Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL: Programa para gestion de ficheros ACA en HDDVD

AUTOR: Anais García Ruiz

TITULACIÓ: Ingeniería Técnica de Informática de Gestion

DIRECTOR: Jordi Fornés de Juan

DEPARTAMENT: Arquitectura de computadores

DATA: 29 de Junio de 2007

TÍTOL: Programa para gestión de ficheros ACA en HDDVD

COGNOMS: García Ruiz **NOM: Anais**

TITULACIÓ: Ingeniería Tecnica en informática

ESPECIALITAT: Informàtica de gestión **PLA: 92**

DIRECTOR: Jordi Fornés de Juan

DEPARTAMENT: Arquitectura de Computadores

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Aquest Projecte té en compte aspectes mediambientals: □ Sí □ No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

En este proyecto encontrarán el analisis, diseño y implementación de una aplicación. Esta aplicación es lector y creador de archivos de contenido avanzado (ACA – ***A****dvanced* ***C****ontent* ***A****rchive*).

Los archivos de contenido avanzado, son archivos utilizados por **HD DVD** (***H****igh* ***D****efinition* ***D****igital* ***V****ersatile* ***D****isc*), que es un formato de almacenamiento óptico desarrollado como un estándar para el DVD de alta definición por las empresas *Toshiba, Microsoft* y *NEC*, así como por varias productoras de cine.

Los archivos de contenido avanzado son usado para agrupar múltiples ficheros. Hecho que hace que mejore el funcionamiento de los HD DVD archivando múltiple contenido y reduciendo así el número de operaciones de búsqueda necesarias para cargar los múltiples archivos. Esto es un paso útil e importante.

Paraules clau (màxim 10):

HD DVD	ACA	Header	Pointer
Estructura ACA			

INVENTARIO DEL MATERIAL APORTADO

- Original de la memoria.
- Seis CDs con copia de la memoria, resumen y código fuente.
- Seis resúmenes en papel.

ÍNDICE

2:00 2:00

PRESENTACIÓN

PLANIFICACIÓN

DOCUMENTACIÓN

HERRAMIENTAS DE TRABAJO Y DESARROLLO

REQUERIMIENTOS

ESPECIFICACIÓN

Casos de uso de sistema

Eventos del sistema

PRESENTACIÓN

2:00 2:00

Introducción

En este proyecto observarán el análisis, diseño e implementación de una aplicación. Esta aplicación es un gestor de archivos de contenido avanzado (ACA – ***A**dvanced **C**ontent **A**rchive*)

Los archivos de contenido avanzado, son archivos utilizados por **HD DVD** (***H**igh **D**efinition **D**igital **V**ersatile **D**isc*), que es un formato de almacenamiento óptico desarrollado como un estándar para el DVD de alta definición por las empresas *Toshiba, Microsoft* y *NEC*, así como por varias productoras de cine.

Un archivo de contenido avanzado es usado para agrupar múltiples archivos. Hecho que hace que mejore el funcionamiento de los HD DVD archivando múltiple contenido y reduciendo así el número de operaciones de búsqueda necesarias para cargar los múltiples archivos. Esto es un paso útil e importante.

Motivación

El HD DVD es nuevo y reciente en el mercado, hecho que hace que todo estudio, aplicación o información en referente a este tema sea original y necesaria en el futuro.

Específicamente los que nos impulsó a realizar esta aplicación, la cual trata de gestionar los archivos de contenido avanzado, es que hoy en la actualidad tan solo hay dos aplicaciones en el mercado: Microsoft HD DVD Interactivity jumpstart y Scenarist de Sony.

Ambas aplicaciones utilizan como entorno de ejecución modo consola o MS-DOS y lo que se ha pretendido o nos ha motivado ha realizar nuestra aplicación; el gestor de ACA es realizar dicha aplicación en entorno grafico, es decir formulario para Windows, y así conseguir dar facilidad a cualquier usuario que trabaje con el HD DVD y el contenido de este.

Objetivos

Nuestros objetivos son realizar una aplicación que gestione archivos de contenido avanzado.

Esta aplicación tendrá un entorno grafico a modo de formulario o ventana de Windows.

En esta aplicación se podrá leer no tan solo los archivos de contenido avanzado creados por nuestra aplicación sino que podrá leer cualquier archivo de contenido avanzado generado por algunas de las aplicaciones existentes en el mercado, objetivo que pensamos cumplir ya que seguiremos el estándar de DVD FORUM.

También los archivos de contenido avanzado generados por nuestra aplicación podrán ser leídos no tan solo por nuestra aplicación sino por cualquiera de la existentes en el mercado ya que a la hora de crearlos seguiremos el estándar DVD FORUM.

Para llevar a cabo todos objetivos indicado en los párrafos anteriores, nos documentaremos sobre el HD DVD y el estándar de DVD FORUM

PLANIFICACIÓN

PLANIFICACIÓN

Fases del proyecto

Es hora de ponerle fechas a este proyecto. Es muy importante hacer una planificación en fechas del proyecto, ya que eso nos ayudará a la gestión más optima del proyecto, y asegurarnos de que se cumplirán los objetivos del proyecto.

Lo primero que haremos será analizar las tareas que componen nuestro proyecto. Cuanto más desglosadas y concretas sean estas tareas, más fácil nos será planificar el tiempo que tardaremos en realizarlas.

FASE 0: ESTUDIO DE LA PROPUESTA

Esta fase consiste en conocer de forma detallada las necesidades por las cuales ha surgido este proyecto, su viabilidad y las tecnologías más apropiados para su desarrollo.

FASE 1: DOCUMENTACIÓN

En esta fase buscaremos información sobre el HDDVD; su funcionamiento componentes, y haremos hincapié en los archivos de contenido avanzado (ACA- *Advanced Content Archive*), es decir buscar toda la información de este tipo de archivo, en especial la estructura seguida por el estándar del de DVDFORUM.

FASE 2: ESTUDIO DEL ENTORNO DE DESARROLLO

En esta fase se realiza el estudio de métodos y técnicas de programación orientada a objetos y a eventos, en el lenguaje de programación *Visual C++* y diseño de *Windows forms* necesarios para desarrollar la aplicación HDACA.

FASE 3: ESPECIFICACIÓN

Con las funcionalidades provenientes del análisis de requisitos hará falta especificar el modelo conceptual, los casos de uso necesarios y eventos del sistema.

FASE 4: DISEÑO DE CLASES

En esta fase, definiremos los diagramas de secuencia para cada una de las tareas que se han definido en la especificación del sistema.

FASE 5: DISEÑO GRÁFICO

En esta fase, construiremos el entorno gráfico de la aplicación, ya que es necesario para que la implementación vaya ligada al entorno gráfico de esta.

FASE 6: IMPLEMENTACIÓN DE LAS CLASES

Una vez se ha realizado el diseño de las funcionalidades del sistema, tan solo nos queda implementar el sistema en el lenguaje escogido y adaptarlo a la interfície gráfica siguiendo las decisiones tomadas en las fases anteriores.

FASE 7: INTEGRACIÓN DE CLASES Y PRUEBAS

En esta fase se probará el sistema en conjunto, la conclusión del trabajo experimental y elaboración del prototipo final.

FASE 8: REDACCIÓN DE LA MEMORIA

En esta fase, se destinará a la elaboración de la memoria, dejando constancia de todo el trabajo realizado en las etapas anteriores.

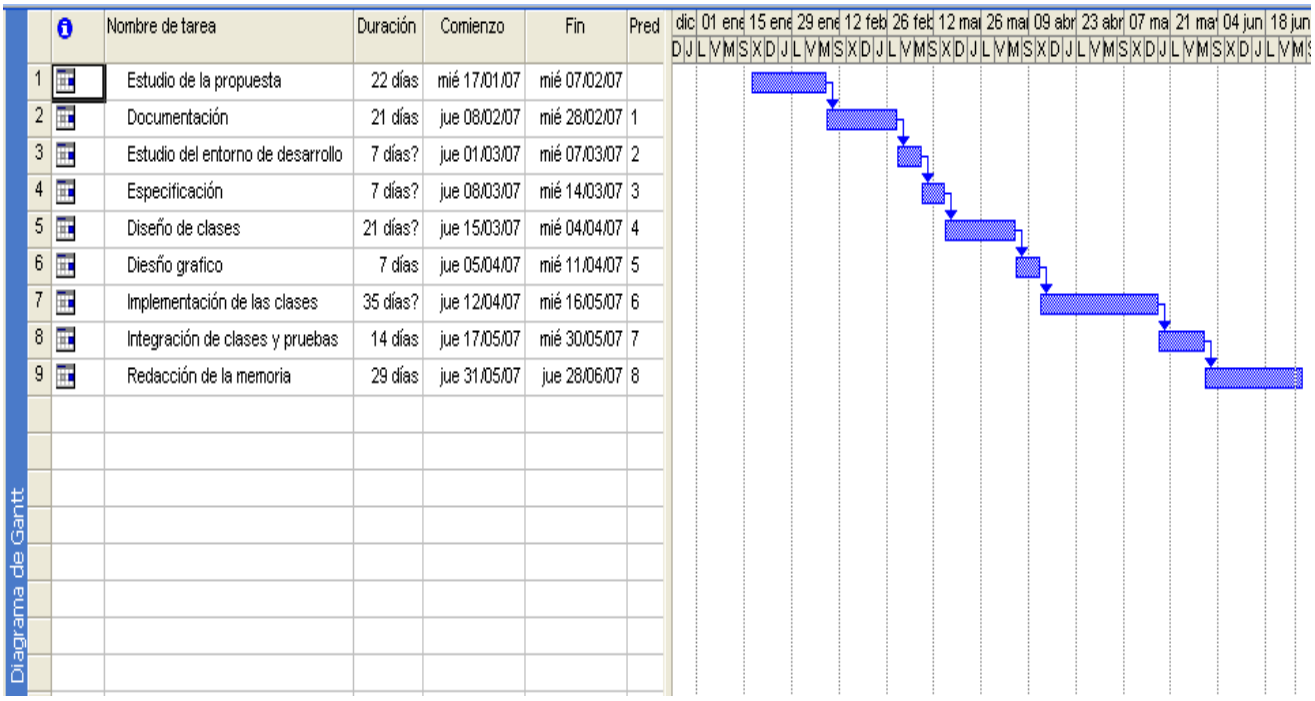
Planificación del tiempo en dias

En esta sección encontraremos el tiempo dedicado y sus respectivas fechas de inicio y final de cada una de las diferentes fases mencionadas en el punto anterior.

Fase	Nombre de la tarea	Inicio	Final	Dias
0	Estudio de la proposta	17/01/2007	07/02/2007	22
1	Documentación	08/02/2007	28/02/2007	21
2	Estudio del entorno de desarrollo	29/02/2007	07/03/2007	7
3	Especificación	08/03/2007	14/03/2007	7
4	Diseño de clases	15/03/2007	04/04/2007	21
5	Diseño grafico	05/04/2007	11/04/2007	7
6	Implementación de las clases	12/04/2007	16/05/2007	35
7	Integración de clases y pruebas	17/05/2007	30/05/2007	14
8	Redacción de la memoria	31/05/2007	28/06/2007	29

Diagrama de Gantt

Es la representación en forma secuencial de las siguientes actividades, obteniendo al final la fecha de cumplimiento del proyecto. También es la representación grafica de las dependencias.



	Nombre de tarea	Duración	Comienzo	Fin	Pred
1	Estudio de la propuesta	22 días	mié 17/01/07	mié 07/02/07	
2	Documentación	21 días	jue 08/02/07	mié 28/02/07	1
3	Estudio del entorno de desarrollo	7 días?	jue 01/03/07	mié 07/03/07	2
4	Especificación	7 días?	jue 08/03/07	mié 14/03/07	3
5	Diseño de clases	21 días?	jue 15/03/07	mié 04/04/07	4
6	Diesño grafico	7 días	jue 05/04/07	mié 11/04/07	5
7	Implementación de las clases	35 días?	jue 12/04/07	mié 16/05/07	6
8	Integración de clases y pruebas	14 días	jue 17/05/07	mié 30/05/07	7
9	Redacción de la memoria	29 días	jue 31/05/07	jue 28/06/07	8

Análisis de costes

El análisis del coste teórico de este proyecto ha sido basado únicamente por el número de horas dedicadas, ya sean de investigarción, especificación, diseño, implementación y documentación. Para ello no hemos tenido en cuenta las herramientas software necesarias, ya que son de libre distribución y no se paga por su uso.

El coste que supone el proyecto en horas de trabajo dedicadas, en función de las diferentes tareas, es:

Grupo	Nombre de la tarea	Horas/Fase	Horas
Investigación	Estudio de la propuesta Documentación Estudio del entorno de desarrollo	132 126 42	300
Especificación	Especificación	42	42
Diseño	Diseño de clases Diseño grafico	126 42	168
Implementación	Implementación de las clases Integración de clases y pruebas	210 84	294
Documentación	Redacción de la memoria	174	174
		Total	810

Precios por hora:

Grupo	Precio €/hora
Especificación y Diseño	50
Investigación, Implementación y Documentación	30

Coste del proyecto:

Grupo	Horas	Precio €/hora	Precio €
Investigación	300	30	9000
Especificación	42	50	2100
Diseño	168	50	8400
Implementación	294	30	8820
Documentación	174	30	5220
		Total	33540

DOCUMENTACIÓN

2:00 2:00

Documentación

HD DVD

HD DVD (**H**igh **D**efinition **D**igital **V**ersatile **D**isc) es un formato de almacenamiento óptico desarrollado como un estándar para el DVD de alta definición por las empresas Toshiba, Microsoft y NEC, así como por varias productoras de cine.

Existen HD-DVD de una capa, con una capacidad de 15 GB (unas 4 horas de vídeo de alta definición) y de doble capa, con una capacidad de 30 GB. Toshiba ha anunciado que existe en desarrollo un disco con triple capa, que alcanzaría los 45 GB de capacidad. En el caso de los HD-DVD-RW las capacidades son de 20 y 32 GB, respectivamente, para una o dos capas. La velocidad de transferencia del dispositivo se estima en 36,5 Mbps.

El HD-DVD trabaja con un láser violeta con una longitud de onda de 405 nm.

Por lo demás, un HD-DVD es muy parecido a un DVD convencional. La capa externa del disco tiene un grosor de 0,6 mm, el mismo que el DVD y la apertura numérica de la lente es de 0,65 (0,6 para el DVD).

Todos estos datos llevan a que los costos de producción de los discos HD-DVD sean bastante reducidos, dado que sus características se asemejan mucho a las del DVD actual.

Los formatos de compresión de vídeo que utiliza HD-DVD son MPEG-2, Video Codec 1 (VC1, basado en el formato Windows Media Video 9) y H.264/MPEG-4 AVC.

En el aspecto de la protección anti-copia, HD-DVD hace uso de una versión mejorada del CSS del DVD, el AACS, que utiliza una codificación de 128 bits. Además está la inclusión del ICT (Image Constraint Token), que es una señal que evita que los contenidos de alta definición viajen en soportes no cifrados y, por tanto, susceptibles de ser copiados. En la práctica, lo que hace es limitar la salida de video a la resolución de 960x540 si el cable que va del reproductor a la televisión es analógico, aunque la televisión soporte alta definición. El ICT no es obligatorio y cada compañía decide libremente si añadirlo o no a sus títulos. Por ejemplo, Warner está a favor de su uso mientras que Fox está en contra. La AACS exige que los títulos que usen el ICT deben señalarlo claramente en la caja.

Las posibilidades del HD DVD se ven enriquecidas con el uso de televisores y monitores que cumplan con el estándar de Alta Definición (medido en 1080i y 720p) que permiten una mejora absoluta en la apreciación de lo que es realmente capaz el formato HD DVD. A su vez, las compañías abocadas en el uso y comercialización de productos HD DVD, han incursionado en sistemas capaces de grabar en vivo material de Alta Definición en los discos HD DVD.

En lo que respecta a la experiencia de disfrutar una película de los mayores estudios

cinematográficos de Hollywood, el formato HD DVD introduce la posibilidad de acceder a menus interactivos al estilo "pop-up" lo que mejora sustancialmente la limitada capacidad de su antecesor, el DVD convencional, el cual poseía una pista especial dedicada al menú del film. Con esta inclusión de menus que pueden aparecer en cualquier parte del film, el HD DVD expande sus ventajas contra otros formatos al utilizar diferentes capas donde se registra la información, lo que permite una lectura diferenciada de los datos, y la superposición de imágenes, como así también una altísima calidad de sonido.

El HD-DVD realiza su incursión en el mundo de los videojuegos tras el anuncio de Microsoft de la comercialización de un extensor para HD-DVD para su popular consola Xbox 360.

Historia

El 19 de noviembre de 2003, los miembros de DVD Forum decidieron, con unos resultados de ocho contra seis votos, que el HD-DVD sería el sucesor del DVD para la HDTV. En aquella reunión, se renombró el, hasta aquel entonces, "Advanced Optical Disc". El soporte Blu-ray Disc que es de mayor capacidad, fue desarrollado fuera del seno del DVD Forum y nunca fue sometido a votación por el mismo.

La especificación actual para el HD-DVD y el HD-DVD-RW se encuentra en su versión 1.0. La especificación para el HD-DVD-R se encuentra en la versión 0.9.

Compatibilidad con anteriores tecnologías

Ya existen lectores híbridos capaces de leer y escribir CD, DVD y HD-DVD. También se ha conseguido desarrollar un disco híbrido de DVD y HD-DVD, de forma que se podría comprar una película que se puede ver en los reproductores de DVD actuales y, además, tener alta definición si se introduce en un reproductor de HD-DVD. Sin embargo, dichos discos necesitan de doble cara (por un lado DVD de doble capa y por el otro HD-DVD de una sola capa), debido a que la capa de datos es la misma en ambos formatos. Se ha conseguido un disco híbrido de una sola cara con una capa de DVD y otra capa de HD-DVD.

Archivo de Contenido Avanzado (ACA)

Un archivo de contenido avanzado es usado para agrupar múltiples archivos. Hecho que hace que mejore el funcionamiento de los HD DVD archivando múltiple contenido y reduciendo así el número de operaciones de búsqueda necesarias para cargar los múltiples archivos. Esto es un paso útil e importante, pero no necesario.

Cuando tenemos archivos contenidos en un archivo ACA, puedes acceder a este como si el archivo ACA fuera un directorio:

file:///hddvddisc/ADV_OBJ/archivefile.aca/content.jpg

Estructura del Archivo de contenido avanzado

Stream/archivado de datos avanzados

Los siguientes archivos deben ser archivados como un fichero sin compresión usando el formato de archivado definido en esta especificació:

- Playlist
- Manifesto
- Markup
- Subtítulos avanzados
- Script
- Imágenes (JPEG/PNG/MNG/Formato de captura de imagen)
- Efectos de audio
- Fuente(texto)
- EVOB para el Set del Video segundario.
- TMAP para el Set del Video segundario.
- Otros.

Entendemos por EVOB (.evo) y TMAP (.map) los archivos de video y audio multiplexados.

El fichero de archivado debe ser multiplexado como un Advanced Stream en un EVOB del Set de video primario, y en este caso, el archivo es dividido en paquetes llamados Advanced Packs(ADV_PCK), y el mismo archivo (con el mismo nombre) debe ser grabado bajo el directorio ADV_OBJ. El Advanced Stream solo puede llevar ficheros archivados(no ficheros individuales como Markup, imágenes, …)
Este fichero archivado debe estar en un disco(bajo el directorio ADV_OBJ) o se debe obtener de la red o del Persistent Storege(Disco Duro del reproductor).

Formato de archivado

El archivado de datos consiste en una cabecera de fichero, uno o varios punteros de búsqueda de datos de recursos, es decir, habrá un puntero por cada uno de los ficheros contenidos en el archivado de datos avanzados, y datos de recurso(archivos) como se muestra en la figura 3.1:

Punteros de búsqueda de datos de recurso #1
…
Punteros de búsqueda de datos de recurso #n
Datos de recursos #1
…
Datos de recursos #n

Figura 3.1

Cabecera del fichero

La cabecera consta de los siguientes campos:

Nombre del campo	Contenido	Nº de bytes
(1) FILE_ID	Identifica el fichero .aca	8 bytes
(2) VERN	Número de versión	2 bytes
(3) FILE_TY	Tipo de fichero	1 byte
(4) ENC_TY	Tipo de encriptación usada por Name String	1byte
(5) SRP_Ns	Nº de punteros de búsqueda de recursos de datos	2 bytes
(6) FILE_SZ	Tamaño del fichero .aca	4 bytes
reserved	Espacio reservado	14 bytes

(1) FILE_ID

Describe “HDDVDACA” para identificar el fichero con el set de caracteres de ISO 8859-1.

(2) VERN

Describe el número de versión de esta especificación. Se refiere a (RBP 32 a 33)

b15	b14	b13	b12	b11	b10	b9	b8
reserved							

b7	b6	b5	b4	b3	b2	b1	b0
Book Part version							

Book Part version … 0001 0000b: version 1.0
Otros: reservado

(3) FILE_TY

Describe el tipo de fichero.

00h: Si contiene datos no comprimidos.

Otros: reservado.

(4) NC_TY

Describe el tipo de texto codificado usado por el recurso Name String de las entradas.

01h: ISO 8859-1.

Otros: reservado.

(5) SRP_Ns

Describe el número de Punteros de búsqueda de datos de recursos.

(6) FILE_SZ

Describe el tamaño del fichero en número de bytes.

Si el tamaño del fichero es desconocido, este campo se rellenará con '0b'.

Punteros de búsqueda de datos

Los punteros de búsqueda de datos de recursos constan de los siguientes campos:

Nombre del campo	Contenido	Nº de bytes
(1) DATA_SA	Dirección donde comienza los datos de recursos	4 bytes

(2) DATA_SZ	Tamaño de los datos	4 bytes
(3) DATA_CRC	CRC de 32 bits de los datos	4 bytes
(4) DATA_MIME_TY	Tipo de MIME de los datos	1byte
(5) DATA_FNAME_LEN	Longitud del nombre del fichero	1byte
(6) DATA_FNAME	Nombre del fichero(DATA_FNAME_LEN bytes)	(DATA_FNAME_LEN bytes)
reserved	Espacio reservado	32 bytes

(1) DATA_SA

Describe la dirección donde comienza los datos del recurso en números de bytes.

(2) DATA_SZ

Describe el tamaño del dato de recurso en número de bytes.

(3) DATA_CRC

Describe el CRC(Códigos de Redundancia Cíclica) del dato de recurso, el cual se calcula usando métodos estandarizados de CRC con condiciones PRE y POST, como se define en ISO 3309 y en ITU-T v.42. El CRC polinomial empleado es:

x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x +1

(4) DATA_MIME_TY

Describe el código del tipo de MIME del dato de recurso. (*veasé tabla siguiente*)

Tipo de Código MIME	Tipo MIME	Sufijo
00h	reservado	reserved
01h	text/hddvdpl+xml	xpl,XPL
02h	text/hddvdmf+xml	xmf,XMF
03h	text/hddvdmu+xml	xmu,XMU
04h	text/hddvdts+xml	xts,XTS

05h	text/hddvdas+xml	xas,XAS
06h	text/hddvdss+xml	xss,XSS
07h	application/ecmascript	js,JS
08h	video/evob	evo,EVO
09h	application/tmap	map,MAP
0Ah	image/jpeg	jpg,JPG
0Bh	image/png	png,PNG
0Ch	image/mng	mng/MNG
0Dh	image/cvi	cvi,CVI
0Eh	image/cdw	cdw,CDW
0Fh	audio/x-wav	wav,WAV
10H	application/font	otf,OTF,ttf,TTF,ttc,TTC
FFh	application/x-data	see Note
Otros	reservado	resevado

(5) DATA_FNAME_LEN

Describe el tamaño del nombre de fichero del dato de recurso.

(6) DATA_FNAME

Describe el nombre del recuso de datos

Recurso de datos

Cada campo es localizado con un puntero de búsqueda a recurso de datos, y el campo que contiene la posición donde se encuentra el recurso a datos del fichero deseado.

Para consultar un recurso de datos deberá s conocer el campo anterior mencionado.

HERRAMIENTAS DE TRABAJO Y DESARROLLO

Herramientas de trabajo y desarrollo

Lenguajes de programación

Para realizar el proyecto decidimos utilizar como lenguaje de desarrollo Visual C++. Creemos que es un tipo de lenguaje que nos dará lugar y facilidad a la hora de implementar la solución propuesta.

Introducción al lenguaje Visual C++

C++ es un lenguaje de programación de alto nivel con el que se puede escribir cualquier tipo de programa.

Una de las ventajas más significativas de C++ sobre otras lenguajes de programación es que soporta diferentes estilos de programación, como son la programación estructurada o la orientada a objetos. Combinando estos estilos se pueden escribir programas elegantes y eficientes; esto es, C++ es un lenguaje multiparadigma por excelencia. En cuanto su portbilidad entre diferentes plataformas de desarrollo, C++ es independiente sólo en código fuente, lo cual significa que cada plataforma diferente debe proporcionar el compilador adecuado para obtener el código máquina que tiene que ejecutarse.

La plataforma utilizada en nuestro caso ha sido Visual C++.

Windows, producto introducido por Microsoft en 1985, es el entorno más popular de interfaz gráfico de usuario (GUI). Para el usuario, Windows es un entorno multitarea basado en ventanas que corresponden con programas. Esto significa que permite ejecutar concurrentemente programas especialmente escritos para dicho entorno y también programas escritos para MS-DOS.

Para el desarrollo de programas, Windows provee rutinas que permiten utilizar componentes como menús, cuadro de dialogo y barras de desplazamiento, entre otros. Así mismo, el programador puede manipular el ratón , el teclado, el monitor, la impresora, los puertos de comunicaciones y el reloj del sistema sin tener en cuenta el dispositivo periférico.

Microsoft Visual C++ es un entorno de programación donde se combinan la programación orientada a objetos(C++) y el sistema de desarrollo diseñado especialmente para crear aplicaciones gráficas para Windows(SDK).

Aunque las aplicaciones Windows son sencillas de utilizar, el desarrollo de las mismas no es una tarea fácil. Por ello, para hacer más asequible esta tarea, Visual C++ incluye, además de varias herramientas que lo convierten en un generador de programas C++, un conjunto completo de clases(*Microsoft Foundation Class,* MFC) que permiten crear de una forma intuitiva las aplicaciones Windows y manejar los componentes de Windows según su naturaleza de objetos. Esto es, MFC es una biblioteca de clases que encapsula las funciones de la API de Windows, para crear y manipular objetos Windows. Así, un objeto Windows, por ejemplo, una ventana, se construye creando un objeto de la clase **CWnd**; esta clase proporcina la funciobaldad necesaria para crear y destruir la ventana, así como para manipular cualquier característica de la misma. Dicha funcionalidad se ha implementado a partir de las funciones de la API de Windows.

Las características más sobresalientes de Visual C++

- Una biblioteca de clases, MFC, que da soporte a los objetos Windows tales como ventanas, cajas de diálogo, controles(por ejemplo, etiquetas, cajas de texto, botones de pulsación, etc.), así como los objetos GDI (*Graphics Device Interface*) tales como lápices (*pens*), pinceles (*brushes*), fuentes (*fonts*) y mapa de bits (*btimaps*). A su vez se comunican entres sí mediante mensajes, también soportados por las MFC.

- Un entorno de desarrollo integrado (editor de texto, compilador, depurador, explorador de código fuente, administrador de proyectos, etc.).

- El editor de textos le ayuda ahora a completar cada una de las sentencias visualizando la sintaxis correspondiente a las mismas.

- Asistentes para el desarrollo de aplicaciones como *AppWizard*, *editores de recursos, ClassWizard, ControlWizard, WizardBara* y *ATL COM Wizard.*

- Galería de objetos incrustados y vinculados (OLE – *Object Linking and Embedding*). Esto es, software autocontenido en pequeñas y potentes unidades o *componentes software* para reutilizar en cualquier aplicación. Asimismo, Visual C++ proporciona soporte para diseñar componentes software a medida.

- Visualización y manipulación de datos de otras aplicaciones Windows utilizando controles OLE.

- Una interfaz para múltiples documentos (MDI – *Multiple Document Interface*) que permite crear múltiples una aplicación con unas ventana principal y múltiples ventanas de documento. Un ejemplo de este tipo de aplicaciones es *Microsoft Word*.

- Personalización de *AppWizard*.

- Cabeceras precompiladas que reducen el tiempo de compilación.

- Editar y continuar. Durante una sesión de depuración, se pueden realizar modificaciones en el código de la aplicación sin tener que salir de dicha sesión, recompilar y reiniciar la depuración. Los cambios son recompilados y aplicados a la aplicación que se está ejecutando.

- Nuevas clases para la programación de hilos (*threads*), para implementar páginas HTML, etc.

- Creación y utilización de bibliotecas dinámicas (DLL - *Dynamic Linf Libraries*).

- Visual C++ integra la biblioteca estándar de C++ e incorpora soporte para la programación de aplicaciones para Internet; forma parte de este soporte la tecnología de componentes activos (*ActiveX*).

- Soporte para el estándar COM (*Component Object Model* – modelo de objeto componente; en otras palabras, componente software) al que pertenecen los componentes activos (*ActiveX* o formalmente controles OLE).

- Objetos de acceso a datos (DAO) que permiten acceder a bases de datos a través del motor de *Access* o de controladores ODBC.

- OLE DB como proveedor de datos y objetos ADO (*ActiveX Data Objects* – objetos *ActiveX* para acceso a datos), como tecnología de acceso a datos, para satisfacer los nuevos escenarios demandados por las empresas, tales como los sistemas de información basados en la Web.

- Soporte para aplicaciones que interaccionen con Internet a través de la *API* para *Internet* de *Windows* (biblioteca *WinInet*).

- Incorporación de nuevas definiciones. Por ejemplo, el tipo **bool** y sus constantes asociadas **true** y **false**, los tipos **_variant_t** (**COleVariant**), **_bstr_t** y **_com_ptr_t**, la directriz **#import**, etc.

Herramienta de desarrollo Microsoft Visual Studio

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML.

Aspectos dastacados de Visual Studio

Esta sección contiene información acerca de algunas de las herramientas y tecnologías más avanzadas de esta versión de Visual Studio.

Visual Studio Tools para Office

Microsoft Visual Studio 2005 Tools para Microsoft Office System puede ayudarle a crear soluciones al extender documentos de Word 2003 y libros de Excel 2003 mediante Visual Basic y Visual C#. Visual Studio Tools para Office incluye nuevos proyectos de Visual Studio para crear el código subyacente en documentos de Word, plantillas de Word, libros de Excel y plantillas de Excel. Para obtener más información, vea Información general sobre el desarrollo de soluciones de Office.

Visual Web Developer

Visual Studio incluye un nuevo diseñador de páginas Web denominado Visual Web Developer que incluye muchas mejoras para la creación y edición de páginas Web ASP.NET y páginas HTML. Proporciona una forma más fácil y rápida de crear páginas de formularios Web Forms que en Visual Studio .NET 2003.

Visual Web Developer incluye mejoras en todas las áreas de desarrollo de sitios Web. Puede crear y mantener los sitios Web como carpetas locales, en Servicios de Internet Information Server (IIS), o en un servidor FTP o SharePoint. El diseñador Visual Web Developer admite todas las mejoras de ASP.NET, incluidas las casi dos docenas de nuevos controles que simplifican muchas tareas de desarrollo Web. Para obtener más información, vea Lo nuevo en el desarrollo Web de Visual Studio.

Aplicaciones para dispositivos inteligentes

El entorno integrado de Visual Studio incluye herramientas destinadas a dispositivos como los PDA y Smartphone. Entre las mejoras se encuentran tiempos de ejecución de dispositivos nativos y herramientas de Visual C++, diseñadores administrados que proporcionan un modo WYSIWYG mejorado específico para cada plataforma y compatibilidad con varios factores de forma, un nuevo emulador, herramientas de control de datos similares al escritorio, y proyectos de implementación para el usuario final que eliminan la edición manual de los archivos .inf. Para obtener más información, vea Lo nuevo en proyectos de Smart Device.

Formularios Web Forms

Los formularios Web Forms son una tecnología que se utiliza para crear páginas Web programables. Los formularios Web Forms se representan como código HTML y secuencias de comandos compatibles con exploradores, lo que permite ver las páginas en cualquier explorador y plataforma. Mediante el uso de formularios Web Forms se pueden crear páginas Web arrastrando y colocando controles en el diseñador y agregando código posteriormente, de forma parecida a la creación de formularios en Visual Basic. Para obtener más información, vea Información general sobre páginas Web ASP.NET.

Formularios Windows Forms

Los formularios Windows Forms sirven para crear aplicaciones de Microsoft Windows en .NET Framework. Este marco de trabajo proporciona un conjunto de clases claro, orientado a objetos y ampliable, que permite desarrollar complejas aplicaciones para Windows. Además, los formularios Windows Forms pueden actuar como interfaz de usuario local en una solución distribuida de varios niveles. Para obtener más información, vea Introducción a los formularios Windows Forms.

Servicios Web XML

Los Servicios Web XML son aplicaciones que pueden recibir solicitudes y datos mediante XML a través de HTTP. no están ligados a una tecnología de componentes particular o a una convención de llamada de objetos y, por tanto, se puede obtener acceso a ellos mediante cualquier lenguaje, modelo de componente o sistema operativo. En Visual Studio, se pueden crear e incluir con rapidez Servicios Web XML mediante Visual Basic, Visual C#, JScript o servidor ATL. Para obtener más información, vea Introducción a la programación de servicios Web XML en código administrado.

Compatibilidad con XML

El Lenguaje de marcado extensible (XML) proporciona un método para describir datos estructurados. XML es un subconjunto de SGML optimizado para la entrega a través de Web. El Consorcio World Wide Web (W3C) define los estándares de XML para que los datos estructurados sean uniformes e independientes de las aplicaciones. Visual Studio es totalmente compatible con código XML e incluye el Diseñador XML para facilitar la edición de XML y la creación de esquemas XML. Para obtener más información, vea Diseñador XML.

Visual Studio Team System

Visual Studio 2005 Team System es una plataforma de herramientas del ciclo de vida del desarrollo de software extensible, integrado y productivo que ayuda a los equipos de desarrollo de software mediante la mejora de las comunicaciones y la colaboración durante todo el proceso de desarrollo. Consta de lo siguiente:

- Documentación de Team Foundation es un servidor de colaboración de equipo extensible que proporciona seguimiento de elementos de trabajo, control de código fuente, información e instrucciones sobre el proceso.
- Documentación de Team Edition para Architects es un conjunto de herramientas de diseño de aplicaciones integradas para el desarrollo de servicios.
- Documentación de Team Edition para Developers proporciona herramientas de calidad del código y rendimiento que permiten a los equipos generar servicios y aplicaciones confiables y críticos.
- Documentación de Team Edition para Testers proporciona herramientas avanzadas de prueba de carga que permiten a los equipos comprobar el rendimiento de las aplicaciones antes de su implementación.

El entorno .NET Framework

.NET Framework es un entorno multilenguaje que permite generar, implantar y ejecutar aplicaciones y Servicios Web XML. Consta de tres partes principales:

- **Common Language Runtime** A pesar de su nombre, el motor en tiempo de ejecución desempeña una función tanto durante la ejecución como durante el desarrollo de los componentes. Cuando el componente se está ejecutando, el motor en tiempo de ejecución e responsable de administrar la asignación de memoria, iniciar y detener subprocesos y procesos, y hacer cumplir la directiva de seguridad, así como satisfacer las posibles dependencias del componente sobre otros componentes. Durante el desarrollo, el papel del motor en tiempo de ejecución cambia ligeramente; a causa de la gran automatización que

permite (por ejemplo, en la administración de memoria), el motor simplifica el trabajo del desarrollador, especialmente al compararlo con la situación actual de la tecnología COM. En concreto, funciones tales como la reflexión reducen de forma espectacular la cantidad de código que debe escribir el desarrollador para convertir la lógica de empresa en componentes reutilizables.

- **Clases de programación unificadas** El entorno de trabajo ofrece a los desarrolladores un conjunto unificado, orientado a objetos, jerárquico y extensible de bibliotecas de clases (API). Actualmente, los desarrolladores de C++ utilizan las Microsoft Foundation Classes y los desarrolladores de Java utilizan las Windows Foundation Classes. El entorno de trabajo unifica estos modelos dispares y ofrece a los programadores de Visual Basic y JScript la posibilidad de tener también acceso a las bibliotecas de clases. Con la creación de un conjunto de API comunes para todos los lenguajes de programación, Common Language Runtime permite la herencia, el control de errores y la depuración entre lenguajes. Todos los lenguajes de programación, desde JScript a C++, pueden tener acceso al entorno de trabajo de forma parecida y los desarrolladores pueden elegir libremente el lenguaje que desean utilizar.

ASP.NET ASP.NET construye las clases de programación de .NET Framework, lo que proporciona un modelo de aplicación Web con un conjunto de controles e infraestructura que facilitan la generación de aplicaciones Web. ASP.NET incluye un conjunto de controles que encapsulan elementos comunes de interfaz de usuario de HTML, como cuadros de texto, botones y cuadros de lista. Sin embargo, dichos controles se ejecutan en el servidor Web, y representan la interfaz de usuario en el explorador como HTML. En el servidor, los controles exponen un modelo de programación orientado a objetos que proporciona la riqueza de la programación orientada a objetos al desarrollador Web. ASP.NET también proporciona servicios de infraestructura, como la administración de estado y el reciclaje de procesos, que reduce aún más la cantidad de código que debe escribir el desarrollador y aumenta la confiabilidad de la aplicación. Asimismo, ASP.NET utiliza estos mismos conceptos para permitir a los desarrolladores la entrega de software como un servicio. Al utilizar características de Servicios Web XML, los desarrolladores de ASP.NET pueden escribir su lógica empresarial y utilizar la infraestructura de ASP.NET para entregar ese servicio a través de SOAP.

UML

Lenguaje Unificado de Modelado (**UML**, por sus siglas en inglés, *Unified Modeling Language*)es un *lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos* (OO). Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.

UML se quiere convertir en un *lenguaje estándar* con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Sin embargo, hay que tener en cuenta un aspecto importante del modelo: no pretende definir un modelo estándar de desarrollo, sino únicamente un lenguaje de modelado. Otros métodos de modelaje como OMT (*Object Modeling Technique*) o Booch sí definen procesos concretos. En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión, por poner un ejemplo. Las diferencias son muy marcadas y afectan a todas las faces del proceso. El método del UML recomienda utilizar los procesos que otras metodologías tienen definidos.

Los Inicios

A partir del año 1994, Grady Booch [Booch96](precursor de Booch '93) y Jim Rumbaugh (creador de OMT) se unen en una empresa común, *Rational Software Corporation*, y comienzan a unificar sus dos métodos. Un año más tarde, en octubre de 1995, aparece UML (*Unified Modeling Language*) 0.8, la que se considera como la primera versión del UML. A finales de ese mismo año, Ivar Jacobson, creador de OOSE (*Object Oriented Software Engineer*) se añade al grupo.

Como objetivos principales de la consecución de un nuevo método que aunara los mejores aspectos de sus predecesores, sus protagonistas se propusieron lo siguiente:

- El método debía ser capaz de modelar no sólos sistemas de software sino otro tipo de sistemas reales de la empresa, siempre utilizando los conceptos de la orientación a objetos (OO).
- Crear un lenguaje para modelado utilizable a la vez por máquinas y por personas.
- Establecer un acoplamiento explícito de los conceptos y los *artefactos* ejecutables.
- Manejar los problemas típicos de los sistemas complejos de misión crítica.

Lo que se intenta es lograr con esto que los lenguajes que se aplican siguiendo los métodos más utilizados sigan evolucionando en conjunto y no por separado. Y además, unificar las perspectivas entre diferentes tipos de sistemas (no sólo software, sino también en el ámbito de los negocios), al aclarar las fases de desarrollo, los requerimientos de análisis, el diseño, la implementación y los conceptos internos de la OO.

Modelado de objetos

En la especificación del UML podemos comprobar que una de las partes que lo componen es un metamodelo formal. Un *metamodelo* es un modelo que define el lenguaje para expresar otros modelos. Un modelo en OO es una *abstracción cerrada semánticamente de un sistema y un sistema es una colección de unidades conectadas que son organizadas para realizar un propósito específico*. Un sistema puede ser descripto por uno o más modelos, posiblemente desde distintos puntos de vista.

Una parte del UML define, entonces, una abstracción con significado de un lenguaje para expresar otros modelos (es decir, otras abstracciones de un sistema, o conjunto de unidades conectadas que se organizan para conseguir un propósito). Lo que en principio puede parecer complicado no lo es tanto si pensamos que uno de los objetivos del UML es llegar a convertirse en una manera de definir modelos, no sólo establecer una forma de modelo, de esta forma simplemente estamos diciendo que UML, además, define un lenguaje con el que podemos abstraer cualquier tipo de modelo.

El UML es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos. El modelado no es más que la construcción de un modelo a partir de una especificación. Un modelo es una abstracción de algo, que se elabora para comprender ese algo antes de construirlo. El modelo omite detalles que no resultan esenciales para la comprensión del original y por lo tanto facilita dicha comprensión.

Los modelos se utilizan en muchas actividades de la vida humana: antes de construir una casa el arquitecto utiliza un plano, los músicos representan la música en forma de notas musicales, los artistas pintan sobre el lienzo con caboncillos antes de empezar a utilizar los óleos, etc. Unos y otros abstraen una realidad compleja sobre unos bocetos, modelos al fin y al cabo. La OMT, por ejemplo, intenta abstraer la realidad utilizando tres clases de modelos OO: el *modelo de objetos*, que describe la estructura estática; el *modelo dinámico*, con el que describe las relaciones temporales entre objetos; y el *modelo funcional* que describe las relaciones funcionales entre valores. Mediante estas tres fases de construcción de un modelos, se consigue una abstracción de la realidad que tiene en sí misma información sobre las principales características de ésta.

Los modelos además, al no ser una representación que incluya todos los detalles de los originales, permiten probar más fácilmente los sistemas que modelan y determinar los errores. Según se indica en la *Metodología OMT* (Rumbaugh), los modelos permiten una mejor comunicación con el cliente por distintar razones:

- Es posible enseñar al cliente una posible aproximación de lo que será el producto final.
- Proporcionan una primera aproximación al problema que permite visualizar cómo quedará el resultado.
- Reducen la complejidad del original en subconjuntos que son fácilmente tratables por separado.

Se consigue un modelo completo de la realidad cuando el modelo captura los aspectos importantes del problema y omite el resto. Los lenguajes de programación que estamos acostumbrados a utilizar no son adecuados para realizar modelos completos de sistemas reales porque necesitan una especificación total con detalles que no son importantes para el algoritmo que están implementando. En OMT se modela un sistema desde tres puntos de vista diferentes donde cada uno representa una parte del sistema y una unión lo describe de forma completa. En esta técnica de modelado se utilizó una aproximación al proceso de implementación de software habitual donde se utilizan estructuras de datos (modelo de objetos), las operaciones que se realizan con ellos tienen una secuencia en el tiempo (modelo dinámico) y se realiza una transformación sobre sus valores (modelo funcional).

UML utiliza parte de este planteamiento obteniendo distintos puntos de vista de la realidad que modela mediante los distintos tipos de diagramas que posee. Con la creación del UML se persigue obtener un lenguaje que sea capaz de abstraer cualquier tipo de sistema, sea informático o nó, mediante los diagramas, es decir, mediante representaciones gráficas que contienen toda la información relevante del sistema. Un **diagrama** es *una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con los elementos del modelo*. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibujan de forma que se resaltan los detalles necesarios para entender el sistema.

Artefactos para el Desarrollo de Proyectos

Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software. Pueden ser artefactos un modelo, una descripción o un software. Los artefactos de UML se especifican en forma de diagramas, éstos, junto con la documentación sobre el sistema constituyen los artefactos principales que el modelador puede observar.

Se necesita más de un punto de vista para llegar a representar un sistema. UML utiliza los diagramas gráficos para obtener estos distintos puntos de vista de un sistema:

- Diagramas de Casos de Uso.
- Diagramas de Clases.
- Diagramas de Comportamiento o Interacción.
- Diagramas de Implementación.

Nuevas características del UML

Además de los conceptos extraídos de métodos anteriores, se han añadido otros nuevos que vienen a suplir carencias antiguas de la metodología de modelado. Estos nuevos conceptos son los siguientes:

- Definición de estereotipos: un estereotipo es una nueva clase de elemento de modelado que debe basarse en ciertas clases ya existentes en el metamodelo y constituye un mecanismo de extensión del modelo.
- Responsabilidades.
- Mecanismos de extensibilidad: estereotipos, valores etiquetados y restricciones.
- Tareas y procesos.
- Distribución y concurrencia (para modelar por ejemplo ActiveX/DCOM y CORBA).
- Patrones/Colaboraciones.
- Diagramas de actividad (para reingeniería de proceso de negocios)
- Clara separación de tipo, clase e instancia.
- Refinamiento (para manejar relaciones entre niveles de abstracción).
- Interfaces y componentes.

El Proceso de Desarrollo

UML no define un proceso concreto que determine las fases de desarrollo de un sistema, las empresas pueden utilizar UML como el lenguaje para definir sus propios procesos y lo único que tendrán en común con otras organizaciones que utilicen UML serán los tipos de diagramas.

UML es un método independiente del proceso. Los procesos de desarrollo deben ser definidos dentro del contexto donde se van a implementar los sistemas.

Herramientas CASE

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Desarrollo Iterativo

Rational Rose utiliza un proceso de desarrollo iterativo controlado (*controlled iterative process development*), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada

iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.

Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Trabajo en Grupo

Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

Generador de Código

Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

Ingeniería Inversa

Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

REQUERIMIENTOS

Requerimientos

¿Que son los requerimientos?

Normalmente, un tema de la Ingeniería de Software tiene diferentes significados. De las muchas definiciones que existen para requerimiento, ha continuación se presenta la definición que aparece en el glosario de la IEEE .

> (1) Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
>
> (2) Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
>
> (3) Una representación documentada de una condición o capacidad como en (1) o (2).

Los requerimientos puedes dividirse en requerimientos funcionales y requerimientos no funcionales.

Requerimientos no funcionales

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

A continuación se describen los requerimientos no funcionales específicos del sistema:

Recursos humanos

- Los usuarios que interactúan con el sistema serán usuarios finales que no tienen porque estar ligados al mundo de la informática.
- El sistema por tanto ha de ser fácil de usar, pues no es necesario que los usuarios finales tengan muchos conocimientos de informática.

Interficie de usuario

- El sistema ha de ser robusto, protegiendo al usuario de cometer errores causados por un uso incorrecto.
- Los periféricos de entrada/salida utilizados por el usuario, únicamente son: teclado, monitor y ratón.

Tratamiento de errores

- El sistema ha responder con avisos e indicaciones al usuario cada vez que se produzca un error.

Interficie del sistema

- El sistema ha de interaccionar con el sistema Operativo Windows, con toda la gama de Windows

Factores de calidad

- La naturaleza de este sistema implica la *portabilidad* en diferentes entornos.
- El sistema ha de ser totalmente fiable, de manera que los usuarios deban tener confianza en su utilización.
- El factor de calidad que menos ha de soportar, es la tolerancia a fallos, se debe de tener en cuenta que muchas veces los archivos pueden estar dañados.

Modificabilidad

- El sistema ha de ser flexible, pues ha de permitir ampliar sus funcionalidades.
- El sistema ha de ser fácil de mantener, es decir que debe existir la posibilidad de añadir mejoras de funcionamiento después de detectar errores.

Seguridad

- No se requiere ningún mecanismo de seguridad en este proyecto.

Requerimientos funcionales

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

Por lo tanto el sistema está dividido en tres partes diferentes, la ejecución de estas partes se hace de manera secuencial, dando lugar a nuestra aplicación.

- Examinar archivos
- Extraer ficheros
- Abrir ficheros.
- Examinar directorios
- Crear ACA

Examinar ficheros

Se encarga de mostrar un explorador de archivos para que puedas seleccionar el archivo .aca que queremos leer.

Extraer ficheros

Se encarga que dado un archivo .aca el cual contiene ficheros, extraer todos los ficheros en una lista de la aplicación y guardarlos en disco, para ello leerá la clase header incluida dentro del archivo .aca que queremos leer y obtendrá el número de ficheros que contiene el archivo .aca, y por último para poder extraerlos leerá de cada uno de los pointers, los datos relacionados con cada uno de los ficheros a extraer.

Abrir ficheros

Se encarga de abrir un fichero en la aplicación asociada a la extensión del propio nombre del fichero. Para ello el fichero tendrá que haber sido seleccionado por el usuario que interactúa con el sistema

.

Examinar ficheros

Se encarga de mostrar un explorador de directorios o carpetas para que puedas seleccionar la carpeta que contiene los ficheros que queremos introducir dentro del nuevo archivo .aca.

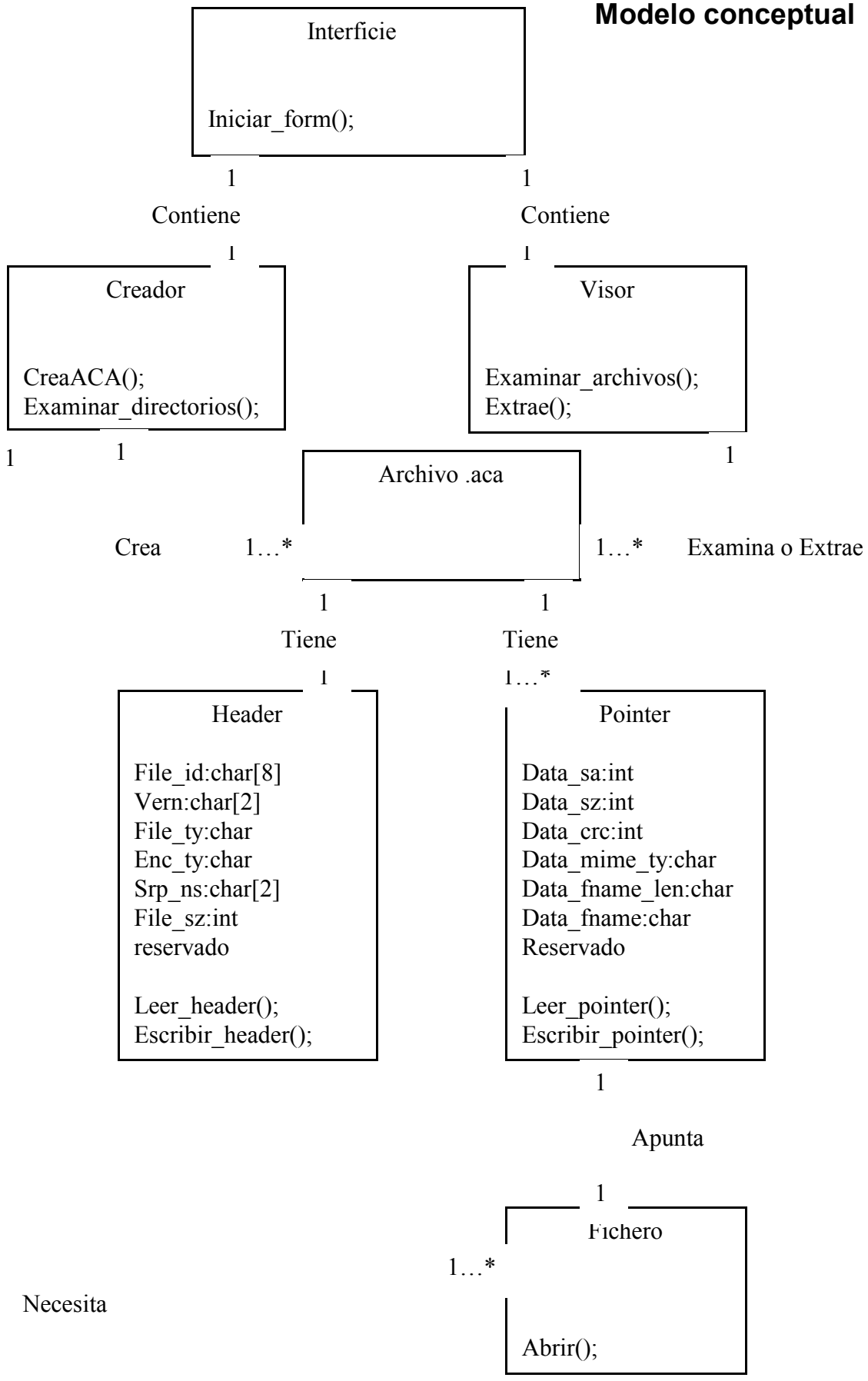
Crear ACA

Se encarga de crear un archivo .aca nuevo y guardarlo en el disco. Para ello el sistema

utilizará la ruta del directorio seleccionada por el usuario, generará un header con los datos extraídos de los ficheros incluidos en el directorio, generará un pointer por cada uno de los ficheros del directorio seleccionado, y por último introducirá todo el contenido de los diferentes ficheros del directorio donde indique el DATA_SA de su respectivo pointer.

ESPECIFICACIÓN

Modelo conceptual



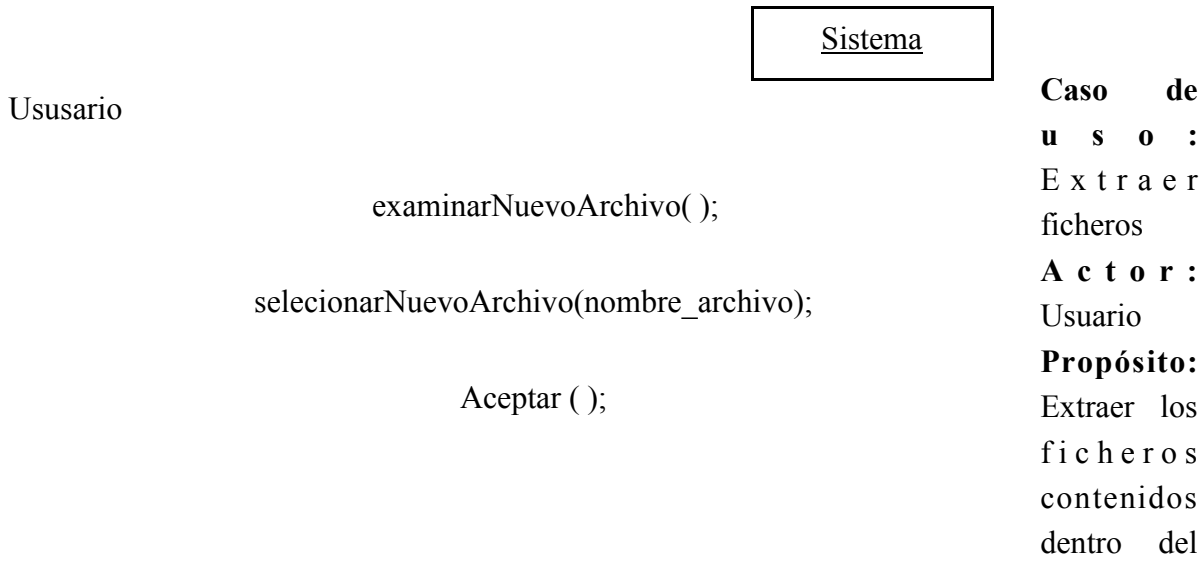
Casos de uso del sistema

Actores

Tan solo hay un actor. Que es el usuario, el cual se encarga de interactuar con el sistema directamente con el sistema. Este, se ocupa de examinar y crear archivos .aca, extraer o abrir los diferentes ficheros contenidos en el archivo .aca.

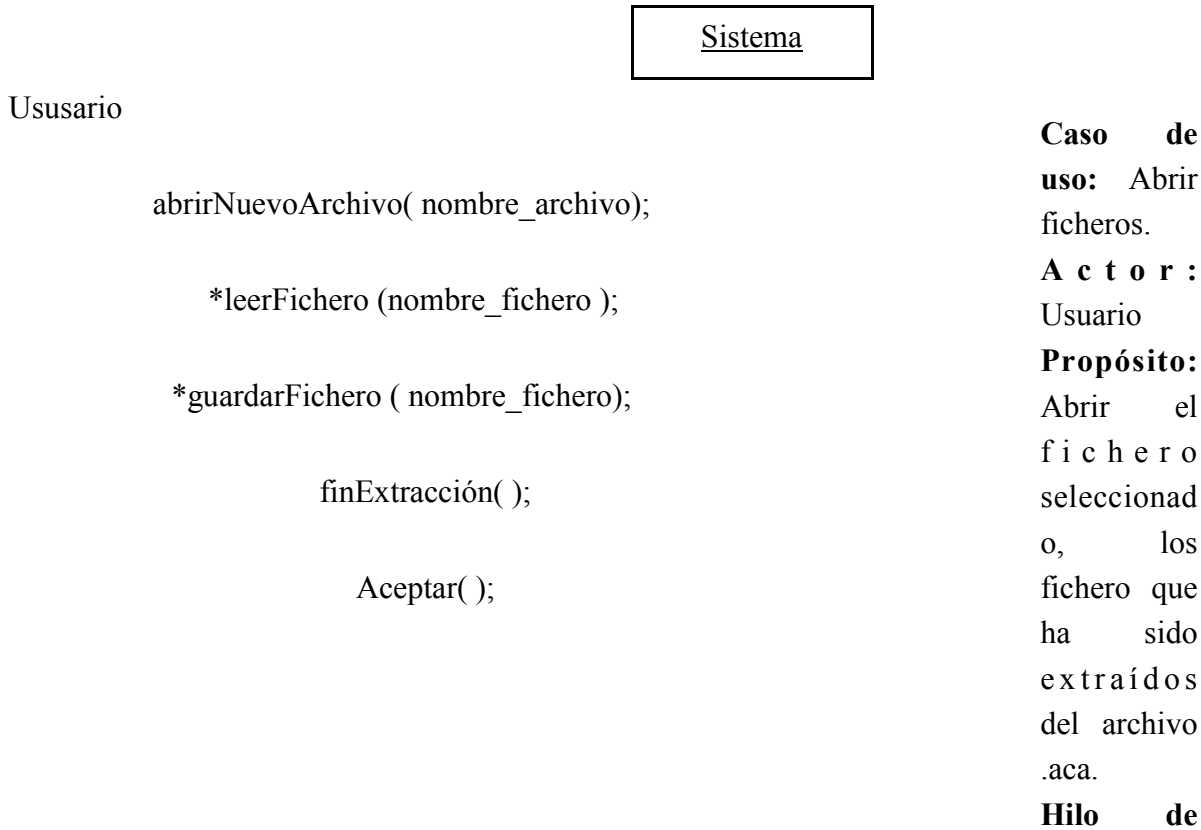
Casos de uso del sistema

Caso de uso: Examinar archivo .aca
Actor: Usuario
Propósito: Observar y obtener la ruta del lugar donde se encuentra el fichero .aca.
Hilo de ejecución: Comprobar la validez del archivo .aca. El usuario tendrá que seleccionar el archivo .aca que quiere visualizar.
Explicación: En primer lugar, el usuario seleccionará el archivo .aca que desea examinar. En segundo lugar, el sistema comprobará si la extensión del archivo es la correcta, es decir si realmente es un archivo .aca. Y en último lugar, el sistema guardará la dirección de la ruta donde se encuentra el archivo .aca seleccionado en un cuadro de testo de la aplicación, la cual se utilizará para las siguientes funciones a realizar.



Caso de uso: Extraer ficheros
Actor: Usuario
Propósito: Extraer los ficheros contenidos dentro del archivo .aca.
Hilo de ejecución: Abre y lee el archivo .aca, y extrae todos los ficheros contenidos dentro del archivo .aca.
Explicación: En primer lugar, abrirá el archivo .aca. En segundo lugar, leerá la clase header

contenida dentro del archivo .aca, de la cual extraerá el número de ficheros que hay dentro del archivo .aca. En tercer lugar, leerá la clase pointer contenida dentro del archivo .aca, de la cual extraerá cada uno de los nombres de los ficheros contenidos en el archivo .aca, las diferentes posiciones donde se encuentra la información de los ficheros y el tamaño de los ficheros. En cuarto lugar, guardará cada uno de los fichero en disco. Y en último lugar, mostrará en una lista del formulario ordenada alfabéticamente los diferentes ficheros contenidos dentro del archivo .aca.



Caso de uso: Abrir ficheros.
Actor: Usuario
Propósito: Abrir el fichero seleccionado, los fichero que ha sido extraídos del archivo .aca.
Hilo de ejecución: Abre en una ventana nueva el fichero seleccionado en la aplicación asociada a su extensión.
Explicación: En primer lugar, el usuario seleccionará un fichero de la lista donde se encuentran los ficheros extraídos del archivo .aca. En segundo lugar, el usuario dará la orden al sistema a través de la interfície de abrir el fichero seleccionado. Y por último el sistema se encargará de abrir el fichero seleccionado en la aplicación asociada a la extensión especificada en su nombre.

Ususario

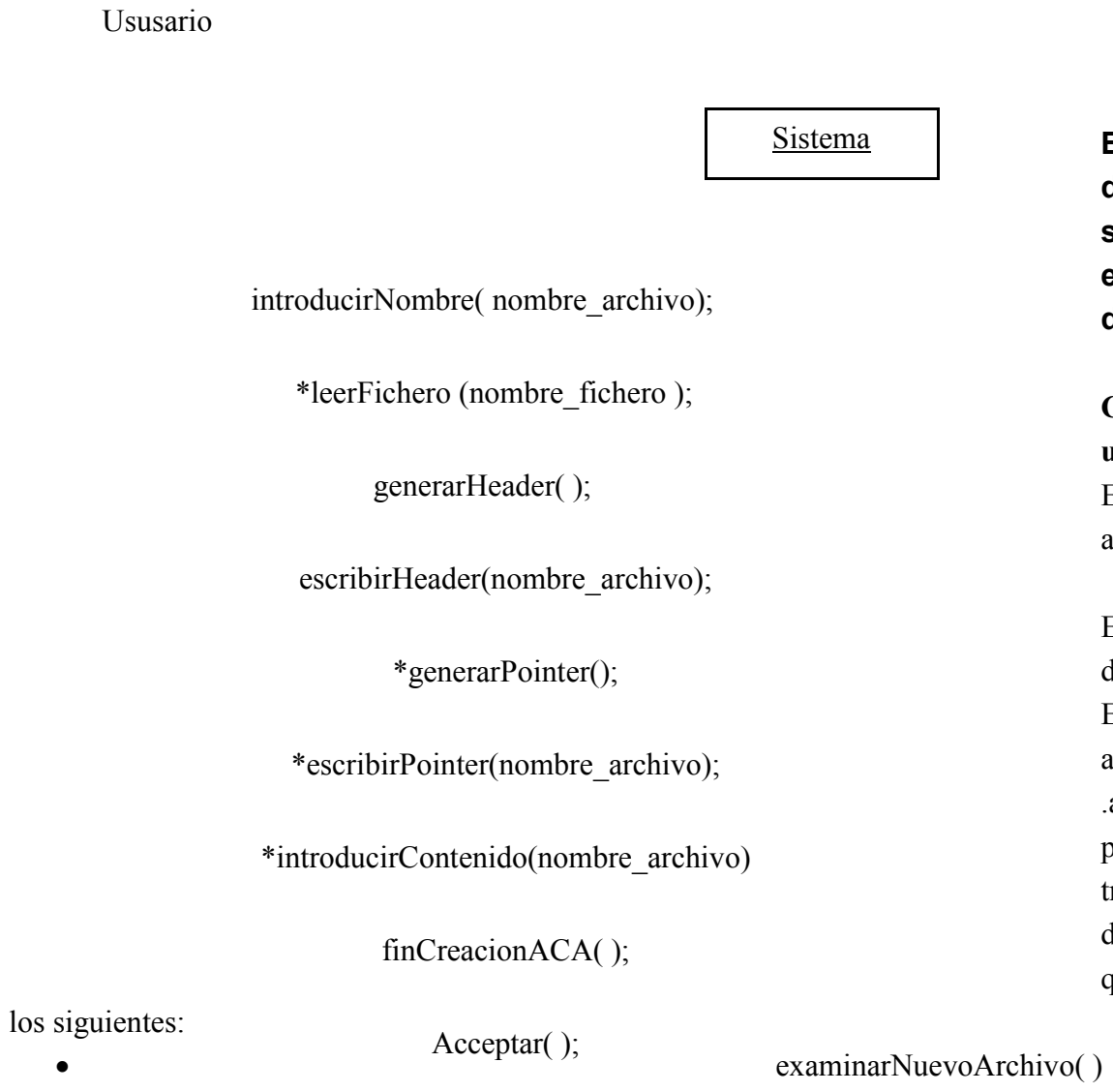
Sistema

seleccionarFichero();

abrirFichero(nombreFichero);

Caso de uso: Examinar directorio.
Actor: Usuario
Propósito: Obtener la ruta del lugar donde se encuentra el directorio, el cual contiene los ficheros a introducir en el nuevo archivo .aca.
Hilo de ejecución: Comprobar que el directorio contenga ficheros. El usuario tendrá que seleccionar el directorio donde se encuentren los ficheros a introducir dentro del nuevo archivo .aca.
Explicación: En primer lugar, el usuario seleccionará el directorio que desea examinar. En segundo lugar, el sistema comprobará si el directorio seleccionado contiene al menos alguna archivo a introducir en el nuevo archivo .aca. Y en último lugar, el sistema guardará la dirección de la ruta donde se encuentra los ficheros a introducir en el nuevo archivo .aca en un cuadro de testo de la aplicación, la cual se utilizará para las siguientes funciones a realizar.

Caso de uso: Crear ACA
Actor: Usuario
Propósito: Dados uno o varios ficheros y un nombre de archivo .aca crear un nuevo archivo .aca.
Hilo de ejecución: Introduce el header, los diferentes pointers y el contenido de cada uno de los ficheros, ya que son elementos imprescindibles para un archivo .aca.
Explicación: En primer lugar, abrirá cada uno de los ficheros. En segundo lugar, leerá cada un de los ficheros y extraerá los datos necesarios para la clase header. En tercer lugar, escribirá la clase header en el nuevo archivo .aca. En cuarto lugar, extraerá y calculará los datos para cada uno de los pointers (en el archivo .aca habrán tantos pointers como número de ficheros a introducir en el nuevo archivo .aca). En quinto lugar, escribirá todos los pointers generados en el nuevo archivo .aca. Y en último lugar, introducirá el contenido de cada uno de los ficheros a introducir en el nuevo fichero .aca, en la posición del fichero marcada en los diferentes pointers ya introducidos.



Eventos del sistema en casos de uso

Caso de uso: Examinar archivo .aca

En el caso de uso Examinar archivo .aca, se producen tres eventos de sistema, que serán los siguientes:

- examinarNuevoArchivo( )
- seleccionarNuevoArchivo(nombre_archivo)
- aceptar()

Nombre evento: examinarNuevoArchivo( )

Parámetros: ninguno.

PRE: ninguna.

POST:

Nombre evento: seleccionarNuevoArchivo(nombre_archivo)

Parámetros: nombre_archivo: string

PRE: El nombre del archivo tiene extensión .aca y el nombre del archivo tiene que existir.

POST:

(1) Se ha seleccionado una archivo con extensión .aca
(2) Se ha obtenido la ruta donde se encuentra el archivo

Nombre evento: aceptar()

Parámetros: ninguno

PRE: Existe un archivo .aca seleccionado.

POST:

(1) Se ha comprobado la validez del archivo .aca
(2) Se ha guardado la ruta donde se encuentra el archivo en el cuadro de texto asignado.

Caso de uso: Extraer ficheros

En el caso de uso Extraer ficheros, se producen cinco eventos de sistema, que serán los siguientes:

- abrirNuevoArchivo(nombre_archivo)
- leerFichero(nombre_fichero)
- guardarFichero(nombre_fichero)
- finExtracción()
- aceptar()

Nombre evento: abrirNuevoArchivo(nombre_archivo)

Parámetros: nombre_archivo: string.

PRE: Existe el archivo y cuadro de texto diferente de vacio

POST:

(1) Se ha comprabado que la procedencia del archivo
(2) Se ha comprobado que le cuadro de texto no este vacio
(3) Se ha abierto el archivo indicado en el cuadro de texto

Nombre evento: leerFichero(nombre_fichero)

Parámetros: nombre_fichero: string

PRE: El nombre del fichero esta contenido en el archivo .aca

POST:

(1) Se ha leido de la clase header incluida en el archivo .aca el indicador que indica la cantidad de ficheros contenido en el archivo .aca
(2) Se ha leido la cantidad de ficheros indicada en el header.

Nombre evento: guardarFichero(nombre_fichero)

Parámetros: nombre_fichero: string

PRE: Existe nombre_fichero.

POST:

(1) Se ha introducido el contenido del fichero en su fichero.
(2) Se ha guardado todos los ficheros leídos en disco.
(3) Se ha extraído todos los ficheros a la lista de la aplicación.

Nombre evento: finExtracción()

Parámetros: ninguno.

PRE: la lista contiene los ficheros extraídos y los ficheros han sido guardados en disco

POST:

(1) El sistema ha lanzado un mensaje de confirmación, que indica que los ficheros han sido extraídos con éxito.

Nombre evento: aceptar()

Parámetros: ninguno.

PRE: el sistema ha lanzado un mensaje de confirmación.

POST:

(1) Se ha aceptado el mensaje de confirmación.

Caso de uso: Abrir ficheros.

En el caso de uso Abrir ficheros, se producen dos eventos de sistema, que serán los siguientes:

- SeleccionarFichero()
- abrirFichero(nombre_fichero)

Nombre evento: SeleccionarFichero()

Parámetros: ninguno.

PRE: la lista de aplicación contiene los ficheros extraídos.

POST:

(1) Se ha seleccionado un fichero de la lista

Nombre evento: abrirFichero()

Parámetros: nombre_fichero: string.

PRE: Existe un fichero seleccionado en la lista de la aplicación.

POST:

(1) Se ha comprobado que existe el fichero seleccionado en disco.
(2) Se ha abierto el fichero en una ventana nueva en la aplicación asociada a la extensión del nombre del fichero.

Caso de uso: Examinar directorio.

En el caso de uso Examinar directorio, se producen tres eventos de sistema, que serán los siguientes:

- examinarNuevoDirectorio( )
- seleccionarNuevoDirectorio(nombre_directorio)
- Aceptar( )

Nombre evento: examinarNuevoDirectorio( )

Parámetros: ninguno.

PRE: ninguna.

POST:

(1) Se ha abierto una nueva ventana donde inicializar la búsqueda del directorio que queremos seleccionar.

Nombre evento: seleccionarNuevoDirectorio(nombre_Directorio)

Parámetros: nombre_Directorio: string

PRE: ninguna.

POST:

(1) Se ha comprobado que existe el directorio y que el directorio no está vacio.
(2) Se ha seleccionado un directorio.
(3) Se ha obtenido la ruta donde se encuentra el directorio

Nombre evento: aceptar()

Parámetros: ninguno

PRE: Existe un directorio seleccionado.

POST:

(1) Se ha comprobado la validez del directorio
(2) Se ha guardado la ruta donde se encuentra el directorio en el cuadro de texto asignado.

Caso de uso: Crear ACA

En el caso de uso Crear ACA, se producen nueve eventos de sistema, que serán los siguientes:

- introducirNombre(nombre_archivo)
- leerFichero(nombre_fichero)
- generarHeader( )

- escribirHeader(nombre_archivo)
- generarPointer( )
- escribirPointer(nombre_archivo)
- introducirContenido(nombre_archivo)
- finCreaciónACA( )
- aceptar()

Nombre evento: introducirNombre(nombre_archivo)

Parámetros: nombre_archivo: string.

PRE: ninguna.

POST:

(1) Se ha comprobado que el archivo tenga la extensión .aca

Nombre evento: leerFichero(nombre_fichero)

Parámetros: nombre_fichero: string

PRE: El nombre del fichero esta contenido en el archivo .aca

POST:

(1) Se ha leido enumerados los diferentes ficheros contenidos en el directorio seleccionado.
(2) Se han abierto y leido los diferentes ficheros.

Nombre evento: generarHeader(nombre_archivo)

Parámetros: nombre_archivo: string

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto.

POST:

(1) Se ha extraído los datos necesarios de los ficheros.
(2) Se ha comprobado que la clase header aceptara los datos extraídos.

Nombre evento: escribirHeader(nombre_archivo)

Parámetros: nombre_archivo: string

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto y que se haya generado el header.
POST:

(1) Se ha introducido el contenido del header en el nuevo archivo .aca.

Nombre evento: generarPointer(nombre_archivo)

Parámetros: nombre_archivo: string

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto.

POST:

(1) Se ha extraído los datos necesarios para cada uno de los ficheros, cantidad de ficheros indicada en el header del archivo .aca
(2) Se ha comprobado que la clase pointer aceptara los datos extraídos.

Nombre evento: escribirPointer(nombre_archivo)

Parámetros: nombre_archivo: string

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto y que se haya generado cada uno de los pointers.
POST:

(1) Se ha introducido el contenido de todos los pointers en el nuevo archivo .aca.
(2) Se mantiene la pre.

Nombre evento: introducirContenido(nombre_fichero)

Parámetros: nombre_fichero: string

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto y fichero distinto de vacío.
POST:

(1) Se ha introducido el contenido de todos los ficheros.

Nombre evento: finCreaciónACA()

Parámetros: ninguno.

PRE: El nuevo archivo ACA se ha creado con éxito y guardado en disco.

POST:

(1) El sistema ha lanzado un mensaje de confirmación, que indica que el archivo ha sido creado con éxito.

Nombre evento: aceptar()

Parámetros: ninguno.

PRE: el sistema ha lanzado un mensaje de confirmación.

POST:

(1) Se ha aceptado el mensaje de confirmación.

Diagramas de interacción

Como estamos en la fase de análisis i la descripción de los casos de uso así como la descripción de los eventos de sistema a cada caso de uso son lo suficientemente claros, hacer los diagramas de interacción seria repetir información así que hemos decidido no hacer ningún diagrama de interacción ya que no aportarían información nueva.

Restricciones no contempladas en el modelo conceptual

R1. Solo se puede extraer los ficheros de un archivo .aca al mismo tiempo, es decir, no podemos extraer los ficheros de dos archivos .aca diferentes al mismo tiempo.

R2. Solo se puede abrir uno de los ficheros de la lista de la aplicación al mismo tiempo, es decir no podemos seleccionar al mismo tiempo dos ficheros de la lista, y llamar a la función abrir.

R3. Tan solo se puede examinar un archivo .aca al mismo tiempo.

R4. Solo se puede crear un archivo .aca al mismo tiempo

R5. Cada fichero contenido dentro del archivo .aca tiene un puntero de búsqueda de tas asignado.

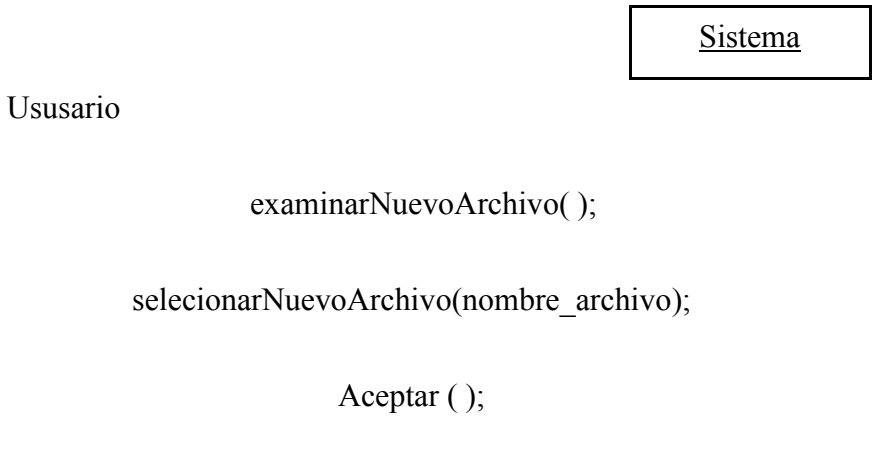
DISEÑO

2:00 2:00

Diseño

Diseño de casos de uso usando principios GRASP

Examinar archivo .aca



Evento: examinarNuevoArchivo( )

Contrato:

PRE: La aplicación esta inicializada.

POST:

(1) Se ha abierto una nueva ventana donde inicializar la búsqueda del archivo .aca que queremos seleccionar.
(2) Se mantiene la pre.

Asignación de responsabilidades

Por el principio de controlador hemos de asignar la responsabilidad de recibir y distribuir cada evento de sistema del caso de uso Examinar archivos.

Aunque haya dos componentes; :Creador y :Visor, no es suficientemente general para representar el sistema global (controlador de fachada); a más a más, si escogemos este como un controlador provocaría alto acoplamiento y baja cohesión. Por estos motivos hemos escogidos un controlador de caso de uso que represente todo el sistema :Interficie.

Evento: selecionarNuevoArchivo( )

Contrato:

PRE: El nombre del archivo tiene la extensión .aca y el nombre del archivo ha de existir

POST:

(1) Se ha seleccionado un archivo con extensión .aca

(2) Se ha creado y obtenido la ruta donde se encuentra el archivo.

Asignación de responsabilidades

Por el principio grasp del experto, el lector crea una ruta donde se encuentra el archivo. He optado por crear un objeto ruta que será creado en el momento de inicializar el sistema.

Evento: Aceptar( )

Contrato:

PRE: Existe un archivo .aca seleccionado

POST:

(1) Se ha comprobado la validez del archivo .aca

(2) Se ha guardado la ruta donde se encuentra el archivo, en el cuadro de texto asignado.

Asignación de responsabilidades

Por el principio del experto, el lector introducirá en el cuadro de texto la ruta obtenida del archivo.

Extraer ficheros

Evento: abrirNuevoArchivo(nombre_archivo)

Contrato:

PRE: Existe el archivo y cuadro de texto diferente de vacío

POST:

(1) Se ha comprobado que la procedencia del archivo
(2) Se ha comprobado que le cuadro de texto no este vacío
(3) Se ha abierto el archivo indicado en el cuadro de texto

Asignación de responsabilidades

Por el principio del experto, el lector abrirá el archivo indicado con su ruta correspondiente.

Evento: leerFichero(nombre_fichero)

Contrato:

PRE: El nombre del fichero está contenido en el archivo .aca

POST:

(1) Se ha leido de la clase header incluida en el archivo .aca el indicador que indica la cantidad de ficheros contenido en el archivo .aca
(2) Se ha leido la cantidad de ficheros indicada en el header.
(3) Se mantiene la PRE

Asignación de responsabilidades

Para poder leer un fichero necesitamos saber si el fichero está incluido o no el archivo .aca, para ello comprobaremos que hay un puntero que apunta a este fichero, y para poder leerlo necesitamos encontrar el fichero y por eso buscamos en el multiconjunto donde se encuentran los ficheros.

Evento: guardarFichero(nombre_fichero)

Contrato:

PRE: Existe nombre_fichero.

POST:

(1) Se ha introducido el contenido del fichero en su fichero.

(2) Se ha guardado todos los ficheros leídos en disco.

(3) Se ha extraído todos los ficheros a la lista de la aplicación.

(4) Se mantiene la pre

Asignación de responsabilidades

En primer lugar buscaremos el fichero que queremos guardar haremos las oportunas comprobaciones como el caso anterior, y por último lugar lo guardaremos en disco.

He optado por crear un objeto disco y un objeto lista que será creado en el momento de inicializar el sistema, y donde iremos guardando y añadiendo respectivamente los ficheros.

Evento: finExtracción()

Contrato:

PRE: la lista contiene los ficheros extraídos y los ficheros han sido guardados en disco

POST:

(1) El sistema ha lanzado un mensaje de confirmación, que indica que los ficheros han sido extraídos con éxito

Asignación de responsabilidades

El sistema buscará el número de ficheros que contenía el archivo y lo comparará, con el número total de ficheros de la lista, si son iguales lanzará un mensaje donde se confirma del éxito de la operación.

Evento: aceptar()

Contrato:

PRE: el sistema ha lanzado un mensaje de confirmación.

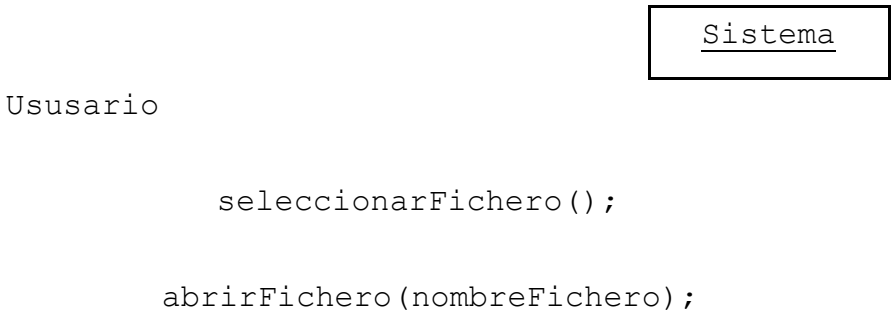
POST:

(1) Se ha aceptado el mensaje de confirmación.

Asignación de responsabilidades

El sistema recibe la aceptación del mensaje de confirmación.

Abrir fichero



E v e n t o :
Seleccionar Fichero()

Contrato:

PRE: la

lista de aplicación contiene los ficheros extraídos.

POST:

(1) Se ha seleccionado un fichero de la lista

Asignación de responsabilidades

Seleccionamos de las lista el fichero que queremos abrir después. La lista contiene los ficheros extraídos del archivo .aca

Evento: abrirFichero()

Contrato:

PRE: Existe un fichero seleccionado en la lista de la aplicación.

POST:

(1) Se ha comprobado que existe el fichero seleccionado en disco.

(2) Se ha abierto el fichero en una ventana nueva en la aplicación asociada a la extensión del nombre del fichero.

Asignación de responsabilidades

Buscamos el fichero seleccionado de la lista, y lo abrimos. Por el principio del experto, la lista obtendrá el fichero seleccionado.

Examinar directorio

Sistema

Ususario

examinarNuevoDirectorio( );

seleccionarNuevoDirectorio(nombre_directorio);

Aceptar ( );

E v e n t o :
examinarNuevoDirectorio( )

Contrato:

PRE: La aplicación esta inicializada.

POST:

(1) Se ha abierto una nueva ventana donde inicializar la búsqueda del directorio que queremos seleccionar..
(2) Se mantiene la pre

Asignación de responsabilidades

Examinaremos un directorio donde estarán todos los ficheros a incluir dentro del nuevo archivo .aca.

Por el principio del experto, el creador obtendrá el directorio examinado.

Crearemos un multiconjunto de directorio donde obtendremos los diferentes directorios.

Evento: seleccionarNuevoDirectorio(nombre_Directorio)

Contrato:

PRE: La aplicación esta inicializada.

POST:

(1) Se ha comprobado que existe el directorio y que el directorio no está vacio.
(2) Se ha seleccionado un directorio.
(3) Se ha obtenido la ruta donde se encuentra el directorio
(4) Se mantiene la pre

Asignación de responsabilidades

Por el principio grasp del experto, el creador obtiene la ruta del lugar donde se encuentra el directorio. He optado por crear un objeto ruta que será creado en el momento de inicializar el sistema.

Evento: aceptar()

Contrato:

PRE: Existe un directorio seleccionado.

POST:

(1) Se ha comprobado la validez del directorio
(2) Se ha guardado la ruta donde se encuentra el directorio en el cuadro de texto asignado

Asignación de responsabilidades

Por el principio del experto, el creador introducirá en el cuadro de texto la ruta obtenida de archivo.

Crear ACA

Evento: introducirNombre(nombre_archivo)

Contrato:

PRE: La aplicación esta inicializada.

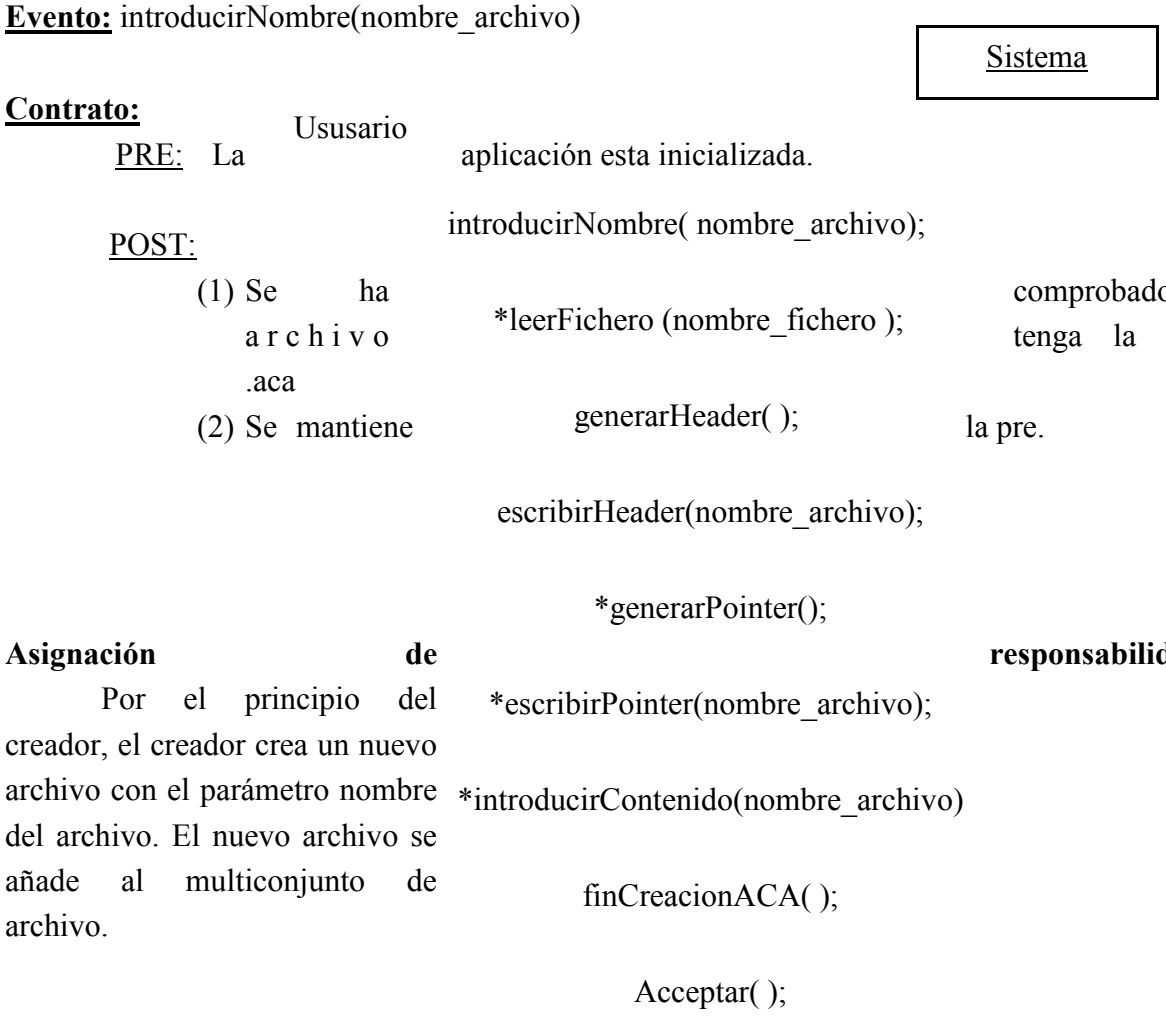
POST:

(1) Se ha comprobado que el archivo tenga la extensión .aca

(2) Se mantiene la pre.

Asignación de responsabilidades

Por el principio del creador, el creador crea un nuevo archivo con el parámetro nombre del archivo. El nuevo archivo se añade al multiconjunto de archivo.



Evento: leerFichero(nombre_fichero)

Contrato:

PRE: El nombre del fichero esta contenido en el archivo .aca

POST:

(1) Se ha leido enumerados los diferentes ficheros contenidos en el directorio seleccionado.

(2) Se han abierto y leido los diferentes ficheros.

Asignación de responsabilidades

Por el principio del experto, el creador tiene la información del fichero necesaria.

Buscamos el fichero ha introducir en el archivo .aca y lo leemos.

Evento: generarHeader(nombre_archivo)

Contrato:

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto.

POST:

(1) Se ha extraído los datos necesarios de los ficheros.

(2) Se ha comprobado que la clase header aceptara los datos extraídos.

(3) Se mantiene la pre.

Asignación de responsabilidades

Buscamos el archivo .aca que vamos a crear y generamos la primera parte de su estructura, es decir, generamos el header.

Evento: escribirHeader(nombre_archivo)

Contrato:

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto y que se haya generado el header.

POST:

(1) Se ha introducido el contenido del header en el nuevo archivo .aca.

Asignación de responsabilidades

Buscamos el nombre del archivo donde queremos introducir el header, buscamos el header generado para ese archivo, y por último lo añadimos al archivo que estamos creando.

Evento: generarPointer(nombre_archivo)

Contrato:

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto.

POST:

(1) Se ha extraído los datos necesarios para cada uno de los ficheros, cantidad de ficheros indicada en el header del archivo .aca
(2) Se ha comprobado que la clase pointer aceptara los datos extraídos.
(3) Se mantiene la pre

Asignación de responsabilidades

Buscamos el archivo .aca que vamos a crear y generamos la parte de la estructura reservada para los diferentes pointers, es decir, generamos cadauno de los pointers.

Evento: escribirPointer(nombre_archivo)

Contrato:

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto y que se haya generado cada uno de los pointers.

POST:

(1) Se ha introducido el contenido de todos los pointers en el nuevo archivo .aca.

(2) Se mantiene la pre.

Asignación de responsabilidades

Buscamos el nombre del archivo donde queremos introducir los diferentes pointers, cogemos los pointers generados para ese archivo, y por último los añadimos al archivo que estamos creando.

Evento: introducirContenido(nombre_fichero)

Contrato:

PRE: nombre_archivo es igual al nombre introducido en el cuadro de texto y fichero distinto de vacío.

POST:

(1) Se ha introducido el contenido de todos los ficheros.

Asignación de responsabilidades

Buscamos cada uno de los ficheros que queremos introducir en el nuevo archivo .aca, y luego añadimos su contenido en el archivo a crear.

Evento: finCreaciónACA()

Contrato:

PRE: El nuevo archivo ACA se ha creado con éxito y guardado en disco.

POST:

(1) El sistema ha lanzado un mensaje de confirmación, que indica que el archivo ha sido creado con éxito.

Asignación de responsabilidades

Buscamos el archivo en disco, si el archivo esta en disco sale un mensaje confirmando que se ha realizado con éxito el nuevo archivo .aca.

Evento: aceptar()

Contrato:

PRE: el sistema ha lanzado un mensaje de confirmación.

POST:

(1) Se ha aceptado el mensaje de confirmación.

Asignación de responsabilidades

El sistema recibe la aceptación al mensaje de confirmación.

2:00 2:00

2:00 2:00

IMPLEMENTACIÓN

Implementación

Examinar archivo

Código

```
     OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog;
     openFileDialog1->Title = "Seleccionar fichero";
     openFileDialog1->Filter = "Ficheros de texto
(*.aca;*.ini)|*.aca;*.ini" + "|Todos los ficheros
(*.*)|*.*";
     openFileDialog1->FileName = txtFichero->Text;
     if (openFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
     {
         txtFichero->Text = openFileDialog1->FileName;
     }
```

Extraer Ficheros

Código

```
listView1->Items->Clear();
     pin_ptr<const wchar_t> a = PtrToStringChars(txtFichero >
Text);
     ifstream fg(a,ios::binary);
     fg.read(reinterpret_cast<char*>(&headerAI),sizeof(headerA));

     verss=headerAI.VERN[0];
     vers=headerAI.VERN[1];

     if(vers==50 & verss==49){
          pin_ptr<const wchar_t> a =
PtrToStringChars(txtFichero->Text);
          ifstream fg(a,ios::binary);
          fg.read(reinterpret_cast<char*>(&headerAI),
sizeof(headerA));
          i=0;
          avanzar=36;
          ifstream fh(a,ios::binary);
          while(i<headerAI.SRP_Ns){
               fh.seekg(avanzar, ios::beg);
               fh.read(reinterpret_cast<char*>(&pointerI),
sizeof(pointer));
               System::String^ XP(gcnew
String(pointerI.DATA_FNAME));
               listView1->Items->Add(XP);
               pin_ptr<const wchar_t> aa = PtrToStringChars(XP);
               ofstream fs(aa, ios::binary);
               r= pointerI.DATA_SZ;
               t= pointerI.DATA_SA;
```

2:00 2:00

```
                avanzar=avanzar+148:
                ifstream fr(a,ios::binary);
                xx="";
                while(¡ Fr.eof ()){
                     getline(fr,texto);
                     xx = xx + texto + '\n';
                }
                xc=0;
                xb=0;
                xc= xx.find("----",xc);
                xb= xx.find("----",xc+1);
                xb= xb-xc-4;
                resp ="";
                resp = xx.substr(xc+4,xb);

                fs<<resp;
                i=i+1;
                xc=xc+4;
              }
              fh.close();
              fg.close();
            }else{
                pin_ptr<const wchar_t> a =
PtrToStringChars(txtFichero->Text);
                ifstream fg(a,ios::binary);
                fg.read(reinterpret_cast<char*>(&headerII),
sizeof(header));
                i=0;
                avanzar=32;
                ifstream fh(a,ios::binary);
                while(i< headerII.SRP_Ns[1]){
                     fh.seekg(avanzar, ios::beg);
                     fh.read(reinterpret_cast<char*>(&pointerI),
sizeof(pointer));
                     System::String^ XP(gcnew
String(pointerI.DATA_FNAME));
                     listView1->Items->Add(XP);
                     pin_ptr<const wchar_t> aa =
PtrToStringChars(XP);
                     ofstream fs(aa, ios::binary);
                     hh= pointerI.DATA_SZ;
                     qq= (0xFFFF0000 & hh) >> 16;
                     pp= (0x0000FFFF & hh) >> 32;
                     r=rotarI(pp,qq);
                     ll= pointerI.DATA_SA;
                     nn= (0xFFFF0000 & ll) >> 16;
                     mm= (0x0000FFFF & ll) >> 32;
                     t=rotarI(mm,nn);
                avanzar = avanzar + 14+pointerI.DATA_FNAME_LEN+32;

                     nombre="";
```

```
(a,ios::binary);
```

```
                    fr.seekg(t, ios::beg);
                    xx="";
                    while(! Fr.eof ()){
                        getline(fr,texto);
                        xx = xx + texto + '\n';
                    }
                    resp ="";
                    resp = xx.substr(0,r);
                    fs<<resp;
                    i=i+1;
                }
                    fh.close();
                    fg.close();

            }
```

Abrir ficheros

Código

```
String^ archivo;
archivo= listView1->SelectedItems[0]->Text;
Process^ pr = gcnew Process;
pr->StartInfo->FileName = archivo;
pr->Start();
```

Examinar Directorio

Código

```
//abre el explorador y da la ruta de la carpeta seleccionada
if (folderBrowserDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK){
     textBox1->Text = folderBrowserDialog1->SelectedPath;
}
//recorre todos los directorios de la carpeta seleccionada
     DirectoryInfo^ di = gcnew DirectoryInfo( textBox1->Text );
     array<FileInfo^>^fiArr = di->GetFiles();
     Collections::IEnumerator^ myEnum = fiArr->GetEnumerator();
     while ( myEnum->MoveNext() ){
          FileInfo^ fri = safe_cast<FileInfo^>(myEnum->Current);
          len=fri->Name;
//introduce en tb8 cada una de la rutas de los directorios a
reccorrer
          textBox8->Text= (textBox1->Text + "\\" + fri->Name);
//introduce en tb5 la ruta de tb8 + lo que ya habia separado por
comas
          textBox5->Text=(textBox5->Text + "," + textBox8);
```

```
//Guardamos en la varible dt_sz el tamaño de cada uno de los
directorios

     dt_sz = tamano();

//convierte la variable dt_sz en string
          _itoa_s(dt_sz,buffer,65,10);

//crea y abre el fichero data_sz.txt y introduce cada uno de los
tamaños separado por comas
          ofstream fb("data_sz.txt", ofstream::app);
          fb<<buffer<<",";
          fb.close();
//calcula la suma de todos los data_fname_len y lo introduce en la
variable total_len
          total_len = total_len + len->Length;
//calcula el número de directorios que tendremos en el fichero.aca
(SRP_NS)
          cont++;
          dta_sz=dta_sz + tamano();
     }
//calcula el FILE_SZ
          head= 34 + (cont*148) + dta_sz;
//introduce en tb3 la suma total de data_fname_len
          String^ t;
          t = total_len.ToString();
          System::String^ XP(gcnew String(t));
          textBox3->Text=(XP);
//introduce en tb4 SRP_NS
          String^ tt;
          tt = cont.ToString();
          System::String^ XPI(gcnew String(tt));
          textBox4->Text=(XPI);
//convierte la variabe cont tipo int en short
          short ns;
          ns=cont;
//crea y abre prueba.txt y introduce el header
          ofstream ff("prueba.txt", ofstream::app | ios::binary);
          strcpy_s(headerBI.FILE_ID,"HDDVDACA");
          strcpy_s(headerBI.VERN,"12");
          headerBI.FILE_TY='1';
          headerBI.ENC_TY='0';
          headerBI.SRP_Ns=ns;
          headerBI.FILE_SZ=head;
          strcpy_s(headerBI.reserved_header,"                ");
          ff.write(reinterpret_cast<char *>(&headerBI),
sizeof(headerB));
```

2:00 2:00

Crear ACA

Código

```
     const char* chars = (const
char*)(Marshal::StringToHGlobalAnsi(textBox3->Text)).ToPointer();
     dt_sa=atoi(chars);
     const char* charsI = (const
char*)(Marshal::StringToHGlobalAnsi(textBox4->Text)).ToPointer();
     srp=atoi(charsI);
//introducimos en aca el string de textBox5
     aca=textBox5->Text;
//buscamos la primera , a partir de la posicion 1
     iii= aca->IndexOf(",",1);
//extraemos la ruta del primer directorio
     acaI=aca->Substring(1,iii-1);
//ponemos la ruta del primer directorio en el tb6
     textBox6->Text=(acaI);
//len = al data_fname_len del primer directorio
     len=data_fname_lenm();
//mime = al data_mime_ty del primer directorio
     mime=data_mime_ty();

     const char* charsIII = (const
char*)(Marshal::StringToHGlobalAnsi(textBox7->Text)).ToPointer();
//abre data_sz.txt lo recorre entero y guarda todo el contenido en el
string xxx
     ifstream fe("data_sz.txt", ios::binary);
     while(! fe.eof()){
          getline(fe,texto);
          xxx = xxx + texto;
     }
//extrae el tamaño del primer directorio del data_sz.txt
     ii= xxx.find(",");
     xxxx= xxx.substr(0,ii);
//convierte el string extraido en un int
     System::String^ XPpp(gcnew String(xxxx.c_str()));
     const char* charsp= (const
char*)(Marshal::StringToHGlobalAnsi(XPpp)).ToPointer();
     srpp=atoi(charsp);
//crea y abre el fichero prueba.txt y introduce el pointer del primer
directorio
     ofstream fq("prueba.txt",ofstream::app | ios::binary);
     dta_sa= 34 + (srp*148)+2;
     pointerI.DATA_SA=dta_sa;
     pointerI.DATA_SZ=srpp;
     pointerI.DATA_CRC=11111111;
     pointerI.DATA_MIME_TY=mimes;
     pointerI.DATA_FNAME_LEN=lenn;
```

```
(pointerI.DATA_FNAME,charsIII);
```

```
strcpy_s(pointerI.reserved_pointer,"                                        ");
      fq.write(reinterpret_cast<char*>(&pointerI),sizeof(pointer));
      //extrae la lista de directorios
      DirectoryInfo^ dii = gcnew DirectoryInfo( textBox1->Text );
      array<FileInfo^>^fiArrr = dii->GetFiles();
      Collections::IEnumerator^ myEnumm = fiArrr->GetEnumerator();
      w=0;
      while ( myEnumm->MoveNext() ){
            FileInfo^ frii = safe_cast<FileInfo^>(myEnumm->Current);

            if(w>0){

//introduce en prueb1.txt el segundo directorio
            if(rt==0){
                  ofstream fq("prueb1.txt",ofstream::app | ios::binary);
            dta_sa= dta_sa + srpp+1;
            rt++;
            pointerI.DATA_SA=dta_sa;
            ifstream fe("data_sz.txt", ios::binary);
            while(! fe.eof()){
                  getline(fe,texto);
                  xxx = xxx + texto;
            }
            hh=xxx.find(",",ii+1);
            iii=hh;
            hh=hh-ii-1;
            xxxx= xxx.substr(ii+1,hh);
            System::String^ XR(gcnew String(xxxx.c_str()));
            const char* ch= (const
char*)(Marshal::StringToHGlobalAnsi(XR)).ToPointer();
            dt_szzI=atoi(ch);
            mime=0;
            len=0;
            textBox6->Text=(frii->Name);
            mime=data_mime_ty();
            const char* charsIIIII = (const
char*)(Marshal::StringToHGlobalAnsi(frii->Name)).ToPointer();
            string lenm;
            lenm=charsIIIII;
            len=lenm.size();
            lenn=len;
            const char* charsIIII = (const
char*)(Marshal::StringToHGlobalAnsi(frii->Name)).ToPointer();
      pointerI.DATA_SZ=dt_szzI;
            pointerI.DATA_CRC=11111111;
            pointerI.DATA_MIME_TY=mime;
            pointerI.DATA_FNAME_LEN=lenn;
            strcpy_s(pointerI.DATA_FNAME,charsIIII);
            strcpy_s(pointerI.reserved_pointer,"
       ");
```

2:00 2:00

```
fq.write(reinterpret_cast<char *>(&pointerI),
sizeof(pointer));
          fq.close();
     }else{
          ofstream fu("prueb1.txt",ofstream::app |
ios::binary);
//introduce el resto de directorios
          ifstream fe("data_sz.txt", ios::binary);
          dta_sa= dta_sa + dt_szzI;
          pointerI.DATA_SA=dta_sa;
          while(! fe.eof()){
               getline(fe,texto);
               xxx = xxx + texto;
          }
               hh=xxx.find(",",iii+1);
               iiii=hh;
               hh=hh-iii-1;
               xxxx= xxx.substr(iii+1,hh);
               String^ str = gcnew String(xxxx.c_str());
               const char* chars = (const
char*)(Marshal::StringToHGlobalAnsi(str)).ToPointer();
               dt_szzI=atoi(chars);
               mime=0;
               len=0;
               textBox6->Text=(frii->Name);
               mime=data_mime_ty();
               const char* charsIIIII = (const
char*)(Marshal::StringToHGlobalAnsi(frii->Name)).ToPointer();
               string lenm;
               lenm=charsIIIII;
               len=lenm.size();
               char lenn;
               lenn=len;
               const char* charsIIII = (const
char*)(Marshal::StringToHGlobalAnsi(frii->Name)).ToPointer();
               pointerI.DATA_SZ=dt_szzI;
               pointerI.DATA_CRC=11111111;
               pointerI.DATA_MIME_TY=mime;
               pointerI.DATA_FNAME_LEN=lenn;
               strcpy_s(pointerI.DATA_FNAME,charsIIII);
               strcpy_s(pointerI.reserved_pointer,"
                   ");
               fu.write(reinterpret_cast<char *>(&pointerI),
sizeof(pointer));
               fu.close();
               iii=iiii;
          }
       }
               w=w+1;
```

2:00 2:00

```
     pin_ptr<const wchar_t> sa=
PtrToStringChars(textBox2->Text);
     ofstream fxd(sa,ofstream::app);
     ifstream ac("prueba.txt",ios::binary);
     while(! ac.eof()){
          getline(ac,txto);

          fxd<<txto;
     }
     fxd.close();
     ac.close();

     pin_ptr<const wchar_t> sn=
PtrToStringChars(textBox2->Text);
     ofstream fxr(sn,ofstream::app);
     ifstream ad("prueb1.txt",ios::binary);
     while(! ad.eof()){
          getline(ad,txto);
          fxr<<txto;
     }
     fxr<<"----";
     fxr.close();
     ad.close();
     DirectoryInfo^ diii = gcnew DirectoryInfo( textBox1->Text
);
     array<FileInfo^>^fiArrrr = diii->GetFiles();
     Collections::IEnumerator^ myEnummm =
fiArrrr->GetEnumerator();
     while ( myEnummm->MoveNext() ){
          FileInfo^ friii =
safe_cast<FileInfo^>(myEnummm->Current);
          open= textBox1->Text + "\\" + friii->Name;
          pin_ptr<const wchar_t> sr= PtrToStringChars(open);
          pin_ptr<const wchar_t> sd=
PtrToStringChars(textBox2->Text);
          ofstream fxt(sd,ofstream::app );
          ifstream at(sr,ios::binary);
          while(! at.eof()){
               getline(at,txto);
               fxt<<txto<<endl;

          }
          fxt<<"----";
          fxt.close();
          at.close();
     }
     MessageBox::Show( "Error: no se ha podido crear el nuevo
archivo!", "Creacion ACA:",MessageBoxButtons::OK,
MessageBoxIcon::Exclamation );
```

COMPORTAMIENTO EN EJECUCION

Comportamiento en Ejecución

Lector ACA

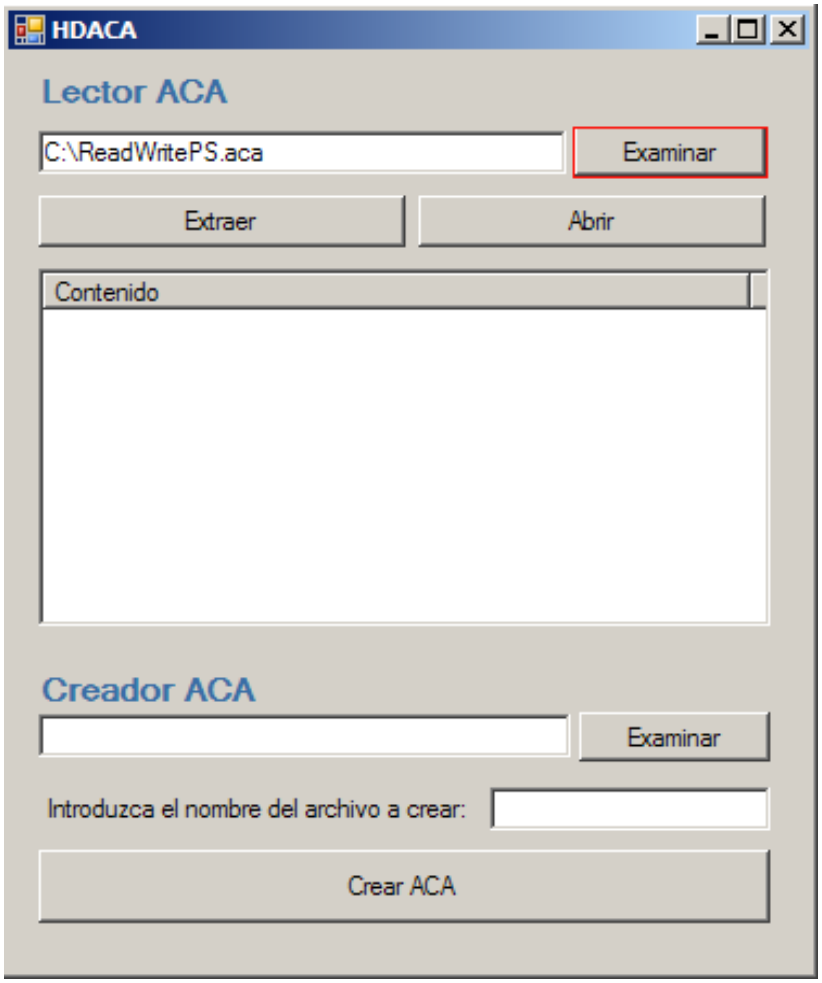


Figura 9.1

Para poder leer un archivo .aca en esta aplicación debemos ir a examinar, donde nos apararecerá la siguiente ventana o formulario. (veasé en la figura 9.2)

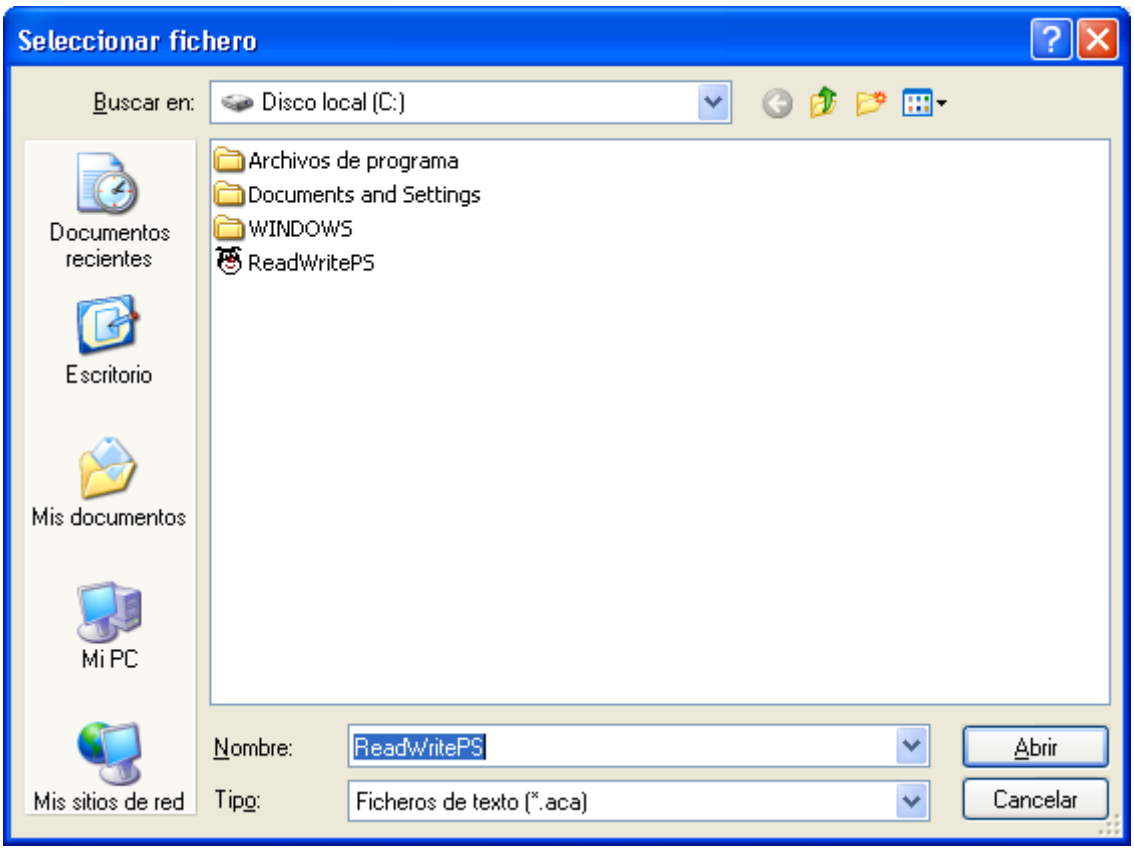


Figura 9.2

En esta ventana buscaremos como en cualquiera de las aplicacaciones que contiene explorador de archivos, el archivo que queremos leer, hay que tener en cuenta que solo funcionará o nos dejará escoger archivo con extensión .aca.

Una vez seleccionado el archivo en el cuadro de texto, situado al lado izquierdo del botón Examinar, aparecerá la ruta del archivo escogido para leer.

El siguiente paso será extraer el contenido o ficheros que contiene el archivo .aca seleccionado. Para ello deberemos de pulsar el botón Extraer como aparece en la figura 9.3

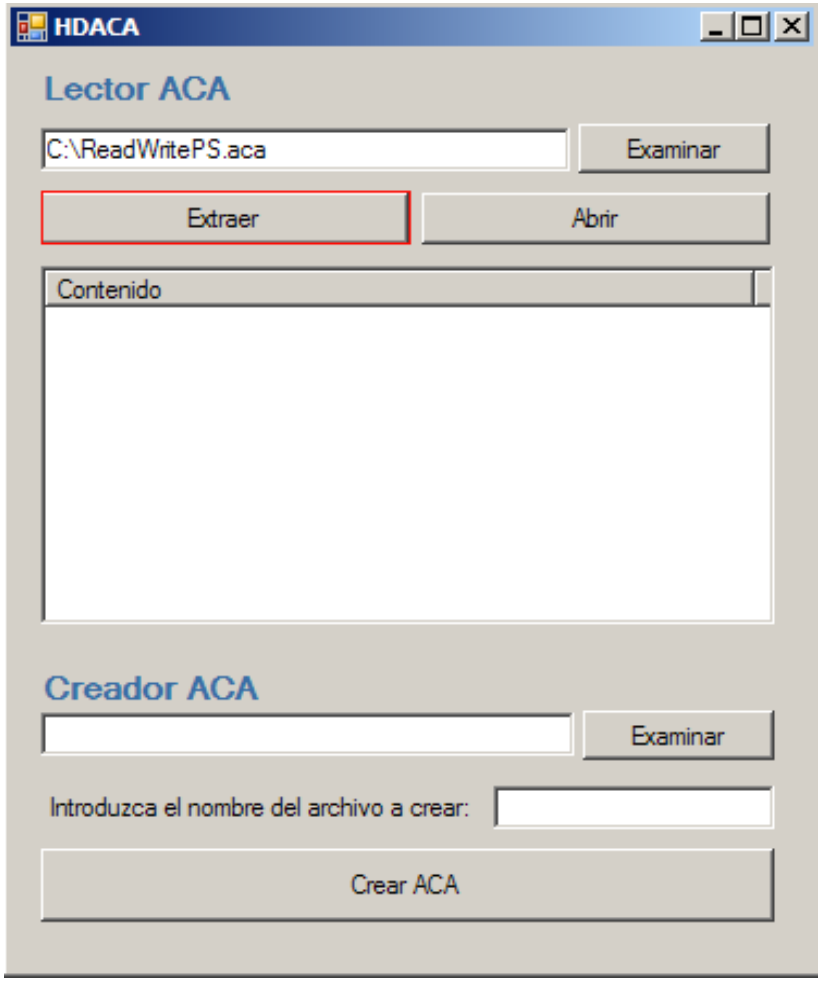


Figura 9.3

Cuando pulsamos el botón extraer aparecen en la lista situada justamente debajo del botón los ficheros contenidos dentro del archivo .aca, como se muestra en la figura 9.4:

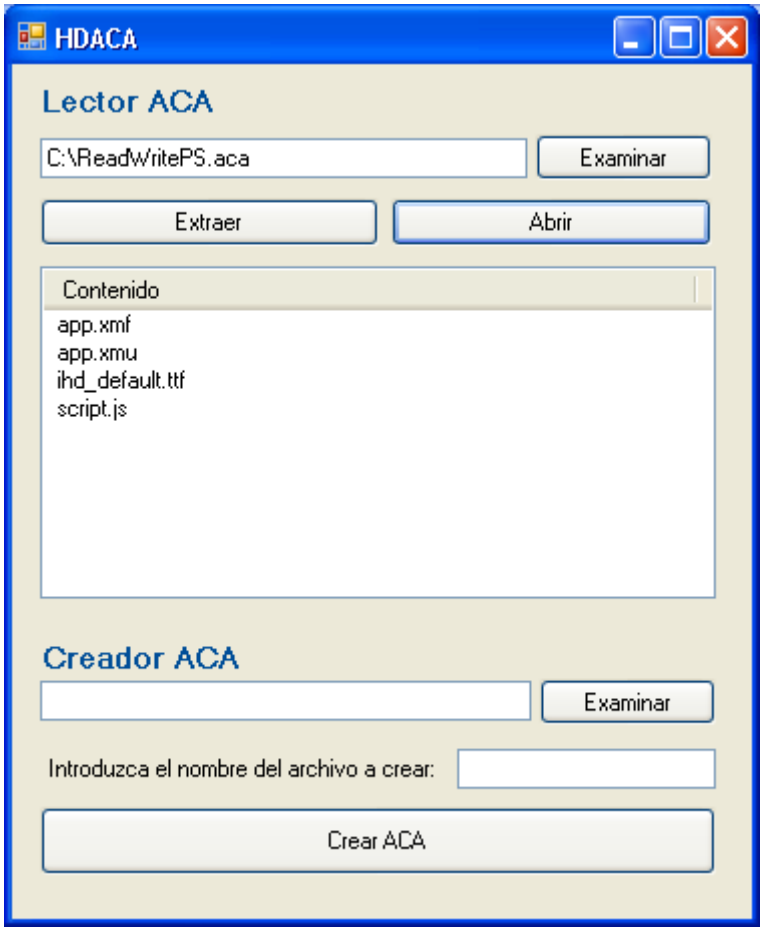


Figura 9.4

Estos ficheros extraídos en la lista que observamos también quedarán guardados en el disco.

Si lo queremos, es abrir alguno de los ficheros que observamos en la lista, lo unico que debemos hacer es seleccionarlos (para seleccionar hay que hacer un clic en el nombre del fichero), y pulsar el botón Abrir como observamos en la figura 9.5:

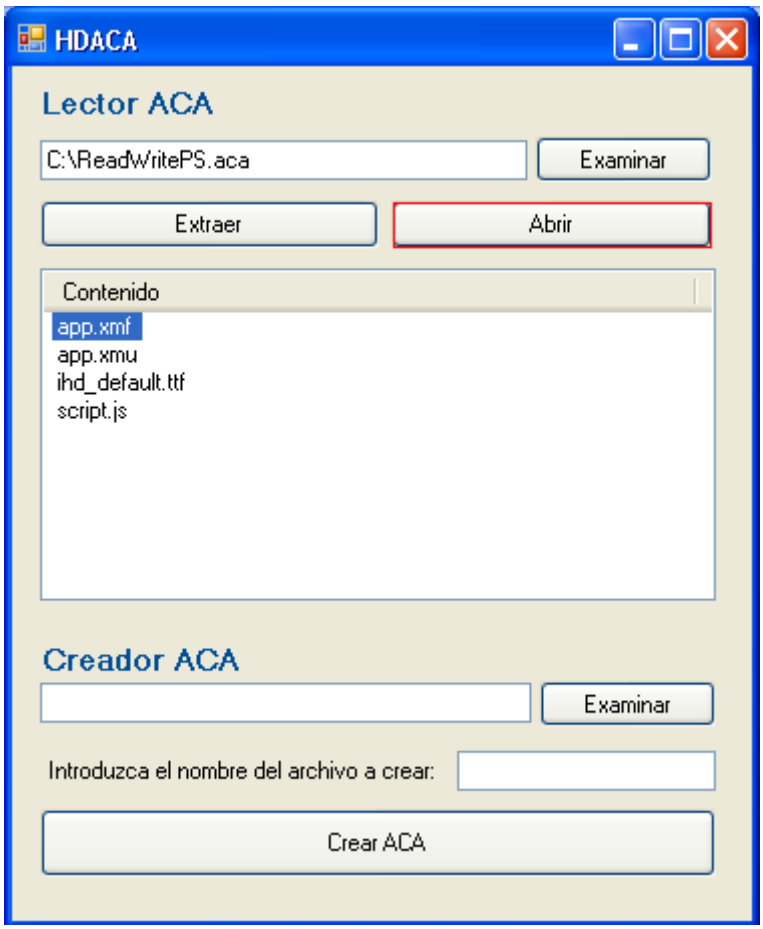


Figura 9.5

Una vez seleccionado el fichero, y después de pulsar abrir aparecerá el fichero abierto en la aplicación asociada a su extension.

Creador ACA

Para crear un nuevo archivo .aca debemos examinar un directorio, en el cual estan contenidos los ficheros que queremos introducir dentro del fichero .aca, para ello mostramos en la

figura 9.6 como examinar un directorio:

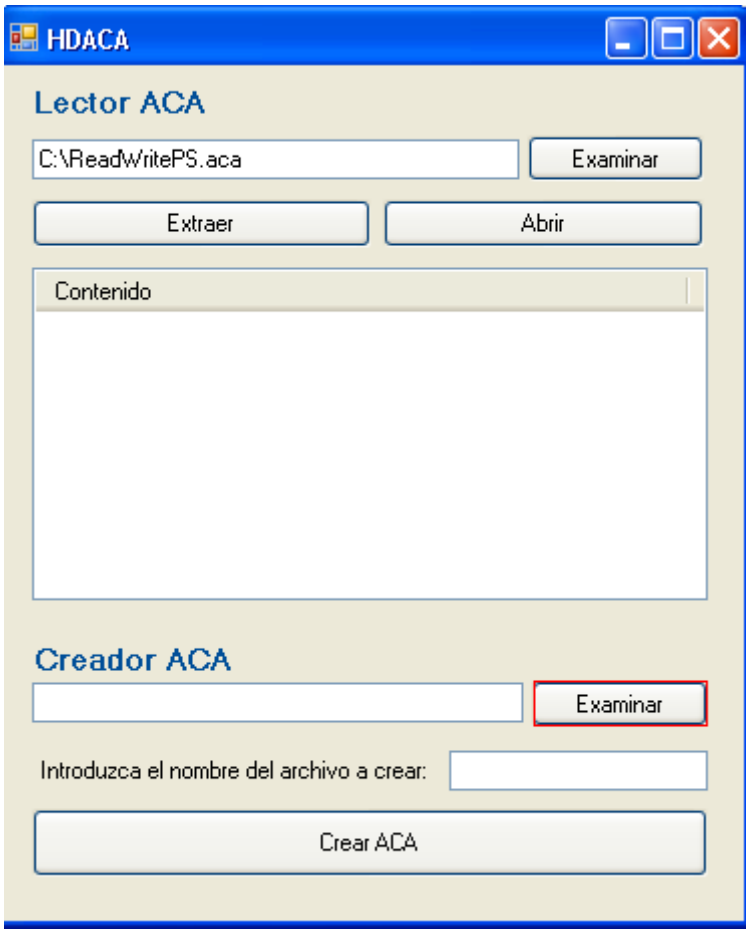


Figura 9.6

Pulsamos el botón Examinar marcado en la figura 9.6, y aparecerá la siguiente ventana (veasé en la figura 9.7), donde debemos seleccionar el directorio o carpeta que contiene los ficheros

que queremos introducir dentro del nuevo archivo .aca.

Figura 9.7

Una vez seleccionado el directorio o carpeta aparecerá en el cuadro de texto, situado al lado izquierdo del botón examinar la ruta del directorio seleccionado.

Lo unico que nos queda es crear el nuevo archivo .aca, para ello debemos introducir el nombre del archivo que queremos crear, pulsar el botón Crear ACA, veasé en la figura 9.8:

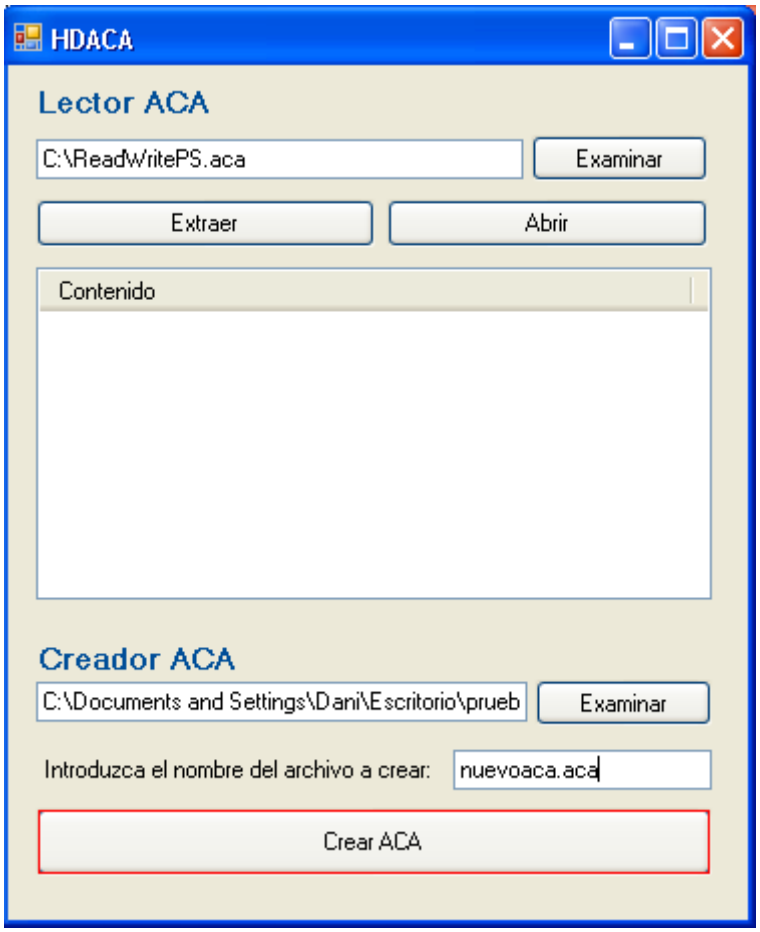

HDACA
Lector ACA
C:\ReadWritePS.aca
Examinar
Extraer
Abrir
Contenido
Creador ACA
C:\Documents and Settings\Dani\Escritorio\prueb
Examinar
Introduzca el nombre del archivo a crear:
nuevoaca.aca
Crear ACA

Figura 9.8

MANUAL DE USUARIO

Manual de Usuario

HDACA es un programa, fácil y sencillo de utilizar; donde podemos tratar los ficheros .aca, es decir podemos crear, extraer y abrir los ficheros .aca.

Este programa tiene partes bastante diferenciadas una es el Lector ACA donde podemos a partir de un archivo .aca, extraer los ficheros contenidos dentro del archivo .aca, y abrir los ficheros extraídos, y la otra parte es el Creador ACA donde a partir de un directorio o carpeta que contenga uno o más ficheros creará el archivo .aca.

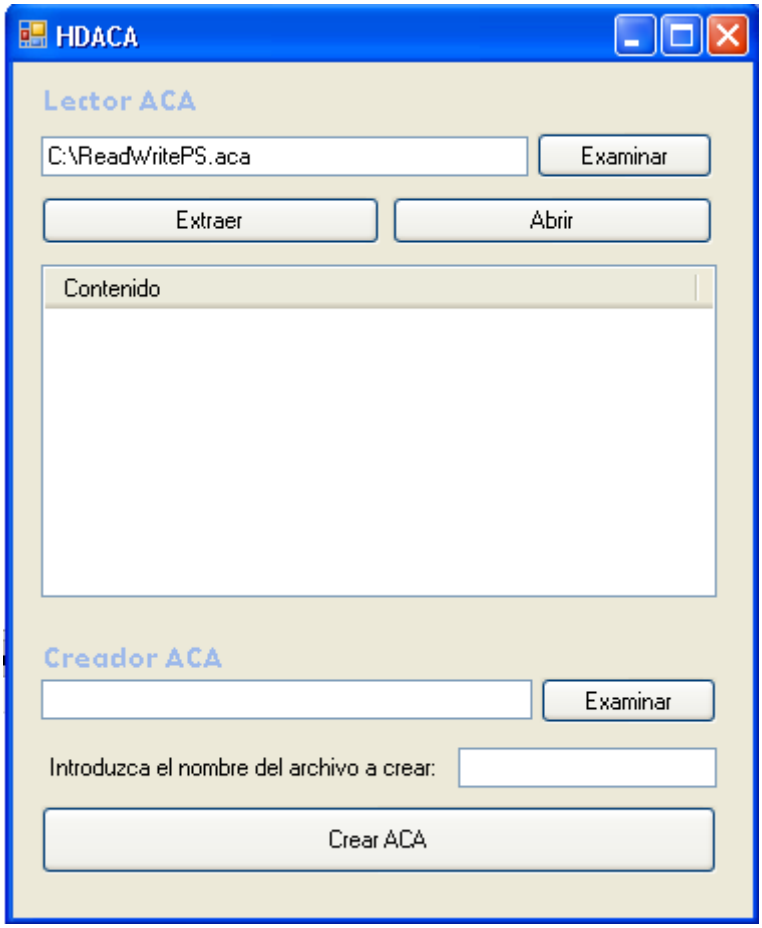


Figura 10.1

Como extraer los ficheros contenidos en un fichero .aca

1. Haz click en el botón **Examinar,** aparecerá la siguiente ventana(veasé en la figura 10.2).

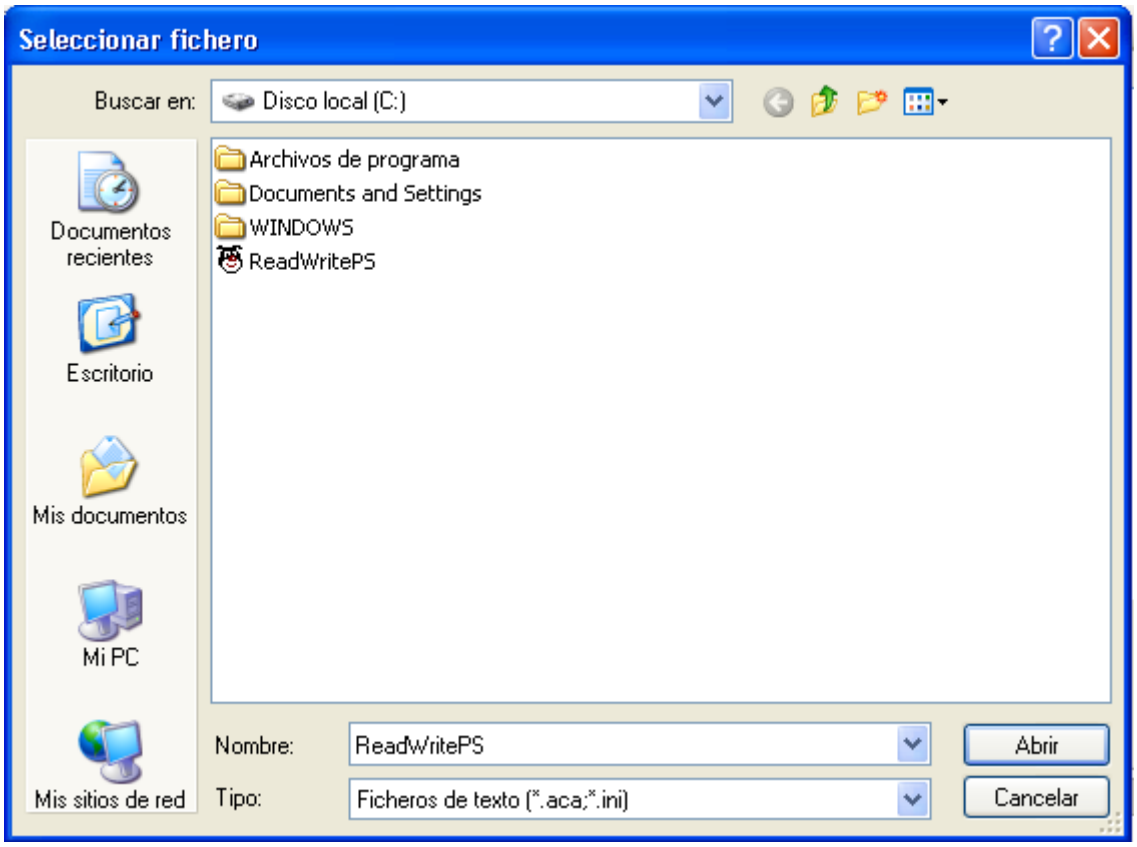


Figura 10.2

2. Busca y selecciona el fichero .aca, una vez seleccionado haz clic en el botón **Abrir.**
3. Ahora tan solo queda extraer los ficheros del archivo .aca, para ello haz clic en el botón **Extraer**.
4. Si se han extraído bien visualizaras en el *listView*, el nombre de los diferentes archivos contenidos en el archivo .aca, como podemos ver en la figura 10.3:

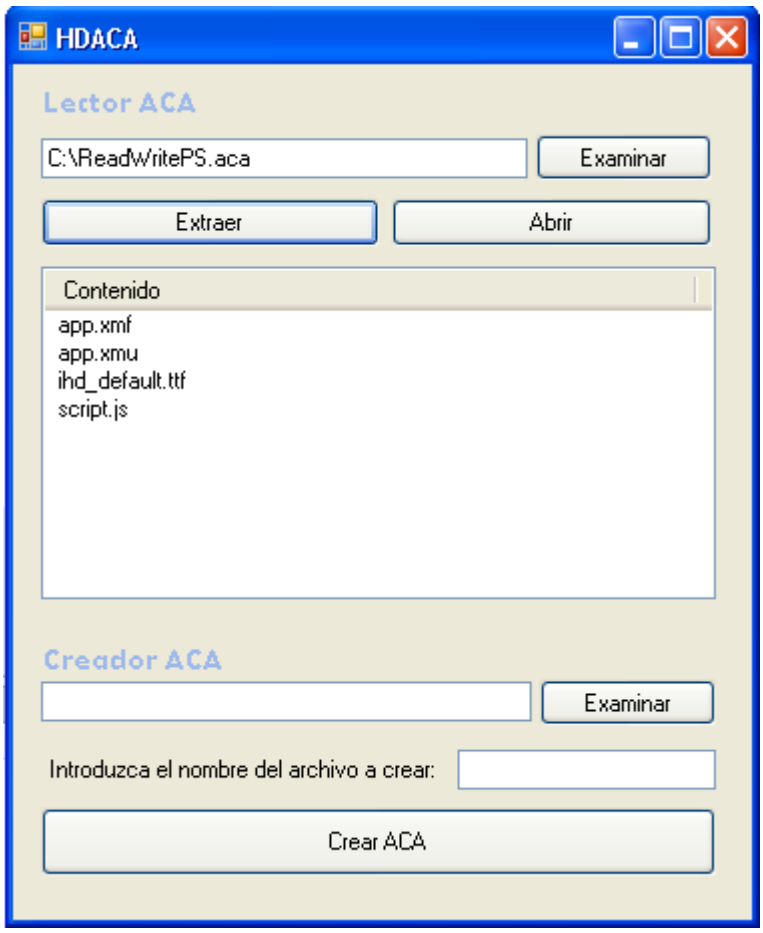


Figura 10.3

En cambio si no se han extraído con éxito, mostrará un menaje de error como el de la figura siguiente:

Figura 10.4

Como abrir los ficheros extraído del archivo .aca

1. Para poder abrir uno de los ficheros extraídos, primero debes seleccionar, el fichero que quieres abrir, para ello haz clic en el nombre del fichero que quieres abrir situado dentro del *listView* . Observe en la figura 10.5 como queda cuando hay un fichero seleccionado.

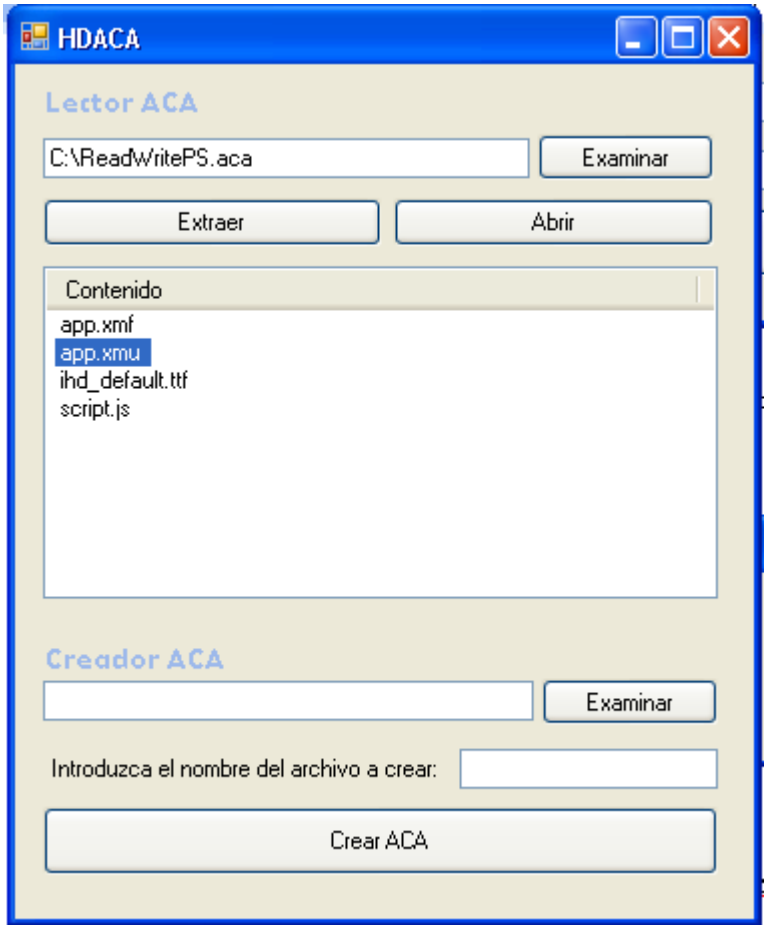


Figura 10.5

2. Con el fichero seleccionado haz clic en el botón **Abrir**.

Como crear un archivo .aca a partir de ficheros situado en una Carpeta

1. Haz clic en el botón **Examinar**, aparecerá la siguiente figura(veasé en la 10.6):

Figura 10.6

2. **Selecciona** la carpeta donde estén situados los fichero que quieres incluir dentro del nuevo archivo .aca.
3. Haz clic en el botón **Aceptar**. (Ten en cuenta que se introducirán dentro del archivo.aca todos los fichero que hayan dentro de la carpeta seleccionada)
4. Una vez seleccionada la ruta donde se encuentra los ficheros a incluir dentro del nuevo archivo .aca, introduce en el *textBox* el nombre del nuevo archivo .aca, veasé en la figura 10.7:

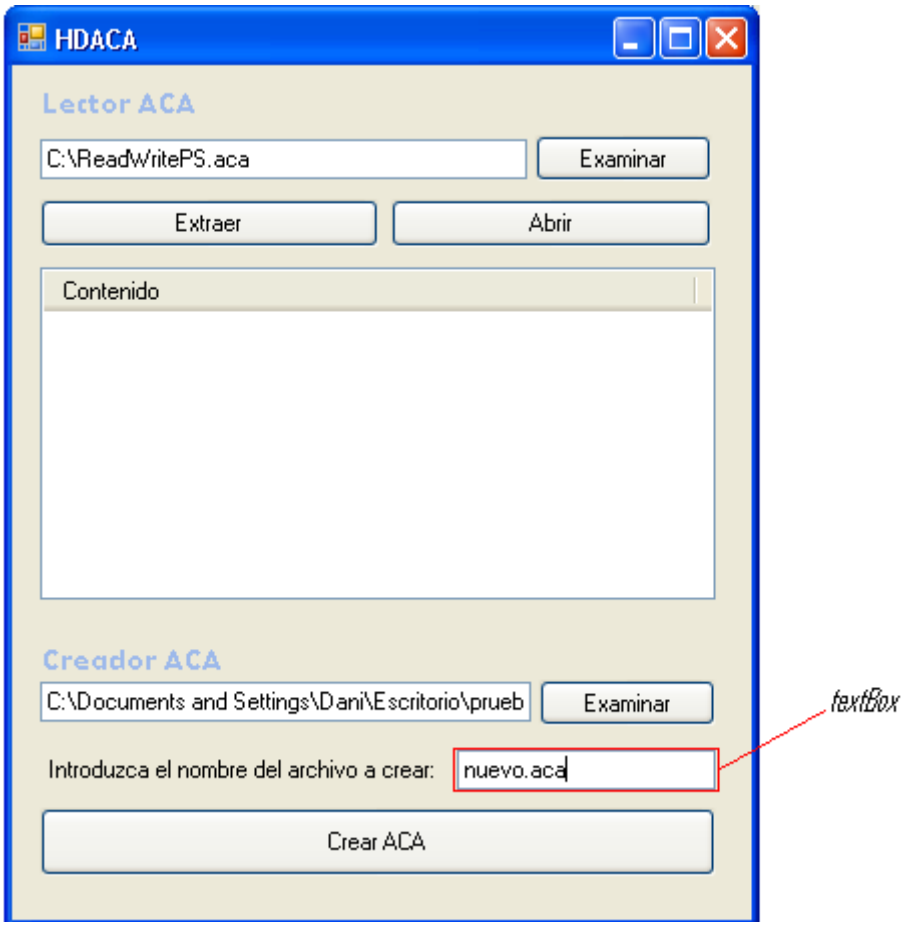


Figura 10.7

5. Haz clic en el botón **Crear ACA**, y si le aparece la ventana que se observa en la figura 10.8, significa es que el nuevo archivo .aca se ha realizado con éxito

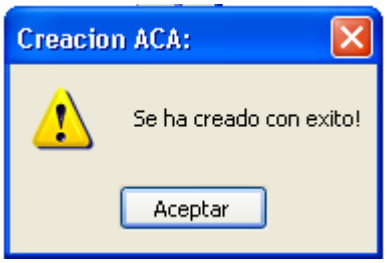


Figura 10.8

En cambio si no se ha creado el nuevo archivo .aca le aparecear el mensaje que se observa en la siguiente figura 10.9:

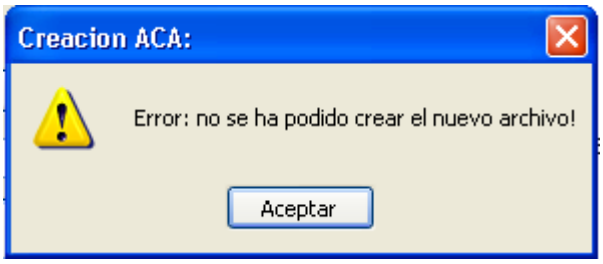


Figura 10.9

2:00 2:00

2:00 2:00

CONCLUSIONES Y MEJORAS FUTURAS

Conclusiones y mejoras futuras

Conclusiones

Los objetivos propuestos al principio han sido cumplidos.

Se ha realizado una aplicación que gestiona archivos de contenido avanzado.

Esta aplicación tiene un entorno grafico a modo de formulario o ventana de Windows.

En esta aplicación puede leer no tan solo los archivos de contenido avanzado creados por nuestra aplicación sino que puede leer cualquier archivo de contenido avanzado generado por algunas de las aplicaciones existentes en el mercado, objetivo que hemos cumplido ya que seguimos el estándar DVD FORUM.

Por último, los archivos de contenido avanzado generados por nuestra aplicación pueden ser leídos no tan solo por nuestra aplicación sino por cualquiera de la existentes en el mercado ya que a la hora de crearlos hemos seguido el estándar DVD FORUM.

Mejoras Futuras

En un futuro esta aplicación podría encriptar y desencriptar los archivos de contenido avanzado.

El método que en la actualidad se utiliza para encriptar este tipo de archivos y en su conjunto el HDDVD es AACS.

Os propongo los siguientes links para el que este interesado en continuar es aplicación realizando la encriptación de los archivos o los HD DVD.

http://es.wikipedia.org/wiki/Advanced_Access_Content_System

http://www.aacsla.com/home

BIBLIOGRAFÍA

Bibliografía

- *DVD Specifications for High Definition Video v1.0,*
  DVD FORUM.

- *UML y Patrones, una introducción al análisis y diseño orientado a objetos y al proceso unificado,* Craig Larman
  PEARSON EDUCACÓN, S.A., Madrid 2003

- *Visual C++ aplicaciones win32*, Fco.Javier Ceballos Sierra
  2a ed. rev. Madrid RA-MA 2003

- *Visual Basic .NET,* Francesco Balena
  McGRAW-HILL/INTERAMERICANA DE ESPAÑA, Madrid 2003
  Prentice Hall, Madrid, 1999

- *Enciclopedia del lenguaje C ++,* Fco.Javier Ceballos Sierra
  Ra-ma cop. Madrid 2003

- *Advanced Content Application Development for HD DVD,* Sonic
  V1.0 Updated August 10, 2006

- *HD DVD-Wikipedia Enciclopedia Libre*
  http://es.wikipedia.org/wiki/HD-DVD