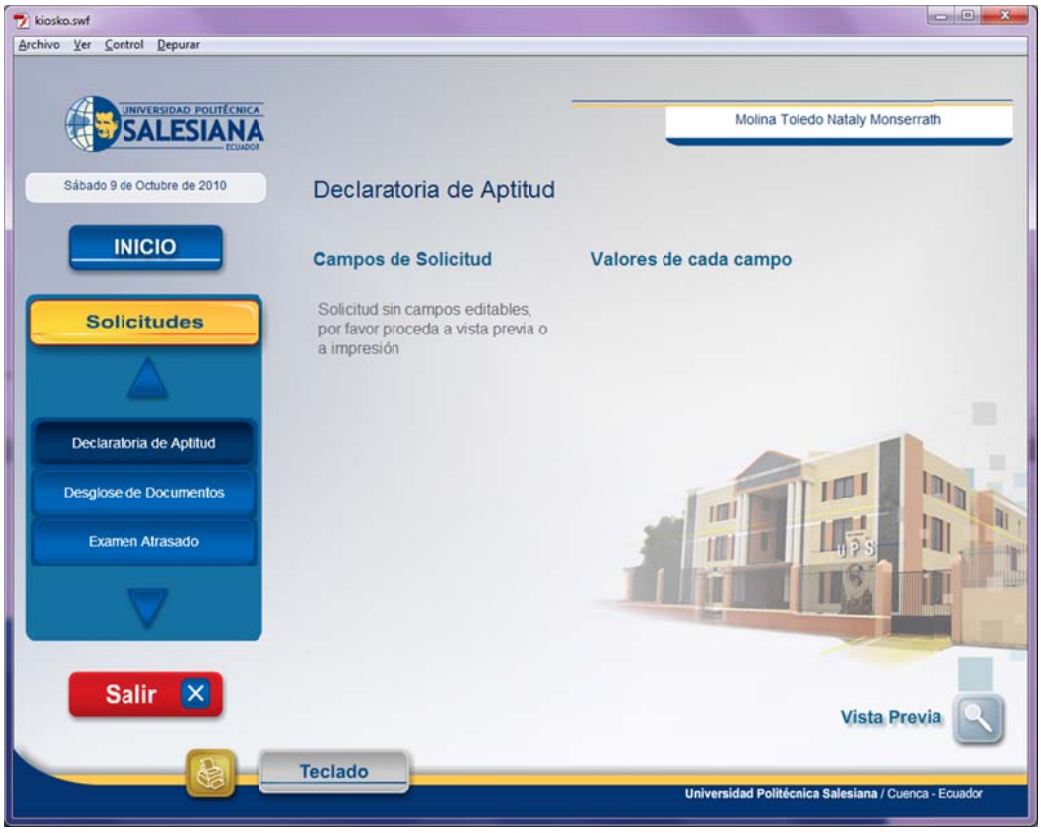
Según el tipo de solicitud es necesaria la introducción de ciertos datos por parte del usuario y para ello se cuenta con un teclado digital que aparecerá al momento de pulsar los campos editables de la solicitud:



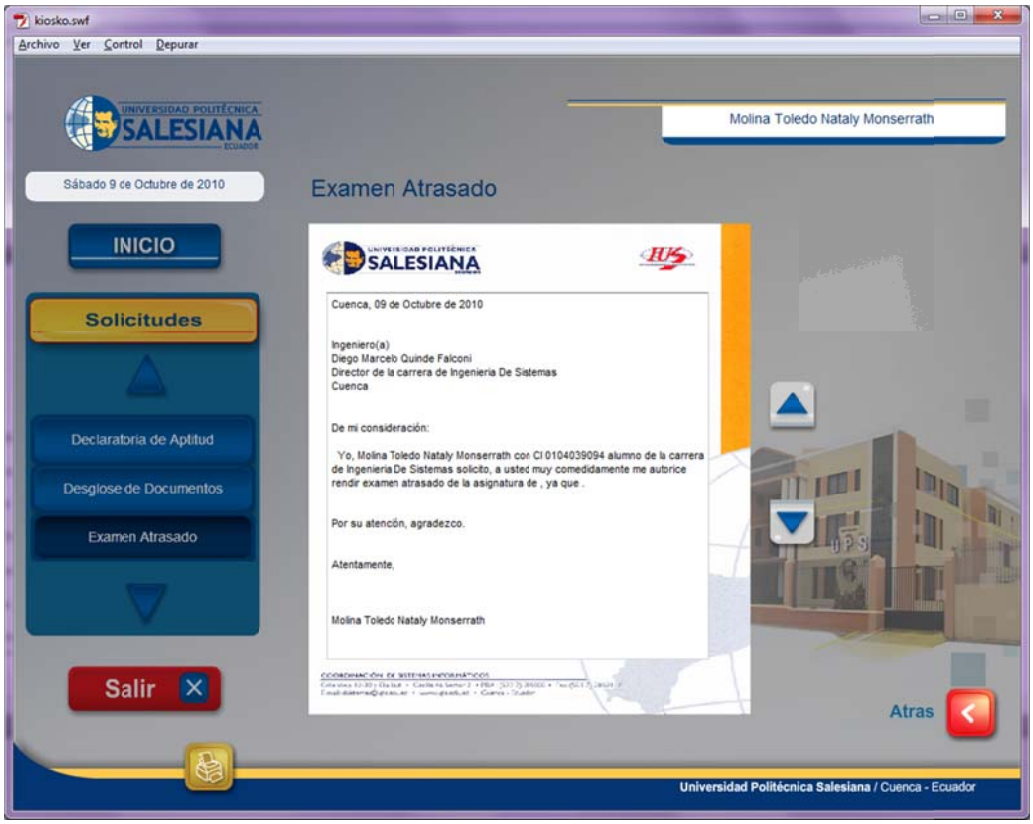
El teclado digital cuenta con todas las opciones de un teclado normal, está formado por letras y números; para cambiar de mayúsculas a minúsculas existe un botón independiente para cada una de estas acciones, y para verificar el estado actual hay un foco que se encuentra de color rojo cuando está en minúsculas y en verde cuando el estado está en mayúsculas. Existe la opción ESCAPE para salir del teclado, BORRAR para corregir errores de digitación, FLECHAS en todas las direcciones para llegar mucho más fácil a la zona del error.

Luego de escribir los datos para la solicitud, el usuario pulsa INGRESAR para registrarlos en la solicitud.

Pueden existir casos en los que la solicitud no requiera de la introducción de datos y en ese caso aparece el siguiente mensaje:



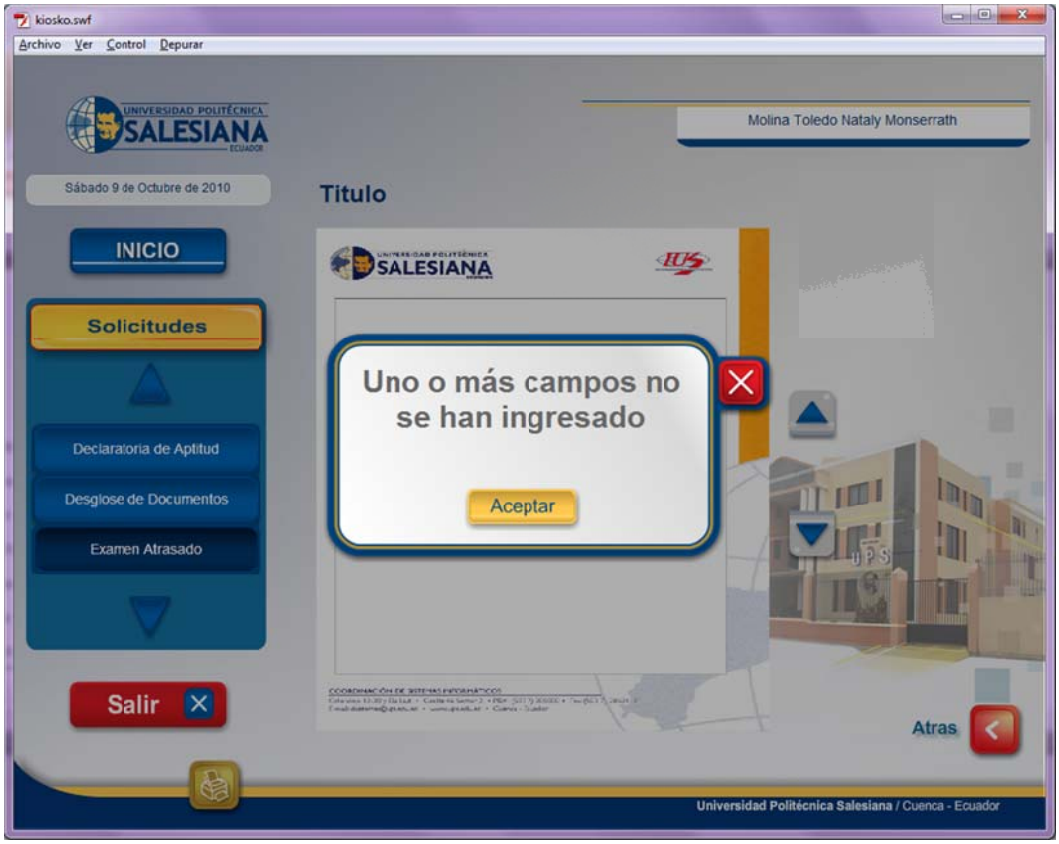
Luego de que se hayan introducido los datos necesarios, el usuario tiene la opción de revisar el formato de la solicitud con la opción VISTA PREVIA, situada en la parte inferior derecha de la ventana.



Si existe algún error en los datos introducidos, el usuario tiene habilitada la opción ATRÁS, que le permite regresar a la pantalla anterior y editar nuevamente los campos.

La opción imprimir está habilitada tanto en la vista previa como en la pantalla anterior a esta, es decir la pantalla de ingreso de datos en la solicitud.

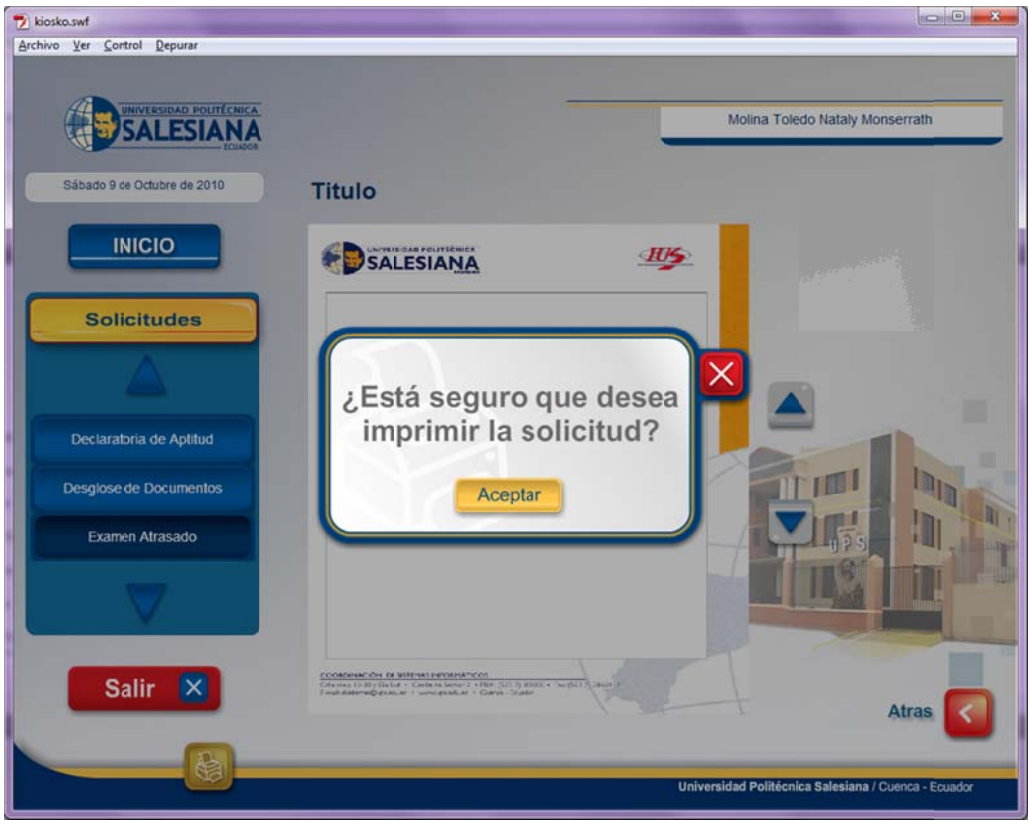
Cuando el usuario lo desee puede pulsar el botón IMPRIMIR, pero si el usuario no se percató que no se ha llenado un dato, al momento de imprimir, aparecerá el siguiente mensaje:



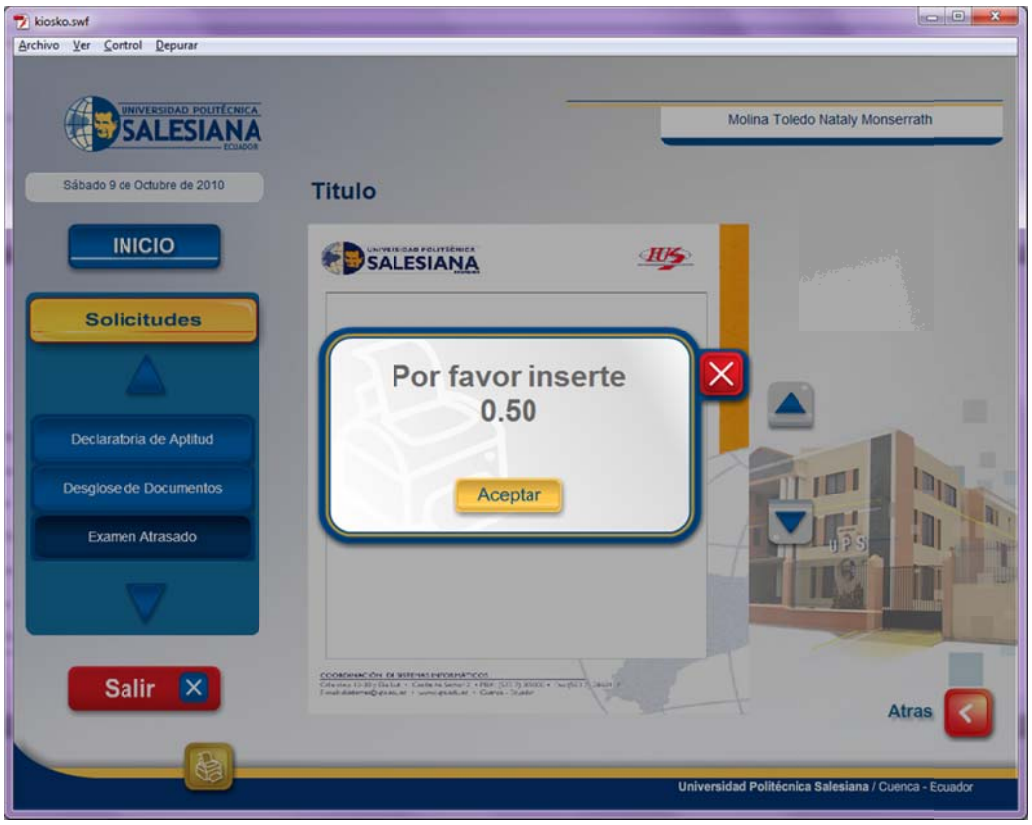
En este caso, el usuario debe pulsar ACEPTAR y proceder a llenar los datos incompletos.

Y a continuación proceder a pulsar nuevamente vista previa y verificar si el formato es correcto, y luego proceder a imprimir la solicitud.

Al pulsar el botón imprimir, por seguridad se le presenta al usuario el siguiente mensaje:

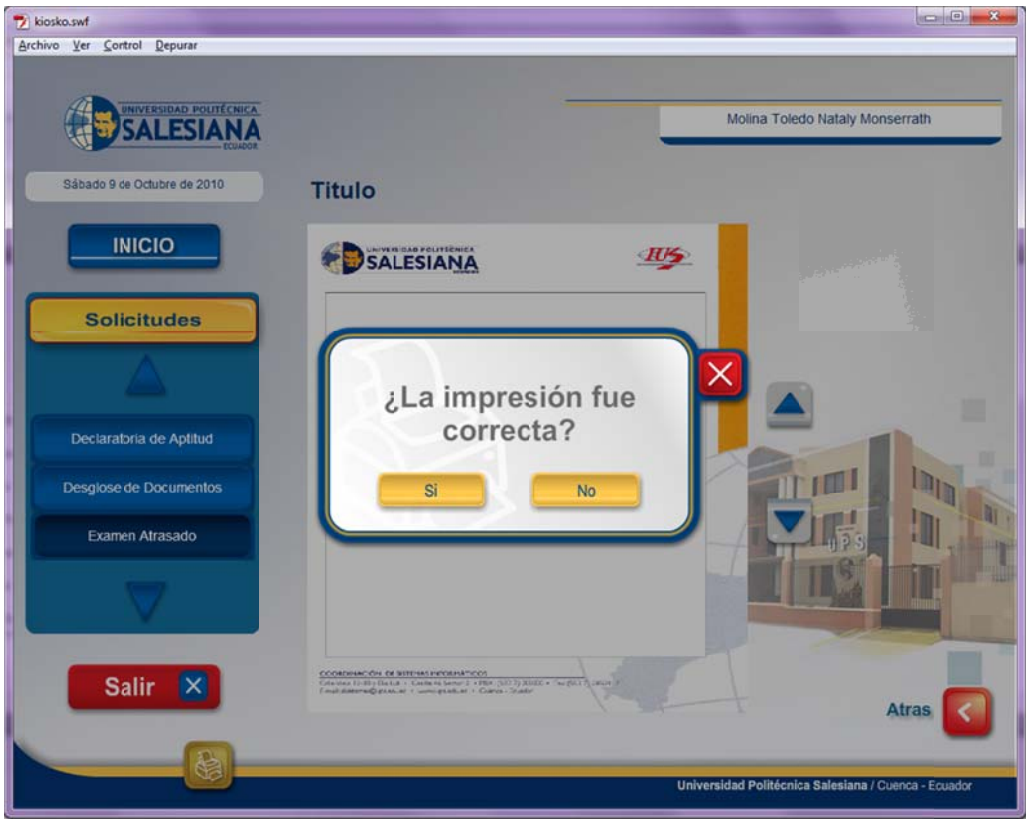


Si no fue un error de digitación, el usuario pulsa SI, y enseguida se le presenta el siguiente mensaje:

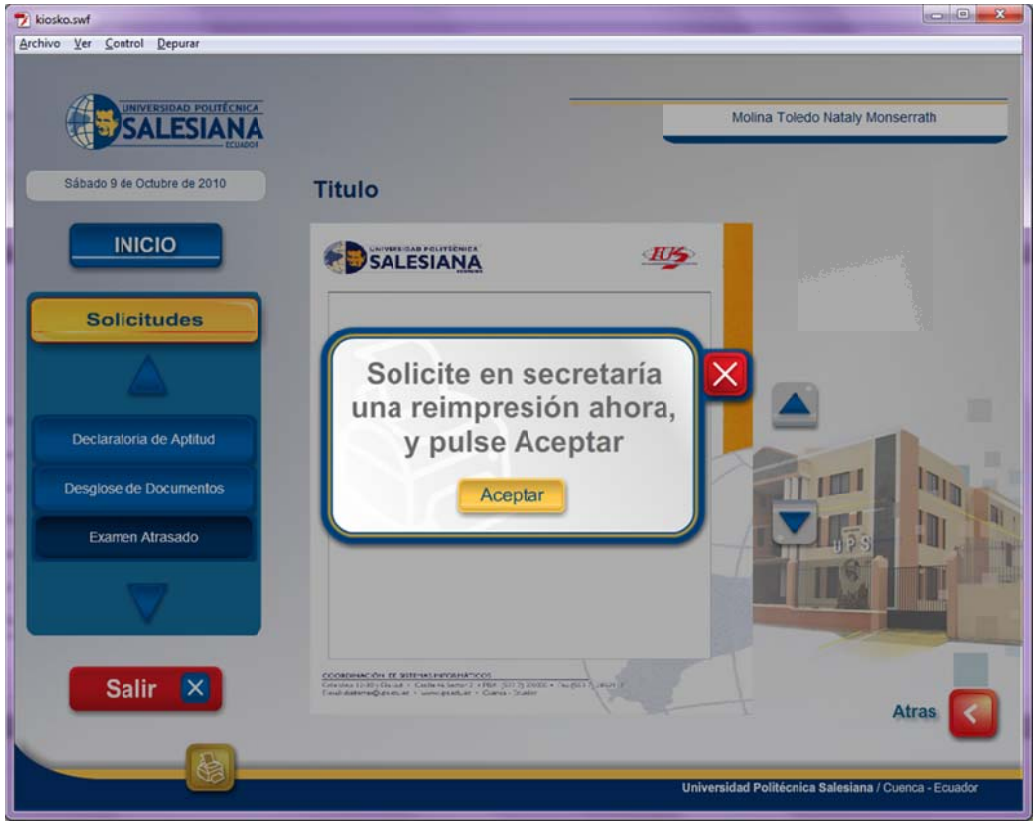


En esta instancia, el usuario debe depositar en el monedero, la tarifa asignada, caso contrario, la solicitud no se imprimirá.

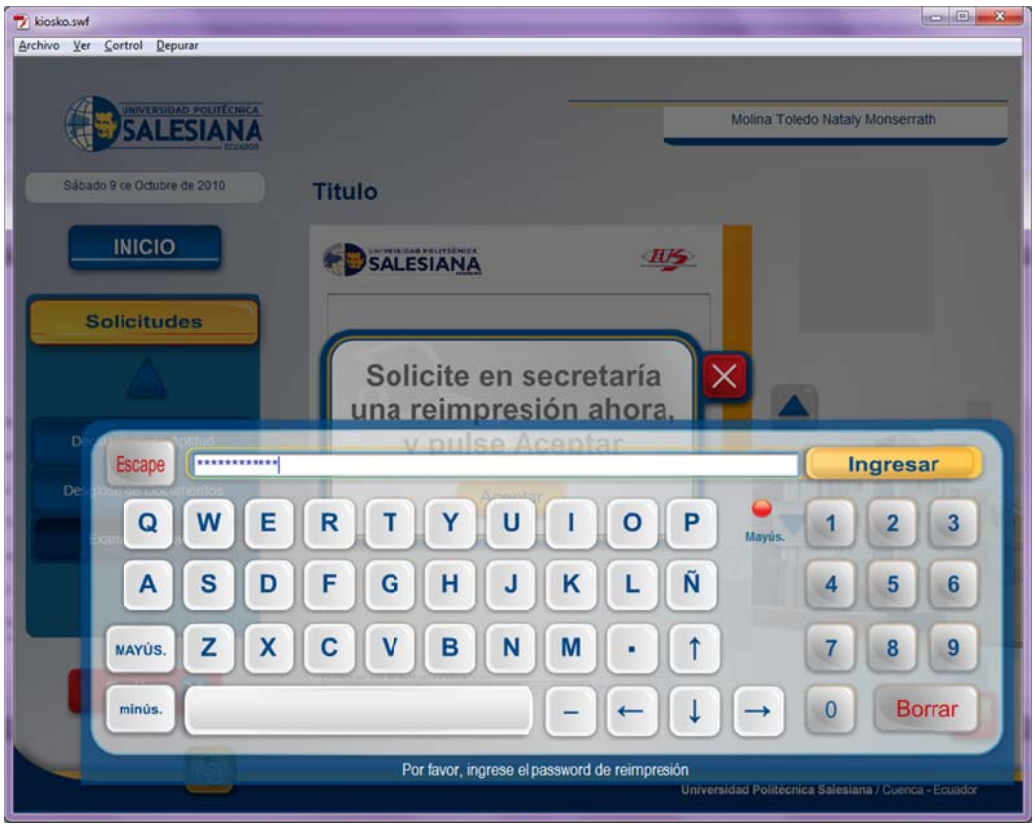
Luego de esto, si la tarifa introducida es la correcta, se presenta al usuario en siguiente mensaje:



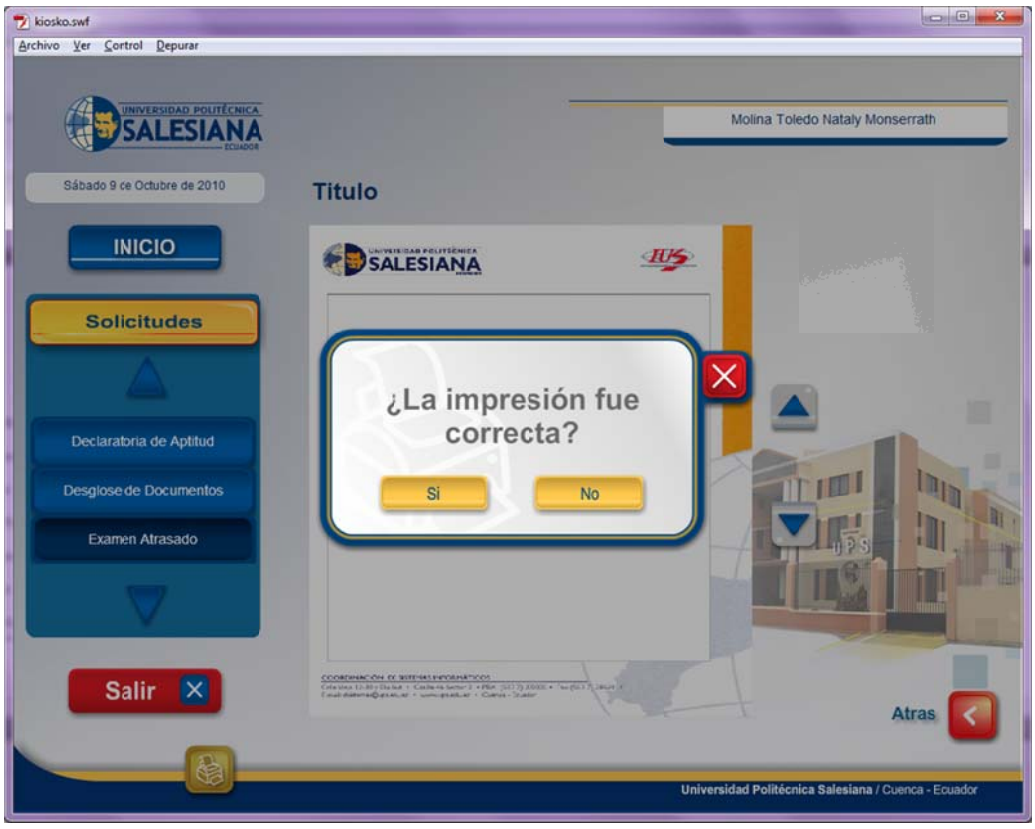
Si no existe ningún error de impresión el usuario pulsa SI y se retorna al menú de solicitudes para el caso en el que el usuario desee realizar otra solicitud pero si existió algún error en la impresión, el usuario debe pulsar NO, y en este caso, la acción es: comunicar a la secretaria del problema, e inmediatamente ella deberá ingresar un password de reimpresión.


kiosko.swf
Archivo Ver Control Depurar
UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
Molina Toledo Nataly Monserrath
Sábado 9 de Octubre de 2010
Titulo
INICIO
Solicitudes
Declaratoria de Aptitud
Desglose de Documentos
Examen Atrasado
Salir
Solicite en secretaría una reimpresión ahora, y pulse Aceptar
Aceptar
Atras
Universidad Politécnica Salesiana / Cuenca - Ecuador

Luego de pulsar ACEPTAR, aparece la siguiente pantalla, que permitirá a la secretaria introducir el código asignado:

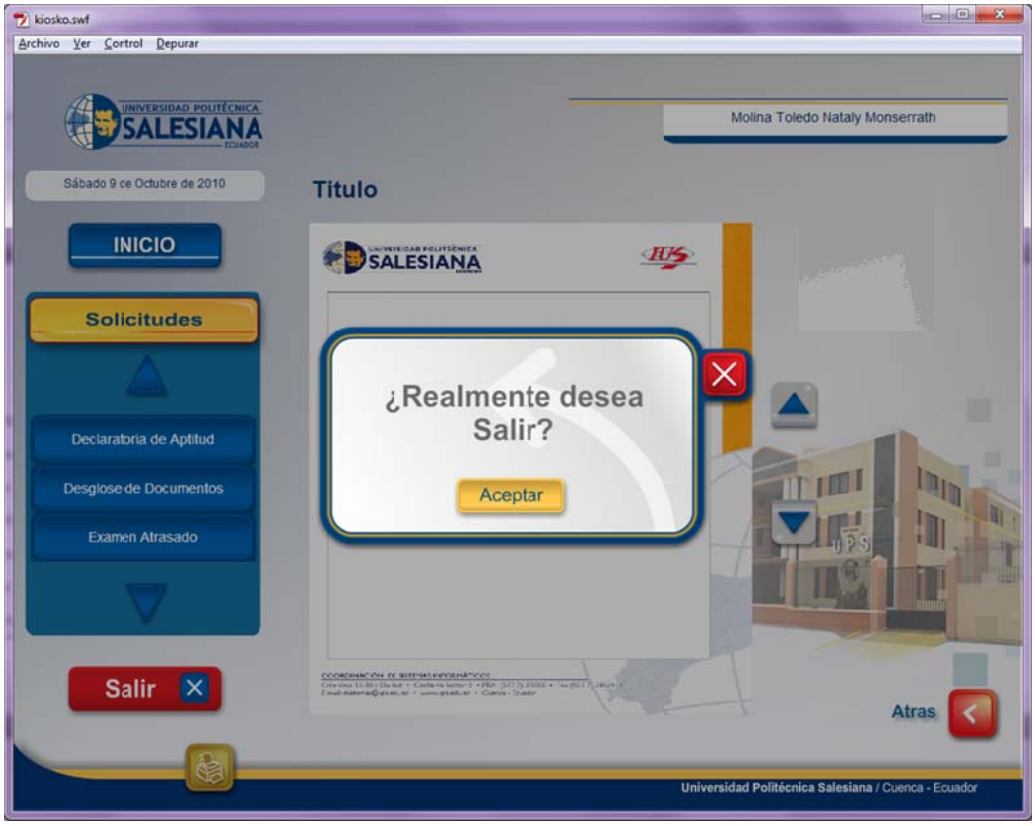


Cuando la secretaria ingresa el password escoge ingresar, y la impresión se ejecuta nuevamente, y aparece el mensaje de confirmación de impresión correcta:



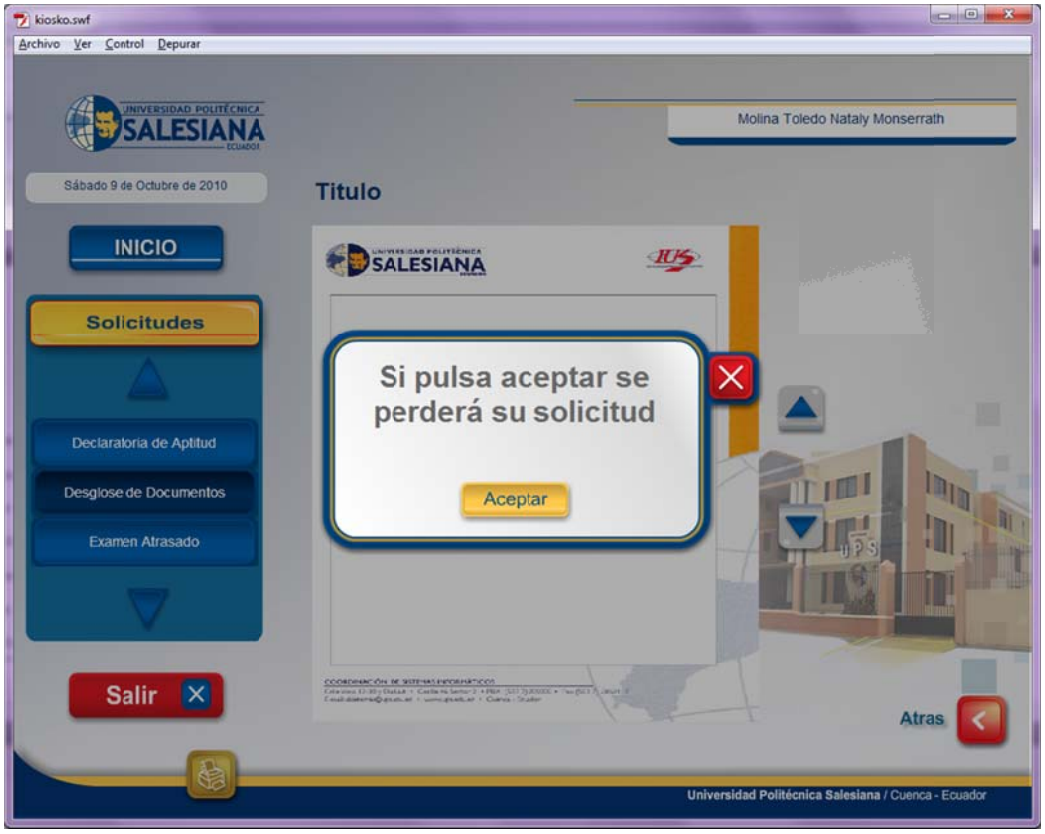
Y si la respuesta es SI, se regresa al menú de solicitudes y caso contrario se repite a la acción descrita anteriormente.

Debido a que se trata de un touch screen, el usuario está más propenso a digitar sin querer ciertas opciones dentro del proceso, debido a ello, esta aplicación cuenta con la respectiva seguridad y si por algún error de digitación se pulsó el botón SALIR, se pregunta la confirmación de la acción a través del siguiente mensaje:



Permitiendo al usuario rectificar la acción.

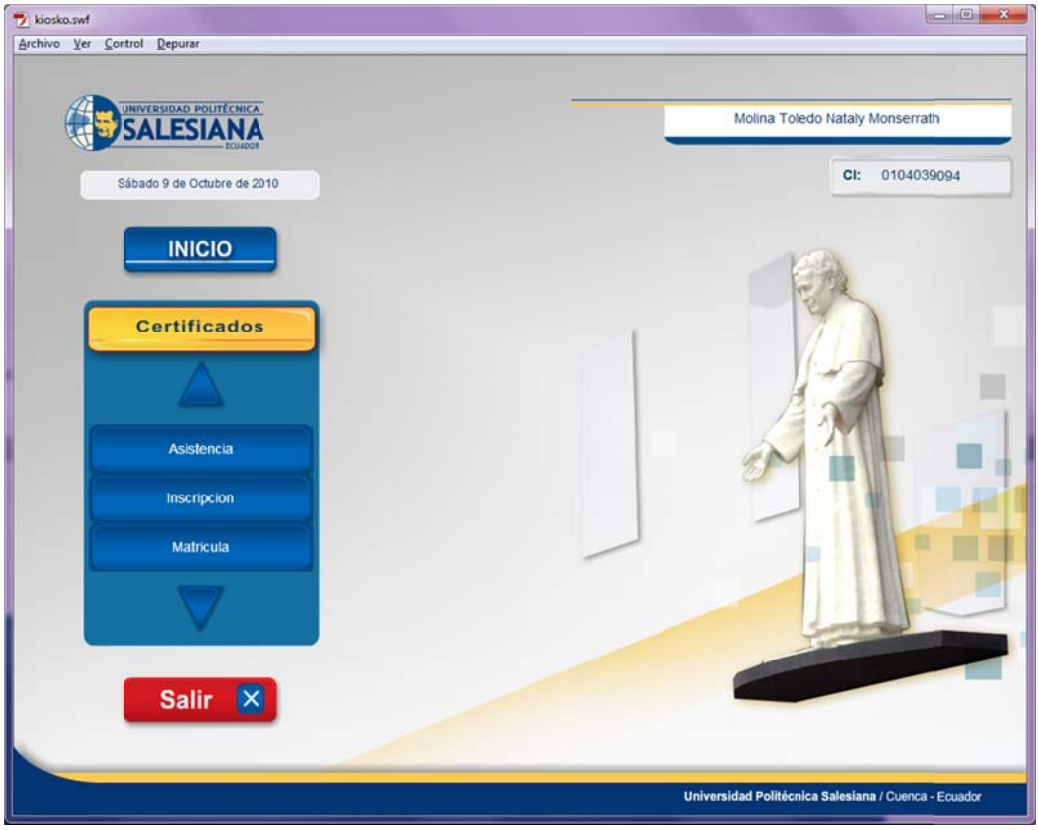
Otro caso dentro de los problemas de digitación puede surgir al momento de seleccionar una solicitud y sin querer el usuario puede cambiar el tipo de solicitud, en este caso, el mensaje será:



Cabe mencionar que al igual que en las otras pantallas, el usuario tiene la posibilidad de regresar al inicio y escoger nuevamente entre SOLICITUDES y CERTIFICADOS, con el botón INICIO, o puede salir de la aplicación pulsando el botón SALIR.

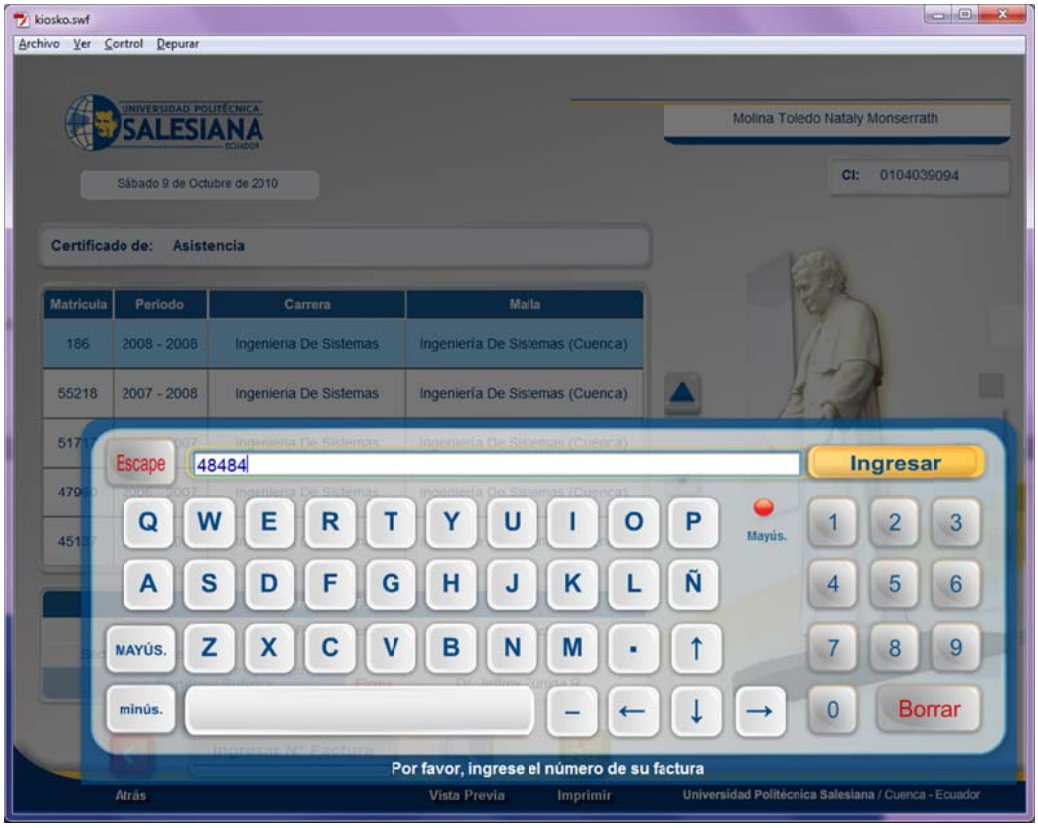
MENÚ CERTIFICADO:

En el caso de que el usuario escoja la segunda opción: CERTIFICADOS, se muestra la siguiente pantalla:

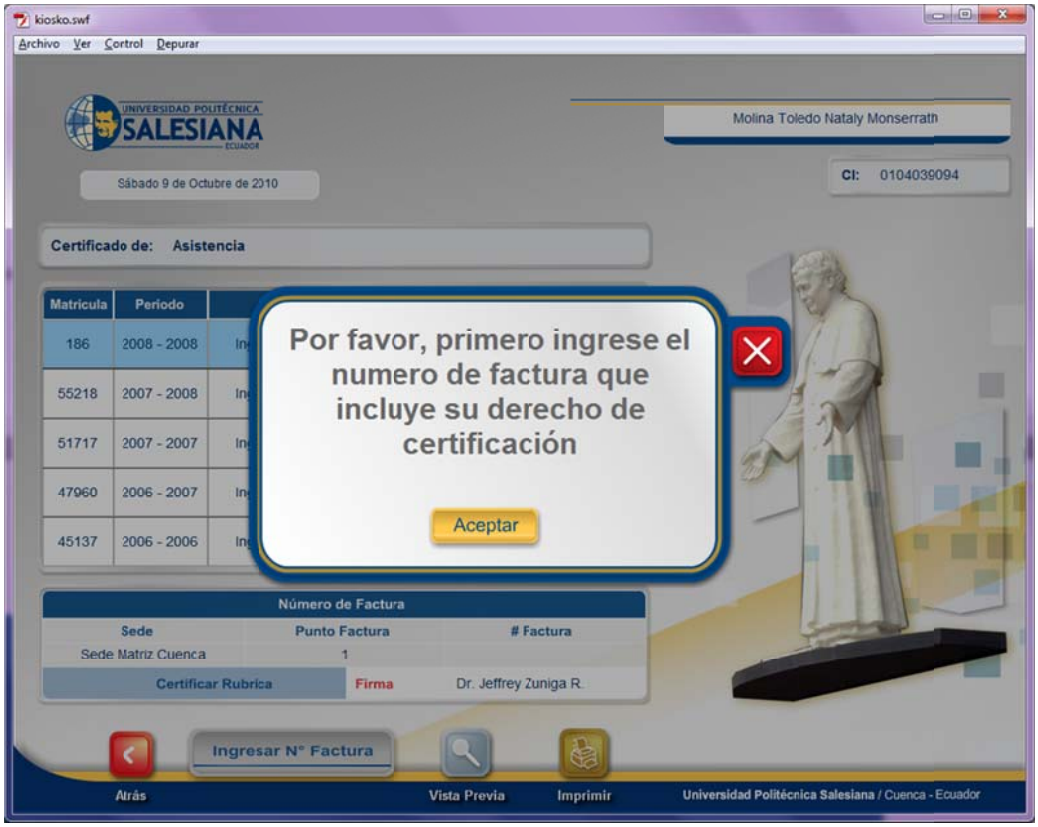


En esta pantalla se presenta al usuario un menú con los certificados disponibles, y según sean sus requerimientos, escoge el adecuado, y se le presenta una pantalla como la siguiente:

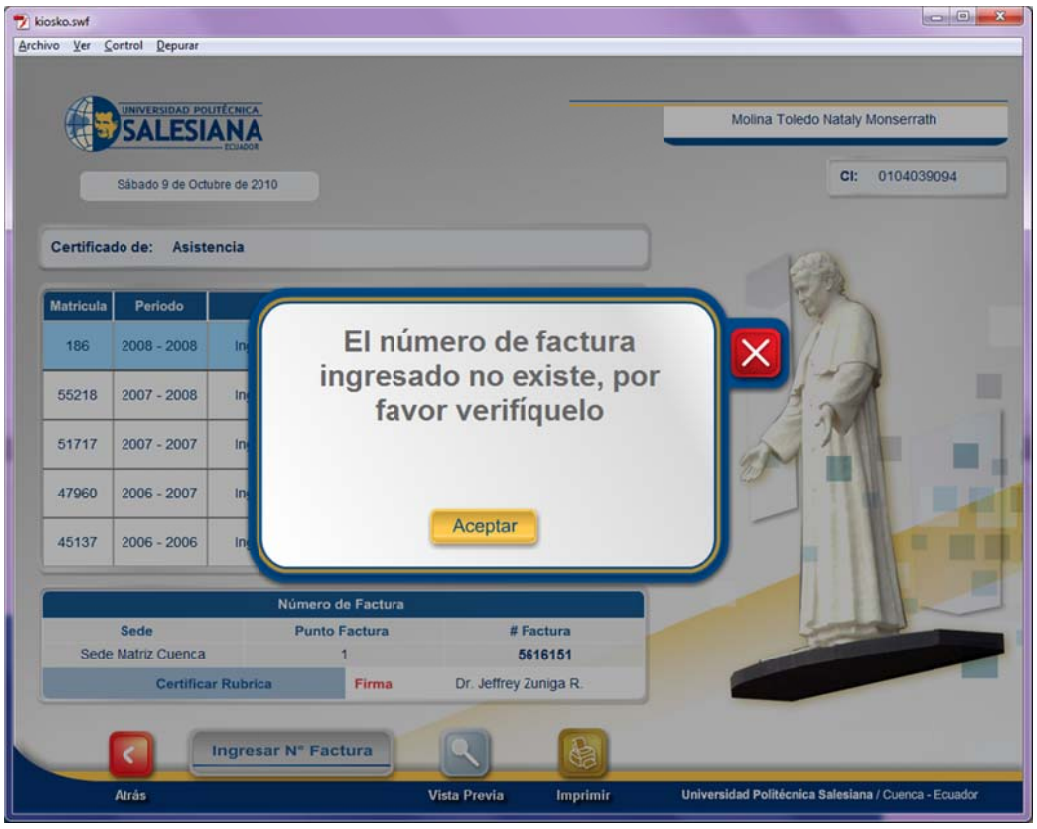
Este menú, carga todos los períodos en los que se ha matriculado el alumno, y según sus requerimientos, debe escoger el adecuado. Luego de esto el usuario debe pulsar el botón INGRESAR EL Nº FACTURA y de inmediato se habilita el teclado y el usuario debe proceder a introducirlo:


kiosko.swf
Archivo Ver Control Depurar
UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
Molina Toledo Nataly Monserrath
Sábado 9 de Octubre de 2010
CI: 0104039094
Certificado de: Asistencia
Matricula
Periodo
Carrera
Malla
186
2008 - 2008
Ingenieria De Sistemas
Ingeniería De Sistemas (Cuenca)
55218
2007 - 2008
Ingenieria De Sistemas
Ingeniería De Sistemas (Cuenca)
Escape
48484
Ingresar
Q W E R T Y U I O P
A S D F G H J K L Ñ
MAYÚS. Z X C V B N M . ↑
minús.
Mayús.
1 2 3
4 5 6
7 8 9
0 Borrar
Por favor, ingrese el número de su factura
Atrás
Vista Previa
Imprimir
Universidad Politécnica Salesiana / Cuenca - Ecuador

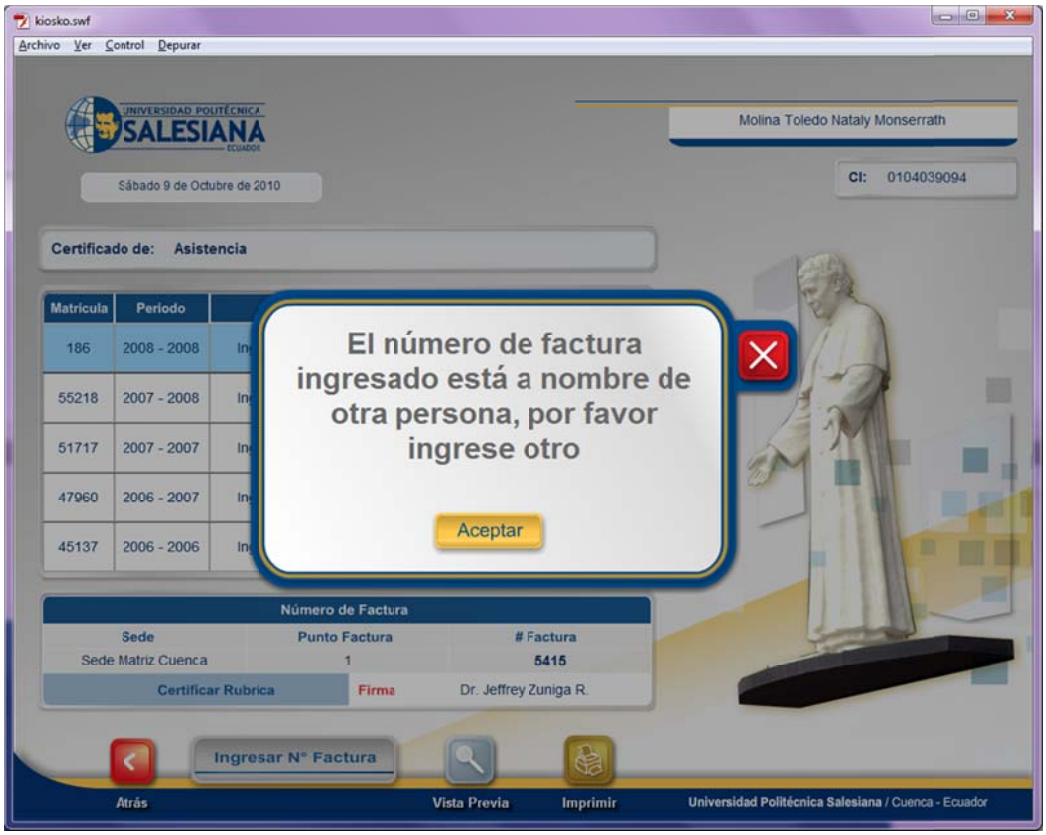
Si por un problema de digitación, el usuario no ingresa ningún número, el mensaje es el siguiente:



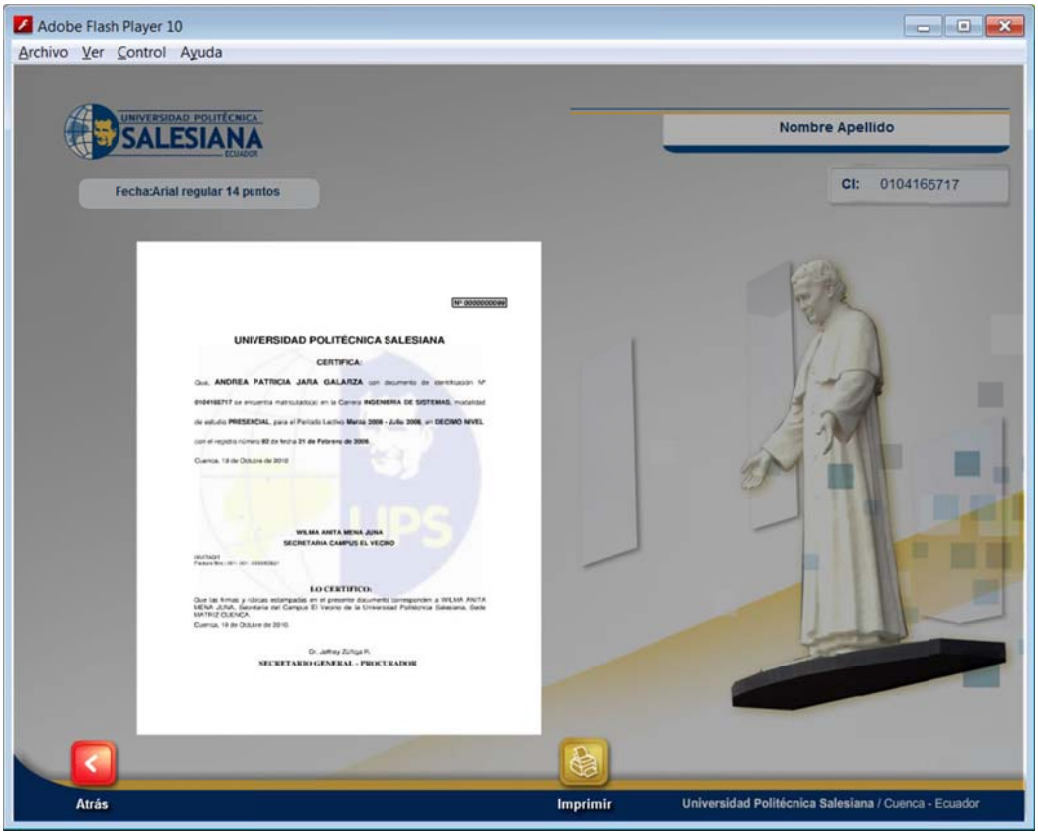
Si al momento de ingresar el número de factura, el usuario ingresa un número que no existe el mensaje es el siguiente:



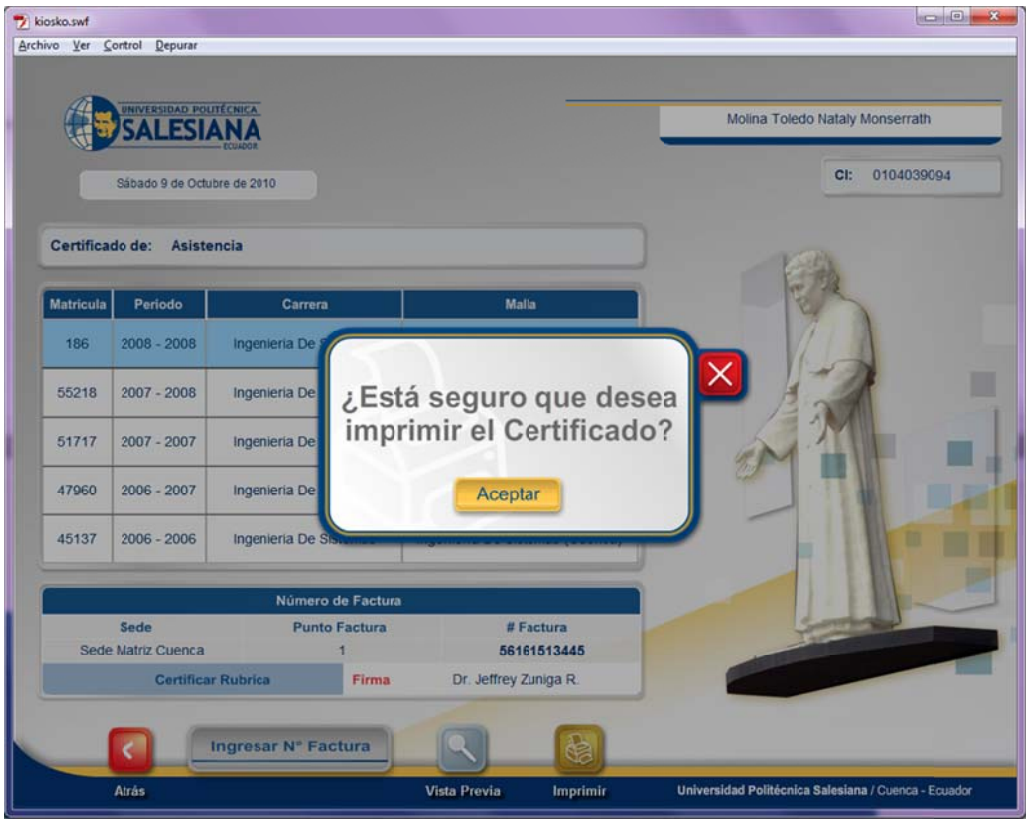
Si el usuario introduce un número de factura correspondiente a otro alumno el mensaje es el siguiente:



Antes de imprimir se puede obtener una vista previa de los resultados, pulsando el botón respectivo, la siguiente imagen expone un ejemplo de ella:

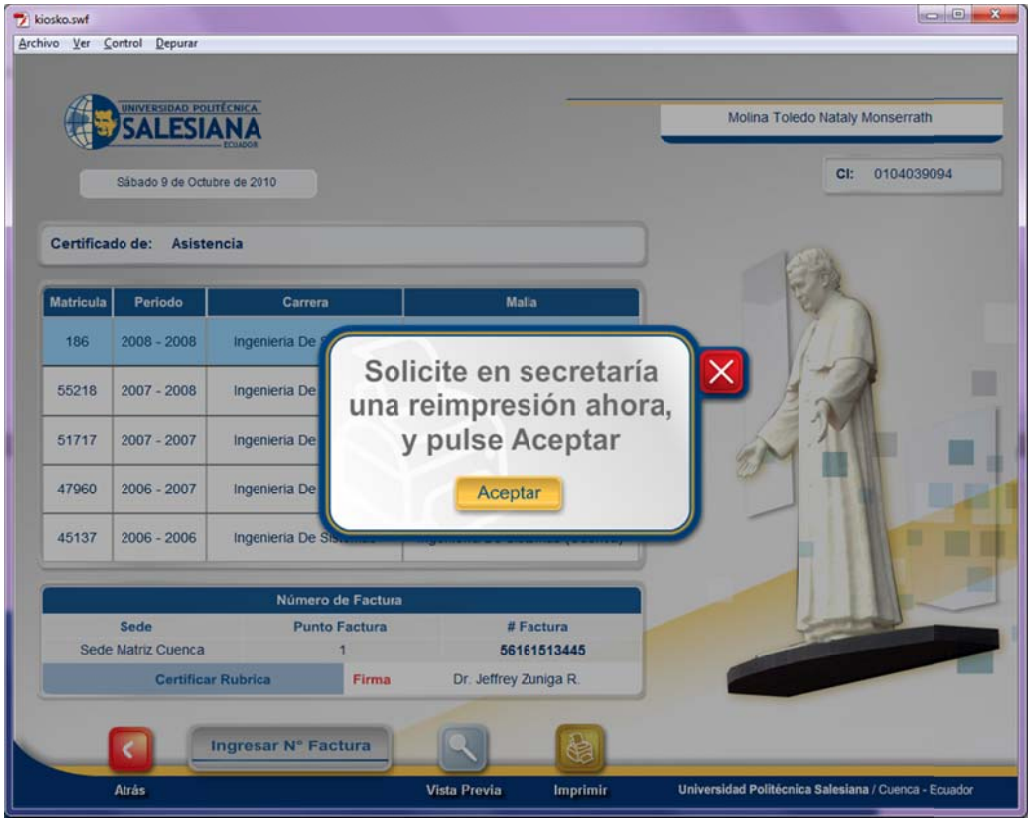


Por motivos de seguridad si el usuario pulsa el botón imprimir aparece el siguiente mensaje:

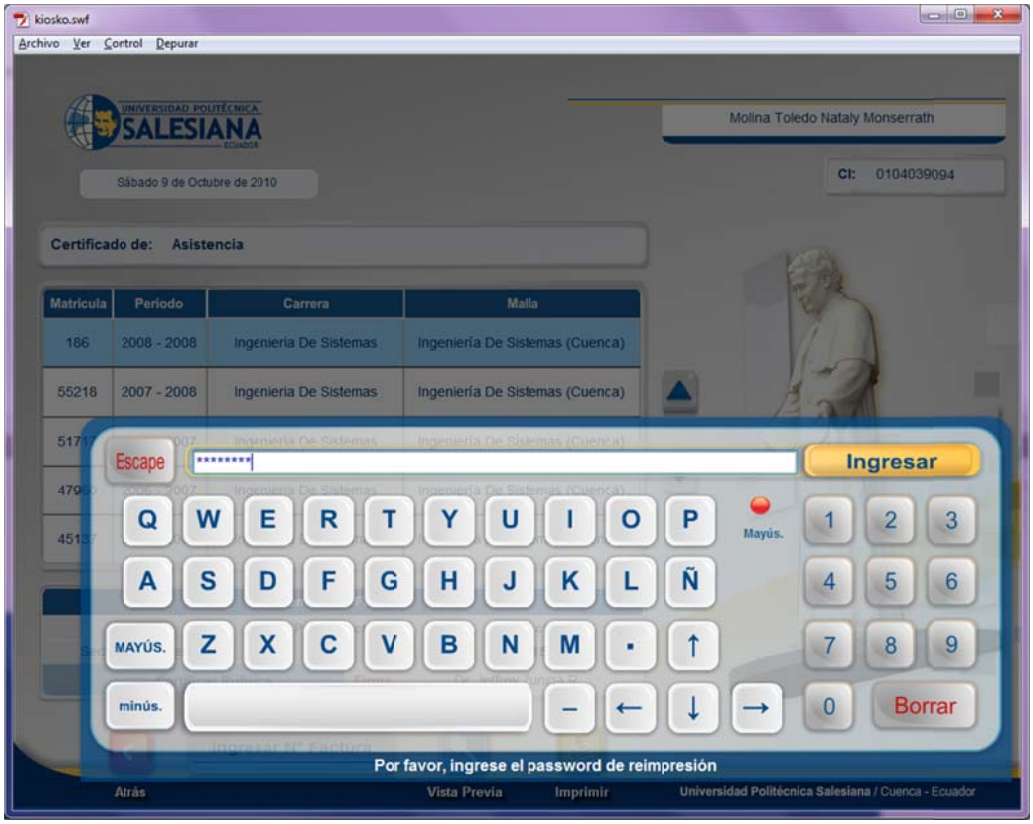


En caso de que no haya sido un error de digitación, el usuario pulsa ACEPTAR y obtiene el certificado; luego de esto, aparece el siguiente mensaje:

Si la respuesta es SI, se vuelve a la página de certificados, en caso de que el usuario requiera otro certificado, caso contrario se pulsa el botón salir. Si la respuesta es NO, se repite el mismo proceso de las SOLICITUDES, es decir, se solicita en secretaría el código de reimpresión:

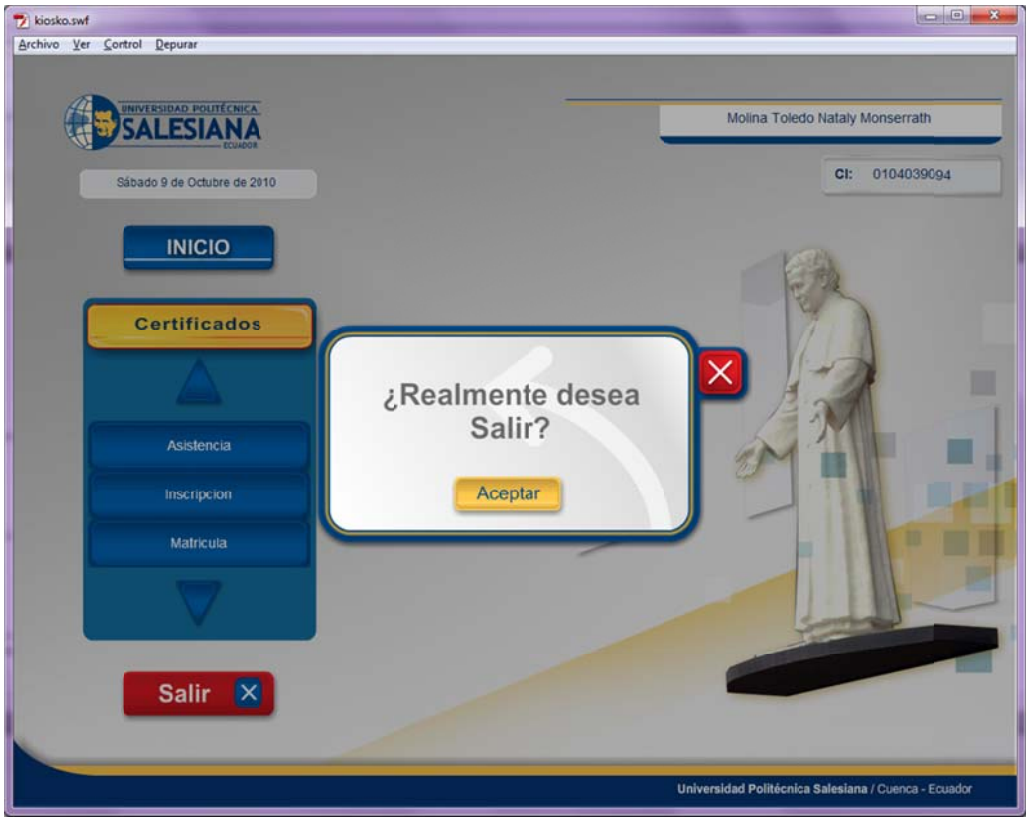

kiosko.swf
Archivo Ver Control Depurar
UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
Molina Toledo Nataly Monserrath
Sábado 9 de Octubre de 2010
CI: 0104039094
Certificado de: Asistencia
Matricula
Periodo
Carrera
Malla
186
2008 - 2008
55218
2007 - 2008
51717
2007 - 2007
47960
2006 - 2007
45137
2006 - 2006
Solicite en secretaría una reimpresión ahora, y pulse Aceptar
Aceptar
Número de Factura
Sede
Punto Factura
Factura
Sede Matriz Cuenca
1
56161513445
Certificar Rubrica
Firma
Dr. Jeffrey Zuniga R.
Ingresar N° Factura
Atrás
Vista Previa
Imprimir
Universidad Politécnica Salesiana / Cuenca - Ecuador

Luego de ello se habilita el teclado y la secretaria debe ingresar el código respectivo:



El certificado se imprime nuevamente, y si existe algún error el proceso se repite, caso contrario se regresa al menú principal de certificados.

Si dentro del proceso de los certificados, por algún error de digitación se pulsó el botón SALIR, se pregunta la confirmación de la acción a través del siguiente mensaje:



Si se desea volver al menú de acciones CERTIFICADOS y SOLICITUDES, se pulsa el botón INICIO, y para confirmar la acción, se presenta al usuario el siguiente mensaje: