MASKAY 2(1), Nov 2012
ISSN 1390-6712

Recibido (Received): 2012/04/08
Aceptado (Accepted): 2012/08/11

Desarrollo de aplicaciones para TDT con GINGA-J

Alulema D.

***Abstract*— This article is an introduction to the events that led to Ecuador to the adoption of ISDB-Tb standard for digital terrestrial television, and is considered an analysis of the Ginga middleware, and especially Ginga-J for developing interactive applications and finally presents an application developed using Netbeans.**

I. INTRODUCTION

La televisión, merecida o inmerecidamente ha tomado un lugar preponderante en los hogares ecuatorianos, constituyéndose en el principal medio de entretenimiento y acceso a la información, pero se ha estancado en su desarrollo, considerando que sus 2 grandes revoluciones fueron la incorporación del color y el teletexto. Es así que el Ecuador el 26 de marzo del 2010, adopta el estándar ISDB-T, para la transición, de la televisión analógica a la televisión digital terrestre (TDT), con lo que se llega a la tercera revolución en la televisión, dando la posibilidad al usuario de elegir lo que quiere ver y cuando lo quiere ver, además le ofrece interactividad empleando un canal de retorno, integrando así contenidos gubernamentales, educativos, de salud, comerciales, etc.; lo que abre un nuevo mundo de oportunidades para quienes las saben aprovechar, ya que es necesario el desarrollo de aplicaciones interactivas, para lo cual el estándar ISDB-T, incorpora el middleware Ginga, para su desarrollo empleando NLC-LUA o JAVA.

II. LA TDT EN EL ECUADOR

Dentro de la perspectiva del actual gobierno por incluir nuevas tecnologías de información (TIC´s), la Televisión Digital Terrestre, tiene su espacio, por lo que el Presidente de la República en decreto ejecutivo 681 del 18 de octubre del 2007, delegó a la SUPERTEL, el análisis, las pruebas y recomendaciones para definir el estándar más adecuado para la realidad del Ecuador; es así que se estableció contacto con el Gobierno de Japón y la Comunidad Europea, para el préstamo de equipos de transmisión de televisión digital para los estándares ISDB-T y DVB-T.

Las pruebas con los equipos se llevaron a cabo el 9 de diciembre del 2008, con los equipos proporcionados por el Gobierno de Japón para el estándar ISDB-T; el 11 de febrero del 2009, con los equipos proporcionados por la Comunidad Europea, para el estándar DVB-T; el 22 de abril del 2009, se adaptan los equipos para ISDB-T para que funcionen con SBTVD (Brasileño) y por último el 24 de Junio del 2009, se prueban los equipos para el estándar DTMB, Chino.

Alulema D., es docente en el Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejército. (cell: 098622881, e-mail: darwin_oafotmail.com)

La SUPERTEL presentó el informe en el que recomendaba adoptar el estándar ISDB-T, con lo que el 26 de Marzo del 2010 el CONARTEL acepta la recomendación para adoptar el estándar ISDB-T, en su versión internacional, por prestar una mejor calidad de señal, mayor número de canales; interacción de usuarios y operadores, entre otros.

La SUPERTEL estima que en un plazo de 6 a 10 años el Ecuador podrá hacer el apagón analógico, en el que todas las transmisiones analógicas cesarán y serán reemplazadas por la TDT, para lo cual los usuarios tendrán dos alternativas para la recepción: disponer de un televisor digital o de un decodificador.

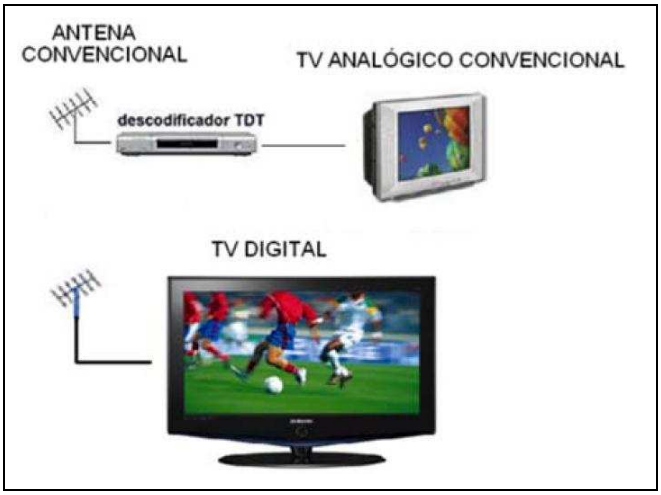

Fig. 1. Sistema de recepción de la TDT

III. DIFUSIÓN DIGITAL DE SERVICIOS INTEGRADOS TERRESTRES (ISDB-T)

Todo estándar se base en tres principios:

- Estándar tecnológico de codificación y transmisión de la señal.
- Arquitectura y plataforma tecnológica del origen de los servicios (middleware).
- Modelo de explotación de negocio.

Es así que uno de los estándares existentes hoy en día es el ISDB, el cual fue creado en Japón, el cual se divide en tres:

- ISDB-S.- Creado para la televisión digital vía satélite.
- ISDB-C.- Creado para la televisión digital por cable.
- ISDB-T.- Creado para las difusiones terrestres

ISDB-T presenta la ventaja de ahorro del espectro radioeléctrico, transmisión simultánea de varios programas por el mismo canal, recepción móvil, inmunidad a multitrayecto, transmisores de menor potencia e interactividad

Cabe notar que el Ecuador adopta el estándar ISDB-T, en su versión internacional, o japonés con modificaciones brasileñas, al cual también se lo conoce como ISDB-Tb, o SBTVD-T (Sistema Brasileño de Televión Digital Terrestres); diferenciándose del estándar ISDB-T en el códec de video utilizando H.264 o MPEG4, en vez de MPEG2; y en el middleware utilizando GINGA en vez de ARIB; lo que determina que la modificación brasileña presenta mejor calidad de video en condiciones más desfavorables y un middleware enfocado principalmente a la inclusión social.

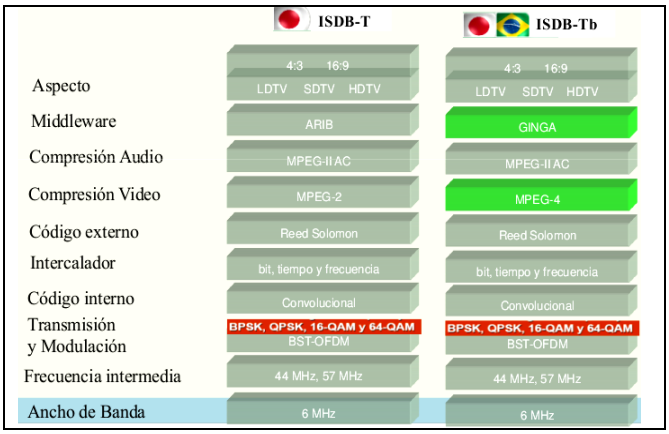

Fig. 2. Comparación del estándar Japonés y el Brasileño

Para que se puedan trasmitir señales compuestas de video, audio y datos, estas deben ingresar a un multiplexor para formar una trama TS (Transport Stream), la salida del multiplexor se conecta a los moduladores, (PDH, SDH, fibra óptica, satelital), para que su señal de salida ingrese al trasmisor y por último la señal va a la antena.

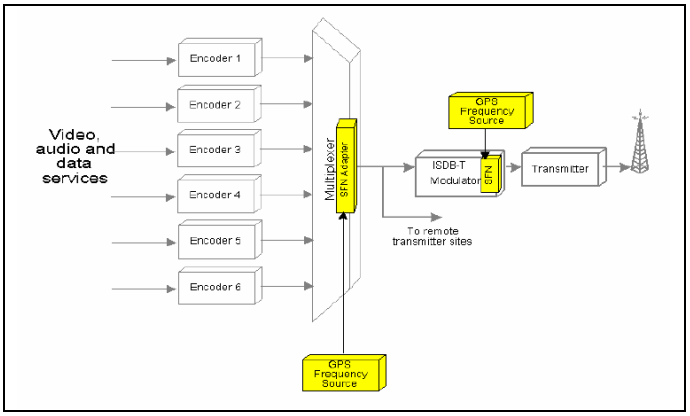

Fig. 3. Composición de la señal

El multiplexor, trabaja con MPEG-2, cabe mencionar que no se debe confundir con la compresión de video de ISDB-T, para fusionar los contenidos de la televisión así como las aplicaciones de interactividad, para formar una trama del Transport Stream; este permite llevar hasta cinco programas en formato SDTV (Standard Definition Television); o un programa de HDTV (High Definition Television) con un programa en SDTV; por un canal digital que utiliza el mismo ancho de banda que un canal de 6 Mhz de televisión analógica.

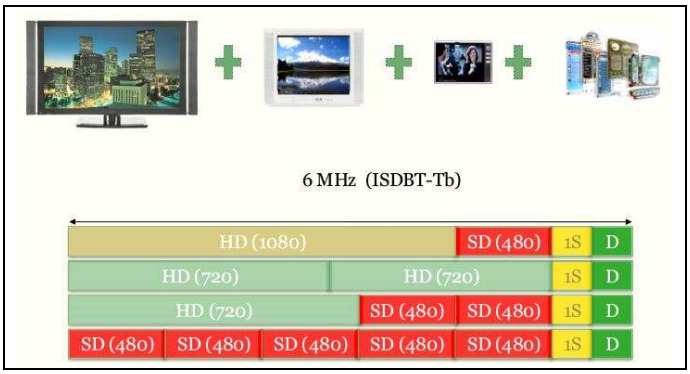

Fig. 4. Posibles divisiones del canal de 6Mhz

El modulador ISDB-Tb, trabaja con COFDM (Coded Ortogonal Frequency Diiivision Multiplexing) para la codificación del canal y la modulación del Transport Stream MPEG-2, el cual soporta altos niveles de multi-trayecto, como los que se encuentran en las zonas urbanas; alta dispersión de retardos entre las señales recibidas; e interferencia co-canal de banda estrecha, como los que se presentan en los sistemas analógicos.

Para la recepción de la señal, es necesario demodular, demultiplexar y decodificar la señal, es necesario un dispositivo que capture dichas señales, llamado set top box, para conectarlos a un televisor analógico o adquirir un televisor que lo tenga incorporado; con lo que los usuarios pueden recibir la programación convencional y una serie de servicios asociados; por lo que es necesario que incorporen, un sistema de acceso condicional CA, interfaces de

programación API y herramientas de navegación; los que están estrechamente relacionados con el hardware y el ancho de banda.

ISDB-Tb permite la recepción portátil y móvil; la primera se la consigue sin mayor dificultad utilizando una antena telescópica en una zona de cobertura; mientras que la segunda consiste en que la señal de televisión no solo pueda ser vista desde cualquier lugar de la zona de cobertura, sino incluso en movimiento. Considerando que hay canales que tiene errores en ráfagas, ISDB-Tb para permitir una mejor recepción móvil, incorpora entrelazado de datos. El entrelazo de datos es necesario porque la intensidad de la señal recibida en un móvil varía con el movimiento, provoca errores ´por ráfaga que son incorregibles con FEC (Correlation Error Forward), por lo que, se coloca antes de la transmisión datos adyacentes con una separación máxima de 0,5 segundos antes del entrelazado temporal, por lo que, el error de ráfaga en el móvil se convierte en un error aleatorio, luego de pasar por el entrelazado temporal, y es corregible por el sistema de corrección de errores.

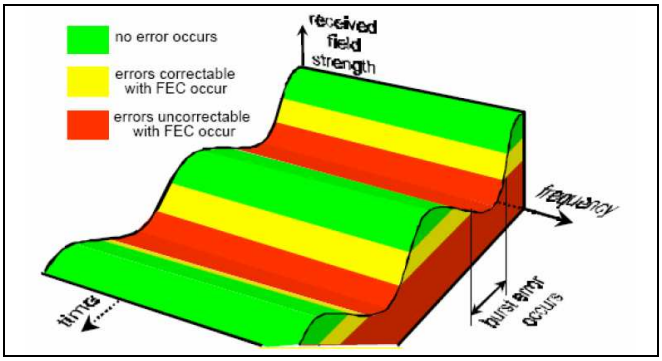

Fig. 5. Variación de la intensidad de la señal

Para conseguir una interactividad total, es necesario un canal de retorno, pero a diferencia de lo que ocurre con la televisón por cable, la arquitectura de la TDT no contempla directamente un canal de retorno para garantizar la interactividad con el proveedor; por lo cual se suele emplear cualquier otra tecnología disponible para el acceso a la red de datos

IV. MIDDLEWARE GINGA

Para el desarrollo de aplicaciones interactivas el estándar ISDB-Tb, propone como middleware GINGA, el cual tiene la posibilidad de desarrollar aplicaciones de manera declarativa, llamando a este subsistema GINGA–NCL (Nested Context Language) y de manera imperativa, llamando a este subsistema GINGA-J (Java), que son exigidos en los receptores fijos y portátiles.

GINGA es la capa intermedia de software entre el sistema operativo del Set Top Box y la infraestructura de ejecución; ahora bien al ser de código abierto, presenta las facilidades de que sea independiente de la plataforma y del tipo de receptor.

Las aplicaciones desarrolladas con GINGA residen en la capa Host, pudiendo ser un celular, un set top box o una TV digital.

Para permitir la compatibilidad entre aplicaciones el grupo DVB propone un sistema unificado llamado GEM (Globally Executable MHP), el cual es adoptado por el estándar ISDB-Tb.

La arquitectura del middleware GINGA, está compuesta por el módulo GINGA-NCL, GINGA-J y GINGA-CC (Common Core).

Fig. 6. Arquitectura del Middleware

GINGA-NCL provee soporte para aplicaciones declarativas escritas en NCL, llamadas XML, que provee interactividad, sincronismo, espacio temporal entre objetos de mídia, adaptabilidad, soporte a múltiples dispositivos y soporte a la producción de programas interactivos no-lineales.

GINGA-J provee para aplicaciones imperativas basadas en Java, llamadas Xlet, que tienen como base una máquina virtual de Java, se basa en tres API´s llamados Verde, Amarillo y Azul. Está compuesto por alrededor de 800 mil líneas de código y se caracteriza por ser: multi-red; multi-sistema; y compatible.

GINGA-CC es el subsistema lógico que provee la funcionalidad común al soporte de los ambientes de programación declarativo e imperativo.

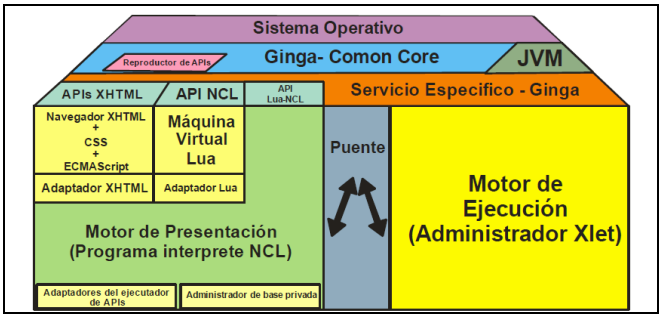

Fig. 7. Arquitectura ampliada del Middleware

GINGA-NCL y GINGA-J se constituyen de los servicios ofrecidos por GINGA-CC, cuya arquitectura se muestra a continuación.

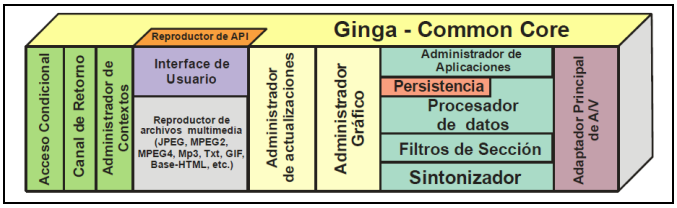

Fig. 8. Arquitectura de GINGA-CC

GINGA-J para mantener la compatibilidad con GEM se basa en los siguientes APIS:

- Verde.- Está compuesto por las APIS compatibles con GEM (Sun JavaTV, DAVIC,DVB y HAVI).
- Amarillo.- Está compuesto del JMF 2.1, que permite trabajar con contenido multimedia.
- Azul.- Define la forma de comunicarse a través de una interfaz común.

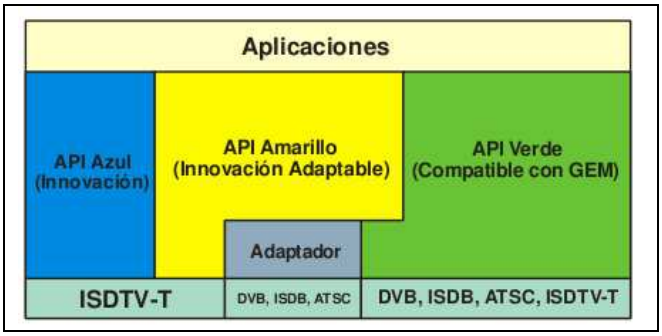

Fig. 9. APIS GINGA-J

El API estándar se ha organizado en los siguientes paquetes:

- Paquete de API JavaTV
- Paquetes de DAVIC
- Paquetes de HAVi
- Paquetes de DVB
- API de control de ajuste
- API de flujos de los medios de comunicación
- API de canal de retorno
- API con la especificación ARIB B-23
- API de integración de los dispositivos
- API de puente Ginga-NCL

JAVATV.- Permite realizar streaming de audio y video; acceso a datos en el canal de transmisión; aplicaciones con interactividad; y gestión del ciclo de vida de las aplicaciones. Proporciona los siguientes paquetes:

- javax.tv.carousel
- javax.tv.graphics
- javax.tv.locator
- javax.tv.media
- javax.tv.net
- javax.tv.service
- javax.tv.util
- javax.tv.xlet

DAVIC.- Proporciona interoperabilidad de extremo a extremo. Proporciona los siguientes paquetes:

- org.davic.media
- org.davic.resources
- org.davic.mpeg
- org.davic.mpeg.sections
- org.davic.net
- org.davic.net.dvb
- org.davic.net.tuning

HAVI.- Permite crear la interfaz de usuario, es una extensión del paquete java.awt, y da soporte para el control remoto. Proporciona los siguientes paquetes:

- org.havi.ui
- org.havi.ui.event

DVB.- Incrementa la funcionalidad de JavaTV, DAVIC y HAVI, como es la información de servicio, intercomunicación entre Xlets, persistencia, etc. Proporciona los siguientes paquetes:

- org.dvb.aplication
- org.dvb.dsmcc
- org.dvb.event
- org.dvb.io.ixc
- org.dvb.io.persistent
- org.dvb.lang
- org.dvb.media
- org.dvb.net
- org.dvb.net.tuning
- org.dvb.net.rc
- org.dvb.test
- org.dvb.ui

El modelo de aplicación Ginga-J debe estar de acuerdo con el modelo de aplicación definido en JAVADTV 1.3:2009, de tal manera que las aplicaciones Ginga-J deben contener una clase implementando la interfaz *javax.microedition.xlet.Xlet*.

Las aplicaciones deben ser ejecutadas en un ambiente orientado a servicios y mantenidas por un gestor de aplicaciones; ya que todo servicio se presenta en un contexto, y la clase que define el contexto es *javax.mcroedition. xlet.ServiceContext.*

Las aplicaciones Ginga-J se pueden controlar tanto por un *zapper (*aplicación residente, desarrollada por el fabricante del set top box*)*, por la emisora, por otra aplicación Ginga-J usando la API "*Application Management and LifeCycle Control*" o por documentos Ginga-NCL.

La invocación de aplicaciones Ginga.J en Ginga-NCL o viceversa, es posible ya que existe un núcleo común, y para lo cual es necesario crear un puente bidireccional:

- En NCL se crea un elemento <link> que referencia a uno <media> que representa a un Xlet.
- En LUA se referencia métodos GINGA-J
- En JAVA se captura un evento NCL.

Cada aplicación Ginga-J debe ser procesada en un ambiente de ejecución aislado, debe haber una entidad de sistema que represente una JVM en donde cada aplicación debe ser ejecutada sin que haya interferencia en la ejecución de cualquier otra aplicación. Para ello, este ambiente de ejecución debe permitir que cada aplicación sea ejecutada por medio de un cargador (*ClassLoader*) propio. Una vez cargada e iniciada una instancia de una aplicación, está prohibida la creación o iniciación de otra instancia de la misma aplicación.

Para poder establecer un canal de retorno para disponer de interactividad total, es necesaria la clase *java.net.Socket* o *java.net.URLConnection*, incluidos en el paquete java.net que se mantiene en Ginga-J.

Las aplicaciones Ginga-J son empaquetadas, autenticadas y autorizadas de acuerdo con las definiciones especificadas en JAVADTV 1.3:2009. Cada aplicación Ginga-J puede contener uno o más archivos JAR. El archivo JAR principal contiene los archivos de clase de la aplicación, archivos de recurso y un archivo *manifest* que describe la aplicación y sus requisitos.

Si la transmisión se hace con el carrusel de objetos, no se puede utilizar el empaquetamiento en un archivo JAR, pero obligatoriamente el sistema de archivos transmitido debe estar organizado de la misma manera.

Si la aplicación es transmitida por el canal de interactividad es obligatorio que ésta sea transmitida en archivos JAR.

A. Interface Xlet

El modelo normal de aplicaciones Java hace una serie de supuestos sobre el ambiente de ejecución que no son compatibles con los set top box, ya que sólo una aplicación se ejecuta en la JVM y que cuando la aplicación deja de funcionar también lo hace la maquina virtual. En un PC, esto no es un problema, pero, si lo es en un sistema en el que no se puede hacer estas suposiciones. El ciclo de vida normal también asume que una aplicación una vez que se carga comienza su ejecución y luego termina, pero esta consideración tampoco es aplicable en la TDT.

El ciclo de vida de los applets de Java en la Web es mucho más adecuado para entender como se va ejecutar una aplicación en la TDT: el navegador web carga un applet de Java en una JVM, la inicializa, y lo ejecuta. Si una página contiene dos subprogramas, los dos se pueden ejecutar en la máquina virtual sin interferir uno con otro. Cuando un applet termina, se retira de la máquina virtual sin afectar ninguna otra aplicación que se ejecuta en la máquina virtual. Debido a que el modelo applet tiene funcionalidades propias de la Web, no es apropiado para todos los casos. El Xlet es la base para todos los sistemas basados en JavaTV. Al igual que los applets, la interfaz Xlet permite una fuente externa para controlar el ciclo de vida de una aplicación, y proporciona a la aplicación una manera de comunicarse con su entorno.

En la creación de un Xlet es necesario implementar la interfaz Xlet que se encuentra en el paquete javax.tv.xlet.

```
public interface Xlet {

        public void initXlet(XletContext ctx)
        throws XletStateChangeException;

        public void startXlet()
        throws XletStateChangeException;

        public void pauseXlet();

        public void destroyXlet(boolean unconditional)
        throws XletStateChangeException;

}
```

Aunque existen algunas similitudes entre un Xlet y un applet, también hay una serie de diferencias. La mayor de ellas es que la ejecución de un Xlet se puede pausar y reanudar; esto debido a que en un entorno como un receptor de televisión digital varias aplicaciones pueden ejecutarse al mismo tiempo, y sin embargo, las restricciones de hardware sólo hacen que una de las aplicaciones pueda ser visible para

el usuario a la vez. Un Xlet es también mucho más simple que un applet.

Los Xlets contienen al menos cuatro métodos que son llamados por la plataforma para informar a la aplicación de cambios de ciclo de vida, los cuales fueron definidos sus prototipos en la interfaz Xlet, antes mencionada.

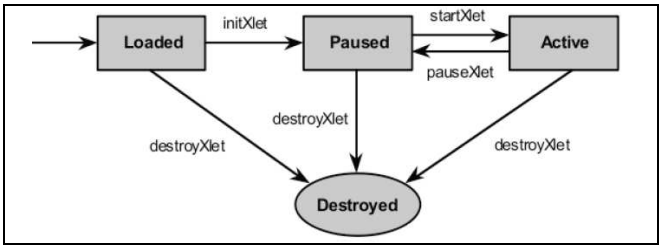

Fig. 10. Ciclo de vida de un Xlet

Después que se obtengan los datos de la aplicación, es creado el objeto que implementa *javax.microedition.xlet.Xlet* usando su constructor. Si el constructor estándar vuelve sin lanzar una excepción, la instancia de la aplicación será considerada como en el estado "Loaded", sino la instancia de la aplicación será considerada como en el estado "Destroyed" y descartada.

Los recursos utilizados por la aplicación son inicializados en el método *initXlet*() y no en el constructor de la clase. Para iniciar la aplicación del método *initXlet* llama a un objeto de tipo *XletContext*, que posee información del contexto de ejecución para la aplicación, incluyendo propiedades y mecanismos para notificación de cambios de estados iniciados por la aplicación, de tal manera que el gestor de aplicaciones pueda cambiar el estado de la instancia de aplicación a "Paused".

El método *startXlet* es llamado para informar a la aplicación que pasa al estado "Active", iniciando su ejecución.

El método *pauseXlet* puede ser llamado para informar a la aplicación que debe pasar al estado "Paused" y que debe minimizar su consumo de recursos. La aplicación puede nuevamente pasar al estado "Active" tras una nueva llamada al método *startXlet*.

Una instancia de aplicación en el estado "Paused" debe reducir su consumo de recursos si tiene la intención de maximizar su probabilidad de supervivencia, ya que no puede mantener todos los recursos por lo escasos que son en los Set Top Box.

El método *destroyXlet* puede ser llamado en cualquier estado y se usa para informar a la aplicación que está lista para terminar su ejecución. La aplicación debe guardar la información de su estado (si fuera posible y necesario) y liberar recursos previamente utilizados lo más rápido posible.

Si un método en la interfaz Xlet lanza la excepción *XletStateChangeException*, el Xlet permanece en el estado en que estaba inmediatamente antes de la llamada que lanzó la excepción. La única excepción a esta regla es el método *destroyXlet.*

V. DESARROLLO DE APLICACIONES CON GINGA - J

Las aplicaciones pueden ser desarrolladas en emisoras de televisión y transmitidas directamente por un canal al Set Top Box; o desarrolladas individualmente, para lo cual es necesario cargar las aplicaciones en el Set Top Box a través de una entrada externa (USB)

Para este caso como la aplicación es desarrollada individualmente, será necesario disponer de un emulador (XletView) y de un IDE (Netbeans) que disponga de las API´s necesarias para ejecutar aplicaciones para TDT.

Para ilustrar el proceso de desarrollo de una aplicación, se trabaja con la presentación de un mensaje simple, para lo cual se procede a crear una aplicación en Netbeans de la misma manera que se crea una aplicación en consola y se referencia los JAR que contienen las API´s para la TDT (javatv.jar y xletview.jar) xletview.jar, es necesario para poder agregar las bibliotecas de MHP y HAVI.

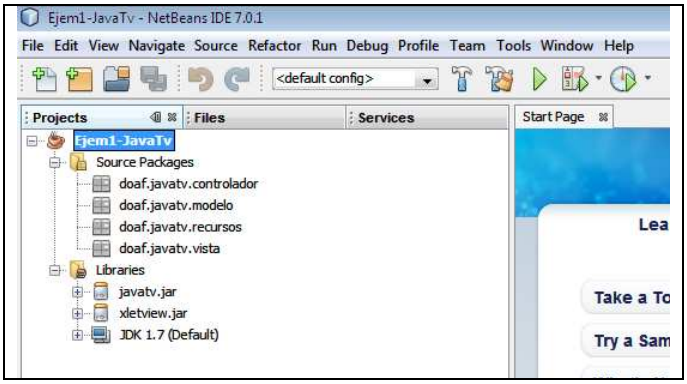

Fig. 11. Referencia a las API´s de Ginga-J

Para el desarrollo de aplicaciones se trabaja con el patrón de diseño MVC (Modelo - Vista- Controlador), con lo que se busca una mayor modularidad de las aplicaciones.

A continuación en el paquete Vista, se crea la clase principal que implementa la interfaz Xlet.

```
package doaf.javatv.vista;

import java.awt.Color;
import java.awt.Font;
```

```
import javax.tv.xlet.Xlet;
import javax.tv.xlet.XletContext;
import javax.tv.xlet.XletStateChangeException;
import org.havi.ui.HScene;
import org.havi.ui.HSceneFactory;
import org.havi.ui.HScreen;
import org.havi.ui.HStaticText;

public class Saludo implements Xlet {

    private XletContext contexto;
    private HScene escena;
    private HStaticText texto;
    private String mensaje;

    @Override
    public void destroyXlet(boolean bln) throws
XletStateChangeException {
    }

    @Override
    public void initXlet(XletContext xc) throws
XletStateChangeException {
        this.mensaje = "DARWIN ALULEMA";
        this.contexto = xc;
    }

    @Override
    public void pauseXlet() {
    }

    @Override
    public void startXlet() throws XletStateChangeException {
        escena =
HSceneFactory.getInstance().getFullScreenScene(

HScreen.getDefaultHScreen().getDefaultHGraphicsDevice()
);
        texto = new HStaticText(mensaje,
escena.getWidth()/3,escena.getHeight()/3,
                escena.getWidth()/3,escena.getHeight()/3);
        texto.setBackground(Color.blue);
        texto.setForeground(Color.red);
        texto.setFont(new Font("Comic Sans
MS",Font.ITALIC,20));
        escena.add(texto);
        escena.setVisible(true);
    }
}
```

Para ejecutar la aplicación en XletView es necesario compilar el proyecto para obtener el .class. En el emulador se crea un nuevo grupo y se carga la aplicación que para el caso particular se llama Saludo.

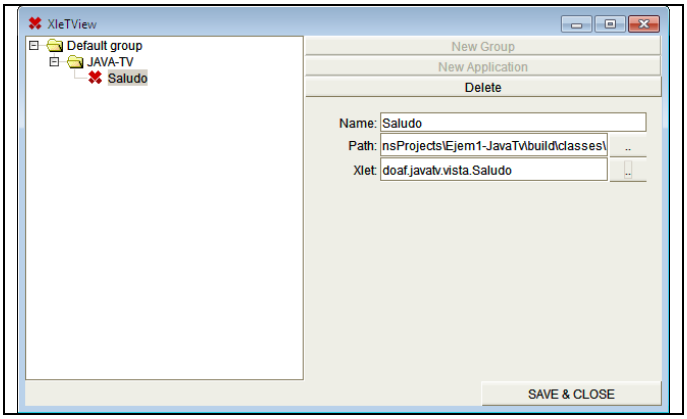

Fig. 12. Aplicación cargada en XletView

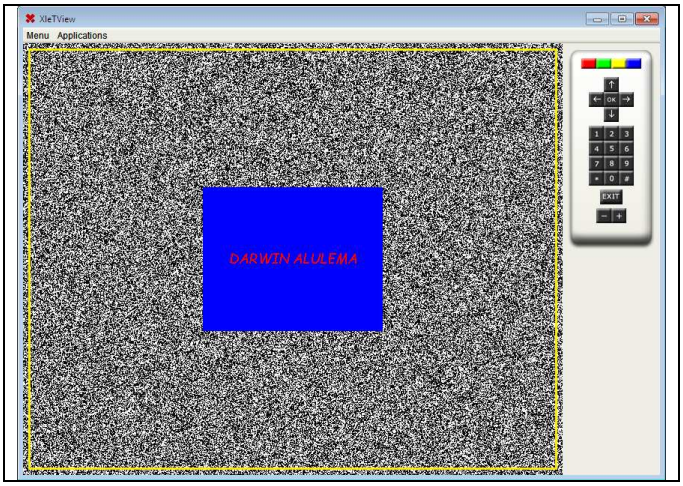

Fig. 13. Ejecución de la aplicación

VI. CONCLUSIONES

El estándar ISDB-Tb, es el resultado de la investigación realiza en Brasil, sobre el estándar japonés, para que sea más adecuado a la región.

GINGA-NCL y GINGA-J, son las posibilidades presentes en el estándar ISDB-Tb para el desarrollo de aplicaciones; el primero es un script, que constituye una de las principales aportaciones al estándar; y el segundo se basa en Java, pero incorpora algunas APIS propias.

Aunque al parecer Ginga-J no ha tenido un mayor desarrollo en el estándar ISDB-Tb, no se debe considerar retirarlo del mismo, ya que provocaría una incompatibilidad con las aplicaciones desarrolladas en los otros middleware existentes para TDT.

Trabajar con Ginga-J para el desarrollo de aplicaciones, tiene la ventaja que los programadores disponen de un lenguaje bastante difundido, aunque la información

específica no sea tan fácil de encontrar, pero permite que un programador con experiencia pueda desarrollar aplicación sin mayores inconvenientes.

El hecho de que Sun Microsystems fuera adquirida por Oracle, ha marcado un punto de tensión con respecto al futuro de Java.

REFERENCIAS

[1] SUPERTEL, Revista Institucional No.3. Ecuador.

[2] SUPERTEL, Revista Institucional No.4. Ecuador.

[3] Llanos, A., Díaz, D. Desarrollo de software Analizador Cabecera de Transport Stream para ISDB-T. INICTEL-UNI.

[4] SUPERTEL, Informe para la definición e implementación de la televisión digital terrestre en el Ecuador. Ecuador.

[5] Universidad Nacional de Ingenieria, Investigación del Estudio del Middleware GINGA y Guía de usuario del Middleware GINGA. Lima, Perú.

[6] Torres, J. ESPE. Curso de Ginga NCL. Quito. Ecuador.

[7] Torres, J. Diseño y desarrollo de una aplicación de contenidos interactivos para TV Digital basada en el middleware Ginga del sistema brasileño. Ecuador. 2010.

[8] Guerra, H. Estudio de factibilidad para la implementación de servicio de Television ISDB-T(Integrated Services Digital Broadcasting-Terrestral) en el Ecuador. Ecuador . 2006.

[9] Armas, F.Análisis de prueba de medición de campo para definir el sistema de TV Digital en el Ecuador. Ecuador. 2008.

[10] Quingaluisa, A, y otros. Estudio e investigación del middleware Ginga-J para el estándar brasileño de televisión digital. Caso práctico: desarrollo de una aplicación interactiva aplicando metodología Openup/Basic.

[11] Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital ASBNT NBR 15606-4

[12] Burlamaqui, y otros. Construcao de programas interactivos para TV digital utilizando o Ginga. http://gingarn.wdfiles.com/local--files/tvdiepoca08/capituloTVDIEPOCAFinal.pdf

[13] Morris, S. Interactive TV Standars. Elsevier

[14] TV Without Borders. http://www.mhp-interactive.org/.