UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

ESCUELA DE ELECTRÓNICA Y TELECOMUNICACIONES

“IMPLEMENTACIÓN DE UN LABORATORIO DE TELEVISIÓN DIGITAL QUE CUMPLA EL ESTÁNDAR ISDB-TB”

Tesis previa a la obtención del título de

Ingeniero en Electrónica y Telecomunicaciones

AUTORES:

José Luis Medina Cartuche

Cristian Ramiro Villa Arias

DIRECTOR:

Ing. Jorge Mauricio Espinoza Mejía, MSc. PhD

Cuenca – Ecuador

Junio – 2014

RESUMEN

La televisión digital terrestre en nuestro país se encuentra en una primera etapa de implementación, siendo uno de sus principales beneficios la interacción del televidente para colaborar en la mejora e implementación de nuevos servicios.

Este proyecto de tesis implementa un laboratorio de televisión digital que permita desarrollar aplicaciones interactivas, lo cual sirve de plataforma para el objetivo central que es desarrollar una aplicación para la captura de información de los canales y programas que un telespectador visualiza para en el futuro poder predecir el comportamiento del mismo.

El laboratorio de televisión digital posee una infraestructura que soporta el desarrollo de aplicaciones interactivas y la codificación y transmisión de contenido digital televisivo. En adición, se diseñó e implementó una aplicación interactiva que funciona sobre la plataforma Ginga-NCL/LUA, un servidor de aplicaciones interactivas y un servidor de base de datos para el almacenamiento de información generada por los televidentes.

Como resultado de este proyecto, la Universidad de Cuenca cuenta con un laboratorio de televisión digital y una aplicación interactiva que permite capturar la actividad del usuario y almacenarla en una base de datos. En futuros trabajos se podría hacer uso de esta información para recomendar al televidente programas que se ajusten a sus gustos y obtener estadísticas de la edad y sexo del mismo para futuros análisis. Finalmente, con el desarrollo del proyecto se fomenta a los docentes y alumnos de la Facultad de Ingeniería al desarrollo de nuevas aplicaciones e investigaciones en el campo de la Televisión Digital.

PALABRAS CLAVE: Televisión Digital Terrestre, laboratorio de televisión digital, ISDB-Tb, Ginga, NCLua, tablas PSI/SI, aplicaciones interactivas, Guía Electrónica de Programación, servidor de aplicaciones.

ABSTRACT

In our country, the digital terrestrial television is now in a first stage of implementation. One of its main benefits is allow the interaction of the viewer with the television stations in order to collaborate in the improvement and implementation of new services.

This thesis project implements a digital television laboratory that allows the development of interactive applications, which is used as a platform for the central objective that is to develop an application for capturing the information of the channels and programs that a viewer sees in a period of time, in order to use this information in the future to predict the behavior of the viewer.

The digital television laboratory has an infrastructure that supports the development of interactive applications and the codification and transmission of digital television content, which constitutes the first stage of this work. In the second stage, we designed and implemented an interactive application that works in the Ginga-NCL/LUA platform, a server of interactive applications and a database server MySQL for the storage of the information generated for the viewers.

As a result of this project, the University of Cuenca now has a functional digital television laboratory and an interactive application that allows capturing the activity of the user and storing it in a database. In future work, this information could be used to recommend programs that fit the preferences of the viewer. Moreover, you can obtain statistics of the age and gender of a viewer that sees a specific program for future analysis. Lastly, the development of this project motivates the faculty and the students of the Department of Engineering to develop new applications and to do research in the field of digital television.

Keywords: digital terrestrial television, digital television laboratory, ISDB-Tb, Ginga, NCLua, PSI/SI tables, interactive applications, programming electronic guide, applications server.

ÍNDICE

ÍNDICE DE FIGURAS

Yo, José Luis Medina Cartuche, autor de la tesis “IMPLEMENTACIÓN DE UN LABORATORIO DE TELEVISIÓN DIGITAL QUE CUMPLA EL ESTÁNDAR ISDB-TB”, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de Ingeniero en Electrónica y Telecomunicaciones. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

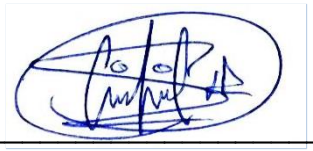
Cuenca, 17 de junio del 2014

José Luis Medina Cartuche

C.I: 1105045361

Yo, Cristian Ramiro Villa Arias, autor de la tesis “IMPLEMENTACIÓN DE UN LABORATORIO DE TELEVISIÓN DIGITAL QUE CUMPLA EL ESTÁNDAR ISDB-TB”, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de Ingeniero en Electrónica y Telecomunicaciones. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, 17 de junio del 2014



Cristian Ramiro Villa Arias

C.I: 0104535935

Yo, José Luis Medina Cartuche, autor de la tesis “IMPLEMENTACIÓN DE UN LABORATORIO DE TELEVISIÓN DIGITAL QUE CUMPLA EL ESTÁNDAR ISDB-TB”, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 17 de Junio del 2014

José Luis Medina Cartuche

C.I: 1105045361

Yo, Cristian Ramiro Villa Arias, autor de la tesis “IMPLEMENTACIÓN DE UN LABORATORIO DE TELEVISIÓN DIGITAL QUE CUMPLA EL ESTÁNDAR ISDB-TB”, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 17 de Junio del 2014

Cristian Ramiro Villa Arias

C.I: 0104535935

DEDICATORIA

El presente Trabajo de tesis lo dedico a mi madre María Cartuche, ella con su cariño y buen ejemplo se ha convertido en eje fundamental en mi vida, a mis hermanos que en todo momento me brindan su apoyo incondicional, a mi sobrino Alejandro que es un ejemplo de superación para mi diario vivir.

José Luis

DEDICATORIA

El presente trabajo va dedicado a Dios quien me otorgó el mejor de los regalos, mis padres Ramiro y Graciela, quienes con su apoyo incondicional, dedicación, esfuerzo, confianza y amor me han sabido guiar por el buen camino en cada paso de mi vida.

A mi hermano Iván, por su apoyo incondicional.

A mi hermana Katherine pues con su ejemplo de perseverancia, consejos y esfuerzo me ha encamino siempre.

Cristian

AGRADECIMIENTO

Agradezco a mis padres cuyo esfuerzo me ha permitido llegar hasta estas instancias en mi vida.

A mis Hermanos por el apoyo emocional, en especial a María Luisa por ser mi modelo a seguir durante mis estudios.

Al Ing. Jorge Mauricio Espinoza por su paciencia y tiempo invertido como Director del proyecto de investigación.

A la Ing. Sofía Arévalo Maldonado por haber compartido sus conocimientos y experiencias en temas relacionados a esta tesis.

A mis amigos y compañeros por haberme dado su apoyo durante mi trayectoria académica.

José Luis

AGRADECIMIENTO

A Dios por brindarme salud y sabiduría para enfrentar las dificultades y retos propuestos.

A mi familia por todo el apoyo incondicional brindado.

A nuestro Director Ing. Mauricio Espinoza Mejía, por apoyarnos en la realización de la misma y brindarnos su confianza.

A la Ing. Sofía Arévalo Maldonado, docente de la facultad de Ingeniería, por haber compartido sus conocimientos y experiencias en temas relacionados a esta tesis.

A un excelente amigo Christian Arias quien compartió conocimientos que fueron de gran ayuda en temas desarrollados en la tesis, mis compañeros y amigos, por haberme brindado su amistad, sinceridad y por el tiempo compartido en el trascurso de mis estudios universitarios.

Cristian

INTRODUCCIÓN

El capítulo introductorio aborda los temas relacionados con los objetivos, alcance y justificación de este proyecto de tesis. Se presenta la problemática que se pretende solucionar en el contexto del proyecto de investigación "Aplicación de Tecnologías Semánticas para disminuir la sobrecarga de información en usuarios de TV Digital". Además, se explican los problemas adicionales que se suscitan al no contar con un laboratorio de televisión digital dentro de la Universidad de Cuenca, justificándose así su implementación y haciendo énfasis en la importancia de este componente dentro del proyecto de investigación junto con las ventajas que brindará al explotar la interactividad con el televidente. Seguido se muestra el alcance del proyecto, en donde se establece las dos etapas que tendrá el proyecto, siendo estas la implementación del laboratorio y el desarrollo de aplicaciones con un servidor de interactividad. El capítulo también expone los objetivos a cumplir dentro del proyecto. Finalmente se describe la contribución de los resultados al término de la tesis, enfocándose primero a los objetivos del proyecto y luego al uso posterior del laboratorio.

CAPITULO 1

INTRODUCCIÓN

1.1. Identificación del problema

En el año 2010 el Ecuador adopta la tendencia mundial de cambio hacia la tecnología digital para la transmisión de Televisión Terrestre, optando por el estándar Japonés-Brasileño (ISDB-Tb[1]). Una de las características más representativas de la introducción de la televisión digital es la difusión de un mayor número de canales que la televisión analógica tradicional, posibilitando además la transmisión y recepción de contenido multimedia e interactivo mediante un canal de retorno. Este hecho, si bien es claramente una de sus mayores ventajas, trae consigo la necesidad de herramientas especializadas que permitan cambiar toda la cadena de producción, principalmente en el consumo del contenido final.

En este contexto, algunos investigadores del Departamento de Ciencias de la Computación y del Departamento de Electrónica y Telecomunicaciones, se encuentran desarrollando un proyecto aprobado y financiado por la Dirección de Investigación (DIUC) denominado: “Aplicación de Tecnologías Semánticas para Disminuir la Sobrecarga de Información en Usuarios de TV digital”, el cual pretende diseñar un sistema de recomendación para la programación televisiva que tome en consideración las preferencias del usuario.

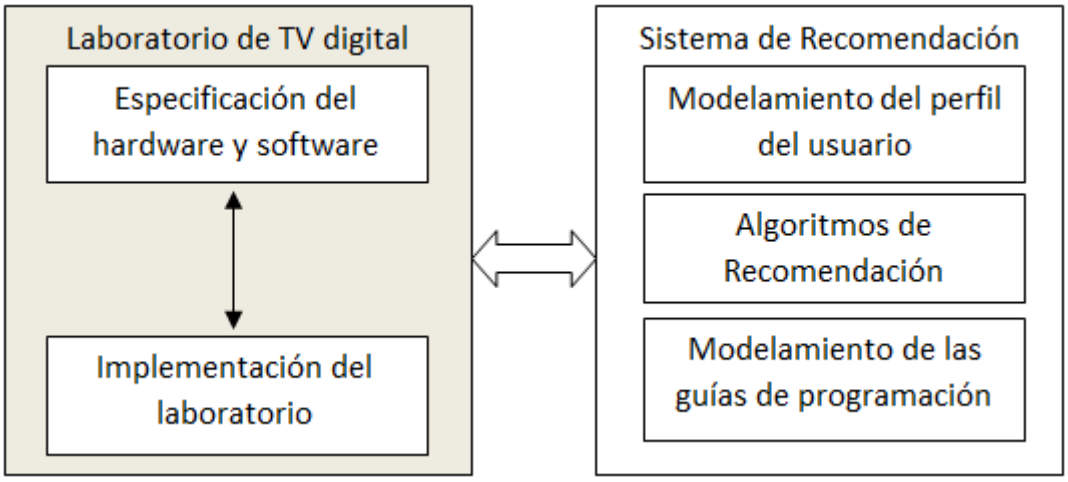


Figura 1: Esquema general del proyecto “Aplicación de Tecnologías Semánticas para Disminuir la Sobrecarga de Información en Usuarios de TV digital” [1].

[1] ISDB-T es un componente del conjunto de normas creado por Japón para las transmisiones de radio y televisión digital. Este componente en particular se refiere a la televisión digital terrestre. ISDB-Tb por otra parte es el estándar técnico brasileño para la transmisión de televisión digital terrestre basado en el estándar japonés ISDB-T.

La etapa denominada “Laboratorio de TV digital”, como se observa en la Figura 1, constituye la primera fase del proyecto. Esta etapa se encargará de buscar las alternativas óptimas para simular un escenario real de transmisión y recepción de señal televisiva. Este diseño estará formado por elementos de hardware y software que posibiliten transmitir y recibir múltiples contenidos televisivos. Por otra parte, la etapa denominada “Sistema de Recomendación”, se encargará de diseñar una infraestructura semántica por medio de la utilización de ontologías[2] para captar las preferencias de los televidentes y los contenidos de los programas. Esta etapa usará además algoritmos de recomendación que permitan seleccionar únicamente aquellos programas que son de interés para el televidente en base a sus preferencias, con el objetivo de no sobrecargar de información.

Respecto a la primera fase del proyecto, es necesario indicar que el trabajo desarrollado en [2] identificó a nivel conceptual los requisitos de hardware y software necesarios para la creación del Laboratorio de TV digital. Sin embargo, en dicho trabajo no se abordan los aspectos técnicos de implementación y puesta en marcha de este laboratorio; trabajo que será abordado en este proyecto de tesis (ver Implementación del Laboratorio en la Figura 1).

Es necesaria la implementación del laboratorio debido a que en la ciudad aún no se puede encontrar canales de televisión que estén transmitiendo sus señales en formato digital. De todas maneras si tuviéramos al alcance canales de televisión transmitiendo en formato digital, sería difícil poder contar con los permisos necesarios para probar las ventajas de esta tecnología. En este sentido podemos hablar de una carencia de un entorno apropiado para probar todas las ventajas que ofrece la TV digital, esto considerando que el funcionamiento del sistema descrito previamente se basa en la generación y transmisión de señales en formato digital.

Para la correcta implementación del laboratorio de TV digital, se necesita de una infraestructura para generar contenidos televisivos, probar su correcto funcionamiento y ejecutar aplicaciones de interacción. La puesta en marcha de este ciclo de vida requiere un equipo de alto costo, sin embargo el objetivo planteado aquí es la instalación y adecuación de equipos, instalación de software y diseño de la arquitectura del laboratorio, que permita simular estas mismas tareas pero haciendo uso de equipos y software de bajo costo. Para el proyecto de tesis, se plantea, el rediseño e implementación de la arquitectura del laboratorio planteado en [2], configuración de herramientas y creación de manuales, para que los docentes y alumnos puedan dar uso y desarrollo a nuevos aplicaciones e investigaciones en el campo de la Televisión Digital.

[2] En esta tesis se usa el término ontología para referirse a la estructura de clasificación y las instancias dentro de una base de conocimientos.

La configuración, instalación y puesta en marcha de los equipos que serán parte del laboratorio de TV digital es solo una parte del problema, Adicionalmente se deberá buscar alternativas para simular y capturar las diferentes actividades que un televidente genera. Para adquirir y adecuar esta información se deberá generar una aplicación interactiva, desarrollada dentro de la plataforma Ginga[3]. Esta aplicación debe permitir hacer uso de una de las muchas ventajas de la TV Digital, que es la interacción que el televidente podrá tener con su televisor o STB[4] a través del control remoto. Esta información debe ser procesada y enviada a través del canal de retorno hacia un extremo de la red, donde se necesita de una plataforma de servicios o servidores que manejen la información capturada y que además transcriban la información para poderla usar en el resto Sistema Recomendador[5]. Pero dicha información no dará mayor beneficio si el resultado de su procesamiento no puede ser retornado al televidente, por lo que, los servidores y aplicaciones deben estar en la capacidad de devolver esta información y mostrarla al televidente que es el objetivo principal del proyecto global.

1.2. Justificación

Como ya se mencionó en la sección previa, uno de los objetivos de este proyecto de investigación es recomendar posibles programas de televisión de entre los múltiples contenidos generados por los Broadcasters[6]. Para cumplir con esta finalidad la arquitectura del sistema se dividió en los siguientes componentes. ”1) Extracción del Perfil del Usuario, 2) Anotación Semántica de las Guías de Programación Electrónica, 3) Enriquecimiento de las Guías de Programación con Recursos Externos y 4) Enriquecimiento del Modelo Semántico” [1].

Este proyecto de tesis se enfocará en dar solución a las necesidades del componente 1 (“Extracción del Perfil del Usuario”). Como se describe en [1], para la captura del perfil, el sistema usará un proceso híbrido. De esta manera, se combinarán métodos de captura implícitos basados en el seguimiento del comportamiento del usuario a través de la interacción con el televisor y la captura de las anotaciones efectuadas en redes sociales y métodos de captura explícitos a través de una aplicación Web. Es parte del desarrollo de esta tesis la recolección de datos del comportamiento del usuario, la guía electrónica de programación (EPG), y la interacción del Usuario/Televisión como se ve en la parte resaltada de la Figura 2. El sistema que permitirá la captura de datos de la interacción del usuario con la televisión se basará en el uso de la información

[3] Ginga hace referencia al middleware usado por el decodificador en el estándar ISDB-Tb
[4] STB: Set Top Box decodificador de señal digital de televisión.
[5] En esta tesis se usa la designación “Sistema Recomendador” para referirse al proyecto “Aplicación de Tecnologías Semánticas para Disminuir la Sobrecarga de Información en Usuarios de TV digital”.
[6] Broadcaster: radiodifusora de televisión

enviada a través del espectro por los Broadcasters en base a la actividad de los televidentes con el STB o el Televisor.

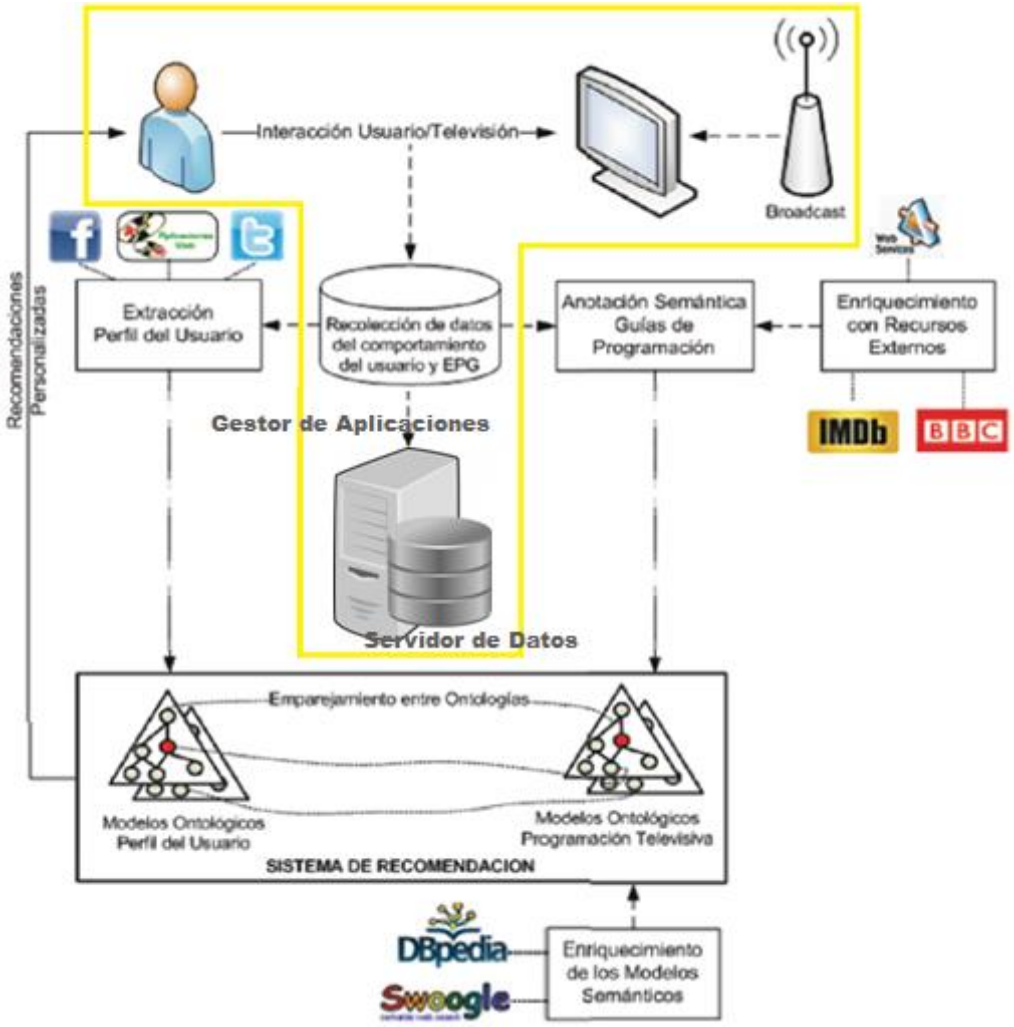


Figura 2: Arquitectura del Sistema [1]

La primera tarea a realizar será simular el funcionamiento de los Broadcasters, esto debido a que, como se explicó anteriormente, una de las problemáticas principales es no contar con canales de televisión transmitiendo contenido en formato digital. Esta simulación será posible una vez terminada la instalación del Laboratorio de TV Digital. El diseño de infraestructura del laboratorio va a estar conformado por los equipos propuestos en [2] . Es necesario hacer notar que el pre-diseño propuesto al no tener un escenario real, podría presentar problemas a la hora de la verificación del funcionamiento de los equipos, instalación de drivers, acoplamiento y sincronización de los equipos para obtener un correcto funcionamiento. Por ello, a medida que se avance en el trabajo de implementación de la infraestructura del laboratorio se deberá realizar cambios al esquema y dar soluciones a eventuales problemas que se puedan presentar.

Otra de las tareas a resolver es la generación de contenido en el estándar ISDB-Tb. Video, audio y datos deben ser generados, codificados, encapsulados y empaquetados, con el objetivo de la generación de un Transport Stream final. Para los fines del proyecto se debe también generar un servidor o servicio que provea al usuario de una EPG, y que permite hacer la captura de los datos para su uso posterior. Estos contenidos también deben ser enviados a través del espectro radioeléctrico, por lo que el laboratorio debe estar en capacidad de soportar estas características.

El siguiente paso sería definir los mecanismos y herramientas necesarias para capturar información del contenido digital radiado. Esta información capturada deberá ser enviada adecuadamente a un sistema informático que tenga la capacidad de almacenar (ver recolección de datos del comportamiento del usuario y EPG en la Figura 2), procesar y después mostrar adecuadamente esta información al televidente, ya que luego de tener la información útil generada por el sistema, se procederá con la visualización del contenido en la TV (contenido de interés para el Usuario). La visualización del contenido será una simulación de la puesta en marcha del sistema recomendador. Todas estas actividades deben ser manejadas por una aplicación interactiva, la misma debe ser desarrollada en Ginga-NCL[7] que es el formato permitido para el estándar ISDB-Tb. Esta aplicación deberá mejorar su funcionamiento al introducir scripts de programación en Lua, robusteciendo el funcionamiento del sistema en distintos escenarios. Mientras que el servidor de datos deberá aportar robustez en la comunicación de la información y el acceso a los recursos almacenados en el sistema.

Cabe recalcar que para la generación de los contenidos digitales (codificación y sincronización de video, audio y datos) se van a crear herramientas amigables al usuario que les facilite crear estos contenidos. Para la instalación de software necesario para programación de las aplicaciones interactivas, también se realizará manuales donde se detallará como realizar cada uno de estos procesos. Estas actividades beneficiarán a posteriores usuarios del Laboratorio, que podrán usar como base el desarrollo obtenido hasta el momento y tratar de resolver con agilidad las necesidades que estos tengan en otros aspectos. Física realizara detallara

1.3. Alcance

El presente proyecto de tesis tiene como finalidad completar dos etapas. La primera etapa consiste en el diseño de la infraestructura física del laboratorio, que permitirá generar, transmitir y recibir una señal de Televisión Digital Terrestre. Esta parte contempla el uso de equipos para la multiprogramación, así como la generación y transmisión de contenidos con EPG y aplicaciones interactivas. Para esto se elaborará un manual que optimice la generación de

[7] Ginga-NCL hace referencia al subsistema lógico necesario para programar apps de TV digital

un flujo de trasporte TS (Transport Stream[8]), con lo cual se minimizará errores de codificación y sincronización del contenido (audio, video y datos). En el manual se detallarán además los mecanismos para la generación y transmisión de aplicaciones interactivas mediante el TS. Además se va a generar herramientas con interfaces amigables para el usuario, éstas tendrán la capacidad de automatizar estos procesos, con la posibilidad que tengan un uso futuro por los docentes y estudiantes de la Facultad.

La segunda etapa corresponde a la implementación, la cual pretende diseñar una aplicación interactiva, así como también implementar un gestor de aplicaciones interactivas y un servidor con una base de datos para el almacenamiento de información (información de la programación vista por el usuario). Dentro de las características planteadas para la aplicación interactiva, se encuentran las funciones de: la App de logueo, App de consulta, envío y almacenamiento de Información, App de captura de información de la actividad del Usuario al mirar televisión, App de proyección de recomendación en la TV. Para que este conjunto de aplicaciones tengan un ciclo de vida en el sistema, es necesario que interactúen con el gestor de aplicaciones interactivas. De igual manera la información enviada desde las aplicaciones ejecutadas en el STB de los Usuarios hacia el servidor debe ser almacenada en una base de datos. La fase de comunicación entre las aplicación interactiva con el gestor de aplicaciones será un modelo de la tecnología TCP/IP.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar la infraestructura e Implementar un laboratorio de Televisión Digital Terrestre que permita la generación, transmisión y recepción de una señal de Televisión Digital Terrestre cumpliendo el estándar ISDB-Tb.

1.4.2. Objetivos específicos

1. Realizar pruebas de generación del flujo de trasporte TS (Transport Stream) que permitan comprobar la disminución de errores de codificación y sincronización de audio, video y datos (información y aplicaciones interactivas) y elaborar un manual en el cual se detalle la generación del TS, la codificación y sincronización del contenido.
2. Probar que la plataforma del Laboratorio permita transmitir y recibir una señal de televisión digital, ya sea generando servicios de multiprogramación, Guía Electrónica de Programación (EPG), y contenidos de aplicaciones interactivas.
3. Diseñar una aplicación que permita automatizar la generación del TS y tenga una interfaz amigable con el usuario.
4. Crear un gestor de aplicaciones y un servidor de datos, cuya tarea principal es el almacenamiento de la información intercambiada entre los

[8] Transport Stream: flujo de transporte de contenidos audio, video y datos (TS)

receptores de Televisión (STB) y el servidor, mediante modelos de la tecnología TCP IP y el uso de Base de Datos.

5. Generar un conjunto de aplicativos para la captura de datos transmitidos a través del Transport Stream e interacción con el usuario. Estas aplicaciones interactivas tendrán como tarea controlar el acceso de los usuarios (iniciar y finalizar sesión), consulta y envío de información al servidor de datos, captura de información de la actividad del usuario y visualización de la recomendación.
6. Diseñar un esquema para integrar el sistema de captura, almacenamiento y visualización de información con el resto de componentes del proyecto "Aplicación de Tecnologías Semánticas para Disminuir la Sobrecarga de Información en Usuarios de TV Digital".
7. Recomendar soluciones alternativas de los elementos que conforman el laboratorio, para mejorar ciertas áreas y componentes del mismo, en base a parámetros técnicos y a conclusiones obtenidas al término del proyecto de tesis.

1.5. Contribución de resultados

Una vez concluido el diseño, la implementación del laboratorio, el desarrollo de la aplicación de captura de actividad del televidente y el desarrollo del servidor de aplicaciones interactivas, se exponen las siguientes contribuciones del proyecto.

1.5.1. Diseño e implementación del laboratorio

El resultado de esta actividad es una plataforma para la generación de contenido, pruebas de transmisión y recepción, y evaluación del estándar ISDB-Tb. En cuanto a la generación de contenido, la plataforma permite codificar, empaquetar y generar flujos de transporte de información para contenido audiovisual, guías electrónica de programación y aplicaciones interactivas.

El laboratorio tiene equipos que permiten transmitir y recibir servicios de televisión digital. En donde los diferentes parámetros tales como modulación, tasas de transmisión, potencia, frecuencia y ancho de banda pueden ser monitoreados y modificados en la etapa de transmisión. Además la plataforma permite al receptor hacer uso del canal de retorno y brindar beneficios adicionales a algunas aplicaciones interactivas. Finalmente permite evaluar las características del estándar ISDB-Tb, permitiendo así al usuario la posibilidad de probar la calidad de los contenidos audiovisuales, la inserción de múltiples servicios por un mismo canal de transmisión y la operatividad de los receptores en distintos escenarios.

1.5.2. Desarrollo de aplicación de captura de actividad

Una consecuencia de esta actividad es la posibilidad de desarrollar aplicaciones Ginga- NCL y scripts de Lua. Dentro del laboratorio se adquirió e

instalo software que permitió crear aplicaciones y emular sus funciones. Se desarrolló un conjunto de aplicativos que permiten capturar la actividad del televidente, esto contribuyo al desarrollo del sistema recomendador. Estas aplicaciones dejan abierta la posibilidad de que los Broadcaster las usen para otros fines adicionales como registro de usuarios en un sistema, verificación de datos de usuario, pruebas de rating y pruebas del canal de retorno. Adicional a esto se establece un procedimiento de cómo los Broadcaster pueden brindar servicios de interactividad y evaluar su desempeño por parte de los televidentes.

1.5.3. Desarrollo de Servidor de Aplicaciones

Al concluir esta parte se dotó al laboratorio de un sistema que controle los requerimientos de las aplicaciones interactivas. Primero se instaló una plataforma que mediante el canal de retorno pueda comunicarse con los receptores del televidente. Adicionalmente, se programó un servidor que le permite a las aplicaciones almacenar la información generada por el usuario y dirigir datos a un usuario en particular. Esta plataforma también permite al laboratorio conectarse con otra infraestructura o un sistema adicional. En caso particular el servidor se conecta mediante un esquema de base de datos a la información generada por el sistema semántico que recomienda ciertos programas de acuerdo al perfil del televidente.

RESUMEN DE CAPÍTULO

El capítulo introductorio aborda los temas relacionados con la presentación del proyecto de tesis en los temas como: planteamiento de la problemática, la justificación y los objetivos del proyecto de tesis. Adicional a esto, el capítulo expone el alcance que se tendrá al culminar con el proyecto. Finalmente, muestra la contribución que deja el proyecto una vez culminados los objetivos planteados.

2

ARQUITECTURA DEL TRANSMISOR Y RECEPTOR DE TDT EN EL ESTÁNDAR ISDB-TB, EN LA IMPLEMENTACIÓN DEL LABORATORIO Y TRANSMISIÓN DE SERVICIOS DE TV DIGITAL

El capítulo dos explica el diseño e implementación del laboratorio de TV Digital, y se complementa con datos adicionales relacionados a la transmisión de un servicio de TV Digital. Primero se plantean las funciones que debe cumplir el laboratorio para posterior a ello describir una arquitectura eficiente que cumpla con las necesidades del proyecto. Luego de esto se describen las herramientas y los equipos con los que se cuenta para el diseño del laboratorio, con estos equipos se generó una arquitectura que haga su uso eficiente. Posteriormente se describe a cada componente del laboratorio, según su implementación actual, se muestra como están conectados los equipos y su función en el laboratorio. Luego se describen otros aspectos que logran generar un servicio de televisión digital como tablas de información, la codificación, multiplexación y transmisión de los contenidos. Se detalla el uso de cada uno de estos componentes y el procedimiento para la generación del Transport Stream final que contendrá todos los servicios a transmitir.

CAPÍTULO 2

ARQUITECTURA DEL TRANSMISOR Y RECEPTOR DE TDT EN EL ESTÁNDAR ISDB-TB, EN LA IMPLEMENTACIÓN DEL LABORATORIO

2.1. Arquitectura real de un Sistema de TV Digital Terrestre

Cuando nos referimos a la arquitectura real para un sistema de Televisión Digital Terrestre (TDT), hablamos de un modelo creado para cubrir un sinnúmero de funciones y actividades para el laboratorio. Este modelo ideal no será implementado por completo, ya que el presente proyecto solo enfoca sus necesidades hacia la gestión de la actividad que genera un televidente. Para saber que modificaciones hacer a este modelo se debe establecer las funciones y servicios que brindará el laboratorio, estos deben estar enfocados en los objetivos del proyecto de tesis. Una vez definidas las funciones se podrá escoger un modelo que cubra con estos requerimientos y finalmente acoplar los módulos de esta arquitectura enfocándolos hacia las funciones y servicios que cubren durante su implementación.

2.1.1. Funciones y servicios del laboratorio de TV digital

Antes de realizar el análisis del diseño de la arquitectura para un laboratorio de TDT es necesario definir las funciones que va a desempeñar el mismo. Estas funciones van destinadas a solventar las necesidades del sistema recomendador descrito brevemente en el Capítulo 1. Estas funciones no son generales para todos los laboratorios que se implementen con fines investigativos. Teniendo en cuenta esto se plantean las siguientes funciones o servicios por parte del laboratorio:

- Codificación de contenidos audiovisuales y datos.
- Multiplexación de contenido audiovisual y datos.
- Ambiente de desarrollo Ginga.
- Multiplexación de aplicativos Ginga con contenido de audio y video.
- Sincronización y transmisión de contenidos audiovisuales, datos y aplicativos Ginga en el estándar ISDB-Tb.
- Amplificación de las señales de radiofrecuencia para obtener una área de cobertura adecuada.
- Recepción de contenidos audiovisuales, datos y aplicativos Ginga en el estándar ISDB-Tb.
- Herramientas de emulación de Set Top Box.
- Acceso a red LAN o WAN mediante canal de retorno.
- Gestor de recursos para aplicaciones interactivas y almacenamiento de información.
- Acceso a bases de datos y servicios interactivos externos.

Una vez definidas las funciones y servicios que tendrá el laboratorio, en la siguiente sección se procede a escoger una arquitectura que se adecue a estas necesidades.

2.1.2. Módulos de la arquitectura

Una solución de arquitectura para un laboratorio ISDB-Tb que cubre las necesidades descritas en el planteamiento del problema, es descrita por el proveedor brasileño EiTV [3]. La misma es una plataforma completa para el uso de servicios interactivos, la generación, codificación, transmisión y recepción de contenidos televisivos; esenciales para cumplir el objetivo principal del proyecto. Esta arquitectura se encuentra representada en la Figura 3, en la misma se distinguen 4 módulos destinados a una función específica del laboratorio. Estos se detallan a continuación.

Estación de generación de señales (Broadcaster): Desempeña las funciones generales del Broadcaster, incluye al equipo de playout, equipamiento responsable de la generación del carrusel de datos para aplicativos Ginga. Realiza funciones de servidor de PSI/SI, EPG, multiplexor, remultiplexor y modulador ISDB-T, provee encoders de video y un transmisor ISDB-T junto a una antena.

El Broadcaster es la parte crítica del laboratorio ya que sin un correcto funcionamiento de este los demás módulos no podrán ser ejecutados.

Ambiente de desarrollo de aplicaciones: Provee el equipo necesario para la generación de aplicativos Ginga, además herramientas de emulación para las aplicaciones generadas. Debe estar adecuada para el envío de aplicaciones hacia el módulo de generación y tener acceso a bases de datos externas.

Proveedor de servicios interactivos (Return Chanel Server): Aquí están presentes las herramientas y equipos para la gestión de las aplicaciones interactivas ejecutadas en el receptor. Estos equipos proveen la comunicación entre el STB, Broadcaster, bases de datos y servicios externos, para ello gestionan los recursos de hardware y software hacia los televidentes.

Entorno del usuario o receptor: Se establece el decodificador de señales en estándar ISDB-Tb, antena y el televisor. Es el medio por el cual el usuario interactúa con el resto de sistemas y debería proveer una correcta comunicación mediante el canal de retorno.

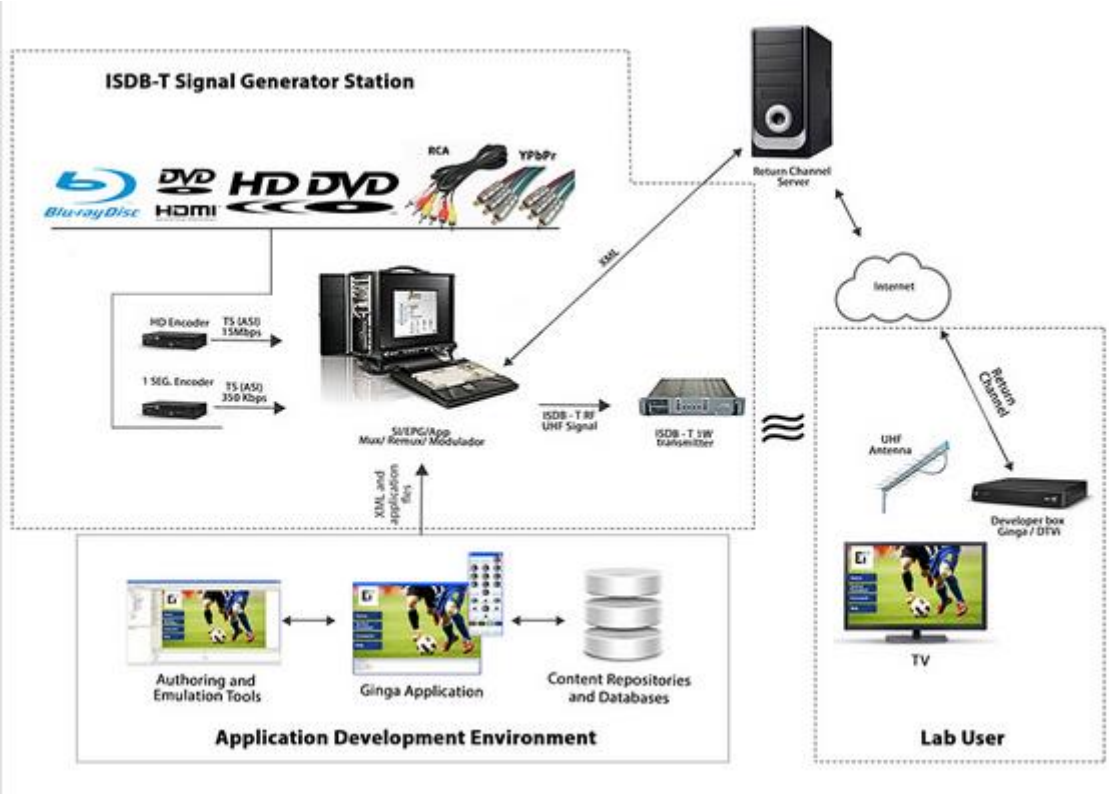


Figura 3: Arquitectura de laboratorio ISDB-Tb [3]

Teniendo claros estos módulos dentro de la arquitectura, procedemos a mostrar cómo se acoplan a las necesidades que debe cumplir el laboratorio.

2.1.3. Acoplamiento entre la arquitectura y funciones

La solución descrita anteriormente debe ser acoplada a las funciones descritas en la sección 2.1.1. Se debe tener claro la función de cada módulo para posteriormente saber qué partes del esquema son indispensables y qué partes se pueden omitir dentro de la implementación del laboratorio. Para ello vamos a describir nuevamente a cada módulo orientado a la función que desempeña su equipamiento.

Estación de generación de señales

Las funciones a desarrollar en este módulo son las que tienen que ver con el tratamiento de los contenidos televisivos hasta llegar a su transmisión. Estas funciones son:

- Codificación de contenidos audiovisuales y datos.
- Multiplexación de contenido audiovisual y datos.
- Multiplexación de aplicativos Ginga con contenido de audio, video y datos.
- Sincronización y transmisión de contenidos audiovisuales, datos y aplicativos Ginga en el estándar ISDB-Tb.

- Amplificación de las señales de radiofrecuencia para obtener una área de cobertura adecuada.

Ambiente de desarrollo de aplicaciones

Este módulo realiza las funciones que tienen que ver con el diseño, desarrollo y prueba de aplicaciones en la plataforma Ginga. Las funciones se detallan a continuación:

- Ambiente de desarrollo Ginga.
- Herramientas de emulación de Set Top Box.
- Acceso a red LAN o WAN mediante canal de retorno.
- Gestor de recursos para aplicaciones interactivas y almacenamiento de información.
- Acceso a bases de datos y servicios interactivos externos.

Proveedor de servicios interactivos

Las funciones del proveedor de servicios interactivos se enfocan en brindar el acceso a información externa, su almacenamiento y la gestión de recursos por parte de las aplicaciones. Las funciones a cumplir son:

- Acceso a redes LAN o WAN mediante el canal de retorno.
- Gestor de recursos para aplicaciones interactivas y almacenamiento de información.
- Acceso a bases de datos y servicios interactivos externos.

Entorno del usuario o receptor

Este módulo provee las funciones que permiten simular un entorno del televidente, las mismas dan acceso al usuario a los contenidos transmitidos por el Broadcaster y a la interactividad a través de las aplicaciones. Las funciones que permiten dar estos servicios son:

- Recepción de contenidos audiovisuales, datos y aplicativos Ginga en el estándar ISDB-Tb.
- Acceso a redes LAN o WAN mediante el canal de retorno.
- Gestor de recursos para aplicaciones interactivas y almacenamiento de información.
- Acceso a bases de datos y servicios interactivos externos.

Con el modelo y las funciones planteados en esta sección se puede tener una idea clara para el diseño de una arquitectura que cumpla estas características. Este diseño se muestra en la siguiente sección.

2.2. Arquitectura a implementar

El diseño de arquitectura debe permitir acoplar los equipos ya adquiridos para el laboratorio sin dejar de cubrir todas las necesidades del proyecto. El acoplamiento da una idea clara sobre las limitaciones que tendrá el laboratorio una vez sea implementado.

2.2.1. Equipos adquiridos previo al diseño de arquitectura

A continuación se detalla la lista de equipos adquiridos para el proyecto y que serán acoplados a la arquitectura del laboratorio siguiendo las recomendaciones del pre-diseño planteado en [2]:

- Dos tarjetas de modulación VHF/UHF USB-2 DTU-215.
- Dos amplificadores de señal de TV 36 Db.
- Una computadora Intel Pentium IV 3 GHz, 1GB de Ram y disco duro de 160GB.
- Una computadora Intel Core 2 1.86 GHz, 1GB de Ram y disco duro de 250GB.
- Una computadora Intel Core 2 2.1 GHz, 2GB de Ram y disco duro de 300GB.
- Una computadora Core I5 3.1GHz, 2GB de Ram y disco duro de 500GB.
- Una computadora Core I7 4 GHz, 4GB de Ram y disco duro de 750GB.
- Un Televisor Analógico LG 21 pulgadas.
- Un Televisor LCD Bravia de Sony 32“ con receptor ISDBT integrado.
- Un Set Top Box marca EITV developer box ISDB-Tb.

Las características completas de estas herramientas se encuentran detalladas en el Anexo A.1. Sin embargo se detalla a continuación las más importantes y junto con los componentes de software adicionales.

USB-2 VHF/UHF Modulator DTU-215: Esta tarjeta moduladora es utilizada para el estándar ISDB-Tb (ver Figura 4). Permite la manipulación de ciertas características en radiación y posee un amplificador de baja potencia controlado por medio de software.

Figura 4: DekTec DTU-215

EITV developer box ISDB-Tb: Este decodificador está destinado para desarrolladores, permite recibir y decodificar IPTV y señales de TDT en el estándar ISDB-Tb. También permite ejecutar aplicaciones GINGA con librerías Lua. Posee un interfaz para conexión a internet vía LAN con una aplicación para navegación (ver Figura 5).

Figura 5: EITV developer box

Software OpenCaster: Es un software libre de código abierto destinado a la generación de TS en MPEG2 (Estándar Europeo). Permite codificar, empaquetar, multiplexar y sincronizar contenido multimedia. Permite la codificación de los datos de aplicaciones interactivas y tablas PSI/SI y se acopla al sistema ISDB-Tb mediante un parche desarrollado por los laboratorios LIFIA. Se ejecuta sobre el sistema operativo Linux.

Teniendo en cuenta estas herramientas a continuación se procede a diseñar la arquitectura que utilice estos equipos y permita cubrir las funciones requeridas por el proyecto.

2.2.2. Diseño de la arquitectura

El diseño de arquitectura se realizó acoplando las funciones vistas en la sección 2.1 con el software y hardware recomendado para el laboratorio en [2]. Adicionalmente a este diseño se le agregará los siguientes requerimientos destinados a cubrir las necesidades del proyecto de investigación global.

- Se requiere al menos dos Broadcasters para la simulación de actividades por parte del usuario.

- Un servidor destinado a la conexión con el resto del sistema recomendador.
- Al menos dos tipos de receptores que permitan emular distintos escenarios por parte del usuario.

Tomando en consideración todos estos requerimientos se diseña la arquitectura del laboratorio mostrada en la Figura 6. Esta arquitectura mantiene el esquema propuesto en la sección 2.1, pero acoplado a las necesidades del proyecto. Aquí podemos ver que dentro del ambiente de desarrollo de aplicaciones es necesario contar con una máquina que tenga instalado el software que permita el desarrollo de las aplicaciones y su posterior almacenamiento. También se distinguen claramente los dos Broadcaster que le permitirán al usuario emular un cambio de estaciones de televisión, generando diferentes actividades.

Se puede apreciar un ligero cambio dentro del proveedor de servicios interactivos. Este solo tendrá conexión hacia los receptores debido a que no es necesario ningún tipo de comunicación con los Broadcaster, esto no impide que el usuario pueda interactuar con sistemas independientes al canal de televisión que transmite las señales. Para los receptores se deja abierta la posibilidad de la instalación de más equipos. Esto nos permitirá visualizar distintos tipos de comportamiento y calidad de servicio por parte del televidente; por lo que se recomienda que estos receptores tengan distintas características entre sí.

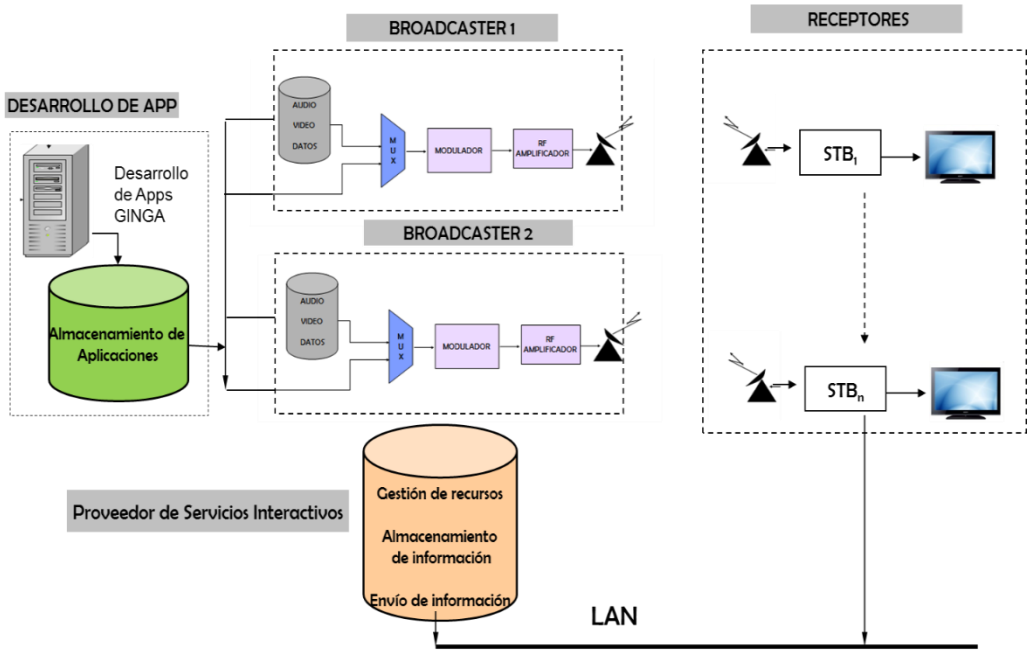


Figura 6: Diseño de arquitectura del laboratorio

Esta arquitectura y las herramientas descritas anteriormente presentan ciertas limitaciones a la hora de implementar el laboratorio. Una de las principales limitaciones se da debido a que las herramientas y equipos se ejecutan en

distintas plataformas de sistema operativo y no se puede tener un acoplamiento exacto de los mismos.

El problema de acoplamiento queda evidenciado en la parte del Broadcaster donde no se puede tener una transmisión en tiempo real. Como se observa en la Figura 7, existe un desfase de tiempo entre la codificación del video, audio y datos, su multiplexación y modulación, debido a que se utiliza el Software OpenCaster[9] para la codificación y multiplexación del contenido. Debido a que este software fue diseñado para el SO Linux, no se puede acoplar de manera inmediata al modulador (Tarjeta DTU-215) ya que ésta tiene una interfaz de trasmisión desarrollada en Windows llamada StreamXpress. Teniendo en cuenta este inconveniente la transmisión no se realizará en tiempo real por lo tanto en este caso no tiene funcionalidad un servidor destinado a la generación del EPG o a la adquisición de aplicaciones, ya que a las mismas no se las puede codificar y actualizar a medida que se transmite la señal de TDT. Es decir el video el audio, datos, EPG y aplicaciones ya están codificados en un solo Transport Stream previo a su transmisión y no pueden ser modificados una vez esta inicie.

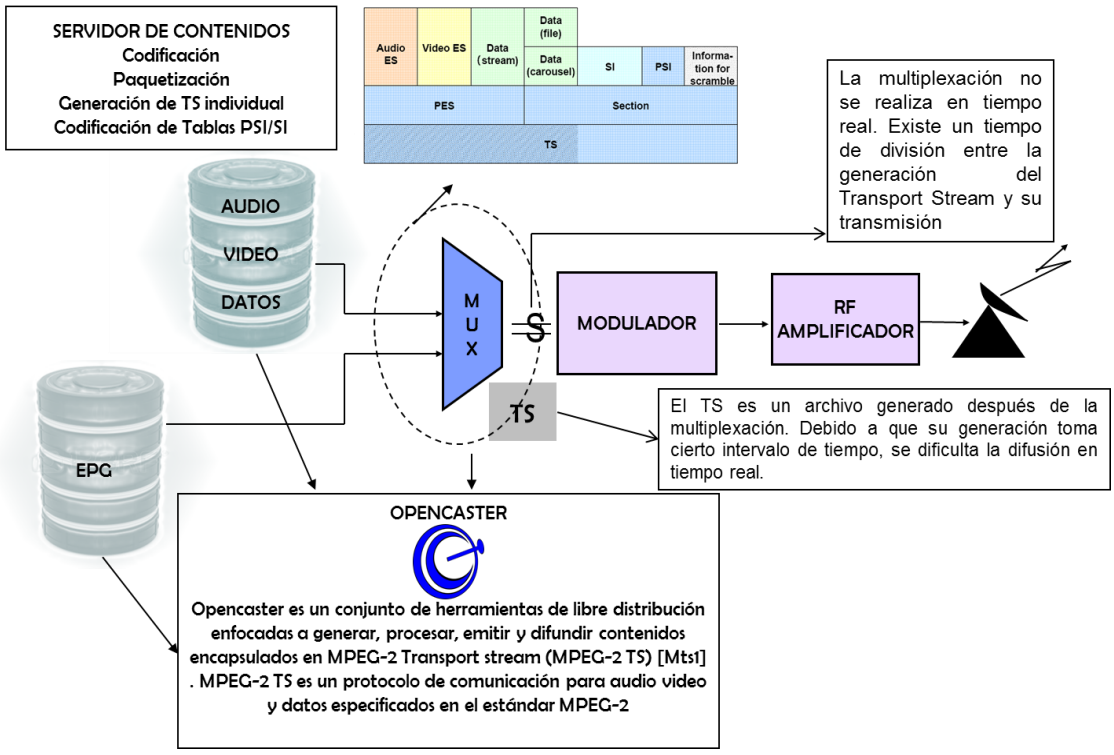


Figura 7: Arquitectura del Broadcaster

[9] OpenCaster es el software destinado al tratamiento de contenidos audiovisuales http://www.avalpa.com/the-key-values/15-free-software/33-opencaster.

2.3. Implementación del laboratorio

Teniendo en cuenta el diseño de la arquitectura planteado en sección 2.2 vamos a describir su implementación dentro de los distintos módulos de la arquitectura general desarrollada en la sección 2.1.

2.3.1. Estación de generación de señales

La arquitectura de este sistema se describió previamente en la Figura 7. Como se indicó anteriormente no se necesita de un servidor de EPG dedicado, debido a la no transmisión en tiempo real. Así el software OpenCaster puede asumir la tarea de los principales componentes como son el servidor de contenidos, generador de EPG, multiplexor y las tareas de sincronización.

En la Figura 8 se puede ver los equipos utilizados en la implementación de uno de los Broadcaster. Primero se cuenta con un computador que soporta la codificación de audio y video. Este equipo debe contener las siguientes herramientas de software:

- OpenCaster: Codificación de Video, audio, datos y EPG.
- Ffmpeg: Herramientas de conversión de audio y video.
- Python: Codificación de Tablas PSI/SI.
- VLC Media Player: Pruebas de visualización de flujo de transporte.

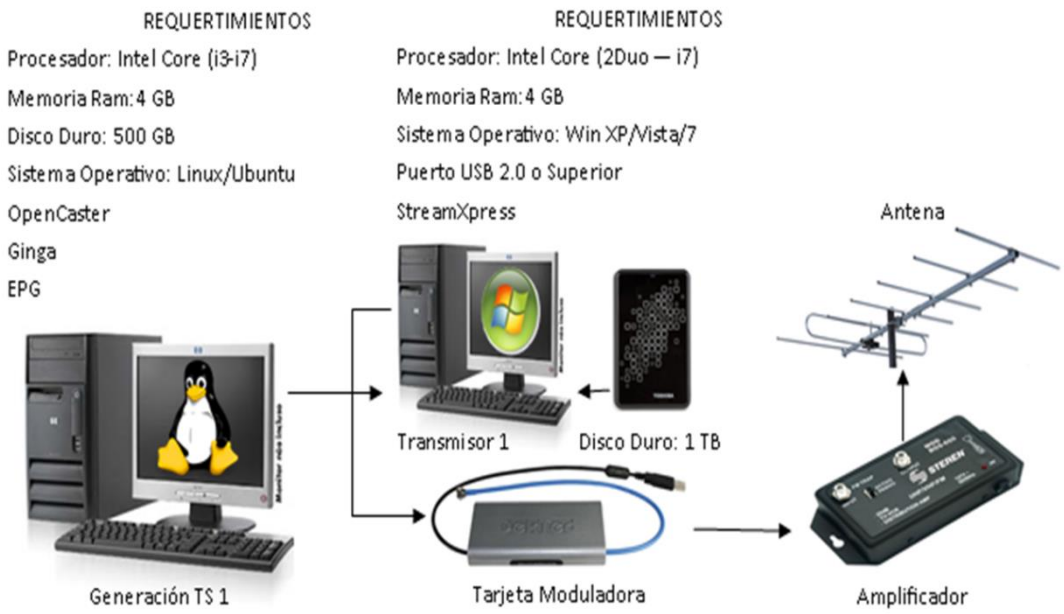


Figura 8: Arquitectura de implementación del Broadcaster

En la misma Figura 8 también se evidencia el uso de otro computador de características normales, sin embargo su capacidad de almacenamiento es más grande debido a que esta computadora también es un repositorio de

contenido multimedia. Además, este se encargará de la comunicación con la tarjeta moduladora DTU- 215, mediante un puerto USB 2.0. Posteriormente de la tarjeta moduladora saldrá la señal a un amplificador, el mismo estará conectado a una antena para radiar la señal generada.

La tarjeta moduladora a su vez tiene una interfaz gráfica de manejo diseñada para Windows llamada StreamXpress. Este software permite visualizar las características de las tablas PSI/SI codificadas en un TS y manejar de manera lógica algunas características de transmisión como: potencia de salida, tipo de modulación, frecuencia, ancho de banda etc. Algunas de estas características se pueden ver en la Figura 9.

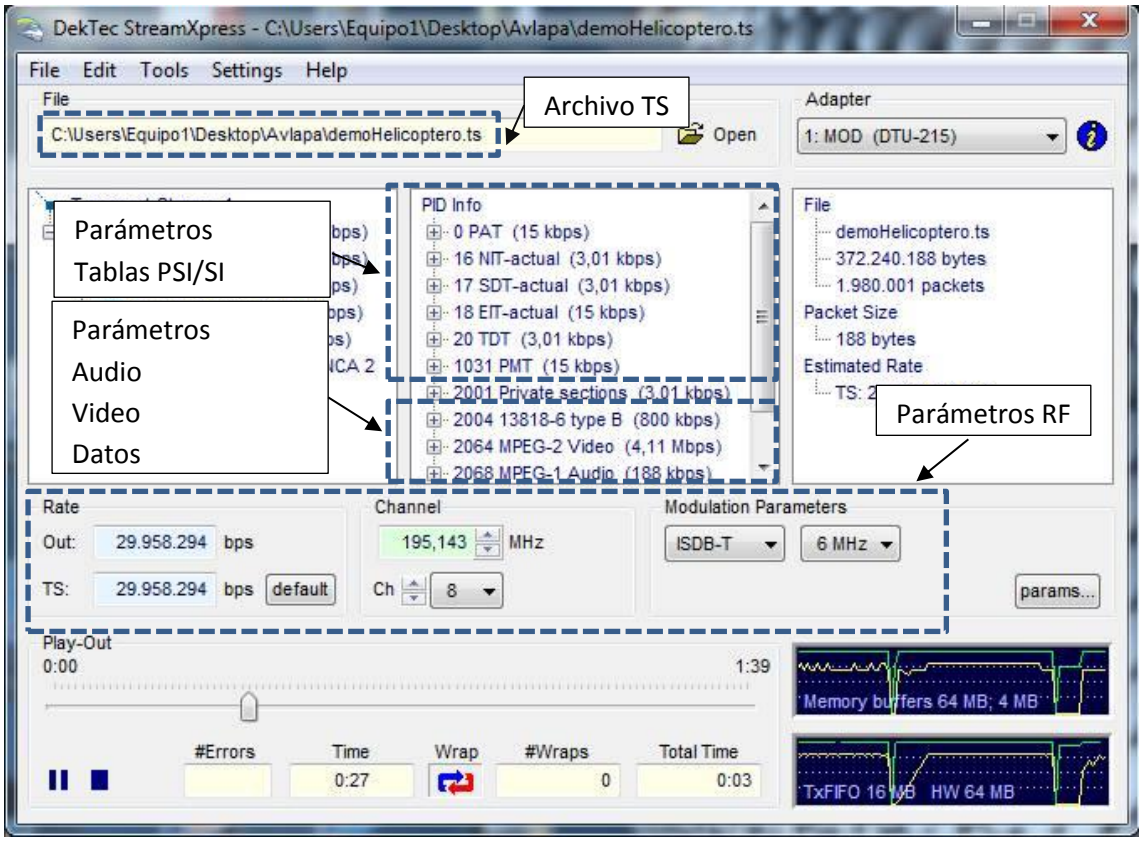


Figura 9: Parámetros de manipulación del StreamXpress

Para distinguir de mejor manera los componentes de la arquitectura dentro de los equipos de la implementación se recomienda revisar el Anexo A.2.

2.3.2. Ambiente de desarrollo de Aplicaciones

Como se ve en la Figura 10, este módulo deberá contar con un computador, en el cual se tiene instaladas herramientas para el desarrollo de aplicaciones GINGA y almacenamiento de las mismas. También se debe adicionar a este computador los componentes de software que permitan emular el funcionamiento de las aplicaciones y herramientas de la comunicación con el proveedor de servicios interactivos, para lo cual se utiliza un router TP-Link.

Los requerimientos de software se detallan a continuación:

- Entorno desarrollo Eclipse: Permite el desarrollo de aplicaciones Ginga-NCL y conexión con el emulador de aplicaciones.
- Entorno de desarrollo Lua: Brinda las herramientas para el desarrollo de scripts de Lua dentro de las aplicaciones.
- Virtual Set Top Box: Nos permite emular el funcionamiento de las aplicaciones en Ginga-NCL.

La instalación de estas herramientas se puede ver en el Anexo A.3.

Figura 10: Implementación del ambiente de desarrollo de aplicaciones

2.3.3. Entorno del usuario o receptor

Las características del laboratorio de usuario o receptor se las puede apreciar gráficamente en la Figura 11, este módulo cuenta con dos antenas receptoras, un STB, y dos televisores. El STB deberá comunicarse con el proveedor de servicios interactivos para lo cual se utiliza el router TP-Link incluido en el módulo anterior. Como se sugirió en 2.2 para poder simular algunos escenarios de usuario se necesita la instalación de más de un receptor, para ello se utiliza el televisor LCD de 32 pulgadas que incluye un demodulador ISDB-T y el STB que será conectado a un televisor analógico de 21 pulgadas mediante la salida de video compuesto (CVBS – A/V[10]), dando así dos características diferentes en la recepción.

El acoplamiento de los receptores al resto de módulos se lo describe en el Anexo A.2.

[10] CVBS – A/V es la interfaz de vídeo analógica que se utiliza en la producción de televisión y en los equipos audiovisuales domésticos

Figura 11: Implementación del receptor

2.3.4. Proveedor de servicios interactivos

El proveedor de servicios interactivos permite a las aplicaciones que se ejecutan en el STB, la gestión de datos y almacenamiento de información. Como se ve en la Figura 12, se concibe a este proveedor como un servidor donde se ejecuta un servicio que acepta las peticiones del usuario. Este servidor tiene las características de hardware necesarias para poder soportar la conexión simultanea de los usuarios. Los requerimientos de software están compuestos por un ambiente de desarrollo en NetBeans y Java que permiten la programación y funcionamiento del servidor. El receptor podrá acceder a los servicios interactivos mediante el router TP-Link, el mismo que emula la conexión a internet mediante una red LAN.

El acoplamiento de este módulo con el resto de la arquitectura se detalla en el Anexo A.2.

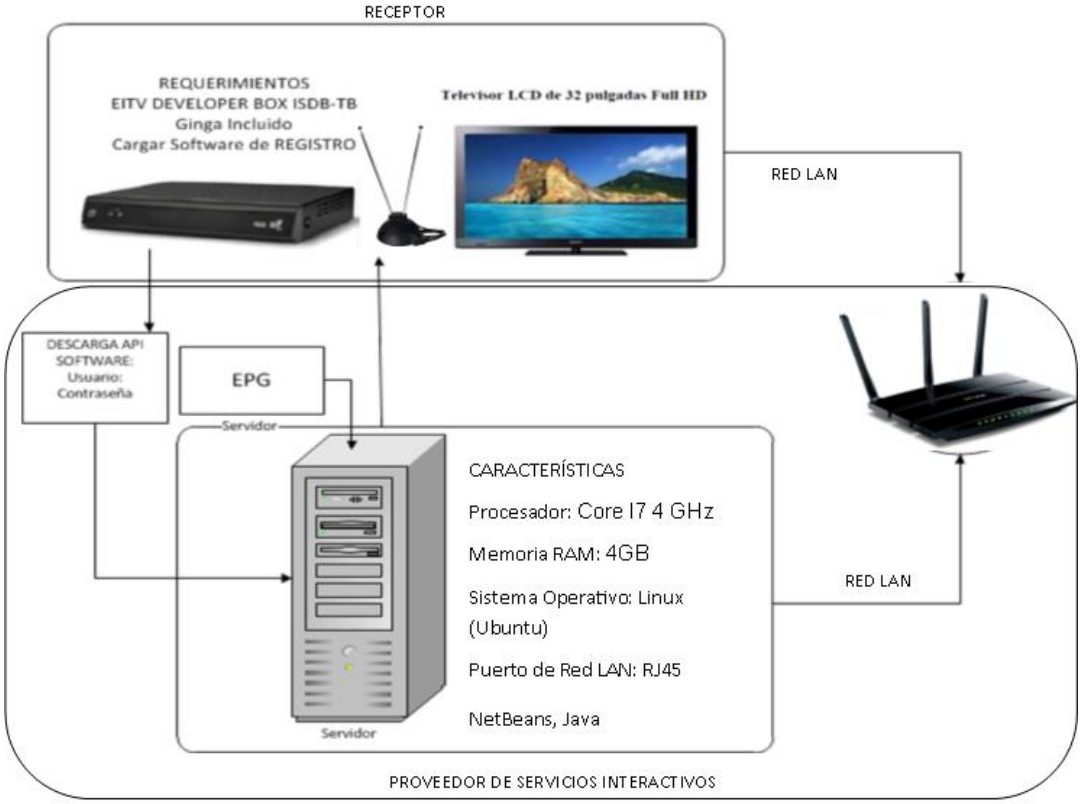


Figura 12: Implementación del proveedor de servicios interactivos

Para poder hacer que el Broadcaster pueda transmitir contenidos audiovisuales y datos, las señales deben incluir referencias e identificadores que muestren como los datos han sido codificados. Estos datos son descritos en la siguiente sección haciendo referencia a las tablas PSI/SI.

2.4. Generación de las tablas PSI/SI con información propia

Las Tablas PSI/SI proveen la señalización y los indicadores que van codificados junto a los contenidos audiovisuales. Si el Broadcaster desea brindar servicios adicionales como EPG y aplicaciones interactivas debe utilizar las tablas de información para que el receptor pueda decodificar estos servicios. Según la información que contengan en el estándar definido en [4] se agregan algunas de estas tablas como obligatorias y otras como opcionales para una transmisión. También según la información que poseen se la clasifica en dos tipos.

2.4.1. Las tablas PSI (Información específica del programa)

Como el TS es un flujo de información de programas audiovisuales que son enviados por el canal de transmisión, el mismo está conformado por diferentes

Elementary Streams[11] que son multiplexados. El TS está conformado por información adicional importante que permite localizar estos ES[12] y demultiplexar los contenidos en forma sincronizada. Ésta relaciona el PID[13] de un ES con los programas al que pertenecen. Las Tablas PSI comprenden 4 tipos de tablas:

Program Association Table (PAT): Nos da información sobre todos los programas presentes en un Transport Stream. Transporta los paquetes que pertenecen a cada programa con su respectivo valor de PID. La tabla PAT es única en cada trama de trasporte y sus paquetes de transporte comienza con un PID = 0.

Conditional Access Table (CAT): Esta tabla nos da información del contenido de acceso condicional del Transport Stream. Esta tabla es necesaria cuando algún programa del TS este codificado, en este caso sus paquetes de transporte viajan con un PID = 1.

Program Map Table (PMT): Todo programa audiovisual que está contenido en un Transport Stream posee una tabla PMT que está asociada con dicho programa. La PMT brinda detalles de cómo está relacionado el programa con los flujos elementales que lo conforman.

2.4.2. Las tablas SI (Información de servicio)

Estas tablas brindan información adicional a las tablas PSI. Contienen información sobre la Guía Electrónica de Programación (EPG), los eventos de cada servicio, los servicios disponibles. También contiene las descripciones textuales, así como también técnicas de cualquier elemento. Estas tablas incluyen, 4 tipos de tablas de inserción obligatoria dentro del Transport Stream que son:

Network Information Table (NIT): Esta tabla transporta información de red física utilizada para transmitir el Transport Stream. Varios canales físicos diferentes pueden conformar una red física, estos a su vez pueden transportar tramas independientes entre sí. El valor de PID que posee la NIT es asignado por la PAT.

Service Description Table (SDT): En la SDT se posee una lista de los servicios asociados a cada Transport Stream. En ella se muestran los nombres y los parámetros asociados a cada servicio, así como su descripción.

[11] Un Elementary Stream hace referencia a un flujo de bits que contiene solo un tipo de dato audio, video o texto.
[12] ES se utiliza como siglas de Elementary Stream
[13] PID es una abreviatura del identificador de proceso dentro de un sistema operativo

Event Information Table (EIT): Esta tabla nos da información relativa de los eventos en curso o futuros. El receptor puede conocer la hora de inicio de un evento, así como su duración, denominación del evento, etc.

Time Date Table (TDT): Esta tabla es utilizada para la transmisión de la fecha actual y de la hora, así el receptor tiene la opción de sincronizarse con el Broadcaster.

La estructura completa de todas las tablas de información se puede ver en el Anexo A.6.

2.4.3. Uso de tablas en el Sistema Recomendador

Para el sistema recomendador se establece como información útil de las tablas mencionadas: el nombre del servicio (canal), título del evento (Programa), descripción del evento, denominación del evento, horario. Estos son los datos descriptivos de mayor disponibilidad, puesto que una EPG, bien conformada es difícil de encontrar, ya que la mayoría de los campos que debe asignar un Broadcaster no son obligatorios.

Toda esta información se las puede obtener de las siguientes Tablas: SDT, EIT, TDT.

SDT (Tabla Descripción del Servicio): Esta tabla contiene (entre otros campos) la información del Nombre Estación de Televisión. Así como el Nombre del Servicio, en el caso de que la estación de televisión posea más canales por ejemplo Canal 7.1 SD, 7.2 HD, etc.

EIT (Tabla Información de Eventos): La tabla EIT posee información del Lenguaje del evento que se está transmitiendo. También contiene la clasificación del programa, así como su título y la respectiva descripción del contenido del mismo.

TDT (Tabla de Hora y fecha): Esta tabla ayuda a la Sincronización del Reloj interno del receptor con el del Broadcaster. Siendo de gran ayuda solo en transmisiones de tiempo real. Hay que tomar en cuenta que la hora varía de acuerdo al Broadcaster.

2.5. Generación del Transport Stream

Uno de los últimos pasos dentro de la codificación del contenido, antes de la transmisión de TV Digital, es la generación del Transport Stream. Este es un flujo final de bits, todos ellos previamente multiplexados que incluyen video, audio, datos, EPG y aplicaciones a transmitir; separados y ordenados con diferentes identificadores para su posterior decodificación.

Todo el proceso de la generación del TS se ilustra en la Figura 13. Aquí observamos que la secuencia de codificación se realiza por separado,

distinguiendo tres partes. Estas partes son codificadas por separado y tienen una secuencia de codificación ordenada, hasta que se multiplexan, para así formar un único servicio de TV Digital a ser transmitido.

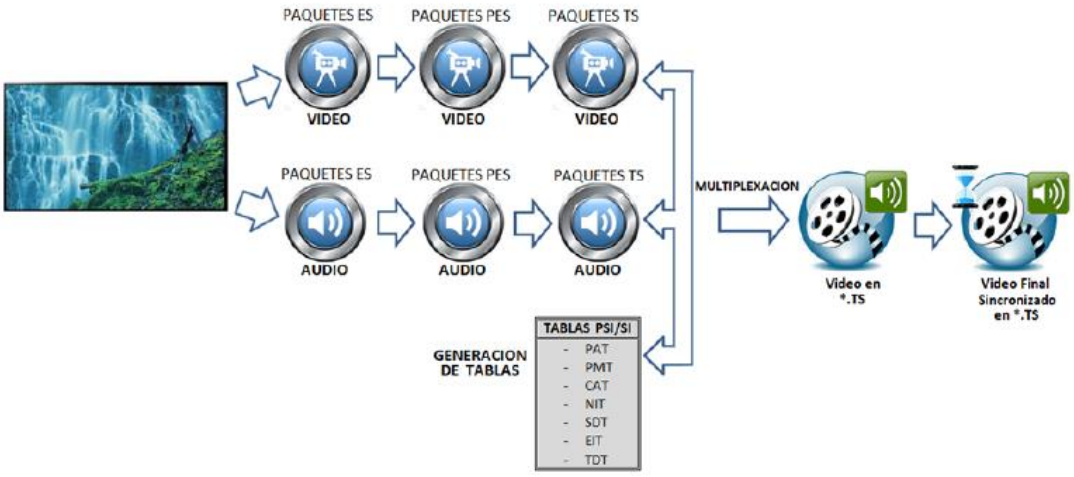


Figura 13: Estructura del Transport Stream

2.5.1. Codificación de video

La codificación del video está compuesta por tres etapas importantes. Primero, la obtención del Elementary Stream que separa el video de un archivo multimedia. Segundo, empaquetar los elementos, donde se encapsulan los ES para así tener un buffer de video de un tamaño dado de bits. Y finalmente está la transformación de estos paquetes en un TS de video. Hay que tener en cuenta que en cada uno de estos pasos previos ya se ha añadido al contenido su identificador, resolución, tasa de transmisión, calidad y ancho de banda.

2.5.2. Codificación de audio

Consiste en tres pasos similares a los descritos anteriormente en la codificación del video. Primero se obtiene un ES separando el audio de un archivo multimedia. El empaquetar se la realiza de igual manera que el video, aquí no se debe especificar el tamaño del buffer, ya que estas vienen definidos por el estándar para MP2[14]. En la transformación de los paquetes en un TS de video, se debe añadir un identificador único del audio; para luego en las Tablas PSI especificar a qué video pertenecen. Algunos parámetros adicionales a considerar en esta etapa son la tasa de muestreo, ancho de banda, parámetros de sincronización con el video, y canales de salida de audio.

2.5.3. Codificación de los datos, EPG y Aplicaciones Interactivas

Antes de codificar los datos se debe tener las tablas PSI/SI previamente llenadas con la información acerca del servicio de TV Digital (ver sección 2.4). A las tablas básicas se les añade las tablas EIT y TDT, las cuales incluyen el

[14] Mp2: formato de compresión de audio con pérdida especificado para el estándar ISDB-Tb

servicio de guía de programación adicional al contenido enviado. Para incluir dentro de los datos una aplicación interactiva se modifican las tablas AIT y TDT añadiendo identificadores y características del carrusel de datos. Una vez generada cada una de estas tablas se hace el cálculo del ancho de banda de acuerdo a la concurrencia con la que el decodificador las necesita.

2.5.4. Multiplexación y generación del TS final

Una vez obtenidas las diferentes secciones del TS procedemos a multiplexarlas todas dentro de un Transport Stream final. Para esto debemos considerar el ancho de banda (BW) al momento de la transmisión. Es necesario considerar que la calidad de video y audio sean las adecuadas y que las diferentes tablas sean recibidas de manera óptima por el receptor. Toda esta información se ordena en un flujo final de bits, distinguiéndolas por su identificador y su BW. Una vez obtenido este TS para ser decodificado satisfactoriamente, se debe sincronizar la tasa de transmisión a 29958294 bps. Hay que tener en cuenta que cuando el contenido no alcanza esta tasa de transmisión, al TS se le añaden paquetes nulos que complementan la tasa de transmisión indicada, de lo contrario se presentarían errores en el receptor, ya que no podrá sincronizar adecuadamente el contenido.

Todos los parámetros de la codificación de audio, video, datos, EPG, aplicaciones interactivas, multiplexación y sincronización de contenido se detallan en el Anexo A. 5.

RESUMEN DE CAPÍTULO

El capítulo ARQUITECTURA DEL TRANSMISOR Y RECEPTOR DE TDT EN EL ESTÁNDAR ISDB-TB, EN LA IMPLEMENTACIÓN DEL LABORATORIO Y TRANSMISIÓN DE SERVICIOS DE TV DIGITAL abordo los temas concernientes a la puesta en marcha del laboratorio. Se mostró como se hizo uso de un diseño de arquitectura para el laboratorio. A este diseño se le fue acoplando los equipos adquiridos y se mostró sus características técnicas y requerimientos para su funcionamiento. Se planteó un diseño por módulos y a en cada módulo propuesto se mostró un conjunto de funciones a cumplir. Finalmente, se completó la puesta en marcha del laboratorio al mostrar cómo se hace la codificación y la multiplexación de los contenidos audiovisuales, EPG y datos adicionales.

DESARROLLO DE LA APLICACIÓN GINGA-NCL Y LUA PARA LA CAPTURA DE DATOS DEL TRANSPORT STREAM

El capítulo tres aborda el tema del desarrollo de la aplicación Ginga-NCLua. Esta aplicación permite al usuario interactuar con el sistema recomendador. En una primera parte se describe su estructura definiendo la funcionalidad de la aplicación, los requerimientos y las tareas que debe cumplir. Posterior a esto se detalla de manera técnica su desarrollo dividiéndola en algunos módulos o aplicaciones según la función que desempeñen. Para cada aplicación se especifica la secuencia de funcionamiento, la implementación y la simulación. Y a su vez se describe como estas aplicaciones interactúan con el servidor para consolidar la aplicación interactiva global.

CAPÍTULO 3

DESARROLLO DE LA APLICACIÓN GINGA-NCL Y LUA PARA LA CAPTURA DE DATOS DEL TRANSPORT STREAM

3.1. Estructura de la Aplicación

Uno de los objetivos principales dentro del presente proyecto es contribuir a la captura de la actividad de un usuario de TV digital. Para ello se planteó el uso de una aplicación que se adapte a las características del middleware GINGA, la cual se encuentra desarrollada en NCL. Para que la aplicación pueda cumplir con funciones complejas que no permiten NCL al ser únicamente un entorno de presentación multimedia, se hace uso de scripts en Lua. Este lenguaje es de uso universal y está adaptado a GINGA como se especifica en [5], permitiendo utilizar el canal de retorno y haciendo más fácil la proyección de datos para la interactividad con el usuario.

3.1.1. Funciones que desempeña la Aplicación

Para que la aplicación nos permita que el usuario interactúe de forma correcta con el servidor de aplicaciones se plantea las siguientes funciones a desempeñar:

Registro: Esta función permite al usuario solicitar su integración al "Sistema Recomendador". Para ello deberá especificar datos como id de usuario y contraseña para el posterior inicio de sesión; además de debe incluir datos como sexo y edad para permitir al sistema generar un perfil previo de usuario según estereotipos[15].

Inicio de sesión: Permite a un usuario previamente registrado poder informar al servidor las actividades generadas sobre el receptor, con lo cual da paso a la captura de la actividad del televidente, envío y almacenamiento de datos por parte del servidor.

Visualización de la recomendación: Permite al sistema poder proyectar recomendaciones almacenadas para un determinado usuario con sesión activa en el sistema. Esta función arma y proyecta una guía de programación para este usuario.

Cierre de sesión: Permite al usuario inhabilitar su cuenta temporalmente en un determinado receptor, adicional a esto notifica a la aplicación que ya no debe enviar la información del usuario al servidor.

Las funciones descritas anteriormente se mostrarán dentro de un menú de usuario donde se puede seleccionar las distintas opciones. Sin embargo, para

[15] Estereotipos hace referencia a la percepción, que se tiene sobre una persona o grupo de personas que comparten ciertas características

poder cumplir con todas las necesidades del proyecto se debe generar funciones que se ejecuten en segundo plano. Estas funciones son las siguientes:

Captura de actividad del Televidente: Esta función permite a la aplicación capturar datos que están incluidos en el Transport Stream como son: número de canal, nombre del canal y nombre del programa. Estos datos posteriormente permitirán predecir sus gustos televisivos y armar un perfil de usuario.

Envió y recepción de los datos: Permite la transferencia de los datos entre el servidor y la aplicación. Con esta función se puede realizar consultas a las bases de datos del servidor de aplicaciones y recibir respuestas adecuándolas para su proyección hacia el televidente.

3.1.2. Diagramas de flujo

Para complementar las funciones descritas anteriormente se hace uso del diagrama de flujo visto en la Figura 14. Aquí se pueden distinguir cuatro flujos diferentes que van a ser elegidos por el usuario. Estos flujos corresponden a cuatro interfaces que serán manejadas por el usuario; estas son: interfaz de registro, inicio de sesión (Log in), recomendación y cierre de sesión (Log out). Las dos primeras se presentarán al usuario en caso de no existir un registro de sesión almacenada en un archivo en el STB, estas interfaces se presentan como el flujo de menú 1. Por otro lado si el registro existe se presentarán las interfaces de recomendación y cierre de sesión puesto que se supone que los datos de un usuario ya se encuentran almacenados en este registro, y estos corresponderían al menú 2.

La interfaz de Registro, en el menú 1, ayudará al usuario a registrar sus datos dentro del servidor y destinar un espacio de memoria en el mismo para almacenar su actividad como televidente. Una vez que un usuario se haya registrado se hace la primera captura de datos, que corresponde a la captura de la información del canal sintonizado.

El flujo de la interfaz de Login es similar al del registro a diferencia que los datos a ingresar son solo un id y contraseña. Estos datos se almacenan y permiten a la aplicación gestionar recursos del servidor verificando estos datos previamente. Posterior a un Login la aplicación sigue un flujo similar para la captura de datos del canal sintonizado.

En el menú 2, la interfaz de Recomendación sigue un flujo cuya finalidad es mostrar al usuario una Guía de Programación. Esta guía es enviada por el servidor y contiene los gustos del televidente, todos estos datos son generados por el resto de módulos del proyecto descrito en [1]. El flujo de la interfaz de Log out es el más simple de todas las cuatro interfaces, lo único que permite es borrar los archivos de registro de una sesión activa. Al borrar estos datos la

aplicación no podrá hacer ninguna consulta al servidor previo a una nueva autorización del usuario con las interfaces en el menú 1.

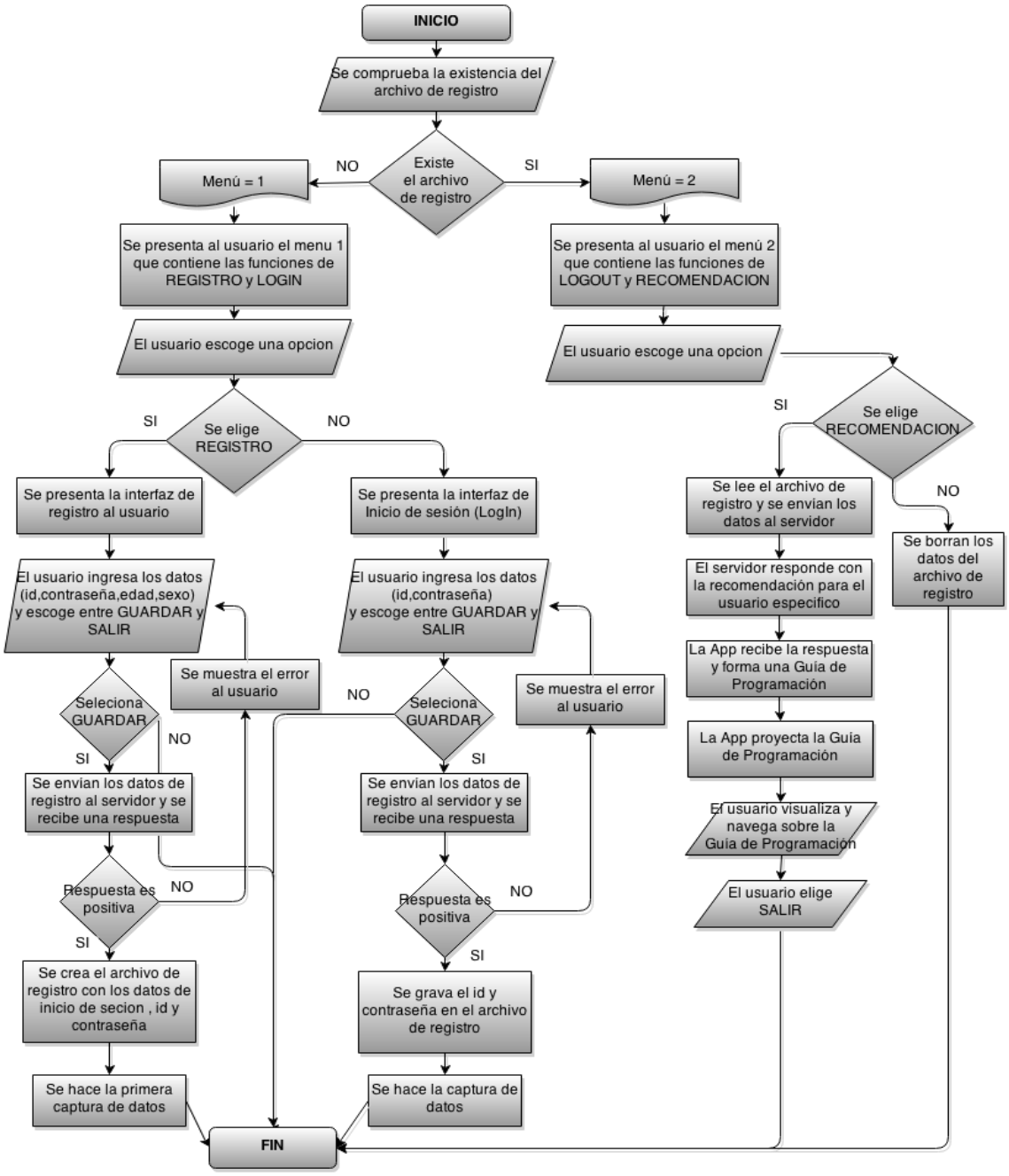


Figura 14: Diagrama de flujo de la Aplicación Interactiva

3.1.3. Características adicionales

Como se mencionó anteriormente la aplicación necesita ejecutar algunas funciones en segundo plano. La ejecución de estas funciones nos permite que el usuario pueda disfrutar de una manera habitual los contenidos televisivos y

que la aplicación interactúe con él en los momentos necesariamente obligatorios.

Lo que habitualmente sucede por parte del usuario es la necesidad de acceder a las aplicaciones almacenadas en un sector de memoria del receptor y ejecutar la aplicación que se requiera. Pero al realizar este procedimiento, se presentarían problemas para el usuario puesto que afectarían sus intereses de entretenimiento al momento de ver televisión. Este problema se presenta al tener en cuenta que la ejecución de una aplicación en Ginga no es permanente y que depende de la permanencia del usuario en un determinado canal. Es decir si se pretende capturar la actividad del usuario en un canal de televisión determinado, se debería ejecutar una aplicación cada vez que cambie el canal.

La solución al problema anterior, requiere que la aplicación se ejecute de manera automática en el receptor. El estándar ISDB-Tb permite la ejecución automática de aplicaciones enviadas a través del Transport Stream. El uso de esta característica permite ejecutar la aplicación cada vez que el usuario cambie de estación. Esto conlleva a que todos los Broadcasters envíen la aplicación junto a sus contenidos de forma obligatoria.

En un escenario real se debe aplicar una medida de regulación a las emisoras. Estas deben enviar la aplicación a través de su infraestructura para beneficiarse de las múltiples tablas estadísticas que se podrán generar con la captura de la actividad de sus usuarios (horas pico de sintonización, rating, preferencia por edades, comparación con otras emisoras). Otro beneficio para las emisoras es la exclusividad, debido a que solo se podrá capturar la información de los canales que transmitan la aplicación, y cuando se forme un perfil del usuario solo se podrá recomendar estos canales.

3.2. Generación de aplicación por Transport Stream

Como se explicó en las características de la aplicación esta será enviada a través del Transport Stream. Como se especifica en [4], al enviar la aplicación se debe generar un carrusel de datos. El mismo permite al receptor recibir la señal y posteriormente ordenar y almacenar la aplicación adicionando los datos que se encuentran codificados en el carrusel.

Para poder codificar el carrusel de datos OpenCaster ofrece la herramienta oc-update.sh. Esta herramienta nos permite adicionar algunos identificadores a la aplicación y codificarla como un flujo de transporte .ts que será multiplexada en el Transport Stream final.

Como se ve en la parte superior de la Figura 15 la carpeta SistemaCaptura contiene todos los objetos NCL, Lua, imágenes y carpetas. Con el comando oc-update.sh (parte resaltada) se va a especificar este directorio y adicionar los identificadores necesarios para la codificación detallados en el Anexo A.7. Posterior a la ejecución de este comando se ha generado el carrusel de datos que se encuentra en SistemaCaptura.ts. Este carrusel de datos podrá ser multiplexado en el Transport Stream final como se especifica en 2.4.

Adicional a la generación del Transport Stream de la aplicación, se debe incluir parámetros e identificadores a las tablas AIT y PMT. Estos parámetros se presentan en la Figura 16 . En la tabla AIT se debe resaltar el uso de “aplication type = 0x0009” que hace referencia una aplicación de tipo Ginga, “application_control_code = 0x01” para que la aplicación se ejecute de manera automática en el receptor. Mientras que en PMT se identifica la aplicación a través de sus identificadores. Las tablas con la información completa se detallan en el Anexo A.7.

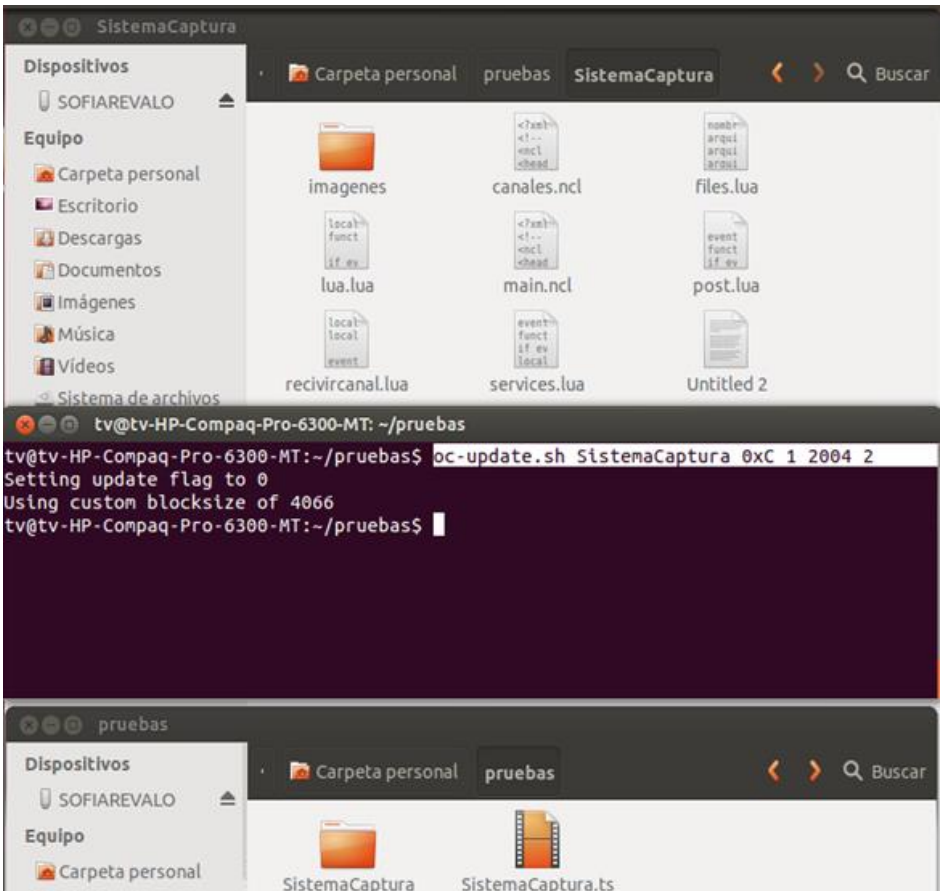


Figura 15: Generación del carrusel de datos para la aplicación GINGA

```
ait = application_information_section(
    application_type = 0x0009, # GINGA-NCL
    common_descriptor_loop = [],
    application_loop = [
        application_loop_item(
            organisation_id = 0x0000000A,
            application_id  = 0x64,
            application_control_code = 0x01, # AUTOSTART
```

AIT

```
stream_loop_item(
    stream_type = 5, # AIT stream type
    elementary_PID = 2001,
    element_info_descriptor_loop = [
        data_component_descriptor (
            data_component_id = 0xA3, # sistema AIT
            additional_data_component_info = ait_identifier_info(
                application_type = GINGA_NCL_application_type,
                ait_version      = 0
            ).bytes(),
        ),
        application_signalling_descriptor(
            application_type = 9, # 9 GINGA-NCL
            AIT_version      = 1, # current ait version
        ),
    ]
),
```

PMT

Figura 16: Adición de parámetros dentro de las tablas AIT y PMT

3.3. Definición de las Aplicaciones Interactivas

Para complementar la funcionalidad de la aplicación interactiva, ésta se divide en múltiples aplicaciones. Estas aplicaciones se encargarán de una determinada función, sirviendo de interfaz para que el usuario interactúe con el servidor. Al implementar el diagrama visto en 3.1.2, se ve necesario el uso de las siguientes aplicaciones: app de registro, app de inicio de sesión (log in), app de recomendación y app de finalización de sesión (log out).

Como se detalla en la Figura 17 cada una de las cuatro aplicaciones tiene una función específica que se detalla a continuación.

- APP REGISTRO presenta la interfaz para adquirir los datos del usuario que desee ingresar al sistema.
- APP RECOMENDACIÓN presenta al usuario una guía electrónica de programación con su respectivo menú de control para que el usuario navegue sobre él.
- APP LOG IN permite recibir datos para el inicio de sesión
- APP LOG OUT permite al usuario indicar que se cierra su sesión y el paso de sus datos por parte del servidor.

Adicional a esto podemos ver que hay otras funciones que se necesitan para la comunicación entre el STB y el servidor, para complementar estas funciones se utilizan scripts de Lua que se ejecutan en el STB y que no necesitan ser vistos por el usuario. Estas funciones se explicaron en la sección 3.1.2 haciendo referencia a funciones que corren en segundo plano.

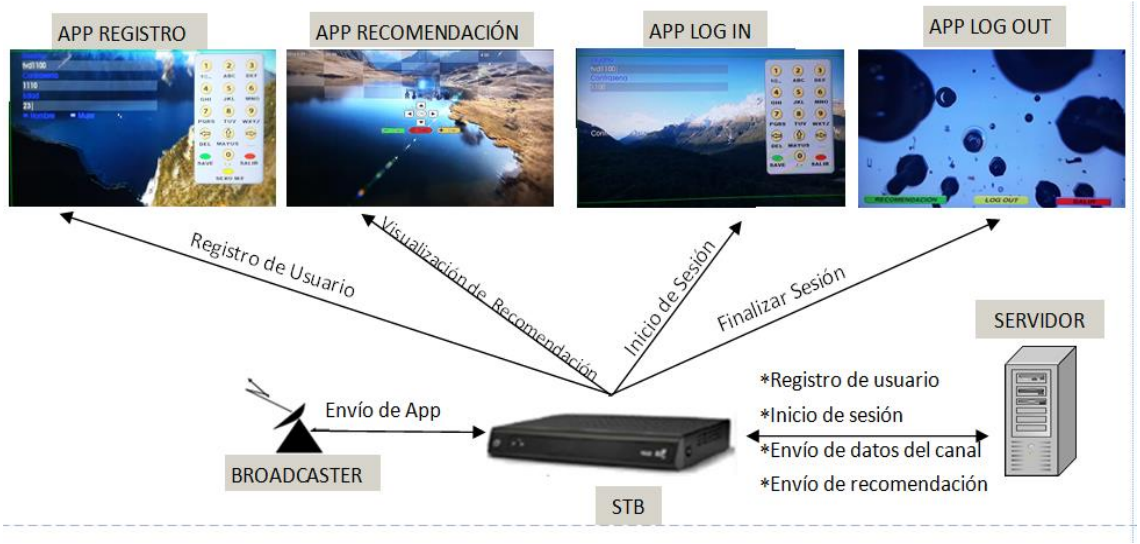


Figura 17: Diagrama de flujo de la Aplicación Interactiva

3.4. Aplicación de registro

La aplicación principal tendrá la tarea de comprobar si existe un archivo de registro creado para el cliente. En caso de no estar registrado nos mostrará las interfaces de usuario como se ve en la Figura 18. Estas interfaces corresponden al menú 1 descrito en la sección 3.1.2. Se agrupó las dos aplicaciones registro e ingreso, debido a que con ambas el usuario podrá tener acceso al sistema. Los permisos generados para la captura de información se dan a partir de que el usuario se haya registrado e iniciado sesión en el sistema.

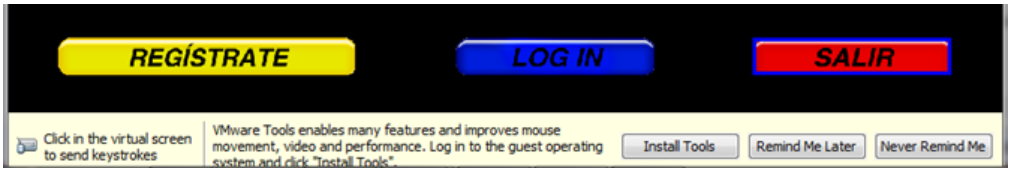


Figura 18: Interfaz del menú 1 registro e inicio de sesión

La aplicación de registro o APP REGISTRO es la interfaz con la cual una persona o familia puede ingresar sus datos al sistema recomendador. La tarea de esta interfaz es adquirir los datos del usuario: *id de usuario, contraseña, edad y sexo*. Estos dos primeros datos sirven para distinguir a una persona de entre los demás usuarios. La edad y el sexo son datos que permitirán al sistema recomendador predecir gustos mediante estereotipos ya que por lo general grupos de la misma edad tienen gustos similares. Algo parecido podemos predecir en grupos de personas del mismo sexo. Otra tarea que tiene esta interfaz es la de almacenar los datos localmente y solicitar el envío hacia el servidor mediante un script Lua de conexión TCP.

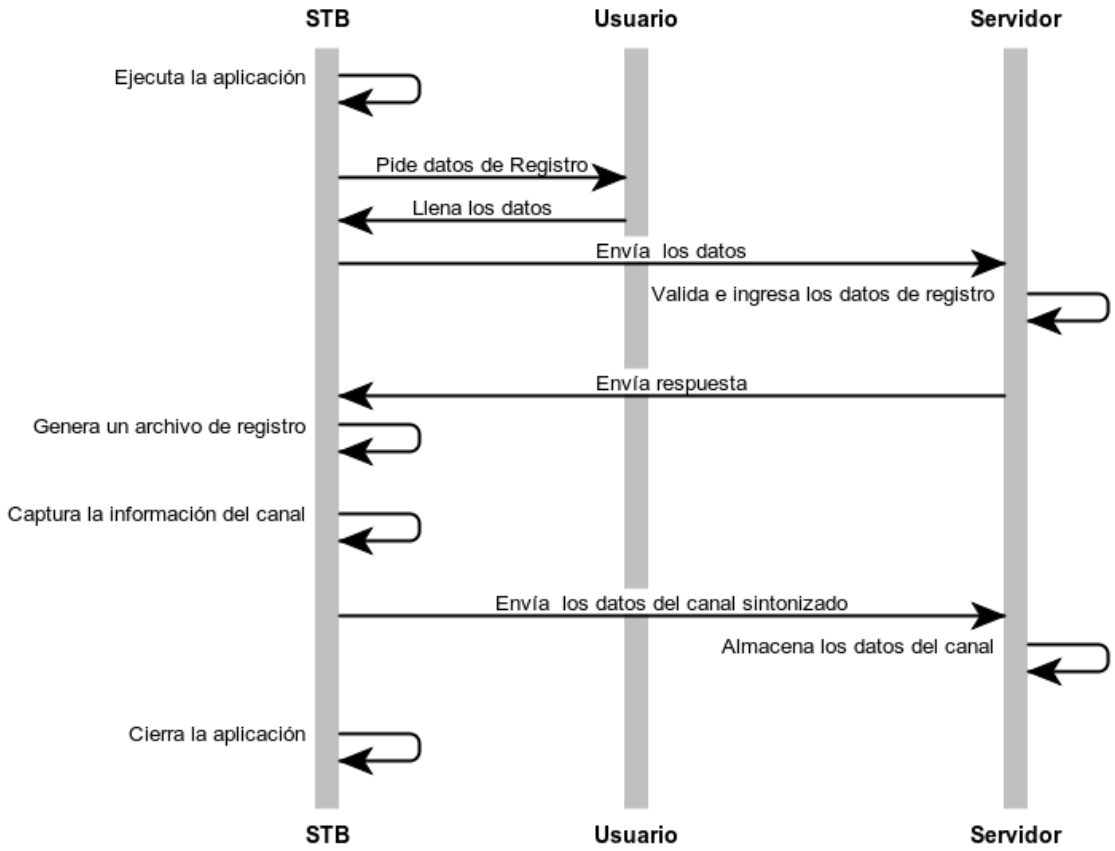


Figura 19: Diagrama de secuencia para la APP REGISTRO

El diagrama de secuencia Figura 19 muestra de mejor manera el funcionamiento de la aplicación. Los puntos principales de este diagrama son:

- Validación de datos de registro por parte del servidor: Se debe garantizar que el nombre de usuario no se repita y que los datos estén completos.
- Generación de un archivo de registro: Este archivo sirve para poder almacenar los datos del usuario localmente y que en un uso posterior no se tenga que hacer consultas innecesarias tanto al usuario como al servidor.
- Captura de la información del canal: Mediante el uso de la APP de Captura que se explicará posteriormente, se podrá hacer uso de los datos transmitidos por el canal a través del Transport Stream para levantar estadísticas de la actividad del usuario.
- Envío de datos: Todas las comunicaciones que se hagan con el servidor (envío y recepción de datos) se los realizará mediante el uso de Funciones Lua con conexiones TCP, estas funciones están descritas en la sección 3.6.

La implementación de estas funciones se ve en la Figura 20. En donde se demuestra la conexión hacia el servidor ubicado en 172.16.146.127 y que recibe peticiones a través del puerto 9999. Posterior a esto se arma un mensaje conteniendo todos los datos del usuario, se envían los datos, se

recibe una respuesta y se validan los datos en la comparación con el string “RegistroCorrecto%”. Finalmente se da paso a la generación del archivo de registro archivo.txt, donde se almacenan los datos del usuario. La implementación completa de la App se muestra en el Anexo A.4.

```
tcp.connect('172.16.146.127', 9999) -- Conectar con el servidor
local result
local mensaje='Registro/'..Usuario..'/'..Contrasena..'/'..Edad..'/'..Sexo..'/fin%'
tcp.send(mensaje)                   -- Envio de datos
result = tcp.receive("*a")          -- Respuesta del servidor
if result then
   if result=='RegistroCorrecto%' then -- Validacion de los Datos
        -- Generando el archivo de registro
        arquivo= assert(io.open('/archivo.txt','w'),'Archivo no puede ser creado')
        arquivo:write('logueado\n'..Usuario..'\n'..Contrasena..'\n'..Edad..'\n'..Sexo)
        arquivo:flush()
        io.close(arquivo)
        canvas:attrColor('white')
        canvas:drawText(dx/20,8*dy/16,'Registro completado con exito')
        canvas:drawText(dx/20,9*dy/16,'Usuario: '..Usuario)
        canvas:drawText(dx/20,10*dy/16,'Contrasena: '..Contrasena)
        canvas:flush()
                event.timer(4000, function()
                    event.post {
                        class  = 'ncl',
                        type   = 'presentation',
                        action = 'stop',
                    }
                end)
```

Figura 20: Implementación de funciones APP REGISTRO

La Figura 21, muestra la interfaz simulada del ingreso de un usuario nuevo al sistema. El usuario visualiza las opciones que tiene para ingresar los datos a través del control remoto. Aquí se puede ver los datos de usuario como son:

- Id Usuario: tvd1100
- Contraseña: 1100
- Edad: 23
- Sexo : Mujer

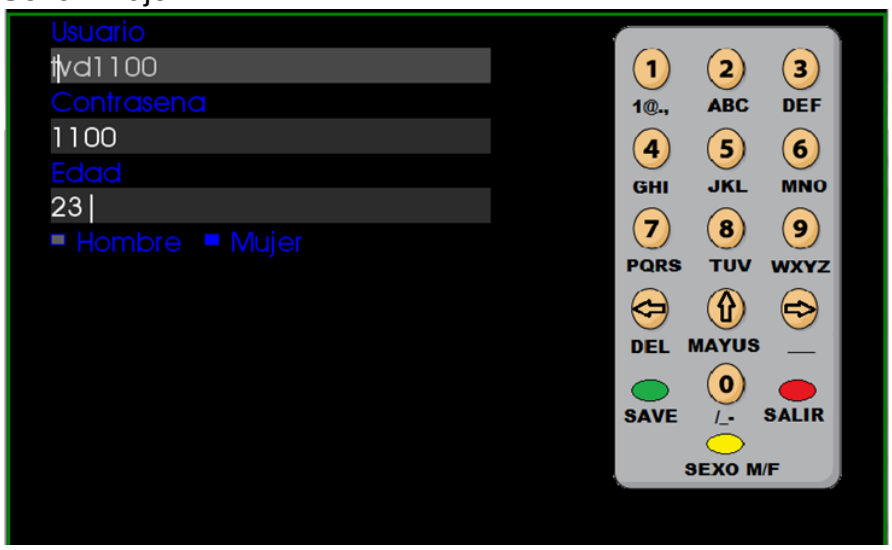


Figura 21: Interfaz de APP REGISTRO

3.5. Aplicación de inicio de sesión

La aplicación de inicio de sesión o APP LOG IN es la interfaz presentada a los usuarios para que estos puedan informar al servidor que existe su autorización para la captura de sus datos. Esta interfaz adquiere los datos del cliente *id usuario y contraseña*, con los cuales se lo identifica del resto de televidentes. Para hacer uso de esta aplicación el usuario debe estar previamente registrado en el sistema. Como función adicional esta aplicación guarda los datos del usuario que inició la sesión y habilita la captura de información del canal de manera automática.

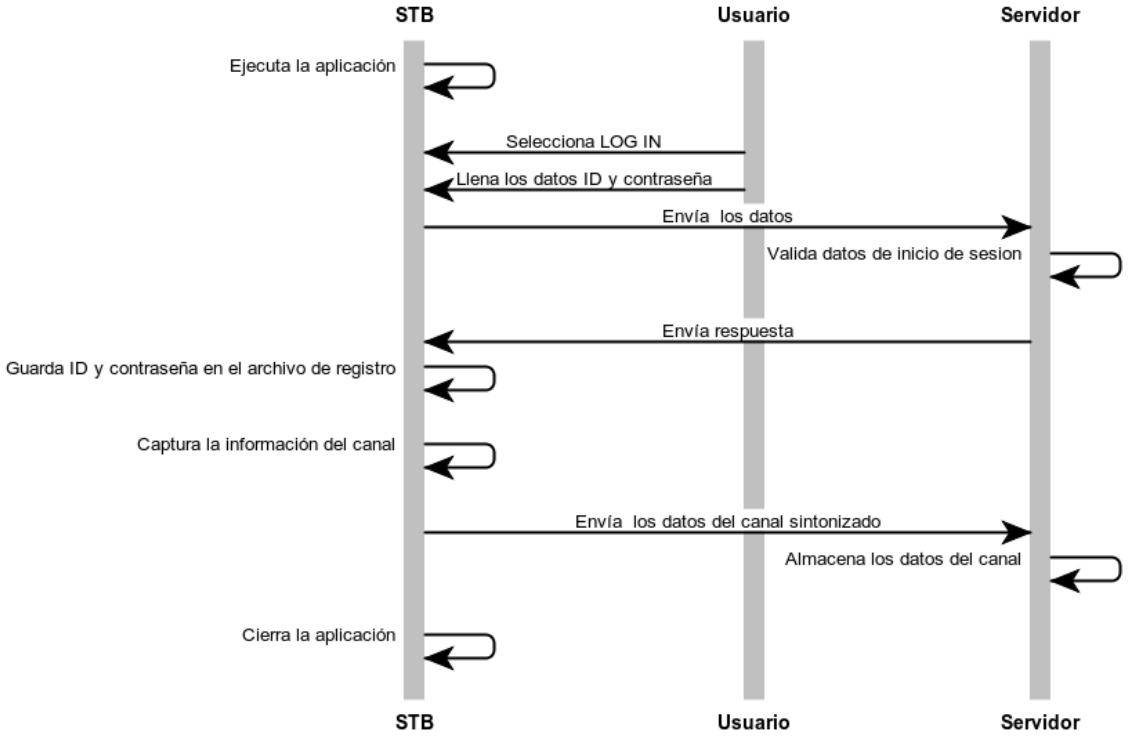


Figura 22: Diagrama de secuencia para la APP LOG IN

En el diagrama de secuencia Figura 22 se muestra el funcionamiento de la aplicación, donde las secuencias principales son:

- Valida datos de inicio de sesión: Permite al servidor garantizar que los datos ingresados pertenecen a un usuario del sistema.
- Guarda ID y contraseña en el archivo de registro: Evita consultas innecesarias al servidor y la aplicación puede darse cuenta por si misma cual fue el usuario que inició sesión.
- Captura de la información del canal: Con el inicio de sesión se habilita nuevamente a la aplicación para capturar, de manera automática, los datos de la estación de televisión .
- Envío de datos: Al igual que en APP REGISTRO el paso de información cliente–servidor es un lapso importante que permite llenar las bases de datos, estos datos posteriormente crearán el perfil del usuario.

La implementación de la aplicación y sus secuencias se especifica en la Figura 23. Como se ve el código es similar a la aplicación de registro con unas pequeñas modificaciones. Primero, al enviar los datos se debe especificar que corresponde a un inicio de sesión mediante el comando “Login/”, los datos enviados aquí son usuarios y contraseña. Otra ligera modificación es a la respuesta recibida del servidor, en este caso es “UsuarioCorrecto%”, esta nos permite completar el inicio de sesión guardando los datos del usuario o informarle al usuario sobre algún error al momento de comprobar sus datos.

```
tcp.connect('172.16.146.127', 9999)
local result
--envío de solicitud al servidor
local mensaje='Login/'..Usuario..'/'..Contrasena..'/fin%'
--recepción de mensaje de confirmación
tcp.send(mensaje)
result = tcp.receive("*a")
if result then
      --comprobación de correcto inicio de sesión
      if result=='UsuarioCorrecto%' then
              arquivo= assert(io.open('/archivo.txt','w'),'Archivo no creado')
              arquivo:write('logueado\n'..Usuario..'\n'..Contrasena..'\n')
              arquivo:flush()
              io.close(arquivo)
              canvas:attrColor('white')
              canvas:drawText(dx/20,6*dy/16,'Logueo completo')
              canvas:drawText(dx/20,7*dy/16,'Usuario: '..Usuario)
              canvas:drawText(dx/20,8*dy/16,'Contrasena: '..Contrasena)
```

Figura 23: Implementación de funciones de APP REGISTRO

La Figura 24 muestra la interfaz simulada del inicio de sesión. El control remoto presenta unas pequeñas modificaciones con respecto al registro, esto debido a que solo se necesita ingresar dos datos. En este ejemplo se pueden distinguir la siguiente información:

- Usuario: tvd1100
- Contraseña: 1100

Una vez el usuario guarde estos datos se desplegará un mensaje que informe si el inicio de sesión se completó con éxito.

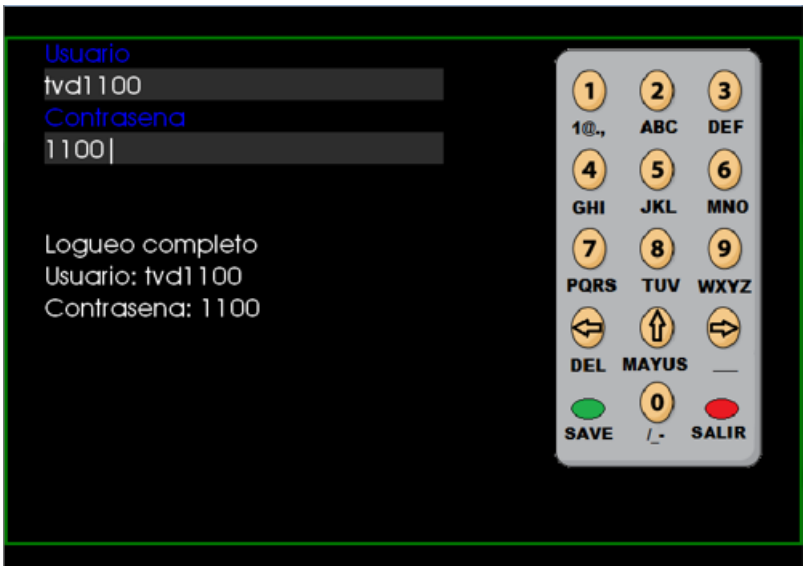


Figura 24: Interfaz de APP REGISTRO

3.6. Aplicación de captura de datos del Transport Stream

Esta aplicación permite al sistema poder capturar los datos transmitidos por el Transport Stream. La App no presenta una interfaz directa al usuario, ya que se ejecuta en segundo plano. Esto permite tener autonomía por parte de la aplicación para realizar la captura de datos en cualquier momento que sea necesario, debido que para el usuario sería molesto presionar un botón aceptando la captura de datos cada vez que cambia de canal. Este tipo de acción se realizó con la ventaja de la ejecución automática de la aplicación al sintonizar un canal de televisión.

Para la ejecución de esta aplicación es necesario previamente tener un archivo de registro generado. De este archivo se leerán los datos del usuario, se le añadirán los datos capturados del canal y posteriormente se enviará la petición de almacenamiento al servidor. Otro aspecto importante se da una vez que el usuario se registre o inicie sesión en donde la aplicación cuenta con los permisos necesarios para hacer esta captura de la actividad del televidente, por lo que el usuario debe estar consciente que si permanece con una sesión activa la aplicación enviará datos al servidor sin que él lo perciba.

Una de las primeras soluciones que se manejaron al momento de realizar la captura de datos del Transport Stream es la planteada en [6] para programación Lua. Aquí se habla de los eventos de la clase “SI” (system information): Este tipo de eventos permite a Lua, realizar una petición al middleware Ginga para poder acceder a los datos del TS. Los tipos de información que maneja este evento son:

- services: Accede a la información del tipo de servicio transmitido por el Broadcaster.
- mosaic: Accede a la información tipo mosaico de algunos identificadores del Transport Stream.

- time: Permite acceder a información de variables de tiempo enviada por el Broadcaster.
- epg: Permite acceder a información contenida en la guía electrónica de programación.

Un evento de clase “si” y tipo “epg” o “service” teóricamente permite al script acceder a información del canal y la programación (ver Figura 25). De esta forma podemos obtener el nombre del canal, nombre del programa e identificador del canal, con lo que se estaría cumpliendo con el objetivo de esta aplicación. El problema de este tipo de librería o evento es que el estándar descrito en [5], no lo contempla como obligatorio para la construcción de un receptor. Tomando en cuenta este problema la aplicación estaría limitada solo al uso de receptores que especifique el uso de esta herramienta, por lo que el modelo se planteó para un entorno ideal de funcionamiento.

```
function tratador (evt)
    if(evt.class== 'ncl' and evt.type=='presentation' and evt.action=='start') then
        --solicitud de datos del servicio
        event.post('out', { class='si', type='services'})
        -- solicitud de datos del EPG
        event.post('out', { class='si', type='epg' , stage='current' })
    end
    if evt.class ~= 'si' then return end
    if evt.type == 'service' then
        --Captura de información
        canalid=data[1].id
        nombrecanal=data[1].providerName
    end
    if evt.type == 'epg' then
        --Captura de información
        nombreprograma=data[1].name
    end
end
event.register(tratador)
```

Figura 25: Implementación de clase “si” para captura de información del Transport Stream

Para dar solución al problema de especificaciones del receptor se plantea la captura de datos de una manera diferente. Como sabemos la aplicación hace uso del Transport Stream para llegar al receptor, puesto que todos estos datos son codificados por parte del Broadcaster, éste puede ordenar los datos del canal y la programación para embeberlos dentro del carrusel de datos que porta la aplicación. Este procedimiento se debe realizar antes de la codificación de los datos y la aplicación. Con esto la aplicación también transportará datos adicionales correspondientes al canal que está transmitiendo la señal. Así, una vez ésta llegue al receptor la aplicación podrá hacer uso de esta información sin tener que hacer peticiones al middleware, ya que las consultas se harán en el carrusel de datos propiamente.

La Figura 26, muestra la implementación del método de captura. El enlace “data/epg.data” es un espacio de memoria en el carrusel de datos donde esta almacenada la información del canal como [número, nombre del canal, nombre del programa]. La ventaja del uso de este espacio de memoria se da según los requerimientos del sistema a utilizar para obtener datos adicionales enviados

por el Transport Stream. Se accede a la información de este archivo a través de “io.open” y una vez direccionado el archivo se extraen sus datos mediante un “arquivo:read()”. Los datos obtenidos se concatenan con la información del usuario obtenida del archivo de registro “archivo.txt”. Para finalmente ser enviados al servidor mediante la siguiente estructura “ ’Historial/’..lineas[2]..’/’..linea1..’/fin%’ “, esta cadena de datos es interpretada por el servidor como una historia de usuario cuyos datos van en la variable lineas[2] y la captura de la actividad en la variable linea1.

```
tcp.connect('172.16.146.127', 9999)
local nombrefile='data/epg.data' --Datos de canal y programación
arquivo= assert(io.open(nombrefile,'r'),'Dato no leido')--Accediendo al Carrusel de Datos
arquivo:seek('set')
linea1=arquivo:read()
arquivo1= assert(io.open('/archivo.txt','r'),'Archivo no leido')
local y=1
local lineas={}
for linea in arquivo1:lines() do
    -- Se lee la información del Carrusel de datos
    lineas[y]=linea
    y = y + 1
end
-- Envío de datos de captura al servidor
tcp.send('Historial/'..lineas[2]..'/'..linea1..'/fin%')
result = tcp.receive("*a")
tcp.disconnect()
```

Figura 26: Implementación de captura de información del canal sintonizado

3.7. Scripts Lua para el envío de datos hacia el Servidor

Para lograr la comunicación cliente-servidor y servidor-cliente se hace uso de clase TCP. Esta clase fue desarrollada en código Lua por la PUC Rio (ver en [5]) y se especifica para el uso del canal de retorno. La misma se la puede descargar en http://tvd.lifia.info.unlp.edu.ar, a la cual se la incluye dentro de la App Ginga-NCLua.

En la Figura 27, se puede observar la secuencia de la estructura para la comunicación entre un cliente y el servidor. Mediante “tcp.execute()” se está preparando una función para realizar conexiones TCP a un servidor. Para conectarse al servidor basta con especificar la IP y el puerto por donde será atendida nuestra petición, en la Figura 26 se especifica la IP: 172.16.146.127 y el puerto: 9999. La conexión se la realiza mediante un “tcp.conect()”. Una vez conectado el envío de datos se da mediante “tcp.send”, mientras que la recepción de datos se realiza mediante “tcp.receive()”. La aplicación podrá estar enviando y recibiendo información mientras la función execute no se haya finalizado. Otra opción para finalizar la conexión es el uso de “tcp.disconect()”.

```
tcp.execute(
    function ()
        -- conexión con el servidor
        tcp.connect('172.16.146.127', 9999)
        local result
                --Envio de información
                tcp.send('Login/tvd1158@hotmail.com/1158/fin%')
                --Recepción de información
                result = tcp.receive("*a")

                if result then
                else
                      result = 'error: ' .. evt.error
                end
        -- Fin de conexión
        tcp.disconnect()
    end
)
```

Figura 27: Implementación de la clase TCP Lua

3.8. Aplicación de finalizar sesión

La aplicación para finalizar sesión o APP LOG OUT junto con la aplicación de recomendación forman parte del menú 2 descrito en el diagrama de la sección 3.1.2. Esto se da debido a que existe un registro en donde el usuario se encuentra con una sesión activa y por ende puede hacer uso de estas interfaces. El menú presentado se ve en la Figura 28.

Figura 28: Interfaz del menú 2 recomendación y cierre de sesión

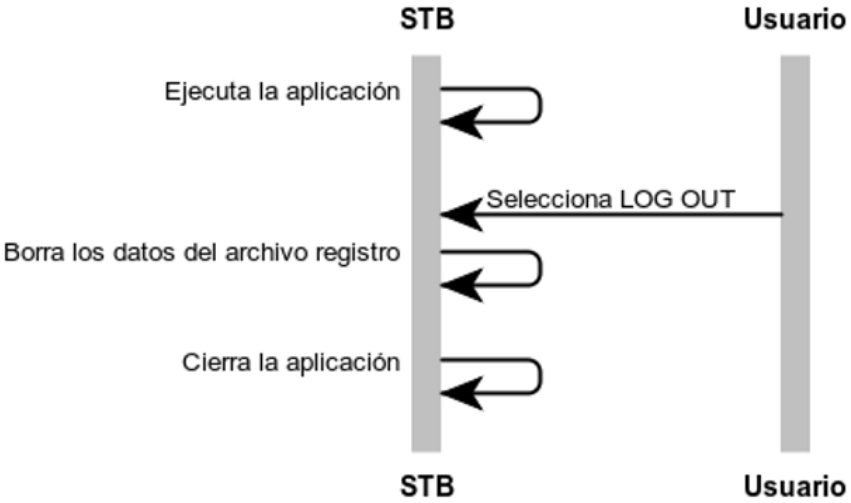


Figura 29: Diagrama de secuencia para APP LOG OUT

Esta aplicación presenta una interfaz más sencilla que las anteriores. Y su diagrama de funcionamiento se ve en la Figura 29. El punto de inicio se da cuando el usuario envía una solicitud de cierre de sesión al seleccionar LOG

OUT (ver Figura 28). Con esto la aplicación borra los datos de los usuarios almacenados en el archivo de registro y niega los permisos para que posteriormente la App pueda capturar información o enviar solicitudes al servidor.

```
arquivo= assert(io.open('/archivo.txt','w'),'Archivo no creado')
arquivo:write('deslogueado')
arquivo:flush()
io.close(arquivo)
event.timer(4000, function()
    local evt = {
        class = 'ncl',
        type  = 'presentation',
        action  = 'stop'
    }
     event.post(evt)
end)
```

Figura 30: Implementación de APP LOG OUT

La implementación de la aplicación se muestra en la Figura 30. Como se explicó anteriormente se deben borrar los datos del archivo de registro "archivo.txt". Para esto simplemente se sobrescribe la información agregando la palabra "deslogueado". Esta palabra sirve como referencia ya que cuando se quiera acceder a los datos, la aplicación lee el archivo y de inmediato niega cualquier petición debido a que no existe una sesión activa. También se debe especificar que "event.timer" brinda un tiempo de espera antes de cerrar la aplicación (en este caso cuatro segundos, posterior al cierre de sesión).

La aplicación no se volverá a ejecutar automáticamente y para acceder a ella el usuario deberá ingresar al repositorio del STB para realizar esta tarea, en caso que desee volver a iniciar sesión en el sistema.

3.9. Aplicaciones para la presentación de recomendación al usuario

La aplicación que proyecta la recomendación al usuario o APP RECOMENDACIÓN es la herramienta mediante la cual el usuario puede acceder a visualizar los contenidos televisivos que el sistema recomienda de acuerdo a su perfil. Como sabemos forma parte del menú 2 (ver Figura 28), ya que para tener una recomendación se debe iniciar sesión de esta forma el servidor puede saber a qué usuario brindar la recomendación.

El diagrama de procedimientos se muestra en la Figura 31. Al iniciar la secuencia se puede ver como se validan los datos del registro que sirven para identificar al televidente entre los otros usuarios. Posterior a esto existen dos puntos importantes a considerar:

- Generar la recomendación para el usuario: El servidor es el encargado de armar un flujo de datos ordenado que contenga la fecha donde existen programas recomendados, los canales que se

recomiendan, los programas que se recomienda y el horario en el cual se transmiten dichos programas.

- Generar una guía de programación: Una vez recibidos los datos del servidor la aplicación se encarga de armar una guía de programación. Esta será visualizada por el televidente y se presentará con una interfaz entendible para el usuario.

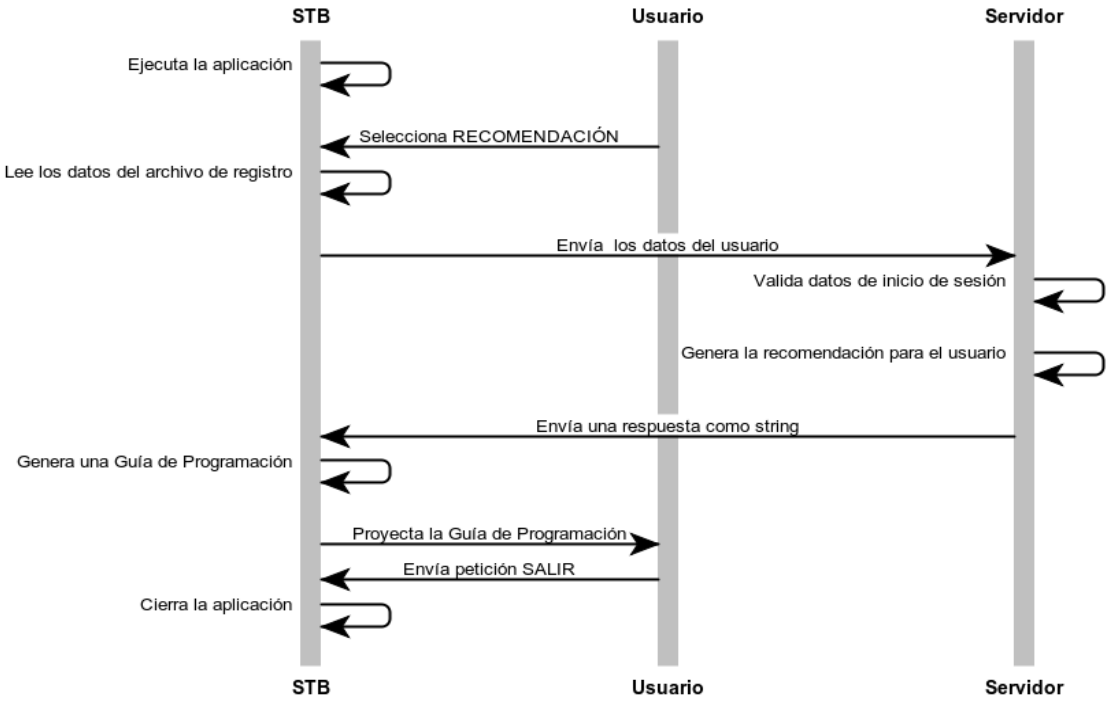


Figura 31: Diagrama de secuencia para APP RECOMENDACIÓN

La implementación de esta aplicación se ve en la Figura 32. Se observa cómo se arma la solicitud de recomendación mediante la cabecera “Recomendacion/”. A ésta se le adiciona la información del usuario que la solicita enviada en “lineas[2]”. Posteriormente se recibe una respuesta por parte del servidor cargada en “result”. Para armar la guía de programación se usa una secuencia “ for, do, end” con la misma se busca dentro de la respuesta del servidor la información de cada día de programación y se la carga en el vector “dias[]”. Para poder diferenciar los días del resto de información el servidor envía la codificación “000” al inicio y “111” al final. Posterior a esto, por cada día se lee los canales, sobre cada canal se lee la programación y se la ordena en un horario; a todos estos datos se los busca dentro de una secuencia de inicio y fin como se hizo con los días.

```
tcp.connect('172.16.146.127', 9999)
 local result
 canvas:attrFont('vera', dys/3)
 canvas:attrColor('white')
 canvas:drawText(10,10,'Conectando Servidor')
 canvas:flush()
local mensaje='Recomendacion/'..lineas[2]..'/fin%'--Solicitud de recomendación
tcp.send(mensaje)
result = tcp.receive("*a")    --Recepción de resultados
local line = ""
 if result then
     line=result
 end
 local indicedias=1
 --Armado de Guía de Programación
 for dia in string.gmatch(line,"000(.-)111") do
         dias[indicedias]=dia
         indicedias=indicedias+1
 end
 numerodias=indicedias-1
 leercanales()
 dibujarrow()
 cargarcanales()
 dibujarcuadros()
 dibujarcolum()
 mostrarcontenido()
```

Figura 32: Implementación de APP RECOMENDACIÓN

En la Figura 33 se visualiza una simulación de la interfaz de usuario para la APP RECOMENDACIÓN. Se puede ver como la guía de programación específica la fecha y un horario. La aplicación también ordena los canales y su contenido televisivo. Este contenido representa los gustos de cada usuario y es diferente para cada televidente. Además presenta un menú de navegación dentro de la guía de programación, con lo cual el usuario podrá visualizar los horarios y finalmente salir de la App.

Considerando el ejemplo de la Figura 33, vemos como el sistema recomienda al usuario ver las Noticias en Ecuador TV (canal 7) el día 14 de abril del 2014 a las 13 horas. Los datos presentados en esta guía de programación son generados por el sistema recomendador y representan una predicción de los programas que se pueden ver en un día y hora particulares, para un usuario específico.

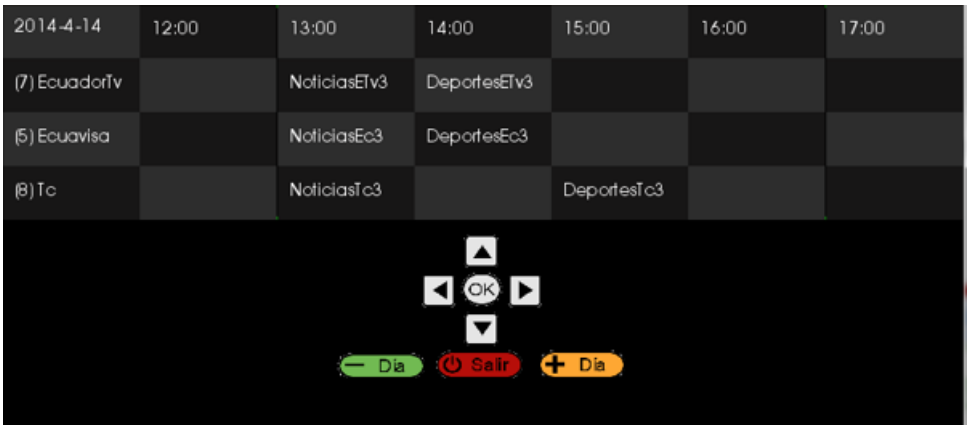


Figura 33: Interfaz de usuario para la APP RECOMENDACIÓN

RESUMEN DE CAPÍTULO

En este capítulo se abordó el desarrollo de una aplicación interactiva, cuya finalidad principal es aportar como un medio para capturar información que describe la actividad que genera un televidente al interactuar con su receptor. Primero se mostró las funciones a cumplir por la aplicación y a partir de estas funciones se planteó un diagrama de flujo y diagramas de secuencia a seguir. Se mostró un conjunto de aplicativos y scripts de programación generados de acuerdo a una necesidad específica. Finalmente se mostró la funcionalidad de cada script dentro de la aplicación con algunos ejemplos que mostraron como el usuario puede ejecutar y utilizar los servicios de la aplicación.

DESARROLLO DEL SERVIDOR DE APLICACIONES INTERACTIVAS Y ALMACENAMIENTO DE INFORMACIÓN CAPTURADA.

El capítulo 4, desarrollo del servidor de aplicaciones y almacenamiento de información capturada, tiene como finalidad abordar los temas concernientes a la estructura y funcionamiento del gestor de aplicaciones y el servidor de base de datos. En principio se explicará la estructura implementada en el servidor de aplicaciones, posteriormente se detallará cada aspecto correspondiente al desarrollo del servidor TCP, la generación de la base de datos y el uso del protocolo TCP para la comunicación entre la aplicación y los servidores. Luego se plantea una estructura para minimizar el fallo referente a la integración de las aplicaciones y servidor con los otros módulos del proyecto. Finalmente se realizará un manual donde se indique el uso de las aplicaciones generadas.

CAPÍTULO 4

DESARROLLO DEL SERVIDOR DE APLICACIONES INTERACTIVAS Y ALMACENAMIENTO DE INFORMACIÓN CAPTURADA.

4.1. Descripción del Servidor de Aplicaciones Interactivas.

Un servidor de aplicaciones proporciona servicios que soportan la ejecución, disponibilidad y procesamiento de datos de una aplicación, solicitados por un cliente. De cierta manera es el corazón de un sistema distribuido, ya que provee de múltiples servicios y responde los requerimientos de un cliente.

4.1.1. Funciones que desempeña el Servidor de Aplicaciones.

La función que desempeña el servidor de aplicaciones es administrar los recursos del servidor y gestionar las aplicaciones. Entre sus funciones está la de mantener el ciclo de vida de la aplicación que se está ejecutando en el STB del usuario, así como también la de sostener la interacción del intercambio de información entre el STB y los servidores (de aplicaciones y base de datos).

De manera más específica el servidor de aplicaciones se encarga de recibir las solicitudes de los clientes, procesar e identificar el proceso a ejecutar. La aplicación posee varios procesos, entre estos están: ingreso de cliente, registro de cliente, captura de actividad del cliente y proveer recomendación de la programación televisiva de acuerdo al perfil del cliente. Después que se haya ejecutado un proceso, el servidor prepara las respuestas y las envía al cliente que realizó la solicitud. Existen procesos en los cuales el servidor de aplicaciones necesita interactuar con el servidor de base de datos para consultar y almacenar información específica.

4.1.2. Diagrama de flujo

En la Figura 34, se muestra un diagrama de flujo de la forma en que funciona el servidor de aplicaciones. En la figura se observa que el servidor espera las conexiones de los clientes, una vez que se establece la conexión con alguno de ellos, se recibe la solicitud que el cliente envía, luego el servidor procesa la función a ejecutar entre las cuales se encuentran: *ingreso/login de un usuario existente en el sistema, registro de un nuevo usuario al sistema, almacenamiento del historial de la actividad del usuario y recomendación de programación televisiva de acuerdo al perfil del cliente*, seguidamente se ejecuta el proceso de la función solicitada y como último el servidor envía la respuesta.

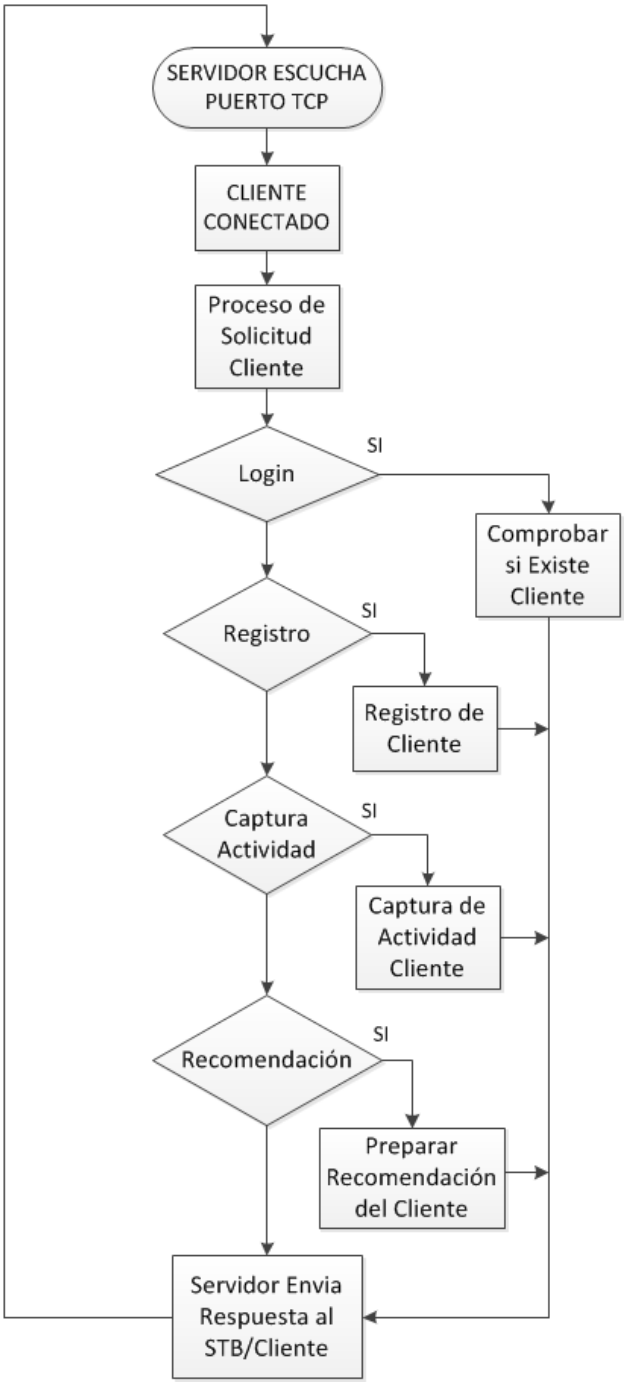


Figura 34: Diagrama de flujo del Servidor de Aplicaciones del proyecto

4.2. Estructura del Servidor de Aplicaciones

Como se mencionó en la sección 4.1, un servidor de aplicaciones proporciona servicios que soportan la ejecución, disponibilidad y procesamiento de datos, para que el servidor funcione de forma correcta es necesario que disponga de una estructura de arquitectura adecuada.

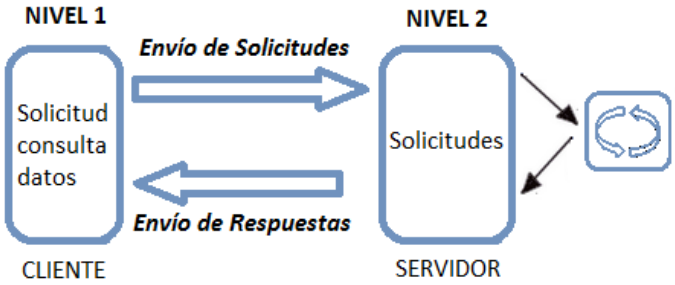


Figura 35: Modelo de 2 niveles de un Servidor de Aplicaciones

Para la estructura del servidor de aplicaciones se utilizará un modelo Cliente-Servidor de dos niveles (ver Figura 35). Esta estructura se utiliza para implementar sistemas en donde el cliente solicita recursos al servidor y éste haciendo uso de sus recursos responde directamente a la solicitud. Esta estructura se utilizará en la etapa del servidor de aplicaciones, ya que al incorporar todo el sistema junto con el servidor de base de datos, se tendrá un modelo de tres niveles, explicado posteriormente.

4.3. Comunicación con la aplicación y uso de protocolos TCP

El protocolo TCP (Protocolo de Control de Transmisión) es uno de los principales protocolos de la capa de transporte del modelo TCP/IP - OSI, este es un protocolo estándar de transmisión de datos que recibe una confirmación de recepción y permite una comunicación fiable entre el emisor y receptor. TCP se utiliza típicamente sobre el Protocolo de Internet.

Para la comunicación del servidor de aplicaciones y de datos con la aplicación ejecutada en el STB se hará uso del protocolo TCP. El servidor de aplicaciones se implementará haciendo uso de Sockets TCP, utilizando la clase java.net.ServerSocket que provee el lenguaje Java, ésta proporciona un mecanismo que permite al servidor escuchar a los clientes y establecer las conexiones con ellos. El servidor implementado escucha un puerto específico y crea un socket de conexión cuando un cliente realiza una solicitud al servidor, mediante este socket el cliente y el servidor se comunican. Para explicar de una mejor manera, un socket es una tubería entre el proceso cliente que realizó la petición y el proceso servidor que lo atiende. A través de un socket TCP ambos procesos pueden enviar y recibir información mediante flujos de bytes.

Del mismo modo, la aplicación desarrollada en el lenguaje Ginga-NCL/LUA que se ejecutará en el STB del cliente, posee la Librería TCP, ésta librería permite realizar el proceso de conexión y desconexión a algún servidor remoto y es utilizada para la transmisión de datos utilizando el canal de retorno del STB.

4.4. Desarrollo del Servidor TCP

La función del servidor es de escuchar y aceptar peticiones de conexión de los clientes. Como se mencionó anteriormente el cliente crea un socket en el extremo de la comunicación e intenta conectar ese socket a un servidor,

cuando se realiza la conexión, el servidor crea un objeto socket en su extremo y se establece la conexión entre el cliente y el servidor. Para que el servidor pueda realizar esta tarea, lo primero es establecer un modelo para su funcionamiento. En la Figura 36 se puede observar la conexión entre el cliente y el servidor.

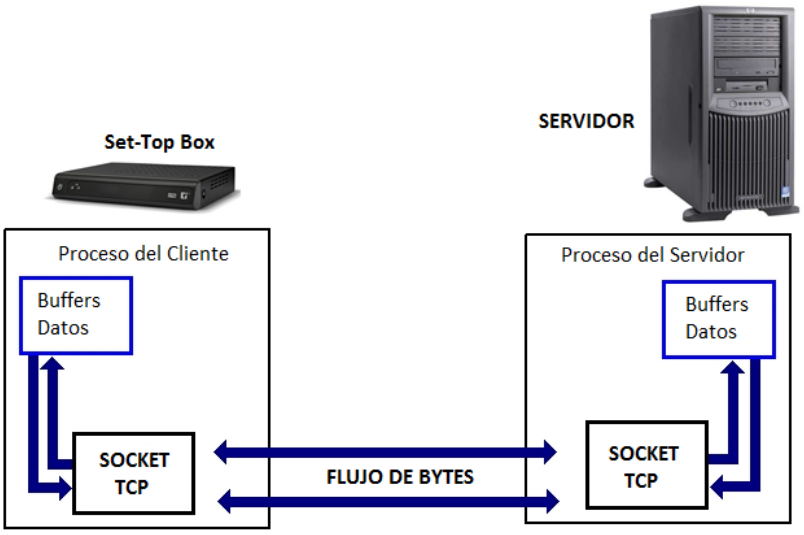


Figura 36: Conexión mediante sockets entre el usuario y servidor

Las conexiones entre dispositivos electrónicos usualmente siguen el modelo llamado Cliente-Servidor, el cliente es un proceso en algún dispositivo que realiza peticiones a otra máquina externa en la red, de la cual necesita cierta información para operar. El servidor es un proceso en un dispositivo de la red que está continuamente esperando peticiones de clientes.

Un socket TCP sirve para establecer la comunicación entre el cliente y el servidor. Los siguientes pasos son ejecutados cuando se establece una conexión entre el dispositivo cliente y el servidor usando un socket:

- El servidor instancia un objeto ServerSocket, denotando el número de puerto habilitado y dirección IP para que el servidor escuche solicitudes de conexión.

- Posteriormente el servidor invocará un método para aceptar nuevas conexiones de la clase ServerSocket, el cual está testeando el puerto en espera de conexiones.

- Como el servidor está a la espera de conexiones, un cliente puede crear una instancia de un objeto Socket, especificando el nombre del servidor (dirección IP) y número de puerto al que requiere conectarse.

- El constructor de la clase Socket intenta conectar el cliente al servidor especificado. Si se logra establecer la comunicación, el cliente dispone de un objeto Socket capaz de comunicarse con el servidor.

- Una vez establecidas las conexiones, la comunicación puede producirse usando flujos de Entrada/Salida. Cada socket tiene un InputStream (flujo de entrada) y un OutputStream (flujo de salida).

Para el manejo de la concurrencia de usuarios que solicitan conectarse al servidor se hace mediante el manejo de hilos (Threads), al momento que el servidor espera conexiones se encuentra en un bucle infinito, cuando hay una conexión se ejecuta un método donde se crea un hilo para atender a ese cliente, una vez que la solicitud de ese cliente es procesada y respondida, el servidor cierra ese hilo para administrar de mejor forma recursos del servidor, de esta forma se mejorará la eficiencia al dar respuesta a varios clientes que requieran conectarse. El servidor puede atender a la vez a varios clientes creando un hilo para cada uno. Una característica importante del procesamiento de los hilos en el servidor, es poder obtener paralelismo si se dispone de una arquitectura multiprocesador.

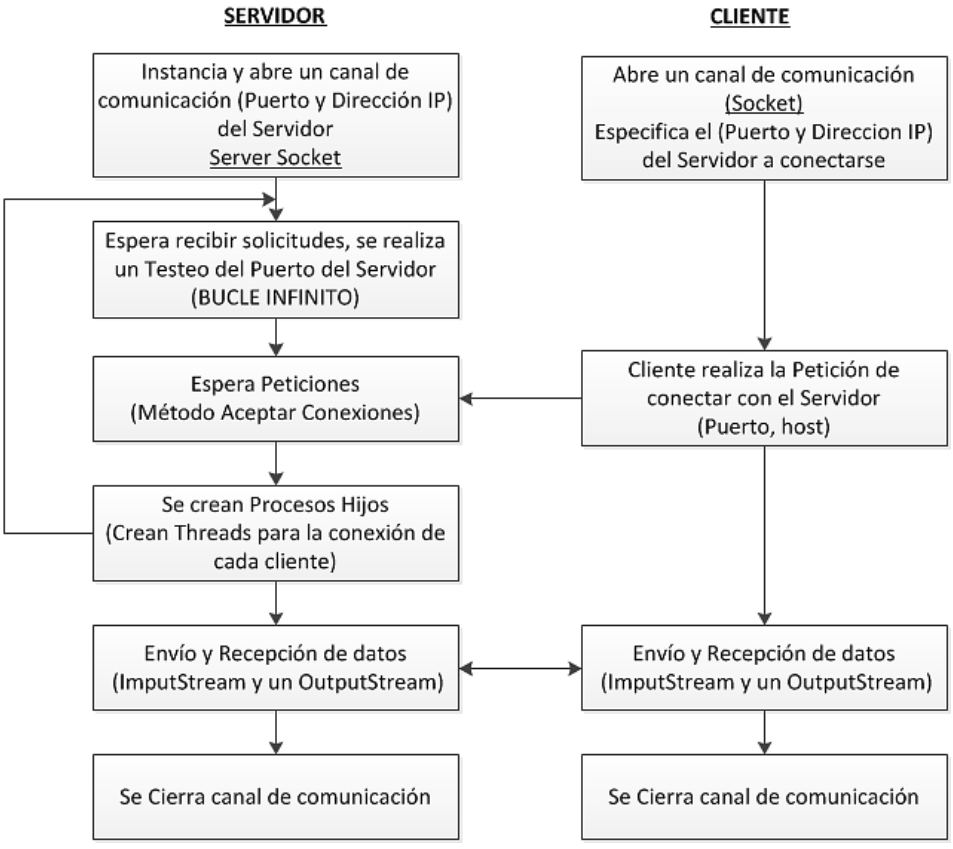


Figura 37: Ciclo de vida de la conexión de un cliente al servidor

El ciclo de vida de la conexión del cliente al servidor haciendo uso de los hilos, se observa en la Figura 37. La imagen describe los procesos que realizan el

servidor y el usuario al momento de establecer una conexión. Los bloques 2, 3 y 4 del servidor, muestra los pasos que se realizan para la creación de hilos de las conexiones de los clientes que requieren conectarse al servidor simultáneamente. Así mismo en el bloque 5 del servidor (ver Figura 37) se puede visualizar que después de establecerse la conexión entre el hilo del servidor con el cliente, se realiza el proceso de enviar y recibir información y seguidamente se termina la conexión, cerrando el canal de comunicación, con lo cual se logra incrementar la eficiencia del servidor.

4.5. Generación de Bases de Datos

Los datos intercambiados mediante un socket entre el cliente y servidor se deberán almacenar o consultar en una base de datos, esta información es importante para el funcionamiento del componente *extracción del perfil del usuario* y sobre todo para el resto de componentes del proyecto como *anotación semántica de las guías de programación electrónica, enriquecimiento de las guías de programación con recursos externos y enriquecimiento del modelo semántico* [1].

Para la generación del servidor de base de datos es necesario contar con una estructura de tres niveles, en la Figura 35 se indicó que un servidor de aplicaciones está conformado por una estructura de dos niveles, pero para que el sistema *Extracción del Perfil de Usuario* funcione adecuadamente, es necesario disponer de un nivel adicional, esta estructura está conformada por el cliente, servidor de aplicaciones y el servidor de base de datos, es necesario esta estructura para que se realice la interacción entre el cliente, el servidor de aplicaciones y el servidor de datos, esta estructura es la adecuada para que el sistema funcione correctamente.

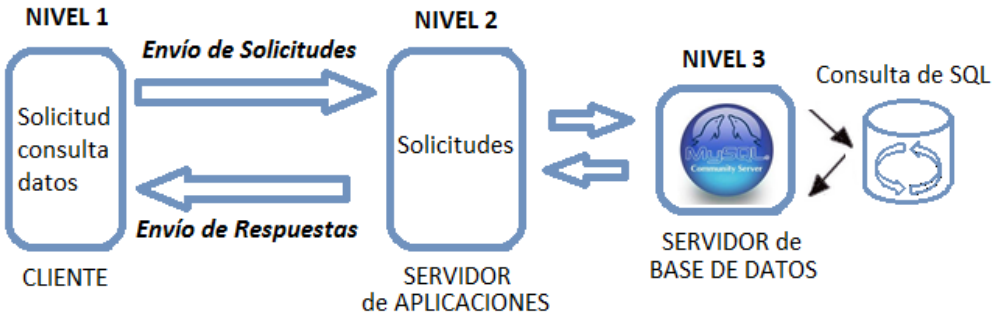


Figura 38: Modelo de 3 niveles para un Servidor de Aplicaciones y Base de Datos

La estructura a implementar en el componente de *extracción del perfil del Usuario* se puede observar en la Figura 38, la cual está dispuesta por tres niveles. El nivel 1 está conformado por el cliente, quien realiza peticiones al servidor de aplicaciones. El nivel 2, compuesto por el servidor de aplicaciones (denominado también *software intermedio*), cuya tarea es proporcionar los recursos solicitados por las peticiones del cliente, pero que requiere de otro

servidor para realizarlo. Nivel 3, conformado por el servidor de datos, proporciona al servidor de aplicaciones la información que requiere.

En el nivel 3, la base de datos soporta el almacenamiento y lectura de datos de forma rápida y estructurada. El modelo de administración de la base de datos que se utilizará en el sistema es el de base de datos relacionales, ya que permite administrar los datos dinámicamente. Para recuperar o almacenar la información se puede realizar mediante consultas, ya que éstas ofrecen una amplia flexibilidad y dominio para administrar la información.

Las consultas al servidor de base de datos se implementarán mediante el lenguaje SQL “Structured Query Language” o “Lenguaje Estructurado de Consultas”, es el lenguaje más habitual para construir las consultas en bases de datos relacionales.

Para implementar la base de datos, primero se estableció un diagrama de las tablas que van a estar relacionadas entre sí, así la información estará ordenada de manera lógica y coherente. La base de datos contiene 3 tablas, que cumplen la función de contener los campos.

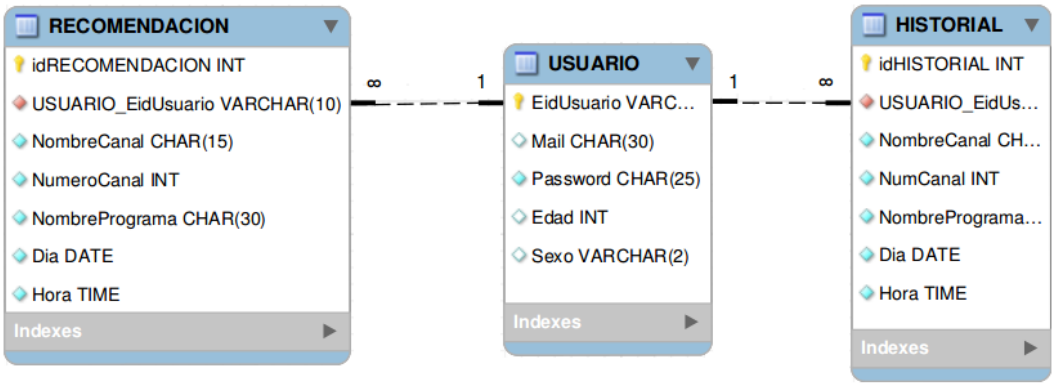


Figura 39: Diagrama de tablas de Servidor de datos

La Figura 39 muestra el diagrama entidad relación que se implementará en el servidor de datos. Este modelo expresa entidades relevantes para el sistema de información, sus inter-relaciones y propiedades con las que están conformadas.

4.5.1. Registro de usuarios

Una de las tablas donde se almacenará la información de los usuarios, es la tabla usuario, la misma que dispone de cinco campos, el campo *EidUsuario* se identifica como Clave Primaria. Dentro del esquema de la base de datos es la tabla principal, puesto que la clave primaria de esta tabla pertenece a las otras tablas como un campo que se identifica como clave foránea, permitiendo almacenar y leer información de cada usuario en las diferentes tablas evitando posibles errores.

El servidor de aplicación después de establecer la conexión con un usuario, procesa que petición el usuario requiere, aquí se analiza si la petición es el de permitir al usuario el ingreso (login) al sistema o de registrar un nuevo usuario a la base de datos.

```
   ###############
 ### FECHA: 2014-04-12
 ### HORA: 11:38:49
Cliente conectado
Ip Cliente : 172.16.146.66

 > > > > SERVIDOR RECIBE DEL CLIENTE:  Registro/tvd5110/5110/21/M/fin


Usuario Ingresado Correctamente
 > > RESPUESTA SERVER: RegistroCorrecto%
```

Figura 40: Respuesta del Servidor de Aplicaciones ante la petición de ingreso al Sistema.

Para el caso de registro de usuarios en la base de datos, en la Figura 40 se observa el proceso que el servidor de aplicaciones muestra en pantalla al momento que se registra un nuevo usuario al sistema. Para el proceso de *registro* el cliente necesita enviar un *id de usuario, contraseña, la edad y sexo del usuario,* toda esta información se envía en una cadena de caracteres donde cada campo es separado por el carácter "*/*". Al procesar la cadena de caracteres en el servidor de aplicaciones, se comprueba mediante una consulta SQL a la base de datos si el *id de usuario* a ingresar ya existe en el sistema, en caso de no existir, toda la información enviada en esta cadena es agregada a la base de datos. En aquellos casos en donde todo es correcto el servidor envía una respuesta al cliente de *RegistroCorrecto*, si el *id de usuario* a ingresar existe ya en la base de datos, el servidor enviará la respuesta de *EidUsuarioExiste*.

```
   ###############
 ### FECHA: 2014-04-12
 ### HORA: 11:11:44
Cliente conectado
Ip Cliente : 172.16.146.66

 > > > > SERVIDOR RECIBE DEL CLIENTE:  Login/tvd1100/1100/fin

El Usuario EXISTE en la base de datos
 > > RESPUESTA SERVER: UsuarioCorrecto%
```

Figura 41: Respuesta del Servidor de Aplicaciones ante la petición de ingreso al Sistema.

En la Figura 41 se observa el proceso de respuesta que entrega el servidor de aplicaciones al momento de recibir una petición del usuario, en este caso, la petición que solicita el usuario al sistema es de *ingreso*. Para el ingreso el cliente necesita enviar el id de Usuario y la contraseña. El cliente envía una

cadena de caracteres donde cada palabra es separada por el carácter “/” de la misma forma que se indicó en el registro. Una vez que se inicia el procesamiento de la cadena de caracteres en el servidor de aplicaciones, se comprueba mediante una consulta SQL a la base de datos, si el id de usuario y la contraseña son correctas, y si toda la información es validada como correcta, el servidor envía una respuesta al cliente de UsuarioCorrecto, si la información es errónea el servidor enviará la respuesta de UsuarioIncorrecto.

4.5.2. Almacenamiento de Bases de Datos de actividades del usuario

Otra de las tablas que pertenece a la base de datos es la tabla historial. Esta tabla almacenará la actividad que realiza el usuario en la TV. Esta tabla dispone de siete campos, el campo EidUsuario se identifica como clave foránea, la misma ayuda a identificar los datos que pertenecen a un cliente existente y que está registrado en la tabla usuario. Esta tabla contiene además una campo idHistorial identificado como clave primaria de tipo entero y además es de valor autoincremental.

```
   ###############
 ### FECHA: 2014-04-12
 ### HORA: 11:11:46
Cliente conectado
Ip Cliente : 172.16.146.66

 > > > > SERVIDOR RECIBE DEL CLIENTE:  Historial/tvd1100/CANAL UCUENCA 2/8/Demo Helicoptero
/fin

Hoy: 2014-04-12
hora: 11:11:46
Historial Ingresado Correctamente
 > > RESPUESTA SERVER: HistorialGuardado%
```

Figura 42: Respuesta del Servidor de Aplicaciones ante la petición de ingreso al Sistema.

La Figura 42 ilustra la respuesta que entrega el servidor de aplicaciones al momento de recibir una petición de almacenar el historial de actividades del cliente, la petición que solicita el usuario al sistema es de *historial*. Para almacenar la actividad el cliente necesita enviar el id de usuario, nombre del canal, numero del canal, y el título del programa. Como se indicó anteriormente el cliente envía la cadena de caracteres donde cada palabra es separada por el carácter “/”. Una vez que se inicia el procesamiento de la cadena de caracteres en el servidor de aplicaciones, no es necesario comprobar si el id de usuario es correcto, ya que esta comprobación se realiza al momento que el usuario ingresa al sistema. Si el proceso se realiza correctamente la información es almacenada en la base de datos mediante una consulta SQL, luego de esto el servidor envía una respuesta al cliente de HistorialGuardado, si el proceso falla el servidor enviará ErrorHistorial.

#	idHISTORIAL	USUARIO_EidUsuario	NombreCanal	NumCanal	NombrePrograma	Dia	Hora
1	1	tvd1100	Ecuavisa	5	Noticias	2014-04-11	08:12:23
2	2	tvd1310	Tc	8	Deportes	2014-04-11	08:11:22
3	3	tvd1251	Ecuavisa	5	En Contacto	2014-04-11	09:50:35
4	4	tvd1310	Teleamazonas	11	TeleNovelas	2014-04-11	10:35:39
5	5	tvd1100	Ecuavisa	5	Noticias	2014-04-11	13:35:12
6	6	tvd1251	Tc	8	Noticias	2014-04-11	13:47:16
7	7	tvd1100	Teleamazonas	11	Noticias	2014-04-11	13:55:11
8	8	tvd1310	EcuadorTv	7	Noticias	2014-04-11	14:06:15
9	9	tvd1100	Teleamazonas	11	Deportes	2014-04-11	14:27:11
10	10	tvd1100	Universidad Cuenca	8	Demo Helicoptero	2014-05-02	10:08:41

Figura 43: Tabla de historial de actividad de los usuarios.

Una de las funciones importantes que cumple el servidor de aplicaciones, se produce al momento de almacenar información de la actividad del usuario. En este caso el servidor de aplicaciones se encarga de agregar información extra a la información enviada por el cliente. Como se observa en la Figura 43, el servidor adiciona los campos de fecha y hora a esta información, estos campos son importantes ya que al momento de procesar estos datos en el resto de sistemas del proyecto, necesitan conocer la hora y fecha en la cual la información es registrada, con esto pueden conseguir una periodicidad de la actividad del usuario, así como obtener el tiempo de visualización de cierta programación.

4.6. Planteamiento de la integración de la aplicación, Servidor Aplicaciones y Base de Datos con los otros módulos del proyecto.

Para el planteamiento de la integración de la aplicación, servidor de aplicaciones y base de datos con los otros módulos del proyecto, se establecen dos soluciones, las cuales se exponen a continuación.

Primera solución

Para un correcto acoplamiento del sistema de *Extracción del Perfil del Usuario* con el resto de módulos del proyecto, es tener a disposición dos bases de datos, una para el módulo de *Extracción del Perfil del Usuario* y la otra para el uso común del resto de módulos del proyecto donde se almacenará la información. La particularidad que existe en esta solución, es que la información almacenada en una de las base de datos tiene que reflejarse inmediatamente en la otra, de cierta manera se tendría un respaldo de la información, pero con la particularidad de necesitar almacenamiento en cada módulo del proyecto. En la Figura 44 se puede observar el esquema planteado para la primera solución. Sin embargo, una de las dificultades que se tendría en este modelo, es la generación excesiva de tráfico de información que se enviaría entre las dos bases de datos, ya que la información se debe estar actualizando periódicamente en las dos partes. Además en esta solución no se dispondría de un manejo centralizado de todo el sistema recomendador.

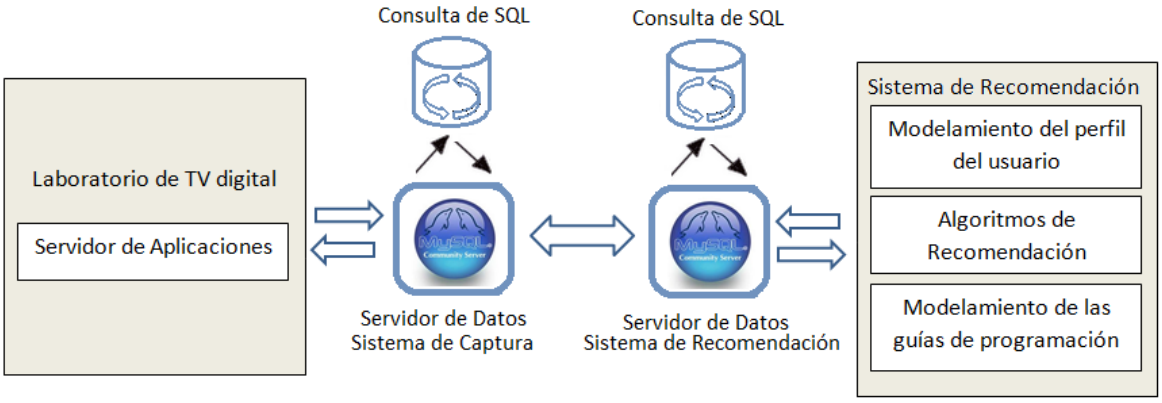


Figura 44: Esquema del proyecto con una Base de Datos general única.

Cabe recalcar que los campos a utilizarse en las tablas de las bases de datos, deben tener un mismo esquema, los datos de las tablas deben coincidir en tipo y dimensión para evitar errores al momento de almacenar y leer información por cualquier modulo del proyecto.

Segunda solución

Como otra solución para que los distintos módulos del proyecto se puedan acoplar de una manera óptima y eficiente, se puede disponer de una base de datos única y general, a la cual se puede tener acceso por todos los módulos del sistema, en la que se podrá almacenar y leer información. El esquema de esta base se puede observar en la Figura 45.

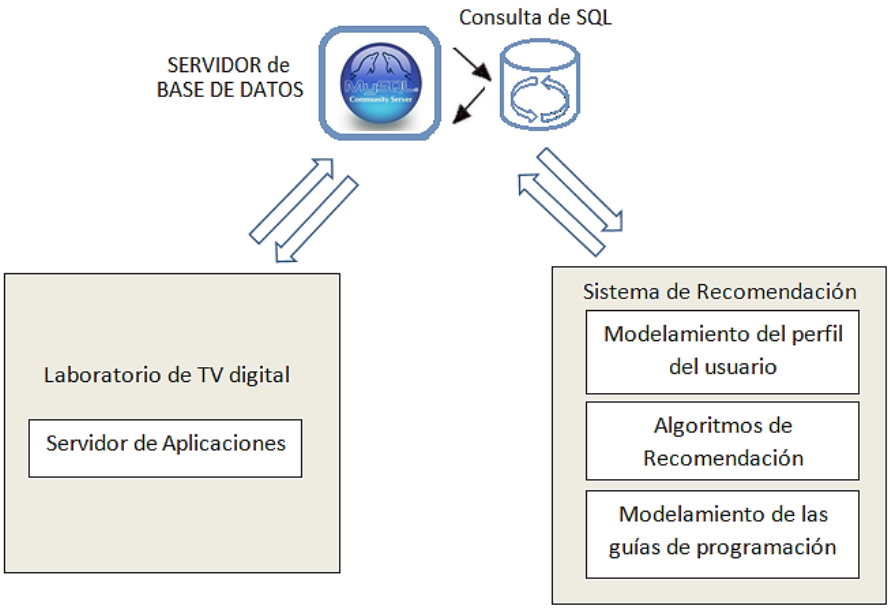


Figura 45: Esquema del proyecto con una Base de Datos general única.

La información almacenada por el sistema recomendador puede ser utilizada por el resto de módulos del proyecto con tan solo realizar una consulta de lectura o escritura de esos datos. Sin embargo, una de las dificultades que se tendrá en este modelo, es la saturación referente a las consultas de lectura y escritura de información que varios clientes y administradores del sistema puedan solicitar al servidor simultáneamente. La ventaja que presta este modelo, es la administración centralizada de los servicios.

Solución recomendada

Una vez expuestas estas dos soluciones, se recomienda optar por la segunda opción, con la administración de solo un servidor de base de datos se tendría un manejo centralizado de la información que requiere el proyecto. De este modo se puede acceder y almacenar de forma rápida la información que el sistema recomendador y el resto de módulos del proyecto almacenen en la base de datos puesto que el acceso es único.

4.7. Generación de un manual para el uso de las aplicaciones.

A continuación se presenta un manual para la instalación y uso de los servidores de aplicación y de datos, con el objetivo de que trabajos futuros puedan hacer uso correcto de los aplicativos generados en el presente proyecto.

La programación del servidor de aplicaciones se realizó en el lenguaje java, para ello se utilizó el entorno de desarrollo NetBeans IDE 7.3. Para ejecutar el proyecto en el software NetBeans, primero se debe abrir el archivo, buscando la carpeta donde se encuentre almacenado.

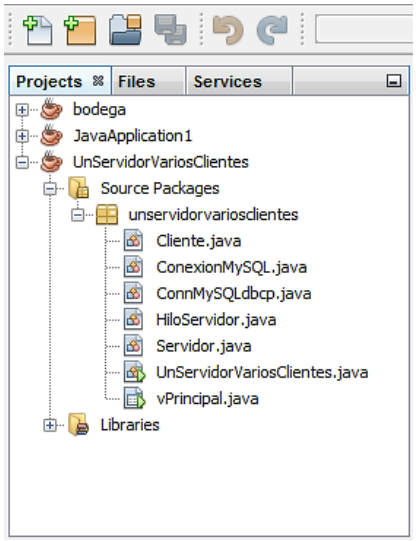


Figura 46: Visualización de los proyectos en NetBeans.

En la Figura 46 se puede observar la localización del proyecto UnServidorVariosClientes, el cual contiene el programa del Servidor de Aplicaciones.

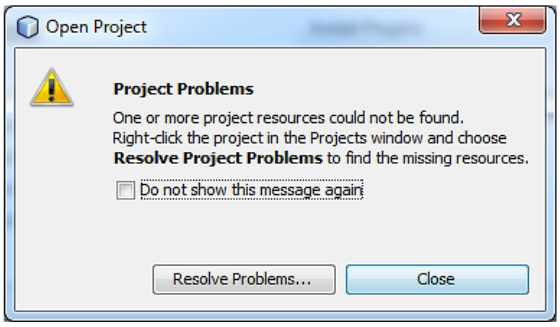


Figura 47: Ventana de problemas al abrir el proyecto del Servidor de Aplicaciones.

Al momento de abrir el proyecto se mostrará un mensaje de Problemas, tal como se puede observar en la Figura 47. Esto ocurre debido a que el proyecto no puede encontrar unas librerías que son necesarias para la ejecución del servidor de aplicaciones.

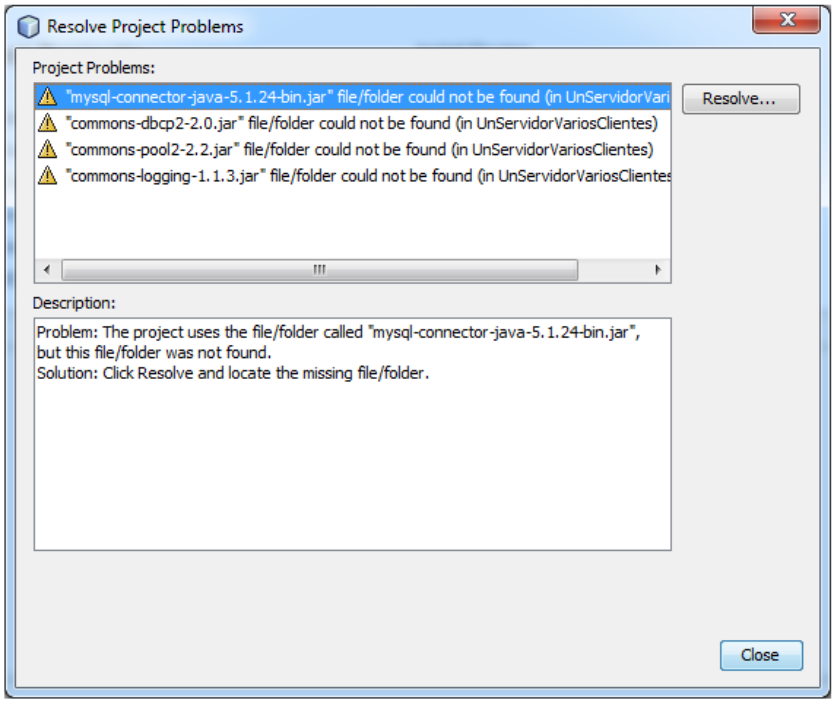


Figura 48: Ventana para solucionar problemas de librerías.

La ventana que se visualiza para solucionar estos problemas se muestra en la Figura 48. Para corregir estos problemas es necesario agregar de forma

manual estas librerías que están localizadas dentro de la carpeta que contiene el proyecto.

Al seleccionar cada una de estas librerías y al dar clic en el botón Resolver, procedemos a buscar de forma manual cada una de estas, por ejemplo dentro de estas se encuentra la librería “mysql-connector-java-5.1.24-bin.jar”, ésta es la librería que sirve para realizar la conexión a la base de datos provista por el programa MySQL, buscamos de forma manual esta librería y la agregamos, tal como se observa en la Figura 49.

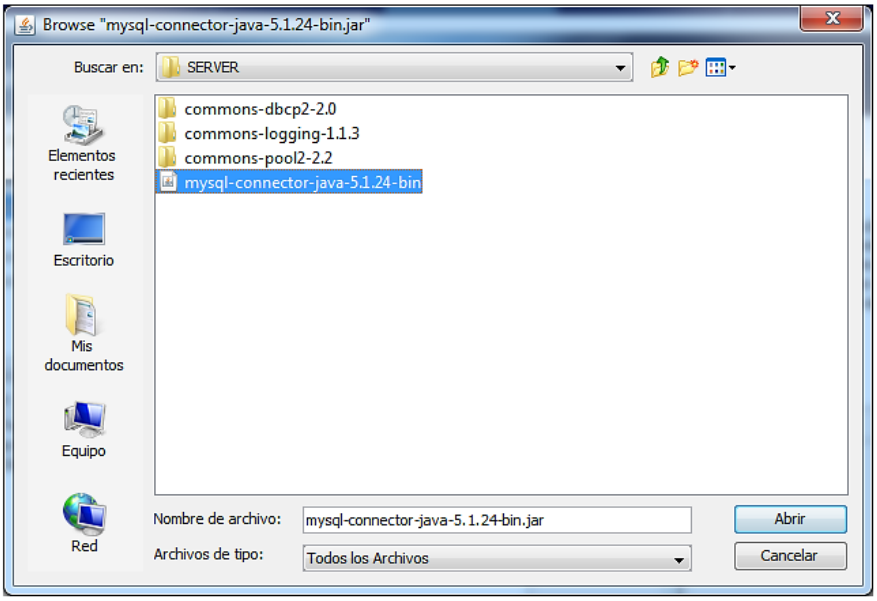


Figura 49: Búsqueda de la librería “mysql-connector-java-5.1.24-bin.jar”.

Cada una del resto de las librerías “commons-pool2-2.2.jar”, “commons-dbcp2-2.0.jar” y “commons-logging-1.1.3.jar” debe ser agregada con el mismo procedimiento indicado anteriormente.

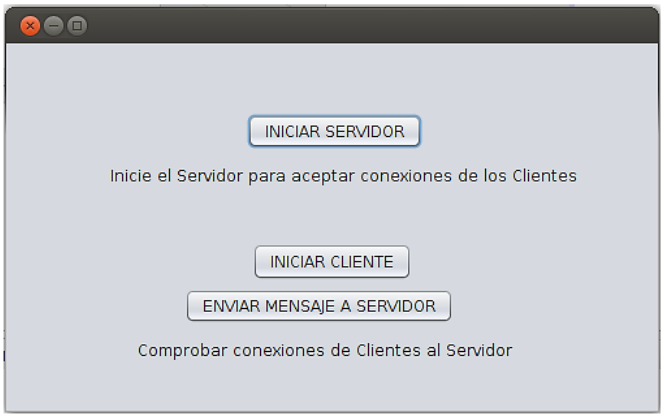


Figura 50: Ventana principal de ejecución de Servidor.

Una vez resuelto los problemas, se procede a ejecutar el proyecto. La Figura 50 muestra la ventana del servidor al momento de su ejecución. Al dar clic en el botón de Iniciar Servidor, estamos ejecutando el servidor, el cual crea un objeto socket en su extremo de la comunicación y espera conexiones de los clientes en el puerto establecido.

La ventana principal del servidor también contiene otros botones como el de Iniciar Cliente y el de Enviar Mensaje a Servidor, estos botones realizan la simulación de la conexión de un cliente al servidor para testear que el funcionamiento del servidor sea correcto.

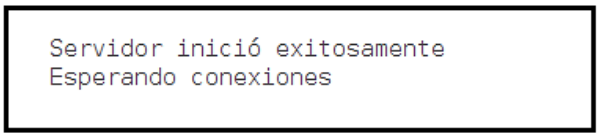


Figura 51: Mensaje al Iniciar el Servidor de Aplicaciones.

Una vez puesto en ejecución el servidor, nos muestra un mensaje en la ventana de consola tal como se observa en la Figura 51, luego inicia el proceso de testeo del puerto a la espera de conexiones.

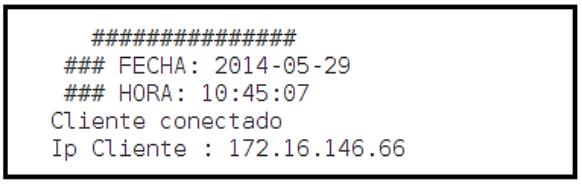


Figura 52: Propiedades que visualiza el Servidor cuando un cliente se ha conectado.

Al momento que el servidor procesa una conexión de un cliente, puede visualizar algunas propiedades de dicha conexión, entre estas están las que muestran en la Figura 52, en esta se visualiza la hora y fecha que el cliente se conectó, además de obtener su dirección IP.

Para conectar la base de datos, se crea la conexión especificando la dirección del servidor, el número de puerto, el nombre de usuario y la contraseña. Una vez creada la conexión, se realiza la conexión al servidor, lo cual se visualiza la ventana mostrada en la Figura 53.

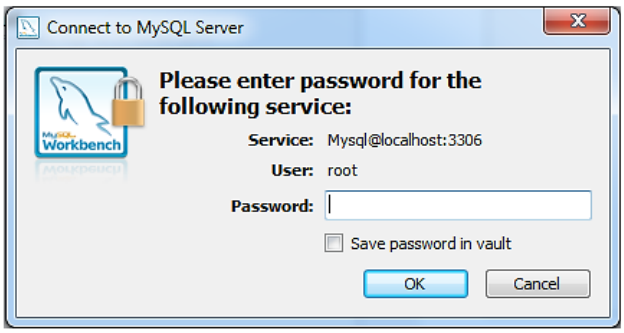


Figura 53: Ventana de conexión a la Base de Datos.

Una vez realizada la conexión a la base de datos Figura 54, se tiene acceso a las tablas y por ende a los datos que están almacenados.

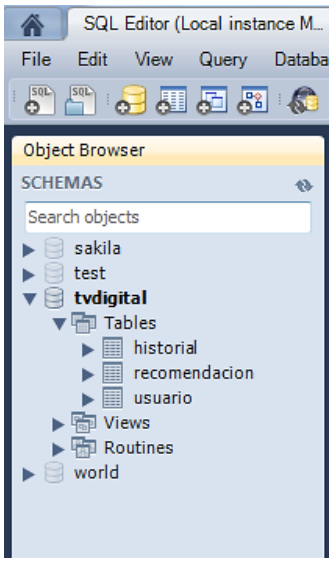


Figura 54: Conexión a la Base de Datos.

RESUMEN DE CAPÍTULO

El capítulo 4, abordó el desarrollo e implementación del servidor de aplicaciones y del servidor de base de datos los cuales darán un correcto funcionamiento al sistema recomendador. Se trató temas concernientes a la estructura implementada en el sistema, la misma que sirvió en la interacción entre los servidores y clientes. Además se trató de detallar el desarrollo del servidor TCP, la generación e integración de la base de datos al sistema y el uso del protocolo TCP para la comunicación entre la aplicación interactiva y los servidores. Posteriormente se planteó dos esquemas para el acoplamiento del sistema “Extracción del Perfil de Usuario” con el resto sistemas del proyecto, los cuales tienen como propósito minimizar el fallo del acoplamiento. Finalmente se desarrolló un manual para la instalación y uso de los servidores, el cual puede ser utilizado como guía para hacer uso correcto de los aplicativos descritos en este capítulo.

ANÁLISIS DE RESULTADOS, CONCLUSIONES Y RECOMENDACIONES.

El capítulo 5, análisis de resultados, conclusiones y recomendaciones, contiene el análisis de los resultados que se obtuvieron de las pruebas realizadas en la etapa de implementación del laboratorio, así como también se encuentran las conclusiones obtenidas al final del proyecto de tesis. Luego se darán algunas recomendaciones y sugerencias de trabajos futuros que se obtienen a partir del proceso de desarrollo de este trabajo.

CAPÍTULO 5

ANÁLISIS DE RESULTADOS, CONCLUSIONES Y RECOMENDACIONES.

5.1. Pruebas de funcionamiento y comunicación entre Aplicaciones y Servidor

En esta sección se identifican las pruebas que se realizaron y los resultados obtenidos. Los resultados se mostrarán mediante imágenes reales que permiten mostrar la interacción entre la aplicación y el servidor.

5.1.1. Descripción de las pruebas a realizar.

La descripción de las pruebas a realizar tiene el propósito de asegurar que todos los dispositivos o medios utilizados en el funcionamiento de la aplicación interactiva y de los servidores sean correctos. En esta sección se plantea los criterios que serán evaluados en la sección 5.1.2.

- Flujo de transporte.
    - El contenido es generado correctamente, referente a la parte de codificación y sincronización.
    - La existencia de información necesaria, para que posteriormente sea visualizada y capturada por la aplicación interactiva.
    - No existe errores en la transmisión del contenido.
    - La simulación de la transmisión simultánea de contenidos de los dos Broadcasters implementados en el laboratorio.

- Aplicación Interactiva.
    - La visualización de la aplicación en una posición apropiada en la pantalla del televisor.
    - Los textos e imágenes contienen una escala adecuada con respecto al tamaño de la pantalla.
    - Los mensajes y otros textos de la aplicación estén libres de faltas ortográficas o tipográficas.

- Servidores.
    - El servidor de aplicaciones recepta las peticiones realizadas por los clientes, procesa y devuelve resultados.
    - Las consultas SQL realizadas por el servidor de aplicaciones al servidor de datos funcionan correctamente y no contienen errores.
    - El proceso de almacenar o extraer información, realizado por el servidor de aplicaciones al servidor de datos funciona de acuerdo a las necesidades de la aplicación interactiva.

5.1.2. Pruebas funcionales.

El propósito de las pruebas funcionales, es asegurar que todos los criterios mencionados en la sección 5.1.1 se cumplan satisfactoriamente. Para ello, se efectuaron las siguientes pruebas funcionales:

- Transmisión y recepción de un Transport Stream (TS).
- Registro de un nuevo usuario al sistema.
- Ingreso al sistema de un usuario ya existente.
- Almacenamiento del historial de la actividad del usuario con la simulación de dos Broadcasters para tener la opción de cambiar de canal y almacenar la información de la actividad.
- Prueba de recomendación de programación televisiva con la cual la aplicación interactiva visualiza todo el contenido que le podría interesar al usuario de acuerdo a su perfil.

Transmisión y recepción de un Transport Stream

En la Figura 55 se observa el proceso de administración del contenido radiado, el cual se logra mediante el uso del software StreamXpress. El software permite realizar la administración del contenido a radiar.

Figura 55: Administración del contenido a radiar mediante el programa StreamXpress

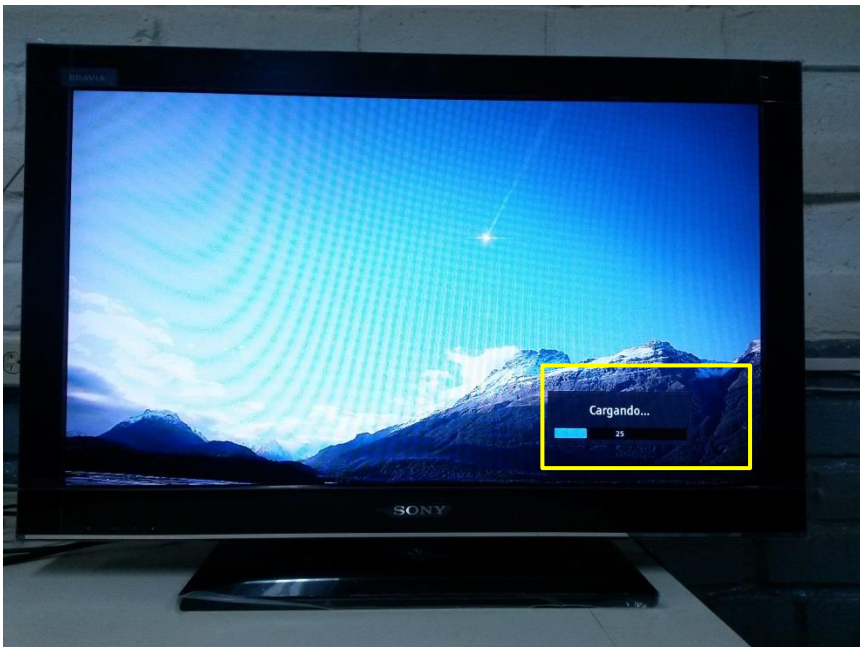


Figura 56: Recepción del contenido radiado, proceso de carga de la Aplicación Interactiva.

Por otra parte, en la Figura 56 y Figura 57, se puede apreciar el proceso de recepción del contenido por parte del televidente, debido a que en el TS se encuentra embebida la aplicación interactiva, el momento que el destinatario recepte el flujo, se inicia automáticamente el proceso de carga de la aplicación. El proceso se inicia nuevamente si el cliente cambia de canal.

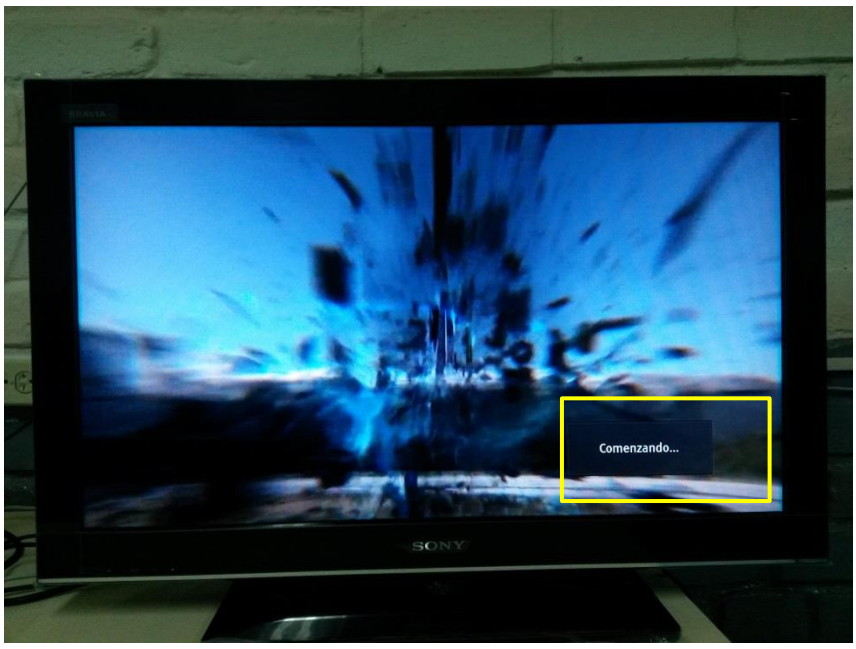


Figura 57: Proceso completo de carga de la Aplicación Interactiva.

Una vez inicializada la aplicación interactiva, se visualiza un botón con el texto “MENÚ”, tal como se observa en la Figura 58, para ingresar al menú presionamos el botón “OK” del control remoto.

Figura 58: Proceso de Inicio de Ejecución de la Aplicación

En la Figura 59, se puede constatar la información del canal que la función de almacenamiento de la actividad del usuario requiere. Además se puedo verificar que el contenido es generado sin errores y se comprueba que la transmisión se realiza de forma correcta.

Con los resultados de esta prueba, se logra cumplir los criterios establecidos referentes al flujo de transporte en la sección 5.1.1.

Figura 59: Información del canal

Registro de un nuevo usuario al sistema

Para el proceso de registro de un nuevo usuario al sistema, la Figura 60 nos muestra un menú de las funciones de la aplicación que están disponibles, se observa esta interfaz debido a que la aplicación detecta que el usuario no está registrado en el sistema.

Figura 60: Menú de la funciones disponibles de la Aplicación Interactiva para un nuevo usuario

Una vez que el nuevo usuario haya seleccionado el botón “REGÍSTRATE” de la Figura 60, la aplicación muestra una nueva interfaz en la que se deberá ingresar los datos del usuario tal como se observa en la Figura 61, una vez ingresado los datos en los campos solicitados el televidente podrá oprimir el botón verde del control remoto para guardar los datos. La información es enviada al servidor de aplicaciones, el cual procesa la solicitud requerida, para este caso la función es de registro, posteriormente el servidor comprueba en la base de datos que el nombre de usuario no exista ya en el sistema y envía una respuesta. Si el proceso es realizado con éxito se muestra el texto de respuesta “Registro completado con éxito”, tal como se observa en el recuadro resaltado en amarillo de la Figura 61, caso contrario nos mostrará un mensaje de error. En el ejemplo se utiliza la información usuario: tvd1200, contraseña: 1234, edad: 19, sexo: Femenino.

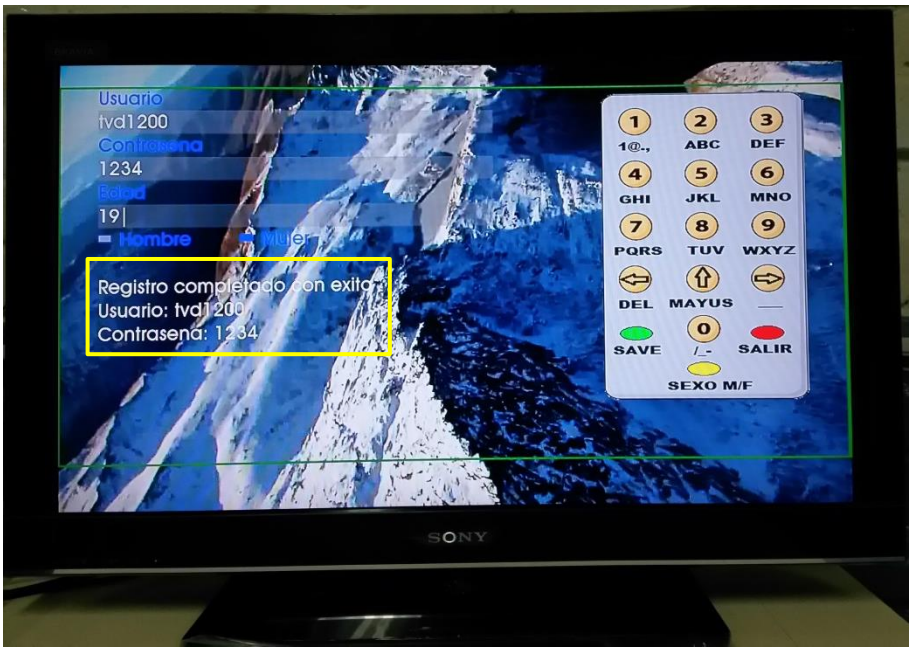


Figura 61: Ingreso de datos para el registro de un usuario

Ingreso al sistema de un usuario ya existente

Para el ingreso de un usuario existente en el sistema, un menú similar al de la Figura 60 aparece en la pantalla, para registrarse el usuario en el sistema debe seleccionar el botón “LOG IN”. Posteriormente la aplicación carga la interfaz de ingreso, tal como se aprecia en la Figura 62.

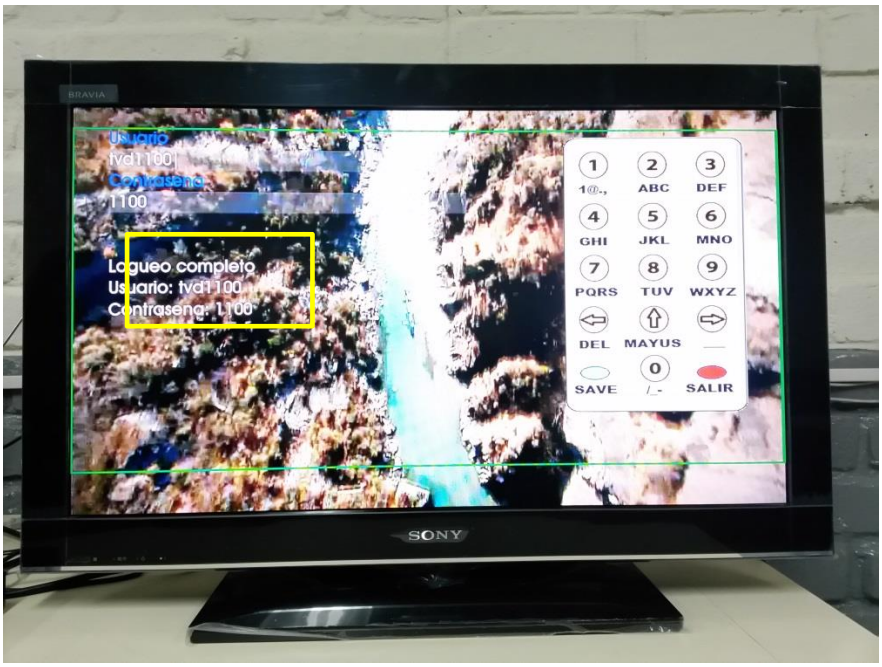


Figura 62: Ingreso de un usuario que existe en el Sistema

Una vez llenado los campos con los datos, la información es enviada al servidor de aplicaciones, el cual consulta a la base de datos la veracidad de los

mismos. Si el proceso se realiza sin errores el usuario ingresa al sistema, la respuesta se puede observar en el recuadro resaltado en amarillo en la Figura 62, en caso de no poder observar este texto existe algún error en los datos ingresados. Para el ingreso de un usuario se hizo uso de los siguientes datos, usuario: tvd1100, contraseña: 1100.

Figura 63: Menú de la funciones disponibles para un usuario que ha ingresado al Sistema

Una vez realizado el proceso de registro o de ingreso de un usuario al sistema, la interfaz de la Figura 63 es la que se muestra por parte de la aplicación interactiva, en la cual se observa un menú con algunas funciones.

Almacenamiento del historial de la actividad del usuario

Para la función de almacenamiento de información de la actividad del usuario que también realiza la aplicación interactiva, no se creyó conveniente visualizar algún mensaje cuando esta operación se haya realizado, puesto que la aplicación es la que realiza la captura de los datos y envía al servidor de aplicaciones, el mismo mediante consultas a la base de datos procede a almacenar dicha información, que posteriormente servirá al resto de módulos del proyecto, para la formación del perfil del televidente.

Prueba de recomendación de programación televisiva

Una última función a comprobar es la recomendación, la cual se puede seleccionar en el menú mostrado en la Figura 63. Una vez realizada la selección de esta función el usuario podrá observar una interfaz en la cual se muestra la programación televisiva de los canales con su respectiva hora, los canales visualizados van de acuerdo al perfil del usuario.

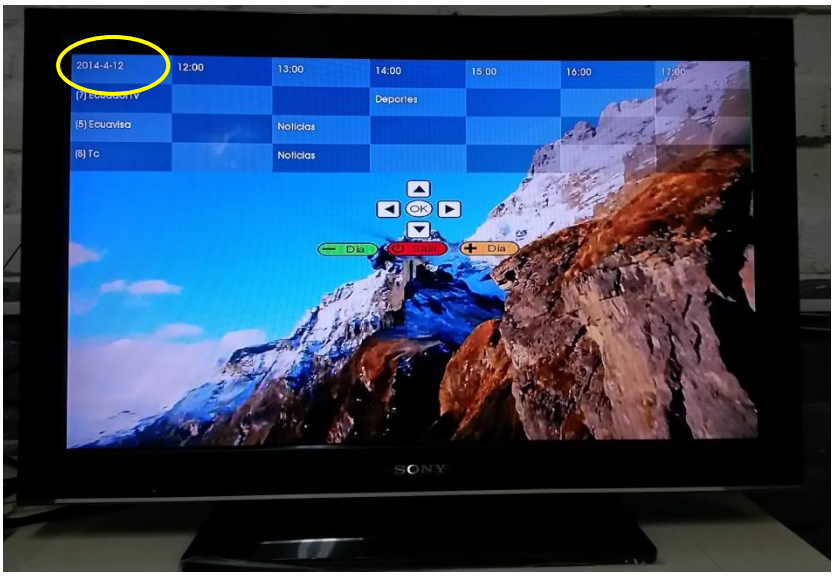


Figura 64: Interfaz de la recomendación de programación mostrada al usuario que corresponde al día 1

En la Figura 64 es posible observar en la esquina superior izquierda la fecha del día en que se realizó la petición programación televisiva, el usuario además puede observar la programación de dos días posteriores a esa fecha, tal como se muestra en la Figura 65 y Figura 66.

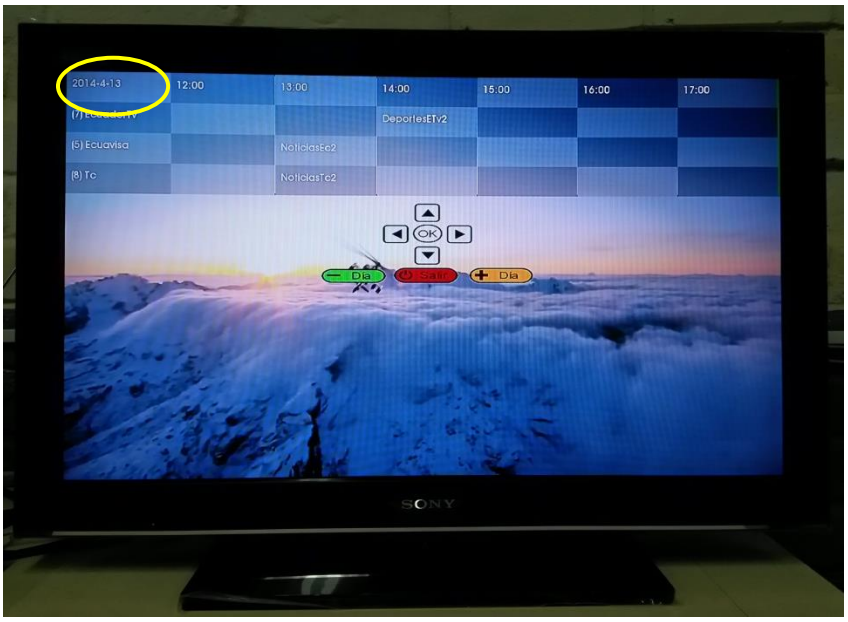


Figura 65: Interfaz de la recomendación de programación mostrada al usuario que corresponde al día 2

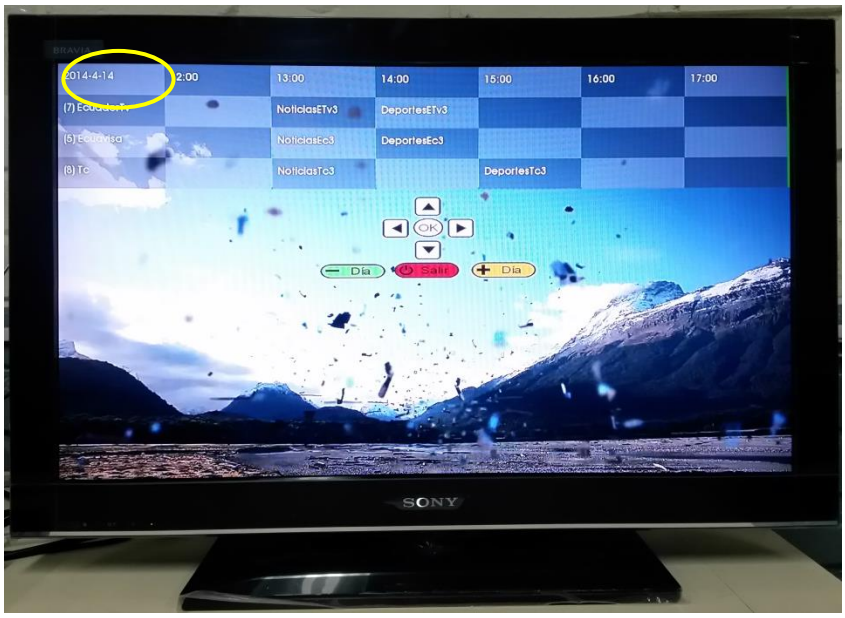


Figura 66: Interfaz de la recomendación de programación mostrada al usuario que corresponde al día 3

5.2. Muestra de resultados en el Servidor de Aplicaciones.

Se comprobó que las consultas SQL de lectura y escritura realizadas por el servidor de aplicaciones al servidor de datos funcionan correctamente. La información enviada de los clientes por la aplicación interactiva, es almacenada correctamente, así como también la información es fácilmente extraída y enviada a los clientes cuando es solicitada.

Los resultados obtenidos y mostrados por el servidor, después de realizar las pruebas funcionales en el apartado anterior, son visualizados a continuación:

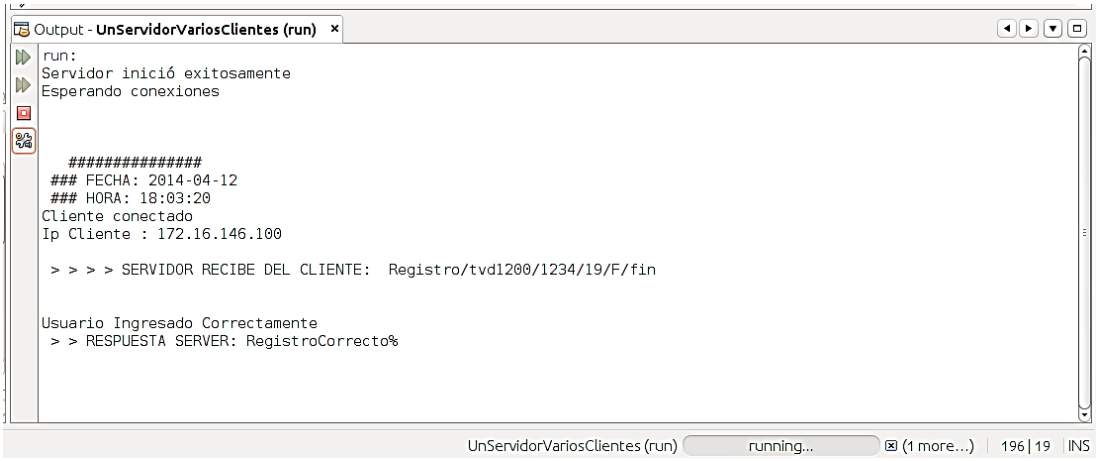


Figura 67: Respuesta del Servidor al realizar la función de registro

El resultado de la prueba de la función de registro de un nuevo usuario al sistema (ver sección 5.1.2), fue correcta y se lo pudo comprobar debido a que el servidor recibe la solicitud enviada por la aplicación interactiva que se ejecuta en el Set-Top Box del cliente. La Figura 67 nos muestra que el servidor recibe la petición de registro, la cual viene acompañada por el nombre de usuario: tvd1200, contraseña: 1234, edad: 19 y sexo: Femenino. Como el proceso realizado no contiene errores, el servidor envía una respuesta al cliente: RegistroCorrecto.

#	EidUsuario	Mail	Password	Edad	Sexo
1	tvd1000	tvd1000@yahoo.com	1000	21	f
2	tvd1100	tvd1100@hotmail.com	1100	18	m
3	tvd1158	tvd1158@hotmail.com	1158	19	f
4	tvd1200	NULL	1234	19	F
5	tvd1251	tvd1251@yahoo.com	1251	20	f
6	tvd1310	tvd1310@hotmail.com	1310	22	m
7	tvd1541	tvd1541@yahoo.com	1541	12	m
8	tvd1615	tvd1615@gmail.com	1615	14	f
9	tvd1865	tvd1865@hotmail.com	1865	35	m
*	NULL	NULL	NULL	NULL	NULL

Figura 68: Tabla registro de la Base de Datos

La manera de comprobar que el proceso de la función registro se haya realizado con éxito, es mostrando la tabla registro de la base de datos, esto se puede observar en la Figura 68. La cual nos muestra la fila de la información agregada a la tabla.

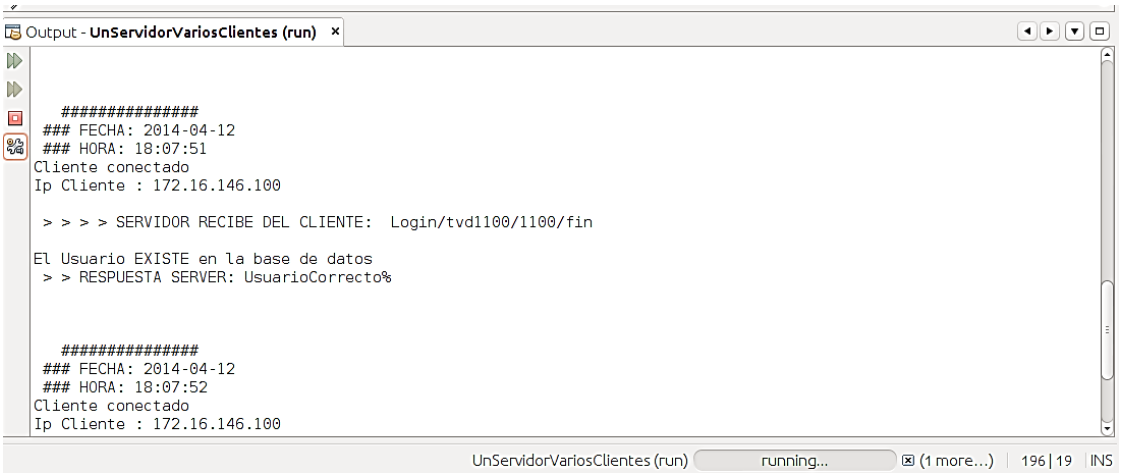


Figura 69: Respuesta del Servidor al realizar la función de ingreso.

El resultado de la prueba de la función de ingreso (login) de un usuario al sistema realizada en el apartado 5.1.2, es correcta debido a que el servidor recibe la solicitud enviada por la aplicación interactiva que se ejecuta en el Set-Top Box del cliente. La Figura 67 nos muestra que el servidor recibe la petición de login, la cual viene acompañada de la información nombre de usuario: tvd1100, contraseña: 1100. Luego de comprobar que el usuario este registrado en el sistema, el servidor envía una respuesta al cliente: UsuarioCorrecto.

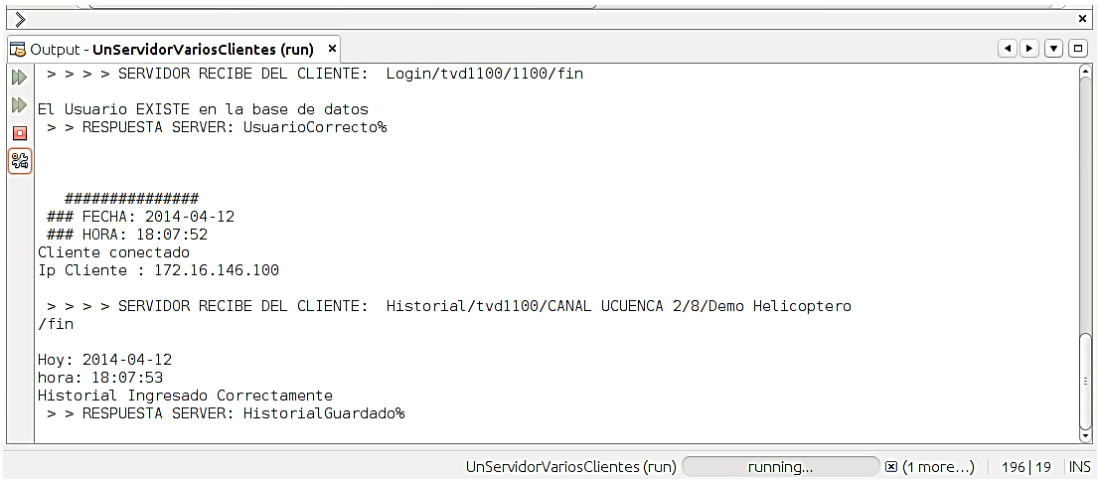


Figura 70: Respuesta del Servidor al realizar la función del almacenamiento del historial del usuario.

De la misma forma que el proceso anterior, el resultado de la prueba de la función que almacena el Historial del cliente en el sistema, el servidor recibe la solicitud enviada por la aplicación interactiva que se ejecuta en el Set-Top Box del cliente. En la Figura 70 se observa que el servidor recibe la petición de historial, la cual es acompañada de la información nombre de usuario: tvd1100, nombre del canal: CANAL UCUENCA 2, número de canal: 8 y el título del programa: Demo Helicóptero. Luego de procesar la solicitud, el servidor envía una respuesta al cliente: HistorialGuardado.

#	idHISTORIAL	USUARIO_EidUsuario	NombreCanal	NumCanal	NombrePrograma	Dia	Hora
109	112	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	17:19:19
110	113	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	17:20:01
111	114	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	17:26:04
112	115	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	11:33:20
113	116	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	11:38:39
114	117	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	11:44:44
115	118	jmjg	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	11:49:08
116	119	jmjg	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	11:49:56
117	121	tvd1200	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	18:05:57
118	122	tvd1100	CANAL UCUENCA 2	8	Demo Helicoptero	2014-04-12	18:07:53
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 71: Tabla historial de la Base de Datos

La forma de comprobar que el proceso de la función que almacena el Historial del cliente en el sistema se haya realizado sin errores, es revisando la tabla historial de la base de datos, esto se puede observar en la Figura 71.

Figura 72: Respuesta del Servidor al realizar la función del recomendación de la programación televisiva al usuario

El resultado de la prueba de la función que provee la recomendación de la programación televisiva de acuerdo al perfil del cliente, se realiza de forma correcta. La Figura 72 nos muestra que el servidor recibe la petición de recomendación que viene acompañada de la información *nombre de usuario: tvd1100*. Luego de procesar la solicitud, el servidor envía como respuesta una cadena de caracteres, la cual es procesada por la aplicación interactiva en el Set-Top Box del cliente y esta posteriormente muestra los resultados.

#	idRECOMENDACION	USUARIO_EidUsuario	NombreCanal	NumeroCanal	NombrePrograma	Dia	Hora
1	2	tvd1100	Teleamazonas	11	Noticias	2014-04-12	08:00:00
2	3	tvd1100	Ecuavisa	5	Noticias	2014-04-12	13:00:00
3	4	tvd1100	Tc	8	Noticias	2014-04-12	13:00:00
4	5	tvd1100	Teleamazonas	11	Noticias	2014-04-12	14:00:00
5	6	tvd1100	EcuadorTv	7	Deportes	2014-04-12	14:00:00
6	7	tvd1100	Teleamazonas	11	Deportes	2014-04-12	15:00:00
7	9	tvd1100	Teleamazonas	11	NoticiasTam2	2014-04-13	08:00:00
8	10	tvd1100	Ecuavisa	5	NoticiasEc2	2014-04-13	13:00:00
9	11	tvd1100	Tc	8	NoticiasTc2	2014-04-13	13:00:00
10	12	tvd1100	EcuadorTv	7	DeportesETv2	2014-04-13	14:00:00
11	13	tvd1100	Teleamazonas	11	NoticiasTa2	2014-04-13	14:00:00
12	14	tvd1100	Teleamazonas	11	DeportesTam2	2014-04-13	15:00:00
13	15	tvd1100	Teleamazonas	11	NoticiasTam3	2014-04-14	08:00:00
14	16	tvd1100	Ecuavisa	5	NoticiasEc3	2014-04-14	13:00:00
15	17	tvd1100	Tc	8	NoticiasTc3	2014-04-14	13:00:00
16	18	tvd1100	EcuadorTv	7	DeportesETv3	2014-04-14	14:00:00
17	19	tvd1100	Teleamazonas	11	NoticiasTam3	2014-04-14	14:00:00
18	20	tvd1100	Tc	8	DeportesTc3	2014-04-14	15:00:00
19	21	tvd1100	EcuadorTv	7	NoticiasETv3	2014-04-14	13:00:00
20	22	tvd1100	Ecuavisa	5	DeportesEc3	2014-04-14	14:00:00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 73: Tabla recomendación de la Base de Datos del usuario 1100

La cadena de caracteres que el servidor envía de respuesta al cliente en la función de recomendación televisiva, está constituida en base a la información existente en la tabla de recomendación correspondiente al *usuario: tvd1100*. La

información de esta tabla se puede observar en la Figura 73. La cadena de caracteres es procesada por la aplicación en el STB del cliente la cual se encarga de estructurar toda esta información para posteriormente visualizar en la interfaz de la programación televisiva. La Figura 64, Figura 65 y Figura 66, son el resultado de este proceso.

En la pantalla del servidor es posible observar el texto de las diferentes solicitudes que envía el cliente, así como también su dirección IP, la hora y fecha en que es realizada. Luego que el servidor procesa la solicitud, es factible observar en la pantalla del servidor que respuesta es enviada al cliente.

5.3. Conclusiones

Al finalizar el proyecto de tesis, afirmamos que el objetivo general como todos los específicos planteados se lograron cumplir, tal como se puede observar en los procesos plasmados a lo largo de cada capítulo.

Con la implementación del laboratorio, la generación del contenido digital, el desarrollo de la aplicación interactiva, así como del servidor de aplicaciones y datos, el objetivo fue cumplido de manera satisfactoria.

A continuación resumimos los principales aportes de este trabajo tesis:

- La implementación basada en software y hardware es una alternativa viable y económica, sin embargo al momento de acoplar estos dos entornos se debe tener en cuenta la instalación y configuración correcta de los drivers para el funcionamiento de los equipos, así como de la correcta instalación del software de gestión y administración.

- El esquema del Broadcaster implementado simula algunas tareas de un canal de televisión, pero para la corrección de problemas eventuales y optimizar la transmisión es necesario dotar al laboratorio con software y equipos que permitan codificar y transmitir el contenido en tiempo real.

- La señal digital transmitida en el laboratorio es visualizada sin inconvenientes en televisores que tienen incorporado un sintonizador digital compatible con el estándar, así como en televisores analógicos que hacen uso de un Set-Top Box. Además se pudo apreciar una correcta ejecución de las aplicaciones interactivas embebidas al contenido digital radiado en cada uno de estos receptores.

- En lo referente a la generación del Transport Stream, con la realización de la aplicación con una interfaz amigable y de fácil uso al usuario se agilita la generación del contenido, ya que la misma codifica, encapsula, empaqueta y sincroniza los datos de manera automática.

- Se pudo comprobar que los mecanismos y herramientas utilizadas en el desarrollo de la aplicación interactiva, el servidor de aplicaciones y datos

funcionan de manera adecuada. La información capturada por la aplicación es enviada adecuadamente al sistema de servidores donde es procesada y almacenada. De igual manera la información requerida por el usuario puede ser transportada hacia él.

- A pesar que el control remoto no es un buen dispositivo de entrada, especialmente en el uso de aplicaciones interactivas, se logró determinar ciertos patrones para que la interacción entre la aplicación interactiva y el usuario sea lo menos tediosa posible.

- En cuanto al canal de retorno, el sistema hace uso de la interactividad remota con este canal, mediante el cual se conecta al servidor de aplicaciones y datos para extraer o almacenar información de los usuarios. Así se pudo comprobar que el canal de retorno permite al televidente ser un actor activo dentro de los servicios que se le están brindando.

- Para las consultas de información, la aplicación ejecutada en el Set-Top Box requiere acceder a la base de datos para extraer o insertar información. Para ejecutar esta tarea se recomienda el uso de la plataforma Java ya que dispone de una variedad de librerías que hacen posible implementar consultas SQL. Estas librerías permiten realizar las tareas necesarias sobre la base de datos y además cuenta con protocolos de comunicación para el uso del canal de retorno.

- Las pruebas de captura de información de la actividad del usuario se ejecutaron en el Set-Top Box EiTV developer Box ISDB-Tb. A pesar que este dispositivo presenta características de un equipo de desarrollo, cuenta con capacidades muy limitadas en cuanto al procesamiento y ejecución de las aplicaciones. Para obtener resultados favorables se ha tenido que adecuar la aplicación para que funcione en equipos de menores características.

- Para que el sistema de captura funcione correctamente, se tiene el servidor de aplicaciones y datos. Estos servidores son utilizados exclusivamente para el intercambio de información cliente-servidor. Teniendo en cuenta esto, se ve la necesidad de contar con una plataforma que no solo maneje la información sino que permita al usuario acceder a un conjunto de aplicaciones, con el fin de independizar la tarea que actualmente realiza el Broadcaster, que es el único medio por el que viajan las aplicaciones.

5.4. Trabajos Futuros

Durante la implementación del laboratorio, del servidor de aplicaciones y el desarrollo de la aplicación interactiva se encontraron temas adicionales en el área de televisión digital que pueden dar solución a ciertos aspectos que no se profundizaron en el estudio del presente proyecto. Los distintos problemas que aparecieron para este proyecto podrían ser solucionados con una investigación adicional de los temas que se mencionan a continuación.

- *Análisis de técnicas y equipos utilizados en Televisión digital.*

Se propone un análisis de técnicas y equipos en el área de TV digital debido a la falta de información en estos temas. Aún no se puede contar con una recomendación en cuanto a que características técnicas deberían tener los equipos de transmisión y recepción de servicios de TV digital. A pesar de que ISDB-Tb cuenta con estándares y recomendaciones para el uso de técnicas y equipos, quedan todavía muchos temas inconclusos. En este trabajo de tesis se presentaron dos problemas puntuales. Primero no se pudo evaluar claramente los equipos utilizados porque no hay una referencia que permita recomendar o no los equipos, evaluando estos equipos según la velocidad de procesamiento, la memoria RAM, la capacidad de almacenamiento, la eficiencia en el funcionamiento y el desenvolvimiento de las aplicaciones. El segundo problema se presentó al utilizar librerías de programación para las aplicaciones, los equipos estaban equipados con el middleware Ginga, pero no todas las aplicaciones fueron compatibles con los mismos. Por ello es necesario una investigación que recomiende que características específicas deben tener los equipos tanto en software y hardware; y a su vez haga una recomendación evaluando su rendimiento.

- *Rediseño del Laboratorio para transmisión de contenidos en tiempo real.*

Un diseño de laboratorio para trabajar en tiempo real es necesario para complementar los servicios de televisión digital dentro del laboratorio. Actualmente no se cuenta con servidores dinámicos de EPG y aplicaciones. Una transmisión en tiempo real permitirá a los Broadcasters generar tablas PSI/SI de contenido que se actualizará durante la transmisión de los servicios de TV digital. Esto hace más fácil la corrección de errores en la transmisión y puede integrar una conexión directa del Broadcaster con servicios brindados por entidades fuera de este entorno.

Esta investigación estará destinada más hacia recomendaciones en el ámbito social. Actualmente en el Ecuador la concesión de frecuencias destinadas al funcionamiento de estaciones de radio y televisión para medios comunitarios se estableció en un 34% (ver Art. 106 en [7]), pero debido a los bajos recursos económicos con los que cuentan estos medios, será complicado instalar plataformas de TDT para una comunidad en general debido al alto costo que esto implica. El laboratorio implementado es una solución de bajo costo y si se adiciona un trabajo investigativo que solucione la transmisión en tiempo real, se puede dar solución al problema presentado para canales comunitarios.

En el rediseño del laboratorio debe constar el incremento de un servidor de contenidos audiovisuales LAN y la corrección de drivers de la tarjeta moduladora para su acoplamiento con el servidor. Esto se puede lograr fácilmente con el uso de software de libre distribución como por ejemplo el VLC Media Player y el SO Linux.

- *Acoplamiento de herramientas de virtualización del STB a hardware de bajo costo*.

Actualmente se cuenta con una herramienta de virtualización del STB la cual se basa en un servidor Linux. Con fines de desarrollo sería importante una investigación que evalué la factibilidad de la instalación de este middleware a un hardware de bajo costo. Se propone la utilización de una microcomputadora por ejemplo Raspberry Pi, cuyo costo es menor comparado a los STB de desarrollo. Este hardware además tiene desarrollados múltiples sistemas operativos basados en Linux con aplicaciones multimedia lo cual facilitaría el acoplamiento al STB virtual. Si la investigación logra demostrar la factibilidad del acoplamiento se puede brindar un receptor de bajo costo y que permite desarrollo tanto en software como en hardware.

- *Gestor de Aplicaciones.*

Este tema se refiere al desarrollo de una aplicación y un servidor que permita el ingreso a un repositorio de aplicaciones desarrolladas en Ginga y la instalación de estas aplicaciones en nuestro receptor. A futuro habrá la necesidad por parte de los desarrolladores de hacer conocer sus aplicaciones al público en general, si existe un servidor que permita esta interacción desarrollador-cliente ayudará a generar oportunidades de inversión y trabajo en el área de TV digital. Por ejemplo en esta tesis la aplicación desarrollada llega al cliente a través del Broadcaster. Con el uso de este servidor propuesto ya no se dependerá del Broadcaster ya que la gestión de descarga, almacenamiento e instalación se llevaría a cabo por el servidor de aplicaciones. Con esto también ayudaría a separar el lazo entre proveedor de contenidos audiovisuales y proveedor de servicios de interacción.

5.5. Recomendaciones

- En nuestro país existe aún un alto desconocimiento sobre el tema relacionado con televisión digital terrestre y las ventajas que presta, por tanto se debería realizar una mayor difusión sobre el tema, por parte de las entidades competentes. Con esto se puede fomentar al desarrollo e investigación de nuevos métodos de implementación de estaciones de televisión digital terrestre para así reducir tiempo y costos de instalación.

- Las entidades de control de la industria y productividad deben brindar parámetros más claros para la adquisición de los equipos para la colectividad. Se cuenta con una normalización que evalúa parámetros muy generales de los receptores, pero muchos de estos parámetros no son relevantes a la hora de evaluar el rendimiento de los equipos.

- Se recomienda estudiar otras funcionalidades que presta el Set-Top Box de desarrollo en cuanto al canal de retorno y su interacción con contenidos interactivos. Un ejemplo de estos servicios es el uso de las redes, por ejemplo implementar servicios web, con la finalidad de optimizar un manejo de recursos del servidor, en cuanto al manejo de conexiones simultáneas de un gran número de usuarios.

- Para futuras versiones del laboratorio de TV digital, se recomienda la adquisición de un servidor que disponga un mayor avance tecnológico que pueda abastecer las siguientes ventajas: codificación y modulación de contenido en tiempo real, compatibilidad de herramientas para la administración, monitoreo y análisis del flujo de transporte TS, que maneje protocolos de comunicación TCP/IP para la administración de las conexiones de los dispositivos de usuario.

- Con la difusión y la creciente demanda de las aplicaciones interactivas para la televisión digital, es recomendable formar un grupo dedicado a la investigación y desarrollo de este tipo de aplicaciones.

- Fomentar el desarrollo de aplicaciones interactivas multimedia educativas, informativas y sociales. Este desarrollo podrá crear un grupo de trabajo que relacione a varias áreas como: psicología, educadoras, diseñadores, ingenieros, etc.

BIBLIOGRAFÍA

[1] Víctor Saquicela and Mauricio Espinoza, "Reduciendo la Sobrecarga de Información en Usuarios de Televisión Digital," *Centro de Investigaci´on en Matemáticas Unidad Zacatecas*, p. 6, 2013.

[2] Arturo Gutiérrez and Miguel Cochancela, Diseño de un Laboratorio de Televisión Digital para la Transmisión de Señales con multiprogramación, contenidos interactivos y guía electrónica de programación, 2013.

[3] EiTV. (2014, Enero) Entretenimiento e Interactividad para Televisión Digital. [Online]. http://www.eitv.com.br/es/solucoes/laboratorio-isdb-t-dtvi

[4] Associação Brasileira de Normas Técnicas, "Multiplexação e serviços de informação (SI)," in *ABNT NBR 15603-1: Televisão digital terrestre*., 2008.

[5] Associação Brasileira de Normas Técnicas, "Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 5: Ginga-NCL para receptores portáteis," in *ABNT NBR 15606-5: Televisão digital terrestre*., 2008.

[6] Francisco Sant' Anna, Carlos Soares, roberto Albuquerque, and Junqueira Simone, "Desenvolvimento de Aplicaçoes Declarativas para TV Digital no Middleware Ginga com Objetos Imperativos NCLua," Curso 2009.

[7] ASAMBLEA NACIONAL DEL ECUADOR, LEY ORGÁNICA DE COMUNICACIÓN, 2013.

[8] LIFIA, "OpenCaster para SATVD-T," UNLP, Buenos Aires, 2011.

[9] Luis Venegas, "GENERACIÓN DE UNA TRAMA BROADCAST TRANSPORT STREAM (BTS) USANDO EL SOFTWARE LIBRE OPENCASTER," PUCP, Lima, Tesis de pregrado 2012.

[10] Pablo Galabay and Freddy Vivar, "MANEJO DEL SOFTWARE GINGA PARA EL DESARROLLO DE APLICACIONES INTERACTIVAS PARA TELEVISIÓN DIGITAL, BASADO EN EL ESTÁNDAR BRASILEÑO ISDB-Tb," Universidad Politècnica Salesiana, Cuenca, Tesis de Pregrado 2012.

[11] Ana Paredes and Maricela Tonguino, "ANÁLISIS DE DESEMPEÑO DEL CANAL DE RETORNO BASADO EN EL DESARROLLO Y TRANSMISIÓN DE APLICACIONES INTERACTIVAS DE TV DIGITAL PARA EL SISTEMA ISDB-Tb," Escuela Politecnica del Ejercito, Quito, Tesis de Pregrado 2012.

[12] Association of Radio Industries and Businesses (ARIB), Caracteristicas del sistema ISDB-T.

[13] Yasuo TAKAHASHI - DiBEG, "Sistema Multiplex e información de Servicio," in *Seminario Técnico de ISDB-T*. Argentina, 2007.

[14] Tatiana Muñoz and Adriana Siguenza, "TV DIGITAL FIJA UTILIZANDO MIDDLEWARE GINGA-NCL APLICADO A UN NOTICIERO DIGITAL," Universidad de Cuenca, Cuenca, Tesis de Pregrado 2012.

[15] Ashish Myles. (2014, Junio) Department of Computer & Information Science & Engineering. [Online]. http://www.cise.ufl.edu/~amyles/tutorials/tcpchat/

[16] Rick Proctor. (2014, Junio) EMBARCADERO DEVELOPER NETWORK. [Online]. http://edn.embarcadero.com/article/31995

[17] Tutorialspoint. Simply Easy Learning. [Online]. http://www.tutorialspoint.com/java/java_networking.htm

[18] Associação Brasileira de Normas Técnicas, "Receptores," in *ABNT NBR 15604-2: Televisão digital terrestre*., 2007.

[19] Associação Brasileira de Normas Técnicas, "Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 2: Ginga-NCL para receptores fijos y móviles," in *ABNT NBR 15606-2: Televisão digital terrestre*. Brasil, 2007, p. 302.

[20] Associação Brasileira de Normas Técnicas, "Multiplexação e serviços de informação (SI) —Parte 3: Sintaxes e definições de informação estendida do SI ," in *ABNT NBR 15603-3: Televisão digital terrestre*., 2007.

[21] Laboratório TeleMídia DI – PUC-Rio , "Part 8 – NCL Digital TV Profiles," in *Nested Context Language 3.0*., 2008, p. 154.

[22] Luiz Fernando Gomes Soares, *Programando en NCL 3.0*. Rio de Janeiro, 2012.

[23] Quezada M, Bernal I Cabezas G, "Sistema de Generación de Aplicaciones Interactivas para TV Digital Para la evaluación de Servicios Masivos," Escuela Politécnica Nacional, Quito, Paper Julio 2013.

[24] P. Vuorimaa C. Peng, "DIGITAL TELEVISION APPLICATION MANAGER," Helsinki University of Technology, Finland, Paper 2001.

[25] Chengyuan Peng, "Digital Television Applications," Helsinki University of Technology, Finland, Documento 2002.

[26] Douglas E. Comer, *Computer Nerworks and Internets*, Fifth Edition ed. New Jersey, United States: Pearson Education, Inc, 2009.

[27] Rick Leander, *Building Application Servers*, 1st ed. New York, USA: SIGS Books, 2000.

GLOSARIO

ABNT: Asociación Brasileña de Normas Técnicas.

APP: Aplicación.

BW: Band Width. Ancho de banda.

CAT: Conditional Access Table. Tabla de acceso condicional.

DIUC: Departamento de Investigación de la Universidad de Cuenca.

EIT: Event Information Table. Tabla de información de evento.

EiTV: Proveedor de equipos de Televisión digital.

EPG: Electronic Program Guide. Guía Electrónica de Programación.

ES: Elementary Stream. Flujo elemental.

HD: High Definition. Alta Definición.

H.264: Estándar de codificación de video HD.

ISDB-T: Integrated Service Digital Broadcasting –Terrestrial.

ISDB-Tb: Integrated System for Digital Broadcast, Terrestrial Brazilian version.

LAN: Local Area Network. Red de Área Local.

LCD: Liquid Cristal Display. Pantalla de cristal líquido.

GINGA: Plataforma para el desarrollo de aplicaciones interactivas.

MPEG: Moving Picture Experts Group.

MP4: Estándar de codificación de video.

MP2: Estándar de comprensión de audio con pérdida.

NCL: Nested Context Language. Lenguaje de programación declarativo.

NIT: Network Information Table. Tabla de información de red.

PAT: Program Association Table. Tabla de asociación del programa.

PES: Packetized Elementary Stream. Paquete de flujo elemental.

PID: Identificador de proceso dentro de un sistema operativo.

PMT: Program Map Table. Tabla del mapa de programa.

PSI: Información Especifica del Programa.

PTS: Presentation Time Stamp. Presentación de marca de fecha.

SDT: Service Description Table. Tabla de descripción de servicio.

SQL: Structured Query Language. Lenguaje estructurado de consultas.

STB: Set Top Box. Decodificador para la señal digital de televisión.

SI: Información del servicio.

TCP: Protocolo de control de transmisión.

TDT: Televisión Digital Terrestre.

TS: Transport Stream. Flujo de transporte.

UHF: Ultra High Frecuency. Frecuencias ultra altas.

VHF: Very High Frecuency. Frecuencias muy altas.

VLC: Video LAN Client. Reproductor de contenidos audiovisuales.

ANEXOS

A.1 Descripción del equipamiento dentro del laboratorio

A continuación se dará una descripción más detallada de los componentes y herramientas utilizadas durante la implementación del laboratorio

Tarjeta Moduladora USB DTU-215 (fuente http://www.dektec.com)

DTU-215

DekTec

USB-2 VHF/UHF Modulator

- No power adapter required
- Fully agile from 36 to 1002MHz
- Channel simulation option

FEATURES

- USB-2 based multi-standard modulator with support for most QAM-, OFDM- and VSB-based modulation standards
- Powered from the USB-2 bus, so no external power adapter is required
- Supports all constellations and modulation modes for each supported standard
- Digital upconversion for excellent signal quality without need for calibration
- Playout of I/Q sample files
- Programmable attenuator
- Free Windows and Linux SDK is fully compatible with other DekTec digital-video output adapters

APPLICATIONS

- Multi-standard test generator
- Demonstrations
- Research and development

KEY ATTRIBUTES

Parameter		Value
RF connector		75-Ω F male
Frequency range		36 .. 1002MHz ±3ppm
Bandwidth (max)		8.0MHz
I/Q sample rate		4.7 .. 9.375MHz
Level	Range	-46 .. -15dBm (QAM) -49 .. -18dBm (OFDM)
	Step size	0.5dB
	Accuracy	±2dB
MER		>40dB
Adjacent channel		-54dB (QAM) -52dB (OFDM)
Phase noise		<-95dBc @ 10kHz
Spectral purity		>50dB (47 .. 1000MHz)
USB port		USB-2
Power (through USB-2)		5V, 500mA
Dimensions (LxWxH)		123 x 62 x 22mm

MODULATION STANDARDS

Modulation	Standard
ADTB-T*/DTMB*	GB 20600-2006
ATSC VSB	ATSC A/53E
CMMB*	GY/T 220.1/2-2006
DVB-C	EN 300 429
DVB-C2*	EN 302 769
DVB-T / DVB-H	EN 300 744
DVB-T2*	EN 302 755
IQ*	Arbitrary I/Q samples
ISDB-T*	ARIB STD-B31
QAM	J.83 Annex A/B/C

* Option

ORDERING INFORMATION

Type	Description
DTU-215-SP	USB-2 VHF/UHF modulator with *StreamXpress*
DTU-215-GOLD	DTU-215 with all options

Please refer to www.dektec.com for the latest pricing and a list of distributors and resellers.

EiTV Developer ISDB-T e IPTV (fuente http://www.eitv.com.br)

CARACTERÍSTICAS TÉCNICAS

Front-End (Tuner + Demodulador):

Frecuencia de Entrada :

UHF: 470MHz(CH14) a 806MHz (CH69)

VHF: 174MHz(CH7) a 216MHz (CH13)

Ancho de Banda: 5.6MHz

Nivel de Señal: -85 dBm la -20 dBm

Señal: Compatible con el sistema ISDB-T

Impedancia de Entrada: 75 Ohms (nominal)

Conexión de entrada/salida: Conector F

Unidad de Procesamiento:

Procesador: STi 7105 (CPU 450 MHz)

Memoria RAM: 256 Mbytes

Memoria Flash: 128 Mbytes

Decodificación de Vídeo:

Estándar: Rec. ITU-T H.264 (MPEG-4 AVC)

Profile: HP@L4.0

Formatos: 480i y 1080i

Frame Rate: 25, 30, 50 y 60MHz

Frecuencia de vídeo: 50 y 60Hz

Decodificación de Audio:

Estándar: ISO/IEC 14496-3 (MPEG-4 AAC) (SBR)

Profile: AAC@L4 y HE-AAC@L4

Interfaces de comunicación:

High Speed USB 2.0

Ethernet – 100 Mbps (RJ45)

Interfaces de salida:

Salida Digital de Audio y Vídeo (HDMI)

Salida de Vídeo Componente (YPbPr)

Salida de Audio Estéreo 1 (D + Y)

Salida de Vídeo Compuesto (CVBS – A/V)

Salida de Audio Estéreo 2 (D + Y)

Salida de Audio Digital (SPDIF coaxial)

CARACTERÍSTICAS ADICIONALES

Interactividad completa (DTVi – Ginga);

Muestra sólo vídeo y audio del canal de difusión;

Carga de aplicativos por aire, vía Internet o red local;

Soporte simultáneo a canales ISDB-T e IPTV (vía UDP y RTP);

Aplicativo gráfico (GUI) vía Web Server para la instalación y configuración de los aplicativos Ginga (DTVi) y canales IPTV;

Ajuste de la imagen en la pantalla del televisor;

Control del bloqueo de canales por edad, con contraseña;

Compatible con los estándares de colores PAL-M y NTSC;

Búsqueda automática de canales;

Guía electrónica de programación (EPG);

Menú en portugués, inglés o español;

Selección de audio.

Televisor LCD Sony KDL-32BX359 (Fuente http://www.sony.es/electronics/tv)

ESPECIFICACIONES TÉCNICAS

Sintonizador (Terrestre)	ISDB-T
Tamaño de la pantalla (medición diagonal)	32″ (80.1 cm)
Pantalla	LCD
Resolución de la pantalla	WXGA
Salida de audio	8W + 8W
Radio FM	SI
Twin Picture	PAP (Fijo), PIP
USB Play (Contenido)	MPEG1 / MPEG2TS / MPEG2PS / MPEG4Visual / MPEG4AVC / H.264 / WMV / Xvid / MP3 / LPCM / AAC / HE-AAC / DolbyDigitalAC3 / WMA / Jpeg
Entradas de video compuesto	2 (1 Lateral / 1 Posterior Hibrido)
Entradas de video componente (Y/Pb/Pr)	1 (Posterior Hibrido Compuesto)
Conexiones HDMI™ (Total)	2 (1 Posterior / 1 Lateral)
Entradas de audio analógicas (Total)	2 (1 Posterior / 1 Lateral)
Salidas de audio digital	1 (Posterior), 1 (Lateral Hibrido)
USB	1 (Lateral)
Conexiones Ethernet	1 (Posterior)

A.2 Diseño e implementación del Laboratorio

A continuación se mostrará como los diferentes módulos de la estructura del laboratorio planteado en la arquitectura han sido implementados. Además se describirá como los equipos están conectados dentro del laboratorio

Equipos del Broadcaster

En el laboratorio implementado, el bloque del Broadcaster es en el cual se almacena y procesa el audio, video, los datos y la EPG, todo lo anteriormente mencionado se multiplexa en un mismo equipo, este computador posee las siguientes características técnicas procesador Core I5 3.1GHz, 2GB de RAM y disco duro de 500GB, así como también cuenta el Sistema Operativo Linux, ya que el software OpenCaster que realiza todas estas funciones fue desarrollado para el SO Linux. En la Figura A1 se muestra el computador que realiza todas estas funciones a la cual se la denomina Broadcaster 1, algunos equipos adicionales también se indican en la Figura A2.

Figura A1: Broadcaster 1

Equipos de Modulación y Transmisión de señal

El laboratorio dispone de dos tarjetas moduladoras (Dektec DTU-215). Estas tarjetas son las encargadas de modular el contenido digital de los (TS) generados. La tarjeta interactúa con el computador mediante el puerto USB 2.0, por lo tanto no se necesita alimentación independiente. Estás tarjeta radia mediante la configuración previa de su propio software llamado StreamXpress, este tiene que ser instalado en el SO Windows.

Para que el contenido digital modulado pueda ser radiado y el mismo llegue sin perdidas en los receptores es necesaria una etapa de amplificación, para esto el laboratorio dispone de dos amplificadores, los cuales dan ganancia a la señal de salida.

Como el laboratorio posee dos Broadcaster de diferentes frecuencias, los contenidos digitales modulados de cada una se mezclan con un splitter[16] de señales UHF/VHF, y así posteriormente puedan ser radiadas por la misma antena.

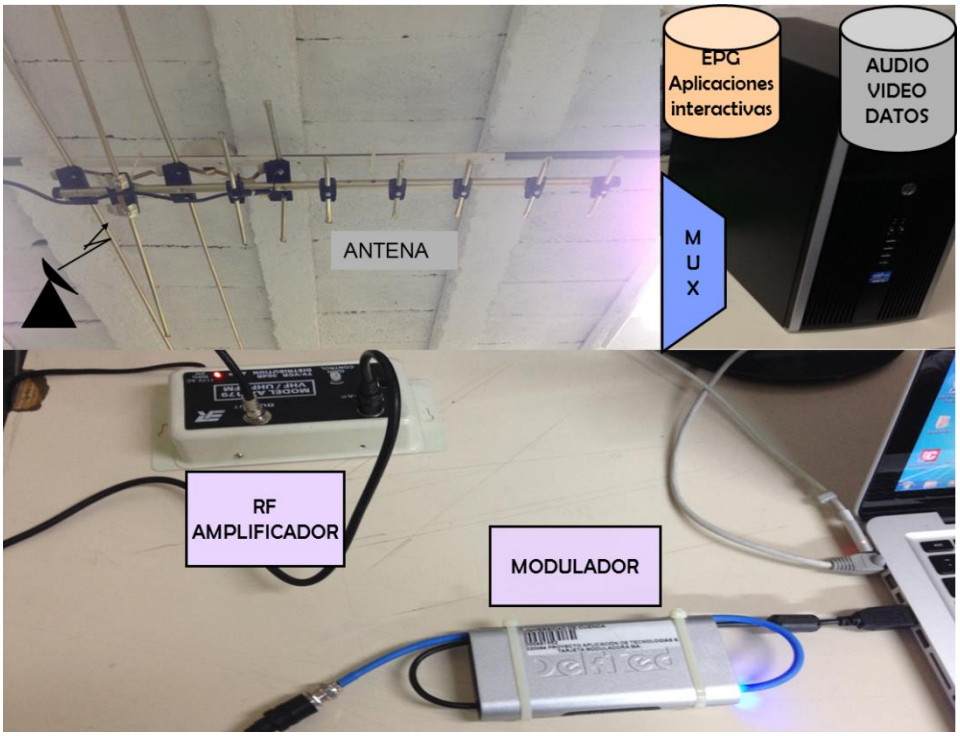


Figura A2: Módulos de arquitectura del laboratorio aplicados a los equipos

Equipos de Recepción

El laboratorio cuenta con 2 tipos de receptores de señal digital, cada uno con diferente tipo de estándar una digital y una analógica. El televisor digital incorpora un receptor interno para demodular la señal digital que es compatible con el estándar ISDB-T. Por otra parte se tiene una televisión Analógica la cual visualiza el contenido digital que proviene del STB. Es necesario un STB porque este demodula la señal digital y lo convierte a analógico para su respectiva visualización en el televisor. Estos dos tipos de receptores se pueden ver en la Figura A3

[16] Splitter es un dispositivo que permite mesclar o dividir una señal electromagnética.

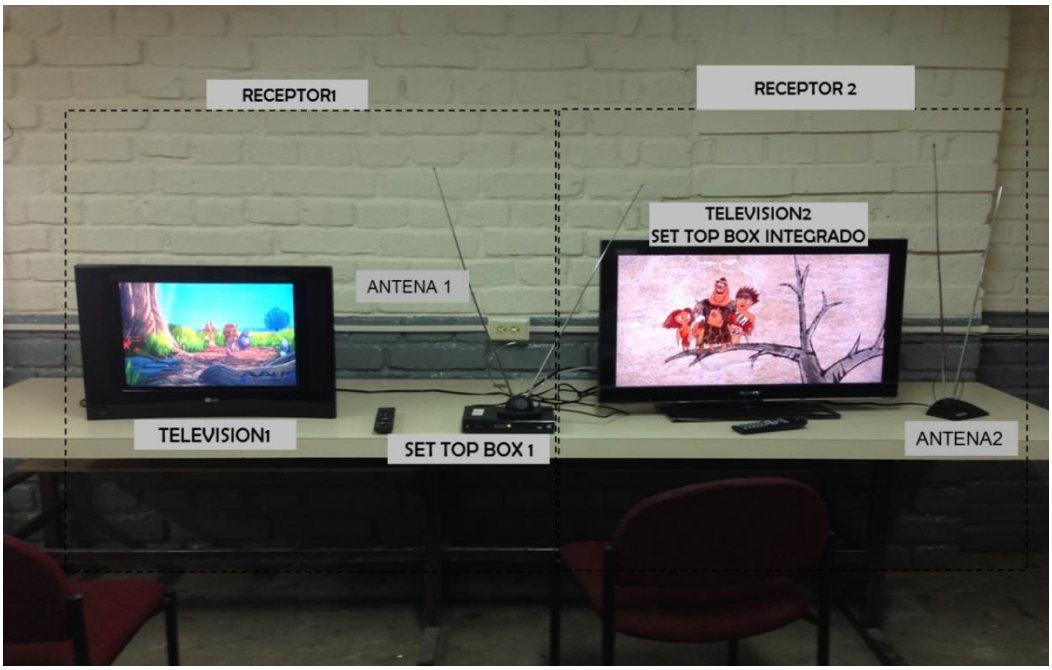


Figura A3: Receptores dentro del laboratorio

Una vez explicado las partes que conforman el laboratorio se muestra el estado actual del laboratorio en la Figura A4.

Figura A4: Laboratorio de TV Digital implementado

A.3 Instalación del ambiente de desarrollo de aplicaciones Ginga

Instalación Virtual Set Top Box

Esta herramienta nos permite emular el funcionamiento de las aplicaciones desarrolladas en la plataforma Ginga-NCL. Este STB es una máquina virtual desarrollada por los laboratorios Telemidia se la puede obtener en http://www.gingancl.org.br/en/ferramentas. Una vez descargada la máquina virtual se debe descomprimir en un repositorio de nuestra elección.

Para su ejecución se necesita tener previamente instalado el software VMWare, en donde procedemos a abrir la máquina virtual en la opción Open a Virtual Machine, aquí seleccionamos la carpeta donde esta descomprimida la máquina virtual llamada ubuntu-server10.10-ginga-i386 como se observa en la Figura A5.

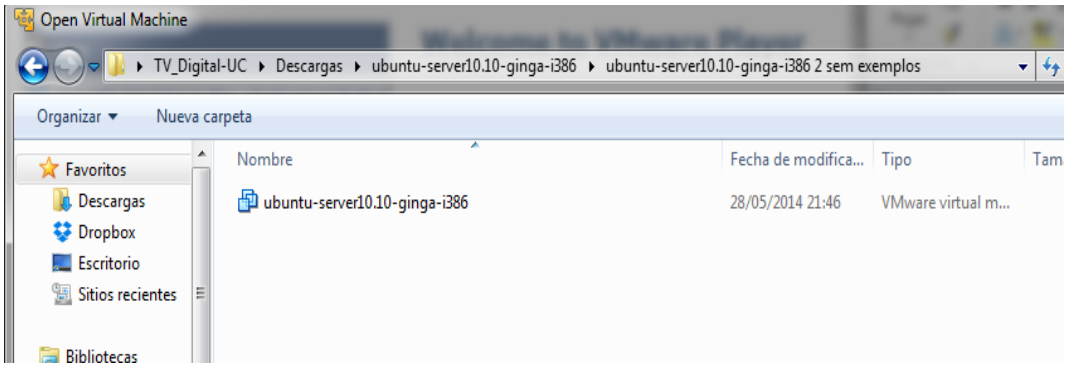


Figura A5: Archivo máquina virtual Ginga

Seleccionamos Play Virtual Machine. Una vez abierta nos muestra un menú donde seleccionamos la resolución adecuada para nuestro ordenador como se ve en la Figura A6.

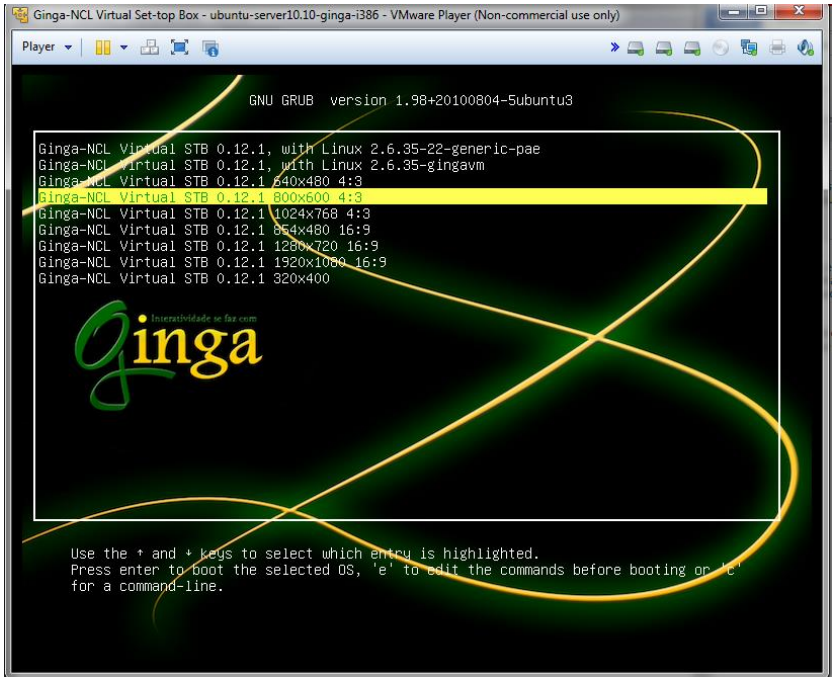


Figura A6: Interface Virtual Set Top Box

Luego de iniciar nos muestra la pantalla principal donde se ve un control remoto virtual con sus teclas equivalentes en el teclado del computador y también muestra la IP asignada al STB como se muestra en la Figura A7. Ya que este es un servidor de archivos remoto esta IP nos ayudara posteriormente a pasar las aplicaciones al Virtual STB.

Figura A7: IP y teclado virtual del STB

En el caso de que no sea asignada la dirección IP, se sigue un procedimiento adicional que consiste en: ir a Virtual Machine Settings, seleccionar Network Adapter y aquí se debe configurar el Network connection en modo bridged. Esto se puede ver en la Figura A8.

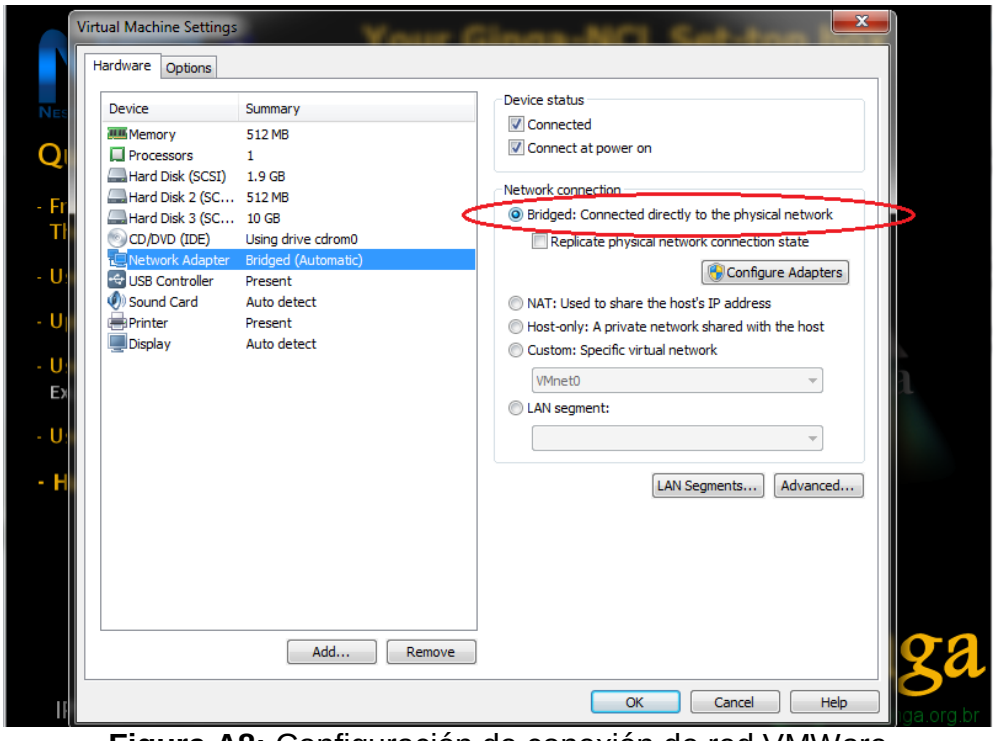


Figura A8: Configuración de conexión de red VMWare

Instalación del Plug-in Ginga-NCL en Eclipse

Como primer paso se debe contar con el software eclipse en el computador, el mismo se lo puede obtener en https://www.eclipse.org/downloads .

El siguiente paso es ejecutar eclipse y dirigirnos a Help->Install New Software (ver Figura A9).

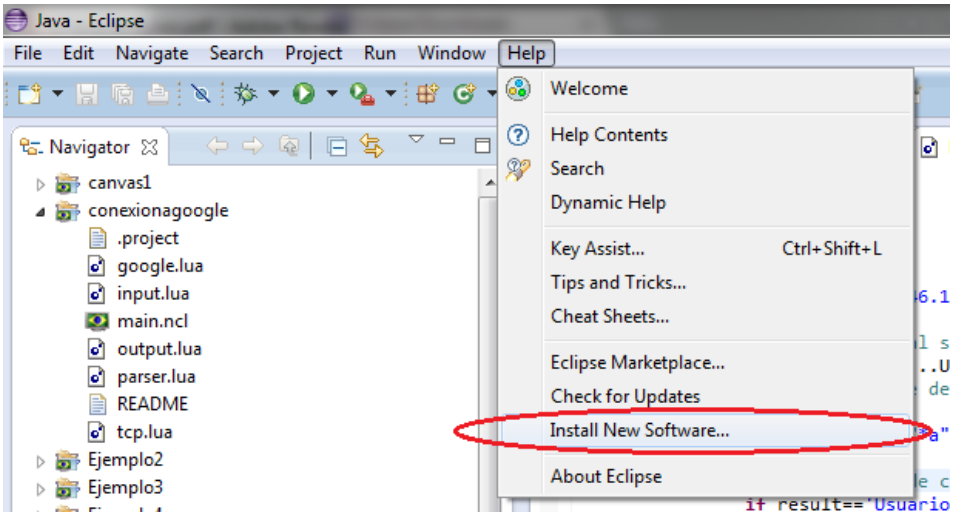


Figura A9: Instalación de complementos Eclipse

Nos presenta una pantalla adicional en donde se debe presionar Add (ver Figura A10).

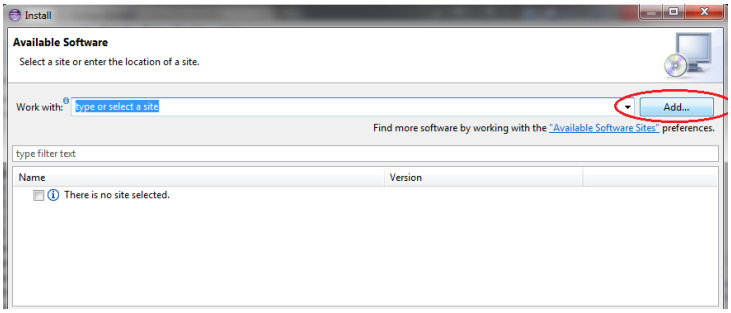


Figura A10: Interface agregar software a Eclipse

Con esto se muestra otra interfaz donde se especifica un nombre y la dirección url del plugin Ginga-NCL. Se deben llenar los datos como en la Figura A11.

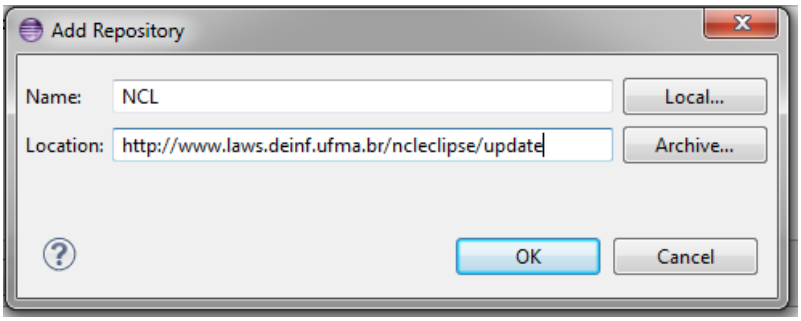


Figura A11: Interface agregar repositorio Ginga-NCL

Al presionar OK, eclipse se conecta a la dirección especificada y muestra el plug-in NCL Eclipse junto con la versión. Seleccionamos el plug-in y presionamos Next para proceder a instalar como se ve en la Figura A12.

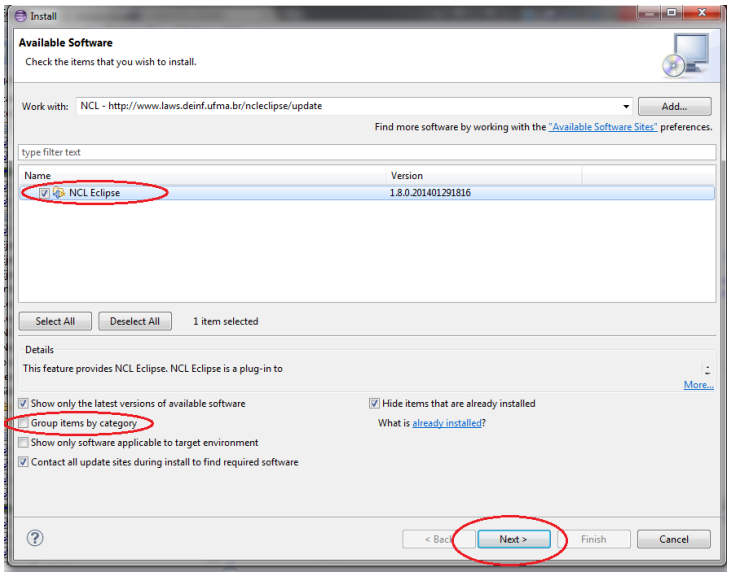


Figura A12: Interface agregar plug-in Ginga-NCL

Finalmente presionamos Finish y esperamos el reinicio eclipse.

Podemos verificar la instalación del plug-in presionando Ctrl+N, donde ya podremos elegir un proyecto NCL como lo muestra la Figura A13.

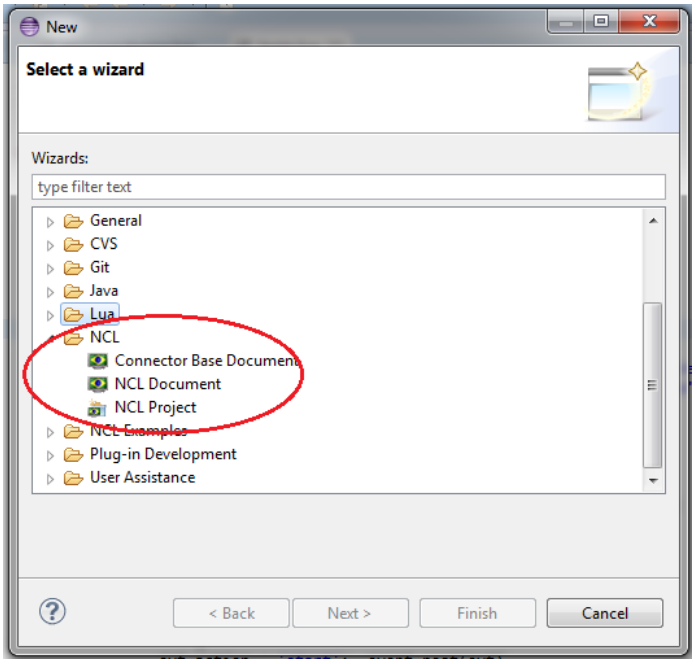


Figura A13: Generación de un proyecto NCL en Eclipse

Instalación del Plug-in Lua

Debido a que actualmente la instalación de este plug-in presenta un error al hacerlo mediante la URL, vamos a instalarlo manualmente.

Primero descargamos los plug-inde Lua 5 a nuestro computador dirigiéndonos a http://files.luaforge.net/releases/luaeclipse/luaeclipse/luaeclipse_0.5.0

Descomprimimos y pegamos la carpeta completa dentro de la carpeta plugins del directorio principal de eclipse como se ve en la Figura A14.

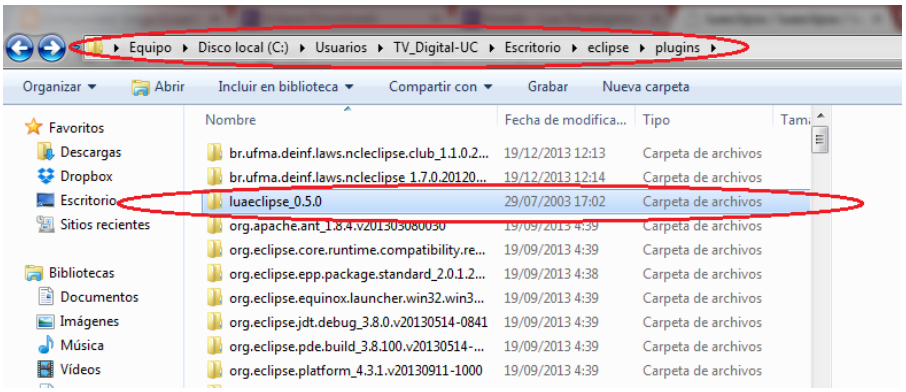


Figura A14: Repositorio plug-in Lua

Ejecutamos eclipse y verificamos que reconozca el plug-in presionando Ctrl+N donde tendremos la opción de poder generar un proyecto Lua (ver Figura A15).

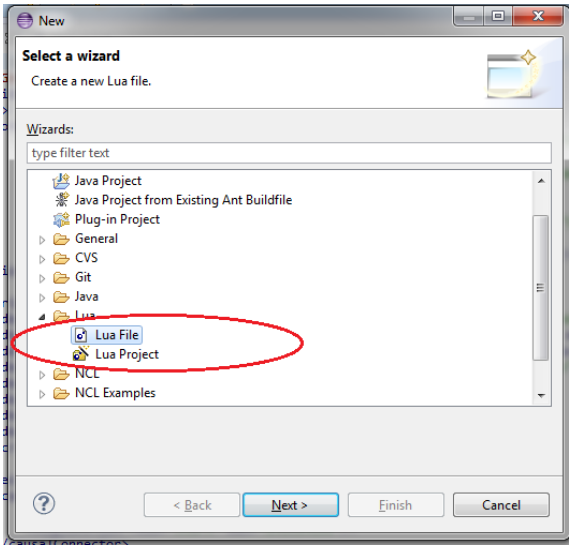


Figura A15: Generar proyecto lua en Eclipse

Configuración de Eclipse para conectarse al Virtual STB

Para poder emular las aplicaciones desarrolladas en eclipse es necesario configurar una herramienta de conexión remota de NCLeclipse.

Primero seleccionamos Windows->Preferences como se ve en la Figura A16

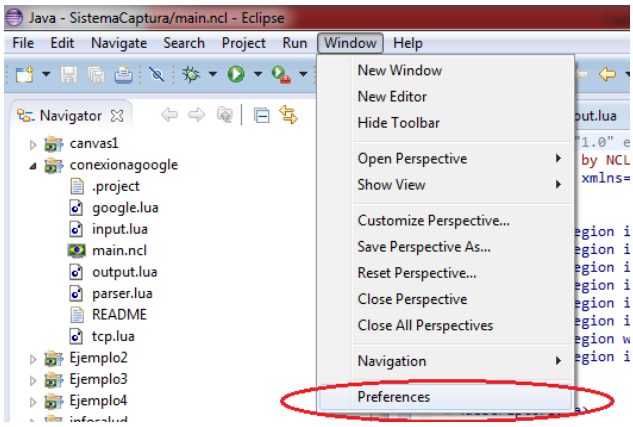


Figura A16: Acceso a la configuración de preferencias

Presentándonos una interfaz, en la cual elegimos NCl->Remote Ginga-NCL Play. Aquí se debe modificar los parámetros con los que está configurado el servidor remoto en este caso el Virtual STB. Los parámetros deben ser configurados como se muestra en la Figura A17. La IP del servidor es la misma que se le asignó al Virtual STB mediante el VMWare y la contraseña del servidor es “telemidia”.

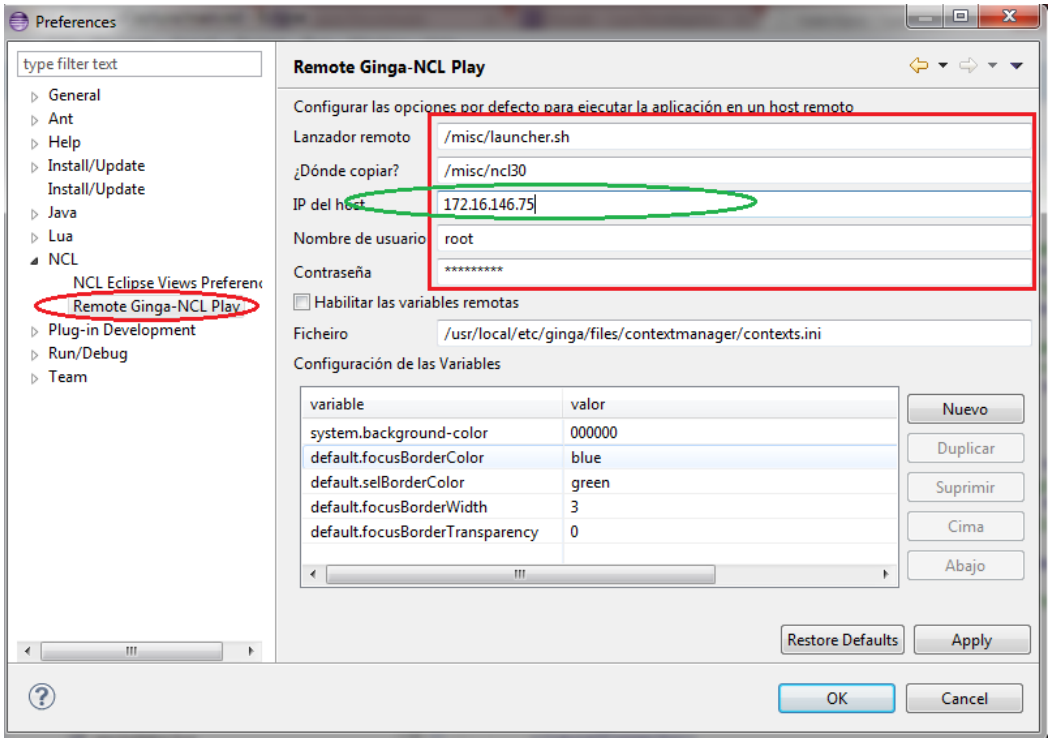


Figura A17: Configuración de conexión remota al STB virtual

Presionamos OK y procedemos a emular la aplicación presionando el botón play de eclipse y eligiendo Run As-> NCL Application. Previo a este paso debemos tener corriendo el Virtual STB en VMWare. Este paso se ve en la Figura A18.

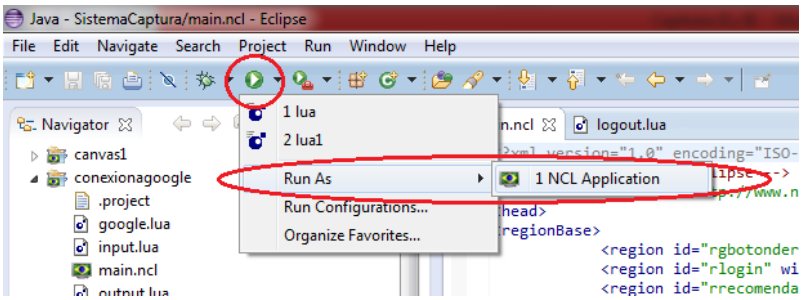


Figura A18: Ejecutar una aplicación Ginga-NCL en el STB virtual

Si las configuraciones son correctas podremos ver la aplicación corriendo sobre el Virtual STB.

A.4 Código NCL y Scripts Lua de la App de captura de datos

La aplicación de captura se compone de un archivo NCL, debido a que este es un lenguaje de tipo declarativo para completar su funcionamiento se tienen 8 diferentes scripts de lua. Como se ve en la Figura A19.

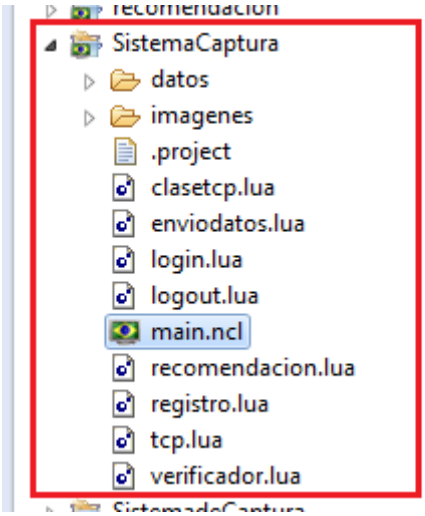


Figura A19: Archivos NCL y Lua de la Aplicación

A continuación se detalla el código presente en cada uno de estos archivos.

main.ncl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generated by NCL Eclipse -->
<ncl id="main" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>
<regionBase>
```

Aquí se introducen las regiones de pantalla a utilizar por la aplicación.

```
<region id="rgbotonderecho" right="5%" width="20%" height="5%" top="90%"/>
<region id="rlogin" width="20%" height="5%" left="45%" top="90%"/>
<region id="rrecomendacion" width="30%" height="5%" left="5%" top="90%"/>
<region id="rsalir" width="20%" height="5%" right="5%" top="90%"/>
<region id="rmensaje" width="100%" height="30%"  />
<region id="rtecladorecom" width="30%" height="20%" left="35%" top="32%"/>
<region width="100%" height="85%" top="5%" id="rregistro"/>
<region id="rgTeclado" width="26%" height="70%" top="7%" right="5%"/>
</regionBase>
<descriptorBase>
```

Se introduce descriptores para cada región utilizada.

```
      <descriptor id="dsbotonderecho" region="rgbotonderecho"
focusIndex="idxbotonderecho"/>
      <descriptor id="dlogin" region="rlogin" focusIndex="blogin"
moveLeft="brec" moveRight="bsalir"/>
      <descriptor id="drecomendacion" region="rrecomendacion"
focusIndex="brec" moveRight="blogin" />
      <descriptor id="dsalir" region="rsalir" focusIndex="bsalir"
moveLeft="blogin" moveUp="do"/>
      <descriptor id="dtecladorecom" region="rtecladorecom"/>
      <descriptor id="dsregistro"  region="rregistro" focusIndex="1"/>
      <descriptor id="dsTeclado" region="rgTeclado"/>
      <descriptor id="dmensaje" region="rmensaje" focusIndex="0"/>
</descriptorBase>
```

```
<connectorBase>
```

Mediante el connectorBase podremos definir el comportamiento de los scripts Lua y archivos multimedia.

```
	<causalConnector id="onBeginStart">
		<simpleCondition role="onBegin" />
		<simpleAction role="start" max="unbounded"/>
	</causalConnector>
	<causalConnector id="onSelectionSET">
		<connectorParam name="vhora" />
		<simpleCondition role="onSelection" />
		<simpleAction role="set" value="$vhora" />
	</causalConnector>
	<causalConnector id="onEndStop">
		<simpleCondition role="onEnd" />
		<simpleAction role="stop" max="unbounded" />
	</causalConnector>
	<causalConnector id="onSelectionStopStart">
		<simpleCondition role="onSelection" />
		<compoundAction operator="seq">
			<simpleAction role="start" max="unbounded"/>
			<simpleAction role="stop" max="unbounded"/>
		</compoundAction>
	</causalConnector>
	<causalConnector id="onSelectionStop">
		<simpleCondition role="onSelection" />
		<simpleAction role="stop" max="unbounded" />
	</causalConnector>
	<causalConnector id="onLogueoStart">
		<simpleCondition role="onEndAttribution" />
		<simpleAction role="start" />
	</causalConnector>
	<causalConnector id="onSetAtributoStart">
	<connectorParam name="var"/>
		<simpleCondition role="onEndAttribution" />
		<compoundAction operator="seq">
			<simpleAction role="start" />
			<simpleAction role="set" value="$var"/>
		</compoundAction>
	</causalConnector>
	 <causalConnector id="onCondIgualBeginStart">
		<connectorParam name="var"/>
		<compoundCondition operator="and">
			<simpleCondition role="onEnd" />
			<assessmentStatement comparator="eq">
				<attributeAssessment role="attNodeTest"
eventType="attribution" attributeType="nodeProperty"/>
				<valueAssessment value="$var"/>
			</assessmentStatement>
		</compoundCondition>
		<simpleAction role="start" max="unbounded"/>
	</causalConnector>
	<causalConnector id="onbeginSet">
		<connectorParam name="var"/>
		<simpleCondition role="onBegin" />
		<simpleAction role="set" value="$var" />
	</causalConnector>
</connectorBase>
	</head>
	<body>
```

Se especifica la ubicación de los archivos multimedia define el puerto de entrada de los archivos media.

```
      <port id="puerto1" component="mdverificarregistrolua"/>
      <media type="application/x-ginga-settings" id="programSettings">
        <property name="service.currentKeyMaster" value="0"/>
      </media>
    <media id="mdTeclado" src="imagenes/teclado.png" descriptor="dsTeclado"/>
    <media id="mdTeclado2" src="imagenes/tecladoregis.png"
descriptor="dsTeclado"/>
    <media id="mdguardar" src="imagenes/guardar.png"
descriptor="drecomendacion"/>
    <media id="mdtecladorecom" src="imagenes/-dias+.png"
descriptor="dtecladorecom"/>
    <media id="mregistrar" src="registro.lua"  descriptor="dsregistro">
            <area id="select"/>
            <property name="text"/>
      </media>
      <media id="enviodatos" src="enviodatos.lua"/>
      <media id="logout" src="logout.lua"    />
       <media id="mlogear" src="login.lua"  descriptor="dsregistro">
            <property name="text"/>
            <property name="envio"/>
      </media>
    <media id="mdverificarregistrolua" src="verificador.lua"
type="application/x-ginga-NCLua">
      <property name="menu"/>
      <property name="envio"/>
      </media>
      <media id="mdsalir" src="imagenes/salir.png" descriptor="dsalir" />
      <media id="mdmenu" src="imagenes/bmenu.png"
descriptor="dsbotonderecho"> <property name="sacarmenu"/>
      </media>
      <media id="mlogin" src="imagenes/login.png" descriptor="dlogin"
type="image/png"/>
      <media id="mrecomendacion" src="imagenes/entrar.png"
descriptor="drecomendacion"/>
      <media id="mlogout" src="imagenes/logout.png" descriptor="dlogin"
type="image/png"/>
      <media id="mregistro" src="imagenes/registrar.png"
descriptor="drecomendacion"/>
      <media id="mmensaje" src="recomendacion.lua" descriptor="dmensaje"
type="application/x-ginga-NCLua">
                  <property name="adhora"/>
                  <property name="adcanal"/>
            </media>
```

Mediante el link xconnector se definen que componentes son los que ejecutan los métodos definidos en connectorBase.

```
      <link xconnector="onSetAtributoStart">
            <bind role="onEndAttribution" component="mdverificarregistrolua"
interface="menu"/>
            <bind role="start" component="mdmenu"/>
            <bind role="set"  component="mdmenu" interface="sacarmenu">
                        <bindParam name="var" value="$get"/>
            </bind>
            <bind role="get" component="mdverificarregistrolua"
interface="menu"/>
      </link>
      <link xconnector="onLogueoStart">
```

```
            <bind role="onEndAttribution" component="mdverificarregistrolua"
interface="envio"/>
            <bind role="start" component="enviodatos"/>
      </link>
      <link xconnector="onLogueoStart">
            <bind role="onEndAttribution" component="mlogear"
interface="envio"/>
            <bind role="start" component="enviodatos"/>
      </link>

      <link xconnector="onSelectionStopStart">
            <bind role="onSelection" component="mregistro"/>
            <bind role="start" component="mregistrar"/>
            <bind role="start" component="mdTeclado2"/>
            <bind role="stop" component="mregistro"/>
            <bind role="stop" component="mlogin"/>
            <bind role="stop" component="mdsalir"/>
      </link>
      <link xconnector="onSelectionStopStart">
            <bind role="onSelection" component="mlogin"/>
            <bind role="start" component="mlogear"/>
            <bind role="start" component="mdTeclado"/>
            <bind role="stop" component="mregistro"/>
            <bind role="stop" component="mlogin"/>
            <bind role="stop" component="mdsalir"/>
      </link>
      <link xconnector="onSelectionStopStart">
            <bind role="onSelection" component="mrecomendacion"/>
            <bind role="start" component="mmensaje"/>
            <bind role="start" component="mdtecladorecom"/>
            <bind role="stop" component="mlogout"/>
            <bind role="stop" component="mrecomendacion"/>
            <bind role="stop" component="mdsalir"/>
      </link>
      <link xconnector="onEndStop">
            <bind role="onEnd" component="mregistrar"/>
            <bind role="stop" component="mdTeclado2"/>
      </link>
      <link xconnector="onEndStop">
            <bind role="onEnd" component="mlogear"/>
            <bind role="stop" component="mdTeclado"/>
      </link>
      <link xconnector="onSelectionStopStart">
            <bind role="onSelection" component="mdmenu"/>
            <bind role="start" component="mdsalir"/>
            <bind role="stop" component="mdmenu"/>
            <bind role="stop" component="mdverificarregistrolua"/>
      </link>
      <link xconnector="onSelectionStopStart">
            <bind role="onSelection" component="mlogout"/>
            <bind role="start" component="logout"/>
            <bind role="stop" component="mdsalir"/>
            <bind role="stop" component="mrecomendacion"/>
            <bind role="stop" component="mlogout"/>
      </link>
      <link xconnector="onEndStop">
            <bind role="onEnd" component="mmensaje"/>
            <bind role="stop" component="mdtecladorecom"/>
      </link>
```

```
	<link xconnector="onSelectionStop">
			<bind role="onSelection" component="mdsalir"/>
			<bind role="stop" component="mlogout"/>
			<bind role="stop" component="mrecomendacion"/>
			<bind role="stop" component="mdsalir"/>
			<bind role="stop" component="mmensaje"/>
			<bind role="stop" component="enviodatos"/>
			<bind role="stop" component="mlogin"/>
			<bind role="stop" component="mregistro"/>
	</link>
<link xconnector="onCondIgualBeginStart">
	<linkParam name="var" value="1"/>
	<bind role="onEnd" component="mdverificarregistrolua" />
	<bind role="attNodeTest" component="mdmenu" interface="sacarmenu"/>
	<bind role="start" component="mlogin"/>
	<bind role="start" component="mregistro"/>
</link>
<link xconnector="onCondIgualBeginStart">
	<linkParam name="var" value="2"/>
	 <bind role="onEnd" component="mdverificarregistrolua" />
	<bind role="attNodeTest" component="mdmenu" interface="sacarmenu"/>
	<bind role="start" component="mlogout"/>
	<bind role="start" component="mrecomendacion"/>
</link>
</body>
</ncl>
```

enviodatos.lua

```
require 'tcp'
function handler (evt)
	if evt.class  ~= 'ncl'         then return end
	if evt.type   ~= 'presentation' then return end
	if evt.action ~= 'start'       then return end
	tcp.execute(
	  function ()
--datos para conectar al servidor
	    tcp.connect('172.16.146.127', 9999)
	    local nombrefile='datos/canal.txt'
--lectura de datos codificados dentro del carrusel de datos
	     arquivo= assert(io.open(nombrefile,'r'),'Archivo no leido')
	     arquivo:seek('set')
	     linea1=arquivo:read()
--lectura de datos de usuario con sesión activa
	     arquivo1= assert(io.open('/archivo.txt','r'),'Archivo no leido')
		local y=1
		local lineas={}
		for linea in arquivo1:lines() do
			lineas[y]=linea
			y = y + 1
			end
--envío de datos que registran la actividad del usuario
		tcp.send('Historial/'..lineas[2]..'/'..linea1..'/fin%')
		result = tcp.receive("*a")
		tcp.disconnect()
	    local evt = {
		  class = 'ncl',
		  type  = 'presentation',
		  action  = 'stop'
```

```
                }
                event.post(evt)
        end)
end
event.register(handler)
```

Login.lua

```
require 'tcp'
text = nil
local TEXT = ''
local CHAR = ''
local linea=1
local Usuario=''
local Contrasena=''
local KEY, IDX = nil, -1
local MAP = {}
--Se define los caracteres a ser utilizados para llenar los campos de usuario
local MAP1 = {
        ['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','_','-' }
}
local MAP2 = {
        ['1'] = { '1'}
      , ['2'] = { '2'}
      , ['3'] = { '3'}
      , ['4'] = { '4'}
      , ['5'] = { '5'}
      , ['6'] = { '6'}
      , ['7'] = { '7'}
      , ['8'] = { '8'}
      , ['9'] = { '9'}
      , ['0'] = { '0'}
}
MAP=MAP1
local UPPER = false
local case = function (c)
      return (UPPER and string.upper(c)) or c
end
local dx, dy = canvas:attrSize()
canvas:attrFont('vera', dy/20)
canvas:attrColor('blue')
canvas:drawText(dx/20,0,'Usuario')
canvas:flush()
canvas:drawText(dx/20,2*dy/16,'Contrasena')
canvas:flush()
--Esta función muestra los caracteres presionados
function redraw ()
      canvas:attrColor(115,113,113,100)
      canvas:drawRect('fill', dx/20,linea*dy/16, dx/2,dy/16)
      canvas:attrColor('white')
```

```
	canvas:drawText(dx/20,linea*dy/16, TEXT..case(CHAR)..'|')
	canvas:flush()
end
local evt = {class = 'ncl',
		type  = 'attribution',
		name  = 'text'}
local function setText (new, outside)
	TEXT = new or TEXT..case(CHAR)
	text = TEXT
	CHAR, UPPER = '', false
	KEY, IDX = nil, -1
	if not outside then
		evt.value = TEXT
		evt.action = 'start'; event.post(evt)
		evt.action = 'stop';  event.post(evt)
	end
end
local TIMER = nil
--Esta función define los intervalos de tiempo para escoger un caracter
local function timeout ()
	return event.timer(1000,
		function()
			if KEY then
				setText()
			end
		end)
end
local function nclHandler (evt)
	if evt.class ~= 'ncl' then return end
	if evt.type == 'attribution' then
		if evt.name == 'text' then
			setText(evt.value, true)
		end
	end
	redraw()
	return true
end
event.register(nclHandler)
local sel = {class = 'ncl',
		type  = 'presentation',
		label = 'select'}
local function keyHandler (evt)
	if evt.class ~= 'key' then return end
	if evt.type ~= 'press' then return true end
	local key = evt.key
	if (key == 'ENTER') then
--ENTER cambia la línea de campo requerido para el inicio de sesión
		canvas:attrColor(115,113,113,100)
		canvas:drawRect('fill', dx/20,linea*dy/16, dx/2,dy/16)
		canvas:attrColor('white')
		canvas:drawText(dx/20,linea*dy/16,TEXT..case(CHAR))
		canvas:flush()
		setText()
		linea=linea+2
		if(linea>3) then
		linea=1
		Contrasena=TEXT
		MAP=MAP1
		else
```

```
            Usuario=TEXT
            MAP=MAP2
            end
            TEXT=''
      elseif (key == 'CURSOR_LEFT') then
--Borra un caracter del campo seleccionado
            setText( (KEY and TEXT) or string.sub(TEXT, 1, -2) )
      elseif (key == 'CURSOR_UP') then
--Cambia el ingreso de caracteres a mayúscula
            UPPER = not UPPER
      elseif (key == 'CURSOR_RIGHT') then
--Representa el espacio
            setText( (not KEY) and (TEXT..' ') )
      elseif (key == 'RED') then
--Termina la ejecución de la aplicación
            terminar = {
             class = 'ncl',
                   type  = 'presentation',
                   action = 'stop',
            }
            event.post(terminar)
      elseif (key == 'GREEN') then
--Envía los datos al servidor
            if linea==1 then
            Usuario=TEXT
      else
            Contrasena=TEXT
      end
      if (Usuario=='' or Contrasena=='' ) then
      canvas:attrColor('white')
      canvas:drawText(dx/20,6*dy/16,'Faltan Datos')
      canvas:flush()
      else
      tcp.execute(
      function ()
--Datos de conexión con el servidor
      tcp.connect('172.16.146.127', 9999)
      local result
--Envío solicitud de inicio de sesión
      local mensaje='Login/'..Usuario..'/'..Contrasena..'/fin%'
      tcp.send(mensaje)
      result = tcp.receive("*a")
      if result then
--Crea el archivo de registro de usuario con los datos de inicio de sesión
            if result=='UsuarioCorrecto%' then
            arquivo= assert(io.open('/archivo.txt','w'),'Archivo no creado')
            arquivo:write('logueado\n'..Usuario..'\n'..Contrasena..'\n')
            arquivo:flush()
            io.close(arquivo)
            canvas:attrColor('white')
            canvas:drawText(dx/20,6*dy/16,'Logueo completo')
            canvas:drawText(dx/20,7*dy/16,'Usuario: '..Usuario)
            canvas:drawText(dx/20,8*dy/16,'Contrasena: '..Contrasena)
            canvas:flush()
            local evt = {class = 'ncl',
                        type  = 'attribution',
                        name  = 'envio',
                        value = 2}
            evt.action = 'start';  event.post(evt)
```

```
            evt.action = 'stop';  event.post(evt)
            event.timer(4000, function()
            event.post {class  = 'ncl',
                        type   = 'presentation',
                        action = 'stop'}
            end)
            else
--Se muestra un mensaje de error para el inicio de sesión  incorrecto
            arquivo= assert(io.open('/archivo.txt','w'),'Archivo no creado')
            arquivo:write('deslogueado')
            arquivo:flush()
            io.close(arquivo)
            canvas:attrColor('white')
            canvas:drawText(dx/20,9*dy/16,'Logue Incorrecto')
            canvas:flush()
            end
      else
      result = 'error: ' .. evt.error
      end
      end)
      end
      elseif _G.tonumber(key) then
            if KEY and (KEY ~= key) then
                  setText()
            end
            IDX = (IDX + 1) % #MAP[key]
            CHAR = MAP[key][IDX+1]
            KEY = key
      end
      if TIMER then TIMER() end
      TIMER = timeout()
      redraw()
      return true
end
event.register(keyHandler)
```

logout.lua

```
function tratador(evt)
      if ( evt.class=='ncl' and evt.type=='presentation' and evt.action==
      'start') then
--Borra los datos de inicio de sesión en el archivo de registro
      arquivo= assert(io.open('/archivo.txt','w'),'Archivo no creado')
      arquivo:write('deslogueado')
      arquivo:flush()
      io.close(arquivo)
--Espera cuatro segundos para finalizar la aplicación
      event.timer(4000, function()
            local evt = {class = 'ncl',
                        type  = 'presentation',
                        action  = 'stop'}
            event.post(evt)
      end)
      end
end
event.register(tratador)
```

recomendacion.lua

```
require 'tcp'
local valor=0
local valor2=0
local valor3=1
local dias={}
local canales={}
local numerodias=0
local numerocanales=0
local canal1
local canal2
local canal3
local dx, dy , dxs, dys
local fechas
--se genera las horas de programación diaria
local
indices={'16/04/2014','00:00','1:00','2:00','3:00','4:00','5:00','6:00','7:00
','8:00','9:00','10:00','11:00','12:00','13:00','14:00','15:00','16:00','17:0
0','18:00','19:00','20:00','21:00','22:00','23:00'}
dx, dy = canvas:attrSize()
dxs=dx/7
dys=dy/4
function tratador (evt)
	if (evt.class=='ncl' and evt.type=='presentation' and
evt.action=='start') then
	leerdias()
	end
end
event.register(tratador)
function dibujarrow()
canvas:attrColor(115,113,113,100)
		for exp1=0,6,2 do
			canvas:drawRect('fill', exp1*dxs,0,dxs,dys)
		end
canvas:attrColor(60,57,57,100)
		for exp1=1, 7,2 do
			canvas:drawRect('fill', exp1*dxs,0,dxs,dys)
		end
canvas:flush()
canvas:attrFont('vera', dys/3)
canvas:attrColor('white')
canvas:drawText(10,dys/6,fechas)
	for exp1=1,6 do
		canvas:drawText(exp1*dxs+10,10,indices[valor*6+exp1+1])
	end
canvas:flush()
end
function dibujarcolum()
	canvas:attrColor(115,113,113,100)
	canvas:drawRect('fill',0,2*dys,dxs,dys)
	canvas:attrColor(60,57,57,100)
	for exp3=1,4,2 do
			canvas:drawRect('fill',0,exp3*dys,dxs,dys)

	end
	canvas:flush()
	canvas:attrFont('vera', dys/3)
	canvas:attrColor('white')
```

```
	if (numerocanales>=valor2*3+1) then
		canvas:drawText(10,dys+10,canal1[1])
		if (numerocanales>=valor2*3+2) then
		canvas:drawText(10,2*dys+10,canal2[1])
			if (numerocanales>=valor2*3+3) then
			canvas:drawText(10,3*dys+10,canal3[1])
			end
		end
	end
	canvas:flush()
end
function dibujarcuadros()
	canvas:attrColor(115,113,113,100)
	for exp1=2,6,2 do
		canvas:drawRect('fill', exp1*dxs,2*dys,dxs,dys)
	end
	for exp3=1, 4,2 do
		for exp1=1, 7,2 do
			canvas:drawRect('fill', exp1*dxs,exp3*dys,dxs,dys)
		end
	end

	canvas:attrColor(60,57,57,100)
	for exp3=1,4,2 do
		for exp1=2,6,2 do
			canvas:drawRect('fill', exp1*dxs,exp3*dys,dxs,dys)
		end
	end
	for exp1=1, 7,2 do
		canvas:drawRect('fill', exp1*dxs,2*dys,dxs,dys)
	end
	canvas:flush()
end
function cargarcanales()
	local tal=1
	canal1={}
	if (numerocanales>=valor2*3+1) then
	for token in string.gmatch(canales[(valor2*3)+1],"'(.-)',") do
		canal1[tal]= token
		tal=tal +1
	end
	canal2={}
	if (numerocanales>=valor2*3+2) then
		tal=1
	for tokn in string.gmatch(canales[(valor2*3)+2],"'(.-)',") do
		canal2[tal]= tokn
		tal=tal +1
	end
	canal3={}
	if (numerocanales>=valor2*3+3) then
		tal=1
	for tokeen in string.gmatch(canales[(valor2*3)+3],"'(.-)',") do
                                    canal3[tal]= tokeen
                                    tal=tal +1
	end
	end
	end
	end
end
```

```
function mostrarcontenido()
	cargarcanales()
	canvas:attrFont('vera', dys/3)
	canvas:attrColor('white')
		for exp1=1,6 do
		canvas:drawText(exp1*dxs+10,dys+10,canal1[valor*6+exp1+1])
		canvas:flush()
		end
		for exp1=1,6 do
		canvas:drawText(exp1*dxs+10,2*dys+10,canal2[valor*6+exp1+1])
		canvas:flush()
		end
		for exp1=1,6 do
		canvas:drawText(exp1*dxs+10,3*dys+10,canal3[valor*6+exp1+1])
		canvas:flush()
		end
end
function leerdias()
	arquivo= assert(io.open('/archivo.txt','r'),'Archivo no leido')
	local y=1
	local lineas={}
	for linea in arquivo:lines() do
		lineas[y]=linea
		y = y + 1
	end
	arquivo:close()
	tcp.execute(
	function ()
--conexión al servidor con petición de recomendación
	tcp.connect('172.16.146.127', 9999)
	local result
	canvas:attrFont('vera', dys/3)
	canvas:attrColor('white')
	canvas:drawText(10,10,'Conectando Servidor')
	canvas:flush()
	local mensaje='Recomendacion/'..lineas[2]..'/fin%'
	tcp.send(mensaje)
	result = tcp.receive("*a")
	local line = ""
	if result then
		line=result
	end
	local indicedias=1
--Busca dentro de el string recibido los días para la programación que están
--codificados dentro de 000 y 111 para saber cuándo inician y cuando terminan
	for dia in string.gmatch(line,"000(.-)111") do
		dias[indicedias]=dia
		indicedias=indicedias+1
	end
	numerodias=indicedias-1
	leercanales()
	dibujarrow()
	cargarcanales()
	dibujarcuadros()
	dibujarcolum()
	mostrarcontenido()
	end)
end
```

```
--leercanales() busca canales con recomendación dentro de un dia los canales
--están codificados entre {{ y }}
function leercanales()
            local indicecanal=1
            fechas = string.match(dias[valor3], '{{(.-)}}') or 'no asigan'
            for canal in string.gmatch(dias[valor3],"23(.-)24") do
                  canales[indicecanal]=canal
                  indicecanal=indicecanal+1
            end
            numerocanales=indicecanal-1
end
local function keyHandler (evt)
      if evt.class ~= 'key' then return end
      if evt.type ~= 'press' then return true end
      local key = evt.key
      if (key == 'CURSOR_RIGHT') then
--mueve la guía de programación hacia la derecha mostrando 6 horas de
--programación adicional
            valor=valor+1
            if valor>3 then
            valor=3
            end
            dibujarrow()
            dibujarcuadros()
            mostrarcontenido()
      elseif (key == 'CURSOR_LEFT') then
--mueve la guía de programación hacia la izquierda mostrando 6 horas de
--programación anterior
            valor=valor-1
            if valor<0 then
            valor=0
            end
            dibujarrow()
            dibujarcuadros()
            mostrarcontenido()
      elseif (key == 'CURSOR_DOWN') then
--cambia el grupo de canales mostrados moviendo la guía de programación hacia
--abajo
            valor2=valor2+1
            if valor2>=((numerocanales/3)) then
            valor2=valor2-1
            end
            cargarcanales()
            dibujarcolum()
            dibujarcuadros()
            mostrarcontenido()
      elseif (key == 'CURSOR_UP') then
--cambia el grupo de canales mostrados moviendo la guía de programación hacia
--arriba
            valor2=valor2-1
            if valor2<0 then
            valor2=0
            end
            cargarcanales()
            dibujarcolum()
            dibujarcuadros()
            mostrarcontenido()
      elseif (key == 'RED') then
--termina la ejecución de la aplicación
```

```
            terminar = {class = 'ncl',
                    type  = 'presentation',
                    action = 'stop'}
            event.post(terminar)
      elseif (key == 'YELLOW') then
--cambia el dia siguiente de programación mostrado en la guía de programación
            valor3=valor3+1
            if (valor3>numerodias) then
            valor3=numerodias
            end
            leercanales()
            dibujarrow()
            cargarcanales()
            dibujarcuadros()
            dibujarcolum()
            mostrarcontenido()
      elseif (key == 'GREEN') then
--cambia el dia anterior de programación mostrado en la guía de programación
            valor3=valor3-1
            if (valor3<1) then
            valor3=1
            end
            leercanales(); dibujarrow(); cargarcanales(); dibujarcuadros()
            dibujarcolum(); mostrarcontenido()
      end
return true
end
event.register(keyHandler)
```

Registro.lua

```
require 'tcp'
text = nil
local TEXT = ''
local CHAR = ''
local linea=1
local Usuario=''
local Contrasena=''
local Edad=''
local Sexo='M'
local KEY, IDX = nil, -1
local MAP = {}
local MAP1 = {
        ['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','_','-' }
}
local MAP2 = {
        ['1'] = { '1'}
      , ['2'] = { '2'}
      , ['3'] = { '3'}
      , ['4'] = { '4'}
```

```
	, ['5'] = { '5'}
	, ['6'] = { '6'}
	, ['7'] = { '7'}
	, ['8'] = { '8'}
	, ['9'] = { '9'}
	, ['0'] = { '0'}
}
MAP=MAP1
local UPPER = false
local case = function (c)
	return (UPPER and string.upper(c)) or c
end
local dx, dy = canvas:attrSize()
canvas:attrFont('vera', dy/20)
canvas:attrColor('blue')
canvas:drawText(dx/20,0,'Usuario')
canvas:drawText(dx/20,2*dy/16,'Contrasena')
canvas:drawText(dx/20,4*dy/16,'Edad')
canvas:drawText(dx/20+dx/50,6*dy/16,' Hombre')
canvas:drawText(4.5*dx/20+dx/50,6*dy/16,' Mujer')
canvas:attrColor(240,240,240,100)
canvas:drawRect('fill', dx/20,6.3*dy/16, dx/60,dy/50)
canvas:drawRect('fill', 4.5*dx/20,6.3*dy/16, dx/60,dy/50)
canvas:attrColor('blue')
canvas:drawRect('fill', dx/20,6.3*dy/16, dx/60,dy/50)
canvas:drawRect('frame', 4.5*dx/20,6.3*dy/16, dx/60,dy/50)
canvas:flush()
function redraw ()
	canvas:attrColor(115,113,113,100)
	canvas:drawRect('fill', dx/20,linea*dy/16, dx/2,dy/16)
	canvas:attrColor('white')
	canvas:drawText(dx/20,linea*dy/16, TEXT..case(CHAR)..'|')
	canvas:flush()
end
local evt = {
    class = 'ncl',
    type  = 'attribution',
    name  = 'text',
}
local function setText (new, outside)
	TEXT = new or TEXT..case(CHAR)
	text = TEXT
	CHAR, UPPER = '', false
	KEY, IDX = nil, -1
	if not outside then
		evt.value = TEXT
		evt.action = 'start'; event.post(evt)
		evt.action = 'stop';  event.post(evt)
	end
end
local TIMER = nil
local function timeout ()
	return event.timer(1000,
		function()
			if KEY then
				setText()
			end
		end)
end
```

```
local function nclHandler (evt)
	if evt.class ~= 'ncl' then return end
	if evt.type == 'attribution' then
		if evt.name == 'text' then
			setText(evt.value, true)
		end
	end
	redraw()
	return true
end
event.register(nclHandler)
local sel = {
    class = 'ncl',
    type  = 'presentation',
    label = 'select',
}
local function keyHandler (evt)
	if evt.class ~= 'key' then return end
	if evt.type ~= 'press' then return true end
	local key = evt.key
	if (key == 'ENTER') then
		canvas:attrColor(115,113,113,100)
		canvas:drawRect('fill', dx/20,linea*dy/16, dx/2,dy/16)
		canvas:attrColor('white')
		canvas:drawText(dx/20,linea*dy/16,TEXT..case(CHAR))
		canvas:flush()
		setText()
		linea=linea+2
		if(linea>5) then
		linea=1
		Edad=TEXT
		MAP=MAP1
		else
			if linea==3 then
				Usuario=TEXT
				MAP=MAP2
			else
				Contrasena=TEXT
				MAP=MAP2
			end
		end
		TEXT=''
	elseif (key == 'YELLOW') then
			if(Sexo=='M')then
			Sexo='F'
			canvas:attrColor(240,240,240,100)
			canvas:drawRect('fill', dx/20,6.3*dy/16, dx/60,dy/50)
			canvas:drawRect('fill', 4.5*dx/20,6.3*dy/16, dx/60,dy/50)
			canvas:attrColor('blue')
			canvas:drawRect('frame', dx/20,6.3*dy/16, dx/60,dy/50)
			canvas:drawRect('fill', 4.5*dx/20,6.3*dy/16, dx/60,dy/50)
			else
			Sexo='M'
			canvas:attrColor(240,240,240,100)
			canvas:drawRect('fill', dx/20,6.3*dy/16, dx/60,dy/50)
			canvas:drawRect('fill', 4.5*dx/20,6.3*dy/16, dx/60,dy/50)
			canvas:attrColor('blue')
			canvas:drawRect('fill', dx/20,6.3*dy/16, dx/60,dy/50)
			canvas:drawRect('frame', 4.5*dx/20,6.3*dy/16, dx/60,dy/50)
```

```
	end
	elseif (key == 'CURSOR_LEFT') then
		setText( (KEY and TEXT) or string.sub(TEXT, 1, -2) )
	elseif (key == 'CURSOR_UP') then
		UPPER = not UPPER
	elseif (key == 'CURSOR_RIGHT') then
		setText( (not KEY) and (TEXT..' ') )
	elseif (key == 'RED') then
		terminar = {
		 class = 'ncl',
			type  = 'presentation',
			action = 'stop',
		}
		event.post(terminar)
	elseif (key == 'GREEN') then
	if linea==1 then
		Usuario=TEXT
	else
			if linea==2 then
				Contrasena=TEXT
			else
				Edad=TEXT
			end
	end

	if (Usuario=='' or Contrasena=='' or Edad=='') then
	canvas:attrColor('white')
	canvas:drawText(dx/20,6*dy/16,'Faltan Datos')
	canvas:flush()
	else
	tcp.execute(
	function ()
	tcp.connect('172.16.146.127', 9999)
	local result
	local mensaje=
'Registro/'..Usuario..'/'..Contrasena..'/'..Edad..'/'..Sexo..'/fin%'
	tcp.send(mensaje)
	result = tcp.receive("*a")
	if result then
	   if result=='RegistroCorrecto%' then
		arquivo= assert(io.open('/archivo.txt','w'),'Archivo no creado')
		arquivo:write(
		'logueado\n'..Usuario..'\n'..Contrasena..'\n'..Edad..'\n'..Sexo)
		arquivo:flush()
		io.close(arquivo)
		canvas:attrColor('white')
		canvas:drawText(dx/20,8*dy/16,'Registro completado con exito')
		canvas:drawText(dx/20,9*dy/16,'Usuario: '..Usuario)
		canvas:drawText(dx/20,10*dy/16,'Contrasena: '..Contrasena)
		canvas:flush()
		event.timer(4000, function()
		event.post {
			class  = 'ncl',
			type   = 'presentation',
			action = 'stop',
			    }
		end)
	   else
		canvas:attrColor('white')
```

```
                canvas:drawText(dx/20,7*dy/16,'Usuario Existente')
                canvas:flush()
            end
        else
        canvas:attrColor('white')
        canvas:drawText(dx/20,11*dy/16,'Error en el registro')
        canvas:flush()
        end
        local evt = {
                    class = 'ncl',
                    type  = 'attribution',
                    name  = 'menu',
                    value = menu
                    }
        evt.action = 'start';  event.post(evt)
        evt.action = 'stop';  event.post(evt)
        end)
        end
        elseif _G.tonumber(key) then
              if KEY and (KEY ~= key) then
                    setText()
              end
              IDX = (IDX + 1) % #MAP[key]
              CHAR = MAP[key][IDX+1]
              KEY = key
        end
        if TIMER then TIMER() end
        TIMER = timeout()
        redraw()
        return true
end
event.register(keyHandler)
```

Verificador.lua

```
require 'tcp'
local menu=1
function tratador(evt)
      if( evt.class=='ncl' and evt.type=='presentation' and evt.action==
'start') then
      nombrefile='/archivo.txt'
      local abrir = io.open(nombrefile,'r')
            if abrir then
                  arquivo= assert(io.open(nombrefile,'r'),'no leido')
                  local y=1
                  local lineas={}
                  for linea in arquivo:lines() do
                        lineas[y]=linea
                        y = y + 1
                  end
                  io.close(arquivo)
                  if (lineas[1]=='logueado') then
                  tcp.execute(
                  function ()
--se conecta al servidor con una solicitud de verificación de datos de
--registro
                  tcp.connect('172.16.146.127', 9999)
                  local result
```

```
                    local mensaje='Login/'..lineas[2]..'/'..lineas[3]..'/fin%'
                    tcp.send(mensaje)
                    result = tcp.receive("*a")
                    if result then
--recive confirmación de usuario valido para mostrar el menú 2 (recomendación
--y logout)
                        if result=='UsuarioCorrecto%' then
                            menu=2
                            local evt = {class = 'ncl',
                                         type  = 'attribution',
                                         name  = 'envio',
                                         value = 2}
                            evt.action = 'start';  event.post(evt)
                            evt.action = 'stop';  event.post(evt)
                        end
                    end
                    tcp.disconnect()
                    local evt = { class = 'ncl',
                                type  = 'attribution',
                                name  = 'menu',
                                value = menu }
                    evt.action = 'start';  event.post(evt)
                    evt.action = 'stop';  event.post(evt)
                    end)
            else
                    local evt = {class = 'ncl',
                                type  = 'attribution',
                                name  = 'menu',
                                value = menu}
                    evt.action = 'start';  event.post(evt)
                    evt.action = 'stop';  event.post(evt)
            end
            else
            local evt = {class = 'ncl',
                    type  = 'attribution',
                    name  = 'menu',
                    value = menu }
            evt.action = 'start';  event.post(evt)
            evt.action = 'stop';  event.post(evt)
            end
      end
end
event.register(tratador)
```

A.5 Codificación y Generación del Transport Stream

Antes de codificar el contenido audiovisual es necesario conocer las características del audio y video del programa a codificar. Para ello se utiliza la herramienta tovid que se la instala en Ubuntu con el comando.

apt -get install tovid

Entre algunas de sus herramientas de esta librería está el descriptor de video, que nos presenta las características del audio y del video de una grabación *tovid id*. En este caso utilizaremos el video Video.avi del cual se extraen las características mostradas en la Figura A20.

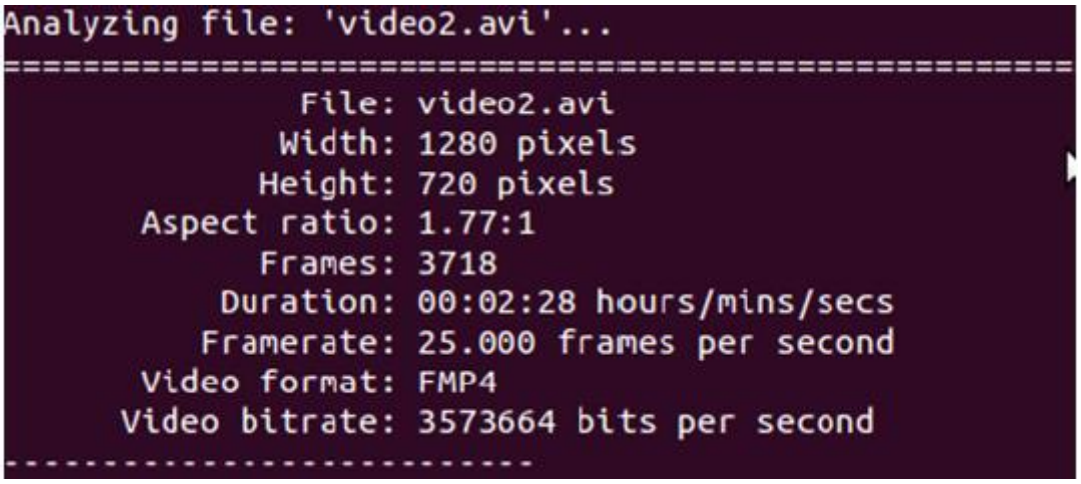


Figura A20: Características de video

La codificación del video y el audio se hace de manera separada. Se parte de un video a codificar en cualquier formato y mediante la librería ffmpeg y el OpenCaster se lo transforma en el formato deseado para la codificación. En este punto utilizaremos el formato .avi, ya que este codifica el video en H.264 y el audio con ACC, características propias del estándar ISDB-Tb. Para esto utilizamos la siguiente línea de código.

ffmpeg -i Video.wmp -s 1080x720 -aspect 16:9 -b 3600k -r 25 video.avi

- *-i Ucuenca.wmp* Indica el video de origen a codificar
- *-s 1080x720* Muestra el tamaño de la imagen en píxeles. Este parámetro es importante según la resolución que deseamos darle al video, en la Figura A21 se muestra las distintas resoluciones.

Formato	Líneas de exploración	Frecuencia de exploración	Pixelación (V x H)	Velocidad de cuadro	Relación de aspecto	Formatos
LDTV	625 total 288 activas	15.625Hz (25P)	352 x 288	24P, 25P o 50i	4:3	3
SDTV	625 total 576 activas	15.625Hz (50i)	720 x 576	24P, 25P, o 50i	4:3 y 16:9	6 (3 x 2)
EDTV	625 total 720 activas	31.250Hz (50P)	1.080 x 720	24P, 25P o 50i	4:3 y 16:9	6 (3 x 2)
HDTV	750 total 720 activas	37.500Hz (50P)	1.280 x 720	24P, 25P o 50i	16:9 y 20:9	6 (3 x 2)
HDTV	1.250 total 1.080 activas	31.250Hz (50i)	1.920 x 1.080	24P, 25P o 50i	16:9 y 20:9	6 (3 x 2)

Figura A21: Parámetros de codificación de video

- *-aspect 16:9* Da la relación de aspecto que tendrá el video codificado. En este caso se puede usar 1.77 o 16:9 definidos según el estándar vistos en la Figura A21.
- *-b 3600k* Indica la tasa de transmisión del video en bps
- *-r 25* Indica la tasa de cuadros por segundo. Las recomendaciones para definir este parámetro se muestran en la Figura A21.
- *video.avi* Es el video de salida en formato avi.

Codificación del Video

Como se mencionó la codificación del video se realiza por separado a la del audio para luego ser multiplexados. La codificación de video consta de 3 pasos con el objetivo final de obtener el Transport Stream de video.

Obtención del Elementary Stream de Video

Para obtener el Elementary Stream (ES) se hace uso de la librería ffmpeg nuevamente. Para este tipo de codificación se debe tener muy en cuenta los parámetros descritos al momento de codificar el video .avi.

ffmpeg -i video.avi -an -r 25 -f yuv4mpegpipe -| yuvdenoise| ffmpeg -i - -an -vcodec mpeg2video -f mpeg2video -b 3600k -minrate 3600k -maxrate 3600k -bf 2 - bufsize 134388 video.es

- *-i video.avi* Muestra el video de entrada (input) a codificar.
- *-an* Indica que la salida es solo video (no audio)
- *-r 25* Codifica el video a una tasa de 25 fotogramas por segundo.
- *-f yum4mpegpipe* Indica que el formato forzado es en mpeg4.
- *- | yuvdenoise |* Suprime el ruido generado al codificar el video

Hay que tener en cuenta que a este mismo archivo ya codificado se lo vuelve a codificar ahora en mpeg2 también permitido con el estándar.

- *-vcodec* Muestra el codec a usar para la codificación del video.
- *-f mpeg2 video* El formato de salida será mpeg2.
- *-b 3600k* Muestra la tasa de transmisión del video en bps.
- *-minrate 3600k* Muestra la tasa mínima de transmisión del video en bps.

- *-maxrate 3600k* Muestra la tasa máxima de transmisión del video en bps.
- *-bf 2* Parámetro "bidirectionally predictive coded picture" el cual contiene diferente información de la trama anterior o siguiente para un grupo de imágenes.
- *-bufsize 134388* Indica el tamaño de buffer a utilizar tiene que ver con El Video Buffer Verifier (VBV). Este valor es importante ya que muestra la memoria temporal necesaria para que el receptor pueda decodificar los paquetes de una manera segura. Es un parámetro de acuerdo a especificaciones del receptor que para este caso es VBV=112.
- *video.es* Es el video de salida en forma de Elementary Stream.

Empaquetar el Elementary Stream

En este punto se forman paquetes a partir de los Elementary Stream que conformaran el TS de video. Para esto se utilizan la librería esvideo2pes este se encarga de encapsular todos los ES generados anteriormente.

esvideo2pes video.es > video.pes

- *video.es* Video de origen en formato de Elementary Stream (ES).
- *video.pes* Salida de video en forma de Paquetized Elementary Stream (PES) o ES empaquetado.

Generación del TS de Video

Finalmente se convierte los paquetes que contienen el Elementary Stream en paquetes Transport Stream. Para esto se considera algunos parámetros definidos anteriormente en el estándar. Esto se consigue utilizando la librería pesvideo2ts.

pesvideo2ts 2067 30 112 4140000 0 video.pes > video.ts

- *2067* Este corresponde al ID de video especificado luego en la tabla PTM
- *30* Es la tasa de fotogramas por segundo definida ya previamente
- *112* Este parámetro indica el valor del verificador del buffer de video (VBV). Se debe tomar en cuenta el VBV escogido previamente en la codificación de video.
- *4140000* Es el ancho de banda (BW) que ocupara el video a la hora de la transmisión. Este debe ser un 15\% mayor al ancho de banda propio del video. Esto para garantizar que se transmitirán todos los paquetes.
  *BW=((BW video)+ (BW video)*0,15)=(3600k+3600k*,15)=4140k*
- 0 Indica que 0 no existe un loop de video 1 si existe.

Se debe tener en cuenta que este comando nos lleva a un bucle infinito. En este caso debemos esperar a que el archivo .ts generado deje de variar su peso en bytes, para luego de esto quebrar su ejecución con Ctr+C. Al final se obtendrán los archivos de la Figura A22.

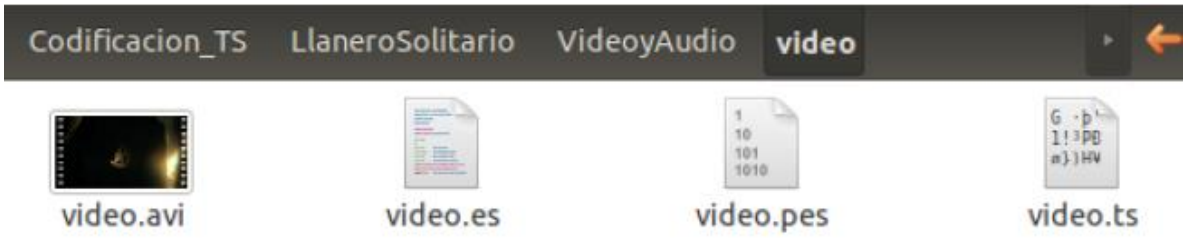


Figura A22: Archivos de video codificados

Codificación del Audio

Esta codificación se la hace por separado, para luego ser multiplexada y sincronizada con el video. Consiste en los mismos 3 pasos al momento de codificar el video.

Obtención del ES de Audio

Utilizamos la librería *ffmpeg* nuevamente, una vez obtenidas las características del audio mediante *tovid* (ver Figura A23), se debe tener muy en cuenta que estos son los parámetros para codificar el audio. Si los parámetros no son compatibles puede disminuir la calidad de audio en lugar de mejorarla.

```
----------------------------
Audio track 1 (Stream 0.1, AID 0):
----------------------------
              Codec: mp2
            Bitrate: 64000 bits per second
      Sampling rate: 44100 Hz
```

Figura A23: Parámetros de audio (tovid)

La estructura del comando de codificación de audio es la siguiente.

ffmpeg -i video.avi -vn -ab 128k -ar 48000 -acodec mp2 -ac 2 audio.es

- -i video.avi Muestra el video de entrada (input) a codificar
- -vn Indica que la salida es solo audio (no video)
- -ab 128k Indica la tasa de bits por segundo de audio.
- -ar 48000 Indica la frecuencia de muestreo del audio, este valor puede ser 48000 44100 y 32000
- -acodec mp2 Muestra el codec a usar para la codificacion del audio , para este caso mp2
- -ac 2 Muestra los canales de audio de salida en este caso 2 sonido estéreo.
- audio.es Es el audio de salida en forma de Elementary Stream

Empaquetar el ES

El proceso de empaquetar el Elementary Stream se realiza de manera similar al video. En este caso se utiliza el comando esaudio2pes, esta librería se encarga de encapsular los ES de audio.

esaudio2pes audio.es 1152 48000 384 3600 > audio.pes

- *audio.es* Audio de origen en formato de ES
- *1152* Este parámetro indica el número de muestras por trama de audio. Este valor depende de la versión y capa MPEG que se esté utilizando, en este caso se utilizó 1152 ya que pertenece a la versión 1 de capa 2. Este valor se lo puede encontrar en la siguiente Tabla

Capas	MPEG 1	MPEG 2 (LSF)	MPEG 2.5 (LSF)
Capa 1	384	384	384
Capa 2	1152	1152	1152
Capa 3	1152	576	576

- *48000* Este parámetro indica la frecuencia de muestreo.
- *384* Aquí se especifica el tamaño de la Trama (FS)
- *3600* Es el parámetro de sincronización. Se utiliza para asegurarse que tanto el video como el audio estén en la misma línea de tiempo.
- *audio.pes* Salida de audio en forma de Paquetized Elementary Stream (PES)

Generación del Transport Stream de Audio

Por ultimo al igual que en el video se transforma los paquetes en un TS. Este será multipexado y sincronizado con el video y los datos a través de las distintas tablas. Para generar este TS utilizamos la librería pesaudio2ts.

pesaudio2ts 2068 1152 48000 384 0 audio.pes > audio.ts

- 2068 Este corresponde al ID del audio. Posteriormente se especificara en la tabla PTM
- 1152 Especifica el número de muestras por trama de audio.
- 48000 Este parámetro define la frecuencia de muestreo
- 384 Define el tamaño de la Trama.
- 0 El 0 indica que no existe un loop de audio.
- audio.pes Salida de audio en forma de Paquetized Elementary Stream (PES)
- video.ts Video de salida como Transport Stream

Recordemos que este comando nos lleva a un bucle infinito. Por lo que debemos esperar a que el archivo .ts generado deje de variar en su tamaño en mega bytes y después pararlo mediante Ctr+C. Con lo que obtenemos los archivos mostrados en la Figura A24.

Figura A24: Archivos codificados de audio

Generación de TS a partir de las Tablas PSI/SI

Una vez que se hayan codificado el video y el audio, es necesario insertar los datos a transportar y multiplexados de una manera ordenada. Como se explicó anteriormente estos datos contienen las distintas tablas como son las PAT, PMT, EIT, SDT, NIT, etc...

Todo este conjunto de datos conforman un servicio de Televisión Digital. La creación de este servicio se lo hace a partir de la creación de algunas tablas. Python es la herramienta que nos ayuda a copilar y generar estas tablas. En la Figura A25 se muestra el código de generación de los archivos .ts que contienen las tablas codificadas.

```
# ESCRIBIENDO LAS TABLAS A ARCHIVOS
out = open("./nit.sec", "wb")
out.write(nit.pack())
out.close()
os.system("sec2ts 16 < ./nit.sec > ./nit.ts")
out = open("./pat.sec", "wb")
out.write(pat.pack())
out.close()
os.system("sec2ts 0 < ./pat.sec > ./pat.ts")
```

Figura A25: Generación de archivos .ts de las tablas PSI/SI

Multiplexación del Audio, Video y Datos dentro del TS

Una vez obtenidos las diferentes secciones del TS procedemos a multiplexarlas dentro de un Transport Stream final. Para esto debemos considerar el ancho de banda (BW) al momento de la transmisión que se necesitara para que la calidad de video y audio sea la adecuada y para que las tablas sean recibidas de manera adecuada por el receptor.

El siguiente ejemplo, con el uso de *tscbrmuxer* aclara de mejor manera la especificación de estos parámetros.

tscbrmuxer 2000000 b:2300000 video.ts b:188000 audio.ts b:15040 pat.ts b:15040 pmtsd.ts b:3008 sdt.ts b:3008 nit.ts b:3008 firsteit.ts b:27431190 null.ts > mux.ts

- *2000000* Nos indica el número de paquetes a transmitir. Para el cálculo de este parámetro se considera al ancho de banda en ISDB-Tb (29958294 bits por segundo), con paquetes de 188 bytes, sabemos que se transmiten 20000 paquetes por segundo por lo tanto en 1 minuto 40 segundos de transmisión de video tenemos el siguiente resultado.
  Paquetes= 20000((1*60)+40)=2000000*
- *2300000 video.ts* Nos indica el ancho de banda dedicado para el video.
- *188000 audio.ts* Especifica el ancho de banda dedicado para el audio.
- *15040 pat.ts* Aquí indicamos el ancho de banda de la tabla PAT. Se sabe que esta tabla debe enviarse al menos 10 veces por segundo, así obtenemos la siguiente tasa.
  *Ancho de banda PAT=188*8bits=1504bits*10/s=15040 bps*
- *15040 pmtsd.ts* Este es el ancho de banda para la tabla PMT.se realiza un cálculo igual al anterior teniendo en cuenta que también debe permanecer actualizada 10 veces en 1 segundo.
- *3008 sdt.ts* Ancho de Banda de la tabla SDT. Se calcula de la misma forma que en los casos anteriores.
- *3008 nit.ts* Parámetro que indica el ancho de banda de la tabla NIT.
- *3008 firsteit.ts* Indicamos el ancho de banda de la tabla EIT. Recordemos que esta tabla contiene la Guía de Programación del canal por lo que es adicional a las tablas básicas generadas.
- *27431190 null.ts* Este parámetro se utiliza para llegar a cumplir con el ancho de banda del estándar. En este caso se llenan de paquetes nulos el TS para así conseguir una tasa de 29958294.
- *mux.ts*: Es el flujo TS de salida.

Sincronización del TS

Hay que tener en cuenta que el archivo *mux.ts* obtenido al final del paso anterior no se encuentra sincronizado con la tasa adecuada para el receptor. Esta sincronización del archivo se la realiza mediante un comando adicional donde se especifica la tasa adecuada mediante la cual el receptor funcionara de manera óptima, debido a que el receptor recibe los datos y los decodifica asumiendo una tasa fija de 29958294 bits por segundo.

tsstamp mux.ts 29958294 > llanero.ts

- *29958294* Es la tasa especificada por el estándar para la transmisión en 6MHz de ancho de banda.
- *llanero.ts* Es el archivo final del Transport Stream y que se envía al receptor.

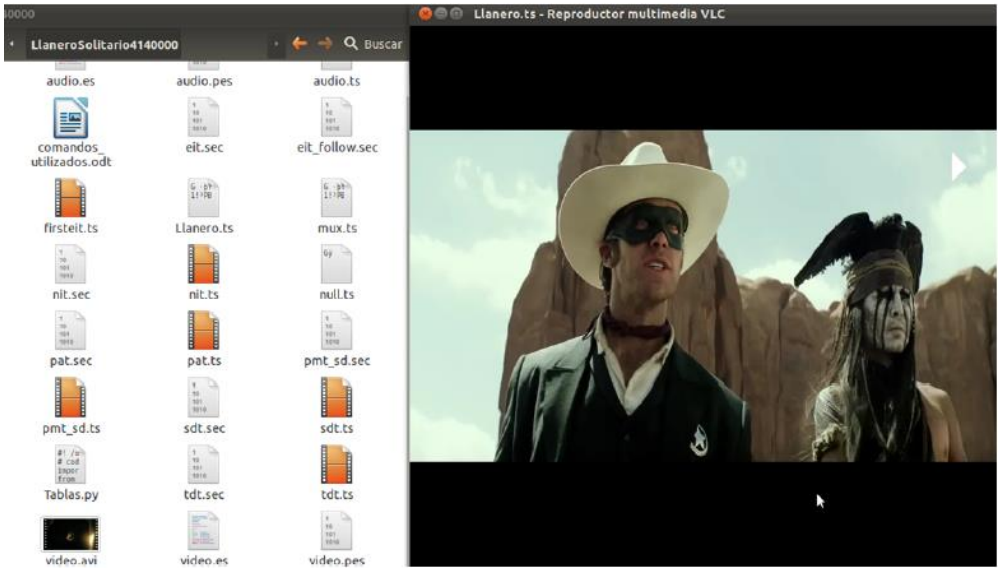


Figura A26: Prueba del Transport Stream en VLC Media PLayer

Finalmente al terminar la generación de todos estos archivos el archivo “llanero.ts” puede ser reproducido por VLC Media Player como en la Figura A26 o puede ser radiado directamente hacia el receptor.

A.6 Codificación de Tablas PSI/SI

A continuación se muestra el código completo para la generación de las tablas PSI/SI.

```
#! /usr/bin/python
# coding: utf-8
import os
from dvbobjects.PSI.PAT import *
from dvbobjects.PSI.NIT import *
from dvbobjects.PSI.SDT import *
from dvbobjects.PSI.PMT import *
from dvbobjects.PSI.EIT import *
from dvbobjects.PSI.TDT import *
from dvbobjects.MHP.AIT import *
from dvbobjects.MHP.Descriptors import *
from dvbobjects.SBTVD.Descriptors import *
```

En este bloque se definen las variables que contienen información propia del Broadcaster.

```
nombre_canal="Universidad de Cuenca"
nombre_programa="Demo Helicoptero"
descripcion_programa="Programa uno transmitido por la Universidad de Cuenca"
fecha_year=114
fecha_month=05
fecha_dia=01
fecha_hora=0x00
fecha_minutos=0x00
fecha_segundos=0x00
tiempo_hora=0x01
tiempo_minutos=0x00
tiempo_segundos=0x00
frecuencia = 195
id_control_remoto = 0x8 # Número del canal
pid_vid=2064
pid_aud=2068
pid_pmt_servicio = 1031
id_ts_ucuenca = 1
id_original_ts_ucuenca = 1
id_servicio = 1
id_red_ucuenca = 1
```

A continuación se define todas las tablas.

```
#18 DEFINICIÓN DE LA RED: NIT
nit = network_information_section(
network_id = id_red_ucuenca,
network_descriptor_loop = [
network_descriptor(network_name = nombre_canal,),
system_management_descriptor(
                        broadcasting_flag = 0,
                        broadcasting_identifier = 3,
                        additional_broadcasting_identification = 0x01,
                        additional_identification_bytes = [],
)
],
transport_stream_loop = [
          transport_stream_loop_item(
          transport_stream_id = id_ts_ucuenca,
```

```
		original_network_id = id_red_ucuenca,
		transport_descriptor_loop = [
		service_list_descriptor(
		dvb_service_descriptor_loop = [
			service_descriptor_loop_item (
			service_ID = id_servicio,
			service_type = 1,
			),
		],
		),
		terrestrial_delivery_system_descriptor(
			area_code = 1341,
			guard_interval = 0x01,
			transmission_mode = 0x02,
			frequencies = [
				tds_frequency_item( freq=frecuencia )
			],
		),
		partial_reception_descriptor (
			service_ids = []
		),
		transport_stream_information_descriptor (
			remote_control_key_id = id_control_remoto,
			ts_name = nombre_canal,
			transmission_type_loop = [
				transmission_type_loop_item(
				transmission_type_info = 0x0F,
				service_id_loop = [
					service_id_loop_item(
					service_id=id_servicio
					),
				]
				),
				transmission_type_loop_item(
					transmission_type_info = 0xAF,
					service_id_loop = [],
				),
			],
		)
		],
		),
],
version_number = 1,
section_number = 0,
last_section_number = 0,
)
# DEFINICION DEL MAPA DE PROGRAMAS: PAT
pat = program_association_section(
transport_stream_id = id_ts_ucuenca,
program_loop = [
program_loop_item(
program_number = id_servicio,
PID = pid_pmt_servicio ,
),
program_loop_item(
program_number = 0,
PID = 16,
),
],
version_number = 1,
```

```
section_number = 0,
last_section_number = 0,
)
# DEFINICIÓN DE LOS SERVICIOS: SDT
sdt = service_description_section(
transport_stream_id = id_ts_ucuenca,
original_network_id = id_red_ucuenca,
service_loop = [
service_loop_item(
service_ID = id_servicio,
EIT_schedule_flag = 1,#0
EIT_present_following_flag = 1,#0
running_status = 4,
free_CA_mode = 0,
service_descriptor_loop = [
service_descriptor(
service_type = 1,#digital television service
service_provider_name = nombre_canal,
service_name = nombre_canal,
),
],
),
],
version_number = 1,
section_number = 0,
last_section_number = 0,
)
# DEFINICIÓN DE LA TABLA: PMT
pmt_sd = program_map_section(
        program_number = id_servicio,
        PCR_PID = pid_vid,
        program_info_descriptor_loop = [],
        stream_loop = [
                stream_loop_item(
                        stream_type = 2,
                        elementary_PID = pid_vid,
                        element_info_descriptor_loop = [
                        ]
                ),
                stream_loop_item(
                        stream_type = 3,#MPEG-2 ---4 MPEG2
                        elementary_PID = pid_aud,
                        element_info_descriptor_loop = []
                ),
        ],
        version_number = 1,
        section_number = 0,
        last_section_number = 0,
        )
#**********TDT*******************
tdt = time_date_section(
        year = fecha_year, # since 1900
        month = fecha_month,
        day = fecha_dia,
        hour = fecha_hora, # use hex like decimals
        minute = fecha_minutos,
        second = fecha_segundos,
    version_number = 1,
    section_number = 0,
    last_section_number = 0,
```

```
        )
#**************EPG*****************
eit = event_information_section(
            table_id = EIT_ACTUAL_TS_PRESENT_FOLLOWING,
            service_id = id_servicio,
            transport_stream_id = id_ts_ucuenca,
            original_network_id = id_red_ucuenca,
            event_loop = [
              event_loop_item(
                      event_id = 1,
                      start_year = fecha_year, # desde 1900
                      start_month = fecha_month,
                      start_day = fecha_dia,
                      start_hours = fecha_hora,
                      start_minutes = fecha_minutos,
                      start_seconds = fecha_segundos,
                      duration_hours = tiempo_hora,
                      duration_minutes = tiempo_minutos,
                      duration_seconds = tiempo_segundos,
                      running_status = 4,
                      free_CA_mode = 0,
                     event_descriptor_loop = [
                       short_event_descriptor (
                       ISO639_language_code = "spa",
                       event_name = nombre_programa,
                       text = descripcion_programa,
                         )
                     ],
             ),
             ],
              version_number = 1,
              section_number = 0,
              last_section_number = 1,
              #last_segment_section_number = 1,
              )
eit_follow = event_information_section(
  table_id = EIT_ACTUAL_TS_PRESENT_FOLLOWING,
  service_id = id_servicio,
  transport_stream_id = id_ts_ucuenca,
  original_network_id = id_red_ucuenca,
  event_loop = [
   event_loop_item(
    event_id = 2,
    start_year = fecha_year, # since 1900
    start_month = fecha_month,
    start_day = fecha_dia+1,
    start_hours = 0x00,
    start_minutes = 0x00,
    start_seconds = 0x01,
    duration_hours = 0x12,
    duration_minutes = 0x00,
    duration_seconds = 0x00,
    running_status = 1,
    free_CA_mode = 0,
    event_descriptor_loop = [
     short_event_descriptor (
      ISO639_language_code = "spa",
      event_name = "Otro Programa",
      text = "El presente contenido es el siguiente programa pruebas TVD U de Cuenca",
     )
```

```
  ],
 ),
],
version_number = 1,
section_number = 1,
last_section_number = 1,
#last_segment_section_number = 1,
)
```

Finalmente el siguiente bloque genera los Transport Stream de cada tabla en formato .ts.

```
# ESCRIBIENDO LAS TABLAS A ARCHIVOS
out = open("./nit.sec", "wb")
out.write(nit.pack())
out.close()
os.system("sec2ts 16 < ./nit.sec > ./nit.ts")
out = open("./pat.sec", "wb")
out.write(pat.pack())
out.close()
os.system("sec2ts 0 < ./pat.sec > ./pat.ts")
out = open("./sdt.sec", "wb")
out.write(sdt.pack())
out.close()
os.system("sec2ts 17 < ./sdt.sec > ./sdt.ts")
out = open("./pmt_sd.sec", "wb")
out.write(pmt_sd.pack())
out.close()
os.system("sec2ts " + str(pid_pmt_servicio) +
" < ./pmt_sd.sec > ./pmt_sd.ts")
out = open("./tdt.sec", "wb")
out.write(tdt.pack())
out.close
out = open("./tdt.sec", "wb") # python   flush bug
out.close
os.system('/usr/local/bin/sec2ts 20 < ./tdt.sec > ./tdt.ts')
out = open("./eit.sec", "wb")
out.write(eit.pack())
out.close()
os.system("sec2ts 18 < ./eit.sec > ./firsteit.ts")
out = open("./eit_follow.sec", "wb")
out.write(eit_follow.pack())
out.close()
os.system("sec2ts 18 < ./eit_follow.sec >> ./firsteit.ts")
```

A.7 Codificación de Aplicación Ginga en el Transport Stream (fuente [8])

A continuación se detalla paso a paso la generación e inclusión de una aplicación dentro del flujo de transporte de un servicio de televisión digital.

Generación del Carrusel de Datos

Primero copiamos todos los archivos incluyendo aplicación y objetos adicionales que se quieran enviar a través del carrusel de datos mediante los siguientes comandos.

$ mkdir gingaapp
$ cp -r <path a la aplicación>/ gingaapp/*

Para generar el carrusel usamos el comando oc-update.sh que viene con OpenCaster.

$ oc-update.sh gingaapp 0x0C 1 2004 2

Esto genera un archivo gingaapp.ts que contiene los paquetes que llevan las secciones del carrusel. Los parámetros del comando son:

- *gingaapp* define el nombre del archivo ts. Este es igual al directorio donde se encuentra la aplicación y demás datos adicionales.
- *0x0C* El association_tag del carrusel generado.
- *1* El número de versión de los módulos generados.
- *2004* El PID en el que se envía el carrusel.
- *2* El carousel_id.

Modificación de las Tablas AIT y PMT

Primero, agregamos los encabezados necesarios en la primera parte del archivo:

```
from dvbobjects.MHP.AIT import *
from dvbobjects.MHP.Descriptors import *
```

Después agregamos la definición de la AIT. Los campos carousel_id, association_tag, el pid y demás se corresponden con los que usamos cuando ejecutamos la herramienta oc-update.sh.

También, en ginga_ncl_application_location_descriptor, tenemos que asegurarnos que initial_class se corresponda con la clase inicial NCL(main.ncl) y base_directory corresponda con el directorio de main.ncl.

```
ait = application_information_section(
application_type = 0x0009, # Codigo utilizado para GINGA-NCL
common_descriptor_loop = [],
application_loop = [
application_loop_item(
organisation_id = 0x0000000A,
application_id = 0x64,
application_control_code = 0x01, # indica la ejecución automática
application_descriptors_loop = [
transport_protocol_descriptor(
protocol_id = 0x0001,
```

```
transport_protocol_label = 0,
remote_connection = 0,
component_tag = 0x0C, # association_tag usado en oc-update.sh
),
application_descriptor(
application_profile = 0x0001,
version_major = 1,
version_minor = 0,
version_micro = 0,
service_bound_flag = 1,
visibility = 3,
application_priority = 1,
transport_protocol_labels = [ 0 ],),
application_name_descriptor(
application_name = "APP_GINGA" #nombre de la aplicación
),
ginga_ncl_application_descriptor(
parameters = [ ] ),
ginga_ncl_application_location_descriptor (
base_directory = "/", # directorio del initial_class
class_path_extension = "",
initial_class = "main.ncl", # nombre del archivo NCL a ser ejecutado.
),
]
),
],
version_number = 0, # cambia cuando se actualize la aplicacion
section_number = 0,
last_section_number = 0,
)
```

Modificamos la PMT, para que el servicio incluya la AIT y el carrusel. Para esto se agregan dos elementary streams.

```
pmt_sd = program_map_section(
program_number = tvd_service_id_sd,
PCR_PID = 2064,
program_info_descriptor_loop = [],
stream_loop = [
stream_loop_item(
stream_type = 2, # video tipo mpeg2 stream
elementary_PID = 2064, # identificador de video
element_info_descriptor_loop = [] ),
stream_loop_item(
stream_type = 3, # audio tipo mpeg2 stream
elementary_PID = 2068, # identificador de audio
element_info_descriptor_loop = [] ),
stream_loop_item(
stream_type = 5, # AIT
elementary_PID = 2001,
element_info_descriptor_loop = [
data_component_descriptor (
data_component_id = 0xA3, # sistema AIT
additional_data_component_info = ait_identifier_info(
application_type = GINGA_NCL_application_type,
ait_version = 0
).bytes(), ),
application_signalling_descriptor(
application_type = 9, # Aplicacion  GINGA-NCL
AIT_version = 1, # version ait actual
),
]
```

```
),
stream_loop_item(
stream_type = 0x0B, # DSMCC stream
elementary_PID = 2004,
element_info_descriptor_loop = [
association_tag_descriptor(
association_tag = 0x0C,
use = 0,
selector_lenght = 0,
transaction_id = 0x80000000,
timeout = 0xFFFFFFFF,
private_data = "",
),
stream_identifier_descriptor(
component_tag = 0x0C,
),
carousel_identifier_descriptor(
carousel_ID = 2,
format_ID = 0,
private_data = "",
),
data_component_descriptor (
data_component_id = 0xA0, # sistema GINGA
additional_data_component_info = additional_ginga_j_info(
transmission_format = 0x2,
document_resolution = 0x5,
organization_id = 0x0000000A,
application_id = 0x0064,
carousel_id = 2,
).bytes(),
),
])
],
version_number = 0,
section_number = 0,
last_section_number = 0,)
```

Se agrega el código que escribe las secciones y los paquetes que las transportan:

```
out = open("./ait.sec", "wb")
out.write(ait.pack())
out.close()
os.system('sec2ts ' + str(2001) + ' < ./ait.sec > ./ait.ts')
```

Se ejecuta el script y se genera las tablas actualizadas y finalmente se multiplexan y sincroniza dentro del flujo de transporte.

$ tscbrmuxer 600000 b:2300000 video.ts b:188000 audio.ts b:15040 pat.ts b:15040 pmt_sd.ts b:3008 sdt.ts b:3008 nit.ts b:3008 ait.ts b:400000 app_ginga.ts b:27031190 null.ts > prueba1.ts

$ tsstamp prueba1.ts 29958294 > prueba1.fixed.ts