Universidad de las Ciencias Informáticas

Facultad # 9

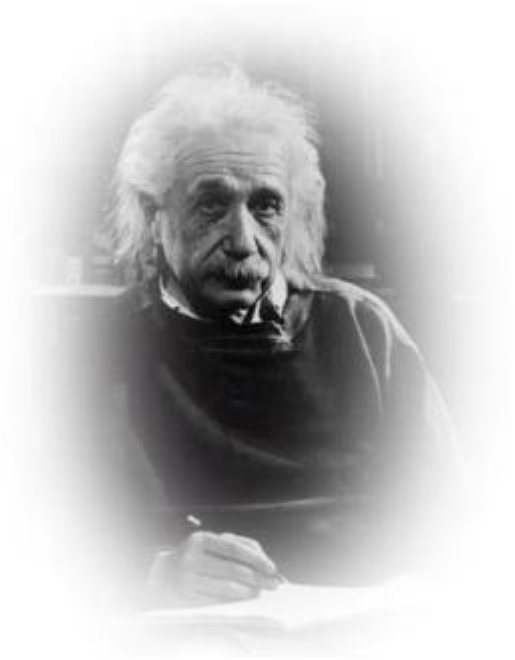
TRABAJO DE DIPLOMA PARA OPTAR POR EL TITULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Título: Plataforma de Transmisión Abierta para Radio y Televisión. Subsistemas Programación y Gestión de Medias.

Autor: Rolando Morejón Mayedo

Tutor: Ing. Eridniel Suárez Contreras

Curso 2008-2009



Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

A mis padres, por haber sido mi inspiración y la razón de todos mis esfuerzos, por lograr este sueño, que también es de ellos.

A mi novia a quien amo con todo mi corazón, por entenderme y amarme como lo ha hecho.

AGRADECIMIENTOS

"Yo sostendría que las gracias son la forma más alta de pensamiento y que el agradecimiento supone la felicidad una duplicada cuando media la sorpresa"

G.K. Chesterton**.**

Les agradezco eternamente a mis padres que durante estos cinco años y toda la vida me han brindado su apoyo, cariño y amor para que un día como hoy pueda estar en este lugar, nada de lo que he logrado hasta hoy hubiera sido posible sin ellos, gracias por existir y por ser parte de mi vida, los quiero.

A mi hermana que a pesar de pasarnos la vida discutiendo la quiero con locura y aunque no lo sepa me inspira a ser mejor cada día.

A mis compañeros de grupo por tener que lidiar conmigo tanto tiempo y enseñarme a ser mejor persona.

A mis amigos, Rocny, Miguel, Ronald, Genry, Barbaro, Renier, Heikel por los buenos y malos momentos que pasamos juntos, por haber entrado a mi vida para quedarse, por hacer que estos cinco años no fueran tan largos.

A mis amigas del alma, Denise, Ailen y Yordania a las cuales les tengo mucho aprecio y cariño.

A mi tutor Ing. Eridniel Suárez por su paciencia, atención y ayuda en el desarrollo de este trabajo.

A mi ángel de la guarda Yandy, que sin él no hubiera sido posible este trabajo, por darme las fuerzas para emprender este trabajo y la confianza de que "si se puede", gracias hermano de corazón te agradezco todo lo que hiciste por mí.

Le agradezco especialmente y le dedico esta tesis a mi compañera, amiga y novia Anabel por ser mi sostén en los momentos difíciles, por tener siempre su apoyo y consideración, por ser la persona que es, te quiero por la persona que eres, nunca cambies!!!.

A la familia de mi novia que también puso su granito de arena para que siguiera adelante.

DECLARACIÓN DE AUTORÍA

Declaro ser Rolando Morejón Mayedo el único autor del presente Trabajo de Diploma, autorizo a la facultad 9 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio y reconozco a la misma los derechos patrimoniales de este trabajo, con carácter exclusivo.

Para que así conste firmo la presente a los 24 días del mes de junio del año 2009.

Rolando Morejón Mayedo	Ing. Eridniel Suárez Contreras

RESUMEN

La Universidad de las Ciencias Informáticas cuenta con una Dirección de Televisión Universitaria (DTU) que está conformada por distintos departamentos. El departamento destinado para la Programación posee la dificultad de no contar con una programación estándar para la proyección de materiales audiovisuales, ni tampoco conocer cuáles son los materiales que se han proyectado en ocasiones anteriores. Otro de los departamentos que presenta problemas es el de Gestión de Medias, donde los materiales que le son proporcionados se reciben en diferentes formatos. Es por ello que se hace necesario el desarrollo de una aplicación que permita de manera centralizada la administración de la programación de los canales de Televisión, Radio y Web, así como la gestión de las medias en el sistema Plataforma de Transmisión Abierta para Radio y Televisión, basada en herramientas libres.

Para el desarrollo del trabajo primeramente se realizó un análisis profundo de las distintas técnicas de programación, metodologías, *frameworks* y plataformas que las soportan. Seguido a esto se describieron las principales funcionalidades de los subsistemas de Programación y Gestión de Medias, incluyendo los distintos módulos que los conforman, así como la estructura que presentan los mismos. Además se detalló el estilo de programación empleado en la implementación de los subsistemas y se realizaron las pruebas unitarias a los módulos que conforman los subsistemas con el fin de validar los resultados obtenidos.

PALABRAS CLAVES: programación estándar, materiales audiovisuales, formatos, herramientas libres, *frameworks,* subsistemas, pruebas unitarias.

ABSTRACT

The University of Informatics Sciences has a Direction of University Television known as DTU (Dirección de Televisión Universitaria) formed by different departments. The department destined for programming has the difficulty of not having a standard programming for playing audiovisual materials, nor for knowing which are the materials that have been previously played. Another department with problems is Media Management, in which the materials that are provided are received in different formats. That is why it is necessary to develop an application that enables the management of programs to be played in Television, Radio and Web, in a centralized manner, as well as the media management in the system Open Transmision Platform for Radio and Television (Plataforma de Transmisión Abierta para Radio y Televisión), based on free software tools.

For the development of such application a deep analysis on the different programming techniques, methodology, frameworks and platforms supporting it was firstly made. Following that, the main functionalities of the Programming and Media Manager subsystems was made, and it included the different modules forming them as well as the structure they have. Besides, the programming style used for the implementation of the subsystems. Finally, the unity tests was carried out on the modules forming the subsystem with the goal of validating the obtained results.

KEYWORD: standard programming, audiovisual materials, formats, free tools, *frameworks,* subsystems, unit tests.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los orígenes de la historia de la televisión comprenden una gran etapa que se expande desde finales del siglo XIX hasta 1935 aproximadamente. Su surgimiento constituyó uno de los avances tecnológicos más importantes de ese siglo, paulatinamente fue alcanzando su desarrollo, al principio fue una mezcla de audio y un conjunto de imágenes en movimiento con ausencia de otros colores que no fueran el blanco y el negro, pero luego con el paso del tiempo y cumplida la etapa de nacimiento y consolidación, llegó la época del color y tras ella la internacionalización del medio y de sus contenidos (1).

Las primeras emisiones públicas de televisión las efectuó la *British Broadcasting Corporation* (BBC) en Inglaterra en 1927 y la *Columbia Broadcasting System* (CBS) y *National Broadcasting Company* (NBC) en Estados Unidos en 1930. En ambos casos se utilizaron sistemas mecánicos y los programas no se emitían con un horario regular. Las primeras emisiones que contaron con una programación, se iniciaron en Inglaterra en 1936, las cuales fueron interrumpidas posteriormente debido a la II Guerra Mundial, la cual frenó todo el desarrollo que estaba teniendo la televisión en el mundo, pero estas transmisiones se reiniciaron cuando se culminó dicha guerra, la cual empezó a tomar el protagonismo que había alcanzado en años anteriores. A partir de la década de 1970, con la aparición de la televisión en color, los televisores experimentaron un crecimiento enorme (2). Este efecto es tal que en la actualidad la televisión es uno de los pasatiempos más populares en todo el mundo.

Cuba no quedó fuera de los avances tecnológicos que se desarrollaban en el mundo, por ello el 24 de octubre de 1950 la Unión Radio Televisión transmite la primera señal de televisión comercial por el canal 4, siendo de este modo, uno de los primeros países en América en proyectar por primera vez la televisión, después de México y Argentina. En 1957 cuando alrededor de 50 países tenían televisión, en Cuba ya se contaba con 7 años de experiencia (2).

Hoy en día no se puede hablar de educación, ni de cultura, ni de desarrollo, si no se integran eficientemente y en todas las esferas de la sociedad las Nuevas Tecnologías de la Información y Comunicaciones (TICs), es por esta razón que en los últimos años en Cuba, se ha promovido el uso masivo de las TICs, insertándolas en dichas esferas. Para darle cumplimiento a este propósito se insertaron televisores, computadoras, reproductores de video en cada uno de los sectores de la sociedad,

para incrementar el uso de las tecnologías. Por otro lado el gobierno cubano creó en septiembre del 2002 la Universidad de las Ciencias Informáticas (UCI), la cual estaría encaminada a promover la producción del *software* en el país.

Un ejemplo del uso de las TICs en la UCI es la creación de canales de televisión propios para apoyar el proceso de enseñanza-aprendizaje en la misma, mediante el empleo de teleclases de las asignaturas que se imparten en la universidad. Los canales de televisión no solo son empleados para fines educativos, sino también para el entretenimiento del personal en la UCI, pues existe un espacio para proyectar variados materiales audiovisuales como: películas, series, documentales, videos y otros.

En la actualidad la universidad cuenta con un portal Web que ser encarga de hospedar todo el contenido multimedia, sin embargo dicho portal está respaldado por herramientas propietarias, las cuales constituyen un gran inhibidor para la realización de las tareas en el mismo, es por eso que en el país se está tratando de migrar a *software* libre. Existen innumerables ventajas que el *software* no propietario brinda, una de las más conocidas, es del alto rango que está obteniendo en la rama de los servidores, es por ello que al desarrollar la plataforma empleando herramientas libres, esta va a ser más potente, mucho más estable y por consiguiente segura, otra de sus ventajas es que cuenta con una interfaz gráfica y muchas aplicaciones para realizar diversas tareas, desde procesamiento de texto hasta montajes de servidores de red, y como estas otras más, por consiguiente si se realiza su migración se podrán realizar numerosos aportes a dicha plataforma para su completamiento y eficiencia en aras de un mejor servicio y funcionamiento.

Para el control de todos los procesos de transmisión de audio y video, la universidad cuenta con la Dirección de Televisión Universitaria (DTU), formada por diferentes departamentos. Dentro de estos se encuentran los encargados de la Programación y el de Gestión de Multimedia, los cuales realizan sus tareas correspondientes de una manera informativa y manual. En el departamento de Programación no se cuenta con una programación estándar para la proyección de materiales audiovisuales, ni se conoce que es lo que se proyecta en ocasiones anteriores y en el de Gestión de Multimedia existen también grandes problemas, pues los materiales audiovisuales que le son proporcionados son recibidos en diferentes

formatos como: avi[1],mpeg-1[2], mpeg-2[3] y otros, para la proyección de dichos materiales es necesario transmitirlos o publicarlos en un formato único, la tarea de realizar su conversión se realiza de manera manual y tediosa por los trabajadores de la DTU.

Conociendo la situación problémica que se plantea, se identificó el siguiente **problema científico:** La ausencia de una aplicación que permita de manera centralizada la administración de la programación de los canales de Televisión y Radio, así como la gestión de las medias en el sistema Plataforma de Transmisión Abierta para Radio y Televisión.

Con la implementación de estas aplicaciones se pretende darle solución a todos los problemas existentes en el Departamento de Televisión Universitaria, mediante funcionalidades que faciliten la programación y la gestión de las multimedia. Dichas aplicaciones pasarán a formar parte de la Plataforma de Transmisión Abierta para Radio y Televisión la cual tiene como objetivo la integración de la Web y las tecnologías de la televisión permitiendo la vinculación de programas de televisión en la Web.

Mediante la fusión de los medios más importantes de comunicación en una aplicación, la riqueza de información disponible en la Web y la de información audiovisual que se está emitiendo en la televisión se ponen a disposición de usuarios en la Web y de telespectadores, logrando así una rica experiencia con la integración de ambos medios de comunicación.

Para darle solución al problema científico mencionado se planteó como **objetivo general:** Desarrollar un *software* que permita la automatización de los procesos de Programación y Gestión de Medias para el sistema Plataforma de Transmisión Abierta para Radio y Televisión.

[1] **AVI,** (*Audio Video Interleave*), en español intercalado de audio y video, es un formato de archivo contenedor de audio y vídeo.

[2] **MPEG – 1**, nombre de un grupo de estándares de codificación de audio y vídeo normalizados por el grupo *MPEG* (*Moving Pictures Experts Group*).

[3] **MPEG – 2,** similar a *MPEG-1*, pero también proporciona soporte para vídeo entrelazado (formato utilizado por las televisiones).

Se traza como **idea a defender:** Si se automatizan los subsistemas de Programación y Gestión de Medias basados en *software* libre, se logrará una mejor organización, calidad de transmisión y almacenamiento de las medias en el sistema Plataforma de Transmisión Abierta para la Radio y la Televisión.

El problema descrito anteriormente genera como **objeto de estudio:** Diseño de la Plataforma de Transmisión Abierta para Radio y Televisión y como **campo de acción:** Automatización de los procesos de Programación y Gestión de Medias para el sistema Plataforma de Transmisión Abierta para Radio y Televisión.

La investigación científica se guió por el conjunto de **tareas** que se describen a continuación para darle cumplimiento a los objetivos tanto generales como específicos planteados:

1. Identificar aplicaciones o soluciones existentes que puedan contribuir a la implementación de los subsistemas.
2. Valorar tendencias y tecnologías actuales.
3. Describir los paradigmas y lenguajes de programación. Desarrollar el prototipo de interfaz de usuario.
4. Valorar el *framework* y el patrón arquitectónico. Integrar con otros componentes o partes del sistema.
5. Describir la metodología de desarrollo.
6. Caracterizar el lenguaje de modelado y las herramientas *CASE*.
7. Describir el servidor Web y el sistema operativo a utilizar.
8. Construir el Modelo de Implementación.
9. Implementar los subsistemas Web y de Transferencia.
10. Desarrollar y mejorar los algoritmos para el procesamiento de información.
11. Desarrollar los Casos de Prueba que certifiquen la veracidad de los algoritmos empleados.
12. Desarrollar los manuales de usuario y de instalación de los subsistemas.

El presente trabajo estará estructurado de la siguiente manera:

Capítulo 1: Fundamentación Teórica

En este capítulo se expone un análisis profundo de las distintas técnicas de programación, metodologías, *frameworks*, así como de las plataformas que las soportan para el desarrollo de la plataforma de Transmisión Abierta para Radio y Televisión.

Capítulo 2 Descripción de la Propuesta.

En este capítulo se realiza una descripción del funcionamiento de los subsistemas de Programación y Gestión de Medias, así como la estructura que presentan los mismos. Además se detalla el estilo de programación que se emplea en la implementación de dichos subsistemas.

Capítulo 3: Validación de la solución propuesta.

En este capítulo se brinda una descripción de las pruebas unitarias atendiendo a su objetivo, alcance y detalles de las mismas, así como las herramientas empleadas para realizarlas. Además se evalúan los resultados obtenidos después de la ejecución de las pruebas.

Capítulo I

Fundamentación Teórica

1.1. INTRODUCCIÓN

En el presente capítulo se realiza un análisis profundo de las distintas técnicas de programación, metodologías, *frameworks*, así como las plataformas que las soportan para lograr un mejor desarrollo de la plataforma de Transmisión Abierta para Radio y Televisión.

1.2. TENDENCIAS Y TECNOLOGÍAS ACTUALES

1.2.1. Aplicaciones Web

Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican.

El *World Wide Web Consortium*[4] (W3C) define el término "Web" como: "el universo de información accesible a través de la red". Una aplicación Web es un sistema que permite a un usuario final acceder a una parcela de información contenida en el universo al que hace referencia la anterior definición de W3C. Las aplicaciones Web están basadas en el paradigma "cliente/servidor". Este paradigma consiste en un servidor que sabe como proporcionar un servicio y un cliente que desea acceder al servicio (3).

El incremento del uso de Internet ha aportado ventajas sustanciales en el desarrollo de las aplicaciones Web, los usuarios para poder utilizarlas sólo tienen que acceder a un servidor Web a través de Internet o Intranet. Una característica importante es la facilidad de actualizar y mantener aplicaciones Web sin distribuir o instalar *software* en miles de clientes potenciales. El acceso y manejo de la información en forma de aplicación Web asegura una manera rápida, sencilla y concisa de realizarse, donde el usuario no necesariamente debe conocer de procedimientos ni protocolos para ello.

Es vital destacar que no siempre fue así, en los primeros años de inicio de la computación "cliente-servidor", cada aplicación tenía su propio programa cliente y su interfaz de usuario, estos tenían que ser

[4] **W3C,** (*Consorcio World Wide Web*), consorcio internacional que produce estándares para la *World Wide Web*. Está dirigida por Tim Berners-Lee, creador original de URL (*Uniform Resource Locator*), HTTP (*HyperText Transfer Protocol*) y HTML (Lenguaje de Marcado de HiperTexto) que son las principales tecnologías sobre las que se basa la Web.

instalados por separado en cada estación de trabajo de los usuarios. Una mejora en el servidor requería una mejora de los clientes instalados en cada una de las estaciones de trabajo, añadiendo un coste de soporte técnico y disminuyendo la eficiencia del personal.

Ventajas de las aplicaciones Web

La creciente popularidad de las aplicaciones Web se debe a sus múltiples ventajas, entre las cuales se pueden citar:

- ✓ **Multiplataforma:** Con un solo programa, un único ejecutable, las aplicaciones pueden ser utilizadas a través de múltiples plataformas, tanto de *hardware* como *software*.
- ✓ **Actualización instantánea:** Debido que todos los usuarios de la aplicación hacen uso de un sólo programa que radica en el servidor, los usuarios siempre utilizarán la versión más actualizada del sistema.
- ✓ **Suave curva de aprendizaje:** Los usuarios, como utilizan la aplicación a través de un navegador, hacen uso del sistema tal como si estuvieran navegando por Internet, por lo cual su acceso es más intuitivo.
- ✓ **Fácil de integrar con otros sistemas:** Debido a que se basa en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.
- ✓ **Acceso móvil:** El usuario puede acceder a la aplicación con la única restricción de que cuente con un acceso a la red privada de la organización o a Internet, dependiendo de las políticas de dicha organización; puede hacerlo desde una computadora de escritorio, una PC portátil o desde una agenda electrónica; desde su oficina, hogar u otra parte del mundo (4).

¿Qué se logra con las ventajas de las aplicaciones Web en un sistema como la Plataforma de Transmisión Abierta para Radio y Televisión?

Estas ventajas le proporcionan a la plataforma mecanismos de comunicación estándares entre los diferentes módulos dentro de ella, que interactúan entre sí para presentar información dinámica al usuario. Además para proporcionar interoperabilidad y extensibilidad entre estos módulos, para que al mismo tiempo sea posible su combinación para realizar operaciones complejas.

1.2.2. Tecnología Cliente-Servidor

Se hace necesario en el uso de aplicaciones Web tecnologías para el acceso y distribución de los servicios, tecnologías que permitan ejecutar y responder de forma segura y óptima las peticiones de los usuarios por muy complejas que estas puedan ser. Teniendo en cuenta los múltiples servicios que brindan las aplicaciones Web, no solo internamente sino también para el servicio público se recurre a la tecnología cliente-servidor.

La tecnología denominada cliente-servidor es utilizada por la mayoría de las aplicaciones que hoy coexisten en Internet. La misma permite a una aplicación de usuario (comúnmente llamado cliente) desde un ordenador, hacer una petición de algún servicio suministrado por otro ordenador dedicado (servidor) a esta función mediante una red ya sea local o externa. Estos servicios pueden ser peticiones de datos a una base de datos de información contenida en archivos o los archivos en sí mismos o peticiones de imprimir datos en una impresora asociada (5).

Un solo servidor puede atender a varios clientes y brindarles la información o pedidos requeridos por los mismos de manera rápida y simultánea, así se ahorra el problema de tener la información instalada y almacenada localmente.

Características principales de la tecnología cliente-servidor

La tecnología cliente-servidor refleja diversas características, entre las que cabe señalar más importantes:

- ✓ El Cliente y el Servidor pueden funcionar como una única entidad y también actuar como entidades separadas, haciendo actividades o tareas de manera independiente.
- ✓ Sus funciones pueden estar en plataformas apartadas o en la misma.
- ✓ Un servidor ofrece servicio a variados clientes de forma concurrente.
- ✓ Toda plataforma pudiera ser escalable absolutamente. Las modificaciones realizadas en las plataformas de los clientes o de los servidores, ya sean por modernización o por substitución tecnológica, se efectúan de manera clara para el usuario final (6).

Principales componentes

Los **Clientes** son los que interactúan con los usuarios, de forma gráfica usualmente. Se comunican con procesos auxiliares encargados de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad (7).

Los **Servidores** son los encargados de proporcionar un servicio al cliente y devolver los resultados. En determinados casos se hallan procesos subalternos encomendados a recibir solicitudes del cliente, verificar protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además manejan los interbloqueos, la recuperación ante fallas y otros aspectos afines. Por los elementos antes expuestos la plataforma computacional asociada a los servidores es más eficaz que la de los clientes. Igualmente deben manejar servicios como la administración de la red, la gestión de la entrada al sistema "*login*", auditorías, recuperación y contabilidad (8).

Para lograr que los clientes y servidores consigan comunicarse se requiere de una **infraestructura de comunicaciones**, la cual es la responsable de proporcionar los mecanismos básicos de direccionamiento y transporte. Los sistemas cliente-servidor actuales, en su mayoría, se centran en las redes locales, por lo cual emplean protocolos no orientados a conexión, esto implica que las aplicaciones deben hacer las verificaciones. La red debe presentar características ajustadas al desempeño, confiabilidad, transparencia y administración (8).

Tipos de servidores

Los sistemas cliente-servidor pueden ser de varios tipos, esto depende principalmente de las aplicaciones que el servidor instala a disposición de los clientes. Entre los que se pueden encontrar:

- ✓ **Servidores de Impresión**, mediante los cuales los usuarios comparten impresoras.
- ✓ **Servidores de Archivos**, con los que los clientes pueden compartir discos duros.
- ✓ **Servidores de Bases de Datos**, donde se puede encontrar una base de datos única.
- ✓ **Servidores de *Lotus Notes***, que permiten el trabajo paralelo de distintos clientes con datos semejantes, documentos o modelos.
- ✓ **Servidor de Aplicaciones:** Windows NT, Novell.

Los servidores Web igualmente se emplean en la tecnología cliente-servidor, aunque incorporan aspectos nuevos y propios a la misma. Por esta razón y teniendo en cuenta que la Plataforma de Transmisión Abierta para Radio y Televisión puede ser accedida simultáneamente por un grupo de usuarios solicitando un servicio o recurso ofrecido por un servidor central, se justifica la utilización de esta tecnología para el desarrollo del sistema.

Con su uso se produce un aumento de la interactividad y se facilita el mantenimiento de las aplicaciones. Además que implicaría un mejor manejo de errores e implementación de estrategias para garantizar la integridad de los datos, si se quisiera modificar la infraestructura del sistema podría resultar sencillo pues integrar más ordenadores o sistemas externos no constituiría un problema.

1.2.3. Ajax

Asociada a la tecnología Web y cliente-servidor aparece el modelo de presentación de datos *AJAX* (*JavaScript* Asíncrono y *XML*) que ofrece comodidad visual para el usuario en términos de interacción con los datos.

Como primera definición del término *AJAX*, se tiene la expuesta por James en su artículo: "*AJAX* no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes" (9). Se puso en circulación por primera vez con el término *AJAX* en el artículo "*AJAX*: *A New Approach to Web Applications*" publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese instante, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación Web que estaba surgiendo.

Las tecnologías que constituyen *AJAX* son (10):

- ✓ *HTML* o *XHTML* y Hojas de Estilo en Cascada (*CSS*) para crear presentaciones basadas en estándares.
- ✓ *Document Object Model* (*DOM*) y *JavaScript* para la interacción y manipulación dinámica de la presentación.
- ✓ *XML*y *XSLT* para el intercambio y manejo de información.
- ✓ *XMLHttpRequest* para el intercambio asíncrono de información.

- ✓ *JavaScript* para la unión de las demás tecnologías.

Las aplicaciones *AJAX* emplean navegadores Web que toleran las tecnologías antes mencionadas. Entre los que se encuentran *Mozilla Firefox*, *Internet Explorer*, *Opera*, *Konqueror* y *Safari*, las cuales eliminan la sobrecarga de páginas por razón de la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de *AJAX* perfecciona la respuesta de la aplicación, pues el usuario jamás se encuentra con una ventana del navegador disponible en espera de la respuesta del servidor. Para justificar el uso de este modelo se exponen las diferencias que existen entre las aplicaciones que promueven este nuevo elemento y las aplicaciones Web tradicionales.

AJAX: Diferencias con las aplicaciones Web tradicionales

En las habituales aplicaciones Web los usuarios interactúan por medio de formularios, que al ser enviados, realizan una petición al servidor Web. El servidor se comporta de acuerdo con lo enviado en el formulario y responde a esto con el envío de una nueva página Web. Todo esto derrocha un gran ancho de banda, pues gran parte del *HTML* enviado en la segunda página Web, ya estaba presente en la primera. Al mismo tiempo, de esta forma no es viable crear aplicaciones con un grado de interacción relacionado al de las habituales o las escritorio (9).

En aplicaciones *AJAX* se consiguen enviar peticiones al servidor Web con el propósito de conseguir solamente la información que se necesita, utilizando para ello algún lenguaje para servicios Web apoyados en *XML* y empleando *JavaScript* en el cliente para procesar la respuesta del servidor Web. Esto causa una mayor interacción gracias a la disminución de información intercambiada entre servidor y cliente, ya que una parte del proceso de la información lo realiza el mismo cliente por medio de *JavaScript*, librando al servidor de esa labor. La compensación es que al descargar inicialmente la página es más lento el proceso, ya que se tiene que descargar todo el código *JavaScript*.

Actualmente existen diferentes *frameworks* para el uso de *AJAX*, cada uno tiene sus peculiaridades. Al realizar una selección es importante tener en cuenta el sistema que se desarrollará, la complejidad del mismo y la habilidad de los desarrolladores. De acuerdo a sus características se decidió escoger el

framework ExtJS por todas la funcionalidades que le aportan a los subsistemas de Programación y Gestión de Medias que se describen a continuación.

ExtJS es un nuevo *framework*, basado originalmente en *YUI*[5] pero actualmente es independiente del *framework* que se utilice (incluso puede usarse sin *frameworks*). Ofrece una gran cantidad de componentes para crear interfaces de usuarios complejas (11). No es más que una biblioteca de *JavaScript* que se utiliza para el desarrollo de aplicaciones Web interactivas usando tecnologías como *AJAX* y *DOM*.

Esta librería incluye componentes de interfaz de usuario de alto *performance* y personalizables como son los cuadros de diálogo, menús, tablas editables, paneles, pestañas y todo lo necesario para construir atractivos desarrollos al estilo de Web 2.0. Además incluye modelo de componentes extensibles y ayuda fácil de usar con todas las especificaciones y ejemplos de casi todos los componentes para una correcta utilización de los mismos. Soporta los navegadores:

- ✓ Internet Explorer 6+.
- ✓ FireFox 1.5+ (PC, Mac).
- ✓ Safari 2+.
- ✓ Opera 9+ (PC, Mac).

Luego del análisis realizado del modelo *AJAX y* el *framework ExtJS* y dadas las ventajas expuestas anteriormente, se propone su uso para la implementación de los subsistemas de Programación y Gestión de Medias de la Plataforma de Transmisión Abierta para la Radio y la Televisión con el objetivo de obtener una simulación de aplicación de escritorio que contenga interactividad entre los distintos elementos presentados, mediante el uso de efectos visuales, constituyendo de esta forma una consumación de los requerimientos no funcionales establecidos por el cliente para el sistema.

El desarrollo de aplicaciones ya sean Web o de escritorio requiere de un grupo de estándares o paradigmas de programación para que se pueda obtener un producto de fácil soporte por otros

[5] **YUI**, conjunto de utilidades y controles, escrito en *JavaScript*, para la construcción de aplicaciones Web interactivas, usando técnicas como *DOM*, *DHTML* y *AJAX*.

desarrolladores al plasmarse una perspectiva común para la construcción del *software*, entendiéndose de esta forma, el conjunto de normas a seguir en la programación de los componentes que conforman los subsistemas.

1.3. PARADIGMAS DE LA PROGRAMACIÓN

La programación de manera general son marcos de referencia que asignan normas sobre cómo se deben hacer las cosas, muestran qué es permitido dentro del paradigma y qué está fuera de lugar. Conjuntamente pueden ser estimados como patrones de pensamiento para resolver posibles problemas que se presenten. Un paradigma distinto involucra nuevas reglas, elementos, límites y formas de pensar, es decir, implica un cambio (11).

Los paradigmas de programación en específico constituyen una visión particular o filosófica para la construcción del *software*, pues se definen como un conjunto de reglas a seguir que facilitan el trabajo en la programación. Establecen recomendaciones de expertos programadores, que tras años de labor, han acumulado una gran experiencia.

En el mundo actual existen diferentes tipos de paradigmas de programación, no es mejor uno que otro sino que cada uno tiene ventajas y desventajas. Utilizando como basamento lo antes expuesto se dará una pequeña descripción de los que tienen mayor importancia para la construcción de la Plataforma de Transmisión Abierta para Radio y Televisión.

1.3.1. Tipos de Paradigmas de Programación

Paradigma Imperativo

Son aquellos que facilitan las automatizaciones por mediación de cambios de estado, concibiendo como estado la condición de una memoria de almacenamiento. Los lenguajes organizados en bloques representan a los contornos anidados, o sea, los bloques pueden estar alojados dentro de otros y dominar sus propias variables. La *RAM*[6] constituye una pila, la cual hace referencia al bloque se encuentra activo

[6] **RAM**, (*Random Access Memory*), es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados, el área de trabajo para la mayor parte del *software* de un computador.

en ese instante situado en la parte superior (12). Calificado como el más usual, está representado por el *C++* o por *BASIC*, por citar algunos.

La programación en este paradigma consiste en determinar qué datos son requeridos para el cálculo, asociar a esos datos unas direcciones de memoria y efectuar paso a paso una secuencia de transformaciones en los datos almacenados, de forma que el estado final represente el resultado correcto. Solamente es empleado en la programación a bajo nivel o lenguaje de máquina, no se destina para la construcción del *software* que se presenta.

Paradigma Funcional

El paradigma funcional está basado en el modelo matemático de composición funcional. En este modelo, el resultado de un cálculo es la entrada del siguiente y así sucesivamente hasta que una composición produce el valor deseado. Como indica su nombre maniobran simplemente a través de funciones. Cada función retorna un único valor dada una lista de parámetros. No se admiten asignaciones generales, es decir, efectos colaterales. La programación funcional provee la capacidad para que un programa se cambie así mismo, o sea, que pueda ejercitarse (12). Representado por la familia de lenguajes *LISP*[7], en específico *Scheme*[8].

Paradigma lógico

Esta programación se basa en un subconjunto del cálculo de predicados, incluyendo explicaciones precisas en forma de las llamadas cláusulas de *Horn*[9]. Un sistema de cláusulas de ese tipo admite un método específicamente mecánico de demostración llamado resolución. Este tipo de paradigma puede derivar nuevos hechos a partir de otros conocidos (12). Ejemplo el *PROLOG.*

Es difícil de comprender y dominar. Su implementación puede resultar compleja cuando los hechos y reglas no son claramente visibles. El uso de este paradigma resulta realmente viable para aplicaciones

[7] **LISP,** lenguaje funcional muy útil en el campo de la Inteligencia Artificial.

[8] **Scheme,** lenguaje de programación desarrollado por Guy L. Steele y Gerald Jay Sussman en la década de los setenta.

[9] **HORN,** es una regla de inferencia lógica con una serie de premisas (cero, una o más) y un único consecuente.

que modelen la inteligencia artificial, bases de datos o procesos. Como no se pretende tratar con problemas de inteligencia artificial en la Plataforma de Transmisión Abierta para Radio y Televisión no se tiene en cuenta el uso de este paradigma.

Paradigma Orientado a Objetos (POO)

Es un paradigma que emplea clases, objetos, métodos, mensajes, atributos para diseñar aplicaciones y programas de computadora. Se basa en varias técnicas de programación, incluyendo herencia, modularidad, polimorfismo, abstracción y encapsulamiento, las cuales aseguran la re-usabilidad de código (13). Lenguaje orientado a objetos completamente, ejemplo de este es el *C*++ (11). Por las ventajas que ofrece para la programación estructurada y el trabajo en equipo es que se escoge como paradigma a usar en la construcción del sistema.

Ventajas de la Programación Orientada a Objetos:

- ✓ Es intuitiva, describe un problema en términos similares a los que utiliza la mente humana.
- ✓ Permite construir soluciones más seguras y con un mantenimiento más sencillo.
- ✓ Escalabilidad de las aplicaciones.
- ✓ Fomenta la reutilización y extensión del código siendo de gran ayuda para los implementadores de sistemas *software*.
- ✓ Permite la creación de programas más complejos y agiliza su desarrollo.
- ✓ Proporciona la creación de programas visuales y de prototipos.
- ✓ Facilita el mantenimiento de los sistemas y el trabajo en equipo.

Dentro de cada paradigma de programación existen diferentes lenguajes de programación los cuales se diferencian en la forma de estructurar las funciones pero de manera general todos siguen el mismo patrón del paradigma asociado. Los lenguajes se dividen en lenguajes de programación del lado cliente y lenguaje de programación del lado servidor los cuales se describen a continuación.

1.4. LENGUAJES DE PROGRAMACIÓN.

En la actualidad existen diferentes lenguajes de programación para desarrollar en la Web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Desde los inicios de Internet, fueron

surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. A medida que pasó el tiempo, las tecnologías fueron desarrollándose y aparecieron nuevos problemas a dar solución. Estos problemas conllevaron al desarrollo de lenguajes de programación para páginas Web dinámicas que permitieran interactuar con los usuarios.

1.4.1. Lenguajes de programación del lado cliente.

Los lenguajes de programación del lado cliente, en la mayoría de los casos, se utilizan en la construcción de las páginas Web clientes que son interpretadas por el navegador. Para la implementación de los subsistemas de Programación y Gestión de Medias se tuvieron en cuenta algunos de ellos, los cuales son referenciados a continuación.

HTML

HTML (Lenguaje de Marcas de Hipertexto) es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. También indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto, las cuales conectan diferentes documentos ya sea en la computadora u otros recursos de Internet (14). Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el contenido con objetos tales como imágenes.

Las principales características de *HTML*, que lo hacen de la preferencia del autor son:

- ✓ Permite crear lenguajes de codificación descriptivos.
- ✓ Define una estructura de documentos jerárquica, con elementos y componentes interconectados.
- ✓ Proporciona una especificación formal completa del documento.
- ✓ No tiene un conjunto implícito de convenciones de señalización.
- ✓ Soporta, por tanto, un conjunto flexible de juegos de etiquetas.
- ✓ Los documentos generados por él son legibles (15).

Este lenguaje se empleará para facilitar la construcción de los subsistemas, pues permite aplicar distintos estilos al texto y definir una estructura lógica para el documento *HTML*. Además le indica al navegador

donde colocar cada contenido, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. Después de tener definido el lenguaje que va a estructurar el contenido de las páginas del sistema se hace necesario escoger otro para la validación de los formularios.

JavaScript

JavaScript es un lenguaje de programación interpretado, lo que significa que, no es necesario compilar los programas para ejecutarlos. Explicándolo de otra manera, los programas que emplean este lenguaje se pueden comprobar directamente en cualquier navegador sin necesidad de realizar otros procesos por medio. Se emplea principalmente para la creación de páginas Web dinámicas.

Es un lenguaje orientado a eventos, ya que se pueden desarrollar *scripts*[10] que ejecuten acciones en respuesta a estos. Conjuntamente está orientado a objetos, por lo cual contiene los elementos necesarios para que los *scripts* puedan acceder a la información de una página, así como proceder sobre la interfaz del navegador (16).

Al ser un lenguaje de programación que se ejecuta en el lado cliente, posibilita la reducción de carga del servidor, siendo esta una característica muy importante para la aplicación que se quiere desarrollar, al poder implementarse distintas funcionalidades con recursos multimedia, los cuales resultan “pesados” a la hora de su uso. Además la vinculación estrecha que existe entre este lenguaje y *AJAX* hace necesaria su utilización.

1.4.2. Lenguajes de programación del lado del servidor.

Los lenguajes del lado servidor son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se remiten al cliente en un formato claro para él. Uno de los más usados actualmente es *PHP*, el cual fue escogido anteriormente por los analistas del sistema que se implementa; a continuación se presenta la validación de su uso por parte de los desarrolladores.

[10] **Scripts**, grupo de lenguajes de programación que son típicamente interpretados y pueden ser tipeados directamente desde el teclado.

PHP

PHP (*Hypertext PreProcessor*), es un lenguaje del lado del servidor, esto indica que funciona en un servidor remoto que procesa las páginas Web antes de que sean enviadas al navegador del usuario. Este lenguaje fue creado para el desarrollo de páginas Web dinámicas. Se puede incluir dentro del código *HTML* con facilidad, además permite una serie de funcionalidades extraordinarias, por las cuales se ha convertido en el favorito de millones de programadores en todo el mundo (17).

Entre sus características fundamentales se encuentran:

- ✓ **Gratuito**. Al tratarse de *software* libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- ✓ **Gran popularidad**. Existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código y que en muchos casos estarán encantados de ayudar cuando alguna persona se enfrente a algún problema.
- ✓ **Enorme eficiencia**. Con escaso mantenimiento y un servidor Web gratuito como Apache, puede soportar sin problema millones de visitas.
- ✓ **Sencilla integración con múltiples bases de datos**. Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos, puede conectarse a *PostgreSQL*[11], *Oracle*[12] o cualquier otra base de datos compatible con *ODBC*[13].
- ✓ **Versatilidad**. *PHP* puede usarse con la mayoría de sistemas operativos, ya sea basado en *UNIX Linux*[14], *Solaris*, *FreeBSD* y *Windows*.
- ✓ **Gran número de funciones predefinidas**. A diferencia de otros lenguajes de programación, *PHP* fue diseñado especialmente para el desarrollo de páginas Web dinámicas. Por ello, está dotado de

[11] **PostgreSQL,** sistema de gestión de base de datos relacional orientada a objetos de *software* libre.

[12] **Oracle,** sistema de gestión de base de datos relacional, desarrollado por *Oracle Corporation*. Se considera a *Oracle* como uno de los sistemas de bases de datos más completos.

[13] **ODBC**, (*Open Database Connectivity*), estándar de acceso a bases de datos desarrollado por *Microsoft Corporation*, el objetivo de *ODBC* es hacer posible el acceso a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos.

[14] **Linux,** es el núcleo o *Kernel* del Sistema Operativo Libre denominado *GNU/Linux.*

un gran número de funciones que simplifican enormemente tareas habituales como descargar documentos, enviar correos, trabajar con *cookies*, sesiones y otros (17).

Por lo anteriormente expuesto se hace indiscutible *PHP* como la solución ideal para la implementación de los subsistemas de Gestión de Medias y Programación. Es un lenguaje multiplataforma aunque ofrece más prestaciones en los entornos *UNIX*[15] pero es perfectamente compatible con *Windows y Linux.* El hecho de ser libre y poder utilizarlo sin necesidad de disponer de una licencia aumenta sus ventajas debido a la situación que mantiene Cuba con respecto al bloqueo y la decisión de migrar los sistemas informáticos existentes a entornos libres.

1.5. PATRÓN ARQUITECTÓNICO: MODELO-VISTA-CONTROLADOR

El *Model-View-Controller* (Modelo-Vista-Controlador, en adelante MVC) se introdujo inicialmente en la comunidad de desarrolladores de *Smalltalk-80*. El MVC divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada. Para ello, emplea las siguientes abstracciones:

- ✓ **Modelo** (*Model*): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- ✓ **Vista** *(View):* Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- ✓ **Controlador** *(Controller):* Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas y otros. Los eventos son traducidos a solicitudes de servicio ("*service requests*") para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores (18).

Este patrón es muy popular, pues ha sido portado a una gran cantidad de entornos y *frameworks*, entre los que se encuentran *WinForms*, *ASP.Net*, *Symfony*. Herramientas de programación visual como *Visual Basic*, *Visual Studio.Net* y otras emplean también alguna variante de este esquema. El MVC es utilizado en múltiples plataformas y lenguajes. Algunos de sus principales beneficios son:

- ✓ Menor acoplamiento.

[15] **Unix,** (registrado oficialmente como **UNIX**®), sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969.

- ✓ Mayor cohesión.
- ✓ Las vistas proveen mayor flexibilidad y agilidad.
- ✓ Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente.
- ✓ Facilita el mantenimiento.
- ✓ Mayor escalabilidad.

Dadas las ventajas que ofrece el MVC, las cuales se pueden distinguir esencialmente en la creación de instancias exclusivas de las vistas, la alta cohesión y la escalabilidad, además de ser menos costoso y factible, es que se escoge este patrón para el desarrollo de los subsistemas de Gestión de Medias y Programación.

1.6. FRAMEWORK

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, le proporciona la estructura al código fuente, forzando al desarrollador a crear el código más legible y fácil de mantener. Al mismo tiempo, facilita la programación de aplicaciones, pues encapsula operaciones complejas en instrucciones sencillas (19).

1.6.1. Symfony

Symfony es un completo *framework*, desarrollado por Fabien Potencier, y diseñado para perfeccionar el desarrollo de aplicaciones Web que separa la capa lógica de negocio, la capa lógica de servidor y la capa de presentación de una aplicación. Facilita diversas herramientas y clases enfocadas en reducir el tiempo de perfeccionamiento de un sistema complejo. Este *framework* informatiza las tareas más comunes, brindando al desarrollador la posibilidad de dedicarse por completo a los aspectos específicos de cada aplicación. Está desarrollado completamente con *PHP* 5[16], última versión de *PHP* que incorpora la programación orientada a objetos. Ha sido probado en numerosos proyectos reales y se utiliza en entornos Web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como *MySQL*, *PostgreSQL*, *Oracle* y *SQL Server* de *Microsoft*. Se puede ejecutar tanto en plataformas (*Unix*, *Linux*) como en plataformas *Windows*.

[16] **PHP 5**, una de las últimas versiones de *PHP* en la cual se ha incorporado la programación orientada a objetos.

Entre sus características principales se mencionan:

- ✓ Fácil de instalar y configurar en la mayoría de las plataformas.
- ✓ Independiente del sistema gestor de bases de datos, pues posee librerías encargadas de manejar la abstracción de datos.
- ✓ Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los más complejos.
- ✓ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de las mejores prácticas y patrones de diseño para la Web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de *phpDocumentor*[17] que permite un mantenimiento sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros (19).

Para el desarrollo de los subsistemas de Programación y Gestión de Medias se escoge como solución factible este *framework* puesto que permite la migración del sistema gestor de base de datos sin tener que hacer cambios en el código fuente de la aplicación, además se puede obtener un código comentado, facilitando de esta forma una guía para cuando se realice cualquier cambio en la aplicación.

Otra de las peculiaridades por lo cual se selecciona y que distingue al mismo es el empleo de *helpers*[18], con los cuales es más fácil el manejo de elementos multimedia sin tener que hacer grandes usos del lenguaje *JavaScript.* Igualmente se aprovecha la valiosa documentación (sobre todo en español) que contiene, pues incluye un libro que explica detalladamente los conceptos más importantes y documentos

[17] **phpDocumentor**, sistema para crear documentación de aplicaciones creadas con *PHP*, conocido también como *phpdoc* o *phpdocu.*

[18] **Helpers**, función *PHP* definida por *Symfony* que generan código *HTML* resultando más eficientes que escribir a mano ese mismo código *HTML*.

relacionados con la *API*[19] y la *Wiki*[20], que sirven de ayuda cuando no se tienen los conocimientos básicos sobre el tema.

1.7. METODOLOGÍAS DE DESARROLLO DE *SOFTWARE*

Las metodologías de desarrollo de *software* conforman un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de *software*. En ellas se van indicando paso a paso todas las actividades a realizar para lograr un buen producto informático, indicando además qué personas deben participar en el desarrollo de dichas actividades y el rol que deben ocupar cada una de ellas. Igualmente definen la información que se debe producir como resultado de una actividad y la necesaria para comenzarla (20).

El objetivo de un proceso de desarrollo es elevar la calidad del *software* mediante una mayor transparencia y control sobre el proceso. Todo proceso de desarrollo de *software* es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, los resultados finales serían impredecibles, además de que no hay forma de controlar lo que está sucediendo en el proyecto y se obtienen clientes insatisfechos con el resultado y desarrolladores aún mas insatisfechos. Una de las metodologías más generales y usadas que existen en la actualidad es el Proceso Unificado de Desarrollo (RUP).

1.7.1. Proceso Unificado de Desarrollo de *Software* (RUP)

El Proceso Unificado de Desarrollo es un marco de trabajo genérico que se especializa en una gran variedad de sistemas de *software*, en distintas áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyecto (21).

RUP es un proceso, que en su modelación define como principales elementos: Trabajadores que conforman el (quién); Actividades que representan el (cómo), Artefactos que son los productos tangibles del proyecto que concretan el (qué), y el Flujo de Actividades que precisan la secuencia de actividades

[19] **API**, (*Application Programming Interface*), conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

[20] **Wiki**, sitio Web cuyas páginas Web pueden ser editadas por múltiples voluntarios a través del navegador Web. Los usuarios pueden crear, modificar o borrar un mismo texto que comparten.

(cuándo). Además está compuesto por nueve flujos de trabajo y cuatro fases, cada una de ellas con un número variable de iteraciones como se muestra en la Figura 1.

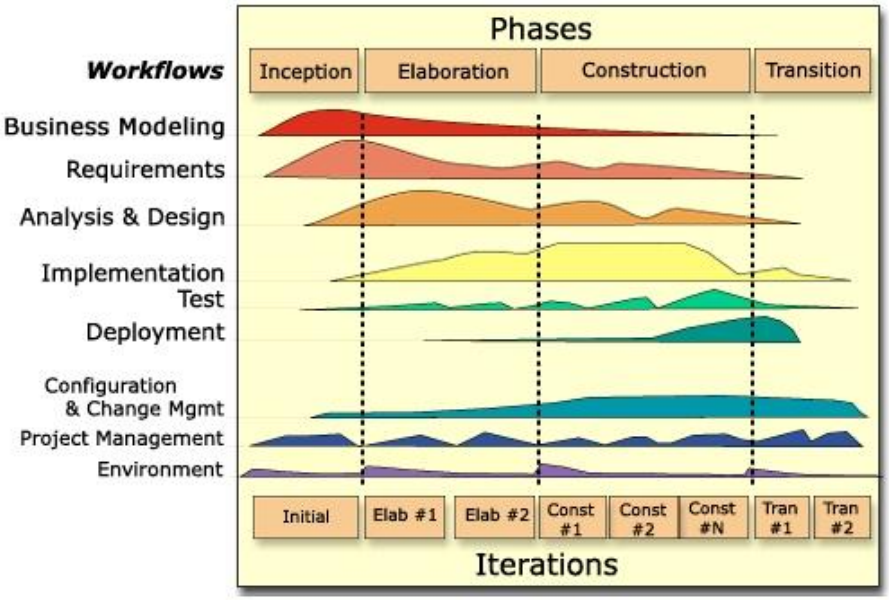


Figura 1. Fases e Iteraciones de la Metodología RUP.

Los elementos del RUP son:

- ✓ **Actividades**: Son los procesos que se llegan a determinar en cada iteración.
- ✓ **Trabajadores**: Son las personas o entes involucrados en cada proceso.
- ✓ **Artefactos:** Puede ser un documento, un modelo o un elemento del modelo (21).

Una peculiaridad de esta metodología es que, en cada ciclo de iteración, se hace estricto el empleo de artefactos, lo cual la convierte en una de las principales en lograr un grado de certificación en el desarrollo del *software*.

Se definió por el grupo de arquitectura, debido a la amplia documentación que se debía generarse y necesaria para darle soporte al sistema, la cantidad de trabajadores organizados en equipos de desarrollo y la complejidad del producto que se implementó, además de que esta metodología de desarrollo de *software* es la más adecuada para aplicarla como guía de trabajo.

1.8. LENGUAJE DE MODELADO

Es un lenguaje visual orientado al modelado de sistemas. Facilita un vocabulario controlado por reglas y símbolos, con el fin de que todos los empleados de un proyecto obvien las ambigüedades y la dispersión conceptual. Ayuda al usuario a comprender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir o gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, disminuyendo así, el costo y tiempo destinado a la construcción de los elementos que confeccionarán el modelo. Dispone de un repertorio de unidades (clases, acciones, objetos, estados y casos de uso). La gramática define reglas de combinación que forman otras unidades más complejas (diagramas y modelos) y tiene una determinada escala de abstracción (22).

Este lenguaje presenta numerosas ventajas, entre las que se destacan: con un número pequeño de elementos UML y sus reglas de combinación, es posible construir y comunicar estructuras y funcionalidades muy complejas. Además de que es un lenguaje consolidado, fácil de aprender y permite una comunicación fluida entre los diferentes actores del modelo. Mediante él es posible establecer una serie de requerimientos y estructuras necesarias para plasmar en un sistema de *software*, previo al proceso intensivo de escribir código.

Es un estándar de modelado utilizado en sistemas orientados a objetos. Es sin lugar a dudas una herramienta muy poderosa de la cual no se puede prescindir cuando se intenta crear un sistema complejo como es el desarrollo de los subsistemas de Programación y Gestión de Medias y se hace necesaria la comprensión general del mismo por parte de todas las personas involucradas.

1.9. HERRAMIENTAS CASE PARA LA MODELACIÓN

Las herramientas *CASE* permiten organizar y manejar la información de un proyecto informático, tornando más flexibles y comprensible los sistemas para los participantes de un proyecto y mejorando además la comunicación entre los mismos. Por esta razón se puede decir que representan una forma a través de la cual se pueden modelar los procesos de negocio de las empresas.

1.9.1. Rational Rose

Rational Rose es una herramienta *CASE* que soporta de forma completa la especificación del *UML* 1.X. Esta herramienta utiliza un proceso de desarrollo iterativo controlado que se lleva a cabo en una secuencia de iteraciones. Cada iteración empieza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se emplearán para dar comienzo a la iteración, como primer paso se identifican los riesgos y luego se prueba la aplicación para que éstos disminuyan (23).

Rose permite que diversas personas trabajen al mismo tiempo en el proceso iterativo controlado, por ello brinda la posibilidad de que cada desarrollador maniobre en un espacio de trabajo privado y tenga un control exclusivo relacionado a la propagación de los cambios en dicho espacio de trabajo. Además facilita mecanismos para realizar la denominada Ingeniería Inversa, esto quiere decir, que a partir del código de un programa, se puede conseguir información acerca de su diseño.

1.9.2. Visual Paradigm

Es una plataforma de modelado de sistema diseñado para arquitectos, desarrolladores, diseñadores, analistas de procesos de negocio y modeladores de datos, con el fin de acelerar todo el modelo del código para el complejo proceso de desplegar las aplicaciones empresariales a través de los mejores de su tipo y la galardonada tecnología de modelado visual que facilita la creación de los procesos de negocio, la visualización de *UML*, en la última notación *UML* 2.1, en 13 tipos de diagramas diferentes y los diagramas de Entidad-Relación(ER), diagramas de requisito, el análisis textual y modelado de datos (24).

Equipo de Apoyo al Desarrollo: Servidor de trabajo en equipo que permite a los desarrolladores de *software* trabajar en los mismos proyectos en paralelo. *Visual Paradigm-UML* también ayuda a los desarrolladores a eliminar lo tedioso de la preparación de documentos automáticos mediante un generador de informes que, como indica su nombre, apoya la generación de informes y las publicaciones en formatos populares tales como *HTML*, *Word*[21] y *PDF*[22].

[21] **Word**, *software* destinado al procesamiento de textos.

[22] **PDF,** formato de almacenamiento de documentos, desarrollado por la empresa *Adobe Systems.*

Características principales:

- ✓ Soporte de *UML* versión 2.1.
- ✓ Modelado de procesos de negocio.
- ✓ Requisitos de gestión.
- ✓ Mapeo objeto-relacional capa generación.
- ✓ *IDE*[23] grandes integraciones.
- ✓ Generación de código e Ingeniería Inversa.
- ✓ Colaboración de los equipos (cliente).
- ✓ *PDF*, *MS Word*, *HTML* de generación de informes.
- ✓ Importación y exportación a *XML* e imagen.
- ✓ Forma Editor, diseñar su propia forma (24).

Además de las características antes mencionadas se pueden mencionar otras que también son de gran importancia para el desarrollo de la plataforma: colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto, genera la documentación del proyecto automáticamente en varios formatos como *Word* o *PDF* y permite el control de versiones. Además se destacan su robustez, usabilidad y portabilidad. De acuerdo a lo expuesto anteriormente *Visual Paradigm* es la herramienta correcta para la modelación de cualquier proyecto sobre *software* libre permitiendo de esta forma la independencia y soberanía tecnológica.

1.9.3. Criterio de Selección de la Herramienta CASE a utilizar

Inicialmente en la realización del Análisis y Diseño de la Plataforma de Transmisión Abierta para Radio y Televisión. Se empleó la herramienta *Rational Rose*, pero luego de haber realizado un estudio acerca de las herramientas *CASE* que existen y que se pudieran emplear para diseñar sistemas multiplataforma se decidió escoger la herramienta *Visual Paradigm* debido a todas las facilidades que la misma le aporta al desarrollo de dicha plataforma, entre ellas:

- ✓ Entorno de creación de diagramas para *UML* 2.0.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.

[23] **IDE**, entorno de desarrollo integrado, es un programa compuesto por un conjunto de herramientas para un programador.

- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.

1.10. SERVIDOR WEB.

Un servidor Web es un programa que implementa protocolos como *HTTP* (*Hypertext Transfer Protocol*) y *HTTPS* (*Hypertext Transfer Protocol Secure*). *HTTP* está diseñado para transferir los llamados hipertextos, páginas Web o páginas *HTML* (*Hypertext Markup Language*). Está disponible por medio de Internet, emplea normas en *XML* y sistema de mensajería. Además no está vinculado a ningún sistema operativo o lenguaje de programación (25).

Un servidor Web se encarga de mantenerse a la espera de peticiones *HTTP* llevada a cabo por un cliente *HTTP* que se suele conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. A modo de ejemplo, al hacer una solicitud mediante el navegador, éste realiza una petición *HTTP* al servidor de dicha dirección, el servidor responde al cliente enviando el código *HTML* de la página; el navegador (cliente), una vez recibido el código, lo interpreta y lo muestra en pantalla.

1.10.1. Apache

El grupo *Apache* creó inicialmente una versión 1 de un servidor Web altamente configurable, el cual se hizo popular rápidamente; en la versión 2, este grupo se ha centrado en la escalabilidad, seguridad y rendimiento. Las principales revisiones de código se han llevado a cabo para establecer una arquitectura *Apache* verdaderamente escalable (26).

Apache presenta características importantes que la distinguen y que se empleaan en el desarrollo de la plataforma, entre las que se encuentran:

- ✓ **Es un servidor altamente configurable de diseño modular.** Es muy sencillo ampliar las capacidades del servidor Web *Apache*. Cualquiera que posea una experiencia decente en la programación de *C++* o *Perl* puede escribir un módulo para realizar una función determinada. Esto significa que hay una gran cantidad de módulos *Apache* disponibles para su utilización.

- ✓ **Sencillo, con configuración basada en un poderoso archivo**: el servidor *Apache* no posee una interfaz de usuario gráfica para su administración. Sin embargo, es lo suficientemente flexible para permitirle repartir la configuración de su *host* virtual en múltiples archivos para no sobrecargar un único archivo con toda la gestión de las múltiples configuraciones de servidores virtuales.

- ✓ **Soporte de autentificación *HTTP*:** *Apache* soporta autentificación básica basada en la Web. Está también preparado para autentificación basada en la asimilación de mensajes, que es algo que los navegadores Web populares ya han implementado.

- ✓ **Soporte de *scripts PHP*:** este lenguaje de *script* ha comenzado a ser muy utilizado y ofrece un extenso soporte de *PHP* (26).

Actualmente se considera como uno de los servidores Web más utilizados en el mundo. Más del 60 % de los servidores Web del mundo emplean *Apache*, según encuestas realizadas por una compañía dedicada a encuestas llamada *Netcraft* (www.netcraft.co.uk/Survey/). La Tabla 1 muestra de una manera clara y actualizada como se evidencia lo antes expuesto en el período de agosto de 1995 a enero de 2009.

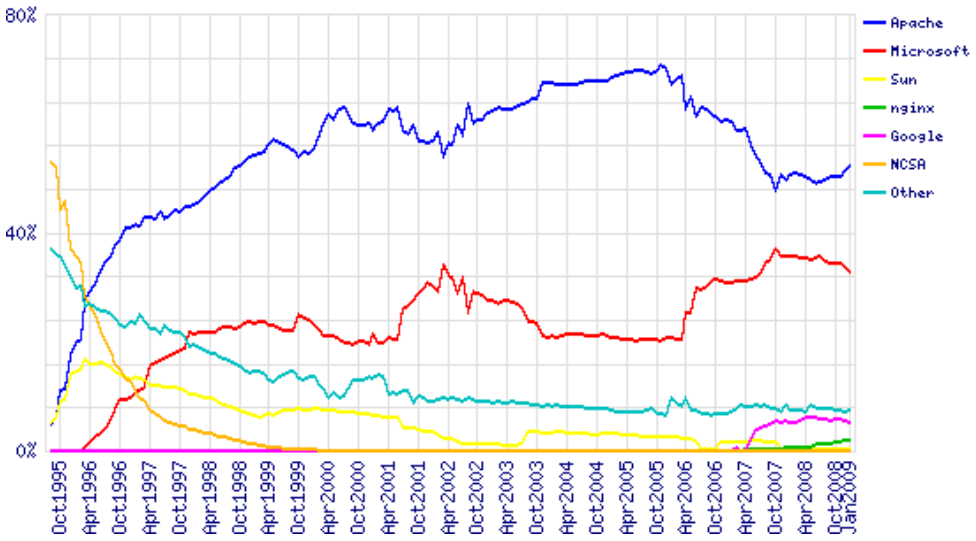


Tabla 1. Cuota de mercado de los servidores principales en todos los dominios.

Sin embargo se puede apreciar como en los últimos años ha sufrido un descenso en su cuota de mercado, pero no ha sido tan significativo, pues sigue estando entre los primeros servidores a nivel mundial.

Se escoge *Apache* como servidor Web para los subsistemas de Programación y Gestión de Medias, ya que presenta mensajes de errores altamente configurables, es una tecnología gratuita con código de fuente abierto que permite su uso por cualquier grupo de desarrolladores, además de que funciona en *Windows*, *Linux* y en otros sistemas de *Unix*.

1.11. SISTEMA OPERATIVO.

El sistema operativo es el programa más importante en una computadora. Para que los restantes programas de uso general funcionen en cada computadora es necesario incluir un sistema operativo. Es el encargado de ejecutar las tareas básicas, entre la que se mencionan, el reconocimiento de la conexión del teclado, enviar la información a la pantalla, no perder de vista archivos y directorios en el disco y controlar los dispositivos periféricos tales como impresoras, escáner y otros. Además se encarga de la seguridad, asegurándose de que usuarios no autorizados no tengan acceso al sistema.

En la actualidad existen sistemas operativos libres y propietarios. Básicamente lo que hace la diferencia entre estos dos tipos de sistemas operativos es que para los libres no es necesario pagar por su uso, brinda libertad de estudiar su código fuente para poderlo modificar atendiendo a las necesidades del usuario, mientras que en el caso de los propietarios se hace imprescindible pagar por su empleo. El sistema operativo *Linux* es un tipo de sistema libre.

1.11.1. Distribución de Linux

Una distribución no es más, que una selección de programas y ficheros, organizados y preparados para su instalación. Estas distribuciones se pueden conseguir mediante Internet, o con la compra de sus CDs[24], los cuales incluyen todo lo que se necesita para la instalación de un sistema *Linux* bastante completo (27).

[24] **CD**, (*Compact Disc*), soporte digital óptico utilizado para almacenar cualquier tipo de información (audio, fotos, video, documentos y otros datos).

Existen muchas y variadas distribuciones de *GNU/Linux*, una de ellas es *UBUNTU*, la cual será empleada como sistema operativo en el desarrollo del trabajo.

Linus Torvalds completó el sistema con su *kernel* (aplicación destinada a comunicar los procesos mediante el *hardware* de la computadora) en el año 1991. A dicho *kernel* se le bautizó con el nombre de *Linux*. A partir de ello, se constituye el sistema *GNU/Linux* (28). Algunas de las características que presenta *GNU/Linux* son:

- ✓ **Multitarea**: La palabra multitarea describe la habilidad de ejecutar varios programas al mismo tiempo. *Linux* utiliza la llamada multitarea preventiva, la cual asegura que todos los programas que se están utilizando en un momento dado serán ejecutados, siendo el sistema operativo el encargado de ceder tiempo de microprocesador a cada programa.
- ✓ **Multiusuario**: Muchos usuarios usando la misma máquina al mismo tiempo.
- ✓ **Multiplataforma**: Las plataformas en las que en un principio se puede utilizar *Linux* son 386, 486, *Pentium*, *Pentium* Pro, *Pentium* II, Amiga y *Atari*, también existen versiones para su utilización en otras plataformas, como *Alpha*, *ARM*, *MIPS*, *PowerPC* y *SPARC*.
- ✓ **Multiprocesador**: Soporte para sistemas multiprocesadores, están disponible para *Intel* y *SPARC*.
- ✓ Todo el **código fuente** está disponible, incluyendo el núcleo completo y todos los controladores (*drivers*), las herramientas de desarrollo y todos los programas de usuario; además todo ello se puede distribuir libremente. Hay algunos programas comerciales que están siendo ofrecidos para *Linux* actualmente sin código fuente, pero todo lo que ha sido gratuito sigue siendo gratuito (28).

Distribución Ubuntu

Ubuntu es un sistema operativo completamente de código abierto construido sobre la base del *kernel Linux* .La comunidad *Ubuntu* está basada en las ideas establecidas en la filosofía *Ubuntu*: que el *software* debe ser accesible totalmente libre de cualquier cargo, que las herramientas de *software* deben ser utilizadas por las personas en su idioma local y sin tener en cuenta cualquier tipo de discapacidad, que las personas pueden tener la libertad de personalizar y modificar su *software* en la forma que les resulte más provechoso. Por esas razones *Ubuntu* (29):

- ✓ Es gratuito y no presenta ningún coste adicional para la edición empresarial.

- ✓ Usa lo mejor en infraestructura de traducciones y accesibilidad que la comunidad del *software* libre es capaz de ofrecer, permitiendo que sea utilizable por el mayor número de personas posible.
- ✓ Se publica de manera regular y predecible, se publica una nueva versión cada seis meses. Permite usar la versión estable actual o ayudar a mejorar la versión actualmente en desarrollo. Cada versión está soportada al menos durante 18 meses.
- ✓ Está totalmente comprometido con los principios del desarrollo de *software* de código abierto.

De acuerdo a ello, se convierte en una distribución orientada al escritorio, pero con bastante estabilidad. Fundamentalmente comparte las ventajas de *Debian* (exceptuando que tiene ligeramente menos paquetes, y que estos no están probados del todo), sumándole el hecho de tener una distribución bastante actualizada.

1.12. CONCLUSIONES PARCIALES.

En este capítulo se abordaron varios temas que conforman la fundamentación teórica para el desarrollo de los subsistemas de Programación y Gestión de Medias. De manera general, está centrado en la justificación de la elección de las tendencias y tecnologías actuales que se tuvo en cuenta para la construcción del prototipo funcional.

Capítulo 2

Descripción de la Propuesta

2.1 INTRODUCCIÓN

En este capítulo se hace una breve descripción de las funcionalidades de los subsistemas que fueron implementadas, así como de los distintos módulos que los conforman. Se muestra el modelo de implementación de la plataforma y los diagramas de componentes correspondientes a los subsistemas de Programación y Gestión de Medias, con el fin de que se entienda mejor la estructura de dichos subsistemas. Se detalla el estilo de programación empleado en el desarrollo de los subsistemas.

2.2 DESCRIPCIÓN DE LOS SUBSISTEMAS

La Plataforma de Transmisión Abierta para la Radio y Televisión está compuesta por diferentes subsistemas, de los cuales se describen los Subsistemas de Programación y Gestión de Multimedia.

2.2.1 Subsistema de Programación

Teniendo en cuenta que el desarrollo de la plataforma está basado en el *framework Symfony*, el cual en su estructura divide el sistema en módulos para una mejor comprensión y desarrollo de los mismos, se decidió dividir el subsistema en dos módulos para la correcta implementación en el *framework*.

Módulo de Programación

Este módulo es el encargado de manejar todo lo relacionado con la programación de la radio y la televisión de la Plataforma de Transmisión Abierta para Radio y la Televisión. Entre sus principales funcionalidades resalta la gestión de los espacios televisivos y de radio, esto no es más que fijar para un canal una programación estándar, por ejemplo para un canal X se adiciona la programación estándar de que todos los lunes, miércoles y viernes de 1:00 pm a 2:00 pm se proyecte el noticiero deportivo, esta programación es inviolable a la hora de confeccionar la programación de dicho canal. Otra funcionalidad del módulo es permitir gestionar las programaciones de radio y televisión de los canales de la plataforma.

Módulo de Reporte

Mediante este módulo es posible visualizar diferentes reportes, entre los que se encuentra el reporte de programación, que muestra todo lo que se proyectó en ocasiones anteriores en los canales, lo cual aporta una gran importancia para la plataforma, debido a que ayuda en la confección de una mejor programación

de sus canales televisivos, teniendo en cuenta lo que se ha programado para no repetir programas que puedan resultar no deseados y a su vez ofrecerle una mayor variedad al público.

Otro reporte con que cuenta este módulo es el de transmisión, el cual muestra todos los problemas que ocurrieron en la transmisión de los programas, elaborados por el transmisor del canal, este puede ayudar a conocer si realmente se proyectó lo que se quería en la programación del canal, de lo contrario se reajusta dicha programación o simplemente se tiene en cuenta que ese programa no se visualizó al público para que conste en dicho reporte. Por último está el reporte de estado de conversión, el cual permite saber en qué estado se encuentra la conversión de los materiales para ser proyectados en el canal.

2.2.2 Subsistema de Gestión de Medias

Este subsistema está integrado por un único módulo, en él se encapsulan todas la funcionalidades del subsistema.

Módulo de Medias

Este módulo se encarga de toda la gestión de las medias en el servidor de la Plataforma de Transmisión Abierta para la Radio y la Televisión, además del chequeo de la integridad de la misma, esto quiere decir que una vez convertida la media se tiene que revisar su calidad, para que no esté desfasada, en otras palabras que el audio y el video estén sincronizados y que el subtítulo esté correctamente en la media, en caso contrario se vuelve a convertir la media o sencillamente no se acepta la misma.

2.3 MODELO DE IMPLEMENTACIÓN

El modelo de implementación denota la implementación actual del sistema en términos de componentes y subsistemas de implementación (21). Describe cómo se organizan los elementos de acuerdo a los mecanismos de estructuración y modularización disponibles en el entorno de implementación, el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes entre sí.

El objetivo principal del modelo de implementación es identificar los componentes físicos de la implementación para que puedan comprenderse y gestionarse mejor. Este modelo define las principales

unidades de integración alrededor de las cuales se organizan los equipos, así como las que se pueden versionar, desplegar y reemplazar separadamente.

En el siguiente modelo de implementación se representan componentes que hacen referencia a librerías y aplicaciones externas del propio *framework* de desarrollo, utilizadas para la implementación de algunas funcionalidades de la aplicación; tal es el caso de *LDAP*[25], para la autenticación mediante dominio, *Propel* para el mapeo de objetos a base de datos (conocido como ORM, de "*object-relational mapping*")para *PHP,* se encarga de la generación del modelo*,* ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario para la interacción con la base de datos, proporcionando persistencia para los objetos y un fácil servicio de consultas, *VLC* para la reproducción de archivos multimedia y la clase Embed_Video la cual se utiliza para crear un reproductor elaborado completamente en *JavaScript* permitiendo visualizar y embeber el video dentro de la aplicación.

A continuación se muestra en la figura el modelo de implementación de la Plataforma de Transmisión Abierta para la Radio y la Televisión.

[25] **LDAP,** (*Lightweight Directory Access Protocol*), en español, Protocolo de Acceso a Directorios Ligeros, protocolo usado para acceder a "Servidores de Directorio".

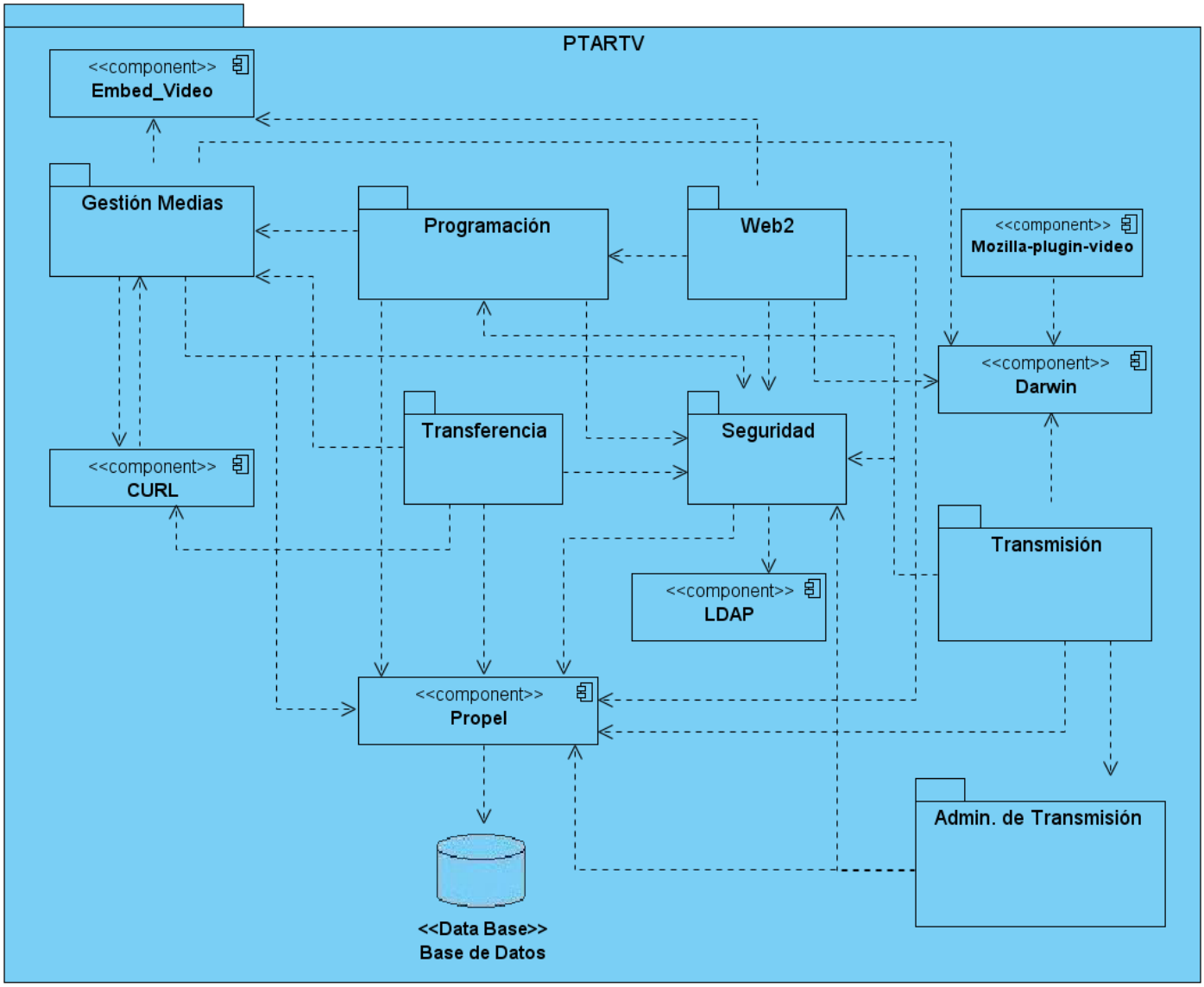


Figura 2. Modelo de implementación.

Seguidamente se visualiza como están confeccionados los subsistemas de Programación y Gestión de Medias en el modelo de implementación, para un mejor entendimiento del mismo.

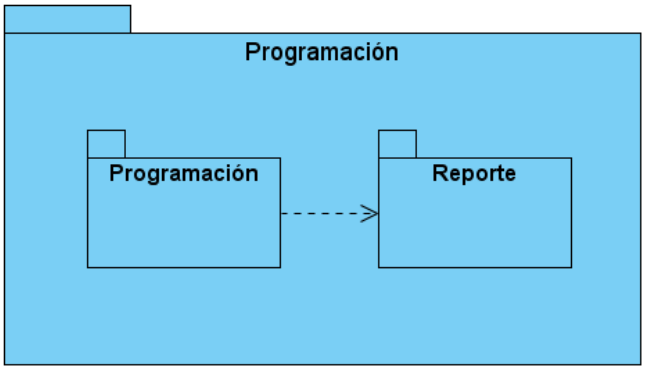


Figura 3. Modelo Subsistema de Programación.

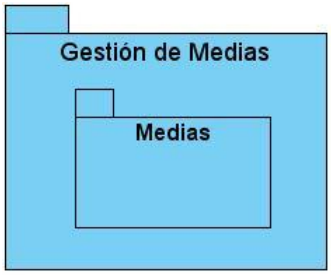


Figura 4. Modelo Subsistema Gestión de Medias.

Para una mejor comprensión del modelo de implementación se detallan como están divididos los subsistemas de Programación y Gestión de Medias.

2.3.1 Diagrama de componentes

Los diagramas de componentes muestran la estructura de componentes, incluyendo los clasificadores que especifican componentes y los artefactos que los implementan. La metodología RUP enfatiza en el uso del diagrama de componentes para modelar los subsistemas de implementación. Los componentes representan los elementos de un modelo dentro de un paquete, como son las clases en el modelo de diseño. Presentan estereotipos estándar como pueden ser:

- ✓ <<*executable*>> programas que se ejecutan en un nodo.
- ✓ <<*file*>> ficheros de datos o código fuente.
- ✓ <<*library*>> librerías estáticas o dinámicas.
- ✓ <<*table*>> tabla de base de datos.

- ✓ <<*document*>> documento.

Seguidamente se representan los diagramas de componentes relacionados con los subsistemas vistos anteriormente.

Subsistema de Programación

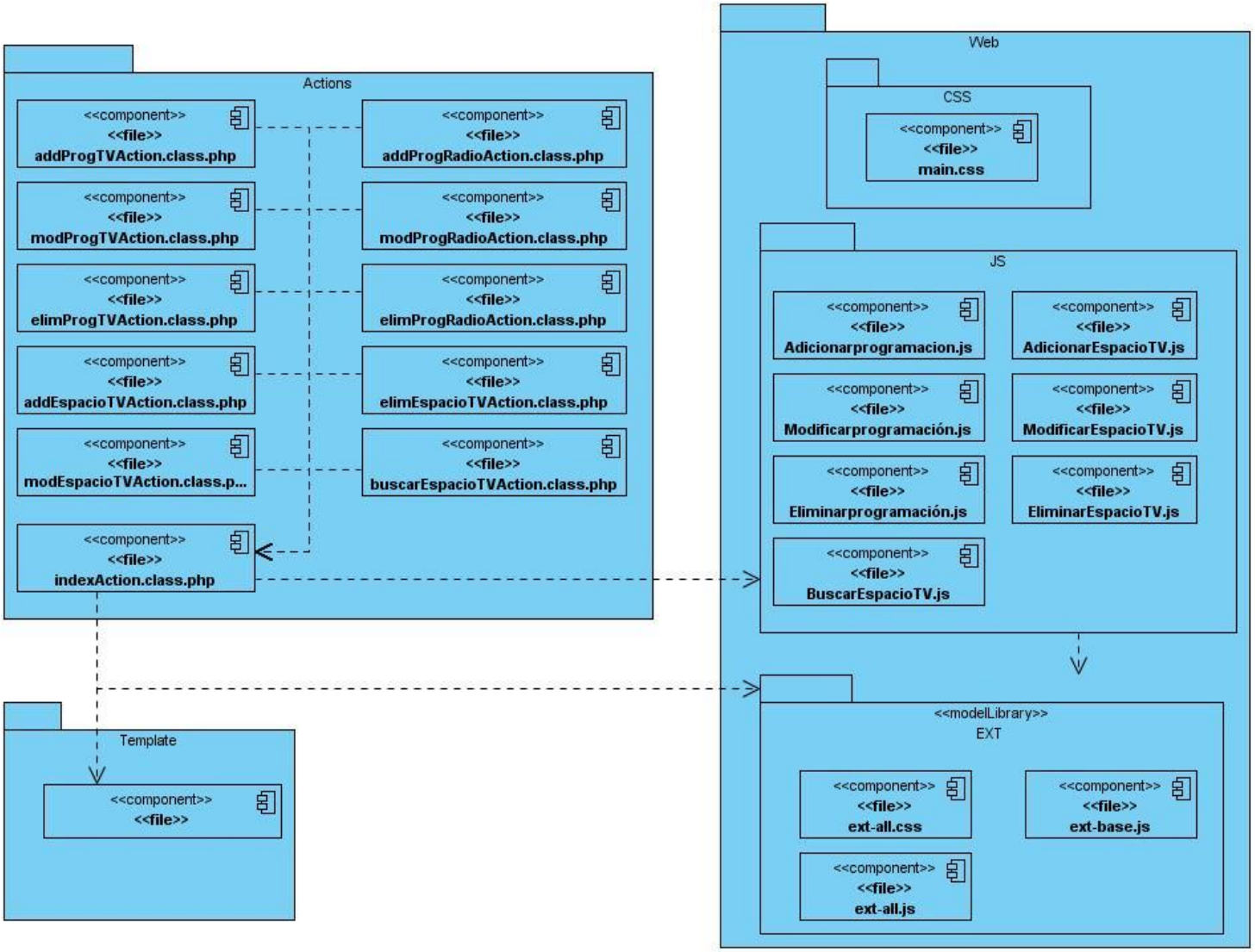


Figura 5. Diagrama de Componente del Módulo de Programación.

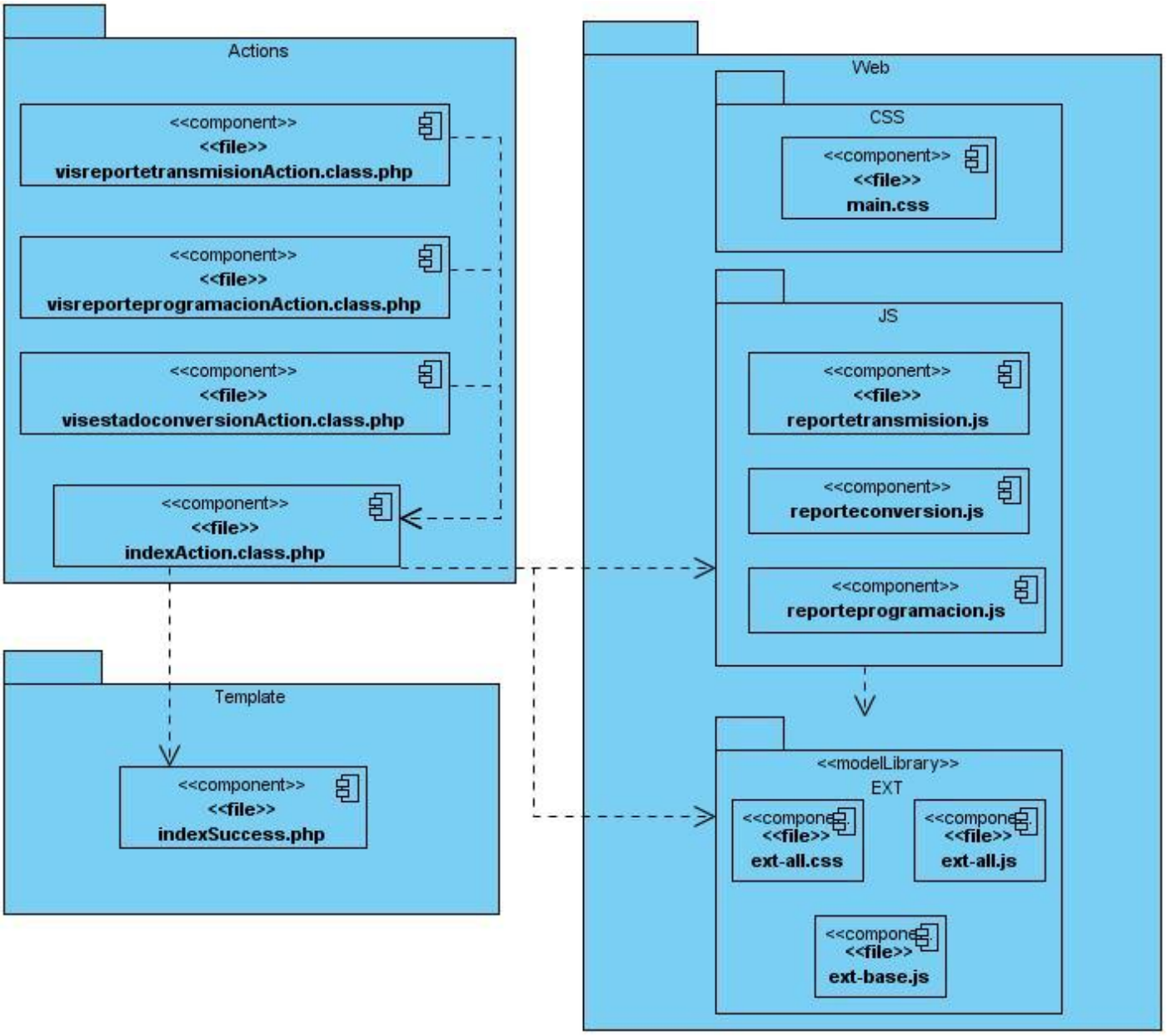


Figura 6. Diagrama de Componentes Módulo de Reporte.

Subsistema de Gestión de Medias

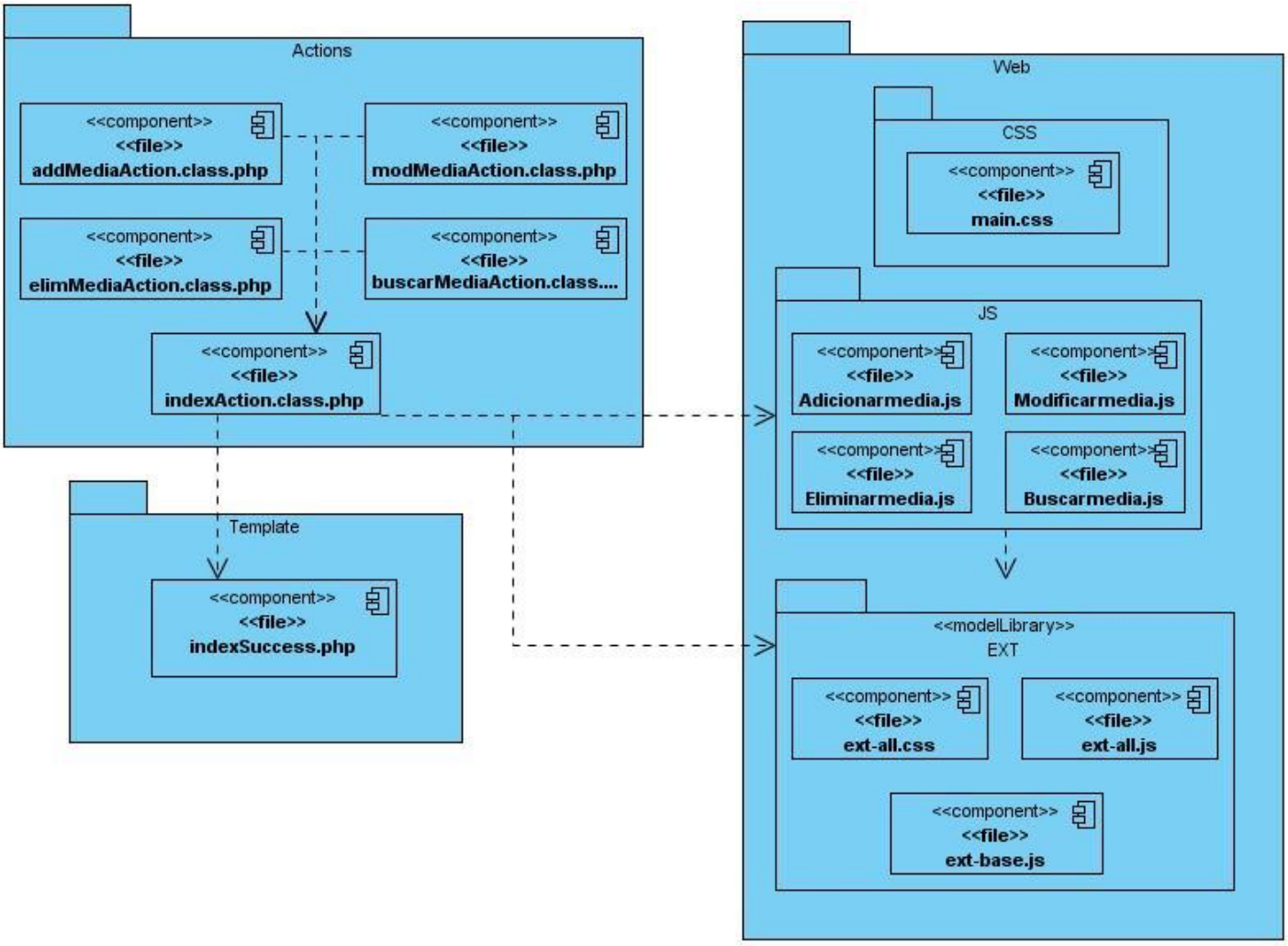


Figura 7. Diagrama de Componentes Módulo de Medias.

2.4 ESTILOS DE PROGRAMACIÓN

Se define estilo de programación a la manera en que se le da un formato al código fuente, esto incluye la forma en que se colocan las llaves, cómo se indenta el código y dónde se emplean los paréntesis. Todo programador posee un estilo propio de escribir su código fuente, pero para poder lograr un buen estilo de programación deberá utilizar una estructura de código que sea fácil de entender, no solo para terceras

personas, sino también para él mismo. Un buen estilo de programación consta de diversos criterios, entre los que se mencionan:

- ✓ Nombres significativos para variables, controles y procedimientos.
- ✓ Indentación (sangrías) y espacios apropiados en el código.
- ✓ Documentar el código (poner comentarios para aclarar).
- ✓ Módulos adaptables.
- ✓ Minimizar alcance de los datos hasta donde sea posible.

Nombres significativos: Los nombres de variables, controles y procedimientos definidos por un usuario deben ser significativos. Con el fin de ayudar al usuario en la asignación de nombres se brinda un estándar de nombres convencionales.

Indentación y espacio apropiado en el código: La indentación se emplea para obtener una mejor visibilidad en el diseño de un programa. La misma muestra las líneas que están sujetas a otras. Por ejemplo, todas las líneas que forman el cuerpo de un ciclo serán indentadas con la instrucción principal del mismo.

Documentar el Código: Las secciones de código más complicadas e inusuales deberán ser documentadas. Cada variable y arreglo tendrán que estar comentarios en el lugar que sean definidos para que su función pueda ser comprendida posteriormente.

Procedimientos Coherentes: Cada procedimiento deberá ser diseñado para una tarea simple, pues si alguno de ellos maneja muchas tareas, es lógico que pueda ser difícil de entender y pueda ocurrir fácilmente un error.

Minimizar alcance de los datos hasta donde sea posible: Las variables y los arreglos pueden ser accedidos mediante códigos en distintas partes de un programa y si son globales desde cualquier lugar. Esto sería lo ideal, pero si no hay cuidado pudiesen ocurrir algunos efectos extraños en otras partes del programa, como colocar un valor en una variable por error. Restringiendo el rango de acceso o el alcance de una variable o un arreglo se puede evitar este problema. Un alcance intermedio es el acceso necesario

para una variable durante todo el simple módulo. Esto significa que cualquier procedimiento en el módulo puede tener acceso a la variable, pero los procedimientos en otros módulos tienen acceso denegado.

Mediante el empleo de un estilo de programación para la confección de los subsistemas de Programación y Gestión de Medias es posible obtener un código más legible, permitiendo de esta forma la reducción de errores lógicos en el código y su reutilización.

En la actualidad existen varios estilos de programación desarrollados por diferentes autores reconocidos, estos son utilizados por una gran parte de los desarrolladores a nivel mundial debido a su fácil estructura y organización. Para la programación de los subsistemas se seleccionó el estilo *Allaman*, también conocido como "*ANSI style*". Seguidamente se describen algunos criterios que lo conforman.

- ✓ **Indentación**

Plantea que se debe usar los sangrados para indentar el código, nunca espacios. Poner las llaves de control en la línea subsiguiente.

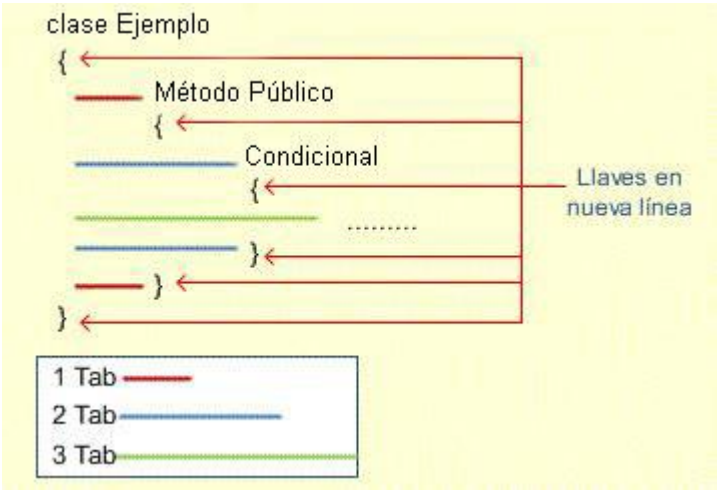


Figura 8. Ejemplo de indentación.

- ✓ **Saltos de Línea**

Define que se debe añadir un salto de línea después del cierre de los paréntesis de los parámetros y detrás de un punto y coma, cuando termina la sentencia.

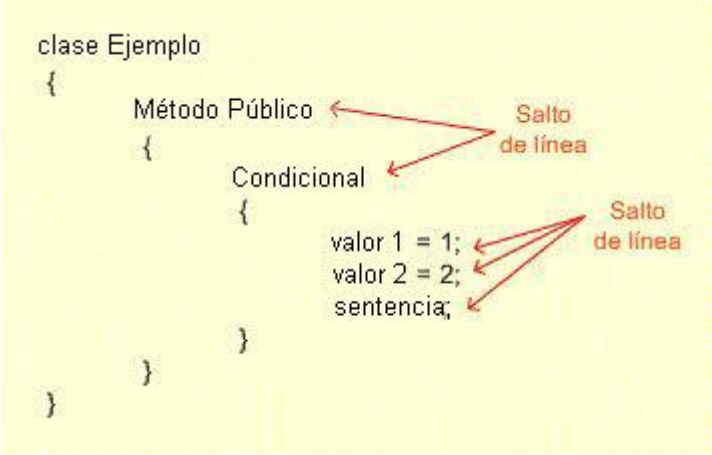


Figura 9. Ejemplo de saltos de línea.

- ✓ **Espacios y líneas en blanco**

Este criterio plantea que se debe:

- Usar espacios en blanco para mejorar la legibilidad del código.
- Usar espacios en blanco a ambos lados del operador de símbolos, después de comas y después de las declaraciones.
- Usar líneas en blanco para separar trozos de código.
- Usar líneas en blanco antes de cada método dentro de la clase.

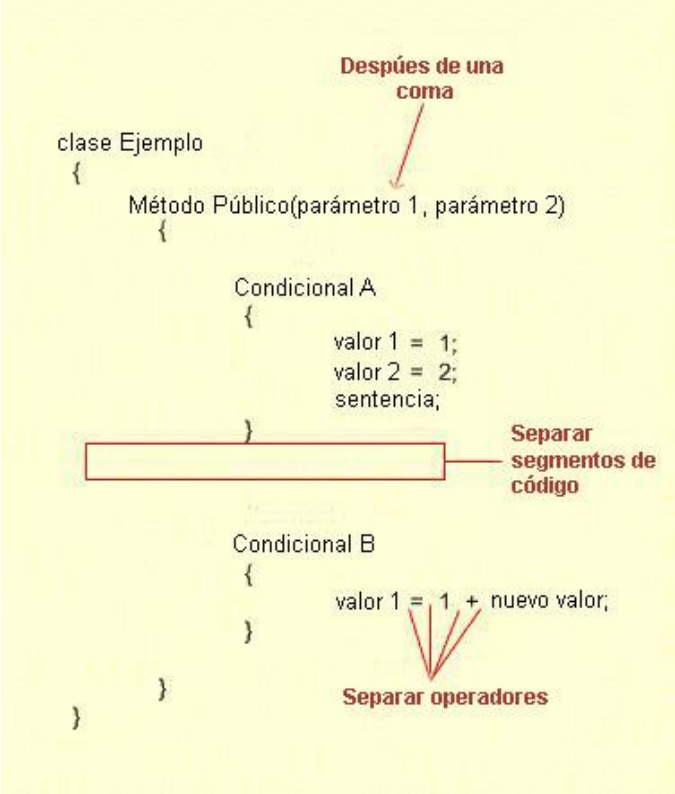


Figura 10. Ejemplo de espacios y líneas en blanco.

El estilo de *Allaman* se caracteriza por conservar un código limpio y claro. Las llaves de inicio y fin coinciden en la misma columna, permitiendo de esta forma que la identificación de cada bloque sea más fácil. Este presenta el inconveniente de que las llaves ocupan una línea enteramente, ocupando más espacio vertical de lectura.

2.5. CONCLUSIONES PARCIALES

En este capítulo se expusieron las distintas funcionalidades que presentan los subsistemas de Programación y Gestión de Medias, haciendo énfasis en los módulos que los componen, con el fin esclarecer su funcionamiento. Se definió y caracterizó el estilo de programación puesto en práctica en la implementación de los subsistemas, para servir de ayuda otros desarrolladores a entender el código fuente.

CAPÍTULO III

Validación de la Solución Propuesta

3.1. INTRODUCCIÓN

En el desarrollo de aplicaciones Web, las pruebas son de vital importancia, ya que aseguran la calidad de la aplicación. En el presente capítulo se aborda todo lo referente a las herramientas y elementos empleados en la validación de la aplicación con el objetivo de garantizar el correcto funcionamiento de la misma.

3.2. AUTOMATIZACIÓN DE PRUEBAS

Los desarrolladores de aplicaciones Web emplean gran esfuerzo en probar correctamente una aplicación, pues resulta ser una tarea verdaderamente tediosa crear casos de prueba, ejecutarlos y analizar sus resultados. Debido a los constantes cambios que surgen en los requisitos de una aplicación y la refactorización continua del código es muy probable que aparezcan errores en la misma, es por ello que la automatización de pruebas es una opción muy útil para la creación de un entorno de desarrollo satisfactorio. Este tipo de pruebas obligan a los programadores a crear nuevas pruebas en un formato estandarizado y muy rígido que pueda ser procesado por un *framework* de pruebas.

Las pruebas automatizadas pueden reemplazar la documentación técnica de la aplicación, ya que ilustran de manera clara el funcionamiento de la aplicación. Estas pruebas comparan el resultado de un método con la salida esperada del mismo, es decir, evalúan “asertos”, que no son más que expresiones de tipo $a == 2. El valor de salida de un aserto puede ser *true* o *false*, lo que determina si la prueba tiene éxito o falla. La palabra “aserto” tiene un uso muy común en las técnicas de automatización de pruebas.

3.3. PRUEBAS UNITARIAS Y FUNCIONALES

Dependiendo del modelo del *software* empleado para el desarrollo de aplicaciones y su implementación estos dos tipos de pruebas pueden jugar un papel primordial. Para el caso de las pruebas unitarias se analiza una porción de código que pueda ser analizada de manera aislada, como por ejemplo funciones y métodos, a los que después de pasarle parámetros de entrada se obtienen otros de salida claramente definidos, es decir, validan la forma en la que las funciones y métodos trabajan de manera particular. Este tipo de pruebas se encargan de validar los casos uno a uno, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto.

Existen numerosos *frameworks* para crear pruebas unitarias, dentro de los que se destacan el *PhpUnit* y *SimpleTest*. En el caso particular de *Symfony* incluye su propio *framework* para pruebas llamado *Lime*, el cual se basa en la librería *Test::More* de *Perl*, además de ser compatible con *TAP*, lo que significa que los resultados de las pruebas se muestran con el formato definido en el "*Test Anything Protocol*", creado para facilitar la lectura de los resultados de las pruebas.

Además de proporcionar soporte para pruebas unitarias, *Lime* tiene otras ventajas que lo caracterizan:

- ✓ Las pruebas de *Lime* son fáciles de leer y sus resultados también lo son, debido que utiliza diferentes colores para mostrar de forma clara la información más importante.
- ✓ El núcleo de *Lime* se valida mediante pruebas unitarias.
- ✓ Emplea el lenguaje *PHP*, el cual es muy rápido y está bien diseñado internamente.

Las pruebas funcionales evalúan la aplicación como un todo, comprueban que se alcancen determinados resultados tras realizar una serie de acciones sobre la aplicación y validan partes de las aplicaciones. Estas pruebas simulan la navegación del usuario, realizan peticiones y comprueban los elementos de respuesta, tal y como lo haría manualmente un usuario para validar que una determinada acción hace lo que se supone que debe que hacer. En ellas se ejecuta un escenario correspondiente a lo que se denomina un "caso de uso".

Para este tipo de pruebas el *framework Symfony* dispone de un objeto especial, llamado *sfBrowser*, que actúa como un navegador que está accediendo a una aplicación, pero sin necesidad de utilizar un servidor Web real. Este objeto permite el acceso directo a los objetos que forman cada petición (el objeto petición, el objeto sesión, el contexto y el objeto respuesta). También dispone de una extensión de esta clase llamada *sfTestBrowser*, que está especialmente diseñada para las pruebas funcionales y que tiene todas las características de *sfBrowser*, además de algunos métodos muy útiles para los asertos.

3.3.1. Herramientas empleadas en las Pruebas

Los elementos descritos anteriormente que incluye *Symfony* para realizar pruebas unitarias y funcionales no son suficientes para la mayoría de los casos, sobre todo a la hora de probar las interacciones del lado cliente. El principal inconveniente que presentan estas técnicas es que no pueden simular el

comportamiento de *JavaScript*. Si se definen interacciones muy complejas, como por ejemplo interacciones con *Ajax*, es necesario reproducir de forma exacta las pulsaciones de teclado que realiza el usuario y ejecutar los scripts de *JavaScript*. Normalmente, estas pruebas se hacen a mano, pero necesitan de la dedicación de mucho tiempo y son propensas a cometer errores.

Una solución a los problemas antes expuestos, es el empleo de *Selenium,* el cual no es más que un *framework* de pruebas escrito completamente en *JavaScript*. La principal ventaja del uso de esta herramienta es que permite realizar una serie de acciones en la página de la misma forma que las haría un usuario normal. Además de la ventaja de *Selenium* sobre el objeto *sfBrowser* de *Symfony* de ser capaz de ejecutar todo el código *JavaScript* de la página, incluidas las interacciones creadas con *Ajax*. Las pruebas de *Selenium* se escriben en *HTML*, por lo que cada caso de prueba consiste en una página *HTML*, con una tabla de 3 columnas: comando, destino y valor.

La figura que aparece a continuación muestra un fragmento del caso de prueba Adicionar Espacio Televisivo donde se observan los elementos *HTML* presentes en el mismo.

```
<title>adicionar</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">adicionar</td></tr>
</thead><tbody>
<tr>
    <td>open</td>
    <td>/programacion/index</td>
    <td></td>
</tr>
<tr>
    <td>click</td>
    <td>link=» Adicionar</td>
    <td></td>
</tr>
<tr>
    <td>verifyTextPresent</td>
    <td>Adicionar Espacio</td>
    <td></td>
</tr>
```

Figura 11. Ejemplo de Caso de Prueba.

Las pruebas de *Selenium* al ser creadas con *HTML*, arriba a que sea muy aburrido y tedioso escribir todo ese código *HTML*. Para ello existe una extensión de *Selenium* para *Firefox* que permite grabar todos los movimientos y acciones realizadas sobre una página y guardarlos como una prueba. Una vez realizados

todos los movimientos y añadidos todos los comandos, se pueden guardar en un archivo *HTML* para añadirlo al conjunto de pruebas. Este conjunto o *suite* de pruebas se define en archivo *TestSuite.html,* el cual se encarga de guardar las referencias a cada uno de los archivos de pruebas guardados independientemente y mostrarlos en una lista.

```
<table id="suiteTable"
       cellpadding="1"
       cellspacing="1"
       border="1"
       class="selenium">
    <tbody>
        <tr><td><b>Test Suite</b></td></tr>
        <tr><td><a href="adicionar">adicionar</a></td></tr>
    </tbody>
</table>
```

Figura 12. Estructura de TestSuite.html.

3.3.2. Diseño de Pruebas

Para validar el funcionamiento de la aplicación se realizan algunas pruebas funcionales apoyadas en el uso de *Selenium* debido a que permite probar el funcionamiento de diferentes formularios varias veces seguidas. Las pruebas que realiza son semejantes a las que haría cualquier usuario desde un navegador, con la ventaja de que las hace mucho mas rápido y evita el trabajo repetitivo de probar una y otra vez lo mismo manualmente. Seguidamente se muestra la relación de dos casos de prueba implementados para la validación de la solución propuesta, el primero consiste en adicionar un espacio televisivo y el segundo en visualizar un reporte de transmisión.

Caso de prueba	Adicionar Espacio Televisivo
Condiciones	El usuario debe estar registrado en la base de datos y su rol debe ser el de Administrador de Programación.
Datos de entrada	Espacios(noticieros, aventuras)
Resultados esperados	1. Que el espacio quede correctamente adicionado.
Resultados obtenidos	1. La prueba es satisfactoria pues el espacio quedó correctamente adicionado.
Observaciones	Al realizar la prueba con la herramienta *Selenium* se obtuvo el resultado correcto.

Tabla 2. Caso de Prueba # 1.

Caso de prueba	Visualizar Reporte de Transmisión
Condiciones	El usuario debe estar registrado en la base de datos y su rol debe ser el de Administrador de Programación.
Datos de entrada	Fecha, Canal, Nada
Resultados esperados	1. Que se muestre el reporte correctamente.
Resultados obtenidos	1. La prueba es satisfactoria pues el reporte se mostró correctamente.
Observaciones	Al realizar la prueba con la herramienta *Selenium* se obtuvo el resultado correcto.

Tabla 3. Caso de Prueba # 2.

En la siguiente figura se muestra el resultado del caso de prueba Adicionar Espacio Televisivo donde se relacionan los elementos presentes a la hora de ejecutar una prueba con la herramienta *Selenium*.

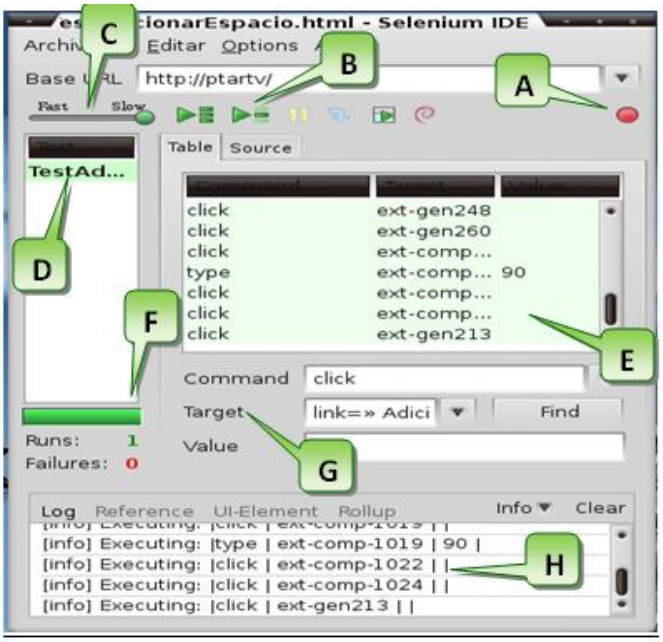


Figura 13. Elementos de Selenium IDE.

A: Graba los movimientos de la navegación.

B: Permite escoger que prueba presente en la *suite* ejecutar, una a una o todas a la vez. Pausar o continuar las pruebas y realizarlas paso a paso.

C: Determinar la velocidad de ejecución de pruebas para que el usuario pueda apreciar la prueba.

D: *Suite* de Pruebas, lista todas las pruebas presentes en la *suite*.

E: Datos de la prueba presentes en el código de cada una.

F: Cantidad de pruebas realizadas: aceptadas (verde) y fallidas (rojo).

G: Permite cambiar los valores de una prueba en tiempo de ejecución.

H: Log. Registra los eventos y sucesos de cada prueba.

3.4. CONCLUSIONES PARCIALES

En este capítulo se hizo referencia a los principales elementos relacionados con los tipos de pruebas referentes a la fase de implementación y algunas de las herramientas empleadas. Es importante señalar que el ejemplo de caso de prueba mostrado anteriormente no fue al único que se le realizaron estas pruebas. El objetivo principal de este capítulo es precisamente conocer como se manejan estos conceptos de pruebas en el contexto de la implementación de una aplicación.

CONCLUSIONES GENERALES

Al culminar con la investigación se llegaron a las siguientes Conclusiones Generales:

- ✓ Mediante la valoración de tendencias y tecnologías actuales fue posible que se seleccionaran los elementos y herramientas que se emplearon en el desarrollo de los subsistemas de Programación y Gestión de Medias.
- ✓ La descripción del estilo de programación seleccionado para la implementación de los subsistemas posibilita una mejor comprensión del código fuente a demás desarrolladores.
- ✓ En el desarrollo de los subsistemas de Programación y Gestión de Medias se implementaron los requisitos funcionales requeridos para su correcto funcionamiento.
- ✓ Para certificar la veracidad de los algoritmos empleados en la implementación de los subsistemas de Programación y Gestión de Medias se desarrollaron Casos de Prueba que aseguran la validación de los mismos.
- ✓ Se desarrolló un manual de usuario para la instalación de los subsistemas, lo cual facilita el trabajo de los clientes con estos subsistemas.

RECOMENDACIONES

Tomando como base la investigación realizada y la experiencia acumulada durante la implementación de los subsistemas de Programación y Gestión de Medias, se proponen las siguientes recomendaciones:

- ✓ Optimizar la codificación de la aplicación y actualizar la versión del framework empleado.
- ✓ Fusionar los subsistemas de Programación y Gestión de Medias para que funcionen como un todo.
- ✓ Buscar alguna variante más eficiente para el *plugin* del navegador *Mozilla* que se encarga de la reproducción de videos.
- ✓ Permitir que los reportes se puedan exportar a formato PDF para una mejor funcionalidad de dichos reportes.
- ✓ Buscar una variante más eficiente para subir las medias al servidor.
- ✓ Dar continuidad al desarrollo y soporte de los subsistemas desarrollados.

TRABAJOS CITADOS

1. *TVCubana*. [en línea] [citado el: 27 de 11 de 2008.] Http://www.tvcubana.icrt.cu/?mod=seccion&id=6.

2. *Avizora*. [En línea] [Citado el: 27 de 11 de 2008.]
http://www.avizora.com/publicaciones/television/textos/historia_television_0001.htm.

3.W3C. [En línea] [Citado el: 10 de 12 de 2008.]
http://www.w3.org/.

4. **Mora, Sergio Luján**. *Programación de aplicaciones de web: historia, principios básicos y clientes web.* **ISBN** 84-8454-206-8.

5. **Márquez Avendaño, Zulaica Rugarcía.***Universidad de las Américas Puebla***.** [En línea] [Citado el: 18 de 1 de 2009.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo_5.html#.

6. **Virú, Ing. Wigberto Martín Nicho.** *Scribd*. [En línea] [Citado el: 20 de 1 de 2009.]
http://www.scribd.com/doc/272974/Cliente-servidor.

7. *Universidad de los Andes*. [En línea] [Citado el: 18 de 1 de 2009.]
http://agamenon.uniandes.edu.co/~revista/articulos/cliser.html.

8. Instituto Superior Colina. [En línea] [Citado el: 20 de 1 de 2009.]
http://labredes.itcolima.edu.mx/fundamentosbd/sd_u1_6.htm.

9. **Garrett, Jesse James**. JJG. [En línea] [Citado el: 15 de 1 de 2009.]
http://adaptivepath.com/ideas/essays/archives/000385.php.

10. **McLaughlin, Brett.** IBM. [En línea] [Citado el: 20 de 1 de 2009.]
http://www.ibm.com/developerworks/xml/library/wa-ajaxintro1.html.

11. *Universidad Tecnológica Nacional*. [En línea] [Citado el: 21 de 1 de 2009.]
http://www.frt.utn.edu.ar/sistemas/paradigmas/page22.html.

12. *Universidad Tecnológica Nacional*. [En línea] [Citado el: 21 de 1 de 2009.] http://www.frt.utn.edu.ar/sistemas/paradigmas/page23.html.

13. *Universidad de Burgos*. [En línea] [Citado el: 19 de 1 de 2009.] http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_1.htm.

14. **Ortíz, Dr. Manuel Martín.** *Universidad Autonoma de Puebla.* [En línea] [Citado el: 20 de 1 de 2009.] http://www.cs.buap.mx/~mmartin/notas/BD_CS2004_v3.pdf.

15. **Chuck Musciano, Bill Kemedy.** *HTML. La guía completa (Español)*, **ISBN** 970-10-2141-X.

16. **Flanagan, David**. *The Defenitive Guide.***ISBN** 9780596101992.

17. **Vázquez, Jose Antonio Gallego.** *Desarrollo Web con PHP y MySQL.* s.l. : Ediciones Anaya Multimedia, 2003. **ISBN** 85-415-1525-5.

18. Ricardo, Febe Ángel Ciudad*. Informática 2009*. [En línea] [Citado el: 2009 de 2 de 10.] http://www.informaticahabana.com/evento_virtual/files/MUL053.pdf.

19**. Potencier, Fabien y Zaninotto, Francois**. *Symfony*, la guia definitiva. 2007. **ISBN** *978-1590597866*

20. *Universidad de Murcia*. [En línea] [Citado el: 9 de 1 de 2009.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

21. **Ivar Jacobson, Grady Booch y James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, 2000. **ISBN** 84-7829-036-2.

22.**Ivar Jacobson, Grady Booch y James Rumbaugh.** *El Lenguaje Unificado de Modelado.Manual de Referencia. 2000.* **ISBN** *78-84-7829-087-1.*

23. *Rational*. [En línea] [Citado el: 22 de 2 de 2009.] http://www.rational.com.ar/herramientas/roseenterprise.html.

24. *Visual Paradigm*. [En línea] [Citado el: 16 de 2 de 2009.] http://www.visual-paradigm.com/product/vpuml/enterpriseedition.jsp.

25. **Cerami, Ethan.** *Web Services Essentials.* s.l. : O'Reilly, 2002. **ISBN** 0-596-00224-6.

26. **Kabir, Mohammed J**. *Servidor Apache 2, 2003.* **ISBN** *9788441514683.*

27. *Linux-es*. [En línea] [Citado el: 20 de 2 de 2009.] http://www.linux-es.org/distribuciones.

28. **Perpiñan, Antonio.** *GNU/Linux Básicamente,* **ISBN** *88-99999-99 -9.*

29. *Ubuntu*. [En línea] [Citado el: 6 de 2 de 2009.] http://www.ubuntu.com/.

BIBLIOGRAFÍA

1. **Rolando Alfredo Hernández León, Zayda Coello González**. *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA. Ciudad de la Habana : EDUNTIV, 2002.* **ISBN** *978-959-16-0343-2.*

2. **CRAIG, LARMAN.** *UML Y PATRONES.INTRODUCCION ANALISIS Y DISEÑO ORIENTADO A OBJETOS . s.l. : PRENTICE HALL HISPANOAMERICANA. 978-970-17-0261-1.*

3. **Darie, Cristian.,** *AJAX and PHP.Building Responsive Web Applications (Inglés) [et.al] / Birmingham, Packt Publishing, 2006, 286 p.*; **ISBN** *1-904811-82-5.*

4. **Frentzen, J. / Madrid, McGraw-Hill,** *Superutilidades para JavaScript (Español) 1999, 549 p.* **ISBN** *84-481-2124-4.*

5. **Ramos Monso, Mario,** *Programación PHP. Sitios Web dinámicos e interactivos (Español), MP Ediciones, 2004, 424;* **ISBN** *987-526-202-1.*

6. **Potencier, Fabien y Zaninotto, Francois**. *Symfony*, la guia definitiva. 2007. **ISBN** *978-1590597866*

7. **Pressman, Roger S**. *Ingeniería de software. Un enfoque practico*. s.l. : McGraw-Hill. **ISBN** *84-481-3214-9*

8. **Orestes Lima Córdova, Eridniel Suárez Contreras**. *Diseño de una Plataforma de Transmisión Abierta para la Radio y la Televisión. . Ciudad de la Habana : s.n., 2008.*

GLOSARIO DE TÉRMINOS

Aplicación: Programa informático creado para facilitar al usuario un determinado tipo de trabajo. Esto lo caracteriza frente a otros programas como los sistemas operativos, las utilidades y los lenguajes de programación.

Cliente: Aplicación informática cuya función es acceder a los servicios que ofrece un servidor, haciendo uso generalmente de una red de telecomunicaciones.

CSS (*Cascading Style Sheets):* En español Hojas de Estilo en Cascada, ofrece la posibilidad de separar la estructura de un documento estructurado escrito en *HTML* o *XML* de su presentación.

DOM (*Document Object Model):* provee un conjunto estándar de objetos para representar documentos *HTML* y *XML*, permitiendo su combinación y ofreciendo una interfaz estándar para el acceso y manipulación.

Ficheros: Directorios. Agrupación de archivos de datos, atendiendo a su contenido, a su propósito o a cualquier otro criterio.

Herramientas CASE: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería.

HTML: Lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

JavaScript: Lenguaje de programación interpretado, utilizado principalmente en páginas Web.

Navegador: Aplicación capaz de interpretar las órdenes recibidas en forma de código *HTML* fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden, permitiendo así la comunicación con el servidor.

Open Source: Literalmente, código abierto. Hace referencia al *software* libre, escrito bajo la Licencia GPL que permite ver, modificar y distribuir el código fuente.

ORM (*Object Relational Mapping*): en español, Mapeo Relacional de Objetos, técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

Radio: Tecnología que permite la transmisión de señales sonoras mediante la modulación de ondas electromagnéticas.

Servidor: *Software* u ordenador que provee servicios a otros programas o equipos denominados clientes.

Televisión: Medio de comunicación que combina los mensajes de imágenes fijas y en movimiento con voz, música, efectos sonoros y especiales.

TIC: Tecnologías de la información y las comunicaciones.

Transmisión Abierta: Señal de libre acceso por la cual no es necesario pagar para consumirla.

Transmisión: Emitir la señal de radio-televisión desde una estación emisora a otra receptora.

UCI: Universidad de las Ciencias Informáticas.

UML (*Unified Modeling Language*): Notación estándar para modelar objetos del mundo.

VLCmedia player: Reproductor multimedia del proyecto VideoLAN. *Software* libre distribuido bajo la licencia GPL. Soporta muchos *códecs* de audio y video, así como diferentes tipos de archivos, además de *DVD*, *VCD*.

W3C *(Consorcio World Wide Web):* Consorcio internacional que produce estándares para la *World Wide Web*. Está dirigida por Tim Berners Lee, creador original de *URL* (Uniform Resource Locator, localizador uniforme de recursos), *HTTP* (HyperText Transfer Pro*tocol*, protocolo de transferencia de hipertexto) y *HTML* (lenguaje de marcado de hipertexto) que son las principales tecnologías sobre las que se basa la Web*.*

Web: Servidor de información www. Se utiliza también para definir el universo www en su conjunto.

WWW (World Wide Web*):* Sistema de información distribuido con mecanismos de hipertexto. Es el universo de servidores *HTTP*, que permiten mezclar texto, gráficos y archivos de sonido juntos.

XML (*Extensible Markup Language*): Conocido como lenguajes de marcas. Es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (*W3C*), no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XMLHttpRequest: Interfaz empleada para realizar peticiones *HTTP* y *HTTPS* a servidores Web. Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, *XML*, *HTML* y codificaciones particulares específicas.

XSLT (*Extensible Stylesheet Language Transformations*): Estándar de la organización *W3C* que presenta una forma de transformar documentos *XML* en otros e incluso a formatos que no son *XML*.