Universidad de las Ciencias Informáticas

Facultad 9

Título: Interfaz de comunicación con la librería libvlc para las aplicaciones de reproducción y transmisión de media dentro del departamento de Señales Digitales.

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autor: Serguey Corvo Roque

Tutor: Ing. Jean Michael Suárez Pérez

Julio del 2010

Ciudad de La Habana

FRASE

Si no conozco una cosa,

la investigaré.

Louis Pasteur

AGRADECIMIENTOS

A mis padres por estar siempre cuando los he necesitado, por sus consejos, por brindarme amor y cariño, apoyarme, confiar en mí, ser una guía en mi camino y ser los mejores padres del mundo. Los quiero mucho.

A mi hermana Yutmay por todo el amor que siempre me ha brindado y complacerme en todo momento.

A mis hermanitas Yeniffer y Lorena por quererme tanto.

A mi primo Bertico por ser el hermano que siempre ha estado a mi lado.

A mi tío Bertico gran parte del hombre que soy hoy te lo debo a ti, gracias donde quieras que estés.

A mis abuelos por todo el amor que me han brindado y darme su apoyo incondicional.

A Osmaldo por apoyarme en todo momento.

A tío Ramoncito por quererme como un padre.

A mis tías Maribel y Dulce por preocuparse por mí.

A mis primas por el tiempo compartido juntos.

A mi cuñado Yusley por todo el apoyo brindado.

A toda mi gran familia por preocuparse siempre por mi bienestar y apoyarme.

A mi novia Anay por todo el amor que me brinda día a día, sin ti no lo hubiera logrado. Te amo.

A la maravillosa familia de mi novia por todo el apoyo que me han brindado y acogerme como uno más de sus hijos.

A mis amigos Erick, Leiser, Omar, Yordy, Yanier, Lionel y al resto de mis compañeros del pre por los gratos momentos que pasamos juntos.

A mis hermanas y hermanos de la uci Katy, Yenisel, Lisandra, Mariem, Yanita, Erieilis, Olga, Yosiel, Cinolkis, Pacheco, Diosbel, Arlen, Ernesto por los momentos que compartimos.

A Joel y Alain por ayudarme en la realización de la tesis.

A todos los compañeros de los dos grupos en los que estuve estos años porque juntos compartimos momentos maravillosos.

A mi tutor por guiarme por el camino correcto para poder terminar con éxito este trabajo.

A los compañeros del laboratorio 3 de uciteve.

A todas las personas que he conocido estos cinco años.

A todos los profesores que influyeron en mi formación como profesional.

A la Revolución, sin ella lograr este sueño hubiera sido imposible.

A la UCI por permitirme conocer personas maravillosas que nunca voy a olvidar.

A todas las personas que hicieron posible la realización de este trabajo

Muchas gracias a todos.

DEDICATORIA

A mis padres por la confianza, el apoyo y la dedicación que me han brindado durante toda mi vida.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____

Serguey Corvo Roque

Jean Michel Suárez Pérez

_____

Firma del Autor

_____

Firma del Tutor

RESUMEN

La reproducción y transmisión de medias son dos de los servicios más utilizados en la actualidad. En la Universidad de las Ciencias Informáticas, en el departamento de Señales Digitales se realizan estos servicios apoyados por el software Video Lan Client (VLC). Sin embargo, VLC no hace posible el control total de los procesos de reproducción y transmisión, por lo que es preciso crear aplicaciones que brinden prestaciones similares sin la necesidad de requerir su uso. En el presente trabajo se desarrolló una interfaz para permitir la comunicación con la librería libvlc de las aplicaciones de reproducción y transmisión de media que utiliza el departamento de Señales Digitales. Para ello se realizó un estudio en los proyectos que reproducen o transmiten media dentro del departamento de Señales Digitales, las prestaciones de VideoLAN para los procesos de reproducción y transmisión de medias, y las tecnologías y tendencias a usar para dar solución al problema existente. La interfaz realizada permitió comunicarse con la librería libvlc y aprovechar las prestaciones que brinda.

Palabras claves: libvlc, media, reproducción, streaming, transmisión.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La reproducción y transmisión de medias a través de las redes se han convertido en dos de los servicios de mayor demanda en la sociedad moderna, esto se debe en gran medida a los avances de la informática y el permanente desarrollo alcanzado por Internet. Su gran utilización en la actualidad para la transmisión de videos educativos, videoconferencias, radio en Internet, así como la distribución de televisión en vivo o video bajo demanda, los han convertido en servicios de gran valor social(1).

En un principio, los archivos de audio y video ubicados en Internet tenían que ser descargados completamente al ordenador. Este proceso se tornaba lento, debido al gran tamaño que suelen tener estos ficheros y el elevado ancho de banda que requieren para su transmisión(1). Una vez terminado el tiempo de descarga del archivo, se podía finalmente disfrutar el producto mediante la utilización de un programa reproductor de audio/video.

No fue hasta el año 1995 que fueron resueltos muchos problemas para poder transmitir audio y video en Internet, gracias a la aparición de la tecnología streaming. Esta consiste básicamente en la compresión de datos y distribución de contenidos multimedia (audio y video) por flujo continuo y de reproducción en tiempo real hacia la PC del usuario(2). Además esta tecnología permite una mayor distribución de contenidos de audio y video en calidad digital y posibilita la interacción del usuario con dichos contenidos (2).

Existen en el mundo diversas librerías que pueden ser utilizadas para realizar streaming, permitiendo al usuario realizar su propia aplicación según sus necesidades. Estas librerías cuentan con un conjunto de funcionalidades que permiten que el trabajo con las medias sea más sencillo, obteniéndose buenos resultados. Una de las librerías que ofrece facilidades para el trabajo con la reproducción y transmisión de audio y video es libvlc, la cual constituye el API[1] subyacente del software libre VideoLAN[2]. Los desarrolladores pueden utilizar las ventajas de libvlc para realizar complejas funcionalidades en procesos de reproducción y transmisión de archivos multimedia.

[1] **API** (A*pplicationProgramming Interface* **en español** *Interfaz de Programación de Aplicaciones***)**
[2] **VideoLAN**(reproductor de video de código libre, que integra diversos *códecs* de audio y video, así como diferentes tipos de contenedores y diversos protocolos de *streaming)*

El departamento de Señales Digitales de la Universidad de las Ciencias Informáticas, ha estado trabajando desde su creación con sistemas de reproducción y transmisión de medias, logrando buenos resultados. Estos procesos (reproducción y transmisión) se realizan apoyados en el software VLC[3], que es capaz de reproducir la mayoría de los archivos multimedia y también puede transmitir estos archivos por la red. Sin embargo, el departamento de Señales Digitales tiene la necesidad de crear aplicaciones que puedan brindar prestaciones similares a las de VLC, sin necesidad de requerir el uso del mismo completamente. Por los motivos antes mencionados surge la idea de utilizar la librería libvlc.

El uso de libvlc permitirá realizar sistemas que sean capaces de lograr mejores prestaciones en los procesos de reproducción y transmisión de medias para el departamento de Señales Digitales, empleando solamente las funcionalidades que dichos sistemas requieran y teniendo un mayor dominio en la reproducción y transmisión de las medias. Sin embargo, la utilización de libvlc de forma directa es bien complicada por la complejidad de sus funciones. A partir de la situación antes planteada se identificó el siguiente **problema a resolver**: Inexistencia de una interfaz que permita la comunicación con la librería libvlc, en las aplicaciones de reproducción y transmisión de media en el departamento de Señales Digitales.

Para dar respuesta a este problema se define como **objeto de estudio** las funciones y métodos que engloba la librería libvlc para reproducción y transmisión de medias seleccionándose como **campo de acción** la automatización de los procesos de reproducción y transmisión de media en plataformas libres dentro del departamento de Señales Digitales.

Para lograr lo antes expuesto se definió como **objetivo general** desarrollar una interfaz de comunicación con la librería libvlc.

Para el desarrollo del objetivo anteriormente planteado se formularon las siguientes tareas investigativas:

- Caracterizar las aplicaciones de reproducción y transmisión de medias en el departamento de Señales Digitales.
- Caracterizar los procesos de reproducción y transmisión de media.
- Describir el funcionamiento de los módulos de VideoLAN.

[3] VLC (Video Lan Client). Reproductor multimedia del proyecto VideoLAN.

- Describir las prestaciones de VideoLAN para la reproducción y transmisión de media.
- Caracterizar las funciones para la reproducción y transmisión de media en la librería libvlc.
- Diseñar una interfaz de comunicación para la librería libvlc.
- Implementar una interfaz de comunicación para la librería libvlc.
- Validar la interfaz propuesta.

Para el desarrollo de esta investigación se establece como **idea a defender**:

El desarrollo de una interfaz de comunicación con la librería libvlc garantizará mayores prestaciones en las herramientas libres de reproducción y transmisión de media, para las aplicaciones de reproducción y transmisión de medias del departamento de Señales Digitales.

Para lograr el cumplimiento de las tareas antes planteadas se hizo necesario utilizar determinados métodos científicos. Entre estos métodos se emplearon:

- **Analítico – sintético:** Para buscar la esencia de los fenómenos relacionados con los procesos de reproducción y transmisión de medias, del funcionamiento de la librería libvlc y los rasgos que los caracterizan y distinguen, a partir del análisis de las diferentes teorías.
- **La entrevista:** Se realizó a líderes y estudiantes del departamento de Señales Digitales de la UCI para conocer la ejecución de los procesos de reproducción y transmisión de medias dentro de los diferentes proyectos que componen dicho departamento.
- **Análisis histórico – lógico:** Se estudió la evolución de los procesos de reproducción y transmisión de audio y video a partir de su investigación en diferentes momentos históricos. Con ello fue factible estudiar particularidades básicas de su esencia, siguiendo una línea lógica en la investigación sobre su desarrollo y funcionamiento.
- **Observación:** Se obtuvo información a partir de realizar visitas a lugares donde se efectúan procesos de reproducción y transmisión de audio y video.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se abordarán los conceptos fundamentales que permitirán un mejor entendimiento para el dominio del problema. Se analizarán las características de los proyectos que realizan los procesos de reproducción y transmisión de medias dentro del departamento de Señales Digitales. Además se realizará un estudio de las prestaciones de VideoLAN para los procesos de reproducción y transmisión de medias.

1.2 Conceptos asociados al dominio del problema.

La **reproducción de sonido** es la acción de emitir ondas sonoras por medio de la voz, la música instrumental o el canto. La reproducción de sonido es clasificada en analógica o digital.

La **reproducción de sonido analógica** es el proceso en el que un altavoz de diafragma de mayor tamaño causa cambios en la presión atmosférica para formar ondas de sonido acústicas. La reproducción analógica es menos duradera que la digital, pues mientras más se utiliza, más calidad pierde. Sin embargo, en condiciones óptimas tiene una excelente calidad.

El proceso de **reproducción digital** es la acción que realiza un programa reproductor para decodificar un archivo, ya sea audio o video, y que este pueda escucharse o mostrarse para ser disfrutado por los usuarios. La ventaja de la reproducción de archivos digitales radica en la durabilidad de la información, ya que los archivos pueden ser reproducidos innumerables veces sin perder la calidad de su creación.

La palabra **video** que proviene del latín “videre” y quiere decir “yo veo”, hace referencia a la captación, procesamiento, transmisión y reconstrucción de una secuencia de imágenes y sonidos que representan escenas en movimiento (3). Una señal de video está formada por un grupo de líneas organizadas en cuadros, que son a la vez fraccionados en dos campos para guardar la información relacionada con el color y la luz de la imagen (3). El número de líneas, de cuadros y la forma de portar la información del color, depende del estándar de televisión concreto (4).

El **audio** es la técnica relacionada con la reproducción, grabación y transmisión del sonido. El audio digital se puede definir como el resultado de la codificación digital (representada por ceros y unos) de una onda sonora o de una señal analógica (5).

Un **códec** es un programa que incluye un conjunto de algoritmos e instrucciones para codificar y decodificar vídeo o audio digital, de forma que se reduzca el tamaño que ocupan. Códec son las iniciales de COdificador / DECodificador. Normalmente los algoritmos de compresión empleados conllevan a la pérdida de la calidad, por lo que es necesario utilizar códecs que logren mayor compresión y calidad (6). Los códecs utilizan diferentes algoritmos, algunos son muy rápidos a la hora de codificar, en cambio otros son demasiado lentos (7). El códec es un componente fundamental dentro de los archivos de audio y video, sin él, estos archivos ocuparían un considerable espacio en el ordenador.

La **transmisión de datos** es la acción de cursar datos a través de un medio de telecomunicaciones, desde el lugar en que son originados hasta el lugar en que son recibidos(8).

La **transmisión de archivos audiovisuales** tiene gran importancia en la actualidad. Es un servicio que posee una alta demanda y aceptación por los usuarios. Permite la comunicación de multimedia sobre las redes que soportan la comunicación de datos, brindando la posibilidad de enviar sonido e imágenes en movimiento a lugares remotos. Los continuos avances de la informática y el aumento del ancho de banda de las redes, han permitido que los archivos de audio y video puedan ser escuchados o vistos en tiempo real. A esta última forma de enviar estos archivos se le conoce como streaming.

Los términos **stream y streaming** se relacionan con la transmisión de información y su inmediata interpretación. Antes, la única forma disponible de ver un video o escuchar un sonido desde Internet era bajar el fichero completo al ordenador esperando el tiempo necesario, y luego mediante un programa reproducir el audio/video (9). Con la utilización de esta tecnología, el cliente puede ver o escuchar un archivo desde internet, sin la necesidad de que sea descargado previamente. El archivo solicitado es almacenado temporalmente en la memoria del ordenador y una vez que se termina de reproducir es eliminado de dicha memoria.

1.3 Objeto de Estudio

Para el desarrollo de la investigación se especificó como objeto de estudio las funciones y métodos que engloba la librería libvlc para reproducción y transmisión de medias.

1.3.1 Descripción General

La librería libvlc representa el API subyacente de VideoLAN. El reproductor VideoLAN (VLC), es un simple envoltorio de acceso a la librería libvlc. Los desarrolladores pueden utilizar la librería libvlc para aprovechar las complejas funcionalidades implementadas por VideoLAN. Libvlc se distribuye como una librería compartida, lo que permite al desarrollador de aplicaciones acceder a la funcionalidad del VideoLAN sin tener que empezar a codificarla el mismo desde cero (2).

1.3.1.1 Prestaciones de Video LAN para la reproducción y transmisión de media.

VideoLAN es el nombre del proyecto detrás de VLC Player. Su valor radica en el número de códecs que contiene por defecto al instalarlo, asegurando la compatibilidad con archivos multimedia de varias extensiones.

Este proyecto es de código abierto y permite su modificación total, por lo que se mantiene en constante desarrollo y actualización, gracias a las mejoras realizadas por la comunidad de código abierto. VLC se encuentra disponible para sistemas operativos Windows, MAC o Linux en casi todas sus distribuciones.

Pero sin dudas lo que hace de VLC uno de los mejores reproductores de la actualidad, es su capacidad de reproducir sin problemas archivos OGG, AVI, WAV, MP3, DVD, MPEG, ACC, DIVX, XVID, FLAC. Incluso puede llegar a reproducir archivos incompletos o dañados.

VLC encierra un gran número de funcionalidades y configuraciones fáciles de usar. Su consumo de recursos es muy bajo (ver Tabla1) y cualquier formato que sea reproducido posee una gran calidad de la imagen y el sonido.

VLC también puede ser usado como servidor para transmitir archivos MPEG-1, MPEG-2 y MPEG-4, DVDs y video en vivo sobre la red o usado como cliente para recibir, decodificar y visualizar flujos MPEG sobre varios sistemas operativos.

Como se ha podido apreciar VideoLAN (VLC) es un reproductor cómodo y sencillo, pero a su vez poderoso. Esto lo convierte en uno de los reproductores más estables de la actualidad.

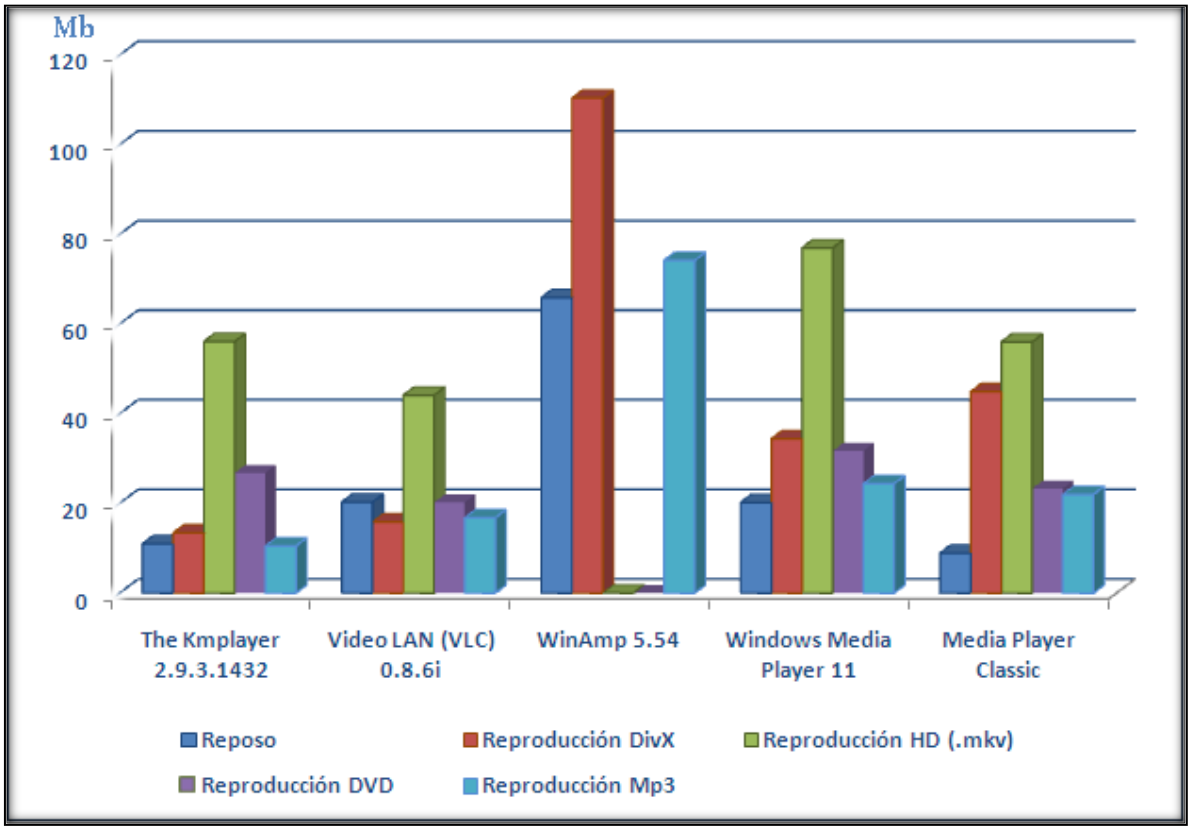


Tabla 1 Comparación entre reproductores actuales en cuanto al consumo de recursos de memoria RAM (Robert Gago)

1.3.1.2 Módulos que conforman VideoLAN

VideoLAN está compuesto por un conjunto de módulos que hacen posible su funcionamiento. A continuación se presentan dichos módulos, donde se muestra cómo cada uno de ellos está conformado por funcionalidades de la librería libvlc.

1.3.1.2.1 Excepciones

El módulo excepciones contiene las funcionalidades que proporciona la librería libvlc para el tratamiento de errores.

- Funciones que integran este módulo:

void libvlc_exception_init (libvlc_exception *p_exception);

Inicializa la excepción.

int libvlc_exception_raised (libvlc_exception *p_exception);

Permite comprobar si se ha producido una excepción.

void libvlc_exception _clear (libvlc _exception * p exception);

Hace un reset[4] de la excepción para ser reutilizada.

char* libvlc_exception _get_message (libvlc _exception * p exception);

Se obtiene el mensaje de la excepción.

1.3.1.2.2 Acceso al VideoLAN

Las funciones de este módulo son muy importantes, pues son las que permiten crear o destruir instancias de la aplicación VideoLAN. El resto de los módulos dependen mucho de la instancia que se haya generado, es decir, sin esta instancia se haría imposible el trabajo con los otros módulos.

- Funciones que integran este módulo:

typedef struct libvlc_instance_t libvlc_instance_t;

Estructura que representa una instancia de la aplicación VLC.

libvlc_instance_t* libvlc_new (int,char);**

Crea e inicializa una nueva instancia.

void libvlc_destroy (libvlc_instance_t *);

Destruye una instancia.

int libvlc_get_vlc_id (libvlc_instance_t*p instance);

Obtiene el id de la instancia.

1.3.1.2.3 Control de la lista de reproducción

En este módulo se encuentran las funciones relacionadas con el control de la reproducción. Permite controlar la reproducción en curso y añadir o eliminar elementos de la lista de reproducción.

[4] **Reset:** Del inglés reponer o reiniciar. Se conoce como reset a la puesta en condiciones iniciales de un sistema(10).

- Funciones que integran este módulo:

void libvlc_playlist_play (libvlc_instance_t * , int, int,char ** , libvlc_exception_t *) ;

Inicia la reproducción de un elemento.

void libvlc_playlist_pause (libvlc_instance_t * ,libvlc_exception_t *) ;

Pone en pausa el elemento que se está reproduciendo.

int libvlc_playlist _isplaying (libvlc_instance_t * , libvlc_exception_t *) ;

Comprueba si hay alguna reproducción en curso.

int libvlc_playlist_items_count (libvlc_instance_t *, libvlc_exception_t *);

Informa la cantidad de elementos de la lista de reproducción.

int libvlc_playlist _stop (libvlc_instance_t *, libvlc_exception_t *) ;

Detiene la reproducción en curso.

int libvlc_playlist_next (libvlc_instance_t *, libvlc_exception_t *);

Pone a reproducir el siguiente elemento de la lista de reproducción respecto al actual.

int libvlc_playlist_prev (libvlc_instance_t *, libvlc_exception_t *);

Pone a reproducir el elemento anterior de la lista de reproducción respecto al actual.

int libvlc_playlist _clear (libvlc_instance_t *, libvlc_exception_t *) ;

Elimina todos los elementos de la lista.

int libvlc_playlist_add (libvlc_instance_t *, constchar *,constchar *,libvlc_exception_t *);

Añade un elemento al final de la lista.

libvlc_input_t* libvlc_playlist_get_input (libvlc_instance_t *, libvlc_exception_t*);

Obtiene el elemento de entrada que se está reproduciendo.

1.3.1.2.4 Control de audio

En este módulo aparecen las funcionalidades que posibilitan controlar el volumen de la aplicación, permitiendo variar el nivel de audio, incluyendo el manejo del estado mudo.

- Funciones que integran este módulo:

void libvlc_audio_togle_mute (libvlc_instance_t *, libvlc_exception_t*);

Pone el audio en estado mudo y al ser llamada de nuevo devuelve el sonido con el valor anterior.

void libvlc_audio_get_mute (libvlc_instance_t *, libvlc_exception_t *) ;

Permite conocer si el audio está en el estado mudo.

void libvlc_audio_set_mute (libvlc_instance_t *, vlc_bool_t ,libvlc_exception_t *) ;

Activa o desactiva el estado mudo.

int libvlc_audio_get_volume (libvlc_instance_t *, libvlc_exception_t *);

Obtiene el nivel del volumen.

void libvlc_audio_set_volume (libvlc_instance_t *, int, libvlc_exception_t *) ;

Permite modificar el nivel del volumen.

1.3.1.2.5 Control de video

En este módulo las funciones realizan el trabajo del tratamiento gráfico del video. También aparecen funcionalidades que permiten controlar el tamaño de la ventana que representa el video. Igualmente, se puede controlar la opción de pantalla completa pudiendo activarla o desactivarla.

- Funciones que integran este módulo:

void libvlc_toggle_fullscreen (libvlc_input_t * ,libvlc_exception_t *) ;

Permite poner el video a pantalla completa.

void libvlc_set_fullscreen (libvlc_input_t * ,libvlc_exception_t *) ;

Activa o desactiva la opción de pantalla completa.

int libvlc_get_fullscreen (libvlc_input_t * ,libvlc_exception_t *) ;

Permite saber si está activa la opción de pantalla completa.

int libvlc_video_get_height (libvlc_input_t * ,libvlc_exception_t *) ;

Devuelve el valor de alto del video en curso.

int libvlc_video_get_width (libvlc_input_t * ,libvlc_exception_t *) ;

Devuelve el valor de ancho del video en curso.

char* libvlc_video_get_aspect_ratio (libvlc_input_t * ,libvlc_exception_t *) ;

Permite saber la relación de aspecto del video.

void libvlc_video_set_aspect_ratio (libvlc_input_t _*, char * , libvlc_exception t *) ;

Permite modificar la relación de aspecto del video.

void libvlc_video_take_snapshot (libvlc_input_t *, char * ,libvlc_exception_t*) ;

Permite hacer una captura instantánea del video.

void libvlc_video_resize (libvlc_input_t *, int, int, libvlc_exception_t *) ;

Reconfigura la ventana del video.

void libvlc_video_set_size (libvlc_instance_t *, int, int, libvlc_exception_t *) ;

Modifica el tamaño por defecto del video saliente.

void libvlc_video_set_viewport (libvlc_instance_t *, const libvlc_rectangle_t, const libvlc_rectangle_t, libvlc_exception_t *) ;

Pone la salida de video viewport[5]para una salida de video sin ventanas.

1.3.1.2.6 VideoLAN Manager (VLM)

En este módulo aparecen las funcionalidades de servidor de streaming, permitiendo la transmisión de flujo transcodificado a través de una red IP.

- Funciones que integran este módulo:

void libvlc_vlm_add_broadcast (libvlc_instance_t *, const char *, const char *, const char *, int, const char *const *, int, int, libvlc_exception_t *);

Permite poner una media en transmisión por broadcast[6]

[5] ***Viewport:*** ventana de visualización(11).

void libvlc_vlm_del_media (libvlc_instance_t *, const char *, libvlc_exception_t *);

Elimina un broadcast.

void libvlc_vlm_set_enabled (libvlc_instance_t *, const char *, int, libvlc_exception_t *);

Pone activo o no activo un broadcast.

void libvlc_vlm_set_output (libvlc_instance_t *, const char *, const char *, libvlc_exception_t *);

Permite editar los parámetros de salida para un broadcast.

void libvlc_vlm_set_input (libvlc_instance_t *, const char *, const char *, libvlc_exception_t *);

Permite editar los parámetros de entrada para un broadcast.

void libvlc_vlm_set_loop (libvlc_instance_t *, const char *, int, libvlc_exception_t *);

Permite activar la reproducción constante.

void libvlc_vlm_change_media (libvlc_instance_t *, const char *, const char *, const char *, int, const char *const *, int, int, libvlc_exception_t *);

Permite editar el broadcast.

void libvlc_vlm_play_media (libvlc_instance_t *, const char *, libvlc_exception_t *);

Empieza a reproducir el broadcast.

void libvlc_vlm_stop_media (libvlc_instance_t *, const char *, libvlc_exception_t *);

Detiene el broadcast en curso.

void libvlc_vlm_pause_media (libvlc_instance_t *, const char *, libvlc_exception_t *);

Pone en pausa el broadcast.

1.3.1.2.7 Control de la señal de entrada

En este módulo las funciones permiten recopilar información sobre el estado de la reproducción de una media. El tiempo de reproducción total y el tiempo de reproducción actual son datos que se pueden obtener con dichas funciones.

[6] **Broadcast:** difundir a través de un medio de comunicación(12).

- Funciones que integran este módulo:

mtime_t libvlc_input_get_length (libvlc_input_t *, libvlc_exception);

Obtiene la duración del elemento de entrada.

mtime_t libvlc_input_get_time (libvlc_input_t *, libvlc_exception) ;

Obtiene el tiempo del instante de reproducción.

void libvlc_input_set_time (libvlc_input_t *, mtime_t , libvlc_exception) ;

Modifica el tiempo de la reproducción.

float libvlc_input_get_position (libvlc_input_t *, libvlc_exception) ;

Obtiene la posición de la reproducción en curso.

void libvlc_input_set_position (libvlc_input_t *, float , libvlc_exception) ;

Modifica la posición de la reproducción.

void libvlc_set_rate (libvlc_input_t *, float rate, libvlc_exception) ;

Permite modificar la velocidad de reproducción de la media.

float libvlc_get_rate (libvlc_input_t *, libvlc_exception)

Permite conocer la velocidad de reproducción de la media.

1.3.1.3 Funciones para la reproducción de media en la librería libvlc

La librería libvlc se compone de un conjunto de módulos y cada uno de estos está integrado por un número de funcionalidades. A continuación se muestran las principales funcionalidades que tiene libvlc para lograr el proceso de reproducción de una media.

void libvlc_media_player_play (libvlc_media_player_t * , libvlc_exception_t *)

A la función libvlc_media_player_play se le pasa como parámetro un objeto media player y un puntero excepción, y su objetivo es poner a reproducir un elemento.

void libvlc_media_player_pause (libvlc_media_player_t * , libvlc_exception_t *)

A la función libvlc_media_player_pause se le pasa como parámetro un objeto media player y un puntero excepción, y su objetivo es cambiar el estado de la reproducción a pausa.

void libvlc_media_player_stop (libvlc_media_player_t * , libvlc_exception_t *)

A la función libvlc_media_player_stop se le pasa como parámetro un objeto media player y un puntero excepción, y su objetivo es detener la reproducción.

void libvlc_media_list_add_media (libvlc_media_list_t *, libvlc_media_t *, libvlc_exception_t *)

A la función libvlc_media_list_add_media se le pasa como parámetro un objeto media_list, una media y un puntero excepción, y su objetivo es añadir una media a la lista de reproducción.

void libvlc_media_list_remove_index (libvlc_media_list_t *, int , libvlc_exception_t *)

A la función libvlc_media_list_remove_index se le pasa como parámetro un objeto media_list, la posición de la media a eliminar y un puntero excepción, y su objetivo es eliminar una media de la lista de reproducción.

1.3.1.4 Funciones para la transmisión de media en la librería libvlc

Para el proceso de transmisión de medias, la librería libvlc también tiene definidas un conjunto de funcionalidades. A continuación se muestran las principales, con una descripción detallada de cada una de ellas.

void libvlc_vlm_add_broadcast (libvlc_instance_t *, const char *, const char *, const char *, int, const char *const *, int, int, libvlc_exception_t *)

La función libvlc_vlm_add_broadcast tiene como objetivo añadir la transmisión de un broadcast. Los parámetros de esta función son: el objeto libvlc, nombre del broadcast, los parámetros de entrada, los parámetros de salida, número de opciones adicionales, opciones adicionales, si se activa o no la transmisión, si la reproducción será constante o no, y un puntero excepción.

void libvlc_vlm_add_vod (libvlc_instance_t *, const char *, const char *, int, const char *const *, int, char *, libvlc_exception_t *)

La función libvlc_vlm_add_vod tiene como objetivo añadir la transmisión de un video bajo demanda (vod). Los parámetros de esta función son: el objeto libvlc, nombre del vod, los parámetros de entrada, número de opciones adicionales, opciones adicionales, si se activa o no la transmisión, opciones de codificación, y un puntero excepción.

void libvlc_vlm_del_media (libvlc_instance_t *, const char *, libvlc_exception_t *)

La función libvlc_vlm_del_media tiene como objetivo eliminar la transmisión de una media. Los parámetros que se le pasan a esta función son: un objeto libvlc, el nombre de la media y un puntero excepción.

void libvlc_vlm_play_media (libvlc_instance_t *, const char *, libvlc_exception_t *)

La función libvlc_vlm_play_media tiene como objetivo poner a transmitir una media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media y un puntero excepción.

void libvlc_vlm_stop_media (libvlc_instance_t *, const char *, libvlc_exception_t *)

La función libvlc_vlm_stop_media tiene como objetivo detener la transmisión de una media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media y un puntero excepción.

void libvlc_vlm_pause_media (libvlc_instance_t *, const char *, libvlc_exception_t *)

La función libvlc_vlm_pause_media tiene como objetivo poner en pausa la transmisión de una media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media y un puntero excepción.

1.4 Descripción actual del dominio del problema

En la actualidad, con el desarrollo de la Internet se han obtenido grandes avances en la transmisión de medios audiovisuales, logrando ser este uno de los servicios de mayor aceptación en la sociedad moderna. Esto ha conducido a la creación y perfeccionamiento de aplicaciones que permitan brindar mejores prestaciones a los usuarios.

El departamento de Señales Digitales de las Universidad de las Ciencias Informáticas, ha estado trabajando desde sus inicios con sistemas de reproducción y transmisión de medias. Algunos de estos

sistemas son: PRIMICIA, Sistema de Catalogación de Medias (SCCM), Plataforma para Trasmisión Abierta para Radio y Televisión (PTARTV) y Plataforma VideoWeb.

PRIMICIA es una plataforma de teletexto donde se pueden mostrar en forma de noticias las informaciones. Cuando se reproduce un material se realiza desde una dirección local o desde un flujo streaming, auxiliándose de un plugin[7] de VLC para Mozilla, incompletando el dominio sobre el medio audiovisual. En este proyecto no se realiza la transcodificación de materiales, ni la transmisión por vías streaming.

SCCM constituye una plataforma de escritorio que permite la captura y catalogación de medias. En el proyecto se observa la reproducción de materiales en los subsistemas de catalogación, audio y video, y se realiza la transcodificación de materiales en los subsistemas de audio y video. Este sistema cuenta con un servidor de streaming que permite la visualización remota de las medias a partir de los flujos que este brinda. Los procesos de reproducción y transcodificación de medias se realizan con la utilización de aplicaciones como Mencoder, VLC y MPlayer. La transmisión se realiza con la comunicación bajo TelNet con el software VLC, impidiendo así el manejo total de los ficheros que se trasmiten.

PTARTV es una plataforma que tiene como objetivo principal integrar y mejorar los procesos de transmisión de medias dentro de la Universidad de las Ciencias Informáticas. Esta plataforma pretende reproducir archivos almacenados en una dirección local o mediante un flujo de streaming determinado. En el proyecto PTARTV se realiza la transcodificación de medias.

VideoWeb constituye una plataforma que pretende mediante el uso de la tecnología streaming brindar acceso a contenido multimedia en la web que pueda ser visto sin ser descargado en los ordenadores. En este sistema se realiza la reproducción de un material que esté almacenado en una dirección local o por un flujo streaming. El flujo de streaming es capturado por una tarjeta y mediante el MPlive es llevado al servidor Darwin para su transmisión. La plataforma VideoWeb tiene dentro de sus funciones la transcodificación de medias.

[7] **Plugin:** programa de ordenador que interactúa con otro programa para aportarle una función o utilidad específica, generalmente muy específica(13).

Los sistemas mencionados efectúan los procesos de reproducción y transmisión mediante la unión de varias herramientas u otras aplicaciones ya existentes, que no permiten un manejo total de los archivos de media, limitando con ello el número de prestaciones a los usuarios finales del producto.

1.5 Situación problemática

La reproducción y transmisión de media en la actualidad a través de las redes se ha convertido en unos de los servicios de mayor popularidad y demanda para la sociedad. Esto se debe en gran medida al desarrollo acelerado de la informática y las comunicaciones, lo que ha hecho posible la transmisión de radio por internet, videoconferencias, televisión en vivo, entre otras prestaciones, las cuales son de vital importancia para la sociedad moderna.

Como consecuencia del alto consumo de este servicio se hace necesaria la creación de nuevas aplicaciones que puedan satisfacer con mayor calidad las necesidades de los usuarios finales. El departamento de Señales Digitales desde su creación ha venido trabajando en este aspecto, desarrollando aplicaciones capaces de cumplir en gran medida las necesidades de la comunidad universitaria. Estas aplicaciones se apoyan en un conjunto de herramientas ya creadas, que no permiten el manejo total de los archivos de media. Por lo que es necesaria la creación de una aplicación independiente, capaz de cumplir las necesidades de los usuarios.

Existen en el mundo numerosas librerías que con su uso facilitan la creación de aplicaciones de forma más sencilla. Sin embargo, lograr la comunicación con estos componentes es una tarea compleja. La librería libvlc brinda un gran número de funcionalidades que permiten desarrollar aplicaciones de reproducción y transmisión de media. En la actualidad, no existe ninguna manera de comunicarse con la librería libvlc, por lo que se hace necesaria la creación de una interfaz que lo posibilite.

1.6 Conclusiones

En este capítulo se describieron de forma general las principales características de la interfaz a desarrollar. Se presentaron una serie de conceptos, que ayudarán a una mejor comprensión del presente trabajo y se realizó una descripción del objeto de estudio. Además se recogieron las causas que hacen necesaria la realización de la interfaz de comunicación con la librería libvlc para el departamento de Señales Digitales de la UCI.

CAPÍTULO 2: TECNOLOGÍAS A UTILIZAR

2.1 Introducción

En este capítulo se abordarán las características fundamentales de las tecnologías a utilizar para dar solución al problema en cuestión, realizándose un estudio del lenguaje de programación, el entorno de desarrollo y la arquitectura. Además se explicarán los patrones de diseño empleados en la construcción de la interfaz de comunicación con la librería libvlc y el estándar de codificación.

Características de UML como lenguaje de modelado

Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema, por lo tanto, es un lenguaje estándar para los planos del software (14).

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML), es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. UML cubre la documentación de la arquitectura y proporciona requisitos, pruebas y las actividades de planificación de proyectos (14).

Entre las principales características de UML se encuentran:

- El sistema de software es diseñado y documentado antes de que sea codificado.
- Los lógicos “agujeros” en la etapa de diseño podrán ser detectados con anterioridad. El software se comportará de la forma esperada y surgirán menos imprevistos.
- El diseño total del sistema dicta el modo en que se desarrollará el software. Las decisiones finales se harán antes de que se encuentre código mal escrito.
- Cualquier modificación en el sistema será más fácil de llevar a cabo sobre la documentación UML (15).

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Por tales motivos y por el dominio que presenta el autor de la investigación sobre este lenguaje, se elige para realizar el modelado de la interfaz.

2.2 Características de C++,como lenguaje de programación

Es un lenguaje de programación derivado del lenguaje C que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente (16).

Entre las principales características de C++ como lenguaje de programación se encuentran las siguientes:

- Programación orientada a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista como una comunicación entre objetos y no como una secuencia estructurada de código. Además, permite una mayor reutilización de código en una forma más lógica y productiva.

- El código escrito en C++ es muy corto en comparación con otros lenguajes de programación, ya que el uso de caracteres especiales es preferible a las palabras claves.

- Ahorro de tiempo, ya que no es necesario volver a compilar la aplicación completa al realizar un solo cambio, sólo el archivo que lo contiene. Además, esta característica permite vincular código C++ con el código producido en otros lenguajes, como el ensamblador.
- El código resultante de una compilación de C++ es muy eficiente debido a su dualidad como lenguaje de alto nivel y lenguaje de bajo nivel (17).

En principio cualquier lenguaje de programación puede ser utilizado para desarrollar la interfaz de comunicación con la librería libvlc. En este caso se define C++ como lenguaje de programación debido a que es el lenguaje que necesita el departamento de Señales Digitales para esta interfaz. Se puede agregar además que la librería libvlc está escrita en dicho lenguaje lo que facilita en gran medida el trabajo con la misma.

2.3 IDE de desarrollo

Para facilitar el trabajo de los desarrolladores de software, son creados los Entornos de Desarrollo Integrado (en inglés Integrated Development Environment o IDE). Un IDE está conformado por un

conjunto de herramientas con las cuales los programadores desarrollan el código. El IDE puede estar desarrollado para la utilización de un solo lenguaje de programación o para un conjunto de estos, según se haya concebido.

Entre los IDEs más utilizados para el lenguaje C++ se encuentran en la actualidad el Visual C++ Studio y el Borland C++ Builder. Sin embargo, se descarta la utilización de estos para la realización de la interfaz por ser IDEs privativos, estando en contra de las políticas seguidas en el departamento de Señales Digitales. Durante la investigación se identificaron otros IDEs que presentan el lenguaje seleccionado en su forma nativa y se encuentran bajo licencia GPL, estos son: Anjuta, Code::Blocks y Qt Creator.

2.3.1 Anjuta

Anjuta es un versátil Entorno de Desarrollo Integrado para el escritorio GNOME[8]. Ofrece una serie de servicios avanzados de programación, que incluyen la gestión de proyectos y un sistema de depuración interactiva, un diseñador integrado, una memoria integrada de valgrind[9], generador de clases y un potente editor de código fuente.

Dentro de las principales características de Anjuta se encuentran:

- Interfaz de usuario

Anjuta tiene un flexible y avanzado sistema de acoplamiento que le permite exponer todos los puntos de vista en la forma deseada. Los diseños son persistentes para cada proyecto, puede mantener diferentes diseños para diferentes proyectos.

- Plugins

Anjuta es muy extensible con plugins. Casi todas las características de Anjuta se implementan mediante plugins que pueden ser dinámicamente activados o desactivados.

[8]GNOME: (acrónimo del inglés *GNU Network Object Model Environment*, *Entorno GNU de Modelado de Objetos en Red*) es un entorno de escritorio para GNU/Linux y otros sistemas derivados de Unix(19).

[9]Valgrind: herramienta de software libre que ayuda a detectar problemas de memoria(20).

- Administrador de Archivos

El gestor de archivos se comporta más o menos como el gestor de ficheros típicos en una vista de árbol. En ella se enumeran todos los directorios y archivos en el proyecto actual (18).

2.3.2 Code::Blocks

Code::Blocks es un programa en C++. Es un IDE libre construido para satisfacer las necesidades más exigentes de los usuarios. Está diseñado para ser muy extensible y totalmente configurable. Construido alrededor de un marco plugin. Cualquier tipo de funcionalidad se puede añadir al IDE mediante la instalación de un plugin.

Dentro de las principales características de Code:: Blocks se encuentran:

- Compiladores:
  - Espacios de trabajo para combinar múltiples proyectos.
  - Dependencias entre proyectos dentro del área de trabajo.
  - Las importaciones de proyectos y espacios de trabajo (el código ensamblador no es soportado).
- Depurador:
  - Código de los puntos de interrupción.
  - Los puntos de interrupción de datos (lectura, escritura y lectura o escritura).
  - Símbolos de función de visualización local y los argumentos.
  - Volcado de memoria personal.
- Interfaz:
  - El resaltado de sintaxis, personalizable y ampliable.
  - Plegado de código para C + + y archivos XML.
  - Interfaz con pestañas.
  - Examinador de clases.
  - Guión inteligente.(21)

La desventaja más notable en Code:: Blocks es su incapacidad de detectar posibles referencias a variables, estructuras o clases declaradas en archivos externos y que están a disposición del programador

mediante el uso de librerías e importaciones. De esta manera se hace un poco más complejo el trabajo con este IDE.

2.3.3 Qt Creator

Qt Creator es una nueva plataforma de entorno de desarrollo integrado disponible junto con las bibliotecas Qt, bajo licencia LGPL. Qt Creator está diseñado para el trabajo con bibliotecas Qt para sus versiones 4.x, en este caso se define utilizar la versión 4.5.0.

Entre las principales características de Qt Creator se pueden mencionar las siguientes:

- Resaltado de sintaxis y completado de código.
- Control de código estático y consejos de estilo a medida que se escribe.
- Apoyo a la refactorización código fuente.
- Ayuda sensible al contexto.
- Coincidencia de paréntesis y los modos de selección de paréntesis.
- Capacidades de edición avanzada (22).

Además de las características antes mencionadas se debe resaltar que las bibliotecas Qt son multiplataforma y las aplicaciones que se apoyan en ellas tienen buena respuesta y un consumo de recursos aceptable.

Los tres IDEs estudiados tienen grandes prestaciones para la realización de la interfaz. En este caso se define utilizar Qt Creator pues actualmente se utiliza en proyectos del departamento de Sistemas Digitales, es un IDE multiplataforma, brinda importantes prestaciones en el completamiento de código y tiene una gran facilidad de uso.

2.4 Arquitectura utilizada

Para el desarrollo de la interfaz de comunicación con la librería libvlc se definió utilizar una arquitectura en capas. Esta arquitectura estará conformada por dos capas, la capa de lógica de negocio y la capa de acceso a los métodos nativos de libvlc. Al diseñar este tipo de sistema de software separado por capas se debe tener en cuenta que dichas capas pueden quedar dentro de uno o más niveles. Existen varios tipos de arquitecturas en niveles, en este caso se decide utilizar la arquitectura 1 tier, donde la capas existentes

se integran en un solo nivel. A este tipo de diseño se le hace llamar monolítico, no presentan interfaz, suelen ser cerrados y son comúnmente utilizados en sistemas heredados.

2.5 Patrones de diseño utilizados

Un patrón de diseño abstrae e identifica los aspectos claves de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizables. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de la responsabilidad. Cada patrón de diseño se centra en un problema concreto, describiendo cuándo aplicarlo y si tiene sentido hacerlo teniendo en cuenta otras restricciones de diseño, así como las consecuencias y las ventajas e inconvenientes de su uso (23).

Para la construcción de la interfaz de comunicación con la librería libvlc se definió utilizar los siguientes patrones de diseño:

Experto: para determinar que clase debe asumir una responsabilidad a partir de la información que posee. Además permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para realizar los que se le oriente.

Creador: para guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El diseño bien asignado permitirá soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Alta cohesión: para manejar la complejidad de las clases dentro de los límites manejables, mejorar la claridad y facilidad con que se entiende el diseño, simplificar el mantenimiento y las mejoras en funcionalidad. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen grandes trabajos.

Bajo acoplamiento: para dar soporte a una dependencia escasa y a un aumento de la reutilización. Los

cambios de las clases afines ocasionan cambios locales, difíciles de entender cuando están aisladas y difíciles de reutilizar debido a que dependen de otras clases.

Controlador: para manejar y controlar los eventos del sistema. Con la utilización de este patrón se logra separar la lógica de negocio de la capa de presentación. De esta manera se logra un mayor control sobre el sistema y se favorece la reutilización de código.

2.6 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física. Un estilo de codificación eficiente facilitará la comprensión y la modificación del código.

Durante la codificación es necesario considerar los siguientes criterios de calidad:

CRITERIO	OBJETIVO
Facilidad de comunicación	Proporcionar al usuario entradas y salidas fácilmente asimilables.
Auto descripción	Proporcionar en el código, explicaciones sobre la implantación realizada.
Simplicidad	La implantación realizada debe hacerse de la forma más comprensible posible.

Tabla 2 Criterios de calidad

A continuación se presenta el estándar de codificación utilizado en el desarrollo de este trabajo.

Principios generales

Los nombres de cada uno de los elementos del programa deben ser significativos; su nombre debe explicar en lo posible el uso del elemento.

- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas.
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.

- Los atributos deben comenzar con letra minúscula y los métodos deben comenzar con letra mayúscula. Ambos serán nombrados en lengua inglesa.
- Los parámetros deben comenzar con la letra p y su nombre lo más similar posible al atributo que se refiere.

Los elementos serán nombrados teniendo en cuenta las siguientes reglas.

Elemento	Reglas de nombramiento
Clases, interfaces y archivos fuente	Nombre sustantivo singular, con la primera letra en mayúscula y las demás en minúsculas.
Variables	Nombre sustantivo en minúscula. Para separar palabras se usará el guión bajo: _

Tabla 3 Nombramiento de elementos

Comentarios

Cada programa deberá comenzar con un comentario que incluya:

- Autor
- Objetivo, o problema que resuelve el programa

Cada función debe tener un encabezado que contenga:

- Objetivo de la función y no descripción del procedimiento.
- Comentarios de apoyo a variables, llamadas a función o inclusión de archivos que no sean obvios al proceso.
- Explicación de uso de argumentos (parámetros) no obvios.

Se utilizarán dos tipos de comentarios.

```
/* En caso de ser un  atributo, un método o algún segmento de código  */
```

```
// ..... para el resto de los comentarios
```

Identificadores de variables

Comenzarán siempre con la primera letra minúscula.

Para distinguir palabras dentro del nombre deberá emplearse un guión bajo (_).

Ejemplo:

media_list

Identificadores de punteros (apuntadores)

Su nombre deberá comenzar con la letra p.

Ejemplo: pAudio

Dónde pAudio es un puntero que podrá tener la dirección del lugar donde se almacena la información del audio.

Identificadores de variables dimensionadas (arreglos, matrices)

Su nombre deberá comenzar con las letras ar.

Ejemplo: arUrl

Donde arUrl es un arreglo con la dirección donde se encuentra una media.

Identificadores de funciones

La primera letra deberá ser mayúscula.

Ejemplo: void Funcion ( );

Métodos de acceso a miembros:

Los métodos de acceso a los miembros de las clases se nombrarán “Get” seguido del nombre de la variable con letra inicial mayúscula, para el caso de retorno de valores y “Set” en caso de la actualización de valores.

Ejemplo:

Para la siguiente variable, los métodos de acceso serían:

int random; //variable

int GetRandom (); //Get

void SetRandom (intp_random) //Set

Identificadores de tipos definidos por el usuario

La primera letra será mayúscula.

Ejemplo: class Clase

Sangrías

- Las sangrías tendrán una longitud de tres espacios.

- Las llaves abren y cierran con la misma sangría.

Ejemplo: void Funcion ( )

```
{
    //Instrucciones
}
```

Líneas y espacios en blanco

- No insertar líneas en blanco entre instrucciones.

  Ejemplo:

```
a = b + c;
int f;   //declaración entre instrucciones
f = a;
```

- Las declaraciones de datos dentro de una función deberán ir al inicio
- Deben incluirse espacios en ambos lados de los operadores binarios.

  Ejemplo:

```
y = 50 + 15 - x;
```

2.7 Conclusiones

De manera general se han caracterizado las herramientas y tecnologías a utilizar en la modelación e implementación de la interfaz de comunicación con la librería libvlc, partiendo de las funcionalidades que debe tener la misma. Estudiando de cada una de ellas, sus principales características y las ventajas que proporcionan al ser utilizadas. Se define utilizar como lenguaje de modelado UML, como lenguaje de programación C++ y como entorno de desarrollo integrado Qt Creator con la utilización de la biblioteca Qt 4.5.0.

CAPÍTULO 3: PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En este capítulo se presentará una descripción de la solución propuesta, generándose un diagrama de clases y describiendo las funcionalidades de cada una. Además se mostrarán los principales elementos gráficos utilizados para validar la interfaz propuesta.

3.2 Diagrama de clase de la interfaz de comunicación de la librería libvlc

En el diagrama de clases realizado se observan las relaciones entre las diferentes clases que componen la solución propuesta. Donde la clase QVLC es la responsable de crear el reproductor accediendo a las funcionalidades del resto de las clases.

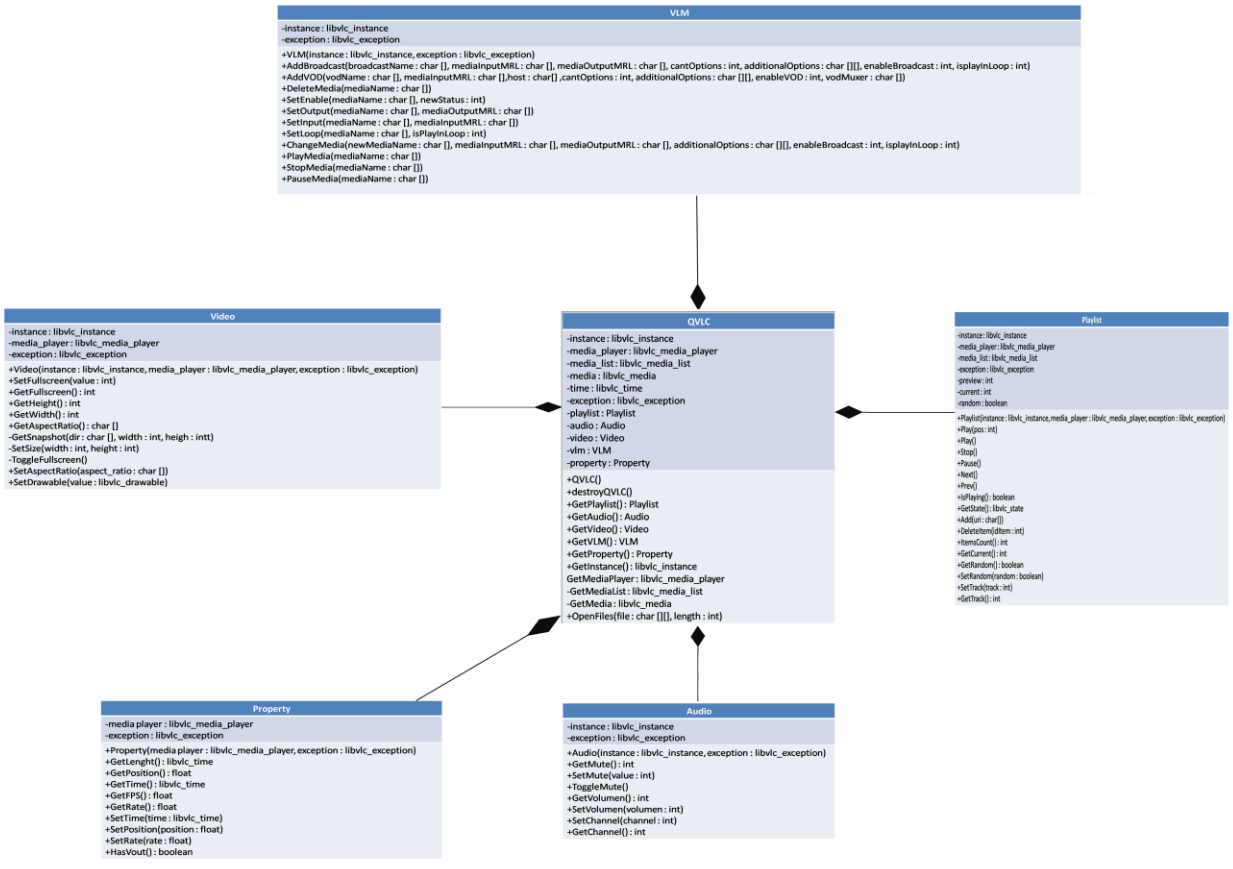


Figura1 Diagrama de clases de la interfaz de comunicación con la librería libvlc

3.2.1 Descripción de las clases de la interfaz de comunicación de la librería libvlc

QVLC: Esta clase es el eje central del reproductor, en ella se tratan los métodos nativos encargados de crear y destruir instancias de la aplicación. Además aparecen en esta clase las instancias del resto de las clases, permitiendo acceder a sus funcionalidades.

VLM: Es la clase encargada de realizar las funcionalidades necesarias para efectuar y controlar la transmisión de medias. Mediante su uso se pueden modificar los parámetros relacionados a un archivo media a transmitir según las características requeridas.

Video: En esta clase se encuentran las funcionalidades encargadas de realizar el tratamiento gráfico del video. Sus métodos permiten personalizar el tamaño de la ventana que representa el video, considerando sus propiedades de largo y ancho. Además hace posible el manejo de la opción de pantalla completa, activándola o desactivándola según se desee.

Audio: Esta clase contiene las funcionalidades encargadas de controlar el volumen de la media en reproducción, adecuándolo según se desee. Incluyendo la opción de activar o desactivar el estado mudo y cambiar el canal de la salida del audio.

Playlist: Es la clase encargada de realizar las funcionalidades necesarias para ejecutar la reproducción de medias y la construcción de una lista de reproducción. Sus funcionalidades permiten modificar según se requiera el estado de reproducción (detención, pausa, reproducción). Además posibilita la opción de reproducir el elemento siguiente o anterior al que se encuentra en reproducción.

Property: Por medio de esta clase se pueden acceder y modificar las propiedades de la media en reproducción, dentro de estas se encuentran: la longitud, la posición, el número de frames por segundo y el tiempo.

3.3 Validación de la interfaz de comunicación con la librería libvlc

Para validar la interfaz propuesta se diseñó un reproductor, que permite interactuar con las funcionalidades de la interfaz. El reproductor podrá transmitir y reproducir media, brindando al usuario un conjunto de opciones a través del panel de botones que se muestra a continuación.

Panel de botones

El diseño del panel de botones del reproductor se muestra en la siguiente figura:

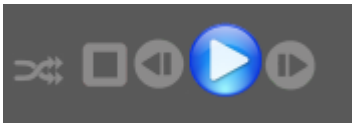


Figura 2 Panel de botones

Los estados de los botones se tratan pulsando con el ratón encima de los mismos. Al presionar un botón, se muestra una imagen distinta a la que aparece cuando no está presionado. Para el botón reproducir se utilizan dos imágenes más que para el resto de los botones, porque cuando el reproductor se encuentra reproduciendo una media, pasa a ser el botón pausa y cuando termina de reproducir vuelve a su estado inicial.

Las imágenes que representan los botones del panel del reproductor son mostradas en la Tabla 4

Habilitado	Presionado	Deshabilitado	Función
			Reproducir
			Pausar
			Más lento
			Más rápido

			Detener
			Siguiente
			Anterior
			Orden aleatorio
			Orden Secuencial

Tabla 4 Imágenes de los botones

La Figura 3 muestra la distribución resultante de los botones en el panel y la Figura 4 el efecto del botón de pausa cuando está presionado.

Figura 3 Distribución de los botones

Figura 4 Botón pausa presionado

Parte gráfica del control volumen

En las Figuras 3 y 4 además del panel de botones, se muestra la parte gráfica del control del volumen, que está formada por un componente “Horizontal Slider” y cuatro imágenes que indican el estado del volumen: mínimo, medio, máximo y mudo.

La Tabla 5 refleja las imágenes utilizadas para el control del volumen.

Horizontal slider	Volumen mínimo	Volumen medio	Volumen máximo	Sin volumen
				

Tabla 5 Imágenes del control de volumen

Control gráfico de la lista de reproducción y transmisión

Las Figuras 5 y 6 muestran la lista de reproducción y transmisión respectivamente. Ambas están compuestas por un componente “ListWidget” que ofrece el nombre de las medias. Al dar doble clic sobre un elemento de la lista, inmediatamente se reproduce o se pone en transmisión la media seleccionada, según la lista que esté activada. En el caso de la lista de reproducción, la media que se encuentre reproduciéndose aparecerá en color azul.

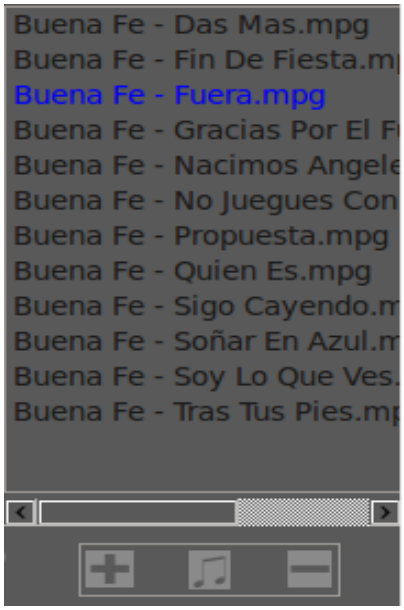


Figura 5 Lista de reproducción

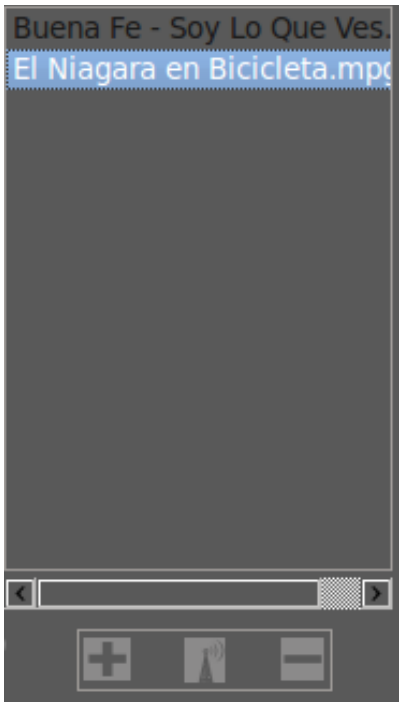


Figura 6 Lista de transmisión

En las figuras anteriores, debajo de las listas de reproducción y transmisión aparecen tres botones, cuyas funcionalidades se describen en la Tabla 6.

Habilitado	Presionado	Deshabilitado	Función
			Añadir una media a la lista seleccionada
			Eliminar una media de la lista seleccionada
			Activar la lista de transmisión
			Activar la lista de reproducción

Tabla 6 Botones de la listas de reproducción y transmisión

La reproducción de un flujo de streaming se realiza especificando la dirección donde se encuentra. Para ello se diseñó una ventana donde se introduce dicha dirección (ver Figura 7). Al igual que para la reproducción de un fichero local, el nombre de la media aparecerá en la lista de reproducción.

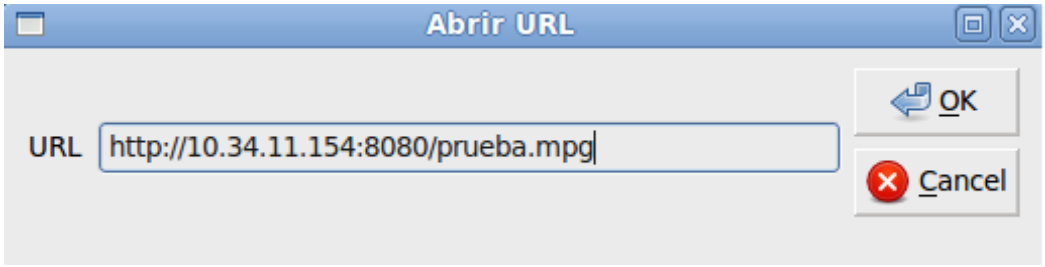


Figura 7 Reproducción de streaming

Para la transmisión de medias se realizó una ventana (ver Figura 8) donde el usuario puede seleccionar el tipo de transmisión, el protocolo mediante el cual se va a transmitir, la dirección y el puerto desde donde se va a emitir, el perfil (formato en que será emitida la media), y pulsando el botón “Abrir” la media que desea transmitir. Una vez seleccionadas estas opciones presionando el botón “Stream” se activa la transmisión.

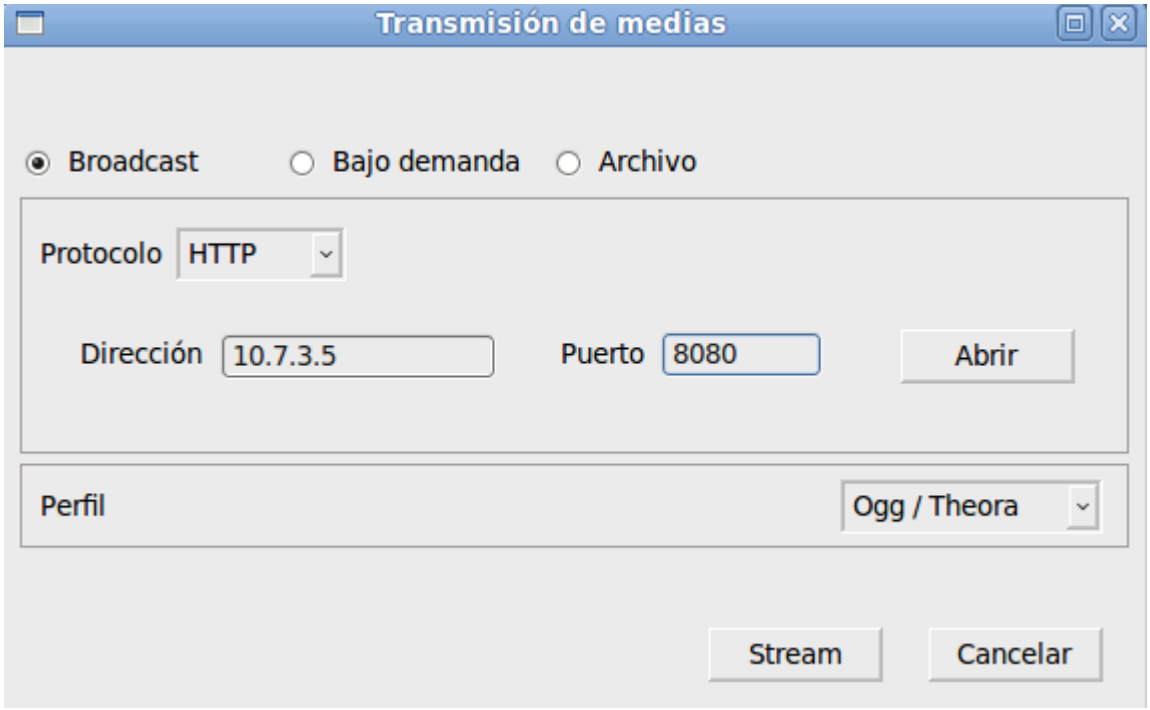


Figura 8 Transmisión de medias

3.4 Conclusiones

En este capítulo se describieron las clases de la interfaz desarrollada para un mayor entendimiento de las mismas. Además se explicaron y mostraron los elementos gráficos del reproductor realizado para probar que la interfaz propuesta cumple con las funcionalidades requeridas.

CONCLUSIONES GENERALES

Con el desarrollo de este trabajo se han estudiado los sistemas que incorporan aplicaciones para la reproducción y transmisión de medias en el departamento de Señales Digitales, lo que ha permitido conocer las razones por las cuales se hace necesario desarrollar la interfaz de comunicación con la librería libvlc. Se ha profundizado en los procesos de reproducción y transmisión de media lo que ha posibilitado conocer su funcionamiento. A partir de la descripción realizada sobre el funcionamiento y las prestaciones de VideoLAN se comprueba que la librería libvlc cumple con los requerimientos necesarios para el desarrollo de las aplicaciones del departamento. El estudio efectuado sobre las funciones para la reproducción y transmisión de media en la librería libvlc ha permitido identificar cuales aplicar para desarrollar una interfaz de comunicación con la librería libvlc. La interfaz realizada permite comunicarse con la librería libvlc y utilizar las funcionalidades que esta librería brinda para los procesos de transmisión y reproducción de media. Con el reproductor desarrollado se comprueba el funcionamiento de la interfaz realizada.

RECOMENDACIONES

El autor del trabajo recomienda:

- Utilizar la interfaz desarrollada para realizar la comunicación con la librería libvlc.
- Continuar la investigación sobre aspectos que no fueron tratados profundamente en el trabajo.
- Realizar implementaciones sobre la misma interfaz para lograr la captura desde dispositivos de audio y video, permitiendo de esta forma incorporar un componente para realizar grabaciones dentro del departamento Señales Digitales.
- Analizar la posibilidad de estructurar la interfaz adaptable a necesidades de un sistema específico, brindando además la incorporación de nuevas funcionalidades.
- Estudiar el sistema de visualización de imágenes de VLC para poder incorporar funcionalidades básicas de procesamiento de imágenes.

BIBLIOGRAFÍA CITADA

1. **Limonta, Ing. Ruperto Sandó.** Integración de los Servidores de Media en las Redes Empresariales. *La Revista BETSIME.*

2. *Los Nuevos medios en la Era digital. Convergencias e industrias de Streaming.* **Sosa, Lic. Leonardo Gabriel.** Salta : s.n., septiembre, 2008.

3. **Jack, Keith.***Dictionary of Video and Television Technology.* United States of America : s.n., 2002.

4. **Velazco, Aylin Estrada and Suárez Pérez, Jean Michael.***Sistema de Captura e Indexación de Video.* La Habana : s.n., 2009.

5. **Alonso Guerrero, Zorilin y Leyva González, Genry.** *Sistema de Captura y Transcripción de Audio.* Ciudad de la Habana : 2009.

6. mundodivx.com. [Online] [Cited: Diciembre 06, 2009.] http://www.mundodivx.com/codecs/index.php.

7. Códecs video y audio. *digitalfotored.* [Online] [Cited: Noviembre 19, 2009.] http://www.digitalfotored.com/videodigital/codecsvideoaudio.htm.

8. **Pellizza, Sergio.***Transmisión de datos.*

9. Características de Streaming. *TELVIR.* [Online] [Cited: Noviembre 19, 2009.] http://oaq.epn.edu.ec/telvir/pagina/contenido/streaming.htm.

10. **Muñoz, Ingrid Capell.** Sistema estereoscópico para tele operación asistida y supervisión de tareas robotizadas. Octubre 2006.

11. **Sotil, Walter and Calvo, Néstor.** TEXTURAS TRANSPARENTES PARA VISUALIZACION DE MALLAS Y DATOS. noviembre 2008.

12. **Tojar, Luis García.** La ideología de Star Wars. 2005.

13. **Rodríguez Huertas, Rosa, y otros, y otros.***APLICACIÓN INFORMÁTICA INTEGRADA DE RECURSOS DIDÁCTICOS PARA ESTADÍSTICA E I.O.* s.l. : Universidad de Oviedo, 2006.

14. **Márquez, Juan Manuel.** SISTEMAS DE INFORMACION. Marzo, 2003.

15. **Carrera, Sara carrera.** Adquisición semi-automática del conocimiento:una arquitectura preliminar. 2007.

16. **Rodríguez, María Pilar Rondon.** Algoritmos y Programación.

17. CPlusPlus.com. [Online] [Cited: Febrero 27, 2010.] http://www.cplusplus.com/info/description/.

18. Anjuta DevStudio: GNOME Integrated Development Environment. [Online] [Cited: marzo 8, 2010.] http://www.anjuta.org.

19. GNOME. *GNOME.* [Online] [Cited: marzo 7, 2010.] http://www.guia-ubuntu.org/index.php?title=GNOME.

20. **Venegas, Carlos.** Depuracion avanzada de Programas.

21. Code:: Blocks. [Online] [Cited: marzo 8, 2010.] http://codeblocks.org/.

22. QT. [Online] [Cited: Febrero 28, 2010.] http://qt.nokia.com/products.

23. **Gamma, Erich, y otros.** *Patrones de diseño.* Oviedo

BIBLIOGRAFÍA CONSULTADA

1. **Limonta, Ing. Ruperto Sandó.** Integración de los Servidores de Media en las Redes Empresariales. *La Revista BETSIME.*

2. **Alonso Guerrero, Zorilin y Leyva González, Genry.** *Sistema de Captura y Transcripción de Audio.* Ciudad de la Habana : 2009.

3. *Los Nuevos medios en la Era digital. Convergencias e industrias de Streaming.* **Sosa, Lic. Leonardo Gabriel.** Salta : s.n., septiembre, 2008.

4. **Jack, Keith.***Dictionary of Video and Television Technology.* United States of America : s.n., 2002.

5. **Velazco, Aylin Estrada and Suárez Pérez, Jean Michael.***Sistema de Captura e Indexación de Video.* La Habana : s.n., 2009.

6. mundodivx.com. [Online] [Cited: Diciembre 06, 2009.] http://www.mundodivx.com/codecs/index.php.

7. Códecs video y audio. *digitalfotored.* [Online] [Cited: Noviembre 19, 2009.] http://www.digitalfotored.com/videodigital/codecsvideoaudio.htm.

8. **Pellizza, Sergio.***Transmisión de datos.*

9. Características de Streaming. *TELVIR.* [Online] [Cited: Noviembre 19, 2009.] http://oaq.epn.edu.ec/telvir/pagina/contenido/streaming.htm.

10. **Muñoz, Ingrid Capell.** Sistema estereoscópico para tele operación asistida y supervisión de tareas robotizadas. Octubre 2006.

11. **Sotil, Walter and Calvo, Néstor.** TEXTURAS TRANSPARENTES PARA VISUALIZACION DE MALLAS Y DATOS. noviembre 2008.

12. **Tojar, Luis García.** La ideología de Star Wars. 2005.

13. **Rodríguez Huertas, Rosa, y otros, y otros.***APLICACIÓN INFORMÁTICA INTEGRADA DE RECURSOS DIDÁCTICOS PARA ESTADÍSTICA E I.O.* s.l. : Universidad de Oviedo, 2006.

14. **Márquez, Juan Manuel.** SISTEMAS DE INFORMACION. Marzo, 2003.

15. **Carrera, Sara carrera.** Adquisición semi-automática del conocimiento:una arquitectura preliminar. 2007.

16. **Ferroggiaro, Federico J.** Lecturas sobre computadoras digitales.

17. **Guerra, Llanes Néstor, et al., et al.**Web Application “Integral Rehabilitation System".

18. **Polanco, Rosell Pupo and Pérez, González Yenier.** Implementación del componente réplica de base de datos para Akademos v2.0. Junio 2009.

19. **Rodríguez, María Pilar Rondon.** Algoritmos y Programación.

20. CPlusPlus.com. [Online] [Cited: Febrero 27, 2010.] http://www.cplusplus.com/info/description/.

21. Anjuta DevStudio: GNOME Integrated Development Environment. [Online] [Cited: marzo 8, 2010.] http://www.anjuta.org.

22. GNOME. *GNOME.* [Online] [Cited: marzo 7, 2010.] http://www.guia-ubuntu.org/index.php?title=GNOME.

23. **Venegas, Carlos.** Depuracion avanzada de Programas.

24. Code:: Blocks. [Online] [Cited: marzo 8, 2010.] http://codeblocks.org/.

25. QT. [Online] [Cited: Febrero 28, 2010.] http://qt.nokia.com/products.

26. **Gamma, Erich, y otros.** *Patrones de diseño.* Oviedo .

27. Visual Paradigm for UML. [Online] [Cited: Marzo 03, 2010.] http://www.visual-paradigm.com/product/vpuml/.

28. VideoLAN - VLC media player. *VideoLAN.* [Online] VideoLAN. [Cited: 01 18, 2010.] https://www.videolan.org/.

29. VideoLan.org. [Online] [Cited: 11 23, 2009.] http://www.videolan.org/videolan/.

30. **Leteurtre, Tristan.***VideoLAN Server developer.* 2002.

31. Rational Rose Enterprise. [Online] [Cited: Marzo 02, 2010.]

http://www.rational.com.ar/herramientas/roseenterprise.html.

32. **Recasens, Llúıs Martínez.***Diseño e Implementación de Módulos para un Sistema de TV P2P.* abril, 2008.

33. **Pasaje, Julio Luis Medina.** Metodología y Herramientas UML para el Modelado y Análisis de Sistemas de Tiempo Real Orientados a Objetos. Junio, 2005.

34. **Hernández León, Rolando Alfredo and Coello González, Sayda.***El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : EDUNIV, 2002. 959-16-0343-6.

35. ExternalAPI. *ExternalAPI.* [Online] VideoLAN. [Cited: 01 15, 2010.] http://wiki.videolan.org/ExternalAPI.

36. **Ciudad Ricardo, Febe Angel.***¿CÓMO CONFECCIONAR UN ESTADO DEL ARTE?* Ciudad de la Habana : s.n., 2008.

37. **Austerberry, David.***The Technology of Video and Audio Streaming.* s.l. : Focal Press, 2005. 0-240-80580-1..

38. *Normas ISO de codificación de contenidos.*

Anexo I. Funciones para la reproducción de media de la librería libvlc

libvlc_media_player_t* libvlc_media_player_new (libvlc_instance_t*,libvlc_exception_t *)

La función libvlc_media_player_new crea un objeto media player vacío, a esta función se le pasa como parámetros una instancia libvlc y un puntero excepción. Devuelve el objeto creado.

libvlc_media_player_t*libvlc_media_player_new_from_media(libvlc_media_t *, libvlc_exception_t *)

La función libvlc_media_player_new_from_media tiene como objetivo crear un objeto media player de media, pasándole por parámetro un objeto media y un puntero excepción.

void libvlc_media_player_next_frame (libvlc_media_player_t * p_input, libvlc_exception_t * p_e)

La función libvlc_media_player_next_frame muestra el fotograma siguiente, pasándole como parámetro a la función un objeto media player y un puntero excepción.

int libvlc_media_player_will_play (libvlc_media_player_t * , libvlc_exception_t *)

La función libvlc_media_player_will_play se le pasa como parámetro un objeto media player y un puntero excepción, y su objetivo es decidir si el archivo puede ser reproducido o no. Si devuelve 1 puede reproducirse, si es 0 no se reproduce.

void libvlc_media_player_set_position (libvlc_media_player_t * , float , libvlc_exception_t *)

La función libvlc_media_player_set_position se le pasa como parámetro un objeto media player, la nueva posición y un puntero excepción. El objetivo de esta función es cambiar la posición de la reproducción en curso.

void libvlc_media_player_next_chapter (libvlc_media_player_t * ,libvlc_exception_t*)

A la función libvlc_media_player_next_chapter se le pasa como parámetro un objeto media player y un puntero excepción, su objetivo es poner a reproducir el siguiente elemento de la lista de reproducción con respecto al actual.

void libvlc_media_player_previous_chapter (libvlc_media_player_t *, libvlc_exception_t *)

A la función libvlc_media_player_previous_chapter se le pasa como parámetro un objeto media player y

un puntero excepción, su objetivo es poner a reproducir el elemento anterior de la lista de reproducción con respecto al actual.

void libvlc_media_player_set_rate (libvlc_media_player_t * , float , libvlc_exception_t *)

La función libvlc_media_player_set_rate se le pasa como parámetro un objeto media player, nueva velocidad de reproducción y un puntero excepción, su objetivo es cambiar la velocidad de reproducción de la media.

void libvlc_media_player_set_time (libvlc_media_player_t *, libvlc_time_t , libvlc_exception_t *)

La función libvlc_media_player_set_time se le pasa como parámetro un objeto media player, el nuevo tiempo de la reproducción y un puntero excepción, su objetivo es cambiar el tiempo de la reproducción.

int libvlc_media_player_is_playing (libvlc_media_player_t * , libvlc_exception_t *)

La función libvlc_media_player_is_playing se le pasa como parámetro un objeto media player y un puntero excepción, su objetivo es comprobar si hay alguna reproducción en curso. Si es 1 hay alguna reproducción en curso, si es 0, lo contrario.

libvlc_state_t* libvlc_media_player_get_state (libvlc_media_player_t * , libvlc_exception_t *)

La función libvlc_media_player_get_state se le pasa como parámetro un objeto media player y un puntero excepción, su objetivo es devolver el estado (reproducción, pausa o detenido) actual de la película en reproducción.

float libvlc_media_player_get_rate (libvlc_media_player_t * , libvlc_exception_t *)

La función libvlc_media_player_get_rate se le pasa como parámetro un objeto media player y un puntero excepción, su objetivo es devolver la velocidad de reproducción de la media.

float libvlc_media_player_get_position (libvlc_media_player_t *, libvlc_exception_t*)

La función libvlc_media_player_get_position se le pasa como parámetro un objeto media player y un puntero excepción, su objetivo es devolver la posición de la película en reproducción.

libvlc_time_t libvlc_media_player_get_length (libvlc_media_player_t * , libvlc_exception_t *)

La función libvlc_media_player_get_length se le pasa como parámetro un objeto media player y un puntero excepción, su objetivo es devolver la duración de la película en reproducción.

Anexo II. Funciones para la transmisión de media de la librería libvlc.

void libvlc_vlm_release (libvlc_instance_t *, libvlc_exception_t *)

La función libvlc_vlm_release tiene como objetivo liberar la instancia vlm creada con respecto a la instancia libvlc creada. A esta función se le pasa por parámetros un objeto libvlc y un puntero excepción.

void libvlc_vlm_set_enabled (libvlc_instance_t *,const char *, int, libvlc_exception_t *)

La función libvlc_vlm_set_enabled tiene como objetivo activar o desactivar la transmisión de una media. Esta función tiene como parámetros un objeto libvlc, el nombre de la media, un entero (si es 1 activa la transmisión y si su valor es 0 lo contrario), como último parámetro un puntero excepción.

void libvlc_vlm_set_loop (libvlc_instance_t *, const char *, int, libvlc_exception_t *)

La función libvlc_vlm_set_loop tiene como objetivo poner la transmisión de la media a reproducción constante. Esta función tiene como parámetros un objeto libvlc, el nombre de la media, un entero (si es 1 activa la transmisión a reproducción contante y si su valor es 0 lo contrario), como último parámetro un puntero excepción.

void libvlc_vlm_change_media (libvlc_instance_t *, const char *, const char *, const char *, int, const char *const *, int, int, libvlc_exception_t *)

La función libvlc_vlm_change_media tiene como objetivo cambiar los parámetros de la media a transmitir. Los parámetros de esta función son: el objeto libvlc, nombre del broadcast, los parámetros de entrada, los parámetros de salida, número de opciones adicionales, opciones adicionales, si se activa o no la transmisión, si la reproducción será constante o no, y un puntero excepción.

void libvlc_vlm_set_output (libvlc_instance_t *, const char *, const char *, libvlc_exception_t *)

La función libvlc_vlm_set_output tiene como objetivo editar los parámetros de salida de la media en transmisión. Los parámetros que se le pasan a esta función son: un objeto libvlc, el nombre de la media, los nuevos parámetros de salida y un puntero excepción.

void libvlc_vlm_set_input (libvlc_instance_t *, const char *, const char *, libvlc_exception_t *)

La función libvlc_vlm_set_input tiene como objetivo editar los parámetros de entrada de la media en transmisión. Los parámetros que se le pasan a esta función son: un objeto libvlc, el nombre de la media, los nuevos parámetros de entrada y un puntero excepción.

void libvlc_vlm_seek_media (libvlc_instance_t *, const char *, float, libvlc_exception_t *)

La función libvlc_vlm_seek_media tiene como objetivo cambiar la reproducción de una media a una posición dada. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media, la posición donde comenzará a reproducir y un puntero excepción.

constchar * libvlc_vlm_show_media (libvlc_instance_t *, const char *, libvlc_exception_t *)

La función libvlc_vlm_show_media tiene como objetivo devolver información sobre la media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media y un puntero excepción.

float libvlc_vlm_get_media_instance_position (libvlc_instance_t *, const char *, int, libvlc_exception_t *)

La función libvlc_vlm_get_media_instance_position tiene como objetivo devolver la posición de la media en reproducción. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media, el identificador de la media y un puntero excepción.

int libvlc_vlm_get_media_instance_time (libvlc_instance_t *, const char *, int, libvlc_exception_t *)

La función libvlc_vlm_get_media_instance_time tiene como objetivo devolver el tiempo de la media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media, el identificador de la media y un puntero excepción.

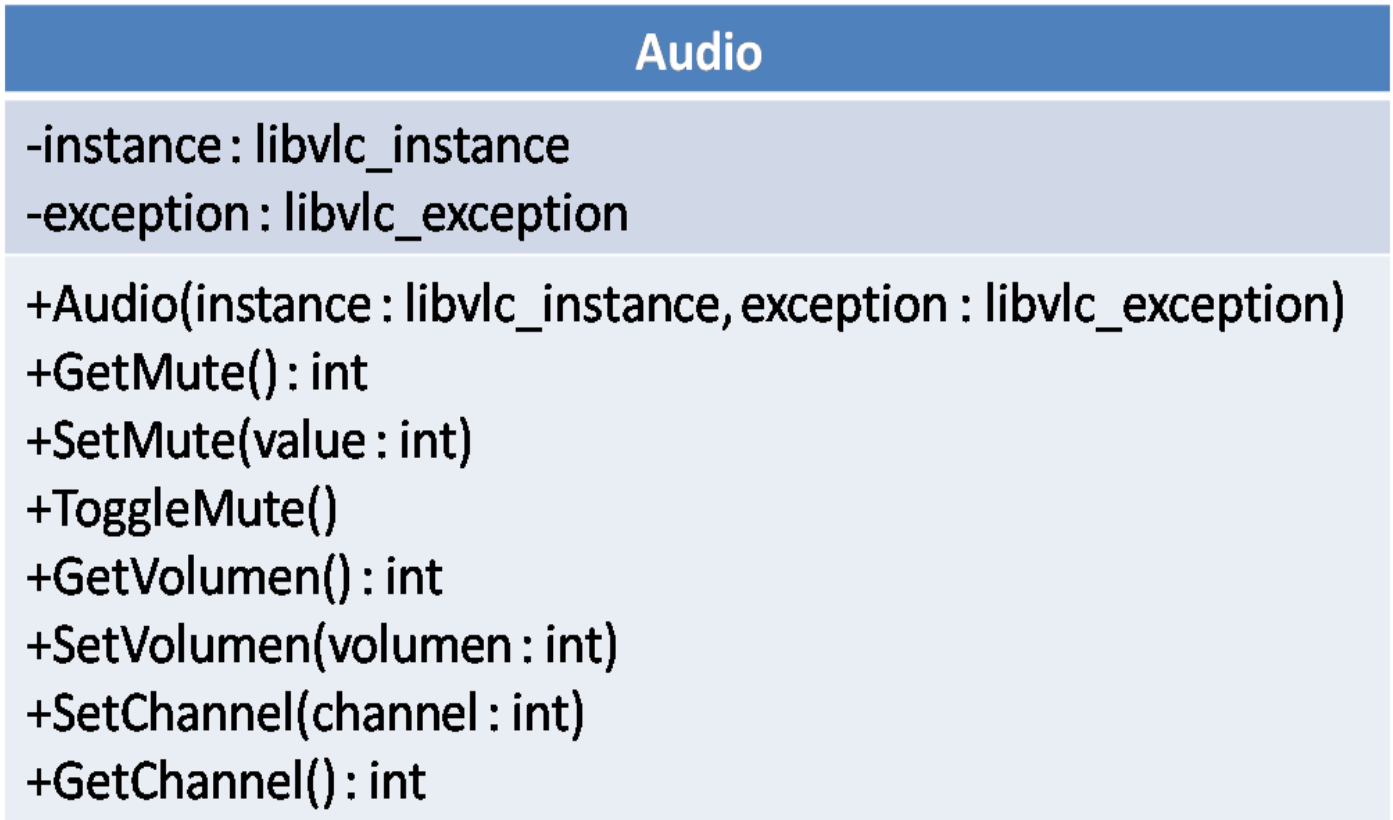
int libvlc_vlm_get_media_instance_length (libvlc_instance_t *,const char *, int, libvlc_exception_t *)

La función libvlc_vlm_get_media_instance_length tiene como objetivo devolver la longitud de la media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media, el identificador de la media y un puntero excepción.

int libvlc_vlm_get_media_instance_rate (libvlc_instance_t *, const char *, int, libvlc_exception_t *)

La función libvlc_vlm_get_media_instance_rate tiene como objetivo devolver la velocidad de la reproducción de la media. Los parámetros que tiene esta función son: un objeto libvlc, el nombre de la media, el identificador de la media y un puntero excepción.

Anexo III. Clases de la interfaz de comunicación con la librería libvlc



Property

-media player : libvlc_media_player
-exception : libvlc_exception

+Property(media player : libvlc_media_player, exception : libvlc_exception)
+GetLenght() : libvlc_time
+GetPosition() : float
+GetTime() : libvlc_time
+GetFPS() : float
+GetRate() : float
+SetTime(time : libvlc_time)
+SetPosition(position : float)
+SetRate(rate : float)
+HasVout() : boolean

VLM

-instance : libvlc_instance
-exception : libvlc_exception

+VLM(instance : libvlc_instance, exception : libvlc_exception)
+AddBroadcast(broadcastName : char [], mediaInputMRL : char [], mediaOutputMRL : char [], cantOptions : int, additionalOptions : char [][], enableBroadcast : int, isplayInLoop : int)
+AddVOD(vodName : char [], mediaInputMRL : char [],host : char[] ,cantOptions : int, additionalOptions : char [][], enableVOD : int, vodMuxer : char [])
+DeleteMedia(mediaName : char [])
+SetEnable(mediaName : char [], newStatus : int)
+SetOutput(mediaName : char [], mediaOutputMRL : char [])
+SetInput(mediaName : char [], mediaInputMRL : char [])
+SetLoop(mediaName : char [], isPlayInLoop : int)
+ChangeMedia(newMediaName : char [], mediaInputMRL : char [], mediaOutputMRL : char [], additionalOptions : char [][], enableBroadcast : int, isplayInLoop : int)
+PlayMedia(mediaName : char [])
+StopMedia(mediaName : char [])
+PauseMedia(mediaName : char [])

QVLC
-instance : libvlc_instance -media_player : libvlc_media_player -media_list : libvlc_media_list -media : libvlc_media -time : libvlc_time -exception : libvlc_exception -playlist : Playlist -audio : Audio -video : Video -vlm : VLM -property : Property
+QVLC() +destroyQVLC() +GetPlaylist() : Playlist +GetAudio() : Audio +GetVideo() : Video +GetVLM() : VLM +GetProperty() : Property +GetInstance() : libvlc_instance GetMediaPlayer : libvlc_media_player -GetMediaList : libvlc_media_list -GetMedia : libvlc_media +OpenFiles(file : char [][], length : int)

Playlist
-instance : libvlc_instance -media_player : libvlc_media_player -media_list : libvlc_media_list -exception : libvlc_exception -preview : int -current : int -random : boolean
+Playlist(instance : libvlc_instance, media_player : libvlc_media_player, exception : libvlc_exception) +Play(pos : int) +Play() +Stop() +Pause() +Next() +Prev() +IsPlaying() : boolean +GetState() : libvlc_state +Add(uri : char[]) +DeleteItem(idItem : int) +ItemsCount() : int +GetCurrent() : int +GetRandom() : boolean +SetRandom(random : boolean) +SetTrack(track : int) +GetTrack() : int

Video
-instance : libvlc_instance -media_player : libvlc_media_player -exception : libvlc_exception
+Video(instance : libvlc_instance, media_player : libvlc_media_player, exception : libvlc_exception) +SetFullscreen(value : int) +GetFullscreen() : int +GetHeight() : int +GetWidth() : int +GetAspectRatio() : char [] -GetSnapshot(dir : char [], width : int, heigh : intt) -SetSize(width : int, height : int) -ToggleFullscreen() +SetAspectRatio(aspect_ratio : char []) +SetDrawable(value : libvlc_drawable)

GLOSARIO DE TÉRMINOS

Ancho de banda: Cantidad de datos que se pueden transmitir en una unidad de tiempo través de la red.

Archivos multimedia: Archivos de audio, video o imagen.

Captura: Proceso por el cual se obtiene información de un dispositivo externo (Cámara, webcam, etc.) y se almacena o se observa en la computadora.

Codificación: Convertir a un formato de video determinado.

Darwin: Servidor de streaming que permite enviar a los clientes streaming de medios a través de la red.

Framerate: Cantidad de frame (imágenes) por segundo, se expresa en (fps).

Interfaz: Zona de contacto o conexión entre dos componentes de "hardware"; entre dos aplicaciones, o entre un usuario y una aplicación.

Librería: Conjunto de subprogramas utilizados para desarrollar software.

Media: Película, imagen o cualquier otro material audio visual que requiere de un uso especial de equipamiento para visualizarlo.

Mencoder: Codificador de vídeo libre liberado bajo licencia GPL.

Módulo: Parte de un programa de una computadora que realiza una o varias tareas específicas.

MPlayer: Reproductor multimedia multiplataforma.

Reproductor: Programa o control de cliente que recibe contenido multimedia digital transmitido desde un servidor o reproducido a partir de archivos locales.

TelNet: (Telecomunication Network) es el nombre de un protocolo de red (y del programa informático que implementa el cliente), sirve para acceder mediante una red a otra máquina.

Transcodificación: Es la conversión directa (de digital a digital) de un códec a otro.