ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

ESTUDIO E INVESTIGACIÓN DEL MIDDLEWARE GINGA J DEL ESTÁNDAR BRASILEÑO DE TELEVISIÓN DIGITAL. CASO PRÁCTICO: DESARROLLO DE UNA APLICACIÓN INTERACTIVA APLICANDO METODOLOGÍA OPENUP/BASIC COMO PARTE DEL PROYECTO ESPE – GINGA.

Previa a la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMÁTICA

POR:

ÁNGEL LEONARDO QUINGALUISA QUISPE

JONATHAN ALBERTO TORRES BELTRÁN

SANGOLQUÍ, 28 DE OCTUBRE DEL 2011

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por los Srs. Ángel Leonardo Quingaluisa Quispe y Jonathan Alberto Torres Beltrán como requerimiento parcial a la obtención del título de Ingenieros en Sistemas e Informática.

Sangolquí, 28 de Septiembre del 2011

_____

Ing. Danilo Martínez

DEDICATORIA

Este proyecto de tesis está dedicado, a mis padres que a pesar de todo lo que paso en nuestras vidas salimos adelante, Ellos que con esfuerzo me sacaron adelante y sin importarles las cosas del pasado siempre están para apoyarme.
A mis hermanos Jorge y Danny que siempre están para regalarme una alegría y una bendición.
A mi mujer y a mis dos pequeños hijos que me regalan su amor y me dan las fuerzas necesarias para seguir cumpliendo nuestras metas.
Esto va dedicado a todos mis amigos con los que viví en las calles, para los que no lograron dejar el camino del mal y a los que en este momento siguen ahí.

ÁNGEL LEONARDO QUINGALUISA QUISPE.

AGRADECIMIENTOS

A Dios que quien me da fuerza y vida, me guía para seguir adelante.
A mis padres los únicos que creyeron en mí y nunca dejaron de luchar por nuestra familia.

A todas las personas que me conocen, amigos primos, hermanos y en especial a mis dos hijos que son la alegría de mi vida y las personas por las que tengo que luchar.

ÁNGEL LEONARDO QUINGALUISA QUISPE

DEDICATORIA

Este proyecto de tesis está dedicado, a toda mi familia, quienes han estado conmigo en momentos difíciles brindándome siempre su apoyo incondicional, especialmente a mi madre de la cual he recibido todo el cariño y confianza del mundo para salir adelante y sin ella cual no sería la persona que soy.

JONATHAN ALBERTO TORRES BELTRÁN.

AGRADECIMIENTOS

A Dios por todo lo que me ha dado en el transcurso de mi vida.
A mi familia que ha sido la razón por la cual se ve reflejado mi esfuerzo.
Al Ing. Danilo Martínez que confió en mis capacidades y me dio la oportunidad de hacer posible la realización de este proyecto de tesis y así lograr culminar con éxitos esta etapa de mi vida.
.

JONATHAN ALBERTO TORRES BELTRÁN

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

El proyecto ESPE-GINGA como parte de la Red Latinoamérica de Cooperación en Investigación, Desarrollo y Formación en el área de Software para TV Digital Interactiva, busca como objetivo difundir y motivar el desarrollo de contenidos interactivos para la TV digital en Latinoamérica.

El Departamento de Ciencias de la Computación de la Escuela Politécnica del Ejército para apoyar al proyecto de implementación del Laboratorio de TV Digital Interactiva del Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejército, decide iniciar la investigación del middleware Ginga-J.

El presente trabajo propone la investigación y creación de una aplicación interactiva con el estándar Brasileño de TV Digital. Para llevar a cabo el proyecto se realizó la investigación de la arquitectura y funcionamiento del MiddelwareGinga-J, además se utilizó el emulador OpenGinga, la arquitectura MVC, metodología OpenUP / Basic, lo que ha permitido obtener como resultado una aplicación interactiva digital cuyo contenido se alimenta de imágenes y archivos XML, demostrando la potencia que tiene el APIGinga-J.

CAPÍTULO I
PLAN DE TESIS

TEMA:

ESTUDIO E INVESTIGACIÓN DEL MIDDLEWARE GINGA J DELESTÁNDAR BRASILEÑO DE TELEVISIÓN DIGITAL. CASO PRÁCTICO: DESARROLLO DE UNA APLICACIÓN INTERACTIVA APLICANDO METODOLOGÍA OPENUP/BASIC COMO PARTE DEL PROYECTO ESPE – GINGA.

1.1. INTRODUCCIÓN

ISDB-T (“IntegratedService Digital Broadcasting – Terrestrial” – Transmisión Digital de Servicios Integrados – Terrestre) es un conjunto de normas que ha sido desarrollado en Japón y sub desarrollado por Brasil para las transmisiones de radio digital y televisión digital.

Ecuador decidió escoger el estándar tecnológico japonés-brasileño para la aplicación de la Televisión Digital Terrestre en el país, lo que se oficializó en el año 2010 por parte de la Superintendencia de Telecomunicaciones, el Consejo Nacional de Telecomunicaciones (Conatel) aceptó la recomendación de la Superintendencia de Telecomunicaciones que se inclinó por la norma japonesa-brasileña de televisión digital ISDB-TB/SBTVD siendo en consecuencia adoptada como norma de televisión digital terrestre en Ecuador.

El Departamento de Ciencias de la Computación en conjunto con el Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejército ha decidido impulsar la investigación y desarrollo de aplicaciones interactivas para televisión digital utilizando el middleware GINGA como parte de un proyecto de tesis de la Carrera de Ingeniería en Sistemas – ESPE.

1.2. PLANTEAMIENTO DEL PROBLEMA

En el Ecuador, las estaciones de televisión locales entregan su programación con el sistema analógico NTSC (National Televisión SystemComitte).

Después de medio siglo de vida, la televisión analógica tradicional en Ecuador entrará a una etapa de televisión digital, que conlleva una mejora en la recepción de la señal de televisión, optimizando el uso del espectro radioeléctrico y aportando una mayor calidad de imagen y sonido, facilita igualmente el acceso a la televisión multicanal y promueve la irrupción de los servicios de la Sociedad de la Información que pueden ser recibidos a través de la propia pantalla del televisor.

Migrar de una tecnología a otra implica fuertes inversiones por parte de las empresas televisoras ecuatorianas. Se espera que el Gobierno haya concretado, junto con el formato digital, las facilidades de financiamiento con Japón y Brasil para aprovechar de forma más eficiente el espectro actualmente utilizado por la televisión analógica, por ampliar la oferta de canales, y por impulsar los nuevos servicios y facilidades que podrá ofrecer la televisión digital; actualmente no existe ninguna empresa encargada del desarrollo de aplicaciones interactivas para televisión digital en el Ecuador ya que los middleware están en la etapa de crecimiento y es casi nula la investigación que existe en este campo.

El proyecto ESPE–GINGA, actualmente está orientado a la investigación de aplicaciones interactivas desarrolladas con el middleware GINGA NLC, pero todavía desconocen el funcionamiento del recién creado GINGA J, por tal razón es la oportunidad de interactuar entre las dos herramientas para obtener un CORE realmente potente que significaría en la ESPE, ser pionero en el desarrollo y utilización de este tipo de aplicaciones en el país.

1.3. **JUSTIFICACIÓN**

Actualmente no existen empresas Ecuatorianas que se dediquen a la investigación de nuevas tecnologías de desarrollo de aplicaciones para Televisión digital, es por eso que en la actualidad sería muy costoso realizar aplicaciones interactivas para este medio de comunicación.

GINGA NLC actualmente es la herramienta más utilizada en el desarrollo de aplicaciones interactivas de TV Digital del estándar Brasileño, pero no es el único middleware que existe para dicho estándar.

ESPE-GINGA es un proyecto que se encuentra en la etapa de desarrollo, la cual consta de extensas cantidades de información sobre GINGA-NLC, pero la investigación del middleware GINGA J muy deficiente actualmente.

Por tal razón este proyecto está dedicado a la investigación y análisis en el emulador de TV de GINGA J, y que de tal manera poder encontrar, debilidades, fortalezas, amenazas y la oportunidad de poder trabajar conjuntamente entre las dos herramientas para lograr así un desarrollo de aplicaciones mucho más compleja.

1.4. OBJETIVOS

1.4.1. Objetivo General

Investigar y desarrollar una aplicación interactiva bajo el MiddellwareGinga J, utilizando la metodología OpenUP / Basic para el estándar brasileño de televisión digital.

1.4.2. Objetivos Específicos

- Investigar la arquitectura y funcionamiento del Middleware Ginga Java.
- Estudio y funcionamiento de los componentes gráficos para el desarrollo de una aplicación de tipo Xlet.
- Desarrollar una aplicación interactiva, aplicando la metodología OpenUP / Basic.

1.5. ALCANCE DEL PROYECTO

Para que esta investigación solvente la necesidad de conocimiento sobre el funcionamiento del Middleware Ginga J, se realizara lo siguiente.

- Investigar las plataformas con las que trabaja el Middleware Ginga J.
- Analizar la arquitectura Ginga J.
- Desarrollar una aplicación interactiva utilizando el Middleware Ginga J, aplicando la metodología OpenUp / Basic.
- Realizar simulaciones de la aplicación desarrollada, en el emulador de TV digital, bajo la plataforma LINUX.

1.6. METODOLOGÍA DE DESARROLLO DEL PROYECTO

Para llegar a la correcta finalización del proyecto la base son dos metodologías, Metodología de la Investigación Aplicada, que nos ayudara en obtener información relevante y fidedigna, para de esta manera nosotros poder aplicar este conocimiento. Metodología OpenUP / Basic, que nos ayudara en el desarrollo de nuestra aplicación siguiendo estos lineamentos. Las mismas se detallan a continuación de una manera más amplia:

1.6.1. Metodología de la investigación aplicada

La Investigación es un proceso que procura obtener información relevante y fidedigna (digna de fe y crédito), para entender, verificar, corregir o aplicar el conocimiento.

Para obtener algún resultado de manera clara y precisa es necesario aplicar algún tipo de investigación, la investigación está muy ligada a los seres humanos, esta posee una serie de pasos para lograr el objetivo planteado o para llegar a la información solicitada. La investigación tiene como base el método científico y este es el método de estudio sistemático de la naturaleza que incluye las técnicas de observación, reglas para el razonamiento y la predicción, ideas sobre la experimentación planificada y los modos de comunicar los resultados experimentales y teóricos.

El enfoque se orienta a la búsqueda de los conocimientos que se adquieren con esta investigación. La investigación aplicada se encuentra estrechamente vinculada con la investigación básica, ya que esta requiere de un marco teórico. Lo que buscamos con esta metodología es primordialmente son las consecuencias prácticas.

1.6.2. Metodología OPENUP / BASIC

OpenUP / Basic está diseñado para equipos pequeños, trabajando juntos en la misma localidad. El equipo tiene que participar a plenitud de la interacción diaria cara a cara.

OpenUP / Basic se enfoca en reducir significativamente el riesgo de manera temprana en el ciclo de vida. Esto requiere unas reuniones regulares de revisión de los riesgos y una implementación rigurosa de las estrategias de mitigación.

Todo el trabajo será listado, seguido y asignado a través de la "lista de ítems de trabajo". Los miembros del equipo necesitan este único repositorio para todas las tareas que necesitan ser registradas y seguidas. Esto incluye todos los requerimientos de cambio, errores y requerimientos de los stakeholder.

Los casos de uso son utilizados para obtener y describir los requerimientos. Los miembros del equipo deben desarrollar habilidades para escribir buenos casos de uso. Los Stakeholders son responsables de revisar y certificar que los requerimientos son correctos. Los casos de uso son desarrollados de manera colaborativa.

OpenUP / Basic es un proceso de desarrollo de software que es mínimo, completo y extensible. Está gobernado por cuatro principios básicos:

- Prioridades compitiendo por un balance para maximizar el valor para los stakeholders.
- Colaboración para alinear los intereses y un entendimiento compartido.
- Evolucionar para obtener continuamente retroalimentación y mejora.
- Enfoque en articular la arquitectura.

1.7. FACTIBILIDAD

1.7.1. Factibilidad Operativa

La Escuela Politécnica del Ejército será el auspiciante del proyecto de investigación de tesis, ayudara y facilitara con la información necesaria.

Los investigadores tienen apoyo y auspicio de la Escuela Politécnica del Ejército para realizar el desarrollo del proyecto de investigación es cual deberá concluir en un plazo máximo de ocho meses a partir de la fecha de aprobación del plan de tesis.

La escuela Politécnica del Ejército colaborará para el proyecto de investigación mencionado con lo siguiente:

- Acceso a Internet desde nuestras Laptops para la comunicación con todos los involucrados en el proceso de investigación del proyecto y los Ingenieros participantes.

1.7.2. Factibilidad Técnica y Económico

Las herramientas que se van utilizar son: lenguaje de programación JAVA (API GINGA-J), emulador OpenGinga, Linux como Sistema Operativo, Eclipse como IDE de desarrollo entre otros.

La investigación del proyecto es económicamente factible, debido a que las herramientas a utilizar son Libres

Tabla 1.1.: Factibilidad Económica

HARDWARE	**CANTIDAD**	**SUBTOTAL**
Computadoras Portátiles	2	2400
Impresoras	1	50
SOFTWARE	**LICENCIA**	
Emulador OpenGinga	1	0
Eclipse IDE	1	0
COMUNICACIONES	**CANTIDAD MESES**	
Internet(Otorgado por la ESPE)	5	0
EXTRAS	**CANTIDAD**	
Papelería	1	255
Copias	35	35
TOTAL		**3340**

CAPÍTULO II

MARCO TEÓRICO

2.1. GINGA

Ginga es el nombre del Middleware[1]Abierto del Sistema Brasileño de TV Digital (SBTVD).

Las aplicaciones que se ejecutan sobre Ginga están clasificadas en dos categorías dependiendo de la forma en las que fueron desarrolladas.

- Aplicaciones procedimentales(Desarrolladas con Java) y
- Aplicaciones declarativas (Utilizan el Lenguaje NLC[2]).

Ginga posee dos entornos para el desarrollo de aplicaciones interactivas Ginga-J (procedimentales) y Ginga-NCL (declarativas).

Ginga se divide en dos subsistemas principales interconectados, Ginga-J llamado máquina de ejecución y Ginga-NCL llamada máquina de presentación, como se muestra en la figura 2.1.

El motor de ejecución es posible gracias a la Máquina Virtual Java (JVM, Java Virtual Machine).

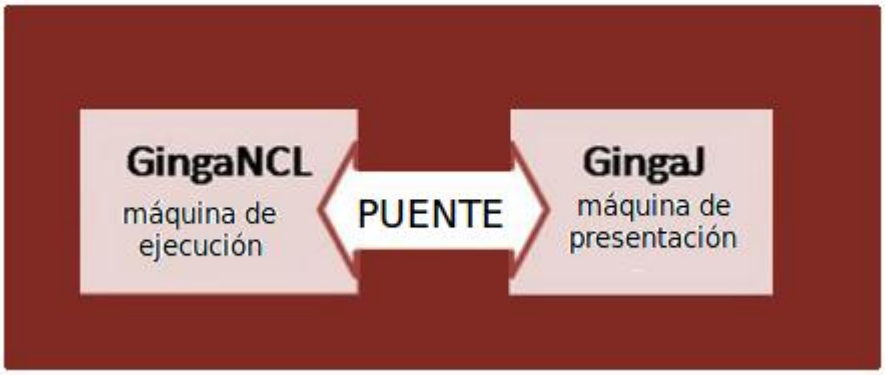


Figura 2.1.: Subsistemas de GINGA

[1] Middleware.- software conecta componentes o aplicaciones para que puedan intercambiar datos entre éstas.

[2] NCL (NestedContextLanguage) es un lenguaje de aplicación XML que permite desarrollar aplicaciones multimedia interactivas del estándar Brasileño.

La llegada de la Tv Digital ha traído consigo funcionalidades computacionales, El nuevo entorno de la Tv digital se vuelve interactivo, ya que las aplicaciones en ella pueden ser de transmisión y ejecución.

Ginga da la posibilidad de ejecutar la misma aplicación en dispositivos diferentes, con capacidad de procesamiento e independientemente de la empresa de los fabrique, esto se resuelve adoptando un Middleware común que puede definirse como el software de la capa de abstracción del hardware con características específicas.

Este es el mecanismo de la TV digital .Un entorno compuesto por un mundo de dispositivos digitales que son desarrollados por diferentes fabricantes, Ginga permite a los proveedores de contenidos desarrollar aplicaciones que se pueda ejecutar en receptores de Tv digital.

2.1.1. Compatibilidad Entre Middleware

Para permitir la ejecución de aplicaciones MHP[3](Multimedia Home Platform) en diferentes plataformas de TV digital, el grupo DVB propone un sistema unificado de condiciones para Middleware llamado GEM[4](GloballyExecutable MHP), este sistema fue adoptado por el estándar Japonés(ISDB ARIB[5]) y Americano(ATSC ACAP[6]).

Ginga el Middleware del estándar Brasileño ISDTV-T el cual tiene compatibilidad con GEM en la parte de Ginga-J.

[3] Es un Middleware que define una plataforma común para las aplicaciones interactivas de la televisión digital, independiente tanto del proveedor de servicios interactivos como del receptor de TV utilizado.
[4] Framework a partir del cual la implementación de un Middleware puede ser instanciada.
[5]El ARIB (Asociación de Industrias y Negocios de Radiodifusión) es la entidad encargada de crear y mantener ISDB(Transmisión Digital de Servicios Integrados)
[6]El ATSC (Comité de Sistemas Avanzado de Televisión) es el encargado del desarrollo de los estándares de la televisión digital en los Estados Unidos, la cual desarrollo ACAP(AdvancedCommonAplicationPlatform) como base común para todos los sistemas de TV interactivo en USA.

2.1.2. Arquitectura GINGA

El Middleware Ginga es la capa de software intermediario para el desarrollo de aplicaciones interactivas para Televisión Digital Terrestre. Los dos ambientes de ejecución (declarativo y procedimental) son exigidos en los receptores fijos y portátiles, mientras que solo el ambiente declarativo es exigido en los receptores portátiles.

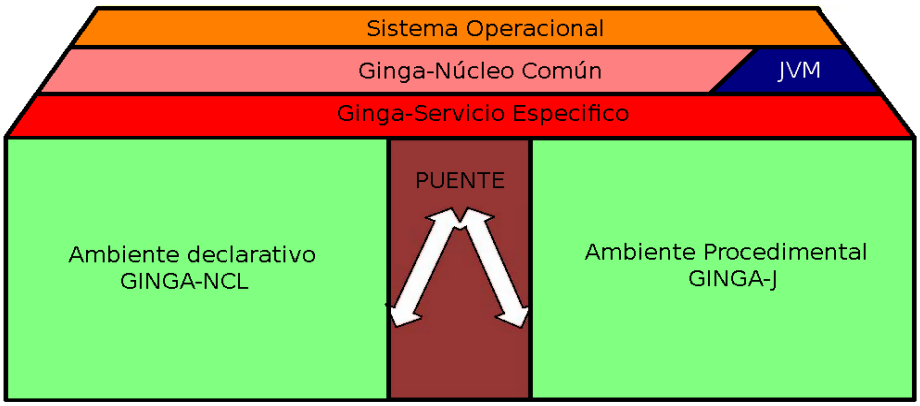


Figura 2.2.: Arquitectura GINGA

La arquitectura de implementación de referencia del Middleware Ginga está dividida en tres módulos Ginga-NCL, Ginga-J y Ginga-CC (CommonCore, Núcleo Común). En la figura 2.2se muestra la arquitectura de software para el Middleware Ginga con sus respectivos módulos.

2.1.2.1. Ginga-NCL (NestedContextLanguage,Abiente declarativo)

Ginga-NCL fue desarrollado por la Pontificia Universidad Católica de Rio de Janeiro PUC-Rio, provee una infraestructura de presentación para aplicaciones interactivas de tipo declarativas escritas en el lenguaje NCL (NestedContextLanguaje). NCL es una aplicación de XML[7] (eXtensibleMarkupLanguage) con facilidades para los aspectos de interactividad.

[7]Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

2.1.2.2. Ginga-J (Ambiente procedimental)

Ginga-J fue desarrollado por la Universidad Federal de Paraíba UFPB, para proveer una infraestructura de ejecución de aplicaciones basadas en lenguaje Java, llamadas Xlet[8] para el ambiente de Tv digital.

Un componente clave del ambiente de aplicaciones procedimentales es el mecanismo de ejecución de contenido procedimental, que tiene como base la máquina virtual de Java. Ginga-J está basado en tres grupos de API's llamados Verde, Amarillo y Azul.

2.1.2.3. Arquitectura Ginga-J.

2.1.2.3.1. Contexto

En la figura2.3 se muestra el ambiente en el que Ginga-J se ejecuta, el software desarrollado en Ginga-J reside en la capa Host (Hardware) el cual puede ser un celular, una TV digital o un Set-top-Box[9] u otro dispositivo que sea compatible con el Middleware.

Todo software desarrollado en Ginga-J puede trabajar con vídeo, audio, datos y elementos de otros medios de comunicación. Dichos flujos son distribuidos por aire, después recibido y procesado por software desarrollado en Ginga-J, Estas aplicaciones son las encargadas de presentar el contenido interactivo a los espectadores.

[8]Nombre que reciben las aplicaciones tipo DVB-Java. Son desarrolladas en lenguaje de programación Java para la interfaz API MHP

[9] Dispositivo que se conecta a un televisor y una señal externa, y que convierte la señal en contenido que es mostrado en la pantalla

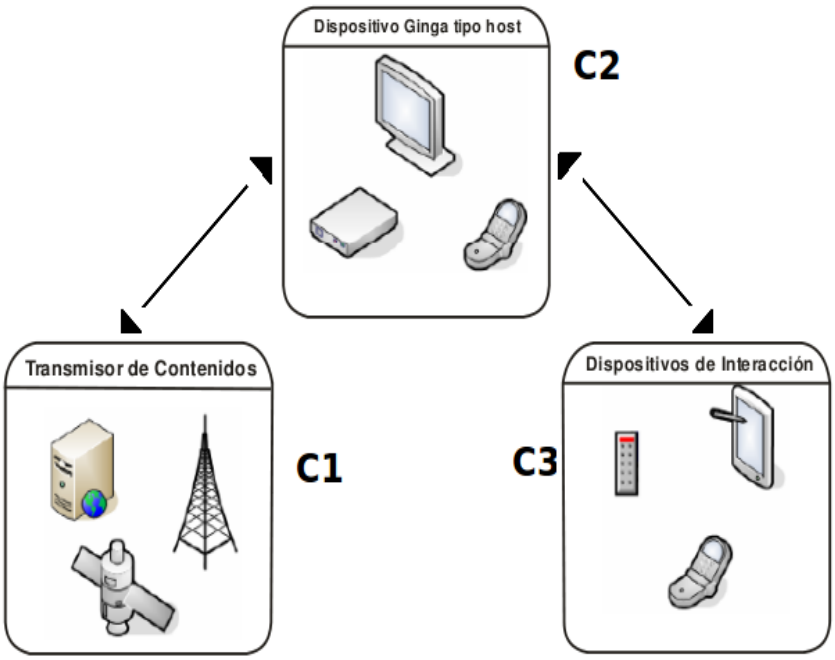


Figura 2.3.: Ambiente de ejecución GINGA-J

El espectador podrá interactuar con la aplicación realizadas en Ginga-J a través de la interacción dispositivos de salida y de entrada(Capa 3,Dispositivos de interacción).El dispositivo de acogida Ginga recibirá aportaciones de los espectadores a través de dispositivos de interacción, tales como los controles remotos o teclados.

En respuesta a la entrada de información del espectador, el dispositivo de acogida Ginga-J presentará la producción visual con salida de audio, en su propia pantalla y altavoces.

Un solo dispositivo puede tener capacidades de entrada y salida al mismo tiempo. Un ejemplo de dispositivo de interacción puede ser un Celular Touch (Compatible) que conectado a la plataforma Ginga-J a través de una red inalámbrica. Usando tal dispositivo de interacción, un espectador puede enviar comandos a la plataforma a través de un Celular Touch por teclado y la plataforma de aplicaciones puede enviar contenido visual que se presentará en la pantalla del celular.

2.1.2.3.2. Arquitectura General

En la figura 2.4 se distingue el modelo de Ginga-J entre hardware y software.

Las aplicaciones nativas pueden correr sin necesidad de las funcionalidades del sistema operativo Ginga pero también pueden utilizar los APIS estándares de Ginga-J. Las aplicaciones Xlets deben utilizar los Apis de Ginga-J.

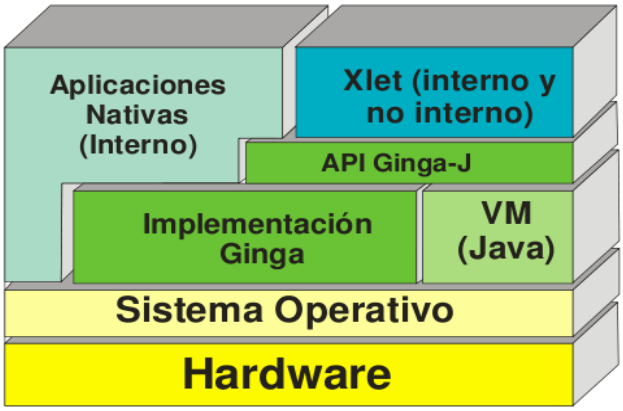


Figura 2.4.: Arquitectura General Ginga-J

Las aplicaciones nativas pueden o no tener prioridad sobre las aplicaciones Ginga-J.

Ejemplo.- Los subtítulos y mensajes emergentes tendrán más prioridad de aplicaciones Ginga-J

2.1.2.3.3. Especificaciones Ginga-J

Ginga-J es parte del Middleware Ginga basado en JVM(JAVA VIRTUAL MACHINE) y algunos APIS adicionales los cuales añaden muchas más innovaciones sin perder la compatibilidad con la TV digital.

El estándar Brasileño de Tv Digital tiene como principal objetivo transmitir la señal de TV digital de manera simultánea a diferentes dispositivos tales como receptores fijos de televisión de alta definición y definición estándar, dispositivos móviles y portátiles, como teléfonos celulares.

La especificación Ginga-J también incluye soporte para para dispositivos que utilizan redes como Bluetooth, Wi-Fi, infrarrojo, Ethernet. Cabe destacar que la especificación Ginga-J proporciona soporte para la interacción entre múltiples usuarios debido a que se puede tener accesos simultáneamente con los diferentes dispositivos.

2.1.2.3.4. Apis Ginga-J

Con el fin de mantener la compatibilidad con el API de GEM, Ginga-J se basa en tres grandes grupos de APIS:

- API Verde (Apis compatibles con GEM), aquí se encuentran incluyendo las Apis provenientes de los paquetes de SunJavaTV, DAVIC [DAVIC, 1999] e HAVI [HAVi, 2001].
- API Amarillo(compuesto por el JMF 2.1[10] API, lo cual es necesario para el desarrollo de aplicaciones, con captura de sonido) y
- API Azul (permite al receptor de TV digital comunicarse con cualquier dispositivo con una interfaz compatible (Con conexión de cable, como Ethernet o PLC, de red o inalámbricas, como Infrarrojos o Bluetooth)), Aquí se encuentra el API que permite el desarrollo de aplicaciones Ginga-J que tengan Ginga NLC (Api Puente); como se muestra en la figura 2.5

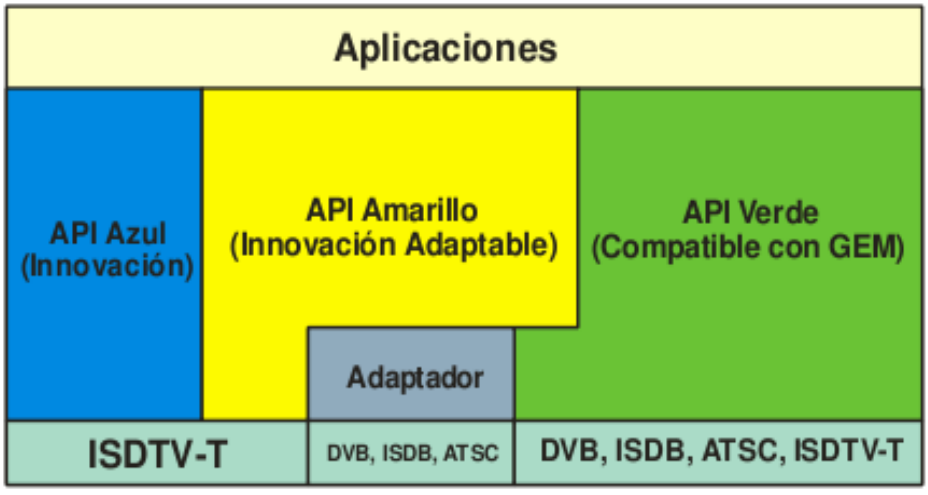


Figura 2.5.: APIs GINGA JAVA

[10] Java Media Framework en una aplicación de java, de la librería javax.media

2.1.2.3.5. API JavaTV

El API JavaTV es una extensión de la plataforma Java que sirve para apoyarla producción de contenidos interactivos de forma procedural para la televisióndigital. El objetivo principal de la API JavaTV es proporcionar un conjunto de métodos, clases e interfaces para facilitar la creación de aplicaciones diseñadas para ejecutarse en distintas plataformas para la recepción de televisión digital independiente de las tecnologías utilizadas en la red de transmisión.

JavaTV soporta un alto nivel de interactividad, calidad gráfica y deprocesamiento para ser ejecutado dentro de un set top box, siempre y cuando este equipada con la máquina virtual de Java, necesaria para interpretar los bytecodesgenerados.
Utilizando su arquitectura la API JavaTV es capaz de realizar funcionalidades tales como:

- **Streaming de audio y vídeo**: Además de la streaming de vídeo y audio procedente de la estación, es posible generar otras aplicaciones con otros flujos.

- **Acceso a datos en el canal de transmisión**: JavaTV pueden recibirdatospara las aplicaciones.

- **Aplicaciones con interactividad**: Las aplicaciones que pueden utilizaresta API pueden procesar datos y devolverlos a través de un canal deretorno.

- **Gestión del Ciclo de vida de las aplicaciones:** permitiendo que lasaplicaciones coexistan con el contenido de TV convencional y quepermite el intercambio de canal sin que la aplicación deje de existir.

La API JavaTV tiene varias librerías, que son responsables de proveer unabásica del sistema. Las librerías están dispuestas de la siguiente forma:

- **javax.tv.carousel:** proporciona acceso a archivos broadcast y directoriode datos a través de APIs que trabajan con el paquete java.io.

- **javax.tv.graphics:** permite que los Xlets, puedan obtener su repositorioprincipal.

- **javax.tv.locator:** proporciona una forma para referenciar datos enprogramas accesibles por la API JavaTV.

- **javax.tv.media:** define una extensión para JMF (Java MediaFramework) con la finalidad de gestionar los medios de comunicaciónen tiempo real;javax.tv.media.protocol: proporciona acceso a un flujo de datosbroadcast genérico.

- **javax.tv.net:**permite acceso a datagramas IP (Internet Protocol) transmitidos en un streambroadcast.
- **javax.tv.service:** proporciona mecanismos para acceder a la base dedatos.

- **javax.tv.util:** soporta la creación y gestión de eventos del temporizador

- **javax.tv.xlet:** proporciona interfaces para el desarrollo de aplicacionesy la comunicación entre las aplicaciones y el administrador.

2.1.2.3.6. API DAVIC

El sistema DAVIC (Digital Audio Visual Council) es un conjunto de especificaciones que presenta algunos requisitos de sistemas audiovisuales para proporcionar interoperabilidad de extremo a extremo. Así, estos patrones se pueden utilizar en sistemas de televisión digital para proporcionar contenido al usuario final y también para permitir la interactividad con el mismo usuario.

El sistema DAVIC tiene su propio API, y por lo tanto puede ser considerado middleware como de alto nivel (aunque es común que empiecen a operar en conjunto con un middleware de bajo nivel). A continuación se enumeran los paquetes que forman parte de la API DAVIC incluido en el Ginga-J.

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

2.1.2.3.7. API HAVi

HAVi (Home Audio Video Interoperability) es un API que permite al programador crear elementos, como la interfaz del usuario. Provee una extensión del paquete java.awt, permitiendo, asimismo, soporte de control remoto, transparencia, entre otros.

El objetivo principal de una interfaz de usuario es proporcionar un entorno operativo fácil de usar. La arquitectura HAVi permite que los usuarios controlen dispositivos de forma familiar a través de un control remoto o delante de una pantalla.

A continuación se enumeran los paquetes que forman parte de la API HAVi incluido en la Ginga-J:

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

2.1.2.3.8. API DVB

En el desarrollo del middleware patrón MHP, el DVB incluye algunos paquetes para extender la funcionalidad ofrecida por JavaTV, HAVi y DAVIC. Estas 27 características incluyen API de información de servicio, de intercomunicación entre Xlets, persistencia, etc. Algunas de las características también se incluyeron en el GEM. A continuación se enumeran los paquetes que forman parte de la API DVB incluida en la Ginga-J [14].

- org.dvb.application
- 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

2.1.2.3.9. Implementación Ginga-J

En la actualidad Ginga-J tiene 800 mil líneas de código, El software Ginga-J fue diseñado con el fin de ser un Middleware de desarrollo procedimental para aplicaciones de TV digital, el cual posee tres características muy importantes:

Ginga-J es:

- Muli – Red.
- Multi – Sistema.
- Compatibles.

2.1.2.3.9.1. Multi – Red

Actualmente el Middleware Ginga-J es compatible con todos los medios que trabajan con TV digital (cable,terrestre,satélite entre otros).

2.1.2.3.9.2. Multi – Sistema

Ginga-J incluye la tabla de procesamiento del Servicio de Información de DVB, ATSC y ARIB, utilizando esta API de Servicio de Información como la salida de datos.

2.1.2.3.9.3. Compatibilidad

Debido a que Ginga-J implementa el API Verde, Ginga-J es capaz de de ejecutar todas las aplicaciones desarrolladas en otros Middleware de GEM

2.1.2.3.10. Componentes de Software de Ginga-J

Ginga-J fue desarrollado basado en el enfoque de componentes de software, este componente facilita la evolución del Middleware, permitiendo la incorporación de nuevas funcionalidades atreves de nuevos componentes y además la reutilización de algunos componentes para el desarrollo evolutivo. Por ejemplo, para celulares, y receptores de TV de alta calidad (con muchos recursos y mucho más costosos), receptores de baja calidad (con menores recurso y, menos costosos).

Estos componentes pueden ser reunidos sobre diferentes perfiles, en función de algunas características de la plataforma. Ginga-J según su funcionalidad puede ser agrupado en siete grandes componentes.

2.1.2.3.10.1. Componente de Acceso de Flujo de Bajo Nivel

Contenedor de elementos responsables para el acceso al flujo de transporte y demultiplexión

2.1.2.3.10.2. Componentes elementales del procesamiento de flujo

Contenedor de elementos responsables de la decodificación y flujo entre los componentes

2.1.2.3.10.3. Componentes de interfaz de usuario

Contenedor deelementos que permiten la interactividad del usuario a través de elementos audiovisuales

2.1.2.3.10.4. Componentes de Comunicación

Contenedor de componentes responsables de la comunicación entre aplicaciones que se ejecutan en los Middleware.

2.1.2.3.10.5. Componentes de Gestión

Componente encargado de la gestión del Middleware, contexto, actualizaciones del Middleware

2.1.2.3.10.6. Componentes de persistencia

Componentes responsables de almacenamiento de datos

2.1.2.3.10.7. Componentes de Acceso

Controla y restringe el acceso a contenidos transmitidos por su proveedor de los mismos

2.1.2.3.11. GingaCommonCore

GingaCommonCore es el núcleo de presentación y ejecución. (GINGA-J, GINGA-NLC).Este subsistema es el puente con el hardware. Aquí es donde se accede al sintonizador de canales, sistema de archivos, terminal gráfico, entre otros.

GingaCommonCore está compuesto por decodificadores de contenido común, los cuales sirven tanto para las aplicaciones de tipo declarativas como para las procedimentales.

Los componentes de GingaCommonCore son:

2.1.2.3.11.1. El Sintonizador

Responsable de la sincronización de los canales, seleccionando un canal físico y los flujos de transporte que están siendo enviados por este canal.

2.1.2.3.11.2. Filtros de Selección

Una vez sintonizado el canal, el Middleware debe ser capaz de acceder a partes específicas del flujo de transporte. Para esto, existe un Filtro de Selección, el mismo que es capaz de buscar en el flujo, la parte exacta que las APIs necesitan para su ejecución.

Deja pasar solo la información requerida por la API. Funcionando exactamente como un filtro.

2.1.2.3.11.3. Procesador de Datos

Es el elemento responsable de acceder, procesar y transferir los datos recibidos por la capa física. También es responsable de notificar a los otros componentes, sobre cualquier evento que se ha recibido.

2.1.2.3.11.4. Persistencia

Ginga es capaz de guardar archivos, incluso después que ha finalizado el proceso que los creó, para que este pueda ser abierto en otra ocasión.

2.1.2.3.11.5. Administrador de Aplicaciones

Es el módulo responsable de cargar, configurar, inicializar y ejecutar cualquier aplicación en cualquier entorno ya sea declarativo o de procedimiento. También es responsable de controlar el ciclo de vida de las aplicaciones, eliminarlas cuando sea necesario, además de controlar los recursos utilizados por esas APIs.

2.1.2.3.11.6. Adaptador Principal de Audio / Video

Con el Adaptador Principal de A/V, las aplicaciones consiguen ver el flujo de audio y vídeo. Esto es necesario cuando una aplicación necesita controlar sus acciones, de acuerdo con lo que se está transmitiendo.

2.1.2.3.11.7. Administrador de Gráficos

Las normas del Middleware definen como se presentan al usuario las imágenes, vídeos, datos, etc., administrando las presentaciones de la misma manera que está definida en el estándar ARIB.

2.1.2.3.11.8. Administrador de Actualizaciones

Es el componente que gestiona las actualizaciones del sistema controlando, descargando las actualizaciones del Middleware siempre que sea necesario, para corregir los errores encontrados en versiones anteriores. Esto de ser hecho en tiempo de ejecución, sin perturbar el uso normal de la TV por parte del usuario.

2.1.2.3.11.9. Reproductor de Archivos Multimedia

Son las herramientas necesarias para presentar los archivos multimedia recibidos, como por ejemplo archivos de tipo MPEG, JPEG, TXT, MP3, GIF, HTML, etc.

2.1.2.3.11.10. Interface de Usuario

Este módulo es responsable de captar e interpretar los eventos generados por los usuarios, tales como, comandos del control remoto y notificar a los otros módulos interesados.

2.1.2.3.11.11. Administrador de Contextos

Es el responsable de captar las preferencias del usuario, notificando a los otros componentes interesados esas preferencias. Esta información puede ser por ejemplo el horario en que el usuario mira la TV, o el bloquear y desbloquear canales, entre otros.

2.1.2.3.11.12. Canal de Retorno

Proporciona la interfaz de las capas superiores con el canal de interacción (o canal de retorno). Además, debe gestionar el canal de retorno de modo que los datos sean transmitidos cuando el canal esté disponible o forzar la transmisión en caso de que el usuario o una aplicación tengan definido un horario exacto.

2.1.2.3.11.13. Acceso Condicional

Este componente está encargado de restringir contenidos inapropiados recibidos por los canales de programación, proporcionando así seguridad para el Middleware.

2.1.2.3.11.14. Ambientes de Emulación para la Programaciónón de GINGA

Las aplicaciones pueden ser desarrolladas tanto por emisoras de televisión como por los usuarios. En el caso que sea desarrollada por una emisora, la aplicación será enviada al Set Top Box, a través de un canal de transmisión. En el caso de ser desarrollada por el usuario esta tendrá que ser enviada al Set Top Box a través de una entrada externa (USB portable, puerta de red, tarjeta de memoria, etc.)

En el ambiente de ejecución de Xlets, o Ginga-J, se define la cantidad de Xletsnecesarios. Esta cantidad depende del número de características distintivas que se llevarán a cabo para resolver el problema.
Se recomienda crear un Xlet para cada función específica, por ejemplo, se ilustra una aplicación interactiva de un programa deportivo, para el cual se han desarrollado los siguientes Xlets:

- xlet1: Visión de la camara
- xlet2: Propaganda
- xlet3: 100 m planos
- xlet4: Salto con garrocha
- xlet5: Lanzamiento de Martillo
- xlet6: Salto alto
- xlet7: Lanzamiento de dardo
- xlet8: Selección de camaras

Como suele ser difícil para un desarrollador de tener una red de televisióndigital experimental, disponible en la mayoría de los casos, el medio ambiente es simulado con el uso de las estaciones de prueba o con emuladores de software. Para el desarrollo de aplicaciones existen varios emuladores, que simulan el papel del middleware, entre ellos los más utilizado para el ambiente Ginga-J es XleTView / OpenGinga y para el ambiente Ginga-NCL el Virtual Set Top Box.

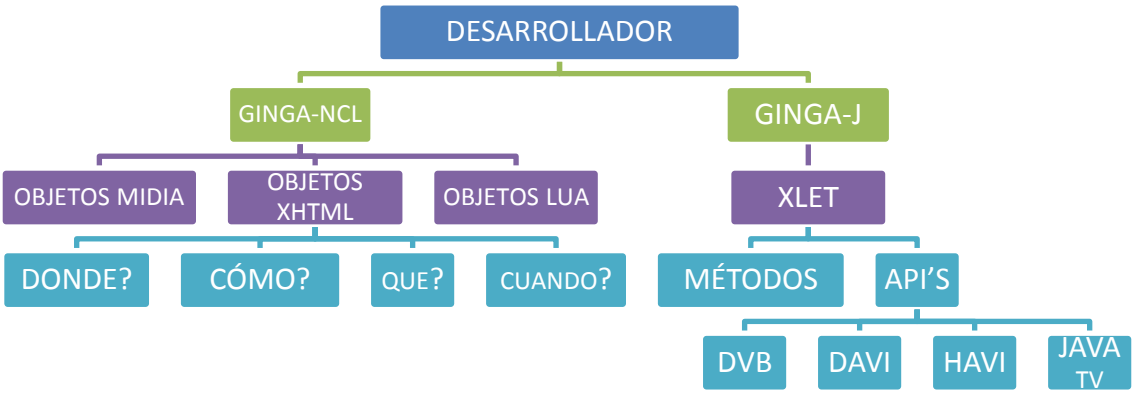


Figura 2.6.: Ambientes desarrollo de aplicaciones

2.1.2.3.11.14.1. Emulador GINGA-J: XLetView

XletView, es un emulador usado para ejecutar Xlets en una PC, es de código abierto, y está bajo la licencia de software libre GPL (General PublicLicense), además de tener una implementación de referencia a la API JavaTV, trae consigo implementaciones de otras APIs especificadas en el estándar MHP (Multimedia Home Platform), como HAVI (Home Audio Video Interoperability), DAVIC (Digital Audio Video Council) e implementaciones especificadas por la propia DVB (Digital Video Broadcasting), además de las bibliotecas de PersonalJava.
XletView está desarrollado en Java y para su ejecución independientemente del sistema operativo, es necesario utilizar el Java 2 Standard Development Kit para compilar Xlets y ejecutar el XletView. Este emulador utiliza la JMF (Java Media Framework) 2.1.1, pero con algunas deficiencias, como la incapacidad de exhibir video MPEG (Moving Picture Grupo de expertos) relacionados o controlados por un Xlet.

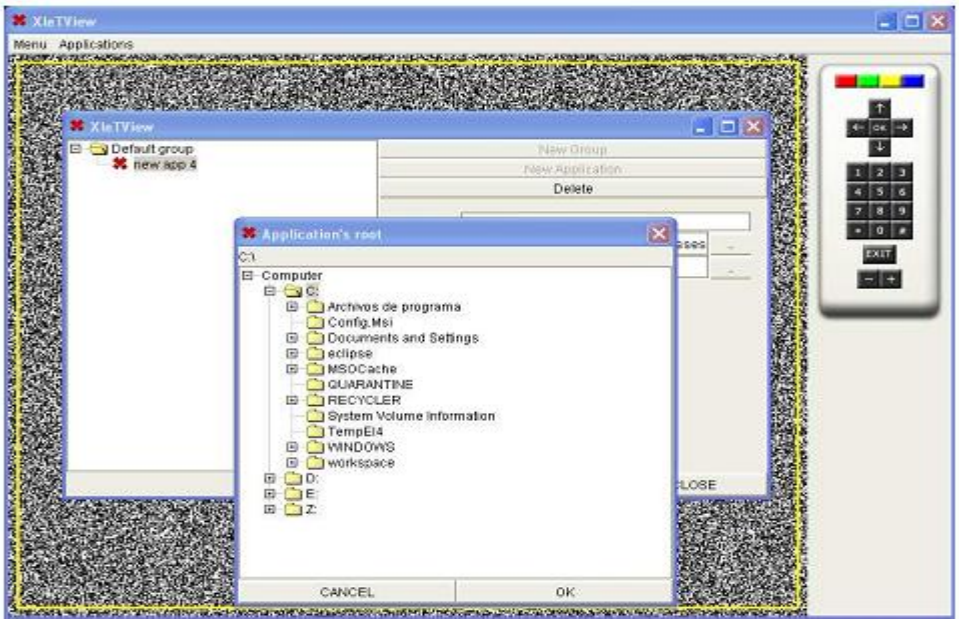


Figura 2.7.: Pantalla ejecución Xlet

2.1.2.3.11.14.2. Emulador GINGA-J: OPENGINGA

Open Ginga es una implementación que hace referencia al ambiente Ginga-J, parte del Middleware Ginga, que es una de las normas del sistema Brasileño de Tv digital.

OpenGinga es un proyecto open source (código libre) está siendo desarrollado por el laboratorio LAVID de la Universidad Federal de Paraina.

En la actualidad existe la máquina virtual con el Sistema operativo Ubuntu 10.04 en cual tiene la implementación del emulador Open Ginga el cual es el que se utilizara para el desarrollo de la aplicación.

Figura 2.8.: Pantalla emulador OpenGinga

2.1.2.3.11.15. Ciclo de vida de las aplicaciones

El Gestor de Aplicaciones debe usar la Xlet API (GINGA J) para ordenar cambios en el ciclo de vida de las aplicaciones.

La propia aplicación puede decidir cambiar su estado. Para ello, debe usar su instancia de javax.microedition.xlet.XletContext para requerir ese cambio al Gestor de Aplicaciones.

Un Xlet tiene cinco estados principales: Loaded, Initialised, Started, Paused y Destroyed.

2.1.2.3.11.15.1. Loaded

El application manager (aplicación de administración que controla el MHP) carga el archivo .class principal del Xlet y crea una instancia de la propia Xlet para llamar a un constructor por defecto.

2.1.2.3.11.15.2. Initialised

Para ejecutar la aplicación interactiva, el application manager llama al métodoinitXlet(), pasándole un objeto XletContextespecífico para el Xlet. El Xlet puede usar el objeto XletContext para precarga aquellos datos que podrían suponer un tiempo excesivo para ser cargados posteriormente, como por ejemplo imágenes.

2.1.2.3.11.15.3. Started

Una vez el método initXlet () retorna, el application manager llama al métodostartXlet () habilitando la interacción de la aplicación con el usuario.

2.1.2.3.11.15.4. Paused

El aplicación manager puede parar la ejecución del Xlet con el métodopauseXlet (), para liberar recursos o porque la funcionalidad de la aplicación lo ha decidido. Posteriormente se puede reanudar mediante startXlet ().

2.1.2.3.11.15.5. Destroyed

Al final del ciclo de vida, el application manager llama al método destroyXlet () liberando de esta manera todos los recursos.

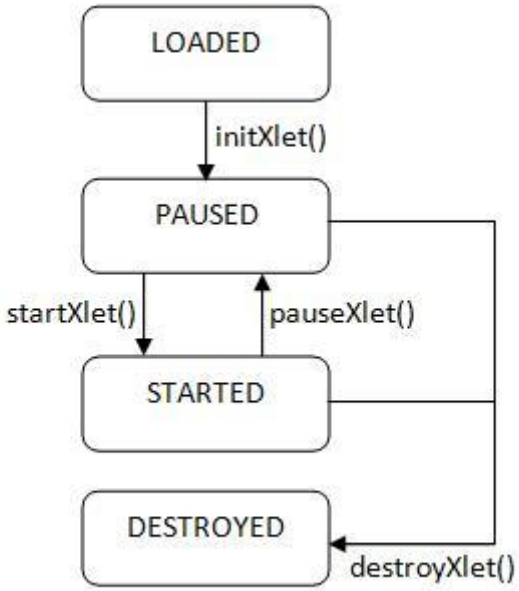


Figura 2.9.: Ciclo de vida Xlet

2.1.2.3.11.16. Interfaz gráfica de usuario

La interfaz gráfica del usuario varios componentes los cuales se detalla a continuación.

- **com.sun.dtv.ui:**define los componentes gráficos especialmente vinculados a televisión.
- **com.sun.dtv.lwuit:** contiene los componentes gráficos que dan soporte a la creación de interfacesgraficas de usuario.
- **com.sun.dtv.lwuit.animations:**habilita tanto componentes gráficos como transiciones animadas.
- **com.sun.dtv.lwuit.geom:**define los elementos geométricos básicos para dibujo.

- **com.sun.dtv.lwuit.layouts:** define tipos útiles de layouts gráficos.

- **com.sun.dtv.lwuit.list:** define estructuras de lista personalizables utilizadas en componentesde otros paquetes como el com.sun.dtv.lwuit.

- **com.sun.dtv.lwuit.painter:**permite dibujar arbitrariamente elementos gráficos a partir de imágenesplanas, escaladas y/o tiled.

- **com.sun.dtv.lwuit.plaf:** permite personalizar la apariencia de los componentes gráficos.

- **com.sun.dtv.lwuit.util:**paquete de utilidades.

2.1.2.3.11.17. Tratamiento de eventos del usuario

El middleware posee dos librerías la cual se encarga de todos los eventos con los que puede interactuar el usuario.

- **com.sun.dtv.ui.event:**mecanismo para tratamiento de eventos específicos de televisión.

- **com.sun.dtv.lwuit.events:**mecanismo de tratamiento de eventos Relacionados a los componentes gráficos.

Tabla 2.1.: Eventos relacionados a los componentes gráficos

Funciones definidas		**Clase com.sun.dtv.lwuit.event. RemoteControlEvent**
Confirma		VK_CONFIRM
Salir		VK_ESCAPE
Volver		VK_BACK
Direccional	Arriba	VK_UP y VK_KP_UP
	Abajo	VK_DOWN y VK_KP_DOWN
	Izquierda	VK_LEFT y VK_KP_LEFT
	Derecha	VK_RIGHT y VK_KP_RIGHT
Coloridas	Arriba	VK_COLORED_KEY_0
	Abajo	VK_COLORED_KEY_1
	Izquierda	VK_COLORED_KEY_2
	Derecha	VK_COLORED_KEY_3

2.1.2.3.11.18. Lista de paquetes mínimos del Ginga-J

2.1.2.3.11.18.1. Paquetes de la plataforma Java

Los siguientes son parte de la plataforma JAVA:

- java.awt
- java.awt.color
- java.awt.event
- java.awt.font
- java.awt.im
- java.awt.image
- java.beans
- java.io
- java.lang
- java.lang.ref
- java.lang.reflect
- java.math
- java.net
- java.rmi
- java.rmi.registry
- java.security
- java.security.acl
- java.security.cert
- java.security.interfaces
- java.security.spec
- java.text
- java.util
- java.util.jar
- java.util.zip
- javax.microedition.io
- javax.microedition.pki
- javax.microedition.xlet
- javax.microedition.xlet.ixc
- javax.security.auth.x500

2.1.2.3.11.18.2. Paquetes de la especiación JavaTV 1.1 y JMF 1.0

Los siguientes son parte de la plataforma JAVATV:

- javax.media
- javax.media.protocol
- javax.tv.graphics
- javax.tv.locator
- javax.tv.media
- javax.tv.net
- javax.tv.service
- javax.tv.service.guide
- javax.tv.service.navigation
- javax.tv.service.selection
- javax.tv.service.transport
- javax.tv.util
- javax.tv.xlet

2.1.2.3.11.18.3. Paquetes de la especiación JAVADTV

Los siguientes son parte de la plataforma JAVATV:

- com.sun.dtv.application
- com.sun.dtv.broadcast
- com.sun.dtv.broadcast.event
- com.sun.dtv.filtering
- com.sun.dtv.io
- com.sun.dtv.locator
- com.sun.dtv.lwuit
- com.sun.dtv.lwuit.animations
- com.sun.dtv.lwuit.events
- com.sun.dtv.lwuit.geom
- com.sun.dtv.lwuit.layouts
- com.sun.dtv.lwuit.list
- com.sun.dtv.lwuit.painter
- com.sun.dtv.lwuit.plaf
- com.sun.dtv.lwuit.util
- com.sun.dtv.media
- com.sun.dtv.media.audio
- com.sun.dtv.media.control
- com.sun.dtv.media.dripfeed
- com.sun.dtv.media.format
- com.sun.dtv.media.language
- com.sun.dtv.media.text
- com.sun.dtv.media.timeline
- com.sun.dtv.net
- com.sun.dtv.platform
- com.sun.dtv.resources
- com.sun.dtv.security
- com.sun.dtv.service
- com.sun.dtv.smartcard
- com.sun.dtv.test
- com.sun.dtv.transport
- com.sun.dtv.tuner
- com.sun.dtv.ui
- com.sun.dtv.ui.event

2.1.2.3.11.18.4. Paquetes específicosGinga-J

Los siguientes son parte de la plataforma GINGA-J

- br.org.sbtvd.net
- br.org.sbtvd.net.si
- br.org.sbtvd.net.tuning
- br.org.sbtvd.bridge
- br.org.sbtvd.ui

2.1.2.3.11.19. Comunicación Ginga-J y Ginga NLC

La comunicación entre Ginga-J y Ginga-NCL se realiza a través de un puente, el cual permite realizar solicitudes procedimentales - declarativas y viceversa.

Esta comunicación se puede realizar de dos formas:

- Permite al entorno declarativo llamar a un XletGinga-J a través de un objeto.

- Se utiliza en los siguientes elementos: <link> (referencia a elementos <media> que representan a códigos Xlet (de tipo “application/x-ginga-NCLet”) soportados por Ginga-J.

En sentido contrario con las funciones Ginga-J se puede controlar los eventos NCL,a través de comandos de edición NCL

2.2. MetodologíaOpenUP / Basic

2.2.1. Introducción

Los proyectos de desarrollo de software tienen diferentes necesidades en sus procesos, siendo estos; el tamaño del equipo y su localización, la complejidad de la arquitectura, innovación tecnológica, conformidad a las normas, entre otros.

Sin embargo, existen diferentes metodologías que ayudan en los proyectos a los desarrolladores de software a tener un control y reducir los riesgos, es por esto que para el desarrollo de nuestra aplicación se ve la necesidad de utilizar la metodología OPENUP / BASIC para estructurar, planificar y controlar los procesos.

2.2.2. Concepto

Es un proceso de desarrollo de software de código abierto, que forma parte del Framework de Eclipse[11], que fue diseñado para equipos pequeños quienes quieren tomar una aproximación ágil.

Comprende procesos iterativos en los cuales se valora la colaboración y el aporte de los Stakeholders[12], sobre los diferentes artefactos a ser entregados en cada fase. Dando como resultado un proceso estructurado, robusto, eficiente y liviano.

Se encuentra organizado dentro de cuatro áreas principales de contenido: Comunicación y Colaboración, Intención, Solución, y Administración.

2.2.3. Principios

Se caracteriza por cuatro principios:

- **Colaborar para alinear intereses y compartir conocimiento** este principio promueve prácticas que ayuda a poseer un buen ambiente de equipo, la colaboración y desarrollo, para compartir el conocimiento del proyecto.
- **Balance de las prioridades que compiten para maximizar el valor para los stakeholders**este principio promueve prácticas que permiten a los Stakeholdersdiseñar una solución que maximicé el valor para los beneficios del proyecto.
- **Centrarse en la arquitectura de principios para minimizar los riesgos y organizar el desarrollo** este principio promueve prácticas que permiten al equipo centrarse en la arquitectura para minimizar los riesgos y organizar el desarrollo.

[11]Framework de modelado y facilidad de generación de código para construir herramientas y otras herramientas basadas en un modelo de datos estructurado.

[12]Stakeholders.- son todos aquellos individuos cuyos objetivos dependen de lo que haga la organización y de los que a su vez dependen de la organización.

- **Evolucionar continuamente para obtener retro alimentación y mejorar** este principio promueve prácticas que permiten al equipo tener una continua y temprana retro alimentación para los Stakeholders, y demostrar el valor a incrementar a ellos.
-

Cada uno de estos principios está apoyado por una declaración en el manifiesto ágil.

Tabla 2.2.: Mapeo OpenUP / Basic y Manifiesto Ágil

Principios de OPENUP / BASIC	Manifiesto Ágil
Colaborar para alinear intereses y compartir conocimiento.	Individuos e interacciones sobre procesos y herramientas.
Balance de las prioridades que compiten para maximizar el valor para los stakeholders.	Colaboración del cliente sobre la negociación del contrato.
Centrarse en la arquitectura para minimizar los riesgos y organizar el desarrollo.	Trabajando sobre el software con una documentación completa.
Evolucionar continuamente para obtener retroalimentación y mejorar.	Responder a los cambios en el seguimiento de un plan.

2.2.4. Iteraciones del ciclo de vida

Los proyectos en general se dividen en iteraciones[13], como se muestra en la figura 2.10, las cuales son definidas en intervalos pequeños de tiempo. Estas ayudan a los equipos de trabajo a enfocar los esfuerzos a través del ciclo de vida de cada iteración de tal forma que se puedan distribuir funcionalidades incrementales de una manera predecible, y de esta manera obtener al final de cada una de las iteraciones versiones totalmente probadas y funcionales.

Dentro de cada tiempo de iteración se ejecutan micro-incrementos[14], los cuales ayudan a realizar pruebas a un código en construcción y los artefactos a entregar, como se muestra en la figura 2.11.

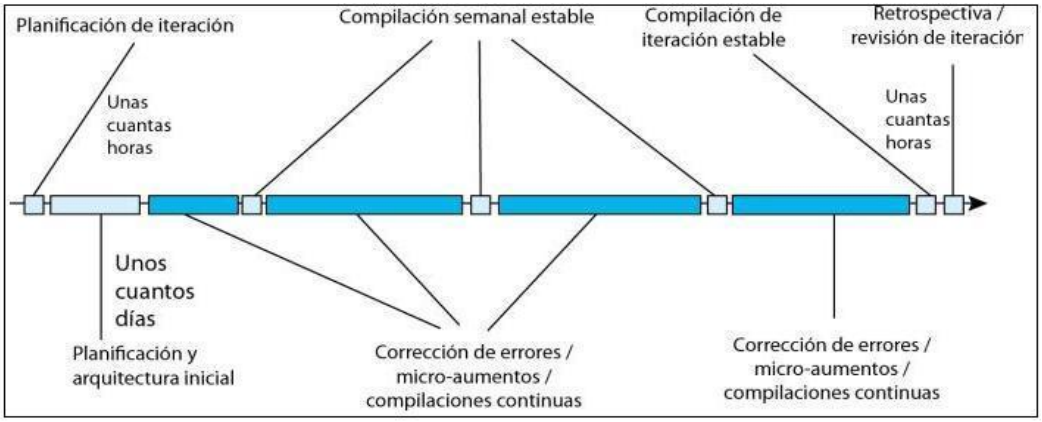


Figura 2.10.: Ciclo de vida de las iteraciones

[13]Iteraciones en proyectos se refiere a un paso previsto en el desarrollo de una parte del Sowtware.
[14]Micro-incrementos.- representa el resultado de horas, o días de trabajo de un grupo de personas, que colaboran para alcanzar los **objetivos** de las iteraciones.

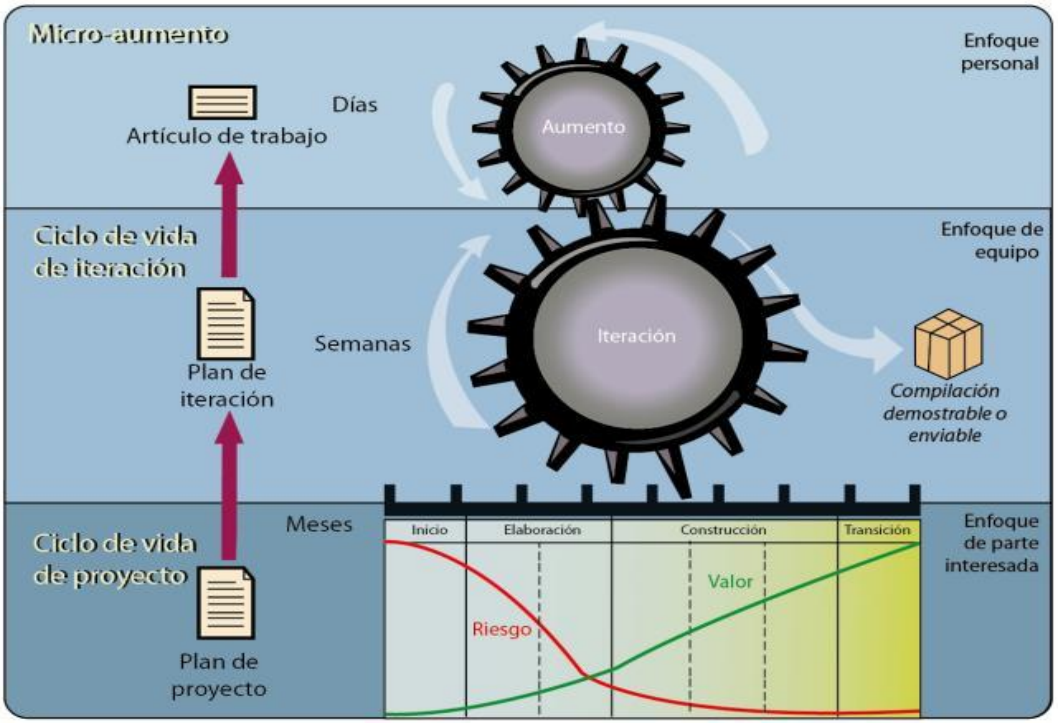


Figura 2.11.: Iteraciones de un proyecto

2.2.5. Ciclo de Vida

El modelo del ciclo de vida de OPENUP / BASIC, es un proceso iterativo cuyas iteraciones se dividen a través de cuatro fases distintas, cada una con un propósito específico, como se muestra en la figura 2.12.

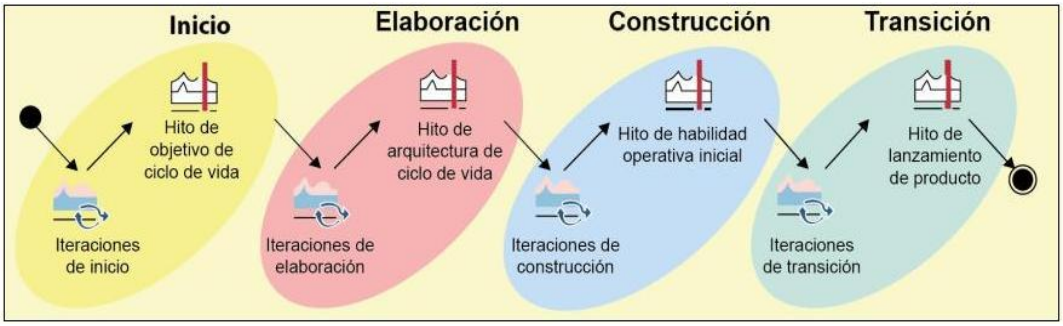


Figura 2.12.: Ciclo de Vida OpenUP / Basic

2.2.5.1. Inicialización

Se debe entender el alcance del proyecto y obtener información para confirmar que el proyecto debe realizarse. El propósito de esta fase es lograr concurrencia entre todos los Stakeholders sobre los objetivos y ciclo de vida para el proyecto.

2.2.5.2. Elaboración

Se debe ver específicamente cuando la arquitectura tiene riesgos significativos. El propósito de esta fase es para establecer la línea base de la arquitectura del sistema y proporcionar una base estable para un mayor esfuerzo en la próxima fase de desarrollo.

2.2.5.3. Construcción

Se debe enfocarse sobre el diseño, implementación, y probar las funcionalidades del desarrollo completo del sistema. El propósito en esta fase es el completar el desarrollo del sistema basado en la arquitectura definida.

2.2.5.4. Transición

El propósito en esta fase es para asegurar que el software pueda ser usado por el usuario, y evalúa la funcionalidad y performance del último entregable de la fase de construcción.

Las fases de inicialización y elaboración también ayudan a establecer y organizar los productos y procesos claves del proyecto, de modo que en la fase de construcción se puede comenzar con un equipo en crecimiento y cumplir los plazos en tiempos mínimos.

2.2.6. Roles

Los roles no representan las responsabilidades individuales en las tareas o prestaciones, sino que son "sombreros" que la gente puede ponerse al trabajar juntos. Cada papel no se limita a describir el intérprete principal de una tarea, en lugar de las funciones incluyen una perspectiva de colaboración al proporcionar intérpretes adicionales para cada tarea.

La reproducción de uno o más funciones puede ayudar a los equipos a expresar puntos de vista diferentes al crear una solución. Esta perspectiva sobre los roles permite a una nueva generación de procesos de desarrollo de software, como se muestra en la figura 2.13.

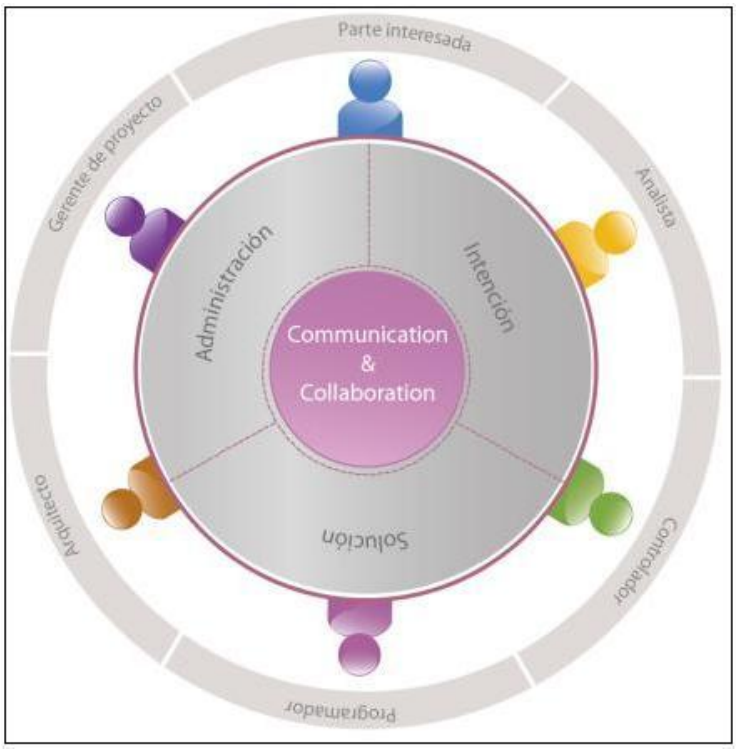


Figura 2.13 Roles OpenUP / Basic

2.2.6.1. Rol Analista

Las personas en este rol representan al cliente y los usuarios finales involucrados, aquí se obtiene los requerimientos de las partes, para llegar a comprender el problema a resolver, capturando y ajustando las prioridades para los requerimientos.

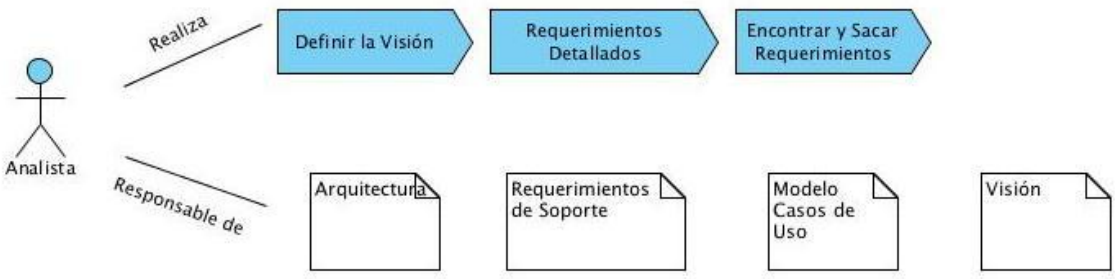


Figura 2.14.: Rol Analista

2.2.6.2. Rol Arquitecto

Este rol es el responsable de diseñar la arquitectura del software, la cual incluye tomar las principales decisiones técnicas que condicionan globalmente el diseño y la implementación del proyecto, además está estrechamente involucrado en organizar el equipo encargado de la arquitectura, trabajando estrechamente con el director del proyecto para conformar el staff y planear el proyecto.

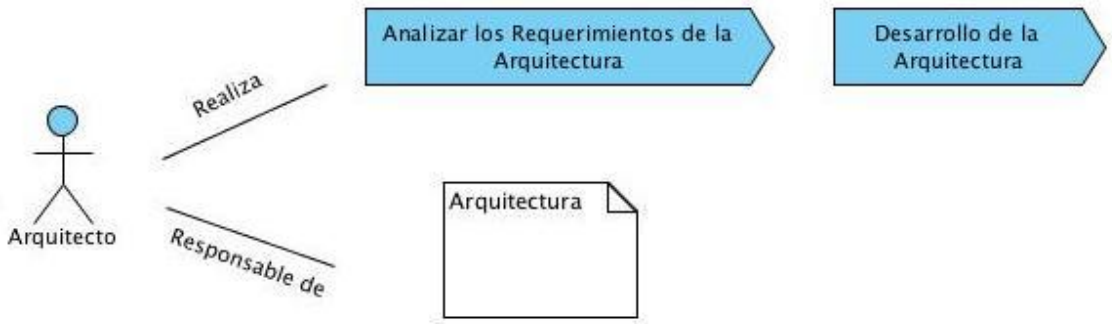


Figura 2.15.: Rol Arquitecto

2.2.6.3. Rol Desarrollador

La persona en este rol es responsable por desarrollar una parte del sistema, incluyendo diseñar esta para que se ajuste a la arquitectura, posiblemente prototipar la interfaz de usuario y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución.

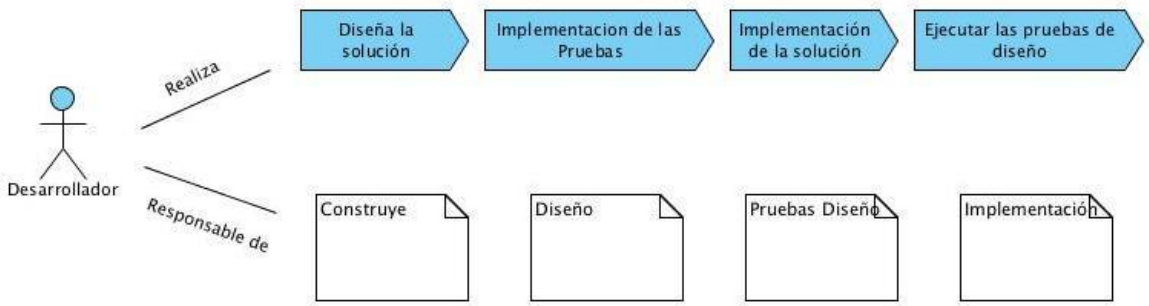


Figura 2.16.: Rol Desarrollador

2.2.6.4. Rol Director del Proyecto

Lidera la planeación del proyecto, coordina interacciones con los stakeholders y conserva el equipo del proyecto enfocado en alcanzar los objetivos del proyecto.

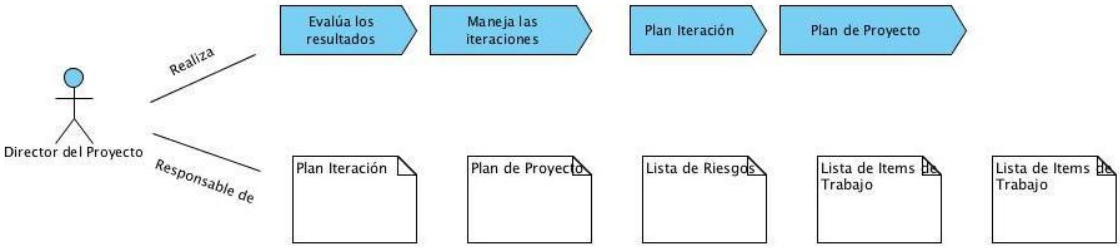


Figura 2.17.: Rol Director del Proyecto

2.2.6.5. Rol Stakeholder

Representa grupos de interés cuyas necesidades deben ser satisfechas por el proyecto. Esto es un rol que podría ser desempeñado por cualquiera que esté (o potencialmente estará) materialmente afectado por el resultado del proyecto.

Figura 2.18.: Rol Stakeholder

2.2.6.6. Rol Pruebas

Es responsable de las actividades principales del esfuerzo de las pruebas. Estas actividades incluyen identificar, definir, implementar y dirigir las pruebas necesarias, como también verificar los resultados de las pruebas y analizar los resultados.

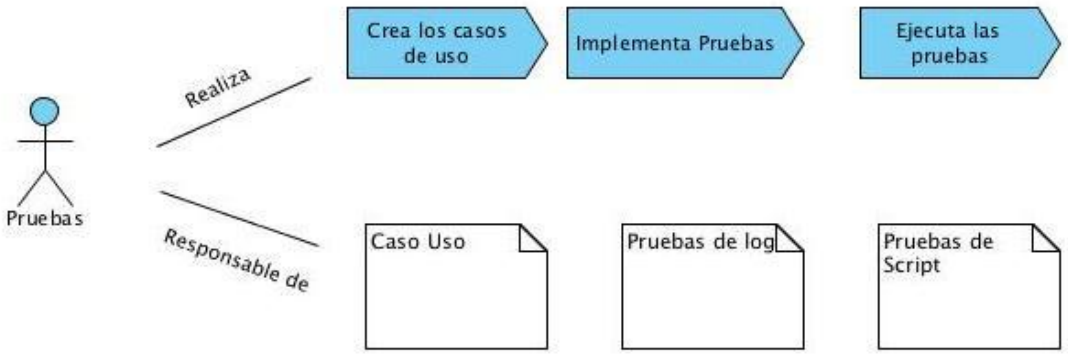


Figura 2.19.: Rol Pruebas

2.2.6.7. Cualquier Rol

Cualquiera en un equipo puede cumplir este rol para llevar a cabo tareas generales.

Figura 2.20.: Cualquier Rol

2.3. DEFINICIÓN DE ERS - IEEE 830

La IEEE 830 es la Especificación de Requerimientos del Software (ERS), cuyos objetivos principales es proporcionar la ayuda pertinente para la elaboración del documento muy útil de Especificación de Requerimientos de Software.

La especificación dice que la ERS es un “Documento que define, de forma completa, precisa y verificable, los requisitos, el diseño, el comportamiento u otras características de un sistema o componente de un sistema”.[15]

[15] IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology

2.3.1. ¿PARA QUE UTILIZAR LA ERS?

La ERS sirve para:

- Los stakeholders describan claramente las necesidades que tienen, es decir el resultado que quieren de las aplicaciones.

- Para reducir el esfuerzo de análisis, diseño y desarrollo; de manera que se evite realizar de nuevo el trabajo ya hecho.

- Realizarse nuevas versiones, que se mejore el sistema ya establecido.

- Se debe tener en cuenta que la especificación de requerimientos del software no describe ningún tipo de detalle del diseño, metodología de desarrollo del software, modo de implementación o gestión del proyecto.

2.3.2. CARACTERÍSTICAS DE LA ERS

- **Correcto.** No hay un método para determinar si el ERS es correcto, lo importante es que se pida lo que realmente se necesita.

- **Completa.** Debe detallar todas las funcionalidades que debe cumplir el sistema, cuya finalidad es tener claro el alcance que tendrá el software.

- **No ambigua.** Los requerimientos del software deben estar detallados en forma clara y precisa, de modo que cada requisito debe tener una sola interpretación y se evite los malos entendidos de dichos requisitos.

- **Verificable.** Al momento de poder comprobar cada uno de los requisitos del software mediante procesos no excesivamente costosos en las que interviene una persona o un equipo, se puede decir que es verificable.

- **Consistente.** Cuando los requerimientos no poseen ningún tipo de contradicción ni redundancias, se puede decir que es consistente.

- **Ordenado con base en importancia y/o estabilidad.** Cada requerimiento especificado deber tener alguna identificación (número, letra, secuencia alfanumérica) para indicar su grado de importancia o estabilidad.

- **Fácil de modificar.** Se deberá contar con un tipo de estructura consistente con presencia de un índice y existencia de referencias cruzadas.

- **Facilidad para identificar el origen y consecuencia de cada requisito.** Debe especificar si el requisito viene tomado como consecuencia de uno anterior, u originado de un resultado posterior. Esto implica que el trabajo de los desarrolladores se facilite al momento que se deba realizar el mantenimiento del software.

2.3.3. DESCRIPCIÓN DEL PROCESO DE DESARROLLO DE LA ERS

En la figura 2.21se muestra el proceso de desarrollo en la Especificación de Requerimientos del Software.

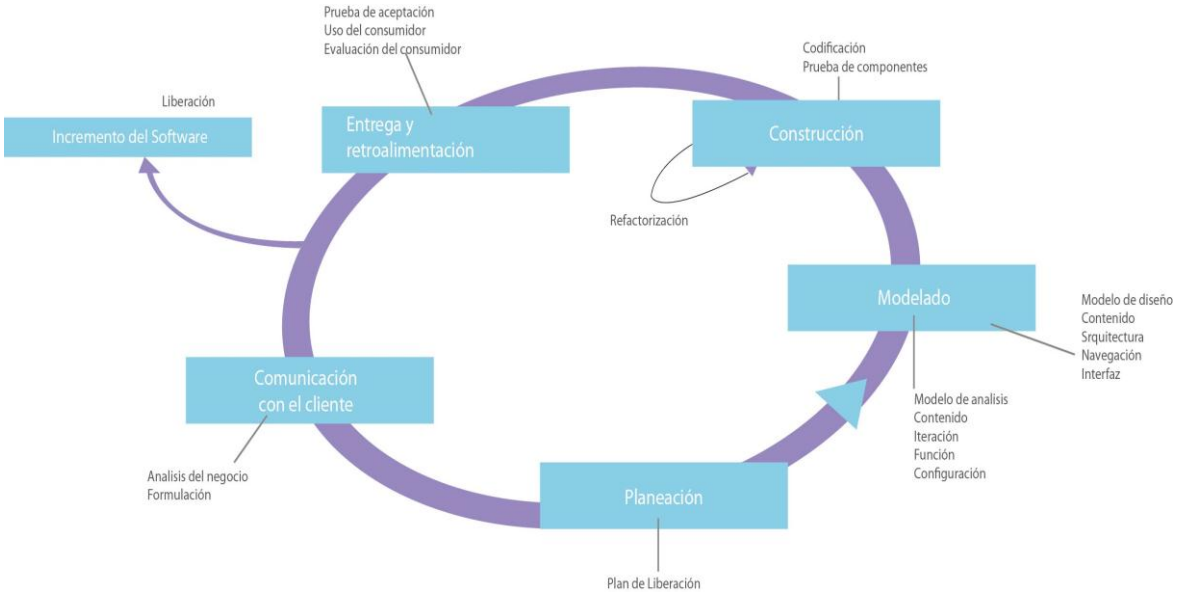


Figura 2.21.: Descripción del proceso de desarrollo de la ERS

En la figura 2.22 se observa el flujo del proceso que existe para obtener requerimientos de usuario en las que interactúan desarrolladores, analistas y diseñadores.

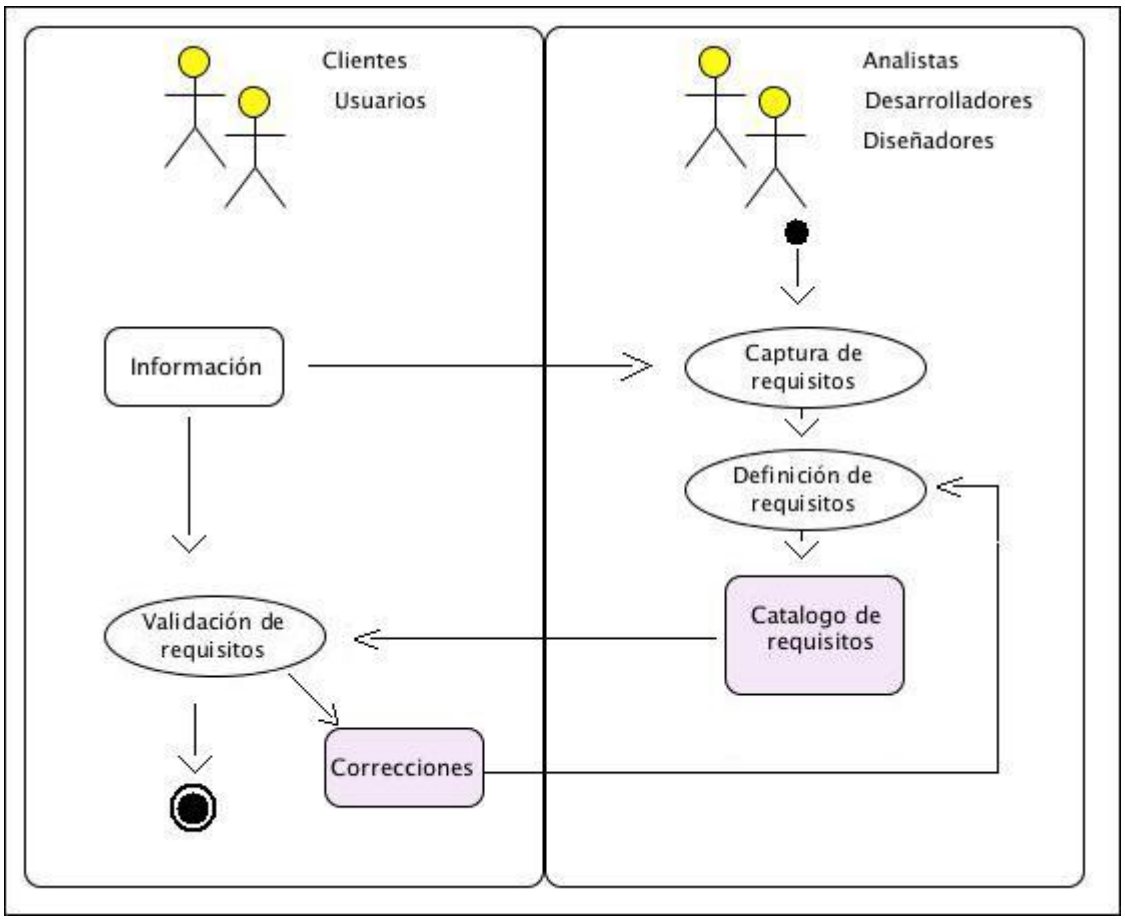


Figura 2.22.: Flujo obtención requerimientos

2.3.4. DOCUMENTO DE LA ERS

Entre el contenido que abarca el ERS están:

1. INTRODUCCIÓN

1.1 Propósito

1.2 Alcance

1.3 Definiciones, Acrónimos y Abreviaturas

1.4 Referencias

1.5 Visión General

2 DESCRIPCIÓN GENERAL

2.1 Perspectiva del Sistema

- Interfaz del sistema
- Interfaz de usuario
- Interfaz de hardware
- Interfaz de software
- Interfaz de comunicaciones
- Interfaz de memoria

2.2 Funciones del sistema

2.3 Características de usuario

2.4 Restricciones

2.5 Suposiciones y dependencias

3 REQUISITOS ESPECÍFICOS

3.1 Requisitos de las interfaces externas(Requisitos No Funcionales)

- Interfaz de usuario
- Interfaz de hardware
- Interfaz de software

3.2 Requisitos Funcionales

- Flujos de Información

3.3 Requisitos de desempeño

3.4 Requisitos Tecnológicos

3.5 Limitantes de diseño

3.6 Atributos de software

3.7 Otro Requerimientos

2.4 PATRONES DE DISEÑO

2.4.1 INTRODUCCIÓN

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

2.4.2 DEFINICIÓN

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

2.4.3 OBJETIVOS

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitarla reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones, sólo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error.

2.4.4 CATEGORÍAS

Según la escala o nivel de abstracción:

- **Patrones de arquitectura:** Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas software.

- **Patrones de diseño**: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.
    - **Idiomas**: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Además, también es importante reseñar el concepto de Anti patrón de Diseño, que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software.

2.4.5 APLICACIONES

Además de su aplicación directa en la construcción de software en general, y derivado precisamente del gran éxito que han tenido, los patrones de diseño han sido aplicados a múltiples ámbitos concretos produciéndose lenguajes de patrones y completos catálogos de mano de diversos autores.

En particular son notorios los esfuerzos en los siguientes ámbitos:

- **Patrones de interfaces de usuario:** esto es, aquellos que intentan definir las mejores formas de construir interfaces hombre-máquina (HCI, GUI).

- **Patrones para la construcción de sistemas empresariales**, en donde se requieren especiales esfuerzos en infraestructuras de software.

- **Patrones para la integración de sistemas (EAI)**, es decir, para la intercomunicación y coordinación de sistemas heterogéneos.

- **Patrones de workflow, esto es para la definición**, construcción e integración de sistemas abstractos de gestión de flujos de trabajo.

2.5 ARQUITECTURA MVC

2.5.1 DEFINICIÓN

Modelo Vista Controlador es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El modelo es el sistema de gestión de base de datos o archivos planos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista, como se muestra en la figura 2.23

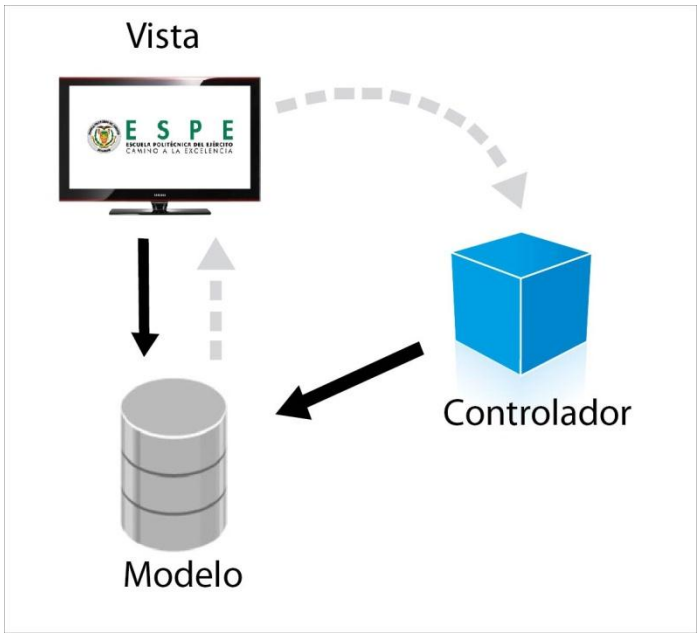


Figura 2.23.: Diagrama Modelo Vista Controlador

2.5.2 DESCRIPCIÓN

2.5.2.1 Modelo:

Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

2.5.2.2 Vista:

Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

2.5.2.3 Controlador:

Este responde a eventos, usualmente acciones del usuario e invoca peticiones al modelo y probablemente a la vista.
Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

i. El usuario interactúa con la interfaz de usuario de alguna forma (porEjemplo: el usuario pulsa un botón, enlace, etc.)

ii. El controlador recibe (por parte de los objetos de la interfaz - vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos.

iii. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo: el controlador actualiza el control remoto de la tv). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

iiii. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón observador para proveer ciertaindirección entre el modelo y la vista,permitiendo al modelo notificar a los interesados de cualquier cambio.

iiv. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio a la vista aunque puede dar la orden a la vista para que se actualice.

iv. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

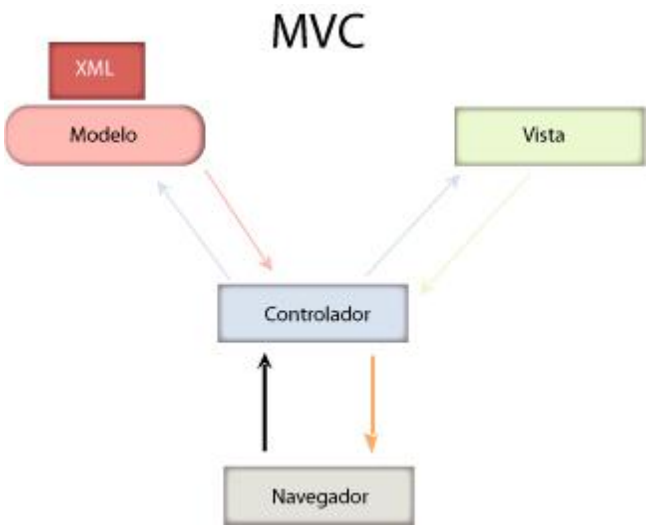


Figura 2.24.: Arquitectura MVC

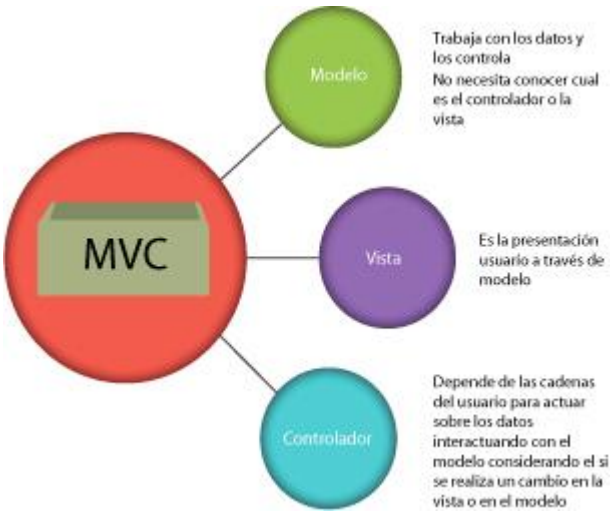


Figura 2.25.: Características del MVC

2.6. XML

2.6.1. DEFINICIÓN

XML es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo. La información estructurada presenta varios contenidos.

2.6.2. ESTRUCTURA XML

Una estructura XML tiene dos estructuras; una lógica y otra física. Físicamente, el documento está compuesto por unidades llamadas entidades. Cada documento comienza con una entidad documento, también llamada raíz.
Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos los cuales están indicados por una manera explícita. Como se muestra en la figura 2.26.

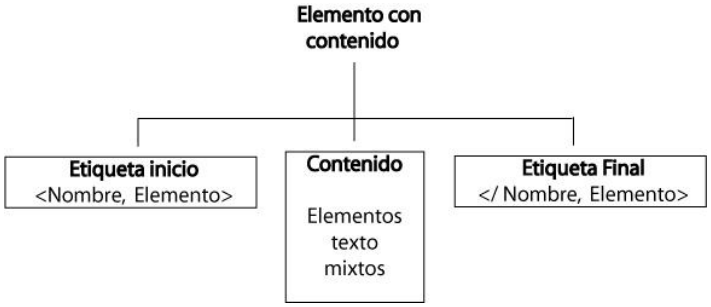


Figura 2.26 Estructura XML

2.6.3. CARACTERÍSTICAS XML

Las características principales de XML son:

- **Extensible;** se puede definir un número ilimitado de etiquetas y además proporciona un marco de trabajo para etiquetado de datos estructurado.

- **Representación estructural de los datos;** proporciona un estándar de datos que puede codificar el contenido, la semántica y el esquema de una amplia variedad de casos que van desde simples a complejos.

- **Los datos son separados de la presentación y el proceso;** permite desplegar y procesar los datos tal como se desee, aplicando diferentes hojas de estilo y aplicaciones, que permite una integración de datos perfecta de fuentes diversas.

- **Conversión de los datos XML en auto descriptivos;** los datos codificados en XML son auto descriptivos, pues las etiquetas descriptivas están entremezcladas con los datos. El formato abierto y flexible utilizado por XML permite su uso en cualquier lugar donde sea necesario intercambiar y transferir información.

2.6.4. PARSING XML

Es la herramienta principal de cualquier XML. Se puede incorporar a nuestras aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML, como se muestra en la figura 2.27

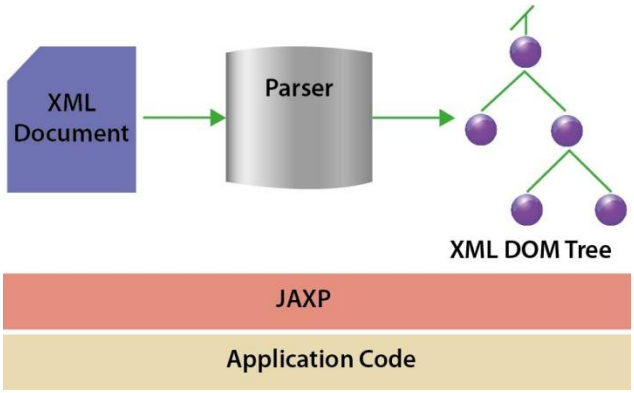


Figura 2.27.:Parsing XML

2.7. JAVA

2.7.1. INTRODUCCIÓN

Java es un lenguaje de programación orientado a objetos, desarrollado por SunMicrosystema principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.[16]

2.7.2. CARACTERÍSTICAS

Entre las características principales que nos ofrece Java son:

- **Orientado a Objetos**
- **Distribuido y Dinámico**
- **Robusto**
- **Seguro**
- **Multitarea**
- **Portable**

Java tiene la característica de al mismo tiempo compilado e interpretado. El compilador es el encargado de convertir el código fuente de un programa en un código intermedio llamado bytecode[17] que es independiente de la plataforma en que se trabaje y que es ejecutado por el intérprete Java que forma parte de la Máquina Virtual de Java.

[16] http://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)

[17] Código intermedio entre el código fuente y el código máquina. Suele tratárselo como un fichero binario que contiene un programa ejecutable similar a un módulo objeto.

2.7.3. MÁQUINA VIRTUAL JAVA

Es el núcleo del lenguaje de programación de Java, en la MVJ se encuentra el motor que en realidad ejecuta el programa Java y es la clave de muchas características principales de Java, como la portabilidad, la eficiencia y la seguridad.

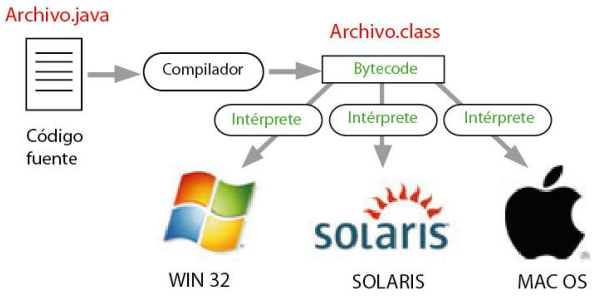


Figura 2.28.: Funcionamiento Máquina Virtual Java

CAPÍTULO III

ANÁLISIS Y DISEÑO

3.1. MODELO DE NEGOCIO

En la figura 3.1 se representa el modelo de los diferentes procesos tecnológicos de la Escuela Politécnica del Ejercito (ESPE).

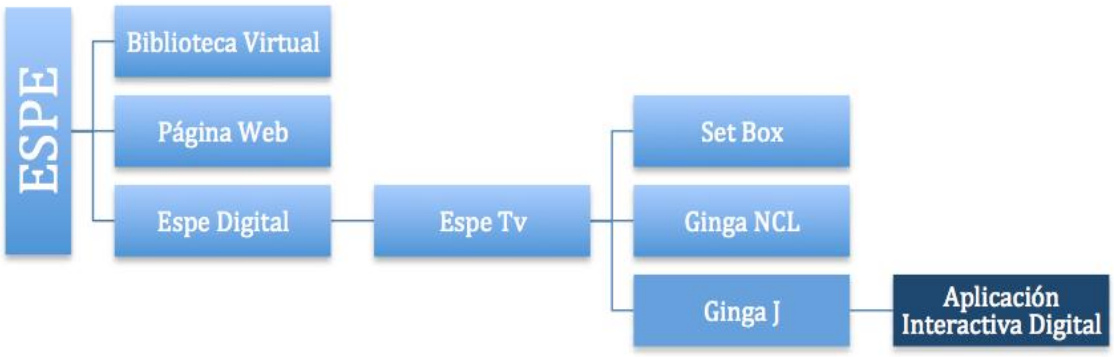


Figura 3.1.: Modelo de procesos tecnológicos de la ESPE.

A continuación en la figura 3.2 se muestra los sub procesos de Ginga J.

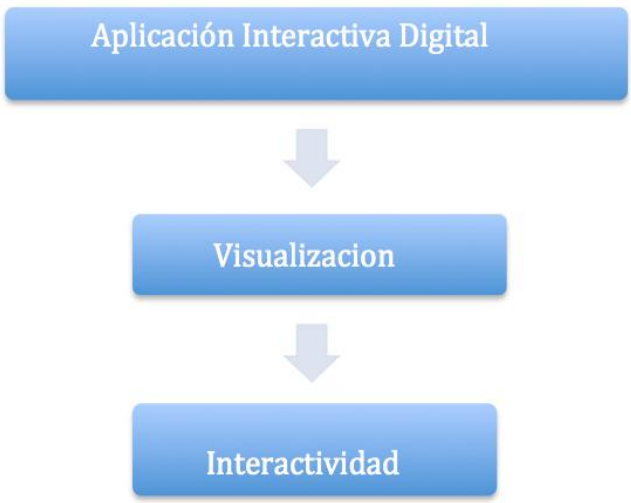


Figura 3.2.: Descripción de Sub Procesos de la Dirección de la ESPE TV

- **Proceso Principal:** Aplicación Interactiva Digital
- **Subproceso:** Visualización**.**

Este subproceso es la parte inicial del proceso, el cual muestra el video mediante el emulador settobox.

En la aplicación interactiva digital las personas podrán visualizar información de la Escuela Politécnica del Ejército.

- **Subproceso:** Interactividad**.**

Este subproceso es la parte donde los usuarios podrán interactuar mediante dispositivos de entrada (control remoto), para poder navegar entre los menús que contendrá información de las diferentes facultades que posee la Escuela Politécnica del Ejército.

3.2. PLAN DE PROYECTO

3.2.1. INTRODUCCIÓN

El objetivo presente trabajo es mostrar el desarrollo de un aplicación interactiva para el proyecto ESPE-GINGA bajo el MiddellwareGinga J basada en la metodología Open UP / Basic para el estándar brasileño de televisión digital.

3.2.2. ORGANIZACIÓN DEL PROYECTO

El trabajo se divide en una serie de áreas de contenido. Cada área de contenido está dirigido por una o varias personas asignadas a un rol, las cuales serán responsables de asegurarse de que cada tarea se cumpla en los tiempos establecidos.

Tabla 3.1. Asignación de Roles

Miembros	Stakeholders	Analista	Arquitecto	Desarrollador	Pruebas	Director de Proyecto
Ing. Danilo Martínez	**X**					**X**
Ing. Santiago Salvador	**X**					**X**
Ángel Quingaluisa		**X**		**X**		
Jonathan Torres		**X**	**X**		**X**	

3.2.3. PRÁCTICAS DEL PROYECTO Y MEDIDAS

Para el desarrollo de la aplicación iterativa digital se desarrolla con OpenUP/Basic, con el objetivo de tener una guíapráctica del desarrollo de la aplicación y no solo una codificación.

Los Artefactos clave que incluye Open UP / Basic:

- Plan de Proyecto.
- Lista de elementos de trabajo.
- Plan de Iteración.

3.2.4. ETAPAS Y OBJETIVOS DEL PROYECTO

Esta sección cubre los objetivos para la aplicación interactiva digital.

- Proceso de interactividad emulador-usuario.
- Un conjunto de interfaces definidas y establecidas.
- Desarrollo de una aplicación interactiva, aplicando la metodología OpenUP / Basic.

Tabla 3.2 Fases e Iteraciones del Proyecto

Fase	Iteración	Objetivo Primario	Fechas de Programación	Estimación de dura-ción (días)
Inicio	I1	Objetivo ✓ Aprobación del proyecto de investigación	01/01/2011 28/02/2011	59 días
Inicio	I2	Objetivo ✓ Reunión confirmando. La organización del proyecto y el plan acordado. ✓ Acuerdo sobre la estructura de OpenUP, ponerse de acuerdo sobre cómo estructurar OpenUP en unidades de trabajo, los principios y los recursos para desarrollar cada unidad. ✓ Identificar las actividades y roles de cada actor.	07/03/2011 10/03/2011	3 días
Elaboración	I3	Objetivo ✓ Estudiar la arquitectura a usar en el proyecto ✓ Investigar la especificación de requerimientos con el estándar IEEE 830	11/03/2011 16/03/2011	5 días

Elaboración	I4	Objetivo ✓ Desarrollo de la documentación de usuario ✓ Desarrollo de la Arquitectura MVC ✓ Elaboración casos de uso, diagramas de secuencia y navegación	20/03/2011 03/04/2011	15 días
Elaboración	I5	Objetivo ✓ Investigar el funcionamiento e instalación del emulador Open Ginga Java ✓ Investigar el funcionamiento y herramientas que se utiliza para el desarrollo de la aplicación. ✓ Investigar funcionamiento de Eclipse Elios con Ginga Java	11/04/2011 19/04/2011	9 días
Construcción	I6	Objetivo ✓ Edición de vídeo para utilizarlo en el emulador Open Ginga ✓ Pruebas sobre el API GINGA J ✓ Desarrollo de clases genéricas para la aplicaron. ✓ Desarrollo de eventos e interfaces utilizadas para la aplicación. ✓ Revisión de la aplicación y documentación de la misma	16/05/2011 14/06/2011	29 días
Construcción	I7	Objetivos		

		✓ Edición de iconos e imágenes a utilizar en el aplicativo. ✓ Desarrollo de la aplicación mediante patrones de diseño y con los estándares de programación.	15/06/2011 30/06/2011	15 días
Construcción	I8	Objetivos ✓ Pruebas de la aplicación ✓ Desarrollo y corrección de la aplicación	01/07/2011 06/07/2011	6 días
Construcción	I9	Objetivos ✓ Pruebas finales de la aplicación ✓ Documentación final de la aplicación	08/07/2011 13/07/2011	6 días
Transición	I10	Objetivos ✓ Final de la aplicación ✓ Finalizar el contenido ✓ Finalizar manual instalación ✓ Finalizar manual de usuario	15/08/2011 30/08/2011	15 días

3.3. PLAN DE ITERACIONES

3.3.1 HITOS CLAVES

Tabla 3.3 Hitos del Proyecto

Hito	Fecha
Aprobación del proyecto de investigación.	28/02/11
Identificación y asignación de los roles con los cuales se va desarrollar el proyecto	10/03/11
Búsqueda de la arquitectura que se adapte al desarrollo de la aplicación	1403/11
Investigación de los estándares y pasos a seguir para el desarrollo de la IEEE 830 en nuestro proyecto.	16/03/11
Realizar la documentación y levantamiento de requerimientos con los estándares IEEE 830, asignar y desarrollar la arquitectura que se va utilizar (MVC).	28/03/11
Desarrollar los casos de uso, secuencia y navegación de la aplicación	03/04/11
Instalación correcta del Emulador Open Ginga Java	13/04/11
Comprobación de la configuración del Emulador, pruebas del funcionamiento de eclipse Elios con el JDK 1.4.2 que utiliza Ginga J	19/04/11
Verificar el vídeo editado en el emulador Open Ginga	18/05/11
Desarrollo de los menús utilizados para la aplicación	14/06/11
Estudio y desarrollo del la aplicación utilizando la arquitectura MVC	30/06/11

Verificación de la correcta codificación de la aplicación utilizando patrones de diseño	30/06/11
Pruebas sobre la aplicación interactiva digital	02/07/11
Corrección de errores y fallas de la aplicación	06/07/11
Pruebas sobre la aplicación interactiva digital	13/07/11
Entregables de la aplicación y documentación	30/08/11
Detener la iteración	31/08/11

3.3.2 OBJETIVOS DE ALTO NIVEL

3.3.2.1. ASIGNACIONES DE TRABAJO PARA LA APLICACIÓN

Tabla 3.4 Asignación de Trabajo

Nombre / Descripción del trabajo	Prioridad	Estimar el tamaño en (puntos)	Asignado a	Estimar el esfuerzo (horas)
Realizar el diseño			• ÁngelQuingaluisa • Jonathan Torres	12
Implementar y probar parte de la aplicación			• Jonathan Torres	4
Actualización de la documentación para el usuario final			• ÁngelQuingaluisa • Jonathan Torres	6
Producir demo para la ESPE-GINGA	3	5	• ÁngelQuingaluisa • Jonathan Torres	40
Edición de documentación del usuario final	2	5	• ÁngelQuingaluisa • Jonathan Torres	85
Realizar cambios demo	2	1	• ÁngelQuingaluisa • Jonathan Torres	20
Edición manual de instalación	2	1	• Jonathan Torres	5

Editar notas de la versión	2	1	• Jonathan Torres	4
Edición manual de usuario	3	2	• ÁngelQuingaluisa • Jonathan Torres	22
Finalización de la aplicación	3	5	• ÁngelQuingaluisa	10

3.3.3 CRITERIOS DE EVALUACIÓN

- 100% de casos de prueba a nivel del sistema aprobado.
- Interacción de la aplicación se demostró que todos los menús e información funcionan
- Documentación del usuario final se debe obtener la aceptación favorable de los usuarios finales.
- Tutorial de la aplicación sean bien recibidos.
- Respuesta favorable a la aplicación por parte de la comunidad Espe-Ginga

3.4. LISTA DE ÍTEMS DE TRABAJO

Tabla 3.5 Lista de Ítems de Trabajo

Nombre / descripción del Ítem	Prioridad	Asignado a	Estimar el esfuerzo (horas)
Realizar diseño de la Arquitectura	3	• Jonathan Torres	12
Realizar diseño de iconos, menús, etc.	1	• Jonathan Torres	8
Implementar demo de la aplicación	2	• ÁngelQuingaluisa	4
Pruebas de la aplicación	2	• Jonathan Torres	8
Desarrollo del 40% documentación para el usuario final	1	• Ángel Quingaluisa • Jonathan Torres	40
Realizar mejoras al demo de la aplicación	3	• ÁngelQuingaluisa	16
Pruebas a las mejoras de aplicación	2	• Jonathan Torres	8
Desarrollo del 70% documentación para el usuario final	1	• ÁngelQuinagluisa • Jonathan Torres	40
Finalización de la aplicación	3	• ÁngelQuingaluisa	40

Desarrollo del 100% documentación para el usuario final	3	• ÁngelQuingaluisa • Jonathan Torres	40
Desarrollo de manuales de instalación	1	• ÁngelQuingaluisa • Jonathan Torres	20
Desarrollo de manuales de usuario	1	• ÁngelQuingaluisa • Jonathan Torres	20

3.4. ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE (ERS)

3.4.1. INTRODUCCIÓN

3.4.1.1. PROPÓSITO

Este documento, presenta la especificación de requerimientos de software de la aplicación interactiva.

Esta especificación se basa en el estándar IEEE 830. Pretende abstraer principalmente los conceptos funcionales de la aplicación interactiva que se espera realizar en base a los requerimientos obtenidos para el desarrollo de la misma.

La aplicación interactiva pretende avanzar con la investigación y el proyecto que tiene en curso la ESPE, específicamente el Departamento de Ciencias Electrónicas conjuntamente con el Departamento de Ciencias de la Computación e Informática.

3.4.1.2. ALCANCE

El presente proyecto tiene como objetivo principal realizar una aplicación que permita mediante dispositivos de entrada de información la interacción entre los usuarios y la aplicación, para lo cual se definirán procesos y entes que intervienen en esta actividad.

3.4.1.3. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

Tabla 3.6 Definiciones

Usuario	Usuario limitado que puede navegar dentro de la aplicación.
X-let	Son aplicaciones en Java para entornos de televisión.
Emulador	Es un software que permite ejecutar programas en una plataforma diferente de aquella para la cual fueron escritos originalmente.
Control Remoto	Es un dispositivo electrónico usado para realizar una operación remota sobre una máquina.

Tabla 3.7 Acrónimos y Abreviaturas

ESPE	Escuela Politécnica del Ejercito
GINGA-J	GINGA JAVA
JPG	JointPhotographicExpertsGroup
API	ApplicationProgramming Interface
J2ME	Java 2 Micro Edition
IEEE	Instituto de Ingenieros Eléctricos y Electrónicos
XML	Extensible MarkupLanguaje

3.4.1.3. REFERENCIAS

3.4.1.4. VISIÓN GENERAL DEL DOCUMENTO

En este documento se presenta un análisis general de la Aplicación diseñada para la ESPE en su proyecto ESPE-GINGA. Se realizará una breve perspectiva a los propósitos generales, explicando la funcionalidad del mismo. Posteriormente se menciona el alcance del presente proyecto en función de las fases que abarca. En la segunda sección de este documento se realiza una descripción de las principales funciones y factores que inciden a nivel general. En la tercera sección se define detalladamente los requisitos que debe satisfacer.

3.4.2. DESCRIPCIÓN GENERAL

3.4.2.1. PERSPECTIVA DEL SISTEMA

La aplicación es netamente interactiva y visual, funcionara en el emulador de TV (OPEN GINGA JAVA) y los usuarios podrán interactuar con ella con el uso de dispositivos de entrada (Control Remoto).
Para este propósito se presenta las siguientes especificaciones de funcionamiento:

1. Interfaz del Sistema:

Open Ginga

2. Interfaz de Usuario:

Operador:

Función Operador

Modificar archivo contenido XML

Usuario

Funciones Usuario

Interactuar con la televisión digital

3. Interfaz de Hardware

- Computador con las siguientes especificaciones:
- Procesador Intel o AMD a 1 Ghz
- Memoria RAM de 512 MB
- Tarjeta gráfica de 512 Mb o superior
- Disco duro de 20 GB

4. Interfaz de Software

La aplicación estará alojada en el emulador Open Ginga, el cual se ejecuta sobre Linux Ubuntu 10.0.4, Máquina Virtual Java J2SDK 1.4.0.2, XML, Archivos Planos.

5. Interfaz de Comunicaciones

El usuario interactuará con la aplicación mediante el teclado el cual está configurado el cual está configurado para emular las opciones de un control remoto cuando se utilice Open Ginga Java.

6. Interfaz de Memoria

Memoria RAM de 512 MB (DDR, DDR2, DDR3, SODIMM, SIMM).

3.4.2.2 FUNCIONES DE LA APLICACIÓN

Esta aplicación es netamente demostrativa, cuya única función es interactuar con el usuario mediante el emulador de TV Digital Open GingaJava en cual esta corriendo la misma

3.4.2.2.1. CARACTERÍSTICAS DE USUARIO

Los usuarios de la aplicación digital podrán ser de distinto nivel de educación académica, desde aquellos que conocen el uso aplicaciones digitales que recién se encuentran inmersos en los mismos. Dentro de la Aplicación se manejará dos tipos de usuarios:

- **Operador**

  Su única tarea dentro de la aplicación es modificar los contenidos del archivo XML.

- **Usuario – Persona**

  Este tipo de usuario podrá interactuar con la aplicación mediante el uso de dispositivos de entrada de información (Control Remoto).

3.4.2.2.2. RESTRICCIONES

Memoria. Restricciones en cuanto a la memoria del emulador Open Ginga Java, posee pocos recursos para ejecutar las aplicaciones interactivas.

SDK. Restricción en cuanto al SDK, la versión actual que utiliza Open Ginga Java es muy antigua, por tal motivo sus funcionamientos y sus componentes son limitados y otros no funcionales.

3.4.2.2.3. SUPOSICIONES Y DEPENDENCIAS

3.4.2.2.3.1. SUPOSICIONES

La aplicación es demostrativa y cualquier cambio o modificación a la misma, deberá realizarse dentro del Proyecto ESPE-GINGA una vez finalizada está investigación.

3.4.2.2.3.2. DEPENDENCIAS

La aplicación interactiva digital solo es adaptable en el emulador Open Ginga Java y en el sistema operativo Ubuntu 10.0.4.

3.4.3. REQUISITOS ESPECÍFICOS

3.4.3.1. REQUISITOS NO FUNCIONALES

3.4.3.1.1. INTERFAZ DE USUARIO

El usuario podrá interactuar con la aplicación y utilizar los diferentes componentes gráficos.

Tabla 3.8 Componentes gráficos de interfaz de usuario

OBJETO	DESCRIPCIÓN
lwit.Label	Permite ingresar información como texto e iconos en una línea específica.
Imagen.png	Imagen con un formato basado en un algoritmo de compresión.

3.4.3.1.1. INTERFAZ DE HARDWARE

Computador con las siguientes especificaciones:

- Procesador Intel o AMD a 1 Ghz
- Memoria RAM de 512 MB (DDR, DDR2, DDR3, SODIMM, SIMM)
- Tarjeta gráfica de 512 Mb o superior
- Disco duro de 20 GB

3.4.3.1.2. INTERFAZ DE SOFTWARE

La aplicación interactiva funciona sobre un sistema Ubuntu 10.0.4., el cual aloja al emulador Open Ginga Java donde se encuentra la aplicación.

3.4.3.1.3. INTERFAZ DE COMUNICACIONES

El usuario interactuará con la aplicación mediante el teclado el cual está configurado el cual está configurado para emular las opciones de un control remoto cuando se utilice Open Ginga Java.

3.4.3.2. REQUISITOS FUNCIONALES

Se especifica los requerimientos funcionales, los cuales definen la expectativa del usuario frente a la aplicación. Los mismos han sido descritos a un nivel esencial sin detalle de los mecanismos informáticos requeridos para resolverlo.

- La aplicación tendrá información sobre las diferentes carreras y departamentos que posee la Escuela Politécnica del Ejército.

- La aplicación deberá permitir la navegabilidad e interactividad emulador usuario.

- El operador podrá modificar el contenido del archivo XML.

- Para la navegabilidad la aplicación tendrá menús definidos estáticamente en los cuales se listaran las diferentes carreras y departamentos.

3.4.3.3. REQUISITOS DE DESEMPEÑO

- El ingreso a la información de la aplicación está clasificado por un menú estático el cual contiene los diferentes enlaces sobre la información de las diferentes carreras y departamentos de la Escuela Politécnica del Ejército.

- Para actualizar la información de manera limitada el operador deberá modificar el archivo de contenido XML.

- Hay que tomar en cuenta que el desempeño e interactividad con nuestra aplicación depende del emulador Open Ginga Java.

3.4.3.4. REQUISITOS TECNOLÓGICOS

El único requisito tecnológico que necesita el usuario para la utilizar la aplicación es:

- Procesador Intel o AMD a 1 Ghz
- Memoria RAM de 512 MB.
- Tarjeta gráfica de 512 Mb o superior
- Disco duro de 20 GB
- Adicionalmente tener instalado el sistema operativo Ubuntu 10.0.4

3.4.3.5. REQUISITOS DE SEGURIDAD

La aplicación no tiene requisitos de seguridad ya que es una aplicación netamente demostrativa.

3.4.3.6. LIMITANTES DE DISEÑO

Memoria. Restricciones en cuanto a la memoria del emulador Open Ginga Java, posee pocos recursos para ejecutar las aplicaciones interactivas.

SDK. Restricción en cuanto al SDK, la versión actual que utiliza Open Ginga Java es muy antigua, por tal motivo sus funcionamientos y sus componentes son limitados y otros no funcionales.

3.4.3.7. ATRIBUTOS DE SOFTWARE

Las características que presenta la aplicación interactiva son las siguientes:

- Facilidad de mantenimiento
- Interfaz gráfica
- Facilidad de uso

3.4.3.8. OTROS REQUISITOS

Realizar un manual de usuario y operaciones para las personas que utilizan y dan mantenimiento al aplicativo.

3.5. MODELO DE CASOS DE USO

3.5.1. CASO DE USO APLICACIÓN INTERACTIVA

Para explicar de mejor manera el funcionamiento de la Aplicación se utilizará diagramación de casos de uso.

En la presente figura 3.3 se muestra el diagrama de caso de uso de la Aplicación Interactiva Digital.

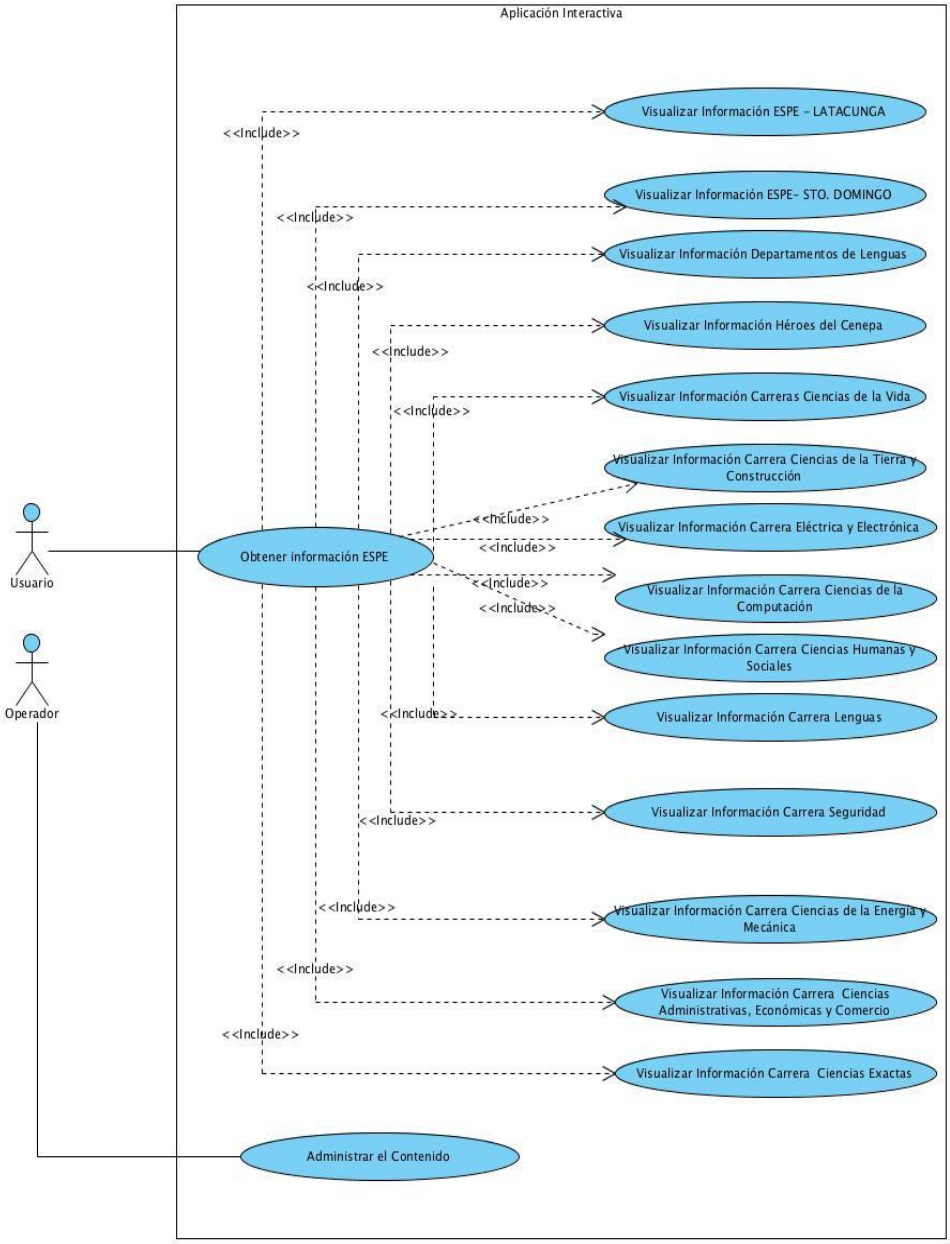


Figura 3.3. Diagrama de Casos de Uso Aplicación Interactiva.

3.5.1.1. CASO DE USO OBTENER INFORMACIÓN ESPE

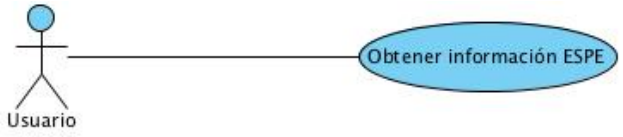


Figura 3.4.: Diagrama de Casos de Uso Obtener Información ESPE

La siguiente tabla describe el caso de uso Obtener Información ESPE

Tabla 3.9 Descripción de caso de uso Obtener Información ESPE

Identificación:	1.1
Nombre:	Obtener Información ESPE
Descripción: Este proceso permite obtener información de las diferentes campos y carreras de la Escuela Politécnica del Ejercito	
Actores: • Usuario de la Aplicación	
Precondiciones: • N/A	
Flujo Normal: • N/A	
Flujo Alternativo: • N/A	

3.5.1.2. CASO DE USO ADMINISTRAR EL CONTENIDO

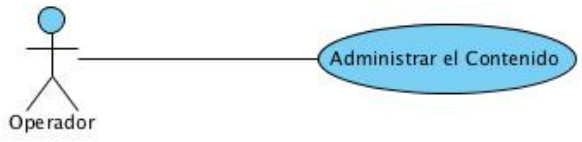


Figura 3.5.: Diagrama de Casos de Uso Administrar el Contenido

La siguiente tabla describe el caso de uso Administrar el Contenido

Tabla 3.10 Descripción de caso de uso Administrar el Contenido

Identificación:	1.2
Nombre:	Administrar el Contenido
Descripción: Este proceso permite manipular la información mediante la modificación de los archivos de contenidos XML.	
Actores: • Operador de la Aplicación	
Precondiciones: • El emulador no debe estar iniciado y se debe respetar a la arquitectura de los archivos XML.	
Flujo Normal: • N/A	
Flujo Alternativo: • N/A	

3.6. MODELO NAVEGACIONAL

Este modelo es útil para percibir mejor la estructura de la aplicación y para mejor la estructura de la navegabilidad.

3.6.1. MODELO DE CLASES NAVEGACIONALES

Se determina las clases que pueden ser vistas a través de la aplicación.

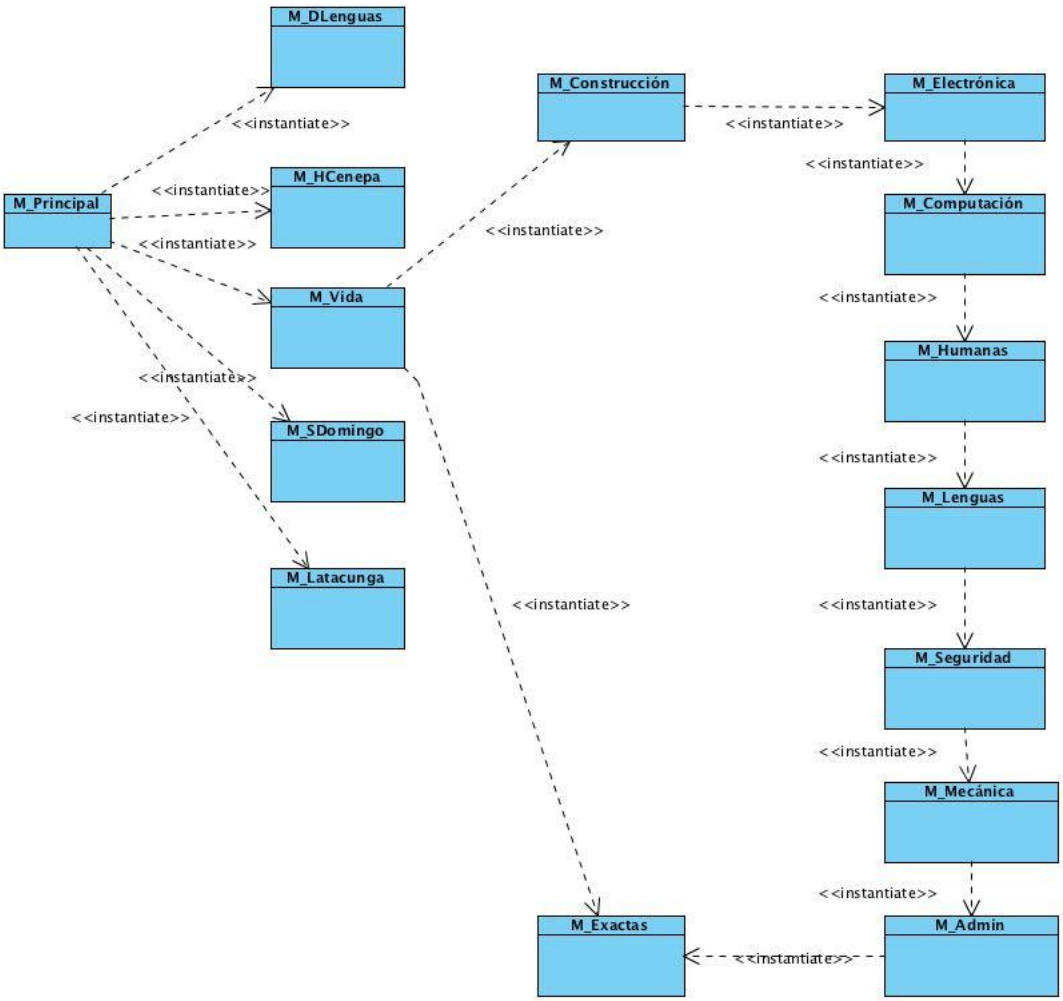


Figura 3.6.: Modelo de Clases Navegaciones de la Aplicación Interactiva

3.6.2. MODELO DE ESTRUCTURA NAVEGACIONAL

El modelo de estructura navegacional define como se alcanzan las clases que podrán ser vistas.

3.6.3. MODELO DE ESTRUCTURA NAVEGACIONAL DEL USUARIO

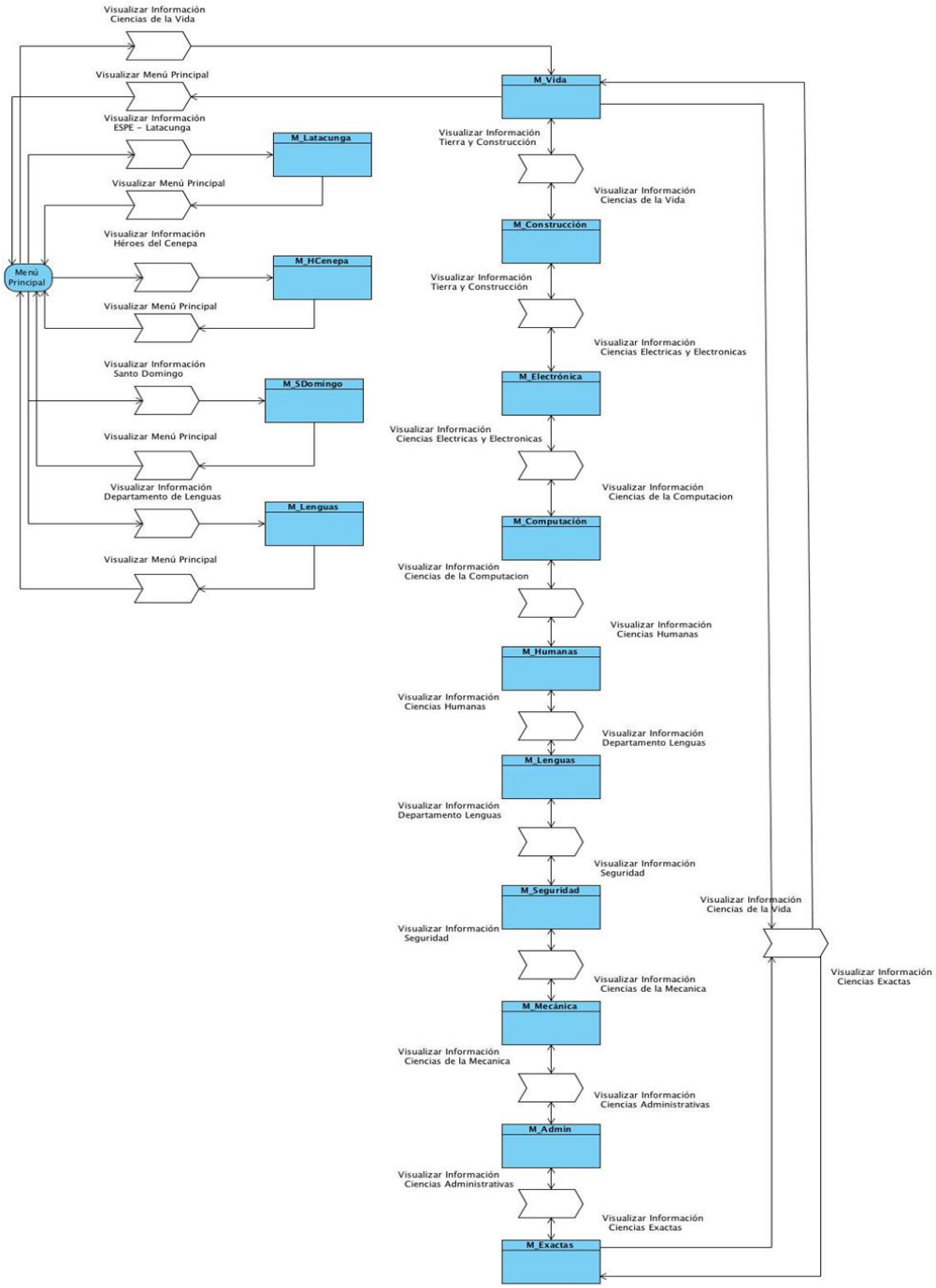


Figura 3.7.: Modelo Navegacional del Usuario

3.7. MODELO DE PRESENTACIÓN

El modelo de presentación consiste en un conjunto de vistas que muestran el contenido y la estructura de las clases y como el usuario puede interactuar con ellos.

El flujo de presentación consiste en modelar la fase de presentación mostrando dónde se presentarán al usuario los objetos de navegación y los elementos de acceso, por ejemplo dónde se muestra el contenido y qué contenido será reemplazado cuando se accione un enlace. El flujo de presentación se visualización modelos de interacción UML (diagramas de secuencia). El modelo de flujo de presentación se basa en los casos de uso.

A continuación se describe los diagramas de secuencia por caso de uso, donde detallamos el modelo, vista, controlador.

3.7.1. DIAGRAMA DE SECUENCIA DEL USUARIO

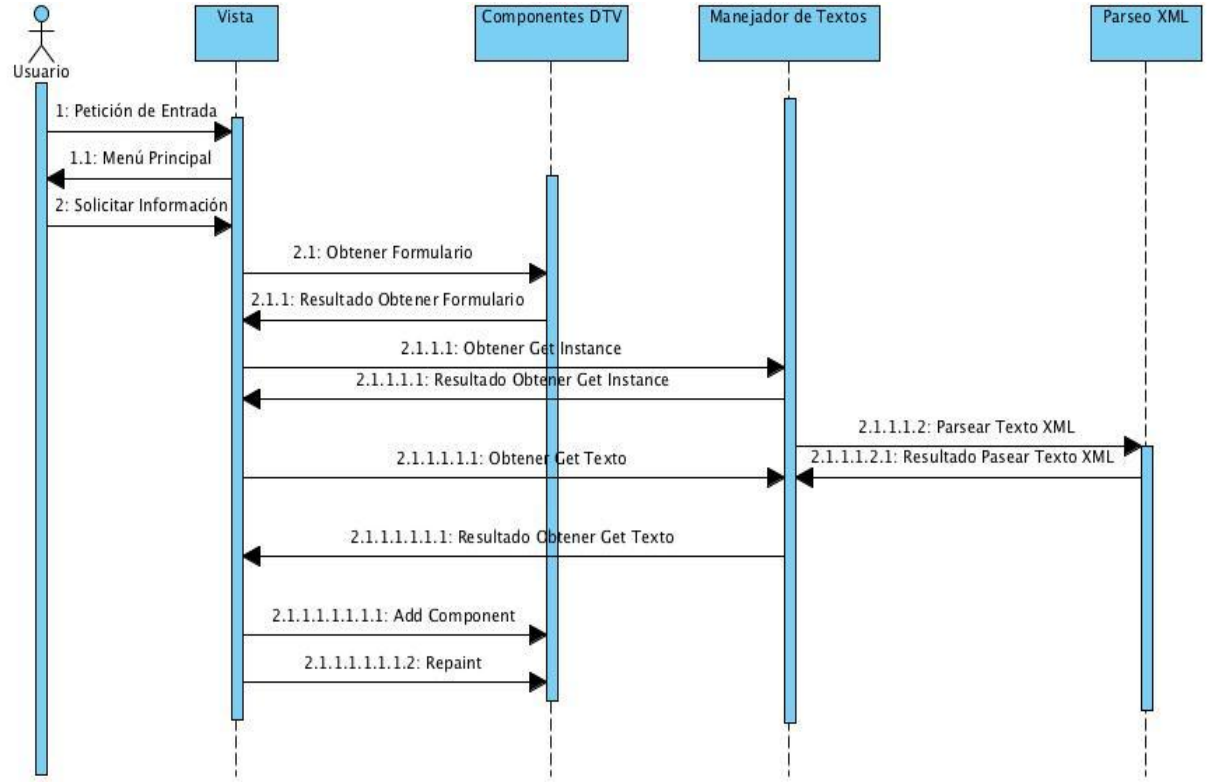


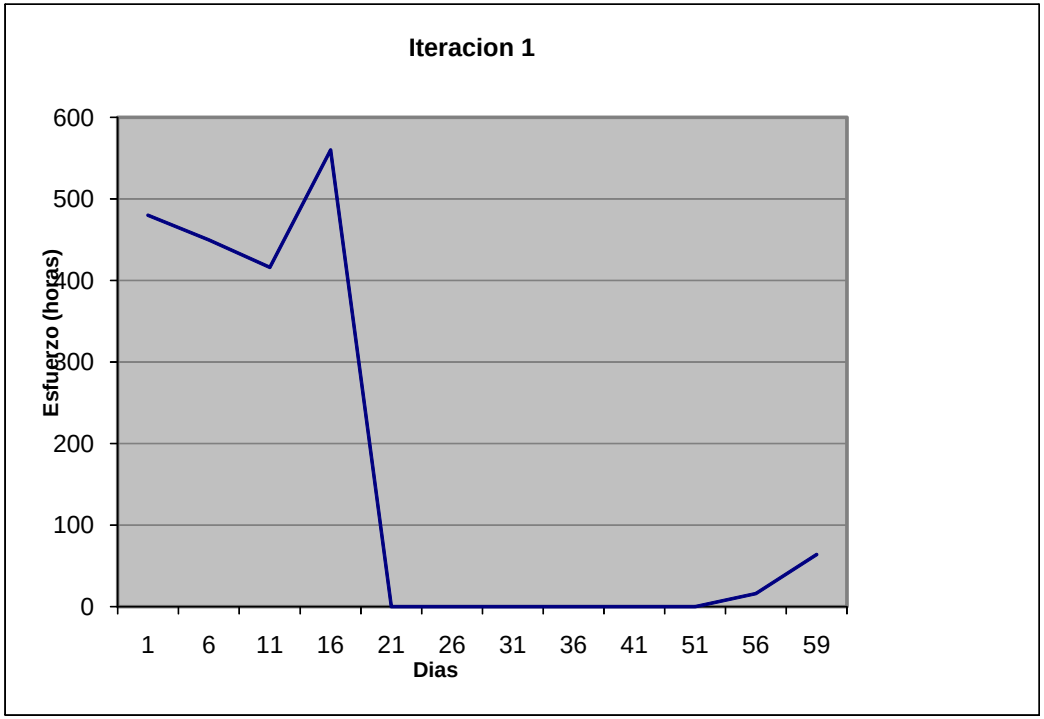
Figura 3.8.: Diagrama de Secuencia

ANEXOS

Iteración 1.: Fase de Inicio

Iteración: I1

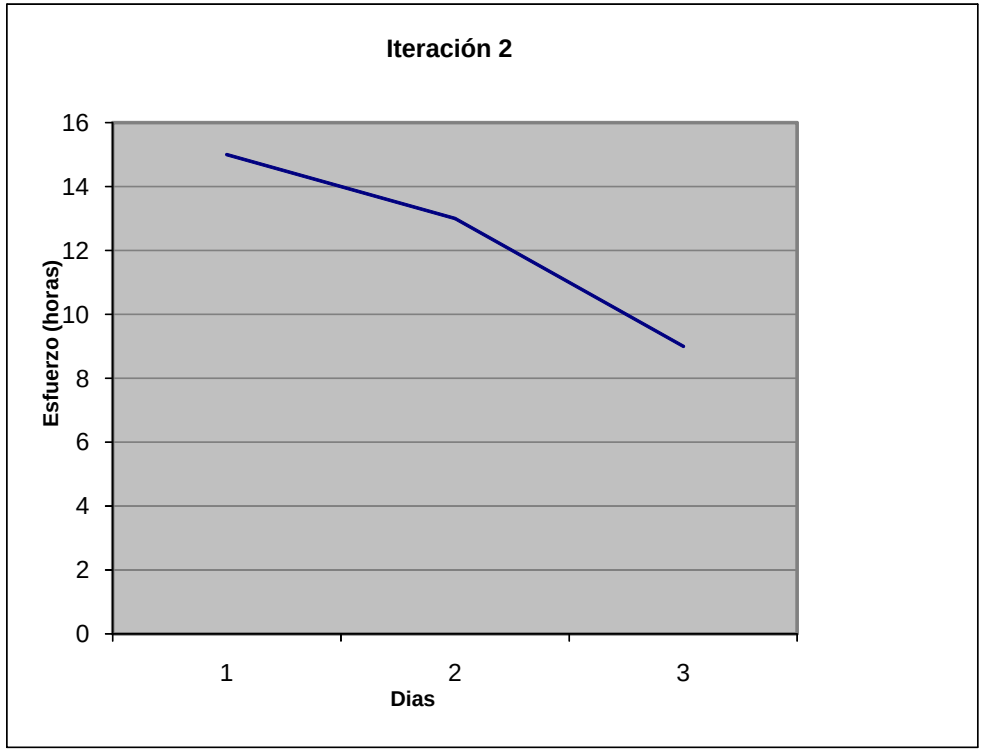
Dias	1	6	11	16	21	26	31	36	41	51	56	59
Esfuerzo (horas) x persona	480	450	416	560	0	0	0	0	0	0	16	64



Iteración 2.: Fase de Inicio

Iteración: I2

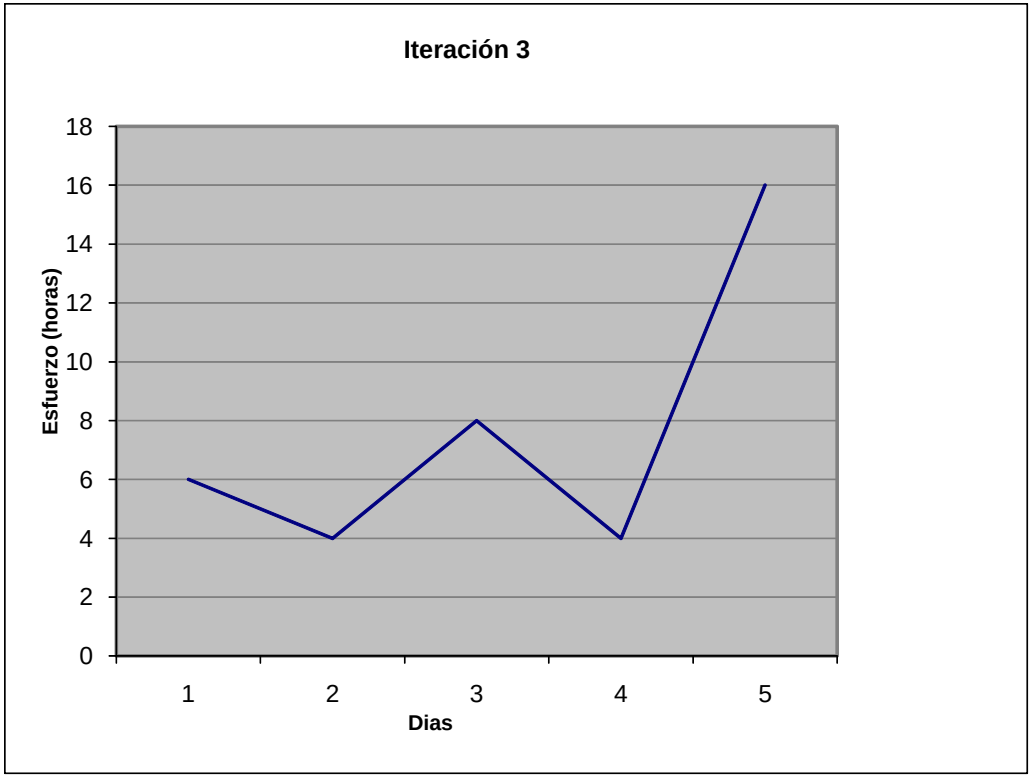
Dias	1	2	3
Esfuerzo (horas) x persona	15	13	9



Iteración 3.: Fase de Elaboración

Iteración: I3

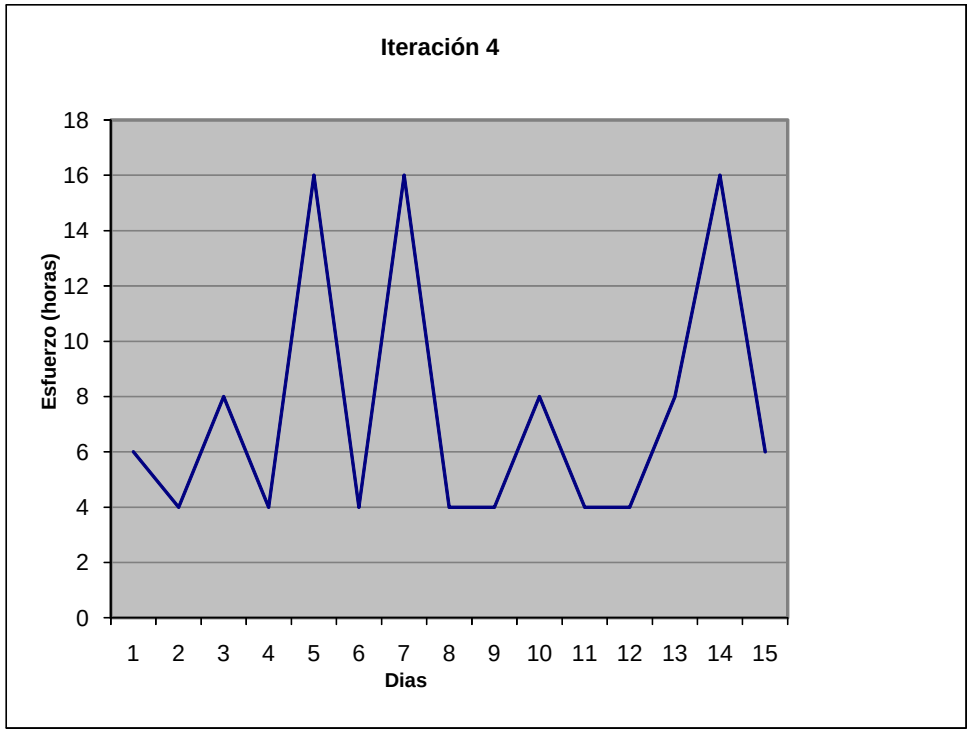
Dias	1	2	3	4	5
Esfuerzo (horas) x persona	6	4	8	4	16



Iteración 4.: Fase de Elaboración

Iteración: I4

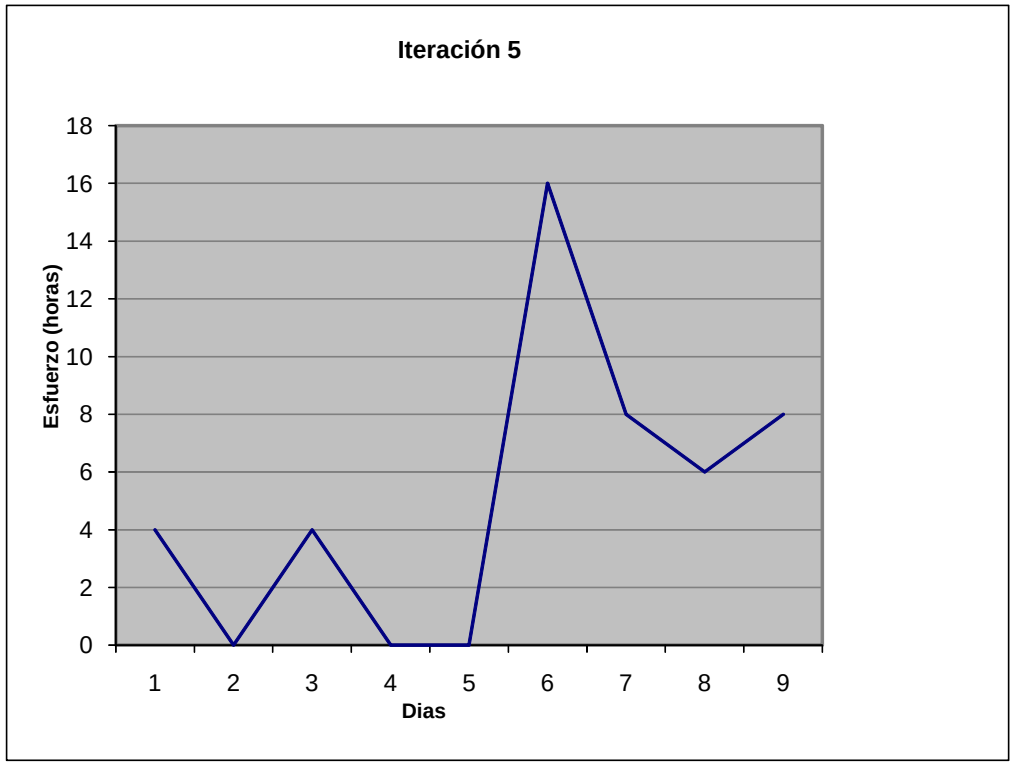
Dias	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Esfuerzo (horas) x persona	6	4	8	4	16	4	16	4	4	8	4	4	8	16	6



Iteración 5.: Fase de Elaboración

Iteración: I5

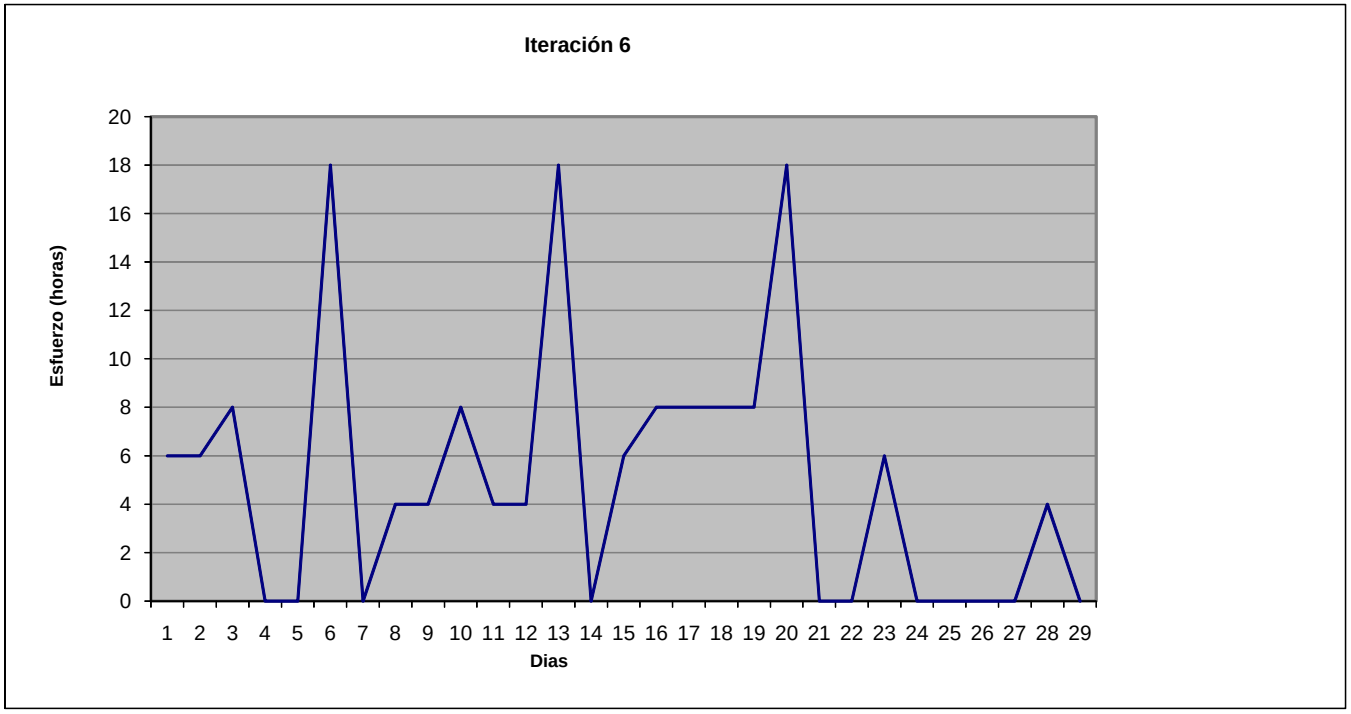
Dias	1	2	3	4	5	6	7	8	9
Esfuerzo (horas) x persona	4	0	4	0	0	16	8	6	8



Iteración 6.: Fase de Construcción

Iteración: I6

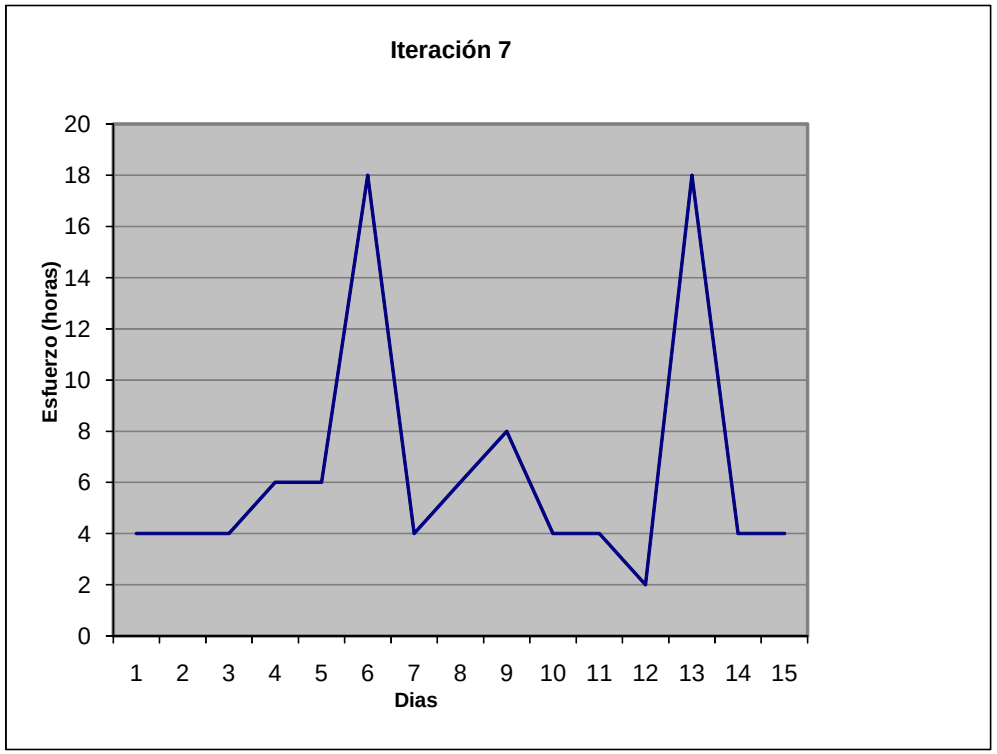
Dias	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Esfuerzo (horas) x persona	6	6	8	0	0	18	0	4	4	8	4	4	18	0	6	8	8	8	8	18	0	0	6	0	0	0	0	4	0



Iteración 7.: Fase de Construcción

Iteración: I7

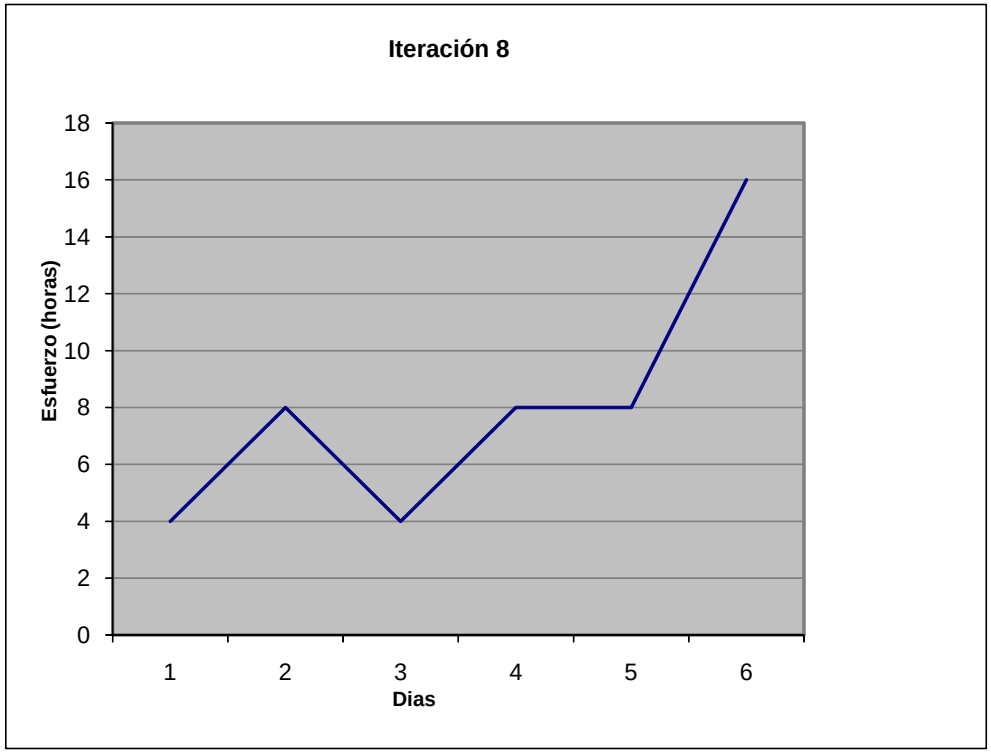
Dias	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Esfuerzo (horas) x persona	4	4	4	6	6	18	4	6	8	4	4	2	18	4	4



Iteración 8.: Fase de Construcción

Iteración: I8

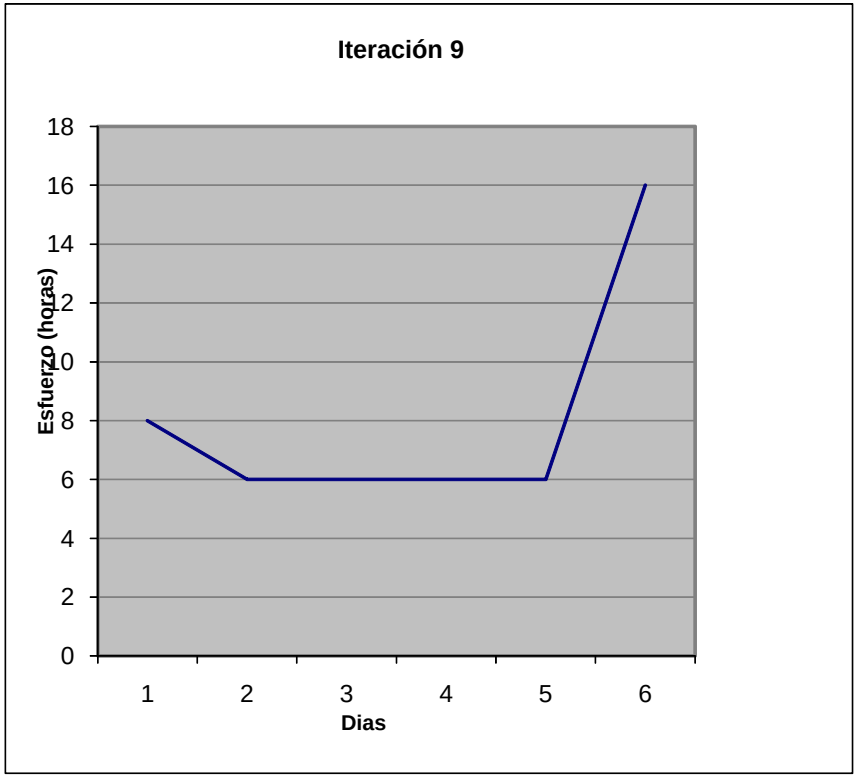
Dias	1	2	3	4	5	6
Esfuerzo (horas) x persona	4	8	4	8	8	16



Iteración 9.: Fase de Construcción

Iteración: I9

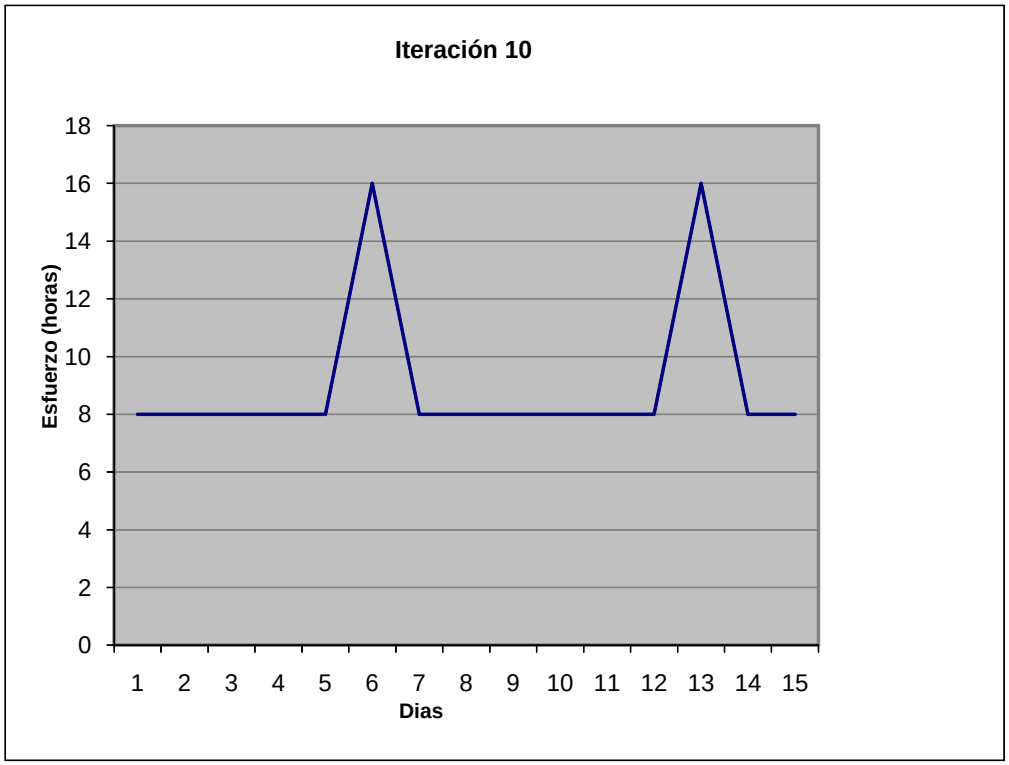
Dias	1	2	3	4	5	6
Esfuerzo (horas) x persona	8	6	6	6	6	16



Iteración 10.: Fase de Transición / Fin de la Aplicación

Iteración: I10

Dias	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Esfuerzo (horas) x persona	8	8	8	8	8	16	8	8	8	8	8	8	16	8	8



CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES.

- La metodología OpenUp / Basic permite el desarrollo de aplicaciones de corto alcance con un grupo pequeño de personas ya que las mismas pueden desempeñar varios roles, así mismo con el uso de las iteraciones en cada una de las fases de desarrollo se puede realizar correcciones preventivas de cada componente a desarrollar.

- La versión actual del emulador OpenGinga tiene limitaciones en la librería API HAVI, por tal razón los objetos gráficos con los que se puede trabajar en el desarrollo de aplicaciones para tv digital aun son limitados.

- El desarrollo de una aplicación en Ginga-J conlleva entender la arquitectura de un Xlet, la cual almacena los componentes gráficos de las aplicaciones basadas en Java TV.

- Los principales problemas que detectamos dentro del desarrollo de este tipo de aplicaciones son 2 que consideramos son de alta importancia, el primero es la restricción en el uso de memoria por lo cual la ejecución de las aplicaciones es limitada y las aplicaciones pueden colapsar; la segunda es la versión de la máquina virtual de java que utiliza el emulador aun se encuentra en una versión básica de tal manera que para realizar la programación orientada a objetos resulta más compleja.

- El middelwareGinga-J no se puede instalar fácilmente en una PC con Sistema Operativo Ubuntu, por esta razón se utiliza la máquina virtual con Ginga-J preinstalado que nos permite levantar un ambiente rápido, siendo esta una opción que está disponible al momento y de esta manera desplegar aplicaciones Ginga J.

- Para el buen funcionamiento de la arquitectura del MiddelwareGinga-J se debe usar las funcionalidades que tienen los API´s estandarizadas Ginga-J, ya que los Xlets para su funcionamiento deben utilizar API´s estandarizados provistos por Ginga-J.

4.2. RECOMENDACIONES

- Para el desarrollo adecuado de aplicaciones de corto alcance se recomienda el uso de la metodología de desarrollo OpenUP / Basic, ya que si se cumple los roles y las iteraciones que se definen en el tiempo estimado tendremos el control que se necesita en el desarrollo de sistemas informáticos.

- El Departamento de Ciencias de la Computación debe crear un club dedicado al desarrollo de aplicaciones de televisión digitales el cual pueda ayudar a la comunidad Ginga-J en el desarrollo de componentes genéricos compatibles con Java Tv para un mejor funcionamiento en los componentes gráficos.

- Para comenzar el desarrollo de aplicaciones de tv digital basadas en Ginga-J se recomienda analizar la estructura de un Xlet aplicando los conocimientos del lenguaje Ginga-J orientado a objetos.

- El proyecto ESPE-GINGA debe investigar las limitaciones de memoria en el emulador para de esta manera aportar en la investigación del proyecto OpenGinga de Lavid.

CAPÍTULO V

BIBLIOGRAFÍA

[1]ABNT NBR 15606-4 – Asociación Brasilera de Normas Técnicas – "Televisión digital terrestre – Codificación de datos y especificaciones de transmisión para radiodifusión digital – Parte 4: Ginga-J – Ambiente para ejecución de aplicaciones de procedimiento" Sistema Brasilero de TV Digital Terrestre, http://www.dtv.org.br/download/pt-br/ABNTNBR15606-4_2010Ed1.pdf,2010.

[2] LEMOS DE SOUZA, Guido, CUNHA, Luiz, COELHO Carlos, "Ginga-J: El Middleware de Procedimiento del Sistema de TV Digital Brasilera", Departamento de Informática, Universidad Federal de Paraíba, http://www.tvdi.inf.br/upload/artigos/gingaj_the_procedural_middleware_for_brazilian_digital_tv_system.pdf, revisado en septiembre de 2009.

[3]AMATLLER, Jordi, BALDO, David, BENELLI, Giuliano, DAINO, Giovanni, Zambon, Ricardo, "Televisión Digital Terrestre Interactiva, el desafío de la interoperabilidad en Brasil", Departamento de la Ingeniería en Informática de la Universidad de Siena, http://downloads.hindawi.com/journals/ijdmb/2009/579569.pdf, revisado en enero de 2010.

[4]PÉREZ GARCÍA, Nelson, "Televisión Digital Terrestre e Interactividad", Universidad de los Andes, http://gitel.ing.ula.ve/Curso-TDT-CENDIT-Bloque2-Parte2.pdf, revisado en noviembre de 2009.

[5]Página Oficial de Ginga Ecuador: http://www.ginga.org.ec/, 2011.

[6]Página Oficial de Ginga Argentina: http://www.ginga.org.ar/, 2011.

[7]Página Oficial de Ginga Brasil: http://www.ginga.org.br/, 2011.

[8]Página Oficial de Ginga Perú: http://www.ginga.org.pe/, 2011.

[9]Página Oficial de Software Público de Brasil:http://www.softwarepublico.gov.br/, 2011.

[10]Página Oficial del Club NCL de Brasil: http://clube.ncl.org.br/, 2011.

[11]Página Oficial del Laboratorio TeleMídiahttp://www.telemidia.puc-rio.br, 2011.

[12]Página Oficial del Laboratorio LAVID: http://www.lavid.ufpb.br/, 2011.

[13]Página Oficial de la Comunidad GingaArgentina:http://comunidad.ginga.org.ar/, 2011.

[14]http://www.dtv.org.br/download/es-es/ABNTNBR15606-4_2010Esp_2010.pdf

[**15]**http://wiki.solar.org.ar/publico:ginga

[16]http://www.pleiad.cl/_media/research/adi/tvd-desarrollo.pdf

[17]http://aat.inictel-uni.edu.pe/files/SET_TOP_BOX%28Informe_de_Avance1%29.pdf

HOJA DE LEGALIZACIÓN DE FIRMAS

ELABORADO POR

Sr. Ángel Leonardo Quingaluisa Quispe

Sr. Jonathan Alberto Torres Beltrán

DIRECTOR DE LA CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

Ing. Mauricio Campaña

Sangolquí, 28 de Octubre del 2011