MIDDLEWARE GINGA EN EL DESARROLLO DE APLICACIONES INTERACTIVAS PARA LA TELEVISIÓN DIGITAL TERRESTRE. APLICATIVO: PROTOTIPO DE UN PORTAL INTERACTIVO PARA LA COMPRA Y VENTA DE PRODUCTOS POR TELEVISIÓN

Oscar Rodríguez[1]

[1] Facultad de Ingeniería en Ciencias Aplicadas, Universidad Técnica del Norte, Av. 17 de Julio 5-21, Ibarra, Imbabura

odrodriguez@utn.edu.ec

Resumen. *La señal digital terrestre en el Ecuador, es un tema poco conocido que ha tomado impulso desde abril del año 2009; en este año es cuando se empieza a analizar los diferentes estándares de transmisión y por consiguiente se toma la decisión de adaptar el estándar brasileño ISDB-Tb, que es una variación del estándar japonés. Este proceso de cambio permite que se abran nuevas puertas para los desarrolladores de software, como lo es la programación de aplicaciones para television digital.*

Una de las herramientas que está tomando auge es el GINGA, la cual es un grupo de tecnologías con mejoras brasileñas que enlazan el sistema operativo con la parte de hardware, permitiendo realizar aplicaciones que interactúan con la señal transmitida al televisor, independientemente de la plataforma de hardware que manejen los fabricantes. Además, es un middleware que maneja diversos tipos de lenguajes y librerías de funciones que permiten el desarrollo fácil y rápido de aplicaciones interactivas.

Palabras Claves

Ginga, Middleware, Interactividad, LUA, NCL, JAVA.

Abstract. *The digital terrestrial signal in Ecuador, is a little known theme that has taken impulse since April of 2009; In this year is when it begins to analyze the different standards of transmission and therefore the decision is made to adapt the Brazilian standard ISDB-Tb, which is a variation of the Japanese standard. This process of change allows new doors for software developers, such as the programming of applications for digital television.*

One of the tools that is gaining upgrade is the GINGA, which is a group of technologies with Brazilian improvements that link the operating system with the hardware part, allowing to realize applications that interact with the transmitted signal to the television, independently of the platform of Hardware that manufacturers handle. In addition, it is a middleware that handles different types of languages and function libraries that allow the easy and fast development of interactive applications.

Keywords

Ginga, Middleware, Interactivity, LUA, NCL, JAVA.

1. Introducción

En la provincia de Imbabura no existe la suficiente información acerca del desarrollo de aplicaciones interactiva para la televisión digital terrestre, puesto que la señal digital aún no es accesible en dicha provincia. Este artículo científico pretende brindar una guía básica en el aspecto de aplicaciones interactivas para las generaciones futuras de desarrolladores de software, especialmente con la herramienta ginga ya que permite muchas funcionalidades por medio de sus componentes.

Ginga nace en el país de Brasil como un proyecto por parte de la Pontificia Universidad Católica de Río de Janeiro. Este middleware nace con dicho nombre debido a que ginga es una cualidad única e indescriptible que distingue al pueblo brasileño, además es un movimiento de la tradicional danza capoeira; una danza que nació como una forma de

entretenimiento de los esclavos, sus movimientos son semejantes a una lucha pero sin contacto físico. Ginga nació como software libre por la necesidad de realizar una inclusión social/digital en la totalidad de hogares, permitiendo que todos tengan acceso al aprendizaje, servicios sociales, información, entre otros por medio de su televisor (López & Oleas, 2012, págs. 40-41-42-43).

El middleware Ginga es una capa de software intermediario que permite el desarrollo de aplicaciones interactivas para TDT independientemente de la plataforma del hardware de los fabricantes y terminales de acceso. Da soporte al desarrollo de aplicaciones tanto empleando un paradigma declarativo, imperativo o ambos. Los dos ambientes de ejecución son exigidos en los receptores fijos y portátiles, mientras que solo el ambiente declarativo es exigido en los receptores portátiles (Comunidad Peruana de Ginga, 2015).

En este documento se realiza un estudio del middleware ginga en el desarrollo de aplicaciones interactivas para la televisión digital terrestre por medio del desarrollo de un sistema prototipo cliente-servidor que servirá para demostrar los verdaderos alcances de la herramienta y también comparar los lenguajes de programación JAVA y NCL usados por el middleware.

2. Materiales y Métodos

El middleware ginga se refiere a un software que abarca un conjunto de servicios que permiten la interacción entre diversas aplicaciones, sistemas operativos, hardware, redes, entre otros. La parte esencial de un middleware es que trabaja independientemente de las plataformas, esto se refiere a que puede integrar aplicaciones distribuidas en entornos heterogéneos facilitando el diseño y desarrollo de los sistemas.

En este artículo describe los lenguajes de programación usados por ginga, canal de retorno e interactividad de las aplicaciones.

2.1 Lenguaje NCL

NCL es un lenguaje de tipo declarativo, los cuales se centran en desarrollar el problema planteando una serie de características que indican lo que se debe hacer para llegar a la solución. Una de las ventajas es que la curva de aprendizaje de este lenguaje es muy pequeña puesto que a cualquier programador le resulta fácil acoplarse al modelo de desarrollo de NCL.

La funcionalidad de este lenguaje se divide en objetos media, regiones, descriptores, puertos, Links y conectores; los cuales se observan en la sintaxis mostrada en la figura 1.

Los objetos media se definen como la parte en donde se va mostrar el contenido, este contenido puede ser de audio, video, texto, imágenes, entre otros.

El siguiente paso es definir las áreas en donde se mostrarán los objetos media, estas áreas son definidas por la etiqueta de región.

Los descriptores se encargan en como mostrar los objetos asignados. Esta etiqueta es la encargada de dar parámetros a los medias.

Los puertos permiten iniciar la presentación de los objetos media, además otorga un orden de presentación.

Los Links y Conectores permiten controlar lo que se va a ejecutar por medio de condiciones y acciones.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <ncl id="main" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
        <head>
        region, descriptor, connector
        </head>
        <body>
        media, port, switch, link
        </body>
    </ncl>
```

Figura 1. Sintaxis del Lenguaje NCL
Fuente. CreaTV Digital, s.f.

2.2 Lenguaje LUA

LUA es un lenguaje desarrollado en los laboratorios de la Pontificia Universidad Católica de Río de Janeiro con la principal característica de ayudar a otros lenguajes que surgieron en el mismo sitio como por ejemplo Ginga-NCL. Es por tal razón que LUA no es un lenguaje tan robusto y debe estar embebido en otro para ser compilado.

LUA ha sido utilizado en algunas áreas de la programación como por ejemplo la creación de videojuegos o la robótica porque es un lenguaje que puede funcionar en diversos sistemas y además es de código libre.

Para que las conexiones entre LUA y NCL sean más fáciles, se desarrollaron nuevas funcionalidades por parte de LUA como es el caso del objeto NCLua que también tiene varios eventos que pueden ser accionadas desde el control remoto de algún televisor. De esta forma se logra reforzar la parte de la vista que es manejada por NCL.

Existen tres principales módulos que ayudan a potenciar la interactividad, el primero es el módulo Event permite que NCLua genere eventos de control remoto o de interactividad en el documento LUA.

El segundo es el módulo settings que permite interactuar con el documento NCL por medio de la propiedad “application/x-ncl-settings” que se declara en la vista de la aplicación interactiva.

El tercero es el módulo canvas que brinda funcionalidades para los sistemas de televisión digital interactivos permitiendo graficar y crear nuevos objetos para ser mostrados en alguna región asignada en NCLua.

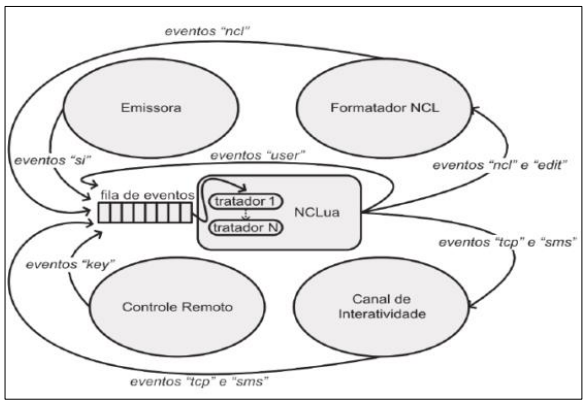

Figura 2. Programación Orientada a Eventos
Fuente. (Riberi F. , 2012)

2.3 Lenguaje JAVA

Este es uno de los lenguajes con sus bases bien cimentadas, puesto que el lenguaje java no es nuevo, sus inicios empiezan a principios de los 90 a cargo de la empresa Sun Microsystems; después de un tiempo esta empresa sería comprada por Oracle Corporation. Para entender a java como lenguaje de programación, antes se debe conocer conceptos básicos de dicha tecnología.

Esta tecnología se ha ganado un amplio lugar en el mercado del desarrollo de sistemas, ya que ha sido la base para proyectos muy grandes. En la actualidad, es prioridad instalar Java Virtual Machine (JVM), para poder ejecutar los sistemas desarrollados con la tecnología java.

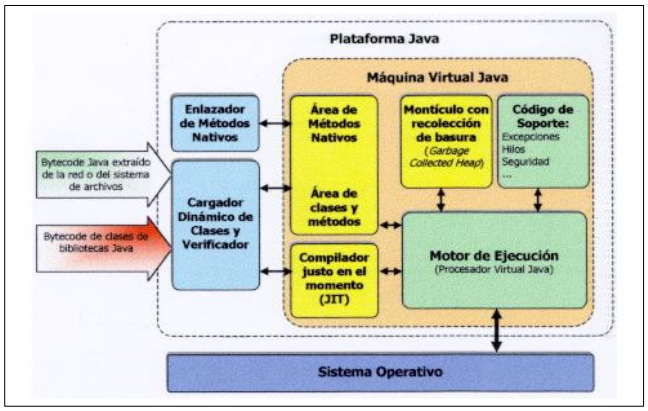

Figura 3. Arquitectura del Sistema de Tiempo de Ejecución Java
Fuente. (Menchaca & García, s.f.)

2.4 Creación del Sistema en el Servidor

Tal como se mencionó anteriormente, el prototipo cuente con dos subsistemas, en el servidor y en el cliente respectivamente.

El Sistema en el servidor será desarrollado con lenguaje JAVA, a su vez se debe crear los servicios web que serán consumidos por el cliente, estos será desarrollados por medio del protocolo SOAP.

Para el desarrollo del Sistema primero se generan las entidades que provienen de una base de datos Postgresql.

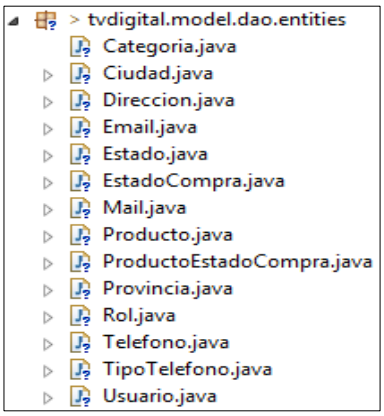

Figura 4. Paquete de Entidades

El siguiente paso es desarrollar el manejador, el cual será el encargado de realizar las consultas directas a la base de datos por medio del lenguaje JPQL.

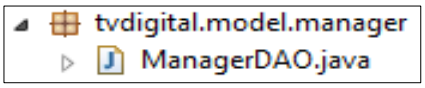

Figura 5. Paquete Manager

El siguiente paquete de clases viene compuesto por los Bean Manejados, los cuales son objetos que posteriormente serán llamados desde las páginas xhtml.

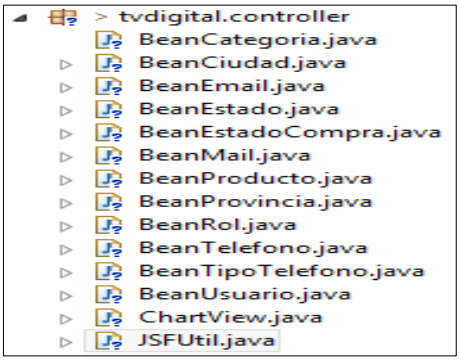

Figura 6. Bean Manejados y Clases Util

Debido a que el prototipo será desarrollador con la arquitectura de cuatro capas, los servicios web son los que componente la capa de Servicios que posteriormente publicará los métodos para que puedan ser consumidos por el cliente en el televisor digital o decodificador.

aprobar()	guardarApellido(String)
cambioEstado(String)	guardarCelular(String)
cancelar()	guardarContrasenia(String)
cancelarProducto(int)	guardarContraseniaTemp(String)
cargarUsuario()	guardarDireccion(String)
crearUsuario()	guardarEmail(String)
eliminarProducto(int)	guardarIdCiudad(int)
erroresCrearUsuario(String)	guardarIdProvincia(int)
generarNombreUsuario()	guardarNombre(String)
getAcceso(int)	guardarNombreUsuario(String)
getCiudades()	guardarRepitaContrasenia(String)
getComprobarAcceso(String, String)	guardarTelefono(String)
getEstado()	guardarUsuarioTemp(String)
getEstadoCliente()	listaCiudades()
getEstadoError()	listaCiudadesByProvincia(int)
getIdProvincia()	listaProvincias()
getListaProductos()	obtenerNombreUsuario()
getListaProductosUsuario()	vaciarAcceso(int)
getTamanioObjErrores()	vaciarUsuario(int)
getUsuarioTemp()	venderProducto(int)

Figura 7. Servicios Web Utilizados En el Prototipo de Compra y Venta de Productos Por Televisión Digital.

2.5 Alojamiento del sistema en el Servidor EC2 de Amazon

Para la publicación de un web service en la plataforma de Amazon EC2 se configuró una instancia con el sistema operativo Linux Ubuntu 14.04 LS.

Una vez instalado el sistema operativo se procede a crear un grupo de seguridad para la instancia en la cual se abre los siguientes puertos de salida y entrada para la máquina virtual de Ubuntu:

- 80 (Puerto TCP)
- 8080 (Puerto TCP para tomcat)
- 5432 (Puerto de Postgresql)
- 22 (SSH para acceso mediante PUTTY)
- 443 (HTTPS)

Puesto que el sistema operativo instalado en la instancia solo permite su acceso por consola, debemos establecer un acceso remoto por medio de la herramienta putty en la cual ingresamos al acceso público de la instancia creada, a continuación se detallan las características de acceso:

- Ubuntu (usuario)
- 54.91.186.81 (ip pública para el acceso)

Además, la instancia debe tener una Key pública generada para el acceso remoto, para adquirir dicha clave, ingresamos a la instancia creada y se escoge la opción "Connect" para que luego se genere la clave por medio de una línea de comandos en Linux, a continuación se muestran los comandos que permiten generar una key:

- ssh -i "ginga.pem" ubuntu@ec2-54-91-186-81.us-west-2.compute.amazonaws.com

Ahora se debe generar un archivo con extensión ".ppk" para que el programa putty pueda leerlo como una key. Para generar el archivo se utiliza el programa "puttygen" en el cual se ingresa el archivo con extensión ".pem" para generar una key privada con la extensión ".ppk"

Luego se crea una aplicación java con la ayuda del IDE eclipse en la cual se realizará un web service localmente. Gracias a las facilidades que ofrece AWS Amazon con su servicio IaaS, solo hay que comprimir la aplicación desarrollada localmente y copiarla en el directorio de tomcat, por lo general es /var/lib/tomcat7/webapps.

Para acceder a la aplicación debemos hacerlo por medio del puerto 8080, tal como se muestra en la siguiente dirección:

- http://54.91.186.81:8080/tvdigital/faces/index.xhtml

Debido a que el web service fue generado en el lenguaje java es imprescindible que se configure el contenedor de servlets Tomcat para la publicación por medio de la ip pública de la instancia creada en EC2 como se vio anteriormente.

Figura 8. Pantalla Principal Del Sistema

2.6 Creación del Sistema para el Cliente

Para consumir los servicios web, será necesario implementar una clase con el lenguaje LUA. Esta clase permitirá el consumo de los métodos publicados. Esto se lo realiza por medio de funciones que se ejecutan en el lado del cliente. Esta clase utiliza la librería ncluasoap para llamar a funciones que permiten la conexión con los servicios web.

```
local msgTable = {
  address = "http://52.33.124.178:8080/tvdigital/services/Servicios",
  namespace = "http://controller.servicios",
  operationName = "getListaProductos"
}
local soapVersion = "1.1"
ncluasoap.call(msgTable, respuesta, soapVersion)
```

Figura 9. Consumo de un servicio web en LUA

Una vez realizada la conexión a los web services, la clase LUA se integra con la clase NCL para mostrar los resultados por medio de la etiqueta <media>.

```
<media id="settings" type="application/x-ginga-settings">
    <property name="service.currentKeyMaster" value="luaIdx"/>
</media>
<media id="lua" src="ServiciosJava.lua" descriptor="dLua" />
```

Figura 10. Conexión NCL y LUA

3. Resultados

El resultado de esta investigación fue exitoso puesto que se realizó la conexión del servidor con el cliente por medio de la capa de servicios. Existieron inconvenientes en las pruebas con el decodificador físico pero en el decodificador virtual se logró realizar un sistema interactivo con casi todas las funcionalidades.

Figura 11. Pantalla Principal del Sistema para el Cliente

A partir del desarrollo de este prototipo se obtuvo información acerca de las herramientas que componen el middleware ginga.

Prioridad	Herramienta	Lenguaje Ginga	
		NCL	JAVA
1	Documentación		
	Cantidad de Documentación en Portugués	4,5	2
	Cantidad de Documentación en Español	3	2,5
	Cantidad de Documentación en Inglés	2	2
2	Ambientes de Desarrollo	5	5
3	Compatibilidad con S. O. (Linux, Microsoft)	5	5
4	Emuladores	4,5	3
5	Facilidad de Instalación	4	2
TOTAL		**28**	**21,5**
PORCENTAJE DE RENDIMIENTO		**80%**	**61,43%**

Tabla 1. Tabla Comparativa

También se realizó un análisis de costo-beneficio en base a los dispositivos utilizados. La siguiente fórmula extrae el costo real de la investigación dependiendo del dispositivo utilizado.

$$CRD = CR + D \quad (1)$$

Donde:

- CRD: Costo real excluyendo los precios del televisor digital, decodificador y decodificador virtual.
- D: Dispositivo (Decodificador, Decodificador Virtual, TV)
- CRD: Costo Real por Dispositivo (Decodificador, Decodificador Virtual, TV)

Figura 12. Valor del costo real dependiendo del dispositivo usado

Dispositivo	Costo Real Inicial ($)	Costo Del Servidor ($/MES)	Mantenimiento de Web Services ($/MES)	Memoria Interna del Dispositivo (gb)
Televisor	1205	11,65	5	0
Decodificador Físico	2205	11,65	5	2
Decodificador Virtual	705	11,65	5	0,5

Tabla 2. Tabla de valores iniciales

VALOR DE LAS CARACTERÍSTICAS	VMAX	PMAX	PM
Costo Real Inicial	2205	10	3,1
Costo del Servidor ($/MES)	11,65	10	10
Mantenimiento de Web Services	5	10	10
Memoria Interna del Dispositivo (gb)	2	10	0

Tabla 3. Valor de las Características

Dispositivos	Costo Real	Costo del Servidor ($/MES)	Mantenimiento de Web Services ($/MES)	Memoria Interna Del Dispositivo (gb)	Beneficio Total	Precio Del Dispositivo ($)	Costo/ Beneficio
Televisor	5,4	10	10	0	25,4	500	0,2508
Decodificador Físico	10	10	10	10	40	1500	0,093333
Decodificador Virtual	3,1	10	10	2,5	25,6	1	125,6

Tabla 4. Análisis del Costo-Beneficio

El análisis del costo beneficio se lo realizó en base a algunos factores que fueron los más relevantes en el caso de estudio.

Una vez encontrado los valores iniciales de cada dispositivo, se procedió a calcular los valores máximos y mínimos de cada característica para formar la tabla 2 con la siguiente fórmula:

$$PM = (VMIN * PMAX)/VMAX \quad (2)$$

Donde:

- PM: Es el puntaje mínimo redondeado al menor.
- VMIN: El valor mínimo de cada columna de la tabla 1 de valores iniciales
- PMAX: Puntaje máximo que se otorga, en este caso será una constante con valor de 10
- VMAX: El valor máximo de cada columna de la tabla 1 de valores iniciales

Finalmente se forma la tabla 3 de análisis de costo-beneficio por medio de las siguientes fórmulas:

$$CR = (V * PMAX)/VMAX \quad (3)$$

Donde:

- CR: Es el puntaje del costo real.
- V: valor de cada dispositivo
- PMAX: Puntaje máximo que se otorga, en este caso será una constante con valor de 10
- VMAX: El valor máximo de cada columna de la tabla 1 de valores iniciales.

$$CB = (BT * 100)/V \quad (4)$$

Donde:

- CB: Es el valor del costo-beneficio.
- BT: Es el beneficio total que resulta de la suma de las características
- 100: Constante
- V: Es el valor de cada Dispositivo.

Como se muestra en la tabla 3, el costo por beneficio más alto para el desarrollo del prototipo es con el decodificador virtual.

4. Conclusiones

El middleware ginga es una herramienta con grandes capacidades para el desarrollo de sistemas interactivos pero aún no es muy bien conocido por los desarrolladores de software puesto que la señal digital en el Ecuador aún no ha cubierto en su totalidad la región y solo existe en las principales ciudades.

La compatibilidad de ginga con televisores importados es muy buena, porque la mayoría de estos cumplen con las características mínimas que necesita dicho middleware.

Los lenguajes NCL y JAVA usados por el middleware ginga permiten facilidades al momento de realizar aplicaciones interactivas, pero la documentación no es suficiente porque en el Ecuador es una tecnología nueva pero en otros países se están implementando aplicaciones con otras herramientas y ginga ha dejado de evolucionar.

Existen varios inconvenientes en la realización de una comparativa entre NCL y JAVA puesto que estos dos lenguajes brindan potencialidades extraordinarias trabajando junto, pero por separado tienen inconvenientes semejantes. El estudio puede generar mejores resultados cuando exista la señal digital en la provincia de Imbabura y a su vez en todo el Ecuador.

La aplicación web realizada con JSF generó algunos inconvenientes que a la par fueron solucionados, puesto que el contenedor de servlets tomcat necesita configuraciones extras para publicar un web service desde un servidor en la web

El servidor de base de datos Postgresql es perfecto para este tipo de aplicaciones porque tiene características muy buenas y no producen inconvenientes con el tamaño de los datos, aunque estos sean muy grandes. Además, la plataforma de AWS Amazon, utilizada para el alojamiento del aplicativo web, es compatible con Postgresql

Los web services generados por SOAP fueron muy fáciles de implementarlos pero su funcionamiento es muy básico puesto que los métodos publicados retornan un arreglo más no un objeto como tal.

El lenguaje NCL conjuntamente con LUA y JAVA permitió realizar un prototipo de compra y venta de productos por televisión que interactúa con el usuario televidente, pero con la evolución de la tecnología aparecieron nuevos lenguajes que permiten un desarrollo más eficientes y mejor.

Los tiempos que se necesitan para realizar un sistema interactivo para televisión digital son muy amplios. Un inconveniente que genera contratiempos y amplía el plazo de entrega del sistema es la capacitación, puesto que la mejor información se encuentra en portugués, además no existe suficiente documentación de fuentes confiables.

Aunque el lenguaje NCL es fácilmente comprensible y permite realizar la interactividad en el televisor digital, no puede valerse por sí solo, se debe aprender otro lenguaje como lo es LUA para las conexiones hacia el aplicativo desarrollado en JSF.

En conclusión, los sistemas interactivos son el nuevo camino que deberán seguir los desarrolladores de software porque la televisión digital aún sigue siendo una de las principales fuentes de entretenimiento de las personas a nivel mundial, no es la única pero si una de las más usadas.

Agradecimientos

Un agradecimiento muy especial a todos los integrantes de mi familia porque me brindan su compañía y apoyo todo el tiempo.

A la Universidad Técnica del Norte porque sus aulas me inculcaron valores que aplicaré en el ámbito personal y familiar.

A todos los ingenieros que fueron mis profesores en las diferentes materias porque de ellos aprendí todas las cosas que hoy aplicaré en el ámbito profesional.

Al Ing. Marco Pusdá por su guía y apoyo para el desarrollo y finalización de esta investigación.

Al Ing. José Luis Rodríguez por su apoyo en el inicio de esta investigación

Finalmente a mis amigos con los cuales compartí y seguiré compartiendo muchas experiencias que nunca olvidaré.

Referencias Bibliográficas

[1] Albornoz Luis A., G. L. (2012). La Televisión Digital Terrestre. Experiencias nacionales y diversidad en Europa, América y Asia. Buenos Aires: LA CRUJIA.

[2] Albornoz, L. A., & García Leiva, M. T. (2 de 6 de 2015). *Universidad Carlos III de Madrid.* Obtenido de http://e-archivo.uc3m.es/handle/10016/14621

[3] Alcalde, P. (2014). *Electrónica.* Ediciones Paraninfo S.A.

[4] Amazon Company. (15 de 12 de 2015). *amazon web services*. Obtenido de https://aws.amazon.com/es/ec2/

[5] Ayala, A. (25 de Agosto de 2014). *Televisión Digital Terrestre, Middleware Ginga*. Obtenido de null pointer exception apuntes, opinion y mas..: http://alejandroayala.solmedia.ec/?p=1774

[6] Baum, G., & Soares, L. (2012). *Ginga Middleware and Digital TV in Latin America*. doi:10.1109/MITP.2012.78

[7] Borranis Bureau Consultores. (6 de Septiembre de 2012). *#Ginga, para potenciar la #TV digital en #Sudamérica*. Obtenido de Borranis Bureau Consultores. Las últimas novedades del mercado, para emprendedores, profesionales y empresarios: http://news.borranisbureau.com/2012/09/06/ginga-para-potenciar-la-tv-digital-en-sudamerica/

[8] Cabezas, G., & Quezada, M. (Noviembre de 2012). Diseño e Implementación de un prototipo para un sistema de generación de aplicaciones interactivas con ginga-ncl para la evaluación de servicios masivos. Quito. Recuperado el 3 de Febrero de 2016, de http://bibdigital.epn.edu.ec/handle/15000/5167

[9] Comunidad Peruana de Ginga. (03 de 12 de 2015). Obtenido de Comunidad Ginga Perú: http://www.gingaperu.org/

[10] Coromina, Á. P.-U. (2005). *TV Digital e integración, ¿televisión para todos?* Librería-Editorial Dykinson, 2005.

[11] El Diario Manabita. (29 de Abril de 2015). Obtenido de El Diario Manabita de libre pensamiento: http://www.eldiario.ec/noticias-manabi-ecuador/264426-apagon-en-5-anos-mas/

[12] Espinoza, T., & Andrés, G. (2014). *Actualización de datos desde carrusel en aplicaciones GINGA Informe de Proyecto*. Recuperado el 12 de Febrero de 2016, de http://www.electronica.utfsm.cl/

[13] ginga.org. (18 de Enero de 2015). *Ginga*. Obtenido de Ginga: http://www.ginga.org.br/es/inicio

[14] Goette, E. (15 de 12 de 2015). *Emanuel Goette, alias Crespo*. Obtenido de http://emanuelpeg.blogspot.com/2012/05/eucalyptus-systems-y-nebula-nubes.html

[15] Google Sites. (17 de Enero de 2015). *Roles: Metodología XP*. Obtenido de Metodología XP: https://sites.google.com/site/xpmetodologia

[16] Google Sites. (s.f.). *Metodología XP*. Recuperado el 5 de 11 de 2015, de https://sites.google.com/site/xpmetodologia/marco-teorico/roles

[17] Instituto Nacional de Estadística y Censos [INEC]. (s.f.). *INEC*. Recuperado el 13 de Febrero de 2016, de http://www.ecuadorencifras.gob.ec/

[18] Jaramillo, A. (1 de Noviembre de 2014). *Tendencias: Diario El Comercio*. Obtenido de El Comercio.com: http://www.elcomercio.com.ec/tendencias/software-ginga-television-digital-ecuador.html

[19] Kazancigil, M. (2013). Xlet-based applications for seismic early warning and Emergency Services in the IDTV environment. doi:10.1109/CTS.2013.6567241

[20] Kezherashvili, B. (s.f.). *Computación en la Nube*. Almería.

[21] La Hora. (29 de Abril de 2015). Obtenido de La Hora: http://www.lahora.com.ec/index.php/noticias/show/1101516289#.VUGSjSGqqko

[22] La Televisión Digital: Fundamentos y teorías (Vol. I). (2009). Barcelona: Marcombo.

[23] Laboratotio de Investigación y Formación en Informática Avanzada. Facultad de Informática. UNLP. (2 de Enero de 2015). *Lifia*. Obtenido de Lifia: http://tvd.lifia.info.unlp.edu.ar/ginga.ar/

[24] López, M., & Oleas, K. (2012). Estudio metodológico para el diseño de interfaces entre el PC y el usuario utilizando ISDB.Tb y Middleware Ginga. Obtenido de http://dspace.espoch.edu.ec/handle/123456789/2912

[25] Menchaca, R., & García, F. (s.f.). *Revista Digital Universitaria*. Recuperado el 13 de Febrero de 2016, de http://www.revista.unam.mx/vol.1/num2/art4/

[26] Microsoft. (2016). *Microsoft Developer Network*. Obtenido de https://msdn.microsoft.com/es-es/library/bb669144.aspx

[27] Ministerio de Telecomunicaciones y Sociedad de la Información. (2014). *Ministerio de Telecomunicaciones y Sociedad de la Información*. Obtenido de Ministerio de Telecomunicaciones y Sociedad de la Información: http://www.telecomunicaciones.gob.ec/television-digital-terrestre-en-el-ecuador/

[28] Ministerio de Telecomunicaciones y Sociedad de la información. (2015). *Ministerio de Telecomunicaciones y Sociedad de la información*. Obtenido de http://www.telecomunicaciones.gob.ec/television-digital-terrestre-en-el-ecuador/

[29] Ministerio de Telecomunicaciones y Sociedad de la Información. (25 de 05 de 2015). *Ministerio de Telecomunicaciones y Sociedad de la Información*. Obtenido de http://www.telecomunicaciones.gob.ec/

[30] Oracle Corporation. (25 de 11 de 2015). *Java*. Obtenido de https://www.java.com/es/about/whatis_java.jsp

[31] Oracle Corporation. (s.f.). *Oracle*. Recuperado el 13 de Febrero de 2016, de http://www.oracle.com/technetwork/java/javaee/overview/index.html

[32] Ordax Cassá, J. M. (2012). *Programación web en java*. España: Ministerio de Educación de España. Obtenido de http://www.ebrary.com

[33] Ordoñez, J. L. (2011). *TDT: Televisión Digital Terrestre (Ciencia Divulgativa)*. Barcelona: Creaciones Copyright.

[34] Paredes, A., Tonguino, N., Olmedo, G., & Acosta, F. (2012). *Performance analysis on return channel for interactive digital TV ISDB-Tb system*. doi:10.1109/LATINCOM.2012.6505993

[35] Quingaluisa, Á., Torres, J., Martínez, D., & Salvador, S. (2010). Estudio e Investigación del Middelware Ginga-J del estándar Brasileño de televisión digital . Caso Práctico : Desarrollo de una aplicación interactiva aplicando la metodología OpenUP / Basic como parte del Proyecto. Recuperado el 15 de Enero de 2016, de http://repositorio.espe.edu.ec/handle/21000/4748

[36] Riberi, F. (2012). *Un Prototipo de Desarrollo NCL para la Plataforma de Televisión Digital*. Recuperado el 5 de Febrero de 2016, de http://41jaiio.sadio.org.ar/sites/default/files/19_SSI_2012.pdf/

[37] Riberi, F. G. (2012). Un Prototipo de Desarrollo NCL para la Plataforma de Televisión Digital. Recuperado el 6 de Febrero de 2016, de http://41jaiio.sadio.org.ar/sites/default/files/19_SSI_2012.pdf

[38] Toledo Freitas, B., Susin, A., & Bonatto, A. (2014). *Ginga MiddleWare on a SoC for Digital Television Set-Top Box*. doi:10.1109/LASCAS.2014.6820290

[39] Torres, J. (28 de Junio de 2011). *Comunidad Ginga Ecuador*. Obtenido de http://comunidadgingaec.blogspot.com/2011/06/middleware-ginga.html

[40] Unión Internacional de Telecomunicaciones. (2013). *Medición de la Sociedad de la Información*. Recuperado el 13 de Febrero de 2016, de http://www.itu.int/en/ITU-D/Statistics/Pages/publications/mis2013.aspx

[41] UTFSM. (15 de Enero de 2015). *Ginga y TVD: Tarea Redes de Computadores II*. Obtenido de Ginga y TVD: http://www2.elo.utfsm.cl/~elo323/prep.html

[42] VV.AA. (2013). Estructura y Tecnología de Computadores I (Gestión y Sistemas). UNED.

[43] Wikipedia. (15 de 12 de 2015). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Eucalyptus_(inform%C3%A1tica)

Sobre los Autores

Oscar Daniel Rodríguez Ortega. Es egresado de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte. Inició sus estudios en la Universidad Técnica del Norte a partir del año 2008. Culminó la secundaria en el año 2007 obteniendo el título en Físico Matemático.