ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE APLICACIONES INTERACTIVAS BASADAS EN GINGA-NCL PARA TELEVISIÓN DIGITAL ENFOCADAS EN LA TEMÁTICA DEL MEDIO AMBIENTE

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

JAIME FABRICIO GUZMÁN HERRERA
jaimeguzmanh@gmail.com

DIRECTOR: ING. DAVID RAÚL MEJÍA NAVARRETE, MSc.
david.mejia@epn.edu.ec

CODIRECTOR: IVÁN MARCELO BERNAL CARRILLO, Ph.D.
ivan.bernal@epn.edu.ec

Quito, Agosto 2018

DECLARACIÓN

Yo, Jaime Fabricio Guzmán Herrera, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en el documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Jaime Fabricio Guzmán Herrera

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Jaime Fabricio Guzmán Herrera bajo nuestra supervisión.

_____	_____
Ing. David Mejía MSc.	Iván Bernal Ph.D.
DIRECTOR DEL PROYECTO	CODIRECTOR DEL PROYECTO

AGRADECIMIENTO

Ante todo, agradezco a Dios y a la Virgen Dolorosa por permitirme culminar de la mejor manera este logro importante dentro de mi vida académica.

Agradezco a mis padres, Jaime y Susana, por haberme inculcado el valor de la responsabilidad y por entregarme de manera incondicional todo su amor, entereza y esfuerzo durante cada etapa de mi vida.

Agradezco a mi esposa Ailed, por ser mi guía, mi compañera y mi fortaleza durante cada paso que he dado para llegar a cumplir cada objetivo de mi vida.

A mi hermano Andrés, por todos sus consejos y palabras sinceras de aliento.

Un agradecimiento especial a mi director y codirector, Ing. David Mejía MSc e Iván Bernal Ph.D. respectivamente, por su apoyo y paciencia durante todo el desarrollo de este Trabajo de Titulación.

Un agradecimiento final para mis amigos Ernest, Pablín, Mono, Deivid, Mauro, Esteban, Pepe Lucho, Roberth, Eduardo, Dianita, Alejita, Majito C y Majito R, por enseñarme el valor de la amistad y poder compartir con ustedes momentos agradables.

DEDICATORIA

Este trabajo se lo dedico a mis padres, a mi hermano y a mi esposa, por ser los pilares fundamentales de mi vida y porque estoy seguro de que serán las personas que estarán conmigo apoyándome en cada objetivo que me proponga.

Jaime Fabricio

CONTENIDO

CAPÍTULO 1

ÍNDICE DE FIGURAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

ÍNDICE DE TABLAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 1

1. FUNDAMENTOS TEÓRICOS

1.1. INTRODUCCIÓN

La inserción de la Televisión Digital Terrestre (TDT) dentro de la sociedad actual introduce un giro radical, tal como se dio en el pasado cuando se implantó la televisión a color en vez de los programas en blanco y negro. A más de conseguir una mejor calidad de recepción de los diversos programas televisivos, se plantea dejar de lado la pasividad del televidente, y que este forme parte de un ambiente mucho más participativo por medio de recursos multimedia y de diversas aplicaciones interactivas.

La TDT surge debido a que, en la actualidad, con la televisión analógica, se desperdicia mucho espectro electromagnético debido a la interferencia intersímbolo (que viene a ser un gran problema al momento de la recepción), produciendo así un mayor gasto en cuanto al número de estaciones transmisoras.

El ancho de banda de un canal analógico es de 6 MHz y lo que la TDT pretende es transmitir varios canales de televisión digital dentro de este mismo espacio. Para ello se utilizan técnicas de compresión de audio y video, y dependiendo de la velocidad de transmisión, se pueden obtener canales de alta definición (HD) y canales de definición estándar (SD).

1.2. TELEVISIÓN DIGITAL

La Televisión Digital Terrestre se basa en la propagación de señales de audio y video, incluyendo datos, que utilizan señales digitalizadas y codificadas, con el fin de revolucionar el concepto tradicional de televisión, y permite al televidente obtener una mayor inclusión social dentro de este espacio, mediante su interactividad con dicho medio, a través de la creación de diversas aplicaciones interactivas a nivel familiar, profesional y empresarial.

En la Figura 1.1 se observa un diagrama de la forma en la que son enviadas las señales de audio y video para la televisión analógica y digital respectivamente.

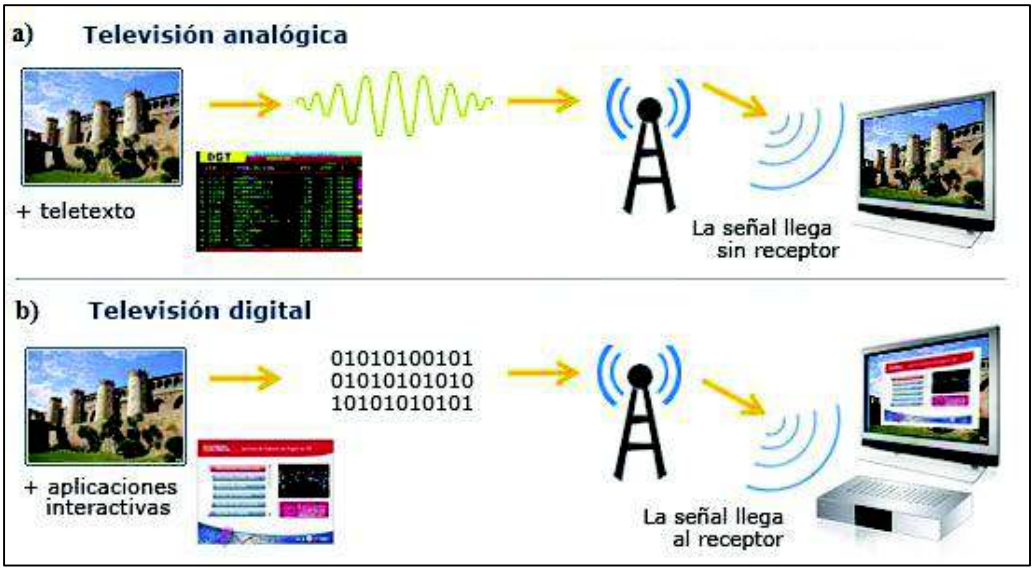

Figura 1.1 Señales de transmisión de audio y video
a) Televisión analógica; b) Televisión digital
Fuente: [1]

El sistema de codificación que se utiliza para la TDT es una señal de video transmitida en formato binario, es decir usando “0” y “1”. Para que una señal analógica (que utiliza la televisión tradicional) sea convertida a una señal digital, debe seguir un proceso en el que intervenga un conversor análogo/digital, el cual transforma la señal televisiva en una cadena codificada de bits, cuyo objetivo es presentar una señal de alta calidad, sin degradación de la misma y robusta frente al ruido.

Por otra parte, la cantidad de bits que se presenta al final del proceso análogo/digital es tan alto, que su transporte no es viable y su almacenamiento conlleva un consumo excesivo de recursos. Sin embargo, la información que contiene el *stream* de bits generados posee un mayor contenido que el ojo humano puede percibir, es decir, posee información redundante considerable. Por tal razón esta redundancia es reducida por métodos de compresión digital, hasta llegar a niveles apropiados que permitan su transporte de manera eficiente y un consumo óptimo de recursos.

1.2.1. CARACTERÍSTICAS DE LA TELEVISIÓN DIGITAL

El uso de la TDT frente a la televisión tradicional trae consigo algunas características importantes y una serie de beneficios. Entre las características más notables se pueden citar a las siguientes:

- Su infraestructura es sencilla y poco costosa debido a que se usa la red terrestre para la transmisión de sus señales, por lo que se puede usar la misma técnica de recepción de la televisión analógica, con la adición de un decodificador *Set Top Box*, sin alterar la calidad de las imágenes enviadas digitalmente.
- El sistema TDT ofrece recepción en movimiento y portátil.
- Se puede utilizar la red de frecuencia única (SFN, *Single Frequency Network*), para realizar transmisiones de una misma señal en el mismo canal de frecuencia, con el fin de lograr un grado mayor de eficiencia del espectro en las bandas de televisión. Más información se la puede obtener en [2].
- Trabaja con una potencia de transmisión menor, comparado con la televisión tradicional.
- La señal digital posee una mayor robustez frente al ruido, la propagación multitrayecto e interferencias, permitiendo una mejor calidad de la imagen.
- Ofrece un mayor número de programas, permitiendo en estos una interactividad directa con el televidente.
- El sistema también introduce el servicio de transmisión móvil terrestre de audio/video digital denominado 1seg (*One seg*). Esta característica es exclusiva para el estándar ISDB-T.
- Permite la transmisión de un canal en alta definición (HDTV) y un canal para teléfonos móviles en un ancho de banda de 6 MHz, reservado para transmisiones de TV analógicas. Además, se puede seleccionar entre dos y tres canales de televisión en definición estándar (SDTV) en lugar de uno solo en HDTV, utilizando la técnica de multiplexación de canales. Cabe mencionar que la combinación de estos servicios puede cambiarse en cualquier momento.

- Existe un incremento en la relación de aspecto. El formato convencional de la televisión analógica es de 4:3, a diferencia de la televisión digital, que define un formato panorámico de 16:9.
- El sistema ofrece la posibilidad de integrarse con facilidad a las TIC (Tecnologías de la Información y Comunicación), porque el televisor puede convertirse en un terminal multimedia que entregue al televidente audio, video y datos, permitiendo así una convergencia entre el televisor, el computador y varios dispositivos de red.

En la Figura 1.2 se indican las distintas combinaciones de los formatos de televisión que se pueden obtener en un canal de 6 MHz de ancho de banda.

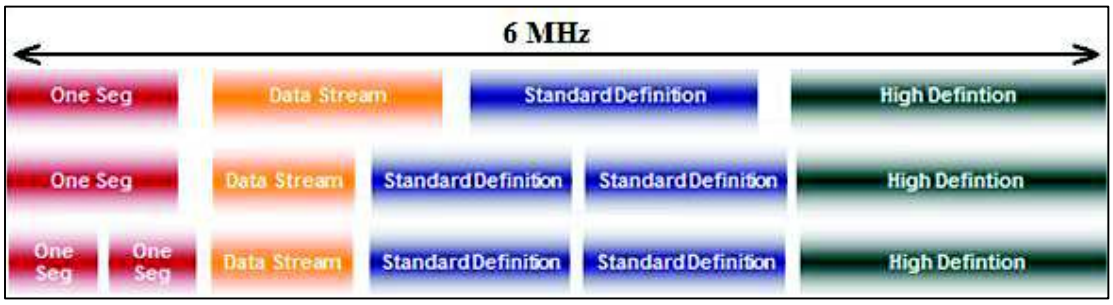

Figura 1.2 Alternativas de uso de canales de televisión en un canal de 6 MHz

1.2.2. ESTÁNDAR SBTVD (SISTEMA BRASILEIRO DE TELEVISÃO DIGITAL)

Es un sistema creado en Brasil que se basa en el estándar japonés ISDB-T y comúnmente es denominado estándar ISDB-Tb. Es utilizado en la mayoría de países de América del Sur y su objetivo, aparte de mejorar aspectos técnicos, es lograr la inclusión de la población a una sociedad de la información, aprovechando que existe al menos una televisión por cada familia.

Los requerimientos técnicos que se consideraron para la creación de este estándar son:

- Televisión en formato 3D
- Alta definición

- Televisión interactiva con el usuario
- Televisión móvil y portátil con señal de calidad
- Recepción robusta de señales digitales en ambientes interiores y exteriores

Los requerimientos sociales que se tomaron en cuenta en este estándar son:

- La integración e inclusión digital de la sociedad
- Apoyo a la educación por medio de la televisión digital a través de contenidos especializados y programas interactivos
- Difusión cultural

Básicamente los cambios realizados en este estándar con respecto al estándar japonés se enfocan principalmente en el *middleware* (Ginga) y en la compresión de video (MPEG-4 parte 10 o H.264) [3].

En la Tabla 1.1 se presentan las principales características de los estándares japonés y brasileño.

	Estándar	
	ISDB-T	**ISDB-TB (SBTVD)**
Aspecto	4:3 y 16:9	4:3 y 16:9
Middleware	ARIB	GINGA
Compresión de audio	MPEG-2 Audio (AAC)	MPEG-2 Audio (AAC)
Compresión de video	MPEG-2 Video	MPEG-4 AVC / H.264
Ancho de banda	6 MHz	6 MHz

Tabla 1.1 Comparación entre el estándar ISDB-T y el estándar ISDB-Tb (SBTVD)

1.3. INTERACTIVIDAD [4]

Es una metodología que se usa en su máxima expresión dentro del entorno de la Televisión Digital Terrestre, puesto que se utiliza para ofrecer información y contenidos dinámicos mientras se transmite un programa televisivo, permitiendo

de esta manera una interrelación más directa y cercana entre el televidente y el televisor; pudiendo el televidente decidir cuándo usar dichos contenidos. La interactividad como tal es la que permite una evolución vertiginosa de la TDT porque intenta que el televidente tome un rol más activo frente a esta nueva era de la televisión por medio de participaciones en encuestas, votaciones, mecanismos de salud y educación, entre otros.

Por medio de la interactividad, el televidente tiene la posibilidad de personalizar la información que se presenta en la pantalla del televisor. Dicha personalización se la puede introducir por medio de algunas herramientas de desarrollo que han sido creadas justamente para esto. Este contenido puede ser enviado durante el proceso de transmisión y almacenarse en el decodificador (receptor) o recurrir al uso del canal de retorno bidireccional para intercambiar información entre el transmisor y receptor. De acuerdo al tipo de contenido que se desee presentar al televidente existen algunos tipos o clases de interactividad. En otras palabras, la interactividad que se ofrece al televidente puede ser tipificada con los contenidos enviados y puede ser visto cuando el televidente lo requiera.

Las ventajas que ofrece la interactividad, a más de lo indicado en párrafos anteriores, es que permite complementar las aplicaciones interactivas de la televisión con servicios públicos o privados, servicios comerciales, y demás sistemas que con la televisión tradicional no se podía realizar, y se debía recurrir al uso del computador y otros elementos que tengan acceso a redes. También permite sincronizar estas aplicaciones con dispositivos móviles y portátiles, para tener una mejor interacción desde el punto de vista del usuario final. Entonces es importante aclarar que un programa de televisión interactivo no es considerado un programa de televisión ni tampoco un programa de computación, sino más bien se considera como un estado intermedio entre estos dos extremos citados.

1.3.1. INTERACTIVIDAD LOCAL

Este tipo de interactividad se caracteriza porque solamente trabaja con aplicaciones que son transmitidas por *broadcast* y son descargadas en el *Set Top*

Box, es decir, en este caso el televidente logra interactuar con la información una vez que esta se encuentre almacenada en el receptor.

Al usar este tipo de interactividad, el usuario no podrá enviar datos de retorno, únicamente tiene la opción de acceder a todos los contenidos que la aplicación presenta. Un ejemplo de aplicaciones más comunes son las guías de programación de cada estación televisiva o la observación de teletexto digital que se puede añadir a un programa cualquiera, además pueden encontrarse juegos o noticias importantes.

1.3.2. INTERACTIVIDAD COMPLETA O REMOTA

Para efectuar este tipo de interactividad se necesita obligatoriamente tener un canal de retorno el cual permite la comunicación entre el receptor y el proveedor del servicio interactivo. Los tipos de canal de retorno usualmente empleados son las líneas *Dial up*, Ethernet, ADSL (*Asymmetric Digital Subscriber Line*), ISDN (*Integrated Services Digital Network*), redes celulares, tecnologías WiMAX (*Worldwide Interoperability for Microwave Access*) y WiFi (*Wireless Fidelity*), entre los más importantes.

Al momento de utilizar la interactividad remota entre las aplicaciones interactivas, el televidente, a más de navegar a través del contenido que pueden presentar estas aplicaciones, tiene la opción adicional de enviar respuestas hacia las estaciones televisivas o incluso hacia otros usuarios, puede participar en encuestas o votaciones, es decir, es en este punto en donde el usuario toma un mayor protagonismo.

En otras palabras, el usuario deja de ser una persona pasiva, que únicamente se dedica a ver programas televisivos, a formar parte activa de todas las interacciones y opciones que las aplicaciones interactivas pueden presentar en los diferentes programas televisivos.

1.3.2.1. Líneas Dial up

Se usa comúnmente con televisores que soportan el estándar europeo. La mayor desventaja radica en que si se usa el teléfono para realizar llamadas, las aplicaciones interactivas no estarán disponibles para el televidente, hasta que el teléfono deje de ser usado. Además, la poca capacidad que posee este canal hace que la transmisión y recepción de respuestas y datos tarden bastante tiempo en ser procesadas.

1.3.2.2. Ethernet y ADSL (Asymmetric Digital Subscriber Line)

Ambas tecnologías se combinan para ofrecer un mejor servicio en cuanto a interactividad se refiere. Para usarlo simplemente basta contratar un proveedor de servicios ADSL y un *router* en ambos extremos, para conectar el computador y el decodificador a la red. Anterior a estos *routers* se debe colocar un *splitter*, para separar la señal de telefonía con la señal de datos. La mayor ventaja es que este mecanismo siempre se encontrará apto para las aplicaciones interactivas.

1.3.2.3. Redes celulares

Se puede usar la tecnología Bluetooth conjuntamente con GPRS/UMTS. Principalmente este tipo de canal de retorno se usa para dispositivos móviles, y el único requerimiento es que el dispositivo sea compatible con estas tecnologías.

1.3.2.4. WiFi (Wireless Fidelity) / WiMAX (Worldwide Interoperability for Microwave Access)

La única ventaja que presenta estos mecanismos es que toda la interactividad se realiza inalámbricamente, pero al igual que Ethernet, requiere de un proveedor de servicios para que las aplicaciones interactivas se encuentren disponibles. Adicional, la arquitectura WiMAX brinda calidad de servicio como una red celular para datos y mayor velocidad de transmisión; servicios que no ofrece la arquitectura WiFi.

1.4. GINGA

Es el nombre que recibe el *middleware* del estándar brasileño ISDB-Tb. Posee una base de tecnologías bien estructuradas e innovaciones brasileñas únicas, haciendo de este *middleware* una especificación avanzada y una completa solución para la mayoría de países de Sudamérica.

Para realizar el desarrollo de aplicaciones interactivas, el *middleware* Ginga presenta dos subsistemas relacionados entre sí, los cuales pueden ser usados de acuerdo a las necesidades requeridas por cada aplicación. Los dos subsistemas son: Ginga-J (Ginga Java), usado para aplicaciones procedimentales[1] realizadas en Java; y Ginga-NCL, usado para aplicaciones declarativas[2] desarrolladas en NCL por medio del lenguaje de *scripting* LUA.

En la Figura 1.3 se presenta la arquitectura del middleware Ginga con sus dos subsistemas interrelacionados entre sí y sus componentes más importantes.

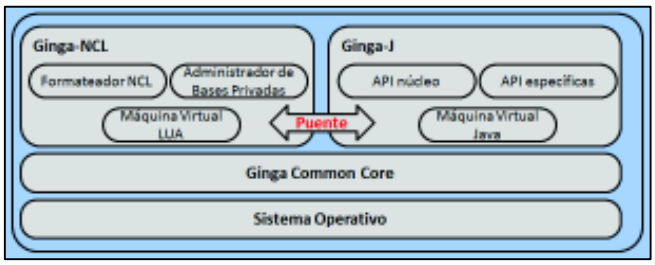

Figura 1.3 Arquitectura del middleware Ginga
Fuente: [5]

1.4.1. Ginga-J

Procesa el contenido de los objetos Xlet (aplicaciones basadas en el lenguaje Java). Este subsistema se basa en la norma ABNT NBR 15606-4 y utiliza como herramienta principal para la ejecución de las aplicaciones, la máquina virtual de

[1] Aplicaciones procedimentales: aplicaciones que controlan los segmentos y secuencia en las que se ejecuta un código.
[2] Aplicaciones declarativas: aplicaciones que utilizan el código necesario para ofrecer una solución, basándose en la lógica y no en una secuencia de instrucciones.

Java. También posee tres API (*Application Programming Interface*): verde, amarillo y azul, que permiten satisfacer las necesidades de la norma brasileña, así como también mantener la compatibilidad con la API del *middleware* GEM (*Globally Executable* MHP[3]).

En la Figura 1.4 se indica que las API de color verde son compatibles con GEM, mientras que las API de color amarillo son aplicaciones que se usan para cumplir con los requisitos que la norma de Brasil especifica, y la API de color azul únicamente se ejecutan dentro del entorno Ginga como tal.

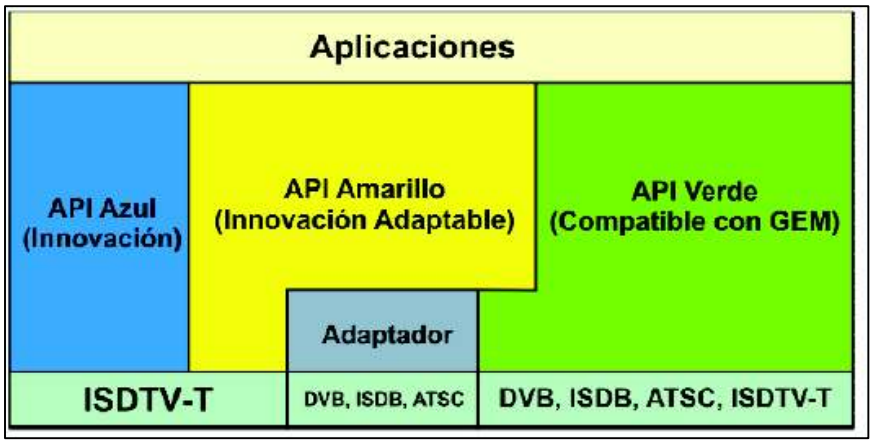

Figura 1.4 Componentes de Ginga-J

Fuente: [5]

1.4.2. Ginga-NCL

Dentro de su infraestructura posee una presentación para aplicaciones declarativas escritas en el lenguaje NCL (*Nested Context Language*), con facilidades para el desarrollo de aplicaciones interactivas, sincronismo, espacio-temporal entre objetos multimedia, adaptabilidad, soporte a múltiples dispositivos y soporte a la producción de programas interactivos en vivo. Las normas que indican la especificación de este sistema son: ABNT NBR 15606-2 y ABNT NBR 15606-5.

---

[3] MHP (*Multimedia Home Platform*): Define una plataforma común para las aplicaciones interactivas de la televisión digital

Al igual que Ginga-J, este subsistema también posee componentes propios, los cuales se enlistan a continuación y se indican en la Figura 1.5.

- Formateador: Se encarga de recibir y controlar aplicaciones multimedia escritas en NCL.
- Programador: Organiza el orden de la presentación del documento NCL. Es el responsable para dar la orden al componente Administrador de la Reproducción para iniciar la reproducción apropiada del tipo de contenido media para exhibirlo en el momento indicado.
- Analizador de XML y Conversor: Traducen la aplicación NCL en la estructura interna de datos de Ginga-NCL que se utiliza para controlar la aplicación.
- Administrador del Diseño: Es el responsable de asociar todas las regiones definidas en una aplicación NCL.
- Administrador de Bases Privadas: Está a cargo de recibir los comandos de edición de los documentos NCL y provee el mantenimiento a los documentos NCL presentados.
- Administrador de reproducción: Evalúa la conexión y expone el contenido multimedia que recibe del Programador.
- Administrador de contextos NCL: Soporta el contenido y las adaptaciones de la presentación de una aplicación interactiva.

Figura 1.5 Componentes de Ginga-NCL

Fuente: [5]

1.5. LENGUAJE NCL (NESTED CONTEXT LANGUAGE)

Este lenguaje se encuentra basado en el modelo conceptual de datos llamado NCM (*Nested Context Model*) que posee tres ideas fundamentales:

- Los nodos representan fracciones de información.
- Los enlaces definen las relaciones entre los nodos.
- Los enlaces no representan una única idea disponible para definir las relaciones.

En la Figura 1.6 se indica la manera en la que se definen los nodos y enlaces, siendo “Capítulo 1” y “Capítulo 2” nodos compuestos que agrupan a los nodos de contenido que son representados por las distintas “Secciones”. Las flechas representan a los enlaces que relacionan a ambos capítulos y a las secciones entre sí.

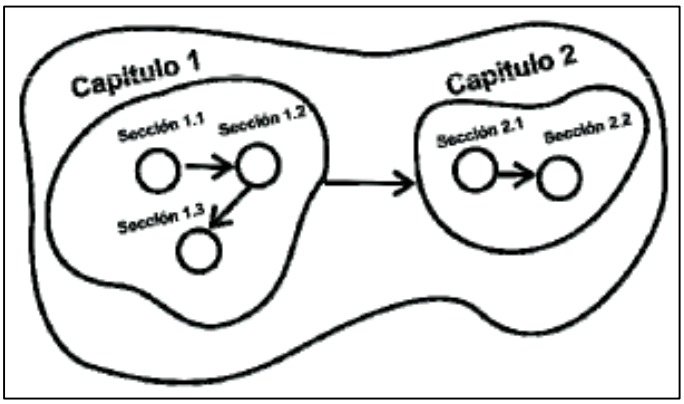

Figura 1.6 Representación del modelo NCM

Ahora hablando del lenguaje NCL en particular, cabe mencionar que es un lenguaje declarativo que especifica sincronismos espacio/tiempo entre objetos multimedia y define aspectos de interactividad. Es un lenguaje con soporte para múltiples dispositivos, adaptación al contenido y de edición en vivo. Además NCL es un software libre.

1.5.1. ESTRUCTURA DEL DOCUMENTO NCL

La estructura de un documento NCL consta de cuatro partes esenciales:

- La primera parte indica qué es lo que se va a exhibir. En esta sección se crean los nodos.

- En la segunda parte se especifica en dónde se van a colocar los nodos creados, por lo que se definen las regiones a utilizarse.
- Dentro de la tercera parte se define la manera en la que se van a reproducir los nodos, por lo que aquí se crean los descriptores.
- Finalmente se indican cuándo van a aparecer los diferentes nodos. Para realizar este paso se crean los enlaces (*links*) y conectores.

En la Figura 1.7 se indica el esquema básico de cómo se debe estructurar un documento basado en NCL para el desarrollo de aplicaciones interactivas, de acuerdo a las ideas presentadas.

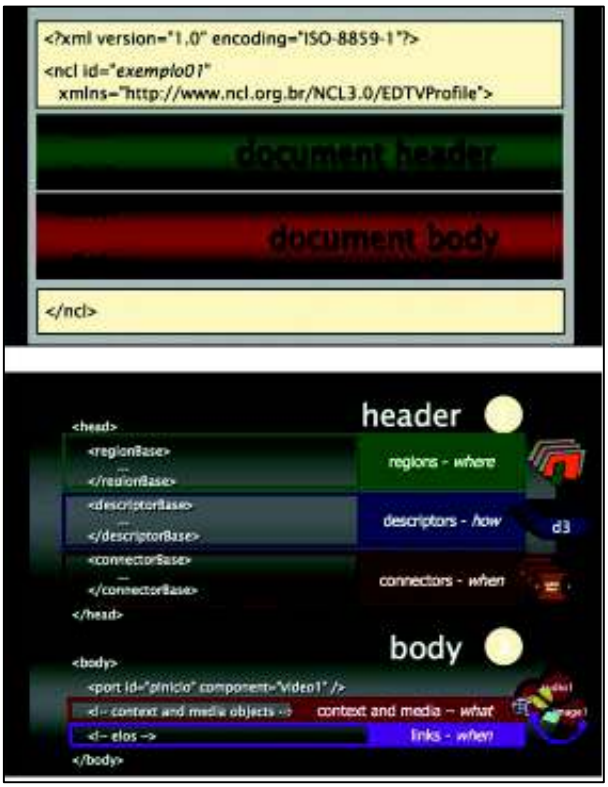

Figura 1.7 Formato de un documento NCL

Fuente: [6]

Un documento NCL se encuentra definido por el elemento raíz llamado `<ncl>`, y sus elementos hijos: el elemento `<head>` y el elemento `<body>`.

La estructura de cualquier documento siempre se inicia con la etiqueta <ncl>, la cual lleva su respectivo id y xmlns[4].

El elemento <head> posee las bases de los elementos referenciados por la aplicación NCL, como son las regiones, descriptores, transiciones, conectores y reglas. Todos ellos se definen en el elemento <body>.

Mientras tanto, el elemento <body> posee a todos los elementos que definen el contenido de la aplicación interactiva como tal, como los objetos multimedia, contextos, enlaces y switches.

El elemento <head> es capaz de contener a los siguientes elementos hijos:

- <importedDocumentBase>: especifica un conjunto de documentos importados de NCL.
- <ruleBase>: grupos de elementos pertenecientes a un conjunto de normas establecidas en <rule> y en elementos <compositeRule>.
- <transitionBase>: agrupa un conjunto de elementos <transition>, cada uno de los cuales define un efecto de transición en los elementos <media> en la presentación.
- <regionBase>: elemento que agrupa un conjunto de elementos <region>. Cada uno de ellos puede contener otro conjunto de elementos <region> anidados, y así sucesivamente, de forma recursiva.
- <descriptorBase>: agrupa un conjunto de elementos <descriptor>, los cuales definen los valores iniciales de las propiedades <media>.
- <connectorBase>: elemento que agrupa las relaciones definidas mediante conectores.
- <meta> y <metadata>: elementos que contienen meta-información acerca de un documento NCL [7].

[4] xmlns: sentencia que declara un espacio de nombres XML.

En cambio, el elemento <body> puede contener los elementos hijos:

- <port>: especifica un puerto para el nodo compuesto con su respectiva asignación a una interfaz, de uno de sus componentes (especificado por el atributo del componente).
- <media>: define el contenido de un objeto multimedia, los cuales son: contenidos de percepción, contenido de código imperativo, contenido de código declarativo o las variables globales de la aplicación.
- <context>: permite estructurar una aplicación NCL. El elemento puede contener elementos con contenido multimedia (elementos <media>), otros elementos anidados <context>, elementos de agrupación con contenido alternativo (elementos <switch>) y relaciones (elementos <link>) entre todos los objetos representados.
- <switch>: elemento que permite la definición de objetos alternativos (representado por <media>, <context> u otros elementos <switch>) para ser presentados. La elección se realiza durante el tiempo de presentación.
- <link>: define una relación entre los objetos multimedia [7].

1.6. LENGUAJE DE SCRIPTING LUA

Es un lenguaje de programación estructurado e imperativo[5] creado en 1993 y que fue introducido como un lenguaje de *script* con una semántica extensible. Esto quiere decir que, aprovechando que es un lenguaje liviano, es capaz de desarrollar *plug-ins* o *snap-ins* de un proyecto sin la necesidad de conocer su código fuente, permitiendo al programador realizar operaciones complejas, de una manera más fácil. También, otra de las características de este lenguaje es que sus valores son de primera clase. Esto quiere decir que todos ellos se pueden almacenar en variables, pueden ser pasados como argumentos de funciones y devueltos como resultados.

[5] Programación imperativa: conjunto de instrucciones que indican de forma secuencial y ordenada cómo realizar una tarea.

Por ser un lenguaje de *script* no sigue una estructura rígida, sino que se adapta a las características del problema a solucionar, pero siempre manteniendo una sintaxis en cuanto a operadores, bucles y sentencias se refiere.

Actualmente es utilizado en la mayoría de videojuegos, pero también se encuentran en aplicaciones profesionales, como es el caso de Adobe Photoshop. Además, aprovechando su tamaño (código fuente de 860 kB), está embebido dentro del *middleware* Ginga y otras aplicaciones con espacio de memoria limitada.

Este lenguaje se ejecuta dentro de una máquina virtual, basada en C y en Linux, por lo que se denomina multiplataforma, permitiendo al programador migrar fácilmente de una plataforma a otra, sin preocuparse por el sistema operativo que se use.

Por otra parte, los lenguajes LUA y NCL se relacionan por medio de *scripts* denominados NCLua, que ofrecen herramientas para la interacción de eventos causa – efecto (módulo `event`) y configuración de pantalla (módulo `canvas`).

1.6.1. MÓDULO EVENT

Por medio de este módulo se establece la conexión entre LUA y NCL, utilizando varios tipos de estructuras (`ncl`, `key`, `tcp`, `user`) para acceder a los eventos que maneja el código NCL.

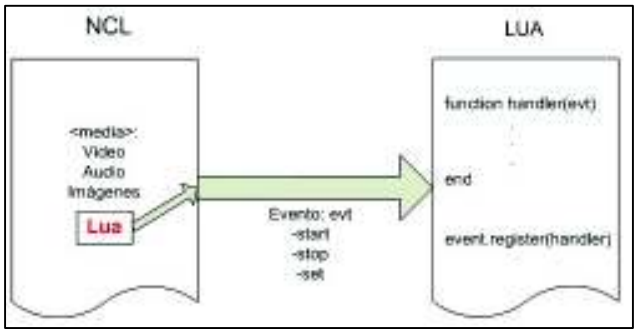

Figura 1.8 Relación entre NCL y LUA por medio de un evento de tipo `ncl`

Fuente: [8]

En la Figura 1.8 se observa la generación de un evento de tipo `ncl` para establecer un puente que permita la conexión entre los archivos LUA y NCL. Aquí se puede observar que dentro del documento NCL se generan los diferentes objetos media, y al momento de utilizar un evento externo, se procede a llamar a una función dentro del lenguaje LUA, para invocar a la acción que se desea.

1.6.1.1. Clase 'ncl'

Esta clase permite la conexión directa de dos o más objetos NCLua insertados dentro de una aplicación. Existen dos tipos de clases `ncl` que soportan estos objetos: `presentation` y `attribution`.

1.6.1.1.1. Presentation

Son eventos que controlan la visualización del nodo NCLua, identificados a través del campo `label`; mientras que el campo `action` señala la acción a realizarse por el *script* NCLua. Un evento de la clase `ncl` y del tipo `presentation` posee la siguiente estructura:

```
class: 'ncl'
type: 'presentation'
area: [string] -- nombre de ancla (label) asociado al evento.
action: [string] -- puede asumir los siguientes valores:
'start', 'stop', 'abort', 'pause' y 'resume'
```

1.6.1.1.2. Attribution

Son eventos que se encuentran asociados a las propiedades de los objetos NCLua, que son identificados a través del campo `name`; mientras que los valores que se asignan a las propiedades de estos objetos se los identifica en el campo `value`. Un evento de la clase `ncl` y del tipo `attribution` posee la siguiente estructura:

```
class: 'ncl'
type: 'attribution'
property: [string] -- nombre de la propiedad (name) asociado
al evento.
action: [string]    -- puede    asumir    los    siguientes
valores: 'start', 'stop', 'abort', 'pause' y 'resume'.
value: [string] -- nuevo valor que se asigna a la propiedad.
```

1.6.1.2. Clase 'key'

Esta clase se utiliza principalmente para representar el uso interactivo del control remoto por el televidente. NCLua no genera eventos para esta clase en especial puesto que el control remoto es un dispositivo únicamente de entrada.

Su estructura es la siguiente:

```
class: 'key'
type: [string] -- puede asumir 'press' o 'release'.
key: [string] -- valor de la tecla en cuestión.
```

1.6.1.3. Clase 'tcp'

Esta clase permite el uso del canal de interactividad por medio del protocolo TCP, por lo que su estructura necesita elementos que permitan abrir, establecer y cerrar una conexión de este canal. El uso y funcionamiento de esta clase será descrito en el Capítulo 3.

1.6.1.4. Clase 'user'

Esta clase permite ampliar la funcionalidad de los objetos NCLua generando sus propios eventos, por lo tanto, no posee una estructura definida. Además, como los eventos de esta clase son de uso interno, no es necesario que la entrada de sus eventos sea igual a `out`.

1.6.2. MÓDULO CANVAS

Este módulo se encarga de realizar operaciones gráficas y contenido dinámico en la pantalla, durante la presentación de una aplicación. Es similar a la creación de descriptores en un documento NCL [8].

Las funciones más importantes y utilizadas por este módulo son las siguientes:

- `canvas:attrSize (width, height)`: devuelve las dimensiones de una región.
- `canvas:attrFont (face, size, style)`: devuelve el nombre, tamaño y estilo de la fuente.
- `canvas:attrColor (R, G, B, A)`: devuelve el color a ser utilizado, en el formato RGBA[6].
- `canvas:drawRect (mode, x, y, width, height)`: devuelve el modo y el tamaño de un rectángulo que será dibujado sobre una región.
- `canvas:drawText (x, y, text)`: devuelve una cadena de texto en la posición indicada.
- `canvas:flush ()`: actualiza el contenido de la pantalla, una vez realizadas las operaciones con las funciones que se están detallando.
- `canvas:drawLine (x1, y1, x2, y2)`: dibuja una línea en la posición indicada, siendo `(x1, y1)` el extremo inicial y `(x2,y2)` el extremo final.

1.7. NCL COMPOSER

Es una herramienta de creación de hipermedia desarrollada por el Laboratorio de TeleMídia del Departamento de Informática de la PUC-Rio. Con esta herramienta se puede construir programas interactivos audiovisuales con poco conocimiento del lenguaje NCL. En la Figura 1.9 se observa que un documento hipermedia puede ser visto mediante diferentes perspectivas: *Structural View, Layout View, NCL Textual View, Validator Plugin, Properties View* y *Outline View* [9].

---

[6] RGBA: formato que usa el espacio de colores: rojo, verde, azul y alfa (canal de opacidad).

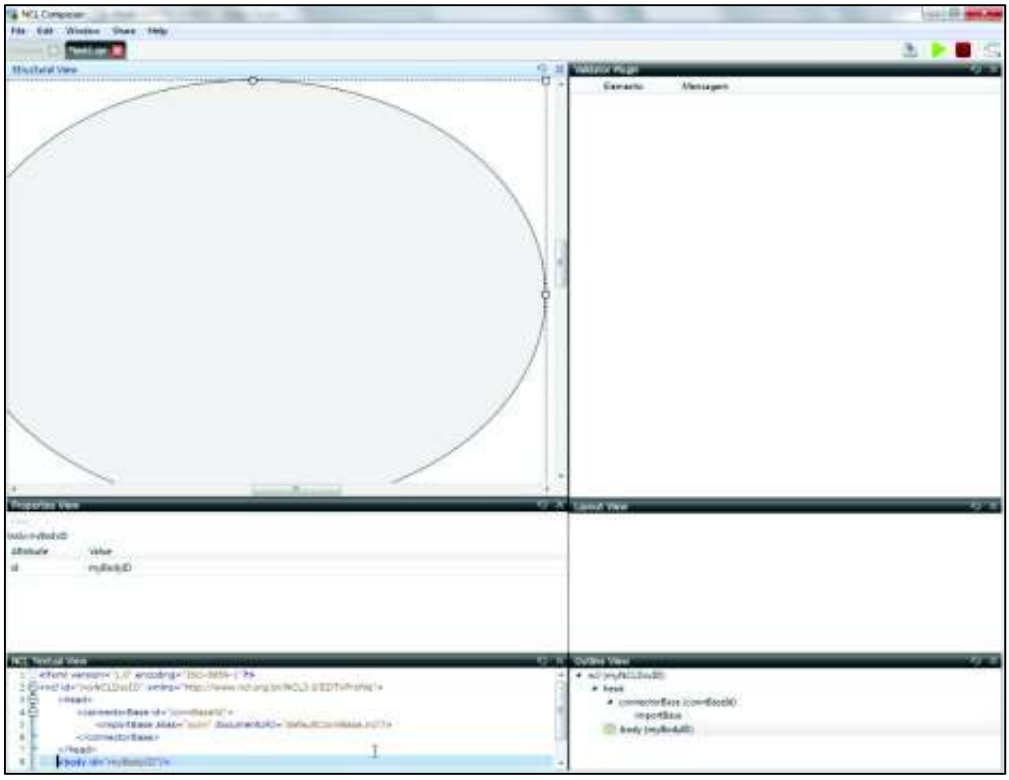

Figura 1.9 Área de trabajo de la herramienta NCL Composer

1.7.1. STRUCTURAL VIEW (VISTA ESTRUCTURAL)

La Vista Estructural muestra los nodos y enlaces entre los nodos. En esta vista se pueden crear nodos media, contextos y enlaces, además de definir sus propiedades.

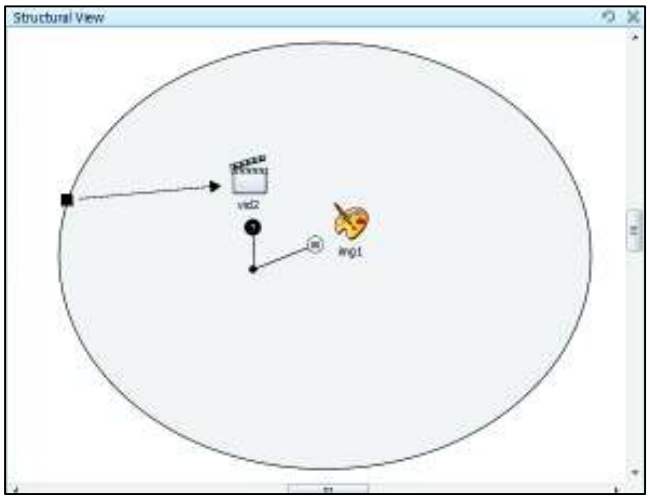

Figura 1.10 Vista Estructural de NCL Composer

En la Figura 1.10 se indica el enlace existente entre dos nodos multimedia. El primer nodo es de video (`vid2`) mientras que el segundo pertenece a una imagen (`img1`); además, se presenta un enlace entre un objeto `port` con el nodo de video, indicando que la aplicación se iniciará con dicho video.

1.7.2. LAYOUT VIEW (VISTA DE DISEÑO)

La Vista de Diseño muestra las regiones de la pantalla en la que los nodos multimedia de un documento pueden ser presentados. En esta vista se elige el formato de resolución de la programación: 4:3 o 16:9 (incluso presenta una opción de resolución de 4:5).

En la Figura 1.11 se muestra la Vista de Diseño de una aplicación interactiva que contiene tres regiones: la pantalla completa donde se proyecta los programas televisivos (`rgVideo`), una región en la que se muestra una o varias imágenes (`rgImg`) y otra región en la que se presenta información textual (`rgTexto`).

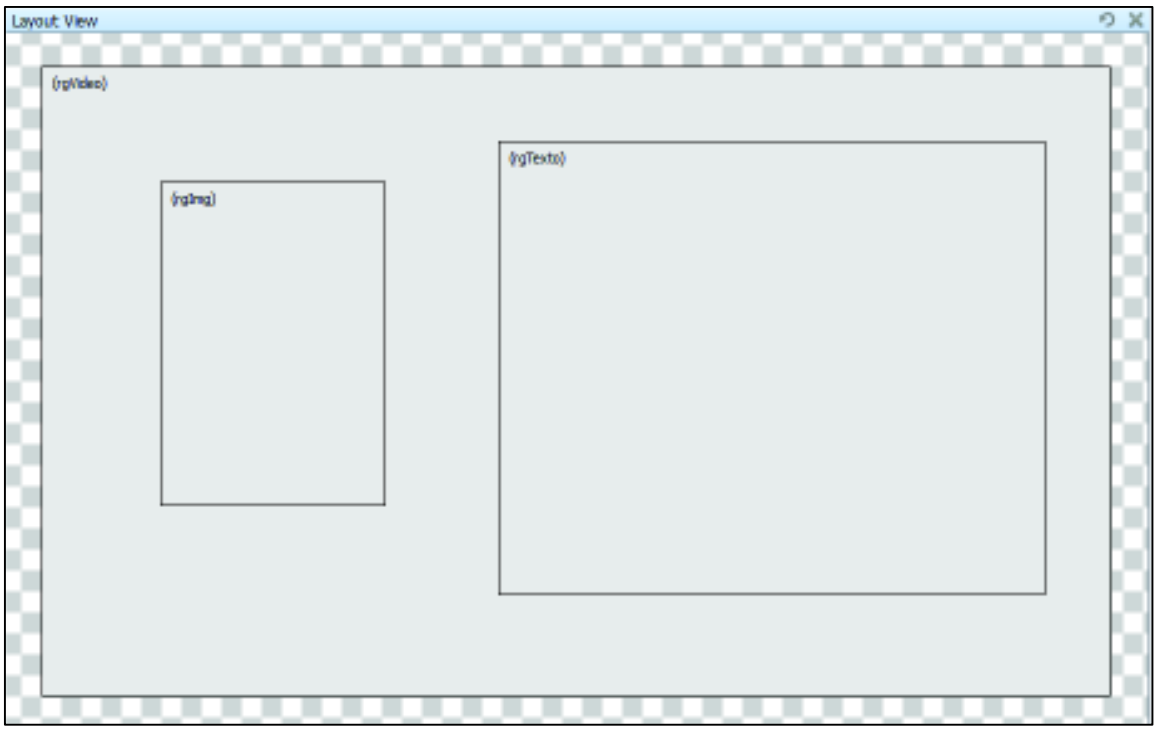

Figura 1.11 Vista de Diseño de NCL Composer

1.7.3. NCL TEXTUAL VIEW (VISTA TEXTUAL NCL)

En la Vista Textual NCL se presenta el código NCL en sí, tal y como se aprecia en la Figura 1.12. En esta vista, el usuario puede editar directamente el código como se lo haría en un editor de texto.

Figura 1.12 Vista Textual NCL de NCL Composer

1.7.4. VALIDATOR PLUG-IN (VALIDADOR DE PLUG-IN)

El Validador de *plug-in* es una vista que realiza una prueba sencilla y eficaz para verificar la autenticidad de todos los objetos insertados en la creación de la aplicación interactiva. En caso de que exista algún error, muestra un mensaje indicando el tipo de error que presenta.

En la Figura 1.13 se observa que, al insertar dos objetos en la Vista Estructural, el Validador de *plug-ins* presenta dos errores, uno por cada nodo insertado. El primer error indica que el nodo `port` no se encuentra enlazado con un nodo multimedia para el inicio de la aplicación, mientras que el segundo error indica que el `id` del objeto `media` insertado no es correcto.

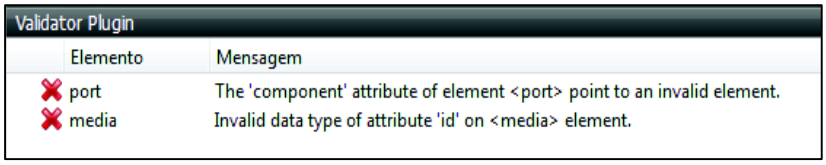

Figura 1.13 Validador de *plug-in* de NCL Composer

1.7.5. PROPERTIES VIEW (VISTA PROPIEDADES)

En la Vista Propiedades se presenta todas las propiedades de los nodos insertados en la aplicación. En esta vista se hacen todas las correcciones necesarias para eliminar los errores que aparecen en el Validador de *plug-ins*.
En la Figura 1.14 se indican todas las propiedades que presenta el nodo correspondiente a una imagen (`img1`) insertado en la Vista Estructural.

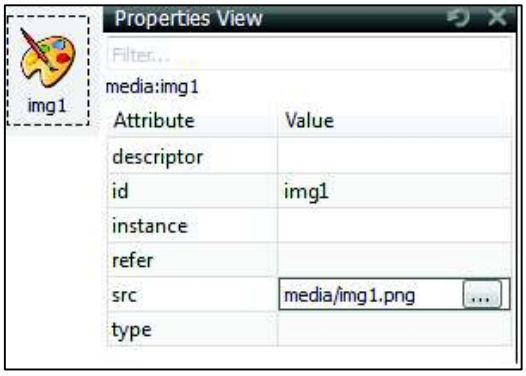

Figura 1.14 Vista Propiedades de la herramienta NCL Composer

1.7.6. OUTLINE VIEW (VISTA ESQUEMA)

La Vista Esquema permite observar la estructura del documento NCL que se está editando.

Un documento XML se puede representar como un árbol. Con el fin de tomar ventaja de esta característica, la Vista Esquema permite navegar por el documento NCL como un árbol, en orden jerárquico: `<ncl>`, `<header>` y `<body>`; cada uno con sus respectivos elementos [10].

En la Figura 1.15 se observan todos los elementos y nodos que han sido creados en un documento NCL, cada uno con su respectivo `id`.

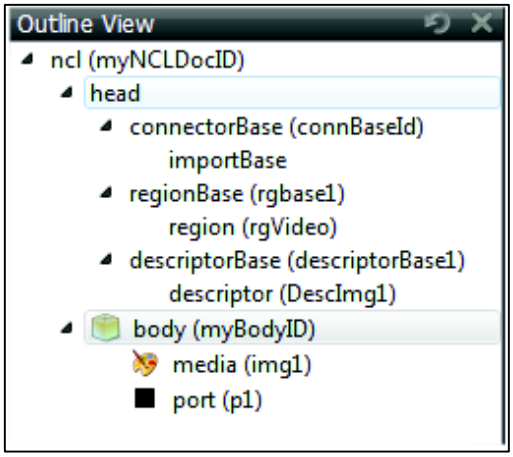

Figura 1.15 Vista Esquema de NCL Composer

Dentro de NCL Composer todas las vistas están sincronizadas entre sí, es decir, cuando se realizan cambios en un modo de vista, éstos son reflejados en los otros modos de vista de la herramienta.

1.8. NORMA ABNT NBR 15607-1

La Asociación Brasileña de Normas Técnicas (ABNT) es el Foro Nacional de Normalización de dicho país y cuyo contenido es responsabilidad de los Comités Brasileños (ABNT / CB), de los Organismos de Normalización (ABNT / ONS) y de las Comisiones de Estudio Especiales (ABNT / CEE). Estas normas son elaboradas por Comisiones de Estudio (CE), formadas por representantes de los sectores involucrados, de los cuales forman parte: productores, consumidores y varias instituciones (universidad, laboratorio y otros).

La norma ABNT NBR 15607-1 [11] fue elaborada por la *Comissão de Estudo Especial de Televisão Digital*, la cual pertenece a la ABNT y se basó en trabajos previos del *Fórum do Sistema Brasileiro de Televisão Digital Terrestre*.

Establece las formas en que un dispositivo receptor puede enviar información a la estación televisiva a través de diferentes medios de comunicación. La transmisión de la difusión de contenidos se lleva a cabo a través de la interfaz aire, a través de la alimentación principal de programación; mientras que un uso más personalizado del contenido puede ser enviado a través del canal interactivo.

Esta norma fue creada en el 2008 bajo el título general "Televisión Digital Terrestre – Canal de Interactividad" y está distribuida en tres partes.

- **Parte 1:** Protocolos, interfaces físicas e interfaces de software.
- **Parte 2:** Dispositivos externos
- **Parte 3:** Interfaces de configuración para las tecnologías de acceso.

Dentro de la Parte 1 se especifica que se utiliza el modelo de referencia OSI para definir los protocolos que se usan en el sistema brasileño de televisión digital. Las capas más bajas, dependiendo de la tecnología de acceso, se definen en las Tablas 1.2 y 1.3.

CAPA	CANAL DE RETORNO	PILA DE PROTOCOLOS
Física	**Ethernet**	- De acuerdo a IEEE 802.3 - RJ-45
	ISDN	- USB - RJ-45, RJ-11
	GSM-GPRS	- De acuerdo a ETSI EN 300 959[7] y ETSI EN 300 910[8] / USB
	WiMAX	- De acuerdo a IEEE 802.16d e IEEE 802.16e / USB
	WiFi	- De acuerdo a IEEE 802.11 / USB

Tabla 1.2 Definición de protocolos para la capa Física del modelo OSI

[7] ETSI EN 300 959: Sistema de telecomunicación digital celular - Modulación.
[8] ETSI EN 300 910: Sistema de telecomunicación digital celular – Radio transmisión y radio recepción.

Los protocolos que más se utilizan en la capa Enlace de Datos son:

- PPP (*Point to Point Protocol*): Protocolo usado para establecer una conexión directa entre dos nodos de una red de computadoras.
- PPPoE (*Point-to-Point Protocol over Ethernet*): Protocolo para la encapsulación PPP sobre una capa de Ethernet.
- IPCP (IP *Control Protocol*): Protocolo de control de red para configurar la dirección IP en un enlace PPP.
- PAP (*Password Authentication Protocol*): Protocolo simple de autenticación para autenticar un usuario contra un servidor de acceso remoto.
- CHAP (*Challenge Handshake Authentication Protocol*): Protocolo de autenticación por desafío mutuo usado por servidores accesibles vía PPP.
- ARP (*Address Resolution Protocol*): Protocolo de comunicaciones, responsable de encontrar la dirección de física que corresponde a una dirección IP específica.
- RLC (*Radio Link Control*): Protocolo de capa 2 utilizado en UMTS y LTE en la Interfaz Aire.
- LLC (*Logical Link Control*): Transporta los datos de protocolo de la red y agrega más información de control para ayudar a entregar ese paquete IP en el destino.
- SNDCP (*Sub Network Dependent Convergence Protocol*): Protocolo que convierte, encapsula y segmenta formatos de red externa en formatos de subred.

En la Tabla 1.3 se muestran los protocolos comúnmente usados en la capa Enlace de Datos según la arquitectura del canal de retorno a utilizar.

Los protocolos utilizados en las capas superiores del modelo OSI, en general, para dar soporte a Internet, se presentan en la Tabla 1.4.

Los detalles del protocolo para el canal de interactividad que lleva una solicitud y para el canal de *broadcast* que lleva la respuesta se presentan en la Tabla 1.5.

CAPA	CANAL DE RETORNO	PILA DE PROTOCOLOS
Enlace de Datos	**Ethernet**	- PPP / PPPoE / IPCP - PAP / CHAP - De acuerdo a IEEE 802.2 / ARP - CCP
	ISDN	- PPP / IPCP - PAP - CCP
	GSM-GPRS	- De acuerdo a ETSI EN 301 344 (RLC, LLC, SNDCP)
	WiMAX	- De acuerdo a IEEE 802.16d y a IEEE 802.16e
	WiFi	- De acuerdo a IEEE 802.2 / ARP

Tabla 1.3 Definición de protocolos para la capa Enlace de Datos del modelo OSI

CAPA	PILA DE PROTOCOLOS
Aplicación	De acuerdo al servicio: - HTTP 1.1 - TELNET - FTP - NNTP - SMTP - POP3 - DNS
Transporte	- TCP - UDP
Red	- IP / ICMP

Tabla 1.4 Definición de protocolos para las capas superiores del modelo OSI

CAPA	PILA DE PROTOCOLOS (CANAL DE BROADCAST)	PILA DE PROTOCOLOS (CANAL DE INTERACTIVIDAD)
Aplicación	Seleccionado de acuerdo al servicio	
Transporte	TCP, UDP	
Red	IP / ICMP	
Enlace de Datos	Sección DSM-CC[9] para datos privados	Protocolos de acuerdo a las funciones utilizadas: PSTN, conmutación de paquetes móvil, ISDN, ADSL, WiMAX, Wi-Fi, etc.
Física	MPEG-2 TS (*Transport Stream*)[10]	

Tabla 1.5 Definición de protocolos para el canal de broadcast y de interactividad

1.9. SERVICIOS DE LA TELEVISIÓN DIGITAL TERRESTRE

La Televisión Digital Terrestre junto a Internet transformarán de una forma radical todo el entorno del medio televisivo. La unión de ambas hará posible que el espectador forme parte activa de este medio a través de las distintas aplicaciones interactivas y servicios que se ofrecerán en el televisor, únicamente por medio del control remoto [12].

También, aprovechando la fusión de estas dos tecnologías, se implementarán nuevos servicios como el *T-Learning*, *T-Health*, *T-Commerce*, *T-Government*, web TV, entre los más destacados.

1.9.1. T-LEARNING

Es un servicio complementario para la educación que pretende ser difundido hacia toda la sociedad por medio de la TDT y se apoya en las herramientas que

[9] DSM-CC (*Digital Storage Media Command and Control*): Marco de desarrollo para controlar los canales asociados a los flujos de datos de tipo MPEG-1 y MPEG-2
[10] MPEG-2 TS: Protocolo de comunicación para audio, vídeo y datos.

posee Internet para fomentar el desarrollo de las habilidades de niños de diferentes edades. Se basan en juegos didácticos e interactivos con contenido educativo que permiten desarrollar el intelecto de la persona que los utilice.

1.9.2. T-HEALTH

Es un servicio que permite al usuario obtener guías en temas médicos relacionados con la alimentación, ejercicio, salud, etc.

1.9.3. T-COMMERCE

Es un mecanismo que permite comprar diferentes productos de Internet a través de las pantallas de un televisor, y que, aprovechando la llegada de la Televisión Digital Terrestre, se están creando diversas aplicaciones interactivas cuya finalidad es potenciar el despliegue y auge de este servicio.

1.9.4. T-GOVERNMENT

Este servicio es utilizado para aplicaciones gubernamentales con el objetivo de llegar a las diferentes clases sociales sobre las diversas actividades y gestiones que realiza el gobierno actual de cada país.

1.9.5. WEB TV

La web TV es un servicio que, utilizando el Internet como recurso complementario, permite transmitir cualquier tipo de programación emitida en un televisor desde su propia página web hacia el mundo entero.

1.10. DESCRIPCIÓN DE SERVICIOS WEB

Un servicio web es un conjunto de diferentes aplicaciones que intercambian datos entre sí, a través de mecanismos de comunicación, sean estos estándares o protocolos, con el objetivo de presentar diferentes servicios por medio de Internet.

Los estándares de comunicación y protocolos que se utilizan para conectarse a un servicio web son:

- XML (*Extensible Markup Language*): es un lenguaje basado en etiquetas y se utiliza como un formato estándar para la descripción y organización de datos.
- SOAP (*Simple Object Access Protocol*): es el protocolo que permite la comunicación de varias aplicaciones, independientemente de su plataforma y lenguaje de programación, por lo que se define como la esencia de los servicios web.
- WSDL (*Web Services Description Language*): es un protocolo que define los diferentes accesos a un servicio web a través de la descripción de interfaces.
- UDDI (*Universal Description, Discovery and Integration*): se define como un registro público en el que se almacena y se publica de forma sistemática toda la información que los servicios web poseen.
- WS-Security (*Web Service Security*): es un protocolo de seguridad que permite la autenticación de usuarios y la confidencialidad del intercambio de mensajes realizados en los servicios web.
- REST (*Representational State Transfer*): arquitectura que presenta su propia API, para realizar el intercambio de datos u operaciones desde y hacia los servicios web, a través del protocolo HTTP con sus respectivos métodos (POST, GET, PUT, DELETE).

1.10.1. USO DE LOS SERVICIOS WEB

Los servicios web son usados principalmente por diversas organizaciones para intercambiar datos entre ellas y con sus respectivos clientes, y así generar sistemas integrados, sin la necesidad de presentar los sistemas y plataformas de información de cada una de las organizaciones.

1.10.2. SERVICIO WCF (WINDOWS COMMUNICATION FOUNDATION) [13]

Es un marco de trabajo de Microsoft que permite la creación de servicios, presentando un amplio rango de funcionalidad por encima de un servicio web

común, debido a que un servicio web únicamente puede accederse por medio del protocolo HTTP, mientras que un servicio WCF puede almacenarse sobre IIS[11] (*Internet Information Server*), un servicio de Windows o servidores WAS[12] (*WebSphere Application Server*), permitiéndole de esta manera, usar una mayor cantidad de protocolos y estándares para su comunicación.

Para el uso de cualquier servicio web, esta arquitectura necesita de tres componentes, denominados ABC (*Address*, *Binding*, *Contract*):

- *Address:* lugar o dirección en donde se encuentra el servicio web.
- *Binding*: especificación del protocolo a utilizarse para la utilización del servicio web.
- *Contract:* indica la operación o actividad que puede realizar el servicio web.

1.10.2.1. Arquitectura WCF [14], [15]

Figura 1.16 Arquitectura WCF

[11] IIS: servidor web del sistema operativo Microsoft Windows.

[12] Servidor WAS: servidor de aplicaciones de software que está construido usando estándares abiertos como: J2EE, XML, y Servicios Web.

En la Figura 1.16 se indica la arquitectura WCF. Esta arquitectura está formada por cuatro capas: Contratos, Tiempo de ejecución del servicio, Mensajería y, Activación y Alojamiento. Cada una de estas capas poseen tareas específicas que se detallan a continuación:

Contratos: Especifican los aspectos del sistema de mensajes.

- Contrato de datos: define los parámetros de cada mensaje creado o utilizado, en formato XML, para que cualquier servicio o sistema pueda procesarlos.
- Contrato de mensaje: define el control y la comunicación específica para la interoperabilidad de los mensajes, por medio del protocolo SOAP.
- Contrato de servicios: se define como una interfaz dentro de un lenguaje de programación para publicar un servicio.
- Directiva y enlace: definen todas las condiciones de conexión y seguridad que deben cumplirse para comunicarse con un servicio.

Tiempo de ejecución del servicio: Capa que agrupa a diversos tipos de comportamiento que aparecen durante el tiempo de ejecución de un servicio.

- Comportamiento de limitación de peticiones: controla el número de mensajes que se procesan durante el uso de un servicio.
- Comportamiento de error: detalla al cliente el suceso de un error ocurrido en un servicio. Cabe recalcar, que se debe controlar la información que se comunica para evitar ataques malintencionados a un servicio.
- Comportamiento de metadatos: especifica la forma en la que un servicio publicará una descripción de sí mismo.
- Comportamiento de las instancias: define el número de veces que un servicio puede ser ejecutado simultáneamente.
- Inspección del mensaje: facilidad para revisar segmentos de un mensaje en tiempo de ejecución de un servicio.
- Comportamiento de transacción: realiza operaciones de transacción frente a algún error ocurrido.
- Comportamiento de distribución: agrupa aplicaciones para exponer adecuadamente diversos tipos de datos.

- Comportamiento de concurrencia: determina la interrelación de las instancias durante el tiempo de ejecución de un servicio.
- Filtro de parámetros: actúa sobre las cabeceras de los mensajes y realiza acciones específicas durante el tiempo de ejecución de un servicio, basándose en diversos filtros.

Mensajería: Se encuentran compuestos por canales que se encargan de procesar los diferentes mensajes existentes en un servicio.

- Canal de seguridad WS: habilita la seguridad en un mensaje.
- Canal de mensajería WS-Reliable: garantiza la entrega del mensaje
- Codificadores: presenta una diversidad de codificaciones, que se utilizan según las necesidades de un mensaje.
- Canal HTTP: define el protocolo HTTP para la entrega de los mensajes.
- Canal TCP: define el protocolo TCP para la entrega de los mensajes.
- Canal de flujo de transacciones: define los modelos de los mensajes para las transacciones.
- Canal NamedPipe (Tuberías nombradas): define la comunicación entre diferentes procesos.
- Canal MSMQ (*Message Queue Server*): permite la interacción con un servicio por medio del encolamiento de mensajes.

Activación y alojamiento: Es la presentación final de un servicio, a través de un programa.

- Servicio de activación de Windows: ejecución automática de un servicio que se ha implementado sobre un servidor WAS.
- .EXE: ejecución de un servicio con *host* propio.
- Servicios de Windows: ejecución de un servicio a través de un agente externo como IIS.
- COM+ (*Component Services*): aplicaciones o componentes que pueden ser alojados como servicios WCF.

CAPÍTULO 2

2. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DEL PLUG-IN PARA EL IDE DE DESARROLLO DE APLICACIONES INTERACTIVAS COMPOSER

En este capítulo se presentan los requerimientos, análisis, la solución propuesta e integración del *plug-in* con la herramienta NCL Composer.

Así mismo, durante el desarrollo de este capítulo se irá presentando parte del código del *plug-in*.

2.1. ANÁLISIS DEL PLUG-IN

El desarrollo del *plug-in* simplificará la tarea del usuario, ya que no tendrá que conocer todo el código NCL, puesto que el *plug-in* generará de forma automática un conjunto de archivos necesarios para que una aplicación de Televisión Digital funcione haciendo uso del canal de retorno para presentar información de un FAQ (*Frequently Asked Questions*).

También, el *plug-in* tendrá la capacidad de generar los archivos LUA con la información necesaria que permita conectarse al servicio web y se integrará con NCL Composer, generando así todo el código que la aplicación interactiva necesite para que pueda ser ejecutada en un Set Top Box o transmitida a través del interfaz aire.

Además, el *plug-in* dispondrá de una interfaz de usuario que permitirá definir ciertos parámetros, como por ejemplo la URL del servicio web, así como la adición, edición y supresión de información obtenida del servicio web de acuerdo a sus necesidades.

2.2. DISEÑO DEL PLUG-IN

Este *plug-in* será el encargado de manipular (editar y especificar) la información de los archivos LUA que se encuentran preconfigurados, permitiendo o generando código específico, basándose en las consideraciones que el usuario ha definido a través de la interfaz gráfica y de generar la versión final de estos archivos LUA que serán usados por la aplicación.

Para que el *plug-in* funcione y se conecte correctamente a NCL Composer se deben implementar las interfaces `IPluginFactory` e `IPlugin`. `IPluginFactory` es la interfaz responsable de crear nuevas instancias de `IPlugin` y de mantener toda la información del *plug-in* (versión, proveedor, autor, etc.).

El conjunto de archivos LUA generado por el *plug-in* permitirán que la aplicación que se cree para Televisión Digital pueda utilizar el canal de retorno a través del protocolo TCP/IP.

Uno de los archivos LUA tendrá el código necesario que permita la conexión al servicio web. Otro de los archivos LUA generados permitirá manipular el teclado del control remoto del *Set Top Box*, estableciendo la manera en la que las letras se pueden generar mediante el control a ciertos números para que el usuario pueda ingresarlas. Finalmente, otro archivo LUA permitirá redimensionar la información obtenida del servicio web en una región generada por el *plug-in*.

Adicional, para el correcto funcionamiento del *plug-in* se necesita de una base de datos con datos preconfigurados que contenga dos tablas, las cuales almacenarán la información necesaria, que posteriormente será consumida por uno de los archivos LUA generados por el *plug-in*. Estas tablas, a manera de ejemplo, tendrán una estructura simple, ya que solo se compondrán de dos campos; el primer campo será el ID que identifique al contenido de información que el usuario requiera y el segundo campo tendrá el contenido de la información

como tal. El *plug-in* leerá esta información que luego será consumida a través del servicio web.

El *plug-in* además necesitará de un servicio web basado en una arquitectura REST, de igual manera preconfigurado, el cual debe poseer cuatro métodos:

- El contenido de una pregunta almacenada en la base de datos.
- El contenido de una respuesta almacenada en la base de datos.
- El contenido de todas las preguntas almacenadas en la base de datos y separados por un carácter especial (/).
- El contenido de todas las respuestas almacenadas en la base de datos y separados por un carácter especial (/).

2.3. IMPLEMENTACIÓN DEL PLUG-IN

Para la manipulación de la información de la base de datos por medio de Qt, se deben utilizar dos clases: `QSqlDatabase` y `QSqlQuery`. La primera clase permite establecer una conexión a una determinada base de datos; mientras que el segundo permite ejecutar las consultas SQL a una base de datos.

La conexión a una base de datos es un proceso simple, en el que el primer paso consiste en establecer la información de la conexión y el segundo paso es abrir la conexión como tal.

En la Figura 2.1 se presenta el código para establecer la conexión con la base de datos. Es necesario indicar que los parámetros insertados deben ser idénticos a los configurados en la base de datos, ya que si difiere uno de ellos, la conexión con la base de datos no se realizará satisfactoriamente. Primeramente se debe utilizar el ODBC[13] (*Open DataBase Connectivity*) adecuado para conectarse con el tipo de base de datos que se esté usando, en este caso y a modo de ejemplo se utilizará una base de datos en SQL Server. Seguidamente se debe establecer

[13] ODBC: Estándar de acceso a las bases de datos desarrollado por *SQL Access Group* (SAG).

la cadena de conexión; hay que tomar en consideración que esta cadena es dependiente del nombre y tipo de la base de datos y del nombre del servidor. Finalmente, las credenciales del nombre de usuario y contraseña de acceso a la base de datos deben tener como argumentos valores específicos que dependerán, igualmente, del tipo de base de datos que se esté utilizando.

```
QSqlDatabase dbFaq = QSqlDatabase::addDatabase("QODBC3");
dbFaq.setDatabaseName("Driver={SQL Server};Server=UIOLCIBERNOSJG\\SQLEXPRESS;Database=faq;");
dbFaq.setUserName("sa");
dbFaq.setPassword("200610390");
```

Figura 2.1 Conexión a la base de datos `faq`

Para manejar el contenido de la base de datos se generaron tres métodos, los cuales son: `agregarFaq()`, `editarFaq()`, `eliminarFaq()`.

El método `agregarFaq` permitirá insertar en la base de datos una nueva pregunta con la información que haya sido forjada por el usuario en la interfaz y luego limpiará los cuadros de texto de la interfaz de usuario. Para ello se crea un objeto llamado `agregar` y como argumento se pasa la sentencia SQL en un objeto `QSqlQuery` y el objeto `database` que contiene información de las bases de datos en la que se debe ejecutar esta sentencia, tal como se observa en la Figura 2.2.

```
QSqlDatabase DBPreg = QSqlDatabase::database ("faq");
QSqlQuery agregar("SELECT id_pregunta, info_pregunta, info_respuesta FROM pregunta", DBPreg);

agregar.exec ("INSERT INTO pregunta (id_pregunta, info_pregunta, info_respuesta) VALUES ('"+id+"','"
              +pregunta+"','"+respuesta+"')");
qDebug() << "registro agregado";
```

Figura 2.2 Código del método `agregarFaq()`

El método `editarFaq` permitirá actualizar una pregunta existente en la base de datos y luego limpiará los cuadros de texto de la interfaz de usuario. Para ello, en la Figura 2.3 se observa que a través de un objeto `editar` se está llamando al método `exec`, el cual tiene como argumento una consulta de actualización que se realizará en la base de datos.

```
editar.exec ("UPDATE pregunta SET id_pregunta = '"+ui->sbxId->text()
             +"', info_pregunta ='"+ui->lineEdPregunta->text()
             +"', info_respuesta = '"+ui->lineEdRespuesta->text()
             +"' WHERE id_pregunta ='"+ui->tblFaq->item(f,0)->text()+"'");
```

Figura 2.3 Código del método `editarFaq()`

El método `eliminarFaq` permitirá suprimir una pregunta existente en la base de datos y luego limpiará los cuadros de texto dela interfaz de usuario. En la Figura 2.4 se observa que a través de un objeto `borrar` se está llamando al método `exec`, el cual tiene como argumento una consulta de supresión a la base de datos.

```
borrar.exec("DELETE FROM pregunta WHERE id_pregunta ='"+ui->tblFaq->item(f,0)->text()+"'");
qDebug() << "registro eliminado";
```

Figura 2.4 Implementación del método `eliminarFaq()`

El siguiente paso es crear el código necesario para integrar el *plug-in* al NCL Composer. Para ello se deben utilizar las interfaces `IPluginFactory` e `IPlugin`. Es por esto que se implementó la interfaz `faqviewfactory`. En la Figura 2.5 se observa el código empleado para presentar la información de versión, nombre, autor, descripción, entre otros que corresponde al *plug-in* diseñado.

```
QString id() const {return "FaqView";}
QString name() const {return "Faq View Plugin";}
QString version() { return "0.1"; }
QString compatVersion() {return "0.1";}
QString vendor() {return "EPN";}
QString copyright() {return "EPN";}
QString description() {return "FAQ Plugin creates the LUA
                        files needed to connect to a WCF
                        service and get answers to
                         questions asked by viewers";}
QString url() {return "http://ginga.epn.edu.ec/";}
QString category() {return "FAQ";}
```

Figura 2.5 `faqviewfactory` implementado para la integración del *plug-in*

`IPlugin` es el propio *plug-in* de NCL Composer. Por lo tanto, cada vez que se crea un nuevo documento en NCL Composer, se genera una nueva instancia del *plug-in*. En otras palabras, cada vez que un documento es modificado o creado, siempre se llamará a una instancia de `IPlugin`.

La implementación de `IPlugin` se lo realizó a través de la clase definida en `faqviewplugin.h`, y es la encargada de integrar este *plug-in* con los otros *plug-ins* de NCL Composer.

La integración con NCL Composer se la realiza por medio de *Slots* y *Signals*. Los *Slots* se emplean para realizar acciones específicas sobre el núcleo de NCL Composer, denominado *Composer-Core*, y los *Signals* se utilizan para establecer una comunicación sobre los otros *plug-ins*. Más información se la puede obtener en [16].

Para esta clase se emplearon los siguientes *Slots*:

- `composer::extension::IPlugin::onEntityAdded`: se utiliza cuando se requiere agregar una entidad[14].
- `composer::extension::IPlugin::onEntityChanged`: se utiliza cuando se requiere cambiar una entidad.
- `composer::extension::IPlugin::onEntityRemoved`: se utiliza cuando se requiere eliminar una entidad.
- `composer::extension::IPlugin::error Message`: se utiliza cuando existe un error por una instancia del *plug-in*.

En la Figura 2.6 se presenta el código necesario para la implementación de la interfaz `IPlugin`. Se emplea el método `getWidget()` el cual retorna un puntero a un objeto de la clase `QWidget` para comunicarse con el núcleo de NCL Composer y crea la instancia necesaria cuando se abre o se modifica un documento en NCL Composer; mientras que mediante `public slots` se

[14] Entidad: Es un objeto concreto o abstracto que presenta interés para el sistema.

publican todos los *slots* que se necesitan para realizar accesos con NCL Composer.

```
public:
    explicit faqviewPlugin();
    ~faqviewPlugin();
    QWidget* getWidget();
    void init();

public slots:
    void onEntityAdded(QString ID, Entity *);
    void onEntityChanged(QString ID, Entity *);
    void onEntityRemoved(QString ID, QString entityID);
    void errorMessage(QString error);
```

Figura 2.6 Código `IPlugin` implementado para el desarrollo del *plug-in*

Adicional, se presenta el esquema de la interfaz de usuario del *plug-in* en la Figura 2.7. En esta interfaz se han agregado algunos elementos:

- Tres cuadros de texto en donde se mostrará el ID de la pregunta, el contenido de la pregunta y el contenido de la respuesta, respectivamente.
- Tres botones, los cuales cumplen con las siguientes funciones: si se presiona el botón `Agregar`, permitirá guardar una nueva pregunta con su respuesta a la base de datos; si se presiona el botón `Editar` se actualizarán los cambios efectuados sobre una pregunta o alguna respuesta en la base de datos; finalmente, si se presiona el botón `Eliminar` se borrará la pregunta y la respuesta de la base de datos
- Cuadros de texto, en los que se deben colocar las URL respectivas del servicio web.

Como última observación, el botón `Crear código NCL` permitirá generar todo el código necesario dentro de NCL Composer, una vez que el *plug-in* se encuentre integrado.

Figura 2.7 Interfaz gráfica del *plug-in* implementado

2.3.1. ARCHIVOS NCLUA

A continuación se describen los archivos LUA que el *plug-in* generará.

2.3.1.1. Archivo tcp.lua

Este archivo va a permitir conectarse a la aplicación generada por el *plug-in* de NCL Composer con el sitio remoto.

En la Figura 2.8 se aprecia que se declara localmente módulos y funciones globales pues, al definir el *script* como un módulo, el acceso al entorno global se pierde. La variable `CONNECTIONS` lista las conexiones TCP activas y la función local `current` obtiene la rutina que esté en ejecución y devuelve el identificador de esta rutina.

```
local _G, coroutine, event, assert, pairs, type
    = _G, coroutine, event, assert, pairs, type
local s_sub = string.sub

module 'tcp'

local CONNECTIONS = {}

local current = function ()
    return assert(CONNECTIONS[assert(coroutine.running())])
end
```

Figura 2.8 Declaración de módulos y revisión de conexiones TCP activas

En la Figura 2.9 se indica la función `execute` que permite iniciar una conexión TCP. El parámetro `f` es una función que debe ejecutar las rutinas necesarias para la realización de una conexión TCP, envío de solicitudes y obtención de respuestas. El parámetro … incluye a todos los parámetros adicionales que existiesen y se pasan a la función que la co-rutina ejecuta a través de la función `resume`.

```
function execute (f, ...)
    resume(coroutine.create(f), ...)
end
```

Figura 2.9 Código de la función `execute` para conexiones TCP

La Figura 2.10 presenta la función `connect`, la cual permite conectarse a un servidor a través del protocolo TCP. Esta función solo devuelve algún parámetro cuando se establece la conexión. El parámetro `host` indica el nombre del servidor con el cual se realiza la conexión y el parámetro `port` indica el puerto que se utilizará para la conexión.

```
function connect (host, port)
    local t = {
        host    = host,
        port    = port,
        waiting = 'connect'
    }
    CONNECTIONS[coroutine.running()] = t

    event.post {
        class = 'tcp',
        type  = 'connect',
        host  = host,
        port  = port,
    }

    return coroutine.yield()
end
```

Figura 2.10 Código de conexión TCP con un servidor

En la Figura 2.11 se presenta el código de la función `handler`, el cual se encarga de tratar los eventos generados por las llamadas a las funciones de la clase `tcp`. El parámetro `evt` es una tabla que contiene los datos del evento capturado.

```
function handler (evt)
    if evt.class ~= 'tcp' then return end

    if evt.type == 'connect' then
        for co, t in pairs(CONNECTIONS) do
            if (t.waiting == 'connect') and
               (t.host == evt.host) and (t.port == evt.port) then
                t.connection = evt.connection
                t.waiting = nil
                resume(co)
                break
            end
        end
        return
    end

    if evt.type == 'disconnect' then
        for co, t in pairs(CONNECTIONS) do
            if t.waiting and
               (t.connection == evt.connection) then
                t.waiting = nil
                resume(co, nil, 'disconnected')
            end
        end
        return
    end

    if evt.type == 'data' then
        for co, t in pairs(CONNECTIONS) do
            if (t.waiting == 'data') and
               (t.connection == evt.connection) then
                resume(co, evt.value)
            end
        end
        return
    end
end
event.register(handler)
```

Figura 2.11 Código de la función `handler` para tratar eventos de conexiones TCP

En la Figura 2.12 se detalla el código de la función `disconnect`, el cual cierra la conexión TCP.

```
function disconnect ()
    local t = current()
    event.post {
        class      = 'tcp',
        type       = 'disconnect',
        connection = assert(t.connection),
    }
end
```

Figura 2.12 Código de la función `disconnect`

En la Figura 2.13 se observa el código de la función `send`, el cual envía una petición TCP al servidor. El parámetro `value` contiene el mensaje que se enviará al servidor.

```
function send (value)
    local t = current()
    event.post {
        class      = 'tcp',
        type       = 'data',
        connection = assert(t.connection),
        value      = value,
    }
end
```

Figura 2.13 Código de la función `send`

2.3.1.2. Archivos mainp.lua y mainr.lua

Estos archivos llaman a `tcp.lua`, para establecer los parámetros necesarios de conexión con el servicio WCF, el cual ofrece todas las preguntas y respuestas almacenadas en la base de datos.

Cabe recalcar que los archivos `mainp.lua` y `mainr.lua` cumplen las mismas funciones, con la diferencia de que el primer archivo consume el servicio WCF para obtener las preguntas, mientras que el segundo archivo se utiliza para la obtención de cada respuesta.

Para devolver el *string* de las preguntas y respuestas enviadas por el servidor se utilizan las líneas de código presentadas en la Figura 2.14.

```
local result = tcp.receive()
if result then
    result = string.match(result, "<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">(.-)</string>")
    or "No se encuentra disponible"
else
    result = "error: " .. evt.error
end
```

Figura 2.14 Código LUA para devolver el *string* con el pedido solicitado

Como se puede ver en la Figura 2.14, en la variable `result` se obtienen los datos entregados por la función `receive()`, estos datos deben ser procesados, siempre y cuando existan. El servicio WCF retorna un documento XML con el pedido solicitado. Este documento tiene que ser procesado, para lo cual se utiliza el método `string.match`, el cual permite indicar las etiquetas que deben ser encontradas en el archivo y se extrae únicamente la información que se encuentra

dentro de las etiquetas `<string>` y `</string>`. Si no existe información aparecerá un mensaje de no disponibilidad de datos

2.3.1.3. Archivo input.lua

La principal función de este archivo es establecer el formato para el texto que se ingresará por medio del control remoto, así como también la generación de una tabla que permita asociar los botones numéricos del control remoto con el alfabeto.

En la Figura 2.15 se indica la función empleada para asociar los números del control remoto con el alfabeto. Su estructura se basa en el teclado que normalmente presenta un teléfono celular.

A modo de ejemplo, se explica el funcionamiento de esta función, si un usuario presiona una vez el número 3, aparecerá la letra “d”; si presiona dos veces, aparecerá la letra “e”; si presiona tres veces, aparecerá la letra “f”; y si presiona cuatro veces, aparecerá el propio número “3”.

```
local MAP = {
      ['1'] = { '1', '.', ',' }
    , ['2'] = { 'a', 'b', 'c', '2' }
    , ['3'] = { 'd', 'e', 'f', '3' }
    , ['4'] = { 'g', 'h', 'i', '4' }
    , ['5'] = { 'j', 'k', 'l', '5' }
    , ['6'] = { 'm', 'n', 'o', '6' }
    , ['7'] = { 'p', 'q', 'r', 's', '7' }
    , ['8'] = { 't', 'u', 'v', '8' }
    , ['9'] = { 'w', 'x', 'y', 'z', '9' }
    , ['0'] = { '0' }
}
```

Figura 2.15 Código LUA para asociar botones numéricos del control remoto con el alfabeto

2.3.1.4. Archivo output.lua

Este archivo configura las opciones de presentación del texto, para que toda la información sea visualizada ordenadamente en una región establecida dentro de la pantalla televisiva.

En la Figura 2.16 se presenta la función `redraw`, la cual permite acoplar el texto para que se presente en una región específica que será definida a través del NCL Composer o del *plug-in*. Esta función también permitirá establecer los formatos y color de fuente que serán visualizados en dicho espacio asignado.

```
function redraw()
  canvas:attrColor('black')
  canvas:drawRect("fill", 0, 0, cw, ch)
  canvas:attrColor('white')

  local tw, th = canvas:measureText("a")
  local charsByLine = tonumber(string.format("%s", cw / tw))
  local desctb = breakString(TEXT, charsByLine)
  local y = 0

  for k,ln in pairs(desctb) do
      canvas:drawText(20, y,ln)
      y = y + th
  end
  canvas:flush()
end
```

Figura 2.16 Función que permite ordenar el formato del texto en una región asignada

La propiedad `canvas` se utiliza para definir el tipo, tamaño y color de fuente de la información a ser visualizada.

Las variables `tw` y `th` almacenan las medidas de largo y ancho de la región, respectivamente, en donde será presentada la información.

La variable `charsbyLine` contiene el número de líneas en el que será presentada la información dentro de la región.

La variable `desctb`, permite dividir las líneas de la información, en palabras completas, sobre la región generada por el *plug-in* en NCL Composer.

2.3.2. RECURSOS QUE REQUIERE EL PLUG-IN

Para manipular los archivos NCLUA externos es necesario que sean recursos de Qt. Para ello deben estar ubicados en la carpeta `Resources` y tienen que ser

incluidos como archivos .qrc (*Qt resource*[15]). Estos archivos han sido colocados en una carpeta colocada en `Resources` denominada `faqview`, tal y como se observa en la Figura 2.17.

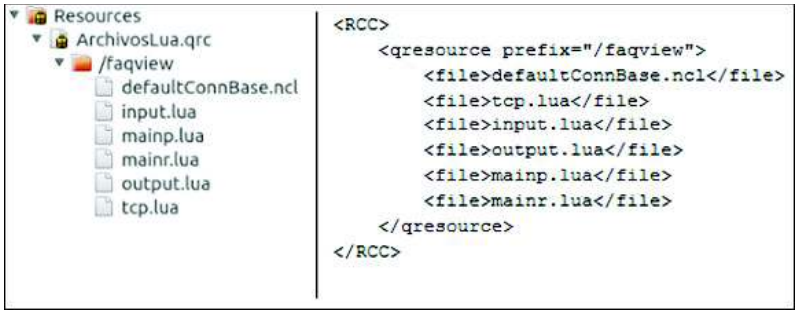

Figura 2.17 Archivos NCLUA almacenados en la carpeta `faqview`

En la Figura 2.17 también se observa que dentro de la carpeta `Resources`, también se incluyó al archivo `defaultConnBase.ncl`. Este archivo está escrito en formato XML y posee las definiciones estándares de los conectores principales que se usan en el desarrollo de una aplicación en NCL Composer.

```
QFile filep1(":/faqview/mainp.lua");
filep1.open(QIODevice::ReadOnly|QIODevice::Text);
QTextStream inp1(&filep1);
QString linep1 = inp1.readAll();
linep1=linep1.replace(QString("@xxx"),page);
filep1.close();
```

Figura 2.18 Código para almacenar y modificar un archivo NCLUA

A manera de ejemplo, para abrir y utilizar el archivo `mainp.lua` que se encuentra en la carpeta dentro de los recursos, se debe implementar el código detallado en la Figura 2.18. Para esto se realiza lo siguiente: se crea un objeto `filep1`, el cual contiene como argumento la ubicación del archivo en mención (`/faqview/mainp.lua`). Después se llama a la función `open()` para abrir el

[15] *Qt resource*: Mecanismo de plataforma independiente para almacenar archivos binarios durante la ejecución de la aplicación y se utiliza cuando se requieren de ciertos archivos a utilizarse en la aplicación sin correr el riesgo de perderlos.

archivo en modo lectura y en modo texto. Luego se debe guardar el contenido del archivo por medio de un objeto denominado `inp1`. Finalmente se debe reemplazar la cadena "`@xxx`" por la información que se encuentra contenida en la variable `page`; esta variable almacena la URL que el usuario definió en el cuadro de texto respectivo en la interfaz gráfica.

2.4. BASE DE DATOS

Como se mencionó anteriormente, se necesitará de una base de datos en donde se almacene la información que será consumida por el servicio web. Para este trabajo, se realizará la creación de una base de datos, cuyas tablas y elementos permitirán identificar preguntas y respuestas, con su respectivo identificador único.

En la Tabla 2.1 se indican los campos que se crearán en la base de datos denominada `faq`.

Tabla	Nombre de columna	Tipo de dato	Descripción
`Pregunta`	`id_pregunta`	`Int`	Valor autonumérico
	`info_pregunta`	`varchar(200)`	Contenido de la pregunta
`Respuesta`	`id_respuesta`	`Int`	Valor autonumérico
	`info_respuesta`	`varchar(200)`	Contenido de la respuesta

Tabla 2.1 Características de los campos de las tablas de la base de datos `faq`

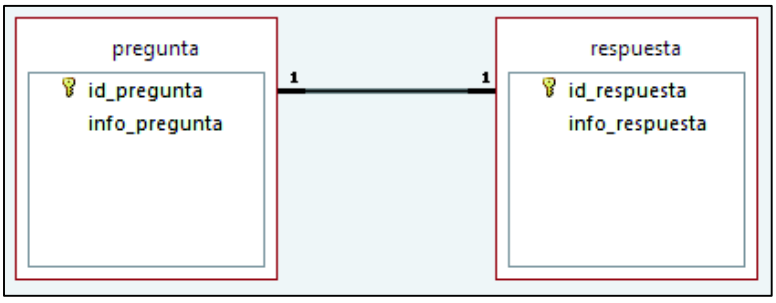

Figura 2.19 Diagrama relacional de las tablas de la base de datos

En la Figura 2.19 se presenta el diagrama relacional de las tablas presentadas en la Tabla 2.1.

2.5. SERVICIO WCF

La elaboración de este servicio web tiene por objetivo crear un documento que será capaz de entregar, en formato XML, las preguntas y respuestas que se obtienen de la base de datos.

Se debe crear una clase para establecer los parámetros necesarios de conexión con la base de datos. Para aquello es necesario configurar cuatro funciones con las características que se muestran en la Tabla 2.2.

Nombre del método	Descripción
`Faq`	Muestra el contenido de una pregunta, de acuerdo al `id` especificado
`Faqs`	Muestra el contenido de todas las preguntas
`Resp`	Muestra el contenido de una respuesta, de acuerdo al `id` especificado
`Resps`	Muestra el contenido de todas las respuestas

Tabla 2.2 Información de los métodos de la clase `Service`

Se debe configurar un contrato de servicio para declarar la interfaz que se publicará en el servicio WCF y cuatro operaciones de contrato, una por cada función creada en la clase `Service`, para cumplir con la descripción que cada una presenta en la Tabla 2.2. En la Tabla 2.3 se muestra la información del contrato de servicio `IService`, y en la Figura 2.20 se presenta su diagrama de clase.

Ruta	Resumen
`/faq/{id}`	Se usa como un método GET de HTTP y cuya ruta es `/faq/{id}`
`/faqs`	Se usa como un método GET de HTTP y cuya ruta es `/faqs`
`/resp/{id}`	Se usa como un método GET de HTTP y cuya ruta es `/resp/{id}`
`/resps`	Se usa como un método GET de HTTP y cuya ruta es `/resps`

Tabla 2.3 Información del contrato de servicio `IService`

Figura 2.20 Diagrama de clase de `Service`

Adicionalmente, se debe modificar el archivo `Web.Config` para disponer de una arquitectura tipo REST, ya que esta arquitectura es la que puede consumir el *plug-in* de forma sencilla. Este archivo se encuentra en formato XML, y es utilizado para cambiar las opciones de configuración (mecanismos de seguridad, de concurrencia, protocolos de intercambio de mensajes, etc.) que requiere el servicio WCF.

El archivo `Web.Config` ya se encuentra preconfigurado, por lo que se deben agregar las líneas de comando que se encuentran enmarcadas en la Figura 2.21. Estas líneas son las siguientes:

`<endpointBehaviors>` indica que el servicio será de tipo REST, permitiéndole al usuario acceder a la dirección URL y obtener una respuesta en formato XML mediante un método de HTTP.

`<webHttp>` indica que la comunicación entre cliente-servidor se lo realizará mediante el protocolo HTTP.

`<protocolMapping>` es un parámetro que debe ser agregado para definir un conjunto de asociación entre los protocolos de transporte y los enlaces usados por el servicio WCF implementado. El uso de `webhttpBinding` indica que la información se intercambiará por medio de peticiones HTTP en lugar de mensajes SOAP.

`<directoryBrowse enabled=”false”/>` es un parámetro que inhabilita la obtención de información, en forma de lista, que se encuentra en el directorio del servicio WCF.

Figura 2.21 Parámetros de configuración del archivo `Web.Config`

2.6. USO DEL PLUG-IN PARA EL DESARROLLO DE UNA APLICACIÓN EJEMPLO PARA TELEVISIÓN DIGITAL TERRESTRE

La GUI del *plug-in*, una vez que forme parte de NCL Composer, se verá como lo indicado en la Figura 2.22. Al abrir la herramienta NCL Composer, el *plug-in* se verá como lo que se está presentando en el recuadro de color rojo.

Para crear una nueva aplicación de Televisión Digital se deben llenar los campos con los datos correspondientes y seguidamente presionar el botón `Crear código NCL`.

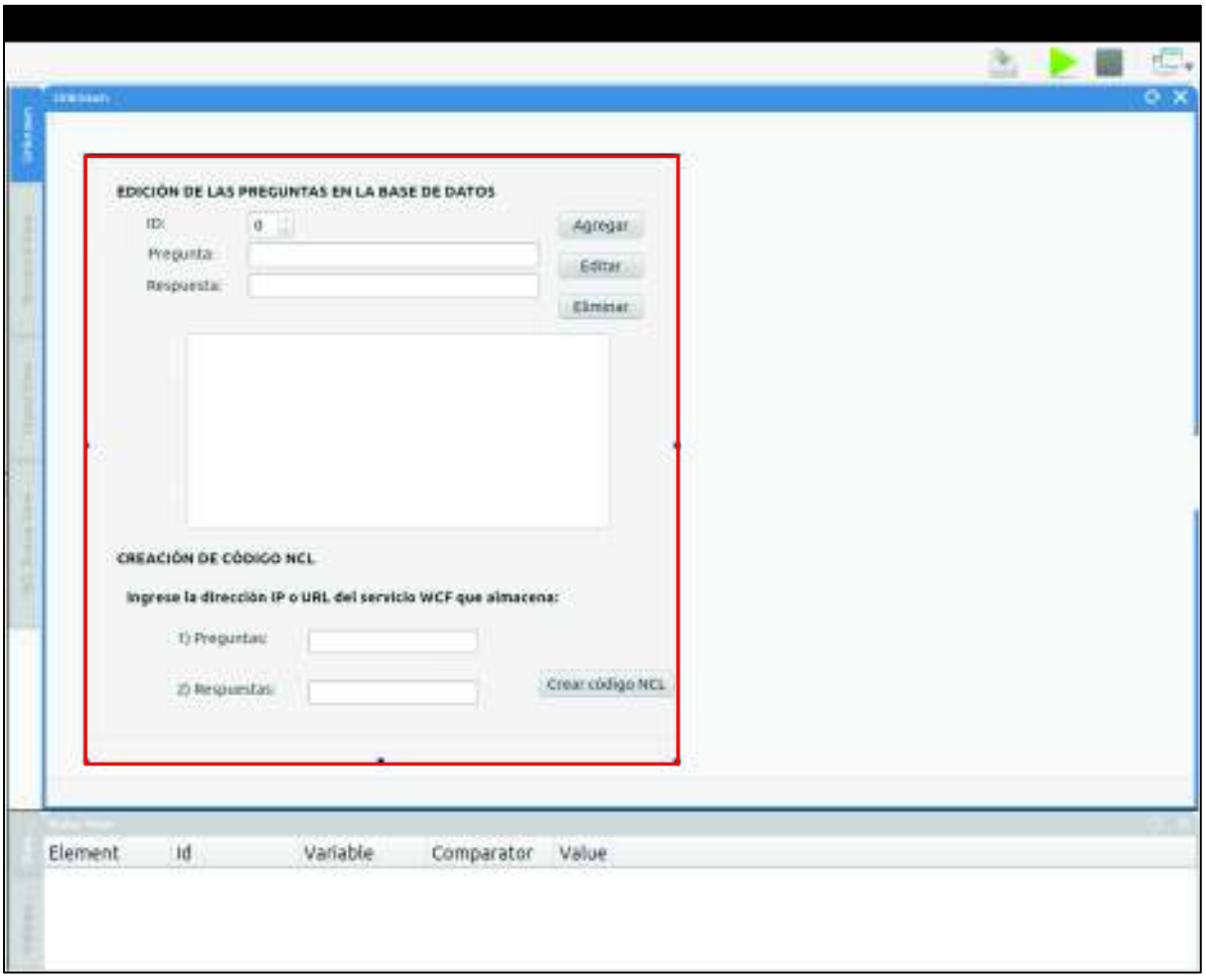

Figura 2.22 Integración gráfica del *plug-in* con NCL Composer

En la Figura 2.23, se presenta la vista *Layout View*, en la que se verá la estructura en donde serán ubicados los distintos elementos en la pantalla del televisor, de acuerdo a lo que se genera a través del *plug-in*.

En la Figura 2.24, se presenta la vista *Textual View*, en la cual se observa el código NCL generado a través del *plug-in* integrado en NCL Composer.

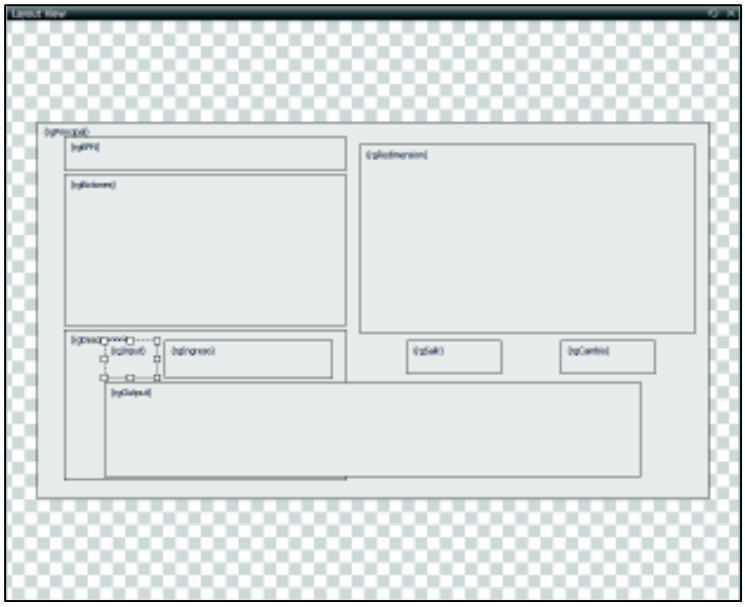

Figura 2.23 Estructura del *plug-in* en *Layout View*

NCL Textual View

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <ncl id="aplicacionFaq" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
3      <head>
4          <connectorBase id="connBaseId">
5              <importBase alias="conn" documentURI="defaultConnBase.ncl"/>
6          </connectorBase>
7          <regionBase id="rgbase">
8              <region id="rgBackground" left="0.00%" top="0.00%" width="100.00%" height="100.00%">
9                  <region id="rgPrincipal" left="0.00%" top="0.00%" width="100.00%" height="100.00%">
10                     <region id="rgInput" left="10.00%" top="58.00%" width="8.00%" height="10.00%"/>
11                     <region id="rgOutput" left="10.00%" top="69.00%" width="80.00%" height="25.00%"/>
12                     <region id="rgRedimension" left="48.00%" top="6.00%" width="50.00%" height="50.00%"/>
13                     <region id="rgInteractividad" left="88.00%" top="6.00%" width="7.00%" height="12.00%"/>
14                     <region id="rgIngreso" left="19.00%" top="58.00%" width="25.00%" height="10.00%"/>
15                     <region id="rgSalir" left="55.00%" top="58.00%" width="14.00%" height="9.00%"/>
16                     <region id="rgEPN" left="4.00%" top="4.00%" width="42.00%" height="9.00%"/>
17                     <region id="rgBotones" left="4.00%" top="14.00%" width="42.00%" height="40.00%"/>
18                     <region id="rgDescripcion" left="4.00%" top="55.00%" width="42.00%" height="40.00%"/>
19                     <region id="rgCambio" left="78.00%" top="58.00%" width="14.00%" height="9.00%"/>
20                 </region>
21             </region>
22         </regionBase>
23         <descriptorBase id="descriptorBase">
           <descriptor id="descInput" region="rgInput" focusIndex="inputIdx"/>
           Invalid data type of attribute 'focusIndex' on <descriptor> element.
25             <descriptor id="descBackground" region="rgBackground"/>
26             <descriptor id="descPrincipal" region="rgPrincipal"/>
27             <descriptor id="descInteractividad" region="rgInteractividad"/>
28             <descriptor id="descSalir" region="rgSalir"/>
29             <descriptor id="descIngreso" region="rgIngreso"/>
30             <descriptor id="descOutput" region="rgOutput"/>
31             <descriptor id="descMenuPreguntas2" region="rgBotones" explicitDur="20s"/>
32             <descriptor id="descAzul" region="rgDescripcion"/>
33             <descriptor id="descAmarillo" region="rgDescripcion"/>
34             <descriptor id="descBotones" region="rgBotones"/>
35             <descriptor id="descClasificacion" region="rgDescripcion"/>
36             <descriptor id="descConcepto" region="rgDescripcion"/>
37             <descriptor id="descFinal" region="rgDescripcion"/>
38             <descriptor id="descGris" region="rgDescripcion"/>
39             <descriptor id="descOrigen" region="rgDescripcion"/>
40             <descriptor id="descReducir" region="rgDescripcion"/>
41             <descriptor id="descRehusar" region="rgDescripcion"/>
42             <descriptor id="descReciclar" region="rgDescripcion"/>
43             <descriptor id="descRecuperacion" region="rgDescripcion"/>
44             <descriptor id="descVerde" region="rgDescripcion"/>
45             <descriptor id="descTransferencia" region="rgDescripcion"/>
```

Figura 2.24 Estructura del *plug-in* en *Textual View*

En la Figura 2.25, se presenta la vista *Structural View*, en la cual se verán todas las relaciones existentes de los diferentes elementos media que fueron creados automáticamente por el *plug-in*.

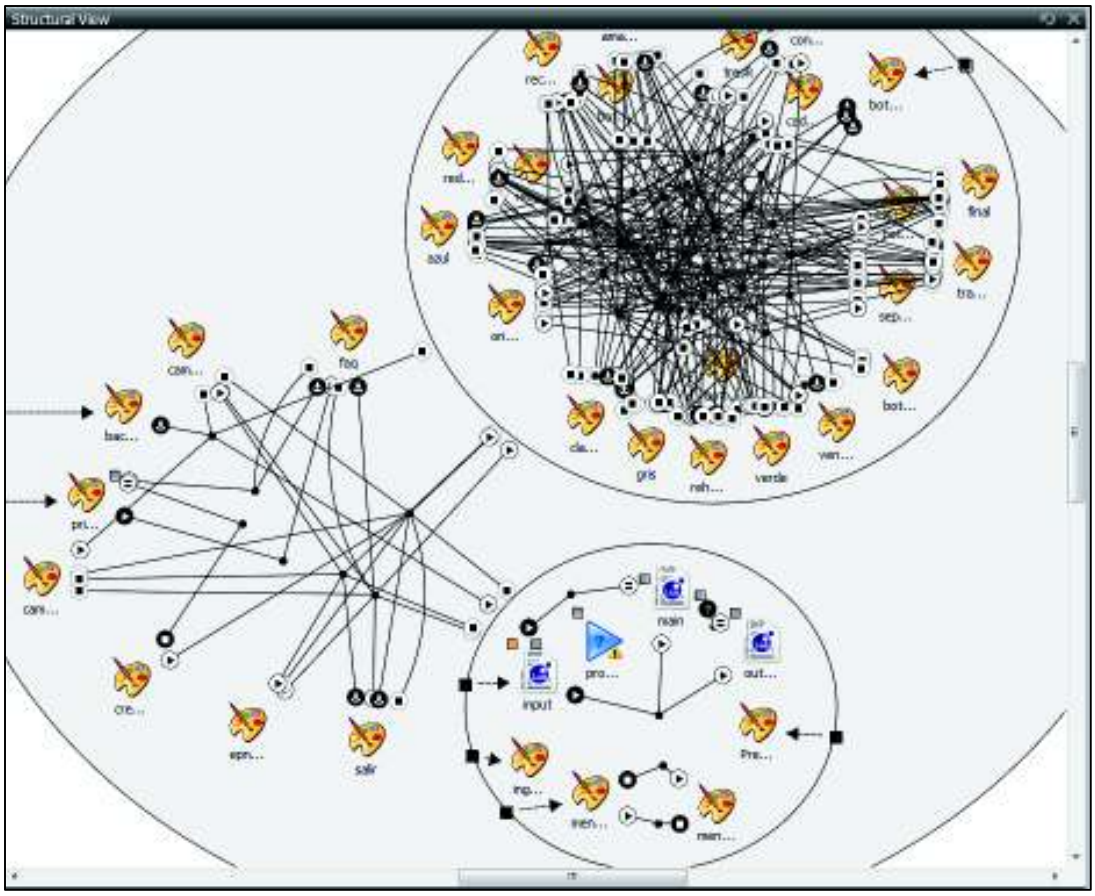

Figura 2.25 Estructura del *plug-in* en *Structural View*

CAPÍTULO 3

3. DESARROLLO DE APLICACIONES INTERACTIVAS, PRUEBAS Y RESULTADOS

El cuidado y protección del medio ambiente en nuestro país es responsabilidad de todas las personas, debido a que en Ecuador existe una gran biodiversidad de flora y fauna en todas las regiones, y algunas de las especies se encuentran en peligro de extinción por diferentes causas.

La protección ambiental, la iniciativa Yasuní ITT y el proceso de reciclaje, son considerados tres temas importantes de carácter ambiental que deben ser difundidos a la ciudadanía, para tomar medidas precautelares y empezar a proteger todos los recursos naturales que enriquecen la naturaleza de nuestro país.

Así mismo se indicará la forma en la que se implementó cada una de las cuatro aplicaciones, con su respectivo código desarrollado, y las pruebas correspondientes en los entornos virtual y real.

3.1. APLICACIÓN INTERACTIVA ENFOCADA EN LA PROTECCIÓN AMBIENTAL

Esta aplicación tiene como objetivo ser un medio informativo en el que se presente al televidente las características más representativas de algunas de las especies de la fauna nacional, y las posibles amenazas que ponen en peligro de extinción a cada una de ellas.

Debido a que esta información puede ser difundida a toda la ciudadanía, esta aplicación debe presentar contenido corto y conciso, para evitar que el televidente pierda su concentración, tanto en la información presentada por la aplicación como en la programación televisiva que se encuentre en la pantalla. De acuerdo a esta premisa, se implementará una barra con imágenes de la fauna presente en

Ecuador, para que el televidente tenga la libertad de elegir cualquiera de ellas y ver su información.

3.1.1. PRESENTACIÓN DE LA INFORMACIÓN

Como la idea es presentar una barra con imágenes, y teniendo en cuenta que dentro de la pantalla televisiva se debe organizar adecuadamente toda la información, se dividirá esta barra en tres páginas, cada una de ellas albergando tres especies distintas.

Para facilidad del televidente, se presentará un menú de ayuda que indique la manera en la que debe desplazarse entre páginas, mientras se ejecute la aplicación interactiva. Este menú se visualiza en la Figura 3.1.

Figura 3.1 Menú de ayuda de la aplicación interactiva enfocada en la protección ambiental

La barra de imágenes que presenta elementos de la fauna ecuatoriana, se implementará en la parte izquierda de la pantalla y mostrará imágenes de tres especies de fauna, como se observa en la Figura 3.2; y una vez que el televidente escoja una imagen de esta barra, en la parte inferior derecha de la pantalla aparecerá la información de la especie escogida, como se aprecia en la Figura 3.3.

Figura 3.2 Barra de imágenes de la aplicación interactiva enfocada en la protección ambiental

Figura 3.3 Información presentada de la especie escogida por el televidente

Una vez que el televidente empleó la aplicación y decide finalizarla, aparecerá durante diez segundos los datos de las personas que intervinieron en el desarrollo de esta aplicación interactiva, tal y como se muestra en la Figura 3.4.

Figura 3.4 Información de los desarrolladores de la aplicación interactiva enfocada en la protección ambiental

3.1.2. DESARROLLO DE LA APLICACIÓN GINGA

El primer paso es generar el código que permita incluir los conectores usualmente usados para este tipo de aplicaciones. En la Figura 3.5 se indica un `alias` denominado `conn` que representa a la biblioteca de conectores y `documentURI` que contiene el documento `defaultConnBase.ncl`, el cual representa al archivo NCL donde están definidos los conectores.

```
<connectorBase id="connBaseId">
  <importBase alias="conn" documentURI="defaultConnBase.ncl"/>
</connectorBase>
```

Figura 3.5 Código NCL para importar el archivo NCL de conectores

El siguiente paso es crear una `regionBase` que permita agrupar a los elementos `region` necesarios, que representarán un espacio en la pantalla en donde se mostrarán los elementos multimedia. Cada elemento `region` se encuentra identificado por un `id`, y la posición y dimensión que ocupa en pantalla se indica (en porcentaje) mediante las propiedades `left`, `top`, `width` y `height`.

En la Figura 3.6 se indica el código para crear un elemento `region` que es identificado por un `id` denominado `rgInteractividad` y su posición dentro de la pantalla ocupa las siguientes dimensiones: `left` que permite ubicar a la región a un 86% de la posición horizontal izquierda de la pantalla; `top` para ubicar a la región a un 6% de la posición vertical superior de la pantalla; `width` y `height` que indican que la región tiene una base del 8% y una altura del 14%, con respecto a las dimensiones de la pantalla. La propiedad `zIndex` configurada con un valor de “1”, indica que la región `rgInteractividad` se colocará por delante de la región base que lo almacena, la cual por defecto tiene un valor de “0”.

```
<region id="rgInteractividad" left="86.00%" top="6.00%" width="8.00%" height="14.00%" zIndex="1"/>
```

Figura 3.6 Código NCL para la creación de un elemento `region`

Una vez que se tienen las regiones definidas, se debe incluir a los elementos multimedia que van a presentarse al televidente. Estos elementos comúnmente son imágenes, videos, sonidos, texto, documentos HTML y archivos LUA.

Igualmente, cada elemento es identificado por un `id`, y se inserta desde una ubicación definida mediante la propiedad `src`. Para asociar un elemento multimedia a una región, se utilizan los descriptores.

En la Figura 3.7 se muestra el código para la inserción de un elemento `media` que es identificado por medio del `id barra1` y que almacena en `src` el elemento y su ubicación es `media/barra1.png`. Se asocia con una región por medio del `descriptor` denominado `descBarra1`.

```
<media id="barra1" src="media/barra1.png" descriptor="descBarra1"/>
```

Figura 3.7 Código NCL para la creación de un elemento `media`

La señal de televisión que es receptada por el *Set Top Box* debe presentarse durante la ejecución de la aplicación. Para ello es necesario colocar un elemento `media`, cuyo `id` es `programacion` y definida por la cadena `sbtvd-ts://0`. En la Figura 3.8 se indica el código NCL para captar la señal televisiva, conjuntamente con la propiedad `bounds`, la cual permite redimensionar el tamaño del video para presentar todos los elementos de la aplicación interactiva.

```
<media id="programacion" src="sbtvd-ts://0" descriptor="descProgramacion">
  <property name="bounds"/>
</media>
```

Figura 3.8 Código NCL para presentar la señal televisiva en el STB

Una vez que todos los elementos media con sus respectivos descriptores se asociaron adecuadamente, es necesario crear los conectores y enlaces necesarios para que la aplicación interactiva funcione correctamente.

El conector `onBeginStart` muestra el botón de inicio de interactividad, una vez que la programación televisiva se presenta en la televisión. En la Figura 3.9 se indica el enlace `link1` que contiene al conector `onBeginStart` y se utiliza para iniciar el elemento media `interactividad` una vez que inicia la programación televisiva `programacion`.

```
<link id="link1" xconnector="conn#onBeginStart">
  <bind role="onBegin" component="programacion"/>
  <bind role="start" component="interactividad"/>
</link>
```

Figura 3.9 Código NCL que indica el inicio de la aplicación interactiva

Para que los elementos media sean presentados correctamente en pantalla, se necesita que la imagen televisiva sea redimensionada. Para ello se utiliza el conector `onKeySelectionSetStop`. En la Figura 3.10 se indica el código para crear el enlace `link2` que asocia al conector `onKeySelectionSetStop`, el cual indica que una vez que aparece el botón de inicio de interactividad y al momento de presionar el botón `OK` del control remoto, la programación televisiva se redimensionará de acuerdo a los valores definidos en la propiedad `var`, mientras que el botón de interactividad desaparecerá de la pantalla.

```
<link id="link2" xconnector="conn#onKeySelectionSetStop">
  <bind role="onSelection" component="interactividad">
    <bindParam name="keyCode" value="ENTER"/>
  </bind>
  <bind role="set" component="programacion" interface="bounds">
    <bindParam name="var" value="48%,10%,40%,40%"/>
  </bind>
  <bind role="stop" component="interactividad"/>
</link>
```

Figura 3.10 Código NCL para el redimensionamiento de la presentación televisiva

El conector `onKeySelectionStartNStopN` se utiliza para iniciar (presentar) o detener (ocultar) diferentes elementos media en la aplicación interactiva. En la Figura 3.11 se indica que el enlace `link4` asocia al conector `onKeySelectionStartNStopN` y se utiliza para presentar los elementos `ctxBarra2` y `botones2`; y a su vez detener u ocultar los elementos `ctxBarra3`,

ctxBarra1, botones1 y botones3, una vez que se presiona la tecla GREEN del control remoto.

Los elementos ctxBarra1, ctxBarra2 y ctxBarra3 son contextos utilizados para agrupar las imágenes, en forma de cintas, de las diferentes especies de animales en peligro de extinción; mientras que los elementos botones1, botones2 y botones3 presenta la información de uso del control remoto al televidente.

```
<link id="link4" xconnector="conn#onKeySelectionStartNStopN">
  <bind role="onSelection" component="botones1">
    <bindParam name="keyCode" value="GREEN"/>
  </bind>
  <bind role="start" component="ctxBarra2"/>
  <bind role="start" component="botones2"/>
  <bind role="stop" component="ctxBarra3"/>
  <bind role="stop" component="ctxBarra1"/>
  <bind role="stop" component="botones1"/>
  <bind role="stop" component="botones3"/>
</link>
```

Figura 3.11 Código NCL para iniciar y detener elementos media cuando se presione la tecla GREEN del control remoto

El conector onSelectionStartNStopN permitirá desplegar la información de la especie de animal escogida por el televidente. En la Figura 3.12 se indica que el enlace link31 asociado al conector onSelectionStartNStopN se utiliza cuando el televidente, por medio de las teclas de navegación (UP, DOWN, LEFT, RIGHT) del control remoto, seleccione el elemento barra7 y se desplegará información relevante del elemento infoPuma (información de los pumas); mientras que los elementos infoTapir e infoDelfin permanecerán ocultos.

```
<link id="link31" xconnector="conn#onSelectionStartNStopN">
  <bind role="onSelection" component="barra7"/>
  <bind role="start" component="infoPuma"/>
  <bind role="stop" component="infoTapir"/>
  <bind role="stop" component="infoDelfin"/>
</link>
```

Figura 3.12 Código NCL para seleccionar información acerca de la especie de un animal

El conector `onEndSet` se utiliza para finalizar con la presentación de la aplicación interactiva. En la Figura 3.13 se indica que el enlace `link9` se asocia al conector `onEndSet` y sirve para redimensionar a su tamaño original la presentación televisiva una vez que se detenga el elemento `creditos`.

```
<link id="link9" xconnector="conn#onEndSet">
  <bind role="onEnd" component="creditos"/>
  <bind role="set" component="programacion" interface="bounds">
    <bindParam name="var" value="0%,0%,251%,251%"/>
  </bind>
</link>
```

Figura 3.13 Código NCL para redimensionar la presentación televisiva a su tamaño original

3.1.3. EJECUCIÓN DE LA APLICACIÓN GINGA

La aplicación interactiva se inicia con la presentación de un ícono en la parte superior derecha de la pantalla, en la cual el televidente al presionar `OK` con el control remoto, accede a la información de nueve especies de animales que se encuentran en peligro de extinción en el país.

En la Figura 3.14 se indica el lugar en donde se presenta el ícono de inicio de la aplicación interactiva dentro de la pantalla televisiva.

Figura 3.14 Ícono que indica el inicio de la aplicación interactiva enfocada en la protección ambiental

Una vez que se inicia la aplicación interactiva, en la parte izquierda de la pantalla aparecerán nueve especies de animales, presentados en una cinta, con el fin de organizar mejor toda la información a presentarse. En la parte central se muestra un menú informativo que le indica al televidente las acciones a tomar con el control remoto. En la parte inferior se indica la información de la especie elegida por el televidente.

En la Figura 3.15 se indican las imágenes de especies de animales, como cóndor, mono y oso de anteojos, que muestra la aplicación.

Figura 3.15 Primer grupo de imágenes de la aplicación interactiva enfocada en el medio ambiente

Para avanzar al siguiente grupo de imágenes, el televidente debe presionar el botón `GREEN` del control remoto y para retornar a páginas anteriores, se debe presionar el botón `RED`.

En la Figura 3.16 se indica el segundo grupo de imágenes de animales. En esta sección se presentan las especies: pacaraná, pecarí y guacamayo.

En la Figura 3.17 se indica el último grupo de imágenes de animales. En esta sección se presentan las especies: puma, tapir y delfín rosado.

Figura 3.16 Segundo grupo de imágenes de la aplicación interactiva enfocada en el medio ambiente

Figura 3.17 Tercer grupo de imágenes de la aplicación interactiva enfocada en el medio ambiente

Para navegar entre las diferentes especies de animales dentro de la cinta, el televidente debe utilizar las teclas de navegación `UP` y `DOWN` del control remoto. Para obtener información detallada de una de estas especies, se debe presionar la tecla `OK`.

En la Figura 3.18 se indica la información detallada del oso de anteojos, una vez que el televidente escogió a dicha especie por medio del control remoto.

Figura 3.18 Información de la especie de animal seleccionado por el televidente

En la Figura 3.19 se indica la información detallada del guacamayo, una vez que el televidente escogió a dicha especie por medio del control remoto en el segundo grupo de imágenes de la aplicación.

En la Figura 3.20 se indica la información detallada del tapir, una vez que el televidente escogió a dicha especie por medio del control remoto en el último grupo de imágenes de la aplicación.

Para culminar con la ejecución de la aplicación interactiva, el televidente puede presionar el botón `EXIT` del control remoto al momento que desee.

Figura 3.19 Información del guacamayo seleccionado por el televidente

Figura 3.20 Información del tapir seleccionado por el televidente

Una vez que se presiona la tecla `EXIT`, aparecerá por un lapso de 10 segundos información de los integrantes que intervinieron en el desarrollo de esta aplicación interactiva, para finalmente redimensionar la señal televisiva a su estado original. En la Figura 3.21 se presenta lo indicado.

Figura 3.21 Presentación de los integrantes que intervinieron en la aplicación interactiva

3.1.4. PRUEBAS CON EQUIPOS REALES

Una vez que las simulaciones fueron ejecutadas correctamente, el siguiente paso es realizar las mismas pruebas pero utilizando equipos reales.

El procedimiento para la ejecución de las pruebas reales son los siguientes:

a. La aplicación interactiva se almacena en una unidad USB.
b. La unidad USB se inserta en el *Set Top Box* a utilizarse, el cual es *EiTV Developer Box*, como se muestra en la Figura 3.22.

Figura 3.22 Set Top Box EiTV Developer Box

c. Una vez que se esté encendido el equipo, ejecutar la aplicación interactiva y verificar que la misma funcione de acuerdo a lo diseñado y a lo simulado.

En la Figura 3.23 se indican las imágenes capturadas de la ejecución de las pruebas con equipos reales para la aplicación interactiva enfocada en la protección ambiental. Para estas pruebas se requirió de un *Set Top Box* real, en el cual se embebió esta aplicación de forma local.

Figura 3.23 Pruebas con equipos reales para la aplicación enfocada en la protección ambiental

3.2. APLICACIÓN INTERACTIVA ENFOCADA EN LA INICIATIVA YASUNÍ ITT

La iniciativa Yasuní ITT es una propuesta que estuvo presente entre los años 2007 y 2013, y proponía dejar alrededor de 856 millones de barriles de petróleo bajo tierra dentro del Parque Nacional Yasuní, específicamente dentro de las zonas de exploración petrolera Ishpingo, Tiputini y Tambococha (ITT), a cambio de que el Estado ecuatoriano perciba montos económicos correspondientes al 50% de las utilidades que obtendría por la explotación de estos recursos (por

parte de apoyo internacional), y la condición que los países más contaminantes tomen conciencia y asuman la responsabilidad del cuidado del planeta.

Con esta propuesta se pretendía proteger a la diversidad biológica presente en el Parque Nacional Yasuní y a los territorios indígenas no intervenidos, ya que este parque fue declarado Reserva de la Biósfera por la UNESCO, por su riqueza natural y cultural [17].

3.2.1. PRESENTACIÓN DE LA INFORMACIÓN

Esta aplicación contará con un mapa con la ubicación del Parque Nacional Yasuní y presentará de forma intermitente, información de la flora y fauna más representativa de la zona. En la Figura 3.24 se muestra el mapa de ubicación que se presentará al televidente en la aplicación interactiva.

Figura 3.24 Mapa de ubicación del Parque Nacional Yasuní

Es conveniente mencionar que este parque alberga cerca de 80 especies de murciélagos, 150 anfibios, 121 reptiles, alrededor de 593 especies de aves y cerca de 4000 plantas; por lo que no sería posible cubrir a todos estos considerando las limitaciones propias de memoria y procesamiento de los *Set Top Boxes*. En la Figura 3.25 se indican cuatro especies de flora y fauna que se presentarán de forma automática en esta aplicación interactiva.

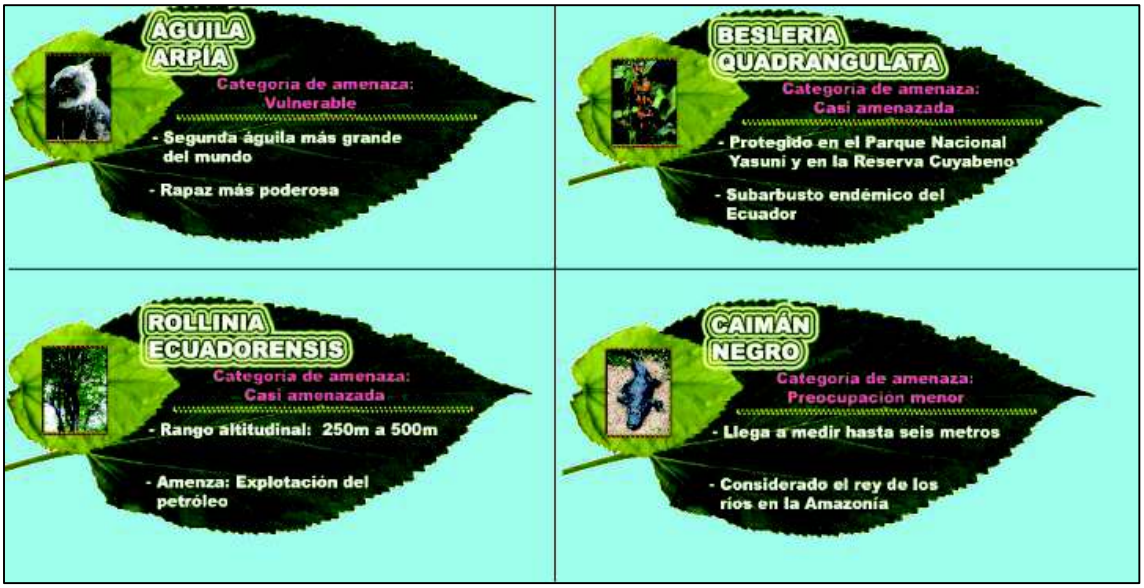

Figura 3.25 Información de algunas especies de flora y fauna que se presentarán en la aplicación interactiva enfocada en la iniciativa Yasuní ITT

También, contará con un menú informativo que permita conocer sobre información y la propuesta de la iniciativa Yasuní ITT, así como también, presentará las características negativas que conllevaría la explotación petrolera en esta zona.

En la Figura 3.26 se indica información que el televidente observará sobre la iniciativa Yasuní ITT.

Figura 3.26 Información de la aplicación interactiva enfocada en la iniciativa Yasuní ITT

En la Figura 3.27 se indican los objetivos que el televidente puede observar acerca de la iniciativa Yasuní ITT.

Figura 3.27 Objetivos de la iniciativa que la aplicación presentará

En la Figura 3.28 se indican las características negativas de la explotación petrolera en el Parque Nacional Yasuní.

Figura 3.28 Características negativas de la iniciativa que la aplicación presentará

3.2.2. DESARROLLO DE LA APLICACIÓN GINGA

Como la metodología para la creación de una aplicación interactiva, es similar a lo detallado en el numeral anterior, únicamente se va a explicar segmentos de código que no fueron presentados en la sección anterior.

En la Figura 3.29 se muestra el código necesario para la creación de un descriptor con un `id` denominado `descMurcielago`, el cual está asociado a una región denominada `rgEspecie`. El elemento multimedia asociada a esta región se presentará en la pantalla durante diez segundos. Esta última característica permite que todas las imágenes pertenecientes a la flora y fauna del Parque Nacional Yasuní se muestren al televidente sin necesidad de presionar algún botón.

```
<descriptor id="descMurcielago" region="rgEspecie" explicitDur="10s"/>
```

Figura 3.29 Código NCL para la creación del descriptor `descMurcielago`

En la Figura 3.30 se presenta el conector `onEndStart_delay`, cuyo `id` se denomina `link8`, e indica que cuando finalice la presentación del elemento multimedia `especieCaiman`, existirá un retardo de dos segundos antes de que inicie y se presente la información contenida en el elemento multimedia `especieBegonia`. Esto evitará el sobrelapamiento entre una imagen y otra al momento de ocurrir la transición entre las diferentes imágenes.

```
<link id="link8" xconnector="conn#onEndStart_delay">
  <bind role="onEnd" component="especieCaiman"/>
  <bind role="start" component="especieBegonia">
    <bindParam name="delay" value="2s"/>
  </bind>
</link>
```

Figura 3.30 Código NCL que indica el uso del enlace `onEndStart_delay`

3.2.3. EJECUCIÓN DE LA APLICACIÓN GINGA

La aplicación interactiva se inicia con la presentación de un ícono en la parte superior derecha de la pantalla, en la cual el televidente al presionar `OK` con el control remoto, accede a la información de la iniciativa Yasuní ITT.

En la Figura 3.31 se indica el lugar en donde se presenta el ícono de inicio de la aplicación interactiva dentro de la pantalla televisiva.

Figura 3.31 Ícono de inicio de la aplicación interactiva enfocada en la iniciativa Yasuní ITT

Esta aplicación interactiva divide la pantalla en tres secciones. En la primera sección ubicada en la parte superior izquierda, se encuentra toda la información relevante acerca de la iniciativa Yasuní ITT. La segunda sección, ubicada en la parte inferior izquierda, indica el mapa de ubicación del Parque Nacional Yasuní. En la tercera sección, que se encuentra en la parte inferior derecha, se presenta de manera cíclica la información de 16 especies endémicas de flora y fauna existente en la región.

Todo lo relacionado a la iniciativa Yasuní ITT, se encuentra en la parte superior izquierda de la pantalla. Si se desea revisar la información más importante de la iniciativa Yasuní ITT, el televidente debe presionar el botón `RED` del control remoto, y el resultado es el que aparece en la Figura 3.32.

Figura 3.32 Presentación de información sobre la iniciativa Yasuní ITT

Si el televidente desea observar las características positivas que promueve esta iniciativa, se debe presionar el botón `GREEN` del control remoto, el resultado se aprecia en la Figura 3.33.

Figura 3.33 Características positivas de la iniciativa Yasuní ITT

Si el televidente desea observar las características negativas que promueve esta iniciativa, se debe presionar el botón `YELLOW` del control remoto, el resultado se aprecia en la Figura 3.34.

Finalmente y cuando el televidente lo requiera, al pulsar el botón `EXIT`, finalizará la aplicación interactiva.

Figura 3.34 Características negativas de la iniciativa Yasuní ITT

3.2.4. PRUEBAS CON EQUIPOS REALES

En la Figura 3.35 se indican las imágenes capturadas de las pruebas realzadas con equipos reales para la aplicación interactiva enfocada en la iniciativa Yasuní ITT.

El procedimiento para la ejecución de las pruebas reales es el mismo que se utilizó en la aplicación interactiva anterior.

Figura 3.35 Pruebas reales para la aplicación enfocada en la protección ambiental

En las imágenes se puede apreciar que mientras la programación televisiva se visualiza en la parte superior derecha de la televisión, también se está ejecutando el contenido de la aplicación interactiva.

3.3. APLICACIÓN INTERACTIVA ENFOCADA EN EL PARQUE NACIONAL YASUNÍ SINCRONIZADA CON UN VIDEO

El Yasuní es un Parque Nacional que se encuentra ubicado en la región amazónica, dentro de las provincias de Pastaza y Orellana. Cuenta con 982.000 hectáreas. En 1979 fue declarado Parque Nacional. En 1989 fue declarada Reserva de la Biósfera por la UNESCO, por su riqueza natural y cultural.

Es relevante mencionar que este parque alberga alrededor del 44% de las aves de la cuenca del Amazonas, por lo que es considerado uno de los lugares avícolas más ricos de este planeta. Datos similares se pueden encontrar en especímenes de variedades como murciélagos, anfibios, reptiles, mamíferos, etc. [18]

3.3.1. PRESENTACIÓN DE LA INFORMACIÓN

Esta aplicación interactiva contará con un video relacionado con el Parque Nacional Yasuní en el que se podrá visualizar y escuchar información relevante y datos curiosos sobre este parque.

La idea de esta aplicación es generar contenido multimedia que refuerce las ideas presentadas por el presentador del video.

Un ejemplo de contenido multimedia se observa en la Figura 3.36. Esta información relacionada a las vías de acceso al Parque Nacional Yasuní se mostrará en la pantalla del televidente cuando el presentador del video indique que existen dos rutas para llegar al Parque Nacional Yasuní.

De igual manera, durante la emisión del video, aparecerán pequeños mensajes sobre datos curiosos de este parque, como por ejemplo la ubicación de las principales lagunas, entre otros.

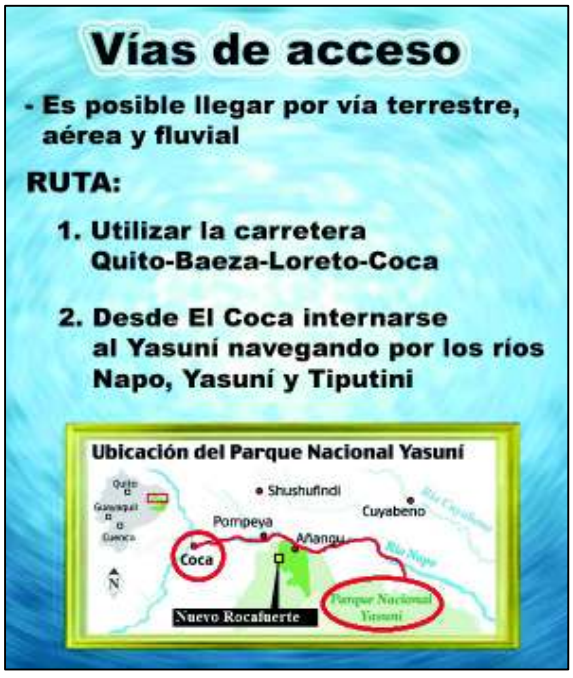

Figura 3.36 Refuerzo multimedia que se presentará cuando el video mencione las vías de acceso al Parque Nacional Yasuní

En la Figura 3.37 se muestra uno de los mensajes que se visualizarán en la pantalla del televidente, categorizado como Datos Curiosos del Parque Nacional Yasuní.

Figura 3.37 Mensaje informativo que se visualizará como dato curioso durante la emisión del video

Cabe recalcar que cierta información será presentada de forma automática, mientras que otras únicamente se visualizarán cuando el televidente presione el botón del control remoto respectivo.

3.3.2. DESARROLLO DE LA APLICACIÓN GINGA

En la Figura 3.38 se presenta el código NCL que muestra la creación de un elemento multimedia cuyo `id` se denomina `yasuni`. Este elemento se encuentra almacenado en `media/yasuni.mp4` y está asociado al `descriptor` denominado `descYasuni`. Además, se adicionan los atributos: `area` cuyos `id` se denominan `segPlesitoceno`, `segcategoriaUICN`, `segLagunaYuturi` y `segLagunaEden`, los cuales indican que cada uno de los elementos multimedia asociados a estas áreas aparecerán a los 25, 114, 413 y 414 segundos, respectivamente. Con esto se puede presentar varias imágenes durante la reproducción de todo el video, reforzando así de forma visual la información entregada al televidente.

```
<media id="yasuni" src="media/yasuni.mp4" descriptor="descYasuni">
  <area id="segPleistoceno" begin="25s"/>
  <area id="segCategoriaUICN" begin="114s"/>
  <area id="segLagunaYuturi" begin="413s"/>
  <area id="segLagunaEden" begin="414s"/>
</media>
```

Figura 3.38 Código NCL que indica la creación del elemento multimedia `yasuni`, con la asociación de sus respectivas áreas

Adicional, se deben crear los descriptores necesarios para indicar la forma en la que los elementos multimedia se presentarán en la pantalla televisiva. En la Figura 3.39 se muestra el código NCL utilizado en la creación de un descriptor con un `id` denominado `descPleistoceno`, el cual se asocia con una región denominada `rgInfoDown` e indica que tiene una duración de 12 segundos en la pantalla televisiva. Además, diferenciando con el resto de descriptores creados en las aplicaciones anteriores, el parámetro `transparency`, con un valor de 0.4, especifica que el elemento multimedia que se encuentra asociado tendrá un efecto transparente del 40% dentro de la pantalla televisiva, con respecto a la imagen original. Con esto lo que se quiere presentar es una imagen informativa al televidente sin tener que opacar o eliminar parte de las imágenes que presenta el video.

```
<descriptor id="descPleistoceno" region="rgInfoDown" explicitDur="12s">
  <descriptorParam name="transparency" value="0.4"/>
</descriptor>
```

Figura 3.39 Código NCL que indica la creación del descriptor `descPleistoceno`

3.3.3. EJECUCIÓN DE LA APLICACIÓN GINGA

La aplicación interactiva inicia con la reproducción automática del video acerca de las principales características y novedades que presenta el Parque Nacional Yasuní. En la Figura 3.40 se indica que el video se ejecuta correctamente.

Figura 3.40 Reproducción del video acerca del Parque Nacional Yasuní

En la Figura 3.41 se puede observar que durante la reproducción del video, aparece una imagen de color verde en la parte superior derecha. Esta imagen quiere decir que existe información adicional sobre el relato que se menciona en el video, y, para que el televidente tenga acceso a esta información, debe presionar el botón de este mismo color en el control remoto.

Figura 3.41 Imagen que señala que existe información adicional sobre el relato del video en reproducción

Si el televidente presiona el botón del control remoto indicado en la figura anterior, se desplegará la información respectiva en la pantalla televisiva. En este caso la información se aprecia en la parte derecha de la pantalla, de acuerdo a lo presentado en la Figura 3.42.

Figura 3.42 Información adicional que se indica en la reproducción del video

En la Figura 3.43 se indica que aparece de forma gráfica la ubicación geográfica de las lagunas que se detallan durante el relato. Esta información se presenta automáticamente y sirve para que el televidente pueda reforzar la información obtenida durante el relato del video.

Figura 3.43 Información acerca de la ubicación de las lagunas que se describen el relato del video

3.3.4. PRUEBAS CON EQUIPOS REALES

Para la sincronización del video, se debe utilizar herramientas que transformen formatos de video convencionales en *Transport Stream* (TS)[16]. Para ello se debe recurrir a una herramienta de código abierto denominado *OpenCaster*. Cabe recalcar que este trabajo de titulación no explica a detalle el funcionamiento del *Transport Stream* ni tampoco la instalación de la herramienta *OpenCaster*. Para obtener mayor información sobre estos temas se recomienda leer [19].

Para generar correctamente el *Transport Stream* con la aplicación interactiva, se requiere de los siguientes elementos:

- Archivo de video

[16] TS: protocolo de comunicación para audio, vídeo y datos especificado en los estándares de MPEG-2

- Archivo de audio
- Creación de las tablas SI/PSI (*Service Information* / *Program Specific Information*)[17]
- Generación del carrusel de objetos de la aplicación
- Multiplexar el *Transport Stream* con la aplicación interactiva

El procedimiento para generar el TS es el siguiente:

a. En el directorio donde se va a trabajar, se debe colocar los archivos necesarios para la generación del *Transport Stream*: `null.ts` (archivo que no contiene información), `tablas.py` (archivo para crear las tablas SI/PSI), `yasuni.avi` (archivo de video) y `aplicación_03` (directorio que contiene la aplicación interactiva), tal y como se aprecia en la Figura 3.44. El archivo `yasuni.avi` se obtuvo de Internet a través del canal de Youtube. Tiene una resolución de 480 x 320 píxeles; el códec de video que utiliza es `MPEG4-Video` y el códec de audio es `MPEG-1 capa 3`.

Figura 3.44 Archivos utilizados para la generación del *Transport Stream*

b. En la Figura 3.45 se ejecuta el *script* escrito en *python* definido en el archivo `tablas.py`, lo cual generará las tablas SI/PSI.

Figura 3.45 Comando que ejecuta el script `tablas.py`

[17] Tablas SI/PSI: Tablas que definen la estructura del *Transport Stream*

En la Figura 3.46 se observan todos los archivos generados automáticamente, después de ejecutarse el *script* del archivo `tablas.py`.

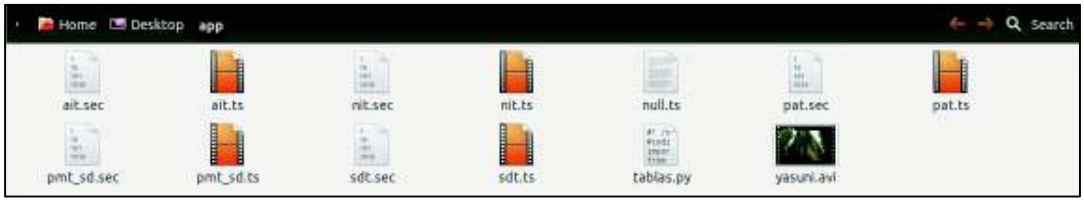

Figura 3.46 Tablas SI/PSI generadas y presentadas en el directorio

c. En la Figura 3.47 se observa el comando que se utiliza para realizar la codificación del audio del archivo `yasuni.avi`.

```
root@ubuntu: ~/Desktop/app
root@ubuntu:~/Desktop/app# ffmpeg -i yasuni.avi -vn -ac 2 -acodec mp2 -f mp2 -ab
128000 -ar 48000 yasuni.mp2
```

Figura 3.47 Comando utilizado para codificación de audio

Los parámetros utilizados para codificar el audio son:

- `-i`: permite indicar el archivo `yasuni.avi`, el cual es el archivo fuente de video que va a ser codificado.
- `-vn`: se utiliza para codificar exclusivamente el audio, independientemente de la señal de video que se encuentre presente.
- `-ac`: define el número de canales de audio. El número 2 indica que se obtendrá audio estéreo.
- `-acodec`: especifica el códec de audio. El formato compatible para *OpenCaster* es `mp2`.
- `-f`: especifica el formato de salida de audio. Para este caso se utiliza, de igual manera, `mp2`.
- `-ab`: indica la velocidad de transmisión utilizada por el audio. El estándar MPEG define este valor en 128 kbps.
- `-ar`: indica la frecuencia de muestreo a la que la señal de audio será procesada. El estándar MPEG define este valor en 48000 Hz.

- `yasuni.mp2`: especifica el nombre del archivo del audio codificado.

d. En la Figura 3.48 se observa el comando que se utiliza para realizar la codificación del video del archivo `yasuni.avi`

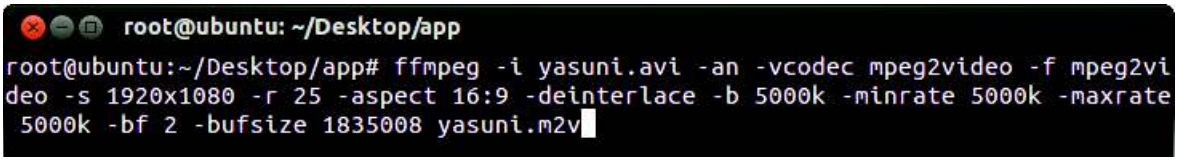

Figura 3.48 Comando utilizado para codificación de video

Los parámetros utilizados para codificar el video son:

- `-an`: se utiliza para codificar exclusivamente el video, independientemente de la señal de audio que se encuentre presente.
- `-vcodec`: especifica el códec de video. El formato compatible para *OpenCaster* es `mpeg2video`.
- `-f`: especifica el formato de salida de video. Para este caso se utiliza, de igual manera, `mpeg2video`.
- `-s`: especifica la resolución del video en píxeles. En este caso se eligió la opción 1920x1080.
- `-r`: especifica los cuadros por segundo del video. En este caso se eligió 25 FPS (*Frames per Second*)
- `-deinterlace`: elimina el entrelazado de las imágenes para obtener una mejor calidad de video.
- `-b`: indica la velocidad de transmisión utilizada por el video. En este caso, este valor se definió en 5.000 kbps.
- `-minrate`: indica la velocidad de transmisión mínima utilizada por el video. En este caso, este valor se definió en 5.000 kbps.
- `-maxrate`: indica la velocidad de transmisión máxima utilizada por el video. En este caso, este valor se definió en 5.000 kbps.
- `-bf`: especifica el número de imágenes bidireccionales que se utilizarán para la compresión. A mayor número de imágenes bidireccionales, mayor

compresión, y por consiguiente, menor calidad de la imagen. En este caso se definió como valor el número 2.

- `-bufsize`: especifica el tamaño del buffer, para que el *stream* de video se codifique correctamente. El valor por defecto es 1'835.008.
- `yasuni.m2v`: especifica el nombre del archivo del video codificado.

e. En la Figura 3.49 se observa el comando utilizado para obtener un PES[18] (*Packetized Elementary Stream*) de audio.

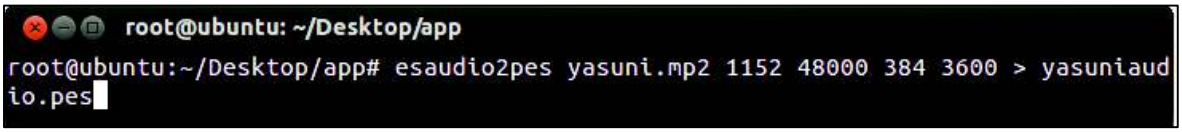

Figura 3.49 Comando utilizado para la obtención de un PES de audio

Los parámetros utilizados para obtener el PES son:

- `1152`: especifica el número de muestras por cuadro en una trama de audio.
- `48000`: indica la frecuencia de muestreo definida en el estándar MPEG.
- `384`: especifica el tamaño de la trama de audio.
- `3600`: define el valor del primer PTS[19] (*Presentation Time Stamp*) para mantener la sincronización adecuada entre audio y video.
- `yasuniaudio.pes`: especifica el nombre del archivo de audio que contiene el *stream* después de ser encapsulado dentro de un paquete PES.

f. En la Figura 3.50 se observa el comando utilizado para obtener un PES de video.

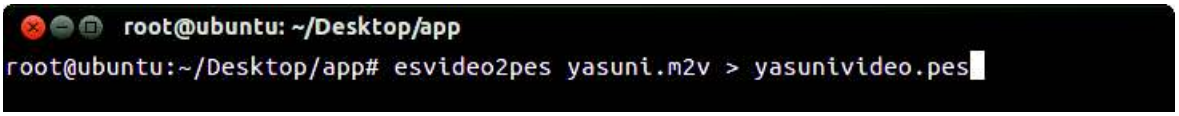

Figura 3.50 Comando utilizado para la obtención de un PES de video

---

[18] PES: Paquetes que definen el transporte de *streams* elementales en paquetes dentro de flujos de programas MPEG y flujos de transporte MPEG.
[19] PTS: Secuencia de programa MPEG que se utiliza para lograr la sincronización de las secuencias elementales separadas de audio y video.

g. En la Figura 3.51 se observa el comando utilizado para obtener un TS de audio. En esta línea de comando se encapsula el paquete PES de audio obtenido mediante la línea de comando según lo presentado en la Figura 3.49.

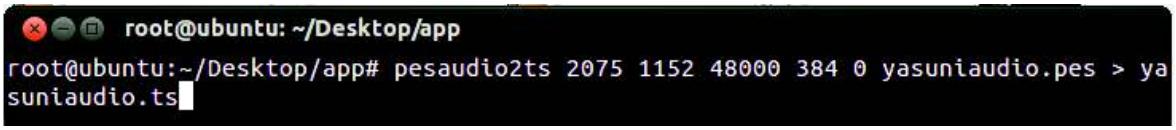

Figura 3.51 Comando utilizado para la obtención de un TS de audio

Los parámetros utilizados para obtener el TS de audio son:

- `2075`: PID (*Packet* ID) que se asigna a los TS de audio.
- `1152`: indica el número de muestras por cuadro dentro de una trama de audio.
- `48000`: indica la frecuencia de muestreo. Este valor se encuentra definido en el estándar MPEG.
- `384`: indica el tamaño de la trama de audio.
- `0`: se especifica para indicar que el audio no se repita continuamente.

h. En la Figura 3.52 se observa el comando utilizado para obtener un TS de video. En esta línea de comando se encapsula el paquete PES de video obtenido mediante la línea de comando presentada en la Figura 3.50.

Figura 3.52 Comando utilizado para la obtención de un TS de video

Los parámetros utilizados para obtener el TS de video son:

- `2065`: PID que se asigna a los TS de video.
- `25`: especifica el número de cuadros por segundo.
- `112`: especifica el valor del búffer, el cual está definido en el estándar MPEG.

- `5750000`: especifica la velocidad de transmisión incrementado en un 15% (de acuerdo a lo recomendado en la documentación de *OpenCaster*).
- `0`: se especifica para indicar que el video no se repita continuamente.

i. En la Figura 3.53 se observa el comando utilizado para generar el carrusel de objetos de la aplicación interactiva.

Figura 3.53 Comando utilizado para la generación de carrusel de objetos

Los parámetros utilizados para obtener el carrusel de objetos son:

- `aplicación_03`: especifica el directorio en donde se almacena la aplicación interactiva.
- `0x0C`: especifica la etiqueta de asociación del carrusel generado, el cual se escribe en las tablas AIT[20] (*Application Information Table*) y PMT[21] (*Program Map Table*), pertenecientes a las tablas SI/PSI.
- `1`: especifica el número de versión del carrusel generado.
- `2004`: especifica el PID de identificación del carrusel generado
- `2`: identifica el carrusel generado para colocarlo dentro de la tabla PMT.

j. En la Figura 3.54 se observa el comando utilizado para la multiplexación de las tablas, los archivos TS y el carrusel de objetos generados mediante línea de comando en las Figuras 3.46, 3.51, 3.52 y 3.53.

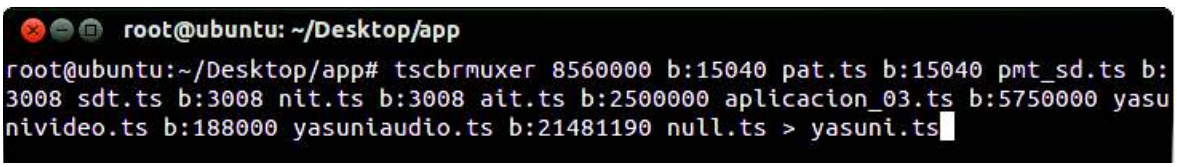

Figura 3.54 Comando utilizado para la multiplexación de archivos TS de audio y video

[20] Tabla AIT: Tabla que contiene información de la aplicación interactiva.
[21] Tabla PMT: Tabla que especifica el PID de cada flujo de datos que constituyen un programa.

Los parámetros utilizados para obtener la multiplexación de las tablas, los archivos TS y el carrusel de objetos son:

- `8560000`: es la cantidad de paquetes a multiplexar.
- `b: 15040`: ancho de banda (bps) que requieren las tablas `pat.ts` y `pmt_sd.ts` para ser transmitidas correctamente (de acuerdo a lo recomendado en la documentación de *OpenCaster*).
- `b: 3008`: ancho de banda (bps) que requieren las tablas `sdt.ts`, `nit.ts` y `ait.ts` para ser transmitidas correctamente (de acuerdo a lo recomendado en la documentación de *OpenCaster*).
- `b: 250000`: ancho de banda (bps) que requiere la aplicación interactiva.
- `b: 5750000`: ancho de banda (bps) que requiere el TS de video.
- `b: 188000`: ancho de banda (bps) que requiere el TS de audio.
- `b: 21481190`: ancho de banda (bps) que requieren los paquetes nulos. Como el sistema ISDB-T tiene un ancho de banda fijo de 29'958.294 bps, se deben restar los valores anteriores para obtener el ancho de banda de los paquetes nulos.

k. Como se estuvo cambiando la posición de los paquetes de audio y video durante la codificación, la llegada de los mismos al *Set Top Box* no será la correcta. Por tal motivo, en la Figura 3.55 se observa el comando utilizado para ordenar correctamente estos paquetes de audio y video. El archivo final ordenado se especifica como `yasuni.fixed.ts`.

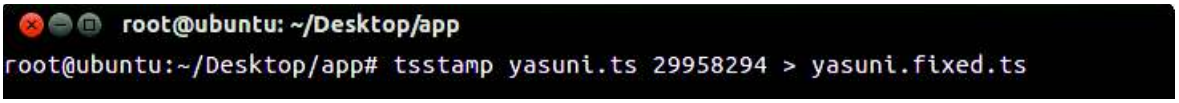

Figura 3.55 Comando utilizado para la sincronización de audio y video en el *Set Top Box*

En la Figura 3.56 se indica el archivo TS final que se transmitirá en el interfaz aire. Este archivo contiene el audio, el video y la aplicación interactiva de forma sincronizada.

Figura 3.56 Archivo TS sincronizado entre video y aplicación interactiva

Para la transmisión del TS `yasuni.fixed.ts` al Set Top Box, se utilizó el transmisor UT-200/UT210 ISDBT y la herramienta *HiDes TS Player*. En la Figura 3.57 se muestra el hardware del transmisor UT-200/UT210 ISDBT.

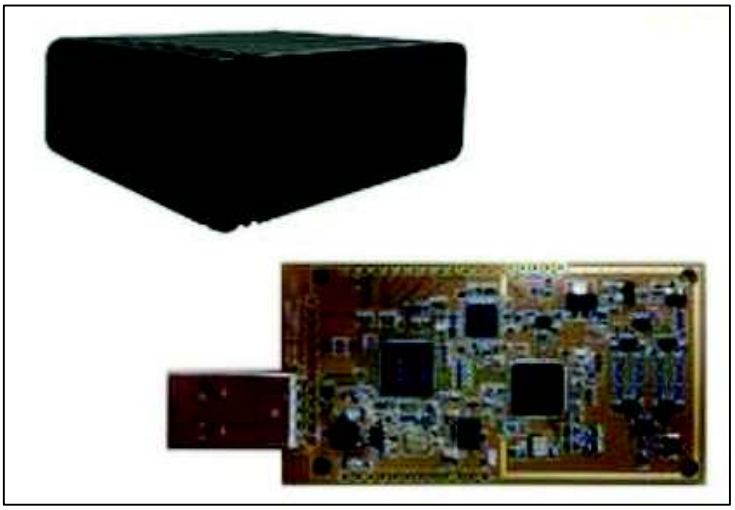

Figura 3.57 Transmisor UT-200/UT210 ISDBT

En la Figura 3.58 se muestra la interfaz de la herramienta *HiDes TS Player*, la cual es la encargada de transmitir el TS al *Set Top Box*. Los parámetros importantes que se configuraron para esta transmisión son:

- Frecuencia: 195143 (VHF 8 ch)
- Intervalo de guarda: (3) 1/4
- Modo de transmisión: (0) 2k:mode 1
- Tasa de código: (3) 5/6
- Constelación: (1) 16 QAM

Además la velocidad de transmisión de salida del transmisor debe ser igual o mayor que los datos de entrada del archivo `yasuni.fixed.ts`.

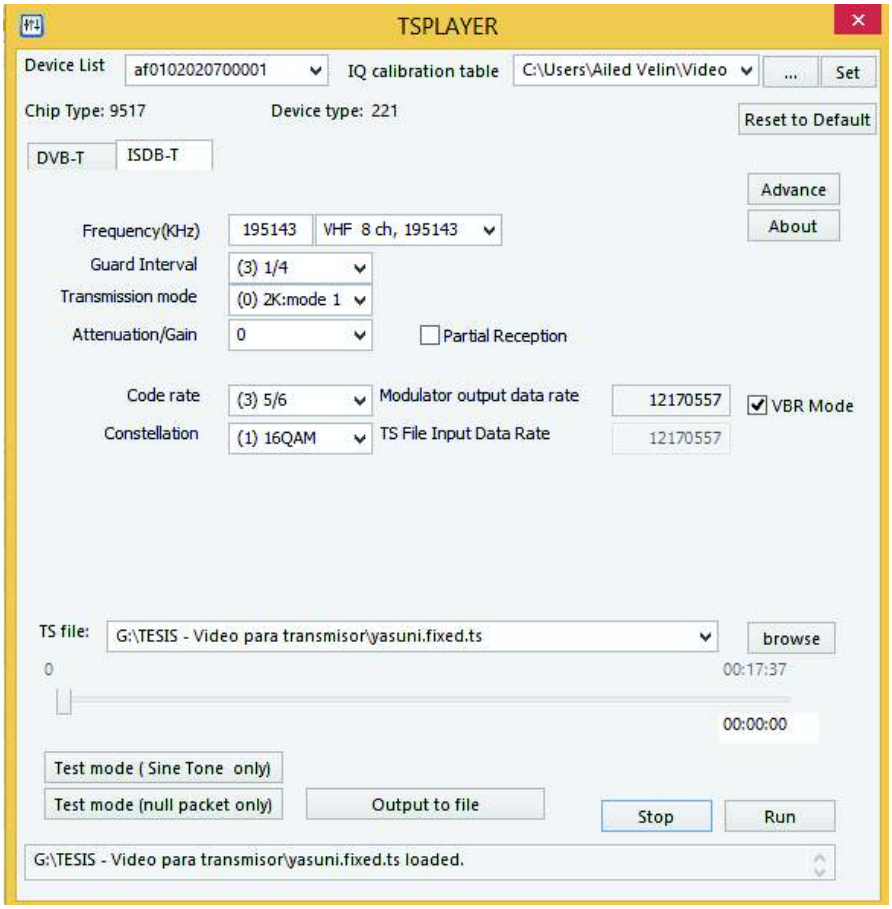

Figura 3.58 Interfaz de la herramienta *HiDes TS Player*

Figura 3.59 Pruebas con equipos reales para la aplicación enfocada en la protección ambiental sincronizada con el video del Parque Nacional Yasuní

Una vez generado el archivo necesario con la respectiva sincronización, se observa que en la Figura 3.59 se presenta las capturas de pantalla, en el que se aprecia que durante la reproducción del video, van apareciendo imágenes informativas que refuerzan la información relatada del video.

3.4. APLICACIÓN INTERACTIVA ENFOCADA EN EL RECICLAJE, GENERADA CON EL PLUG-IN DESARROLLADO

Es importante presentar esta aplicación, puesto que en la actualidad el reciclaje es uno de los procesos que permite crear responsabilidad social sobre el cuidado del medio ambiente, para así intentar generar conciencia para salvar a especies de flora y fauna que se encuentran en peligro de extinción.

3.4.1. PRESENTACIÓN DE LA INFORMACIÓN

La información se presentará en una región de la televisión, que tiene relación con los ciclos que contiene el reciclaje, la razón del porqué se debe reciclar y la manera de cómo proceder. Además, existirá una región adicional en donde se despliegue las preguntas más comunes que la ciudadanía tiene sobre el reciclaje. Las respuestas a estas preguntas se desplegarán cuando el televidente presione con el control remoto la respuesta que quiera ver de acuerdo al número de pregunta.

En la Figura 3.60 se muestra el menú principal generado para tener un mayor conocimiento sobre el proceso de reciclar.

Figura 3.60 Menú desplegado para obtener información del reciclaje

En la Figura 3.61 se observa todo el ciclo que tiene el reciclaje. La información de este ciclo avanzará mientras se presiona los botones de flecha izquierda y derecha del control remoto. El inicio del ciclo es el Origen del reciclaje y culmina con el Reciclado Final.

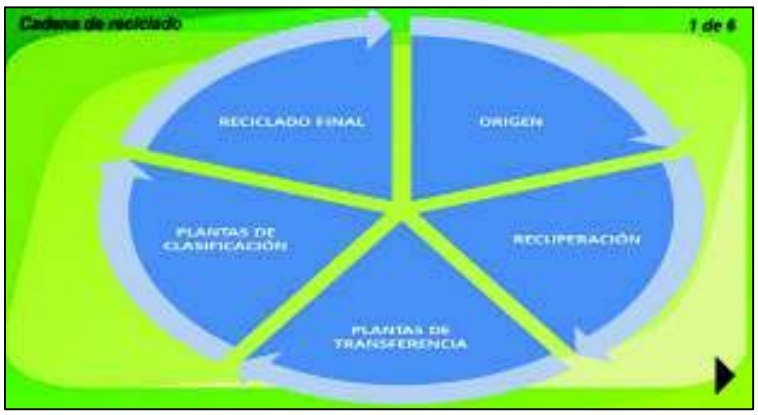

Figura 3.61 Información presentada al televidente sobre el ciclo de reciclaje

3.4.2. DESARROLLO DE LA APLICACIÓN GINGA

Para el desarrollo de esta aplicación se recurrirá al uso del *plug-in*, que se implementó en el capítulo anterior.

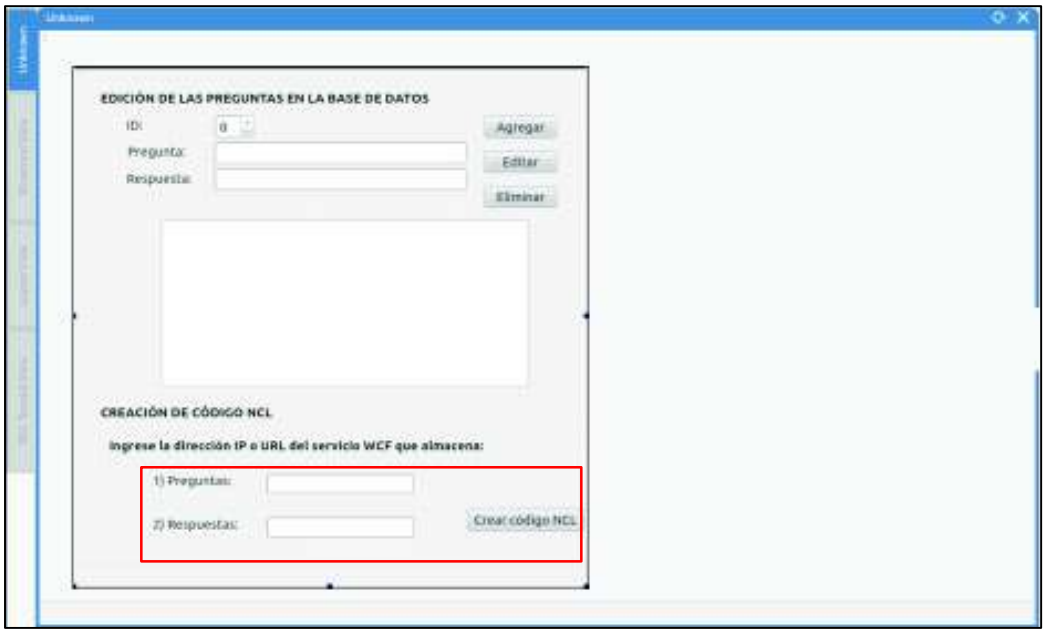

Figura 3.62 Utilización del *plug-in* para la creación de la aplicación interactiva enfocada en el reciclaje

Únicamente se debe colocar la URL del servicio web donde se encuentran las preguntas y respuestas relacionadas al reciclaje. Una vez ingresado este valor se debe presionar el botón `Crear código NCL`. En el recuadro remarcado de color rojo de la Figura 3.62 se observan los campos en donde deben ser colocadas las URL respectivas.

Una vez ingresado las URL, se generará el código NCL necesario para la aplicación interactiva. En la Figura 3.63 se presenta una imagen de todo el código NCL generado por el *plug-in*.

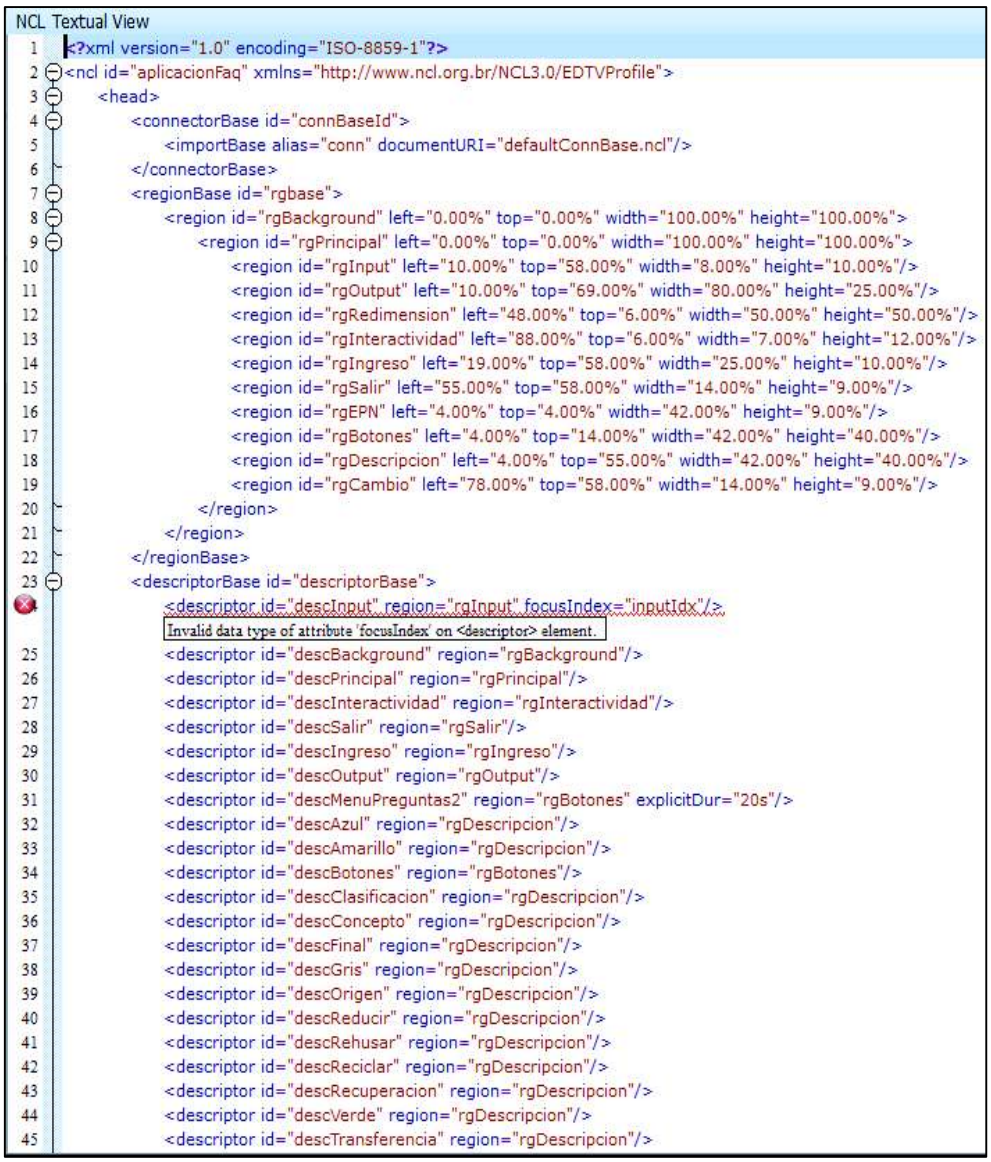

Figura 3.63 Código NCL implementado para la aplicación interactiva enfocado en el reciclaje

3.4.3. EJECUCIÓN DE LA APLICACIÓN GINGA

En la Figura 3.64 se indica el lugar en donde se presenta el ícono de inicio de la aplicación interactiva dentro de la pantalla televisiva.

Esta aplicación interactiva, presenta una lista de preguntas frecuentes, todas ellas relacionadas con el reciclaje, en la cual el televidente puede elegir, por medio del control remoto, para obtener la respuesta asociada. La elección de esta pregunta lo realiza digitando el número correspondiente mediante el uso del control remoto sobre el recuadro respectivo situado en la parte central izquierda de la pantalla.

Figura 3.64 Ícono de inicio de la aplicación interactiva bidireccional

En la Figura 3.65 se presentan todas las secciones de la aplicación interactiva. En la parte inferior (recuadro negro) se presentará la respuesta, la cual se obtendrá a través del canal de retorno.

En la Figura 3.66 se observa la respuesta a la pregunta elegida por el televidente. El televidente escogió la pregunta dos, y esta petición viajó a través del canal de retorno, se obtuvo una respuesta, se lo procesó y se presentó el resultado en la parte inferior.

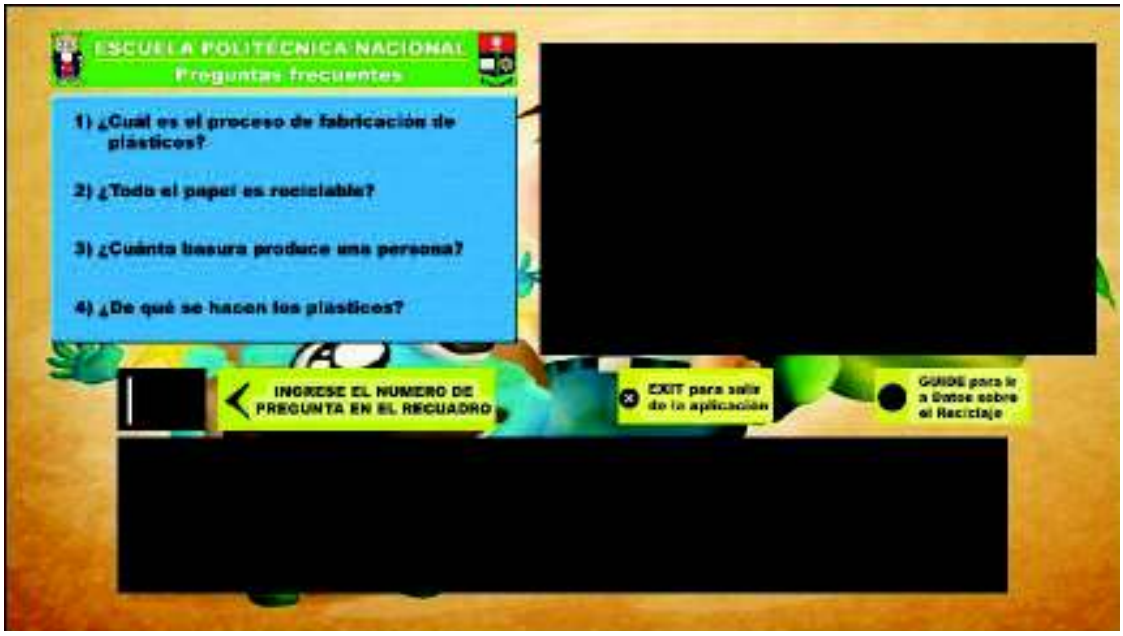

Figura 3.65 Presentación de las características en la aplicación interactiva

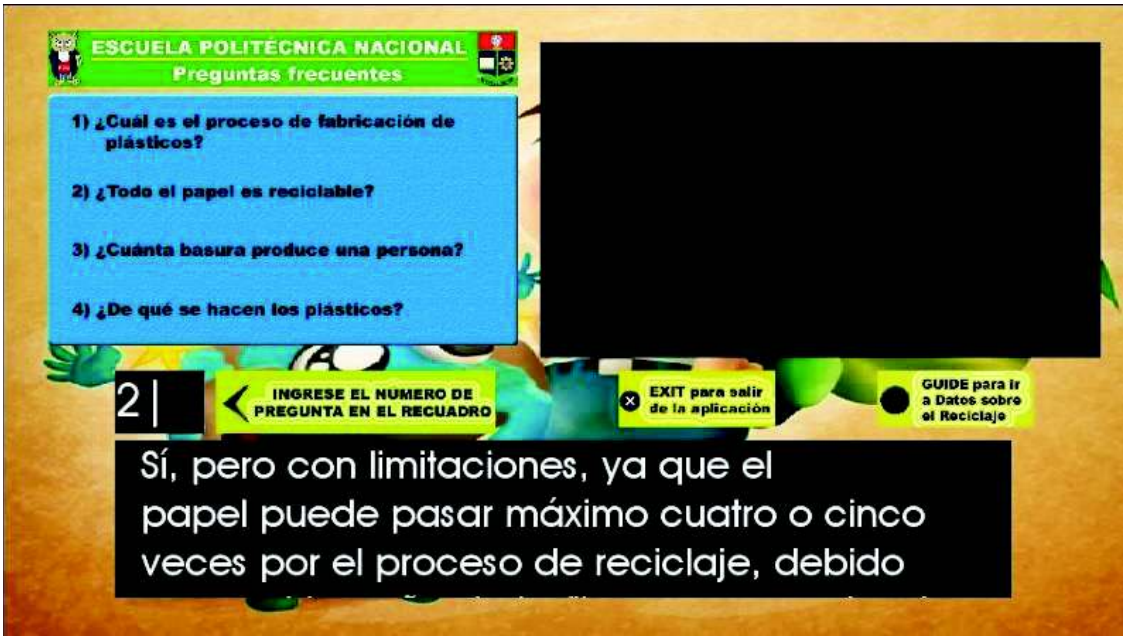

Figura 3.66 Presentación de la respuesta a la pregunta dos

3.4.4. PRUEBAS CON EQUIPOS REALES

En la Figura 3.67 se indican las imágenes que muestran la ejecución de la aplicación interactiva enfocada en el reciclaje mientras la programación televisiva se sigue presentando en la parte superior derecha de la televisión.

Figura 3.67 Pruebas con equipos reales para la aplicación enfocada en el reciclaje, usando el canal de retorno

3.5. ENCUESTAS MOS DE LAS APLICACIONES INTERACTIVAS

Las Encuestas MOS (*Mean Opinion Score*) se realizan para obtener un indicador de calidad sobre el resultado final de las aplicaciones interactivas generadas. Estas encuestas se expresan en valores numéricos promedio y sus niveles de aceptación son:

- 5: Excelente
- 4: Muy bueno
- 3: Bueno
- 2: Regular
- 1: Malo

Las encuestas sobre las aplicaciones interactivas se realizaron a un grupo de 50 personas, cuyas edades oscilaban entre 20 y 45 años. Los parámetros que se tomaron en cuenta para medir la calidad de cada aplicación interactiva fueron: interfaz gráfica, facilidad de uso de la aplicación y contenido informativo.

Para la realización de las encuestas, se reunió a todas las personas en la Sala de Reuniones de un Conjunto Habitacional. Ahí se explicó brevemente la forma en la que se debe utilizar el control remoto para interactuar con las aplicaciones y los parámetros que deben tener en cuenta al momento de ser encuestados.

Para cada aplicación se establecieron nueve preguntas, de las cuales las cinco primeras son comunes para todas las aplicaciones y las restantes son preguntas exclusivas de cada aplicación.

En la Tabla 3.1 se presentan los valores promedio que se obtuvieron después de que todas las personas observaron e interactuaron con la aplicación interactiva enfocada en la protección ambiental. Cabe recalcar que estos valores son positivos porque la experiencia del usuario para el manejo de la aplicación, así como su contenido fue el adecuado, considerando la heterogeneidad de la edad de las personas presentes.

Pregunta	Promedio
1) ¿Es completamente visible el ícono de inicio de interactividad de la aplicación interactiva?	4.7
2) ¿Es legible en la televisión la información presentada de la aplicación?	4.6
3) Dificultad que tuvo para el uso del control remoto	4
4) ¿Existió distorsión de imágenes o texto durante la ejecución de la aplicación interactiva?	4.3
5) Durante la ejecución de la aplicación, ¿pudo observar con atención la programación televisiva?	4
6) ¿Tenía conocimiento de las especies que se encuentran en peligro de extinción?	4.5
7) ¿Usted conocía las características generales de cada una de las especies presentadas?	4.1
8) La información presentada, ¿es útil para concientizar a la ciudadanía sobre el problema de extinción existente en algunas especies de animales?	4.8
9) ¿Cuál es su apreciación general sobre la aplicación presentada?	4.7

Tabla 3.1 Encuesta MOS de la aplicación enfocada en la protección ambiental

En la Tabla 3.2 se muestran los resultados de la encuesta sobre la aplicación interactiva enfocada en la iniciativa Yasuní ITT. Estos resultados muestran que el

contenido fue de gran interés para el televidente, ya que algunos de los temas indicados eran de desconocimiento general, puesto que actualmente, no existe gran difusión informativa sobre la iniciativa Yasuní ITT.

En cuanto a la interfaz gráfica, existen valores que indican que algunas imágenes o texto presentados no se encontraban muy claros durante su presentación. Esto ocurrió porque la elección de la combinación de colores no fue adecuada para separar imágenes de fondo con respecto al contenido informativo de la aplicación. Después de estos resultados, se procedió a cambiar los colores de fondo para que el contenido se muestre de mejor manera al televidente.

Pregunta	Promedio
1) ¿Es completamente visible el ícono de inicio de interactividad de la aplicación interactiva?	4.2
2) ¿Es legible en la televisión la información presentada de la aplicación?	4.6
3) Dificultad que tuvo para el uso del control remoto	4.2
4) ¿Existió distorsión de imágenes o texto durante la ejecución de la aplicación interactiva?	4.8
5) Durante la ejecución de la aplicación, ¿pudo observar con atención la programación televisiva?	4.2
6) ¿Tenía conocimiento sobre la información general de la iniciativa Yasuní ITT?	4.2
7) ¿Usted conocía la ubicación geográfica del Parque Nacional Yasuní?	4
8) ¿Usted conocía de algunas de las especies de flora y fauna que se encuentran en el Parque Nacional Yasuní?	4.8
9) ¿Cuál es su apreciación general sobre la aplicación presentada?	4.9

Tabla 3.2 Encuesta MOS de la aplicación enfocada en la iniciativa Yasuní ITT

En la Tabla 3.3 se muestran los resultados de la encuesta de la aplicación enfocada en la protección ambiental sincronizado con un video. El video que se

presentó fue un pequeño documental sobre el Parque Nacional Yasuní, y la principal dificultad que se observa en los televidentes es la lectura de la información que se muestra. Esto ocurre porque parte de la información se presenta automáticamente sin la necesidad de utilizar el control remoto, por lo que les tomó de sorpresa a algunos televidentes y no alcanzaban a leer todo el contenido. Sin embargo, consideraron que la aplicación posee calidad en cuanto a su contenido.

También, los televidentes evidenciaron en ocasiones que el video se congelaba o se distorsionaba. Este problema se presentó porque el transmisor utilizado no es de uso corporativo, por lo que presenta bastantes limitaciones durante su funcionamiento.

Pregunta	Promedio
1) ¿Son completamente visibles los íconos de interactividad de la aplicación interactiva?	4.2
2) ¿Es legible en la televisión la información presentada de la aplicación?	4
3) Dificultad que tuvo para el uso del control remoto	4
4) ¿Existió distorsión de imágenes o texto durante la ejecución de la aplicación interactiva?	4
5) Durante la ejecución de la aplicación, ¿pudo observar con atención la programación televisiva?	4.5
6) ¿Usted tuvo dificultad para leer todo el contenido informativo que se presentó durante toda la transmisión del video?	4.1
7) ¿Se observó con claridad, y se presentó adecuadamente la ubicación geográfica de los ríos y lagos mencionados en el video?	4.5
8) Después de observar el video y la aplicación conjuntamente, ¿le agradaría idea de conocer el Parque Nacional Yasuní?	4.8
9) ¿Cuál es su apreciación general sobre la aplicación presentada?	4.6

Tabla 3.3 Encuesta MOS de la aplicación enfocada en la protección ambiental sincronizada con un video

En la Tabla 3.4 se muestran los resultados de la encuesta realizada para la aplicación enfocada en el reciclaje. Esta aplicación fue desarrollada por el *plug-in* y utilizó el canal de retorno, por lo que la expectativa de esta aplicación era alta.

Los resultados de cada una de las preguntas muestran que el contenido les permitió a los usuarios conocer más sobre el proceso del reciclaje. Así mismo, al momento de obtener las respuestas, existía una pequeña demora en aparecer en la pantalla. Esto se debió porque existía una pequeña latencia al momento de utilizar el canal de retorno para consultar y presentar la respuesta en la pantalla televisiva. Esto se soluciona colocando una mejor capacidad en el ancho de banda del canal de retorno, y en lo posible no recurrir al uso de redes inalámbricas porque esto causa mayor retardo para la presentación de una respuesta, que al final puede causar en molestia para el televidente.

Pregunta	**Promedio**
1) ¿Es completamente visible el ícono de inicio de interactividad de la aplicación interactiva?	4.8
2) ¿Es legible en la televisión la información presentada de la aplicación?	4.5
3) Dificultad que tuvo para el uso del control remoto	4.3
4) ¿Existió distorsión de imágenes o texto durante la ejecución de la aplicación interactiva?	4
5) Durante la ejecución de la aplicación, ¿pudo observar con atención la programación televisiva?	4.3
6) ¿Usted tenía conocimiento sobre el proceso de reciclaje?	4.7
7) ¿Usted tuvo dificultad para elegir alguna pregunta que se mostró en la aplicación?	4.4
8) ¿Usted quedó satisfecho con las respuestas que se presentaron a cada pregunta de la aplicación?	4.7
9) ¿Cuál es su apreciación general sobre la aplicación presentada?	4.7

Tabla 3.4 Encuesta MOS de la aplicación enfocada en el reciclaje

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- Al realizar las simulaciones de las aplicaciones interactivas se concluye que algunas imágenes y elementos no actúan de la forma esperada, pero esto no quiere decir que el desarrollo de estas aplicaciones no funcione, puesto que existen ciertos errores de comunicación entre las herramientas NCL Composer, Ginga NCL y Virtual STB, especialmente porque estas herramientas se encuentran desarrolladas sobre diferentes sistemas operativos.
- Para el desarrollo del *plug-in* se debe tener mucho cuidado con la versión del lenguaje Qt que se está utilizando y la versión de la herramienta NCL Composer, para evitar problemas al momento de su respectiva compilación. Para observar cuáles deben ser las versiones adecuadas se recomienda dirigirse a la página oficial de NCL Composer o contactarse con uno de los integrantes del grupo de desarrollo y solicitar la información requerida.
- En el uso de *OpenCaster* para la generación de archivos TS se debe ser muy cauteloso con los parámetros que se digitan en cada sentencia ejecutada, puesto que si alguna de ellas es digitada de manera errónea, el video resultante no se transmitirá adecuadamente entre el transmisor y el STB.
- Las imágenes y los elementos presentes en una aplicación interactiva deben ser muy livianos en cuanto a su espacio en disco se refiere, puesto que el peso total de una aplicación no debe exceder los 6 MB. Si este peso se sobrepasa existirá una sobrecarga en el STB y por consiguiente la aplicación no se ejecutará de acuerdo a lo esperado por el televidente.
- Para el desarrollo de aplicaciones interactivas, se debe utilizar elementos e imágenes fáciles de entender y leer para los usuarios finales, porque se

debe tener en cuenta que las aplicaciones se difundirán a un grupo extenso y heterogéneo de personas.

- Si se quiere desarrollar aplicaciones interactivas utilizando el canal de retorno, se debe obligatoriamente recurrir a lenguaje LUA como mecanismo de apoyo, ya que el lenguaje NCL por sí solo no es capaz de realizar este tipo de interacciones.
- Al momento de modificar varios elementos en las diferentes vistas que presente el software NCL Composer, siempre se debe esperar un cierto tiempo hasta que todas sus vistas se sincronicen, ya que, si por error o descuido no se sincroniza o se interrumpe esta operación, los cambios efectuados para el desarrollo de una aplicación interactiva no se verán reflejados y su ejecución presentará comportamientos anómalos.
- Al momento de ejecutar las pruebas reales utilizando el canal de retorno, se deben configurar correctamente los parámetros de red, tanto del servicio WCF como del STB, para transmitir correctamente toda la información deseada. Además, en caso de no obtener lo esperado se debe configurar el Firewall del servidor para permitir los puertos necesarios a los que se requiere establecer comunicación.
- Para la transmisión del video sincronizado, se recurrió al uso del transmisor UT-200/UT-210 ISDBT a través de la herramienta *HiDes TS Player*, el cual es un reproductor de flujo de transporte para sistema operativo Windows y puede leer y reproducir transmisiones de transporte compatibles con MPEG-2.
- La implementación de las cuatro aplicaciones interactivas intenta concientizar a la ciudadanía sobre la problemática actual existente en el medio ambiente y todas las soluciones que se encuentran a nuestro alcance para rescatar, principalmente, a todas las especies de flora y fauna que se encuentran en peligro de extinción.
- La idea de la implementación de la aplicación interactiva sincronizada con un video, permite reforzar todas las ideas que se indican en dicho video, ya sea a través de imágenes o frases aclarativas, para que el televidente pueda obtener una mejor percepción de todas las ideas y las precauciones que se deben tener para proteger la reserva del Parque Nacional Yasuní.

- La creación de un *plug-in* para la herramienta NCL Composer facilita a un usuario la creación, y posterior implementación, de una aplicación interactiva, sin la obligatoriedad de tener el *expertise* en lenguaje NCL. Esto permitirá que más personas se involucren activamente en la generación de contenidos positivos sobre varios temas de interés social, político, médico, humano, etc.
- Las simulaciones de las pruebas que se realizan sobre las aplicaciones interactivas requieren de una máquina virtual, el cual contiene embebido un STB virtual, que posee sus limitaciones propias de hardware, razón por la cual no aseguran la calidad que se requeriría, ya que únicamente se verifica el funcionamiento de la aplicación, mas no su comportamiento frente a un televisor y un STB real, por lo que al momento de realizar las pruebas reales, se pueden observar diferencias a nivel visual y de *performance* sobre las simulaciones realizadas.

4.2. RECOMENDACIONES

- Para la creación de herramientas adicionales que permitan extender la funcionalidad de NCL Composer, se recomienda tener conocimientos básicos de lenguaje C++ y de Qt, ya que son requisitos indispensables para unir correctamente un *plug-in* con el núcleo de la herramienta en mención.
- Se recomienda revisar el manual o seguir algún curso de capacitación, para entender de mejor manera el uso de la herramienta *HiDes TS Player*, ya que es un requisito fundamental si se necesita transmitir video en alta definición, con sus respectivas aplicaciones interactivas, hacia el STB.
- Se recomienda revisar la manera de compilar el código fuente del sistema NCL Composer con la herramienta *cmake*, para no tener problemas con la ejecución de esta al momento de adjuntar cualquier *plug-in* generado por la herramienta Qt Creator.
- Es recomendable generar correctamente los contratos de servicio dentro del servicio WCF, para consumir o utilizar adecuadamente esta herramienta dentro de la programación del lenguaje LUA.

- Se debe tener cuidado al momento de conectarse al servicio WCF desde la herramienta Qt Creator, ya que los *drivers* de conexión que usa son distintos si se intenta compilar en el sistema operativo Windows que en un equipo con sistema operativo Linux. En este punto se verificó que algunos *drivers* de conexión no son compatibles, por lo que se debe recurrir al uso de paquetes o herramientas externos, para abrir o cerrar satisfactoriamente la conexión mencionada.
- Para realizar transmisiones de video sobre una aplicación interactiva, se necesita de una herramienta de software libre, el cual sea capaz de codificar y multiplexar audio y video en formato MPEG-2 y que se basen en el estándar ISDB-Tb; puesto que si se opta por el uso de herramientas licenciadas para la obtención de mejores resultados, su costo es muy elevado y actualmente las empresas de televisión son las únicas que se encuentran en la capacidad de adquirirlos.
- Como punto de partida de este trabajo de titulación, se podría realizar un estudio adicional para generar transmisión de video sobre aplicaciones interactivas en dispositivos móviles, ya que actualmente es un medio tecnológico de gran inclusión en la sociedad y se utilizaría para transmitir información útil a la ciudadanía.
- Para que el usuario pueda manejar correctamente la herramienta OpenCaster, se recomienda descargarlo de la página oficial del Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA), ya que han sido los encargados de realizar modificaciones a esta herramienta al formato ISDB-Tb, puesto que originalmente OpenCaster fue diseñado en base al estándar europeo DVB-T.

REFERENCIAS BIBLIOGRÁFICAS

[1] «Transmisión Analógica vs Digital», http://dvarval.blogspot.com/2011/01/transmision-analogica-vs-digital.html.

[2] R. Jarrín y C. Morejón, «Diseño de una red de frecuencia única para un canal de televisión en la banda UHF con la norma ISDB-Tb para la zona geográfica P», Escuela Politécnica Nacional, Quito, 2012.

[3] «Que es ISDB-T? nuevo estandar de transmision de TV terrestre», http://www.taringa.net/posts/info/3265104/Que-es-ISDB-T-nuevo-estandar-de-transmision-de-TV-terrestre.html.

[4] «INFORMACIÓN SOBRE LA NUEVA TELEVISIÓN DIGITAL», http://televisiondigitalterrestretdt.com/interactividad_tdt.htm.

[5] «Comunidad Ginga Ecuador», http://comunidadgingaec.blogspot.com/2011/06/middleware-ginga.html.

[6] «Ginga: Desarrollo de aplicaciones para TVD», http://www.ramiropol.com.ar/ginga-desarrollo-de-aplicaciones-para-tv-digital/.

[7] NCL Handbook, «About NCL», http://handbook.ncl.org.br/doku.php.

[8] «NCL y LUA», http://www2.elo.utfsm.cl/~elo323/ncl.html.

[9] C. Soares Neto, Gomes Soares Luiz, R. Ferreira Rodrigues y S. Junqueira Barbosa, «Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer», http://www.ncl.org.br/documentos/TutorialNCL3.0-2ed.pdf.

[10] «NCL Eclipse: Ambiente Integrado para o Desenvolvimento de Aplicações para TV Digital Interativa em Nested Context Language», http://www.sohand.icmc.usp.br/~rigolin/downloads/sbrc2009/pdfs/salaodeferramentas1.pdf.

[11] Associação Brasileira de Normas Técnicas, «Canal de interatividade», http://www.abnt.org.br/imagens/Normalizacao_TV_Digital/ABNTNBR15607-1_2008Ed1.pdf.

[12] J. Vila Rosas, «La TDT abre las puertas a la televisión digital educativa (T-Learning)», http://es.scribd.com/doc/6239931/La-TDT-abre-las-puertas-a-la-

television-digital-educativa-TLearning.

[13] Avanet, «Diferencias entre Web Services y WCF», http://avanet.org/diferencias-entre-web-services-y-wcf-windows-communication-foundation.aspx.

[14] Microsoft Developer Network, «Arquitectura de Windows Communication Foundation», https://msdn.microsoft.com/es-es/library/ms733128(v=vs.110).aspx.

[15] Microsoft Developer Network, «Introducción a Windows Communication Foundation», https://msdn.microsoft.com/es-es/library/bb972290.aspx.

[16] F. Becerra , «Diseño e Implementación de Aplicaciones interactivas basadas en Ginga-NCL para Televisión Digital en el área de Eduación Superior», Escuela Politécnica Nacional, Quito, 2014.

[17] Yasunidos, «Yasuní ITT», http://sitio.yasunidos.org/es/yasuni-itt.html.

[18] I. Valencia, «El proyecto Yasuní ITT», http://www.monografias.com/trabajos103/proyecto-yasuni-itt/proyecto-yasuni-itt.shtml.

[19] T. Moncayo y M. Pozo, «Generación del flujo único de paquetes de transporte TS de acuerdo a la norma ISDB-Tb y desarrollo de una aplicación para sus análisis», Escuela Politécnica Nacional, 2014.

[20] «eXtreme Programming (XP)», http://www.slideshare.net/joaquin_win/extreme-programming-456979.

[21] P. Galabay y F. Vivar, «Manejo del software Ginga para el desarrollo de aplicaciones interactivas para televisión digital, basado en el estándar brasileño ISDB-Tb», UPS, Quito, 2012.

[22] Microsoft, «Developer Network», http://msdn.microsoft.com/en-us/library/ee816881(v=vs.110).aspx.

[23] Microsoft, «Características de SQL Server Express», http://msdn.microsoft.com/es-es/library/ms165636%28v=sql.105%29.aspx.

[24] SOURCEFORGE.NET, «Ciclo de vida de un proyecto XP», http://oness.sourceforge.net/proyecto/html/ch05s02.html.

[25] MasterSoft, «El método Scrum»,

http://www.mastersoft.com.ar/MsWeb/otros_archivos/NotaScrumPCUsers.pdf.

[26] J. Cortizo Pérez, D. Expósito Gil y M. Ruiz Leyva, «eXtreme Programming», http://www.josek.net/publicaciones/xp.pdf.

[27] Instituto Español de Tecnologías de la Comunicación, «Ingeniería del Software: Metodologías y Ciclos de Vida», http://www.inteco.es/file/N85W1ZWFHifRgUc_oY8_Xg.

[28] CRISP, «Kanban», http://www.crisp.se/gratis-material-och-guider/kanban.

[29] «La evolución de la Televisión», http://schokolade916.wordpress.com/category/uncategorized/.

[30] Revista ONOFF, «Middleware Ginga y sus Posibilidades Educativas», http://www.onoff.cl/middleware-ginga-y-sus-posibilidades-educativas-en-pantalla/.

[31] M. Santos da Silveira, «TV Digital», http://es.scribd.com/doc/67760149/Comunicacao-de-Dados.

[32] N. Handbook, «The <bind> Element», http://handbook.ncl.org.br/doku.php?id=bind.

[33] «Televisión Digital Terrestre», http://es.wikipedia.org/wiki/Televisi%C3%B3n_digital_terrestre.

[34] Ministerio de Telecomunicaciones y Sociedad de la Información, «Televisión Digital Terrestre en el Ecuador», http://www.telecomunicaciones.gob.ec/television-digital-terrestre-en-el-ecuador/.

[35] VASS Digital, «SCRUM. La metodología de desarrollo ágil por excelencia».

[36] TeleMídia Laboratory, «How to: Create an NCL Composer Plugin», [http://composer.telemidia.puc-rio.br/en/doc/tutorial/how_to_create_a_plugin_to_ncl_composer_0.1.x.

[37] G. Inc., «TeleMidia/nclcomposer», https://github.com/TeleMidia/nclcomposer/blob/master/src/plugins/debug-console/CMakeLists.txt.

ANEXOS

ANEXO A

A.1 Código de la aplicación interactiva enfocada en la protección ambiental

A.2 Código de la aplicación interactiva enfocada en la iniciativa Yasuní ITT

A.3 Código de la aplicación interactiva enfocada en la protección ambiental sincronizado con un video

A.4 Código de la aplicación interactiva enfocada en el reciclaje

A.5 Código del *plug-in* qnclfaqview

(CD Adjunto)

ANEXO B

B.1 Código del servicio WCF

(CD Adjunto)

ANEXO C

C.1 Tablas PSI

C.2 Guía de OpenCaster Versión Lifia DEFENSA

(CD Adjunto)