Arquitectura de la Plataforma de Transmisión Abierta para Radio y Televisión

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Dayami Chávez Ayala

Tutor:

Ing. Yoandri Quintana Rondón

Ciudad de La Habana, mayo de 2010.
"Año del 51 aniversario del triunfo de la Revolución"

“El futuro pertenece a quienes creen en la belleza de sus sueños”

Eleanor Roosevelt

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2010.

_____ _____

Firma del Autor Firma del Tutor

Diplomante

Nombre y apellidos: Dayami Chávez Ayala

Sexo: F **Grupo:** 9505 **Correo electrónico:** dchavez@estudiantes.uci.cu

Tutor

Nombre y apellidos: Yoandry Quintana Rondón

Sexo: M **Institución:** Universidad de las Ciencias Informáticas (UCI)

Dirección de la institución: Carretera a San Antonio de los Baños, Km. 2 ½, Reparto: Torrens, Municipio: Boyeros, Provincia: Ciudad de La Habana.

Teléfono del trabajo: 07-835-8284 **Cargo del trabajador:** Profesor

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Año de graduación: 2009 **Correo electrónico:** yqrondon@uci.cu

Institución donde se graduó: UCI

Oponente

Nombre y apellidos: Angel Dayán Marín Abreu

Sexo: M **Institución:** UCI

Teléfono del trabajo: 07-835-8284 **Cargo del trabajador:** Profesor

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Año de graduación: 2009 **Correo electrónico**: admarin@uci.cu

Institución donde se graduó: UCI**.**

A mis padres Idolina y Rafael, por la vida, el amor, el ejemplo, por confiar en mí y darme fuerzas para seguir adelante.

A Reinier, por haberme dado tanto cariño y amor en estos años de carrera.

A mi tutor Yoandry, por su ayuda y consejos a pesar de ser tan joven.

A Tata y a toda mi familia, porque imaginarla orgullosa de mí me impulsó a lograr esta meta.

A mis amigos de la vocacional David, Yano y mi querida Daylín, por ellos estoy hoy aquí.

A los amigos de la universidad quienes han estado siempre en las buenas, en las malas y de los que más he aprendido: Ale, Frank, Ángel, Yordani, Pimi, Maggie.

A mis amigas Mailín, Olga, Yailet, Annies, por las risas, las ocurrencias y los años compartidos.

A mi tía Eumelia, a Ailín y Ailec por ayudarme y quererme tanto en estos años.

A mis compañeros de proyecto, por lo mucho q he aprendido con ellos: Dunier, Jorgito, Yumar, a Pedro que tantas veces levanté de la PC.

A muchos amigos más que hice en todos estos años: Arlen, Pepito, el Mello, Zule, Yake, Pedri.

A los que ya no están cerca, pero me hicieron feliz: Lorena, Yanet, Yaricel, Glendys, Alexis, Elio.

A mis amigos de Mantua a pesar de la distancia siempre me han apoyado: Dhoa, Lienna, AnaT, Greysis, Henry.

A mi tribunal por sus oportunas críticas, a Ronald.

A todos los que han confiado en mí…..gracias.

A mi mamá y mi papá por tanto amor y entrega, por inspirarme con tantos ejemplos de genialidad.
"Que adorno más grande puede haber para un hijo que la gloria de un padre, o para un padre que la conducta honrosa de un hijo."

A mis sobrinitos Adita, Adiel y Katerine, ahora comienzan ustedes.

A los que no pudieron verme realizada, mis abuelos y mi adorada Katia.

A mis amigos que me apoyaron en todo momento, a los distantes, a los cercanos, a los de siempre.

En los últimos años el avance de las Tecnologías de la Información y las Comunicaciones (TIC) han despertado un gran interés en las comunidades de desarrollo del mundo donde Cuba no se encuentra ausente. Las necesidades actuales que tiene toda organización para el logro de sus objetivos, demandan la construcción de grandes y complejos sistemas de software que requieren de la combinación de diferentes tecnologías y plataformas de hardware y software para alcanzar un funcionamiento acorde con dichas necesidades. Lo anterior, exige de los profesionales dedicados al desarrollo de software poner especial atención y cuidado al diseño de la arquitectura, bajo la cual estará soportado el funcionamiento de sus sistemas.

Hoy día muchos sistemas fracasan o no alcanzan el total de los requerimientos establecidos porque muchas veces la arquitectura de software se encuentra deficiente en su concepto o diseño, o quizás no cuentan con la arquitectura del software adecuada para el sistema que se desea desarrollar.

El presente trabajo pretende conformar la propuesta de una arquitectura de software robusta y confiable que permita el desarrollo de un sistema modificable, interoperable, seguro, funcional y disponible, que asegure el mantenimiento, flexibilidad y reusabilidad de la Plataforma de Transmisión Abierta para Radio y Televisión, apoyándose en el uso de herramientas libres.

Palabras claves: plataforma, transmisión, streaming, arquitectura.

In recent years, the advancement of Information and Communications Technologies (ICT) have attracted great interest in the communities of the developing world where Cuba is not absent. The current needs that have all organization to achieve its objectives require the construction of large and complex software systems. Those systems involve the combination of different technologies and hardware and software platforms to achieve a performance in line with these requirements.

This work require from the professionals that are dedicated to software development and to pay special attention to the design of architecture, which will be supported under the functioning of their systems. Today, many systems fail or don't reach all requirements established because most of the time the software architecture is deficient in its concept or design, or may not have the appropriate software architecture for the system to be developed.

The present work tries to shape the proposal of a robust and reliable software architecture that allows the development of a modifiable, interoperable, secure, functional, and available system that ensure the maintenance, flexibility and reusability from the Open Transmission Platform for Radio and Television, based on the use of free tools.

Keywords: platform, broadcasting, architecture.

Índice

Introducción 1
Capítulo 1 5
1.1. Introducción 5
1.2. ¿Qué es una plataforma de transmisión? 5
1.3. Características del sistema 6
1.4. Sistemas similares a la Plataforma de Transmisión Abierta para Radio y Televisión 6
1.4.1. Inter-nos 7
1.4.2. Yahoo.com 7
1.4.3. Tv.com 8
1.4.4. YouTube.com 8
1.5. Arquitectura de software 8
1.6. Definición de Arquitectura de Software 9
1.7. Estilos y Patrones Arquitectónico 9
1.7.1. Estilo Arquitectónico 9
1.7.2. Patrones Arquitectónicos 11
1.8. Conclusiones 14
Capítulo 2 15
2.1 Introducción 15
2.2 Metodología de desarrollo de software 15
2.2.1 Rational Unified Process RUP 15
2.3 Lenguaje Unificado de Modelado 16
2.4 Herramientas CASE 17
2.4.1 Visual Paradigm 17
2.5 Tendencias y tecnologías actuales 18
2.5.1 Modelo Cliente Servidor 18
2.5.2 Aplicaciones Web 19
2.5.3 AJAX 19
2.5.4 Streaming de Audio y Video 20
2.6 Lenguaje de Programación 21
2.6.1 Lenguajes de programación del lado del cliente 22
2.7 Lenguajes de programación del lado del servidor 23
2.8 Framework 24
2.8.1 Symfony 24
2.8.2 Hibernate 25
2.9 Entorno de Desarrollo Integrado (IDE) 26
2.9.1 NetBeans 26
2.10 Sistemas Gestores de Bases de Datos 26
2.10.1 PostgreSQL 27
2.11 Sistemas de Streaming 27
2.11.1 VideoLan 28
2.11.2 Darwin Streaming Server 29
2.12 Servidor Web 29
2.12.1 Apache 30

2.13 Sistema Operativo 30
2.13.1 UBUNTU Linux 30
2.14 Software para el control de versiones (CVS) 31
2.14.1 Subversion con cliente RapidSVN 31
2.15 Salvas Automáticas 31
2.15.1 Bacula 31
Capítulo 3 33
3.1 Introducción 33
3.2 Estructura del equipo de desarrollo 33
3.3 Subsistemas de la aplicación 33
3.4 Descripción de la arquitectura 35
3.4.1 Vista de casos de uso 36
3.4.2 Vista Lógica 41
3.4.3 Vista de Procesos 45
3.4.4 Vista de Implementación 45
3.4.5 Vista de Despliegue 47
3.5 Requerimientos no funcionales del sistema 48
3.6 Conclusiones 52
Capítulo 4 53
4.1 Introducción 53
4.2 Propósito de evaluar una arquitectura de software 53
4.3 Evaluando la arquitectura de software 53
4.4 Resultados que se obtienen al evaluar una arquitectura de software 54
4.5 Atributos por los cuales puede ser evaluada la arquitectura de software 54
4.6 Principales métodos de prueba de arquitectura de software 56
4.6.1 ATAM 57
4.6.2 ARID 57
4.7 Propuesta de estrategia de evaluación de la arquitectura 58
4.7.1 Estrategia de evaluación temprana (ARID) 58
4.7.2 Estrategia de evaluación tardía (ATAM) 61
4.8 Valoración de propuestas arquitectónicas similares 63
4.9 Conclusiones 66
Conclusiones Generales 67
Recomendaciones 68
Trabajos citados 69

Introducción

Hoy en día, los medios de comunicación constituyen una herramienta persuasiva que permite a las personas mantenerse en continua interacción con los distintos sucesos políticos, sociales y económicos tanto a escala nacional como internacional.

En las sociedades contemporáneas, es cada vez mayor la importancia de los medios masivos y en particular de la radio y la televisión, pues logran modificar la forma en que los hombres conocen y comprenden la realidad que los rodea influyendo aceleradamente sobre la forma de actuar o de pensar de estos.

No ajena a esta tecnología de avanzada que tiene sus orígenes a finales del siglo XIX, el 24 de octubre de 1950 en Cuba, la Unión Radio Televisión transmite por el canal 4 la primera señal de televisión comercial, siendo de este modo, uno de los primeros países en América después de México y Brasil en proyectar por primera vez la televisión.

Así, a lo largo de las venideras décadas, se suscitaban en Cuba cambios y adelantos en esta rama de las comunicaciones, hoy día no se puede hablar de cultura, educación, ni de desarrollo si no se habla de las nuevas Tecnologías de la Información y las Comunicaciones (TIC), y es por ello que se ha promovido su uso masivo, se han creado salas de video y clubes de computación en todos los sectores de la sociedad con el motivo de aumentar el uso de las tecnologías, así como la creación el 22 de septiembre de 2002 de la Universidad de las Ciencias Informáticas (UCI), actualmente uno de los pilares de mayor importancia para la economía del país en la industria del software.

Enmarcado en la importancia de las nuevas TIC, desde un principio fueron creados en la universidad varios canales de televisión con fines tanto educativos, culturales, informativos así como de entretenimiento. Este proceso es controlado por la Dirección de Televisión Universitaria (DTU) y apoyado por el Departamento de Señales Digitales de la Facultad 9, una de las facultades con que cuenta esta universidad. En uno de los proyectos de este centro se desarrolla la Plataforma de Transmisión Abierta para Radio y Televisión (PTARTV) con el fin de integrar y organizar todos los procesos que se realizan en dicha DTU. Algunos de estos procesos son: la transmisión manual de ficheros de video, lo que provoca en ocasiones errores humanos debido a la necesidad de mantener a una persona generando estas transmisiones en el propio momento de dicha actividad, además la codificación del fichero y su transferencia al lugar de transmisión, es un proceso que requiere de una

persona que utilice algún sistema conversor de medias, y luego realice la mayor parte del trabajo de forma manual, lo cual puede inducir problemas que se traducen en horas perdidas.

Un primer análisis arrojó que no todos los sistemas logran el éxito y una de las causas para su fracaso es el mal diseño de la arquitectura pues su desarrollo es una de las etapas fundamentales y, en muchos casos, la más importante en el desarrollo de software, pues es aquí donde se aportan todos los conocimientos, creatividad y experiencia para crear la mejor propuesta de solución que se le dará al cliente, que cumpla con los requerimientos establecidos para el sistema en desarrollo, así como sus preocupaciones principales de lo que esperan del sistema. Según estudios realizados para lograr el desarrollo de la PTARTV se hace necesario diseñar una arquitectura de software robusta que cumpla con todas las necesidades para lograr una integración uniforme del mismo y garantizar que el producto resultante tenga la calidad requerida.

De la situación anterior se identificó el **problema científico**: ¿Qué arquitectura de software utilizar para el desarrollo de la Plataforma de Transmisión Abierta para Radio y Televisión sobre software libre para lograr que la misma sea modificable, reutilizable, flexible y robusta?

La investigación se enmarca en el **objeto de estudio**: Las arquitecturas de software usadas para el desarrollo de sistemas informáticos para los medios de comunicación en radio y televisión y el **campo de acción:** La arquitectura de software para la Plataforma de Transmisión Abierta para Radio y Televisión.

Como **idea a defender** se tiene que el diseño de una arquitectura de software robusta, que cumpla con los requerimientos y atributos de calidad establecidos para la Plataforma de Transmisión Abierta para Radio y Televisión garantizará un esquema para la realización del producto.

Para dar solución al problema planteado se define como **objetivo general** diseñar una arquitectura de software para la Plataforma de Transmisión Abierta para Radio y Televisión que permita que el sistema sea modificable, interoperable, seguro, funcional y disponible, apoyándose en el uso de herramientas libres.

Para lograr este objetivo se desarrollan las siguientes **tareas de la investigación**:

1. Caracterizar diferentes sistemas que permitan la transmisión abierta de radio y televisión.
2. Valorar el estado del arte de varios estilos y patrones de la arquitectura de software y tomar posición en cuanto al más idóneo para este tipo de aplicación.

3. Identificar las herramientas y tecnologías a utilizar para el desarrollo.
4. Plantear la propuesta arquitectónica para la Plataforma de Transmisión abierta para Radio y Televisión.
5. Diseñar la estrategia de prueba para la arquitectura de software seleccionada.
6. Comparar al menos con 2 casos donde se hayan utilizado propuestas arquitectónicas en sistemas similares para la transmisión abierta para Radio y Televisión.

Para la realización de estas tareas se emplearon diferentes **métodos de la investigación:**

Métodos Teóricos

- "Histórico - Lógico": este método permite determinar las características esenciales de los principales estilos arquitectónicos existentes y realizar el estudio de la trayectoria real de determinados elementos que servirán de guía para la construcción de la arquitectura de software conveniente para el desarrollo de software que se requiere.
- "Analítico - Sintético": permite sintetizar la información necesaria para el desarrollo de la plataforma que se pretende, ya sea de los diferentes sistemas que permitan la transmisión abierta de radio y televisión o los estilos arquitectónicos. Lo que posibilita la comprensión de la misma, con ideas claras y concisas.
- "Modelación": en el desarrollo de la investigación es necesaria la modelación de la arquitectura que se propone, es por eso que se utiliza este método para crear modelos con vistas que favorezcan la comprensión de la misma.

Métodos Empíricos

- "Observación": Este método facilita conocer el panorama real de una situación mediante la percepción directa, con su utilización se determinaron los rasgos imprescindibles en el desarrollo de la arquitectura para la Plataforma de Transmisión Abierta para Radio y Televisión.
- "Entrevista": Se realizaron dos entrevistas, la primera con el objetivo de conocer las principales tendencias de la arquitectura de software, patrones y estilos arquitectónicos; la segunda entrevista se hizo para comprender el funcionamiento de la DTU, que es la encargada de brindar servicios de audio y video por la web en la UCI. (Anexo 2)

El documento está constituido por cuatro capítulos:

Capítulo 1. Fundamentación Teórica

En este capítulo se abordan los elementos relacionados con el estudio del estado del arte y la fundamentación teórica. Se realiza un estudio de las arquitecturas más comunes para determinar cuál es la adecuada para desarrollar el proyecto antes de iniciar el trabajo, apoyando la investigación en el estudio de las plataformas de transmisión abierta para radio y televisión existentes y en las características fundamentales del sistema.

Capítulo 2. Herramientas para la solución.

Se dispone de un análisis sobre las principales herramientas, metodologías de desarrollo, lenguajes de programación, ambientes de desarrollo entre otras definiendo finalmente cuáles son las apropiadas para el desarrollo de la plataforma.

Capítulo 3. Propuesta de la arquitectura del software.

En esta sección se muestra la propuesta de la arquitectura para la Plataforma de Transmisión Abierta para Radio y Televisión basada en el estudio realizado en los capítulos anteriores.

Capítulo 4. Evaluación de la arquitectura del sistema.

En este capítulo se realiza la evaluación de la arquitectura desarrollada mediante el diseño de dos estrategias de pruebas. Además se realiza una comparación con dos sistemas de similar propósito.

Capítulo 1
Fundamentación Teórica

1.1. Introducción

En este capítulo se realizará un análisis de los sistemas similares a la plataforma, analizando sus características y funcionalidades. Se abordan temas relacionados con la arquitectura de software, estilos y patrones arquitectónicos, principales características y cuáles de estas se ajustan a las necesidades de la plataforma.

1.2. ¿Qué es una plataforma de transmisión?

En Telecomunicación, una plataforma es un conjunto de elementos interconectados que se utilizan para transmitir una señal de un lugar a otro. La señal transmitida puede ser eléctrica, óptica o de radiofrecuencia. (1)

En Informática, una plataforma es precisamente el basamento, ya sea de hardware o software, sobre el cual pueden realizarse varias tareas. (2)

Se define como plataforma de transmisión, al conjunto de elementos interconectados a partir de la base de intercambios y de conectividad, gestionado por ciertas reglas, ciertas capacidades y ciertas limitaciones en un ambiente determinado, permitiendo la realización de numerosas tareas referentes a la acción de transmitir contenidos por un medio; integra la posibilidad de controlar el proceso de transmisión.

La PTARTV será capaz de difundir las informaciones mediante un sitio web o de un televisor, haciendo uso de las nuevas tecnologías, logrando que se pueda acceder al contenido audiovisual que pudiera emitirse por estos medios en todo momento. Generalmente estas informaciones vienen asociadas a imágenes, sonidos y videos, normalmente procedentes de un repositorio de medias o de una transmisión en vivo.

Con la implementación de esta aplicación se pretende dar solución a todos los problemas existentes en la DTU, mediante funcionalidades que faciliten la programación, la gestión de las medias, la transmisión, la transferencia y el entorno web de la aplicación.

1.3. Características del sistema

El sistema informático PTARTV será una aplicación mixta pues poseerá subsistemas web y de escritorio, estará estructurado en ocho subsistemas que se relacionan entre sí y actúan como un todo para brindar un resultado final eficiente y acorde a las necesidades de los usuarios.

Este sistema debe ser seguro y funcional. Para ello debe permitir:

- Gestionar los canales de la plataforma y la programación de los mismos.
- Interrupción y reanudación de la transmisión.
- Monitorear los canales en transmisión.
- Conversión de las medias a un formato especificado.
- Chequeo de la integridad de los materiales convertidos.
- Transmitir las medias acorde a la programación del canal.
- Conmutar las señales entrantes.
- Transcodificar las medias previstas para publicación una vez transmitidas.

Para lograr con éxito todas las funcionalidades antes mencionadas se debe tener en cuenta que el sistema:

- Debe ser desarrollado con herramientas libres.
- Utilizar un sistema gestor de base de datos lo más seguro posible.
- Dar respuesta rápida ante cada uno de los procesos que debe brindar el sistema.
- Las herramientas seleccionadas deben ser factibles para cualquier plataforma.

1.4. Sistemas similares a la Plataforma de Transmisión Abierta para Radio y Televisión

En el ámbito internacional y nacional existen varias soluciones que ofrecen un recurso parcial para integrar varios servicios de radio y televisión tanto de video en vivo (live-video) [1]como de video bajo

[1]Se pueden brindar servicios similares a la televisión, el usuario solo puede ver la información que se está transmitiendo en el instante que accede.

demanda[2], ejemplos que se analizan son los sitios web que ofrecen contenido de audio y video, además de los servicios comunes como son los foros de discusión, encuestas digitales y chat. Los sitios internacionales youtube.com, tv.com, yahoo.com, el sitio de distribución de contenidos multimedia de la UCI nombrado inter-nos, y el sistema de televisión de la propia universidad, ofrecen soluciones parciales a los servicios que se desean integrar. A continuación se exponen y analizan las características de estos sistemas para una mejor comprensión del tema.

1.4.1. Inter-nos

El sitio Inter-nos (Anexo 3) tiene como función principal contribuir con el proceso docente educativo en la UCI, publicando teleclases, materiales educativos, políticos y de apoyo a la docencia. Integra servicios de audio y video por la web usando como reproductor embebido al reproductor de Windows Media. De manera complementaria publica películas, seriales televisivos, documentales, programas de radio, festivales de artistas aficionados, la transmisión de cuatro canales de televisión y siete de radio, todos ellos en vivo.

Usa tecnología de streaming para la transmisión de los archivos multimedia. Para este fin se emplea *Windows Media Server (WMS)*, tecnología de Microsoft para proporcionar el servidor de streaming, el formato de video que utiliza es *Windows Media Video* (WMV). Está implementado usando el lenguaje de programación ASP, como servidor web *Microsoft Internet Information Services* (IIS), y el sistema gestor de base de datos Microsoft SQL Server 2000. El proceso de publicación de media se realiza de forma manual, desde el sitio se habilita el vínculo, pero se debe copiar el fichero a la carpeta indicada para que el servidor logre conformar el streaming.

Las modificaciones al sitio se realizan directamente al código, y debido a esto suele ser un proceso complicado, pues no cuenta con un sistema de gestión de informaciones. El diseño de la página es sencillo, sin muchas imágenes o animaciones por lo que es de rápida navegabilidad. En ocasiones se ralentiza la transmisión del streaming pues este sitio cuenta con gran audiencia.

1.4.2. Yahoo.com

El sitio yahoo.com (Anexo 4) hace una integración de los servicios de audio-video a demanda y en vivo, posibilitando a los clientes la visualización y escucha de situaciones que están ocurriendo en el

[2] Similar a cuando se usa una video casetera, permite avanzar, pausar, aumentar la velocidad, es decir, interactuar con el flujo de datos recibido.

propio instante de tiempo que se conectan, además ofrece la posibilidad de ver o escuchar programas grabados alojados en el propio portal digital. Este sitio posibilita la recepción de transmisiones de radio y televisión de carácter deportivo. Su principal atracción viene dada por la inmediatez de la noticia al hacerla llegar con un retardo de tan solo 0,25 minutos.

1.4.3. Tv.com

El portal tv.com (Anexo 5) ofrece servicios interactivos que posibilita al cliente su influencia en la programación televisiva que se transmite por los canales convencionales. El usuario vota por sus programas favoritos y de esta forma se puede obtener la demanda de los clientes de cierto género audiovisual o programa en particular. Este sitio además brinda varios espacios a la divulgación e información de películas, seriales televisivos y actores. Este sitio ofrece la posibilidad de reproducir videos desde la propia página mediante un reproductor flash.

1.4.4. YouTube.com

El portal YouTube.com (Anexo 6) contiene gran cantidad de videos publicados de diversas categorías, ofreciendo la posibilidad de verlos mediante un reproductor flash. Realizado sobre herramientas libres ha logrado gran eficiencia en sus servicios, para ello utiliza herramientas como el servidor web Apache, lenguaje de programación Python, sistema operativo GNU/Linux (Suse), gestor de base de datos MySQL, para transmitir los videos utiliza un servidor web ligero como es el Lighttpd y los videos para celulares lo transmite usando *Darwin Streaming Server*. Permite subir videos y compartirlos para que otros puedan acceder al mismo, esta funcionalidad es clave en su popularidad. Cada video tiene un ranking de calidad, donde los usuarios pueden elegir y valorar el video, esto da la posibilidad de tener una visión de su nivel de popularidad. Posee un buscador de videos, lo que facilita encontrar el deseado, pues a pesar de estar categorizados, son muchos y cada día se añaden nuevos, siendo difícil de otra manera acceder al contenido que se quiere. Posee un diseño sencillo y agradable, no cuenta con gran cantidad de imágenes o animaciones extras y es de fácil navegación.

1.5. Arquitectura de software

Las diferentes instituciones y organizaciones demandan cada vez más de productos de mediana y gran envergadura, estos precisan diferentes tecnologías y plataformas de hardware y software para alcanzar el funcionamiento deseado. La calidad de estos productos es indispensable, por lo que los sistemas deben ser modificables, interoperables, seguros, funcionales y disponibles, asegurando el

mantenimiento, flexibilidad y reusabilidad. Para lograr estos requisitos es ineludible el diseño de una arquitectura de software robusta y confiable que se adapte a las necesidades de cada software.

1.6. Definición de Arquitectura de Software

La definición de Arquitectura de Software (AS) no es única, existen varias definiciones y en algunas ocasiones suelen resultar polémicas. Para comprender mejor este argumento es necesario conocer el criterio de algunos de los principales expertos del mismo.

La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. (3)

Según la IEEE std. 14_71-2000: la Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (4)

Como se puede apreciar estos conceptos tienen algunos puntos en común pero no consolidan una única idea. Cada autor expresa su criterio y brinda su propia definición. Simplemente, la AS es el pilar fundamental en el desarrollo de las aplicaciones y define cuales son las principales características del software, con cierto nivel de abstracción.

La arquitectura de software representa los principales componentes que facilitan el funcionamiento de las aplicaciones. Incluye las descripciones de estos componentes, lo que permite comprender el objetivo y la funcionalidad de cada uno de ellos. La AS es clave en el desarrollo de los sistemas informáticos, constituye un basamento y guía para alcanzar el éxito.

1.7. Estilos y Patrones Arquitectónico

Dentro del marco de la AS es muy habitual escuchar los términos de estilos y patrones arquitectónicos. A continuación se brindan definiciones, puntos de vistas y característica relacionados a los mismos.

1.7.1. Estilo Arquitectónico

Se define un estilo arquitectónico como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. (5)

En Arquitectura de software en la práctica (*Software Architecture in Practice)* (3) se proporcionan cinco grupos de estilos:

1. Flujo de datos(El sistema es visto como una serie de transformaciones sobre piezas sucesivas de datos de entrada. El dato ingresa en el sistema, y fluye entre los componentes, de uno en uno, hasta que se le asigne un destino final).
    - Proceso secuencial por lotes.
    - Red de flujo de datos.
    - Tubería-filtros.
2. Llamado y retorno (estilo dominado por orden de computación, usualmente con un solo hilo de control, un programa principal controla varios subprogramas que se comunican mediante el uso de llamadas).
    - Programa principal / Subrutinas.
    - Tipos de dato abstracto.
    - Objetos.
    - Cliente-servidor basado en llamadas.
    - Sistemas en capas.
    - Modelo Vista Controlador.
3. Componentes independientes (dominado por patrones de comunicación entre procesos).
    - Independientes, casi siempre concurrentes.
    - Sistemas orientados por eventos.
    - Procesos de comunicación.
4. Centrados en datos (Se caracteriza por un almacenamiento central complejo, manipulado por sistemas independientes).
    - Repositorio.
    - Pizarra.

5. Máquina virtual (caracterizado por la traducción de una instrucción en otra, simulando funcionalidades que no son nativas al hardware o software sobre el que está implementado).
    - Intérprete.
    - Sistema basado en regla.

Se puede considerar un estilo arquitectónico como un conjunto coordinado de restricciones arquitectónicas que restringe los rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo. Cuando se habla de estilo arquitectónico se deben tener presente las famosas cuatro C, componentes, conectores, configuración y c*onstraint* (restricciones).

1.7.2. Patrones Arquitectónicos

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones.

Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que puedes usar esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces. (6)

1.7.2.1. Patrones arquitectónicos más comunes

Arquitectura en capas

La arquitectura en capas consiste en aislar la lógica de la aplicación y en convertirla en una capa intermedia bien definida. En la capa de presentación se realiza relativamente poco procesamiento de la aplicación; las ventanas envían a la capa intermedia peticiones de trabajo. Y esta se comunica con la capa de almacenamiento del extremo posterior. (7)

El principal objetivo del patrón en capas es dividir, fraccionar y llegar a separar la Presentación (donde muestras y obtienes datos), de la Lógica de Negocio (donde se realizan operaciones) y con todo esto se obtendría independencia, así, si se realizan cambios de arquitectura se puede acceder a los datos o cambiar el negocio o modificar la presentación y solo sería en esa porción de la capa como se muestra en la Figura 2.

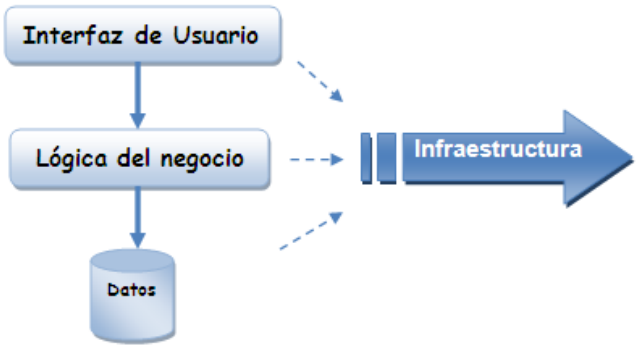

Figura 1: Arquitectura en capas.

Modelo Vista Controlador (MVC)

Este patrón (Figura 3) se observa frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo son los datos provenientes de la lógica de negocio recogidos en el controlador y el controlador invoca a la lógica de negocio, utilizando los objetos de negocio. La vista y el controlador constituyen la interfaz del usuario.

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Responde a eventos, usualmente acciones del usuario que puede invocar cambios en el modelo y probablemente en la vista.

Algunas arquitecturas son más recomendables implementar con ciertas tecnologías mientras que hay otras que no son compatibles entre sí, en este caso se utilizará el framework de PHP *Symfony* el cual implementa una arquitectura MVC.

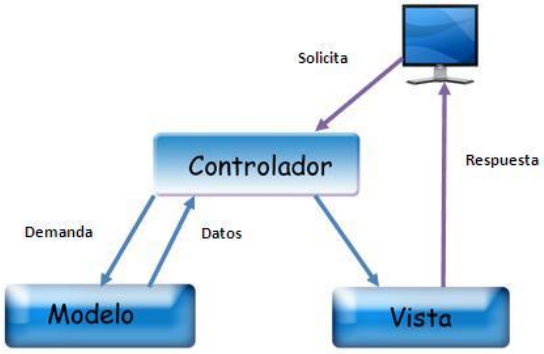

Figura 2: Modelo Vista Controlador (MVC)

Arquitectura Orientada a Servicios (SOA)

La Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante servicios web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular. (8)

Esta arquitectura no es algo que pueda estar al alcance de todas las empresas. En primer lugar, únicamente tiene sentido en caso de que el tamaño del departamento de tecnologías de la información sea considerable y esté formado, al menos, por más de un sistema primario. Es decir, SOA aporta su mayor beneficio dentro de sistemas complejos y heterogéneos, por eso, para pequeñas empresas es difícil que pueda aportar un valor directo. Dentro de las grandes corporaciones, con sistemas tecnológicos complejos y heterogéneos, SOA no va a proporcionar los mismos beneficios a cada una de ellas.

Arquitecturas multicapas orientadas a objetos

Una arquitectura multicapas que se adecue a los sistemas de información orientados a objetos, incluye la división de las responsabilidades que se encuentran en la arquitectura clásica de tres capas. Las responsabilidades se asignan a los objetos de software. (7)

En las arquitecturas multicapas orientadas a objetos los componentes del estilo se basan en principios orientados a objetos donde se puede realizar encapsulamiento, herencia y polimorfismo. Las unidades de modelado, diseño e implementación, los objetos y sus interacciones son el centro de las

responsabilidades en el diseño de la arquitectura y en la estructura de la aplicación. En cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases. En tantos componentes, los objetos interactúan a través de invocaciones de funciones y procedimientos.

Entre los términos de estilo y patrón arquitectónico se puede apreciar que existe gran relación pero la principal diferencia radica en el nivel de abstracción. Los estilos presentan un nivel de abstracción muy alto, sin especificar detalles, mientras que los patrones se adaptan a problemas específicos, dando una solución obtenida por la experiencia en el tema.

1.8. Conclusiones

Se expusieron conceptos relacionados a los procesos asociados a la PTARTV los cuales resultaron fundamentales para una mejor comprensión de los aspectos tratados en la presente investigación.

Se analizaron las funcionalidades de plataformas que brindan servicios de audio y video a través de la web, entre ellas, es.youtube.com, tv.com, yahoo.com de carácter internacional e Inter-nos en Cuba, y se tomó de ellas las mejores experiencias para el sistema que se desea desarrollar.

Se abordaron temas relacionados a la arquitectura de software, analizando características y ventajas de los principales estilos arquitectónicos como flujo de datos, llamado y retorno, componentes independientes, centrados en datos y máquina virtual; y patrones arquitectónicos como, Arquitectura SOA, Arquitectura en capas, Arquitectura Orientada a Objetos y Modelo Vista Controlador.

Con el estudio desarrollado en este capítulo se seleccionaron los patrones Arquitectura Orientada a Objetos y Modelo Vista Controlador por las ventajas que presentan para aplicarlos en el sistema y plantear así una mejor propuesta arquitectónica.

Capítulo 2
Herramientas para la Solución

2.1 Introducción

En el presente capítulo se abordarán temas relacionados con las tecnologías actuales más adecuadas para el desarrollo de la PTARTV. Aquí se analizan la metodología de desarrollo, la herramienta CASE, el entorno de desarrollo (IDE), lenguaje de programación, el gestor de bases de datos y otras herramientas necesarias para lograr la evolución del sistema.

2.2 Metodología de desarrollo de software

Una pregunta fundamental en el desarrollo de cualquier software es ¿qué metodología utilizar? pues todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, no se obtiene el resultado deseado y el cliente no queda satisfecho. A continuación se expondrán las principales características de dos metodologías de desarrollo, *Rational Unifield Process* y *Extreme Programming,* sin dejar de mencionar antes otras muy importantes como Scrum y Fdd.

2.2.1 Rational Unified Process RUP

Como metodología, RUP es un proceso que define claramente quien, cómo, cuándo y qué debe hacerse. Éste aporta herramientas como los casos de uso, que definen los requerimientos. Permite la ejecución iterativa del proyecto y del control de riesgos.

RUP es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos. (9)

Descripción general

- El RUP es un proceso de ingeniería de software.
- Utiliza el paradigma de orientación a objetos para su descripción.
- Es un marco de proceso configurable para satisfacer necesidades específicas.
- Implementa las mejores prácticas de desarrollo de software.

Características de RUP

1. Guiado por casos de uso: Los casos de uso capturan requerimientos funcionales y representan piezas de funcionalidad que brindan un resultado de valor al usuario.
2. Centrado en la Arquitectura: Comprende los aspectos estáticos y dinámicos más importantes del sistema.
3. Iterativo e Incremental: El trabajo se divide en piezas pequeñas o mini proyectos; cada uno proveyendo un subproducto incremental.

RUP la seleccionada pues es muy utilizada para la documentación y realización de software orientados a objetos. El equipo de desarrollo cuenta con experiencia en el uso de esta metodología, aumentando la rapidez del desarrollo del producto. RUP propone varias iteraciones en el proceso de desarrollo de software posibilitando la corrección de errores y mejoras del software a medida que avanza el proyecto, lo que permite a los desarrolladores poder realizar un producto con gran calidad. Brinda una fuerte y abundante documentación, permitiendo un futuro entendimiento de cada componente del producto. Esto es fundamental para la PTARTV pues con esta metodología el sistema se va desarrollando y documentando al mismo tiempo por si algún miembro del equipo no puede seguir con el trabajo el que ocupe su lugar tenga por donde guiarse y conozca qué fue lo que se realizó hasta la fecha.

El enfoque RUP está basado en modelos y utiliza un lenguaje bien definido para tal fin, Lenguaje Unificado de Modelado (UML), por lo que se hace necesario detallar algunas caracterizas de este.

2.3 Lenguaje Unificado de Modelado

UML es un lenguaje para visualizar, especificar, construir, y documentar los artefactos que se crean durante el proceso de desarrollo. Es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Es un lenguaje gráfico, que puede ser usado en todas las fases de desarrollo de software y que permite representar los sistemas con varios modelos parciales, lo que facilita su entendimiento y la comunicación.

La representación en UML de un software está formada por las 4+1 vistas o modelos parciales separados, relacionados entre sí, estas vistas son:

- ✓ Vista de casos de uso.

- ✓ Vista lógica.
- ✓ Vista de procesos.
- ✓ Viste de implementación.
- ✓ Vista de despliegue.

Numerosos autores definen que UML no es, en sí, un lenguaje de descripción arquitectónica pues su forma de expresar ciertas características, sobre todo dinámicas de las estructuras no es suficiente para los arquitectos. (10)
Para realizar la modelación con este lenguaje se utilizan herramientas que se han desarrollado propiamente para ello, estas reciben el nombre de herramientas CASE.

2.4 Herramientas CASE

Las herramientas (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering o* Ingeniería de Software Asistida por Ordenador) CASE son numerosas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

2.4.1 Visual Paradigm

El Visual Paradigm es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto, constituye una herramienta profesional para el modelado UML que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. Posibilita el modelado de base de datos, requerimientos, proceso de negocio, permite realizar todo tipo de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm es la herramienta seleccionada pues emplea UML para el modelado, es la herramienta por excelencia para ser utilizada en un ambiente de software libre. Permite crear tipos diferentes de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar.

Además otra de las ventajas que posee el Visual Paradigm es que se adecua al entorno de desarrollo permitiéndoles a los diseñadores, analistas y a todo aquel que trabaje con el mismo, una mayor organización y claridad en el trabajo. Esta herramienta es bastante común y muy utilizada a nivel mundial por la variedad de funciones que permite realizar, lo que posibilita su utilización para la evolución de PTARTV.

2.5 Tendencias y tecnologías actuales

Se vive hoy en día una revolución informática formidable, los sistemas para computadoras se han orientado en gran magnitud al desarrollo de aplicaciones web. Teniendo en cuenta las funcionalidades que brindan tecnologías como: Cliente Servidor, AJAX (*Asynchronous Java Script And XML*), streaming de audio y video, y el progreso cada vez mayor del tratamiento de contenido multimedia, resulta vital el uso de cada una de ellas para la creación de software orientado a la radio y la televisión.

La implementación de la PTARTV está muy unida al uso de estas tecnologías. A continuación se realiza un análisis del porqué de la selección y las numerosas ventajas que brindan para el sistema a desarrollar.

2.5.1 Modelo Cliente Servidor

En el mundo de TCP/IP las comunicaciones entre computadoras se rigen básicamente por lo que se llama modelo Cliente-Servidor, éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones. El término Cliente-Servidor fue usado por primera vez en 1980 para referirse a computadoras que se encontraban en red. (11)

En este modelo, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.

Las características que se pueden apreciar del modelo cliente servidor son:

- ✓ 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.
- ✓ Si ocurrieran modificaciones en el servidor estas afectarían en ninguna o poca escala al cliente.

Para el desarrollo del sistema una de las ventajas fundamentales que brinda esta tecnología es la de ser una infraestructura versátil, modular y basada en mensajes que mejora la portabilidad, la interoperabilidad y la escalabilidad de los sistemas. Además con el uso de este modelo aumenta la interactividad y se facilita el mantenimiento de las aplicaciones.

2.5.2 Aplicaciones Web

La utilización de los sistemas de escritorio permite que la información y la lógica se encuentren difundidas en cada ordenador que las utiliza. De esta forma se puede originar duplicidad de datos ya que no existe unificación. Además, actualizar o corregir un sistema resulta realmente complejo pues la modificación implica a cada ordenador. Estas situaciones contribuyeron a la creación de una nueva forma de sistemas.

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

2.5.3 AJAX

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

El término aparece en febrero de 2005 en el artículo "*AJAX: A New Approach to Web Applications*" publicado por Jesse James Garrett. Según el autor; AJAX no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de manera autónoma y que se unen de formas nuevas y sorprendentes. (12)

Actualmente existen numerosos *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.

Dojo es un conjunto de herramientas código abierto DHTML escrito en JavaScript que contiene librerías (APIs) y controles (*widgets*) que permiten facilitar el desarrollo de aplicaciones web sobre

tecnología AJAX. Este contenedor (*toolkit*) funciona como un sistema de empaquetado inteligente que relaciona la abstracción de eventos, almacenamiento de APIs en el cliente, efectos de interfaz de usuario y su interacción con AJAX.
Dojo representa una ventaja en tema de soporte para AJAX, porque posee la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado por el lado del cliente.

Dojo se provee de una arquitectura singular que utiliza una capa de abstracción con la que se pueden utilizar en un navegador otros transportes o IFrames[3] ocultos además de diferentes formatos de datos. En su diseño se encuentra un sistema de paquetes que facilitan el desarrollo modular. Se inicializa con una jerarquía de paquetes de espacios empezando por un paquete "dojo" y luego se puede cargar cualquier otro paquete vía XMLHttpRequest utilizando las funcionalidades ofrecidas en el arranque.

La importancia y utilidad de Dojo es la de abstraer al desarrollador de las complejidades del DHTML facilitando su entendimiento, implementación y el poder de resolver asuntos relacionados con la navegación.

Luego del análisis realizado del modelo AJAX *y* el *framework* Dojo y dadas las ventajas expuestas anteriormente, se utilizan para la implementación de la PTARTV 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.

2.5.4 Streaming de Audio y Video

Es una tecnología para transmitir video o audio por la web, de forma que optimiza la reproducción y el aprovechamiento del ancho de banda. Permite la reproducción de archivos multimedia sin la necesidad de copiarlo completamente, debido al gran tamaño que caracteriza a los mismos, la tecnología streaming se ajusta perfectamente a las necesidades de la web.

El streaming funciona de la siguiente forma: cuando se realiza una petición de algún fichero al servidor de streaming, este comienza a enviar el fichero al cliente que lo solicitó. El flujo es entregado directamente de la fuente al reproductor en tiempo real mediante un proceso continuo, solo almacenando una pequeña porción del streaming en el buffer del reproductor (el tamaño del buffer

[3] Elemento HTML que permite insertar o incrustar un documento HTML dentro de un documento HTML principal.

depende del reproductor), del cual se reproduce la media, a medida que se reproduce del buffer, se elimina esa información y se copia la siguiente. De esta manera si varía la velocidad de copia de la red o se pierde la conexión con el servidor, se utiliza la información del buffer para continuar reproduciendo (mientras mayor sea el tamaño del buffer, mayor será el tiempo que se pueda estar sin conexión o con poca velocidad de conexión con el servidor). (13)

Algunos de los componentes que conforman esta tecnología son:

- **Servidor de Streaming:** Software encargado de atender las peticiones de los usuarios para acceder al contenido multimedia.

  Los servidores pueden brindar dos tipos de contenidos:

  - ✓ **Bajo demanda o a la carta** (VOD, siglas en inglés): Permite al usuario hacer peticiones personalizadas de los ficheros almacenados en el servidor. Se realizan de manera individual, cada cliente que solicita una secuencia suele tener el control total de la misma pudiendo detenerla y reanudarla según su voluntad. Esto se debe a que se proporciona una sola ruta de acceso a los datos para cada cliente que solicita el contenido.
  - ✓ **Difusión** (*broadcast*): Es un modo de transmisión de información donde un emisor envía información a una multitud de receptores de manera simultánea, sin necesidad de reproducir la misma transmisión por cada receptor. Es usada para transmitir video en vivo pues permite una audiencia de gran volumen. Los clientes que la reciben no pueden controlar el inicio del material, la velocidad de reproducción, ni rebobinar la secuencia. Es el servidor el que tiene el control de la secuencia.

- **Reproductor o Cliente:** Software cuyo propósito es la reproducción de los archivos suministrados por el servidor de streaming. Es necesario que esté instalado en el ordenador para recibir las secuencias del servidor.

La tecnología del streaming de audio y video se ha convertido en una tecnología necesaria para la automatización de procesos de transmisión de radio y televisión, haciendo uso de redes de telecomunicación o cualquier otra infraestructura. Las innegables ventajas y posibilidades que ofrece son esenciales para la solución de la plataforma que se desea.

2.6 Lenguaje de Programación

Los lenguajes de programación son un conjunto de símbolos, sintaxis y reglas semánticas que permiten la comunicación entre una persona y un ordenador. Constituyen una técnica estándar de comunicación para entregarle instrucciones a la computadora.

2.6.1 Lenguajes de programación del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor. En el caso de una aplicación web esto permite que la página pueda ser albergada en cualquier sitio. Pero la página no se verá bien si el ordenador cliente no tiene instalado la versión correcta del navegador y los plugins[4] adecuados. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y esto puede afectar la seguridad.

2.6.1.1 HTML

De las siglas de (*HyperText Markup Language)* Lenguaje de Marcas de Hipertexto es el lenguaje de marcado más utilizado para la construcción de páginas web.

Es un lenguaje sencillo, permite definir documentos de hipertexto a base de ciertas etiquetas que marcan partes del documento dándoles una estructura o jerarquía. Presenta el texto de una manera estructurada y agradable, con enlaces (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas y con inserciones multimedia (gráficos, sonido, video). El lugar donde se encuentra esta información puede ser el mismo documento o cualquier otro lugar de Internet. (14)

2.6.1.2 Java Script

Es un lenguaje de tipo script compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de internet, es un lenguaje de programación interpretado, como ventaja clave de este lenguaje se puede mencionar que es interpretado por todos los navegadores modernos, debido a que este lenguaje está provisto de una implementación del (*Document ObjectModel* o Modelo de Objetos del Documento) DOM estándar diseñado por W3C que incorporan Konqueror, Mozilla Firefox desde su primera versión, Internet Explorer desde su versión 6.0, Netscape Navigator, Opera desde la versión 7 entre otros. El navegador tiene la responsabilidad de interpretar las sentencias.

[4] Programas que se relacionan con un sistema, con el objetivo de añadirle funcionalidades.

La PTARTV se verá beneficiada por las características mencionadas anteriormente, siendo de vital importancia para su correcto funcionamiento. No se debe olvidar la vinculación existente entre el lenguaje JavaScript y el conjunto de tecnologías AJAX por lo que prescindir de su utilización en la realización de este trabajo no es una opción.

2.7 Lenguajes de programación del lado del servidor

La programación del lado del servidor es una tecnología que consiste en el procesamiento de una petición de un usuario mediante el reconocimiento, ejecución e interpretación de un script en el servidor para generar páginas HTML dinámicamente como respuesta, las cuales son comprensibles para el cliente.

Un lenguaje que se ejecuta en el lado del servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo.

2.7.1.1 PHP

Acrónimo de *Hypertext Preprocessor* es un robusto lenguaje que fue diseñado en 1994 por Rasmus Lerdorf, entre sus principales características destacan las siguientes:

- Libre, multiplataforma y posee numerosas funcionalidades de manera nativa.
- Tiene comportamiento modular pues brinda la posibilidad de adicionar nuevas funcionalidades a través de nuevos módulos.
- Permite conectarse a varios sistemas de bases de datos brindando múltiples funcionalidades, posee conectividad con PostgreSQL y MySQL que son gestores muy utilizados en el desarrollo de aplicaciones web.
- Cuenta con amplia documentación tanto en su página oficial como en distintos foros y publicaciones.
- Permite el empleo de programación Orientada a Objeto.
- No es necesario que se le especifique el tipo de datos de las variables.
- Maneja excepciones a partir de la versión 5.0.

Resulta indiscutible que PHP es la solución por excelencia a la implementación de la PTARTV. El hecho de ser libre y poder utilizarlo sin necesidad de disponer de una licencia aumenta sus ventajas. PHP posee soporte para la gran mayoría de las plataformas existentes en la actualidad. Es eficiente

para el desarrollo de aplicaciones web y logra integrarse con un gran número de servidores, incluso propietarios. Su sintaxis clara y bien definida es lo que permite que sea sencillo de aprender y utilizar.

2.7.1.2 Java

El objetivo principal de Java es conseguir un entorno de desarrollo de software que sea independiente de la plataforma de ejecución. Necesita 128 MB de memoria por sesión de usuario. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Java se diseñó para ser similar a C++ y así facilitar un rápido y fácil aprendizaje. (15)

Java reduce un 50% los errores más comunes de programación al eliminar muchas de las características de éstos, entre las que destacan:

- Aritmética de punteros.
- No existen referencias.
- Registros (*struct*)
- Definición de tipos (*typedef*)
- Necesidad de liberar memoria (*free*)

Una desventaja importante a considerar en la utilización de este lenguaje de programación es la necesidad de elevadas prestaciones del hardware para que este funcione correctamente, aspecto que se valora y se prevé un rendimiento aceptable con los recursos técnicos disponibles.

Se decidió utilizar el lenguaje Java por todas las facilidades expresadas anteriormente, su uso será en dos subsistemas de la PTARTV que serán aplicaciones de escritorio.

2.8 Framework

El término *framework* es muy utilizado en el campo de la informática, normalmente tratado como *framework* de desarrollo, no es más que un esquema para el desarrollo y/o la implementación de una aplicación, proporciona una estructura al código fuente, forzando al desarrollador a crear código legible y más fácil de mantener.

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Un *framework* facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (16)

2.8.1 Symfony

Symfony es patrocinado por Sensio, una agencia web francesa. Su primer nombre fue Sensio Framework y sus clases contenían el prefijo sf. Cuando se decide lanzarlo como un *framework* de código abierto, se acuerda llamarle Symfony.

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica del servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (16)

Está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios 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 *nix (Unix, Linux, etc.) como en plataformas Windows.

La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos. Los formularios incluyen validación automatizada y relleno automático de datos. La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor. La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario. Las interacciones con Ajax son muy fáciles de implementar mediante los *helpers* que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código. (17)

Symfony es sin lugar a dudas, la solución factible para el desarrollo de la PTARTV. Dos de las ventajas importantes de Symfony frente a otros *frameworks* son la calidad que brinda al código fuente y la cantidad de documentación disponible. El número de sus colaboradores ha crecido rápidamente y su labor va desde proponer parches y mejoras hasta detectar errores de la documentación.

2.8.2 Hibernate

Hibernate es un *framework* que constituye un motor de persistencia que implementa múltiples funcionalidades. El centro de la arquitectura de Hibernate lo constituyen una serie de interfaces que realizan el grueso de las funcionalidades del *framework*, dentro de ellas están:

Hibernate proporciona un dialecto orientado a objetos (HQL) fácil de manejar y familiar a SQL que aunque no es un lenguaje de manipulación de datos como este, puesto que es usado solamente para extraer datos no para borrar, insertar o actualizar, permite realizar complejas consultas. (18)

Este framework se utilizará para el mapeo de la base de datos cuando se trabaje con el lenguaje Java en los subsistemas que lo requieren.

Para realizar una correcta implementación de la PTARTV utilizando los lenguajes y *framework* definidos existen herramientas que proporcionan a los desarrolladores de numerosas facilidades en la confección del código fuente, es el caso de los entornos integrados de desarrollo.

2.9 Entorno de Desarrollo Integrado (IDE)

Es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

2.9.1 NetBeans

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento. Los desarrolladores acostumbrados a trabajar con herramientas basadas en la tecnología Java, tales como el conjunto de herramientas *Borland JBuilder*, descubren que la migración a NetBeans puede acelerar en forma significativa los esfuerzos de desarrollo. (19)

El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Les sirve a los programadores para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. La versión de NetBeans 6.8 soporta el desarrollo en: Java, Ruby, C/C++, Groovy y PHP.

2.10 Sistemas Gestores de Bases de Datos

Un Sistema de Gestión de Base de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una Base de Datos (BD), por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones. Pueden ser de propósito general o específico. (20)

Hoy en día existen numerosos SGBD que tienen características propias, no obstante todos deben poseer los siguientes aspectos: abstracción de la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia.

2.10.1 PostgreSQL

El SGBD relacional orientado a objetos conocido como PostgreSQL está derivado del paquete Postgres escrito en Berkeley, distribuida bajo licencia BSD[5]. Con más de una década de desarrollo tras él, PostgreSQL ha demostrado ser un gestor de bases de datos de código abierto muy avanzado, ofreciendo control de concurrencia multiversión (MVCC), soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación como pueden ser C, C++, Java, Perl, TCL y Python. (21)

La lista de lenguajes escritos anteriormente puede actualizarse añadiendo al Object Pascal, PHP, Ruby, Pike y C#.

La utilización de este SGBD en la PTARTV obedece al propósito de la elaboración de un sistema con gran robustez y alto nivel de escalabilidad. Trabajar con este gestor es factible ya que es muy rápido, tiene buenas utilidades de administración, es confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños, sin límites en los tamaños de registros. Consecuentemente tiene un conjunto muy práctico de características desarrolladas en cooperación muy cercana con los usuarios.

2.11 Sistemas de Streaming

Para la realización de esta investigación una pregunta fundamental es ¿cómo se transmiten los flujos de video hasta su receptor? Por lo que es preciso definir que tecnología se va a usar para lograr esta meta.

[5]Licencia de software otorgada principalmente para los sistemas (*Berkeley Software Distribution*) BSD. Es una licencia de software libre, tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público, permite el uso del código fuente en software no libre.

Un sistema de streaming es una aplicación informática cuya función es realizar streaming de audio y video o de uno de los dos. Para una mayor comprensión de este tipo de aplicaciones se recomienda retomar el tópico 2.5.4 del presente trabajo donde se valora el funcionamiento de esta tecnología.

Existen varios servidores streaming bajo licencias libres que pueden usarse en la plataforma, pero presentan actualmente algunas deficiencias que hacen que no se ajusten a todas las necesidades de dicha plataforma. Entre los principales servidores streaming libres están: FFmpeg y FFServer, Darwin Streaming Server (DSS), VideoLan (VLC), MPEG4IP.

2.11.1 VideoLan

La solución para streaming de VideoLan incluye dos programas:

- VLC media player: el cual puede ser utilizado como un servidor y como un cliente para hacer streaming y recibir streaming por la red. VLC es capaz de hacer streaming con todos los archivos que puede leer.
- VLS (VideoLAN Server): el cual puede hacer streaming de archivos MPEG-1, MPEG-2 y MPEG-4, DVDs, canales digitales de satélite, canales de televisión digital terrestre y videos en vivo por la red en multidifusión o unidifusión. La mayoría de las funcionalidades del VLS pueden ser ahora encontradas en el VLC.

El VLC es capaz de transmitir todos los archivos que sea capaz de reproducir por lo que puede ser utilizado como servidor streaming para transmitir desde un dispositivo de captura o desde un fichero previamente almacenado con numerosas opciones de codificación en tiempo real. El proceso de creación de un nuevo flujo a transmitir lleva una serie de pasos que deben ser realizados desde la interfaz (ya sea gráfica, web o telnet) que incluyen la selección del tipo de flujo (en demanda o en vivo), la selección del origen (dispositivo de captura, DVD, archivo previamente almacenado, etc.), selección del formato de salida (transcodificación en tiempo real a numerosos formatos). En caso que se cierre la instancia del VLC que se encuentra configurada y transmitiendo los contenidos hay que realizar el proceso nuevamente para cada contenido, esta es una de las principales deficiencias con que cuenta este sistema hoy en día.

Para que VLC resulte recomendable como servidor de streaming para la PTARTV debe lograrse que sea capaz de leer, configurar y transmitir automáticamente los archivos que se encuentren almacenados en una carpeta determinada. Lo descrito anteriormente debe realizarse cuando se inicie el servidor y es necesario que detecte cuando se añade una nueva media a la carpeta. Debe tenerse

en cuenta que es preciso mantener la instancia del VLC en ejecución de forma constante, brindando servicios. Esto es fundamental para una aplicación que funcione como servidor.

2.11.2 Darwin Streaming Server

Es la versión de código abierto de la tecnología del Apple's QuickTime Streaming Server que permite enviar un streaming de multimedia a clientes a través de internet utilizando los protocolos RTP y RTSP. Basado en el mismo código base del QuickTime Streaming Server, el Darwin Streaming Server proporciona un alto nivel de personalización y funciona en una gran variedad de plataformas permitiendo manipular el código fuente para satisfacer las necesidades de los desarrolladores. Darwin Streaming Server es un proyecto de código abierto pensado para desarrolladores que necesiten hacer streaming de archivos QuickTime o MPEG-4 en plataformas alternativas como Windows, Linux y Solaris; o para aquellos desarrolladores que necesiten extender y/o modificar el código del servidor streaming existente para satisfacer sus necesidades.

Ventajas de Darwin Streaming Server

- Está disponible para descarga gratuita bajo la Apple Public Source License.
- Una vez instalado el servidor crea una carpeta en la cual se deben copiar los archivos en formato 3gp, mov, mp4 y mp3 (luego de someterse a un proceso denominado hint) para video. Los ficheros que se encuentren en esta carpeta están publicados y pueden ser accedidos desde la red mediante el protocolo RTSP.
- El servidor cuenta con una interfaz web que permite visualizar su estado y además posibilita su configuración y administración.

Aplicaciones de streaming seleccionadas

En un diseño inicial de la PTARTV se había propuesto únicamente como servidor streaming al VLC, pero según se planteó en el tópico 2.11.2 de presente trabajo se hace evidente que el mismo posee algunas deficiencias hasta el momento no solucionadas. Es por ello que se propone junto al VLC, el Darwin Streaming Server como servidor streaming de video bajo demanda atendiendo a todas las ventajas expresadas anteriormente.

2.12 Servidor Web

Un servidor es una computadora que entrega a otras computadoras (los clientes), una información que estas últimas requieren bajo un lenguaje común, denominado protocolo. Dependiendo del tipo de la

petición, el servidor web buscará una página web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. Por lo tanto en el servidor web es donde se almacena la información estática accedida y/o las aplicaciones que la generan.

El servidor web será fundamental en el desarrollo de las aplicaciones del lado del servidor que se implementen pues funcionarán sobre él.

2.12.1 Apache

Servidor web de código abierto. Su desarrollo comenzó en febrero de 1995, por Rob McCool, en una tentativa de mejorar el servidor existente en el Centro Nacional de Aplicaciones de Supercomputación (NCSA) de los Estados Unidos.

Hoy en día es el servidor web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de libre distribución que publica su código fuente, lo que permite que cualquier programador pueda modificarlo y colaborar así a su desarrollo. Dado el gran uso del servidor web Apache a nivel internacional, su integridad y robustez, así como las enormes posibilidades que brinda al estar implementado sobre software libre; resulta la opción viable para la PTARTV.

2.13 Sistema Operativo

El sistema operativo es el programa (o software) más importante de un ordenador. Para que funcionen los otros programas, cada ordenador de uso general debe tener un sistema operativo. Los sistemas operativos realizan tareas básicas, tales como 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, etc. (22)

En sistemas grandes, el sistema operativo tiene incluso mayor responsabilidad y poder, es como un policía de tráfico, se asegura de que los programas y usuarios que están funcionando al mismo tiempo no interfieran entre ellos. El sistema operativo también es responsable de la seguridad, asegurándose de que los usuarios no autorizados no tengan acceso al sistema. (22)

Al abogarse por el uso del software libre para el desarrollo del sistema PTARTV se impone entonces la necesidad de una correcta selección del sistema operativo a utilizar como plataforma.

2.13.1 UBUNTU Linux

Ubuntu es una distribución GNU/Linux que brinda las opciones de un sistema operativo centrado en ordenadores personales, aunque también ofrece soporte para servidores. Se ha convertido sin dudas en una de las distribuciones más importantes a nivel mundial.

Tiene su base en Debian GNU/Linux y su principal objetivo es la facilidad y libertad de uso, la fluida instalación y los lanzamientos regulares cada 6 meses aproximadamente de nuevas versiones.

Ubuntu se convierte para las necesidades de este trabajo en una solución robusta. Por tanto, será usado como sistema operativo para la PTARTV la última versión de la distribución Ubuntu.

2.14 Software para el control de versiones (CVS)

El control de versiones es el arte de manejar cambios en la información. Ha sido desde siempre una herramienta crítica para los programadores, quienes típicamente emplean su tiempo haciendo pequeños cambios al software y luego deshaciendo o comprobando esos cambios al día siguiente.

2.14.1 Subversion con cliente RapidSVN

Subversion es un sistema de control de versiones libre y de código fuente abierto. Soporta el manejo de ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto que recuerda todos los cambios hechos a sus ficheros y directorios, permitiendo recuperar versiones antiguas de datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Existen varias interfaces a subversion, ya sea programas individuales como interfaces que lo integran en entornos de desarrollo

RapidSVN es un cliente gráfico para subversion escrito en C++, un programa de control de versiones sustituto de CVS. Es fácil de usar, tanto por quienes ya conocen subversion como para quienes empiezan, pudiendo acceder a direcciones SVN, subir y descargar contenido y sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones, etc. Además tiene la ventaja de funcionar en varias plataformas y de disponer de un completo manual en línea. (23).

2.15 Salvas Automáticas

2.15.1 Bacula

Bacula es un conjunto de herramientas de respaldo muy amplia pues son capaces de cubrir de manera genial las necesidades de respaldo de equipos bajo redes IP. Es liberado bajo la licencia GPL versión

2. Está basado en la arquitectura cliente/servidor que resulta muy eficiente y fácil de manejar, dada la gran cantidad de funcionalidades y características que brinda como son copiar y restaurar ficheros dañados o perdidos. Gracias a su desarrollo y estructura modular, Bacula se adapta tanto al uso personal como profesional, para cantidades de ordenadores muy grandes. (24)

Después de esta caracterización se aprecia que Bacula es un software libre para realizar copias de seguridad muy potente que apenas tiene nada que envidiar a sus equivalentes en software propietario. No es una aplicación monolítica sino que es un conjunto de varios servicios junto con una interfaz de usuario. Los servicios tienen responsabilidades establecidas y utilizan la red para comunicarse. Es una herramienta muy útil, extremadamente flexible y con características profesionales. Contiene una buena documentación y se integra fácilmente en entornos con sistemas heterogéneos.

2.16 Conclusiones

En este capítulo se realizó la comparación de las principales características de un conjunto de tecnologías, herramientas y tendencias asociadas a la posible propuesta para solucionar el problema científico que concierne la presente investigación.

Como aplicaciones de streaming se concluye que no es suficiente con el VLC por lo que a se propone utilizarlo junto con el Darwin Streaming Server.

No es suficiente con un solo lenguaje para el desarrollo de la plataforma pues son muchas las funcionalidades que esta debe realizar por lo que se propone Java para los subsistemas de escritorio y PHP para los subsistemas web ambos sobre el entorno de desarrollo Netbeans.

Capítulo 3
Propuesta de la arquitectura de software

3.1 Introducción

Este capítulo tiene como objetivo lograr una mejor visión del sistema, organizar el desarrollo, posibilitando esto que el desarrollo del sistema sea de forma eficiente. Aquí se obtienen dos artefactos fundamentales (Línea base de la arquitectura y el documento de Descripción de la arquitectura) donde se describe la solución arquitectónica del sistema, además otros artefactos definidos por la metodología RUP como son las vistas arquitectónicas.

3.2 Estructura del equipo de desarrollo.

Miembros	
Líder de proyecto	Revisor técnico
Analista	Planificador
Implementador	Documentador
Arquitecto de software	Diseñador Gráfico
Diseñador de base de datos	Ingeniero de prueba

Tabla 1: Equipo de desarrollo

3.3 Subsistemas de la aplicación

- ✓ Subsistema Web.
- ✓ Subsistema Administración de la Transmisión.
- ✓ Subsistema Transmisión.
- ✓ Subsistema Gestión de Medias.
- ✓ Subsistema Programación.
- ✓ Subsistema Transferencia.
- ✓ Subsistema Seguridad.
- ✓ Subsistema Reporte.

Responsabilidad de cada subsistema

Subsistema Web

El subsistema web será la interfaz externa de la PTARTV. Haciendo uso de él los usuarios que se conecten podrán satisfacer sus preferencias televisivas consumiendo los servicios brindados por esta aplicación informática. Es importante comprender que el framework Symfony utilizado para el desarrollo de la aplicación estructura los sistemas en módulos. Por la razón antes expuesta se conformará este subsistema en siete módulos que permitan una implementación clara y correcta.

- ✓ Módulo noticias web.
- ✓ Módulo publicación web.
- ✓ Módulo sección web.
- ✓ Módulo cartelera.
- ✓ Módulo estadísticas.
- ✓ Módulo encuestas.
- ✓ Módulo foro.

Subsistema Administración de la Transmisión

Este subsistema será el encargado de la gestión de los canales de la plataforma.

Subsistema Transmisión

El subsistema de Transmisión será el encargado de transmitir los recursos multimedia programados, para que sean presentados a los televidentes. Permite el monitoreo de los canales una vez que se están transmitiendo y permitirá además el encadenamiento de los canales.

Subsistema Gestión de medias

Se encargará de toda la gestión de las medias en el servidor de la plataforma, así como del chequeo de la integridad de las mismas. Este subsistema contará con un módulo:

- ✓ Módulo de medias.

Subsistema Programación

Manejará todo lo relacionado con la programación de la radio y la televisión de la plataforma. Entre sus

principales funcionalidades resalta gestionar las programaciones de radio y televisión de los canales de la plataforma. Poseerá dos módulos:

- ✓ Módulo de Programación.
- ✓ Módulo de Reporte.

Subsistema Transferencia

La responsabilidad de este subsistema será codificar los materiales de audio y video a los formatos requeridos para la transmisión y publicación dentro de la PTARTV. Además, este subsistema se encargará del chequeo de la integridad de los ficheros codificados, permitiendo que sean copiados al servidor de medias y brindará un reporte del estado de la conversión. Contará con dos módulos:

- ✓ Módulo conversión.

Subsistema Seguridad

Poseerá la responsabilidad de ocuparse de toda la seguridad del sistema, gestionando para ello el acceso de los usuarios a la aplicación y los permisos de estos. Un módulo:

- ✓ Módulo administración.

Subsistema Reporte

Este subsistema será el encargado de mostrar todos los reportes que se generan en los demás subsistemas, pueden ser por días, por semanas o por otros intervalos de tiempo. Debe mostrar además una presentación de todos estos reportes unidos en un formato específico. Sus módulos son:

- ✓ Módulo reporte programación.
- ✓ Módulo reporte web.
- ✓ Módulo reporte transmisión.
- ✓ Módulo reporte administración de la transmisión.
- ✓ Módulo reporte transferencia.
- ✓ Módulo reporte gestión de medias.
- ✓ Módulo reporte seguridad.

3.4 Descripción de la arquitectura

Para proporcionar una mejor comprensión arquitectónica del sistema se utilizan las vistas arquitectónicas definidas por la metodología RUP. Serán modeladas utilizando *Visual Paradigm*.

- Vista de casos de uso.
- Vista lógica.
- Vista de procesos.
- Vista de implementación.
- Vista de despliegue.

3.4.1 Vista de casos de uso

A partir de la vista de Casos de Uso, se puede definir los escenarios o los casos de uso que serán de interés para cada iteración del ciclo de desarrollo. Esta describe los escenarios o casos de uso que tienen significación y que encapsulan la funcionalidad central del sistema.

Los casos de uso arquitectónicamente significativos, son aquellos que describen funcionalidades imprescindibles para el sistema, y que a través de estos se valida la arquitectura propuesta para el mismo.

El sistema cuenta con 34 casos de uso, de los cuales 28 son arquitectónicamente significativos (Anexo 6). Están representados por subsistemas de la siguiente forma:

Subsistema	Cantidad de Casos de Uso
Web	10
Transferencia	2
Programación	6
Gestión de Medias	2
Seguridad	1
Administración de la transmisión	1
Transmisión	4
Reporte	8

Tabla 2: Cantidad de casos de uso por subsistema

A continuación se muestran los diagramas que representan los casos de uso arquitectónicamente significativos para cada subsistema:

Subsistema Web

- CU Gestionar noticias.
- CU Gestionar publicación.
- CU Gestionar secciones.
- CU Gestionar cartelera.

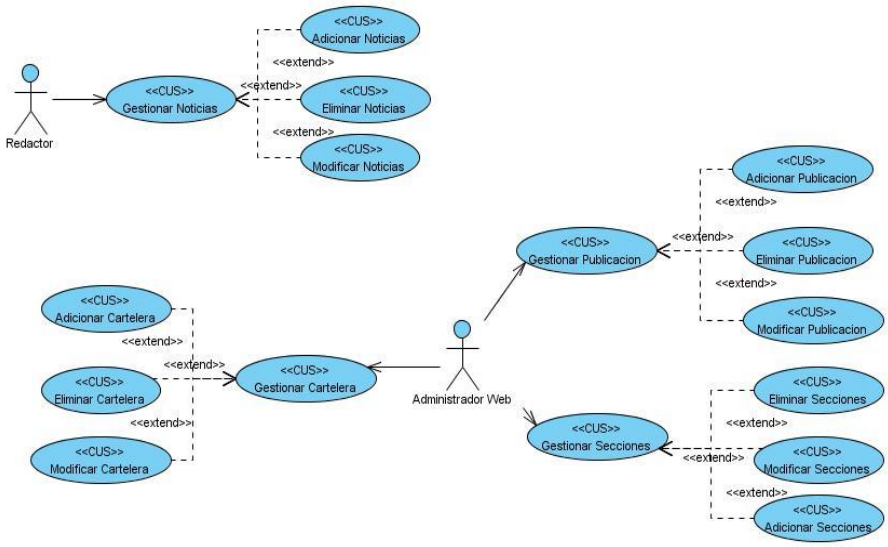

Figura 3: Vista de casos de uso del subsistema Web

Subsistema Administración de la Transmisión

- CU Gestionar canales.

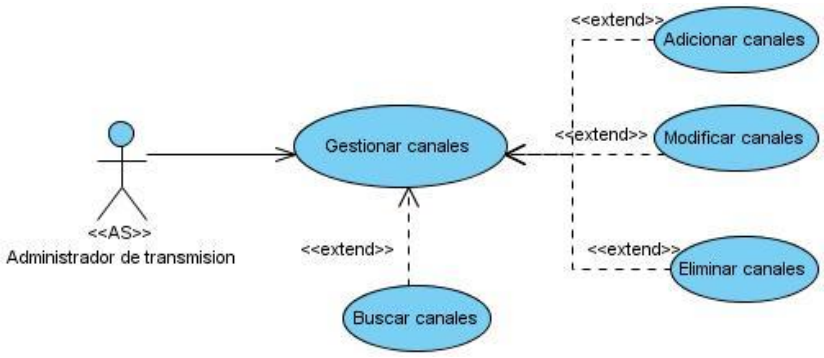

Figura 4: Vista de casos de uso del subsistema Administración de la Transmisión

Subsistema Transmisión

- CU Realizar transmisión.
- CU Interrumpir transmisión.
- CU Monitorear canales.
- CU Encadenar canales.

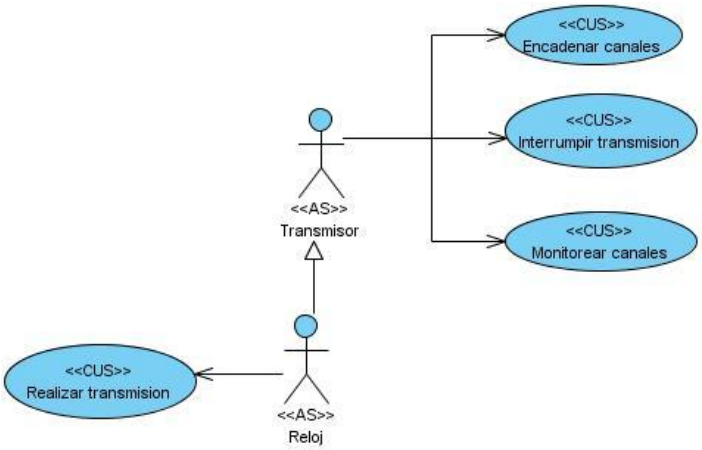

Figura 5: Vista de casos de uso del subsistema Transmisión

Subsistema Gestión de medias

- CU Gestionar media.
- CU Chequear integridad.

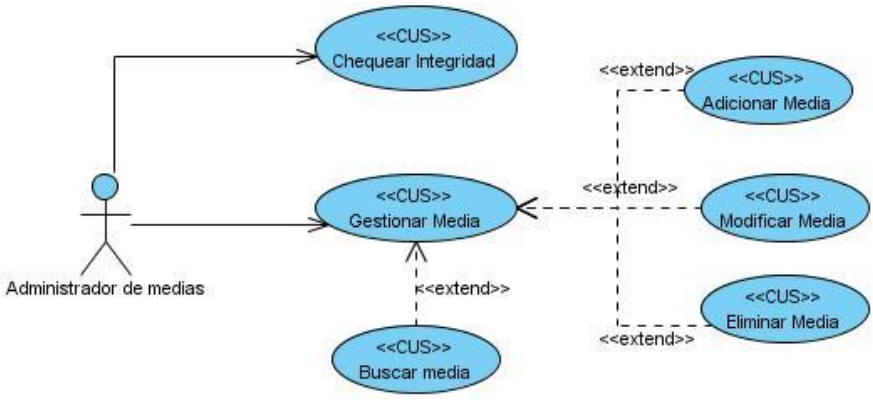

Figura 6: Vista de casos de uso del subsistema Gestión de medias

Subsistema Transferencia

- CU Convertir media.
- CU Transferir media.

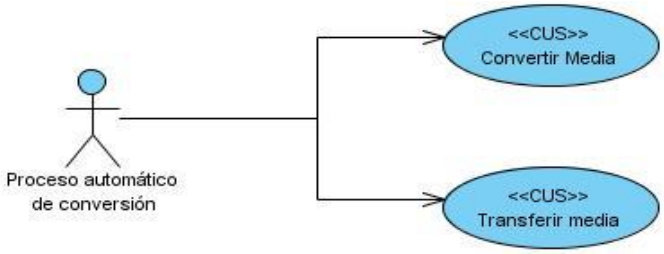

Figura 7: Vista de casos de uso del subsistema Transferencia

Subsistema Programación

- CU Gestionar programación para radio.
- CU Gestionar programación de televisión.
- CU Visualizar reporte de programaciones.
- CU Gestionar espacios televisivos.
- CU Gestionar espacios de radio.
- CU Rellenar huecos en la programación.

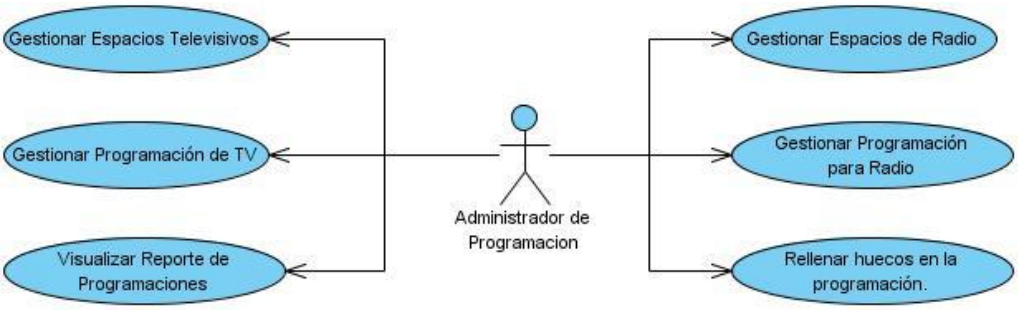

Figura 8: Vista de casos de uso del subsistema Programación

Subsistema Seguridad

- CU Gestionar Usuarios

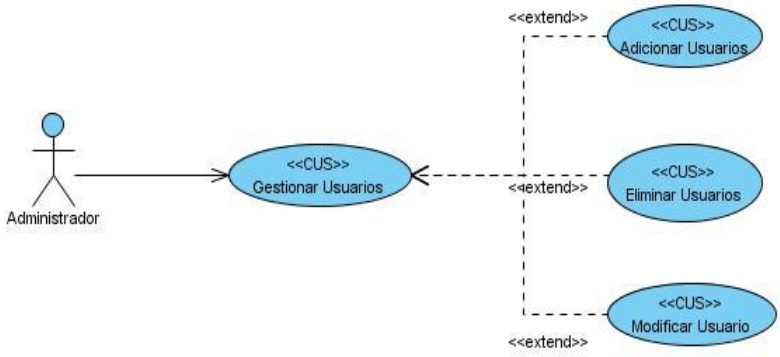

Figura 9: Vista de casos de uso del subsistema Seguridad

Subsistema Reporte

- CU Visualizar reporte web.
- CU Visualizar reporte de transferencia.
- CU Visualizar reporte de transmisión.
- CU Visualizar reporte de programación.
- CU Visualizar reporte de medias.
- CU Visualizar reporte de administración de la transmisión.
- CU Visualizar reporte de seguridad.

- CU Visualizar presentación de reporte.

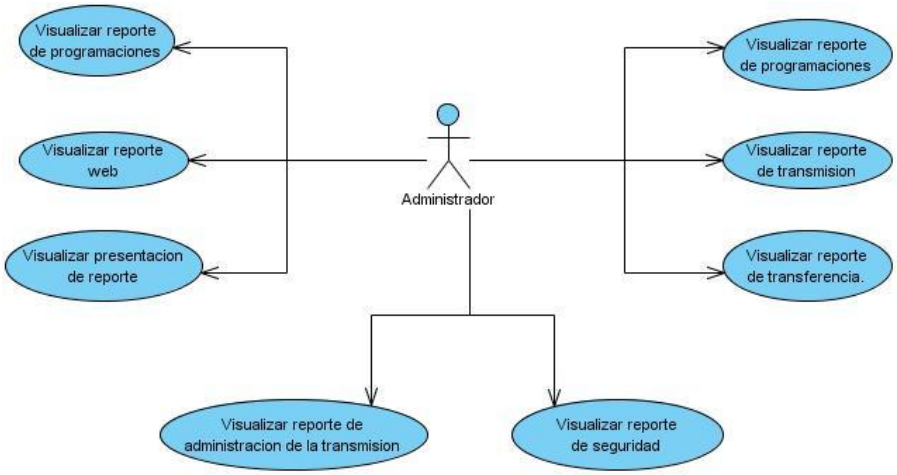

Figura 10: Vista de casos de uso del subsistema Reporte

3.4.2 Vista Lógica

La vista lógica se encarga de describir el diseño mostrando solo las clases más importantes y su organización en paquetes y subsistemas, y la organización de éstos últimos en capas. Esta vista permite observar en el interior del sistema como está diseñada su funcionalidad, tanto su comportamiento estático como su comportamiento dinámico.

El patrón que se aplicará es el MVC, que es actualmente el más usado en la confección de aplicaciones web debido a las ventajas que trae consigo como la forma en que organiza los elementos de la aplicación.

Para los subsistemas web se usará el *framework* Symfony que adapta su estructura a la organización de dicho patrón.

- **La capa del Modelo**
  - Abstracción de la base de datos
  - Acceso a los datos
- **La capa de la Vista**
  - Vista
  - Plantilla
  - Layout

- **La capa del Controlador**
  - Controlador frontal
  - Acción (16)

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de SGBD, solamente es necesario actualizar la capa de abstracción de la BD.

La capa de la vista también puede aprovechar la separación de código. Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. Normalmente sólo cambia el interior de la página. Por este motivo, la vista se separa en un layout y en una plantilla.

El controlador normalmente se divide en un controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página. Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares.

La siguiente imagen muestra la distribución de los paquetes más significativos dentro de cada una de las partes definidas para la aplicación web y la dependencia entre ellos.

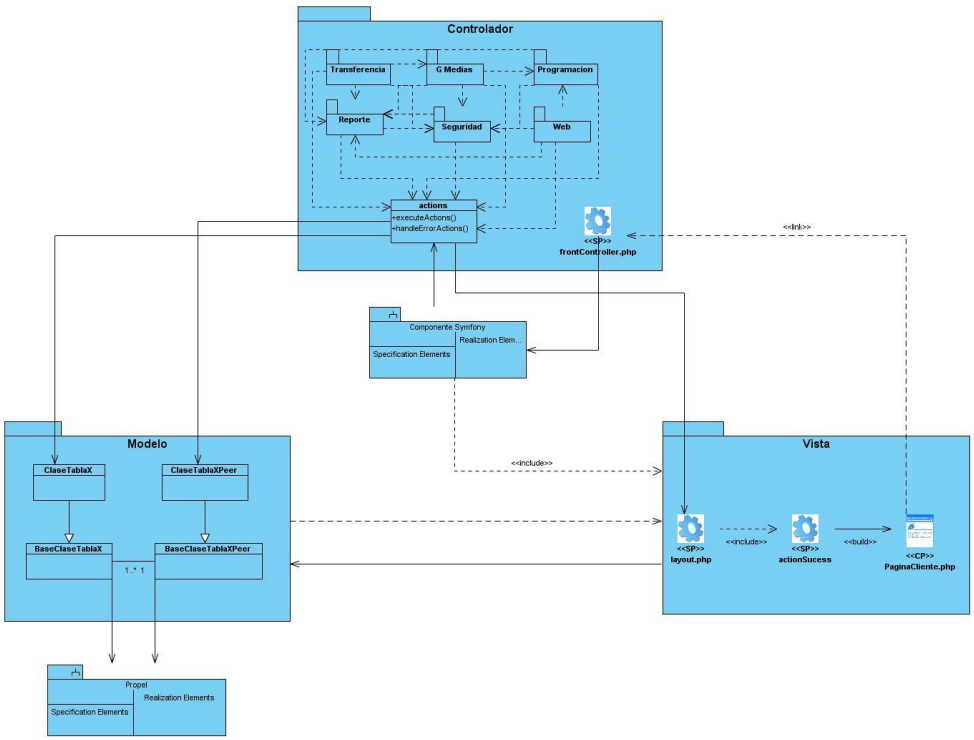

Figura 11: Vista Lógica de los subsistemas web

Elementos del modelo arquitectónicamente significativos.

Subsistema Propel: Será el ORM utilizado para el desarrollo de la solución informática pues facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la BD mediante objetos, con la que se puede recuperar, insertar y modificar datos. No se hace necesario preocuparse por las conexiones de la BD y escribir SQL, solo es preciso definir la base de datos en formato XML u obtener la definición desde una base de datos ya existente.

Subsistema componente Symfony: Represan todas las clases del *framework* que serán utilizadas durante el funcionamiento del sistema. Dígase validadores de formularios, *helpers* de objetos y formularios, plantillas, componentes de seguridad, etc.

Paquete controlador: Una parte importante de su trabajo es común a todos los controladores de la aplicación. Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el controlador se ha dividido en un controlador frontal (frontControler.php), que se encarga de realizar las

tareas comunes y las acciones (actions.php), que incluyen el código específico del controlador de cada página. Habrá un actions.php por cada uno de los módulos pero el controlador frontal será el mismo para todo el sistema.

Paquete vista: Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el *layout* genérico, el pie de página y la navegación global. En la mayor parte de las veces sólo cambia el interior de la página. Por este motivo, la vista se separa en un *layout* y en una plantilla (el nombre de las platillas está compuesto por el nombre de la acción que la origina seguido por sufijo que puede ser SUCCESS o ERROR). El *layout* será global en toda la aplicación o la mayoría de las páginas. La plantilla sólo se encarga de visualizar las variables definidas en el paquete del controlador.

Paquete del modelo: Solo contiene las clases encargadas del acceso a los datos almacenados en el gestor de base de datos, las cuales utilizan el ORM Propel para el acceso a los mismos.

A continuación se ilustra la vista lógica de los subsistemas de escritorio pertenecientes a la aplicación.

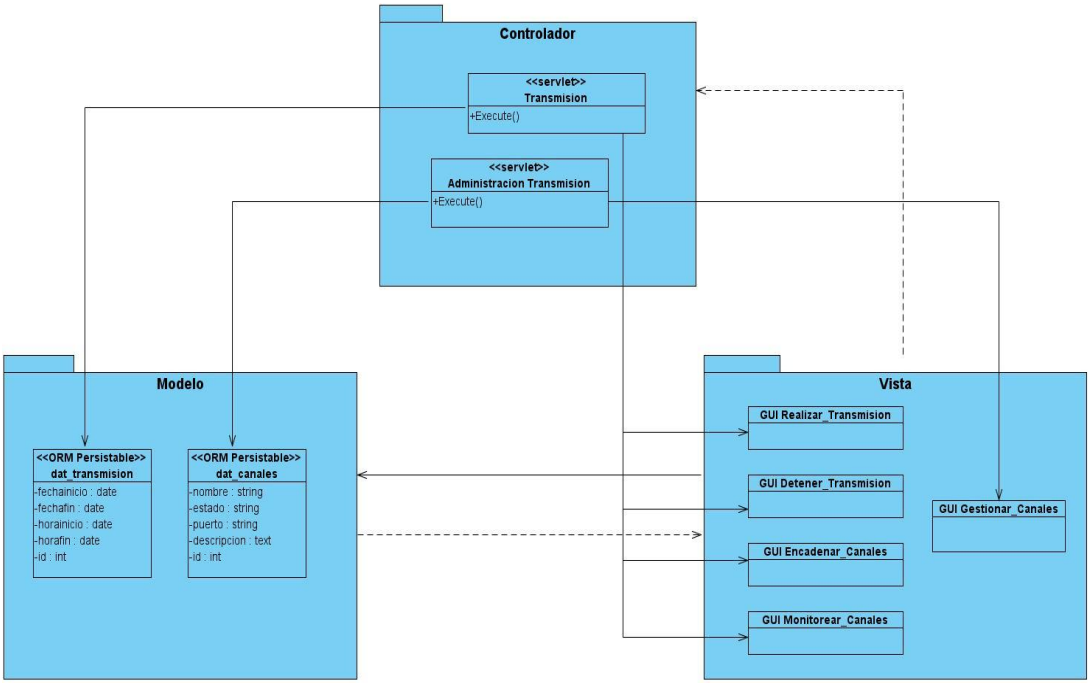

Figura 12: Vista lógica de los subsistemas de escritorio

Elementos del modelo arquitectónicamente significativos

Paquete controlador: En este componente se encuentran los *servlet***,** estos forman parte del programa de servidor Java y amplían la funcionalidad de un servidor web mediante la generación dinámica de contenidos y la interacción entre los componentes de la aplicación.

Paquete vista: Contiene las interfaces gráficas de usuario (GUI) que son técnicas que utiliza gráficos para proporcionar una interfaz fácil de usar en aplicaciones.

Paquete modelo: En este paquete se muestra una representación de los datos persistentes en la aplicación.

3.4.3 Vista de Procesos

La vista de procesos suministra una base para el entendimiento de la organización de los procesos del sistema, ilustrados en el mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos o hilos concurrentes.

El sistema se basa en la arquitectura cliente-servidor sobre la plataforma web, donde cada instancia del sistema en el cliente es independiente de la ejecución de otra. La concurrencia de utilización del servidor web y la utilización de la BD se maneja a través de los propios servidores.

El sistema no cuenta con tareas que requieran ejecutarse periódicamente sin la intervención del cliente.

Por lo expuesto anteriormente no se requiere una vista de procesos.

3.4.4 Vista de Implementación

La vista de implementación proporciona una descripción de las principales capas y módulos de componentes de la aplicación. Los paquetes principales de la aplicación por cada uno de los módulos son:

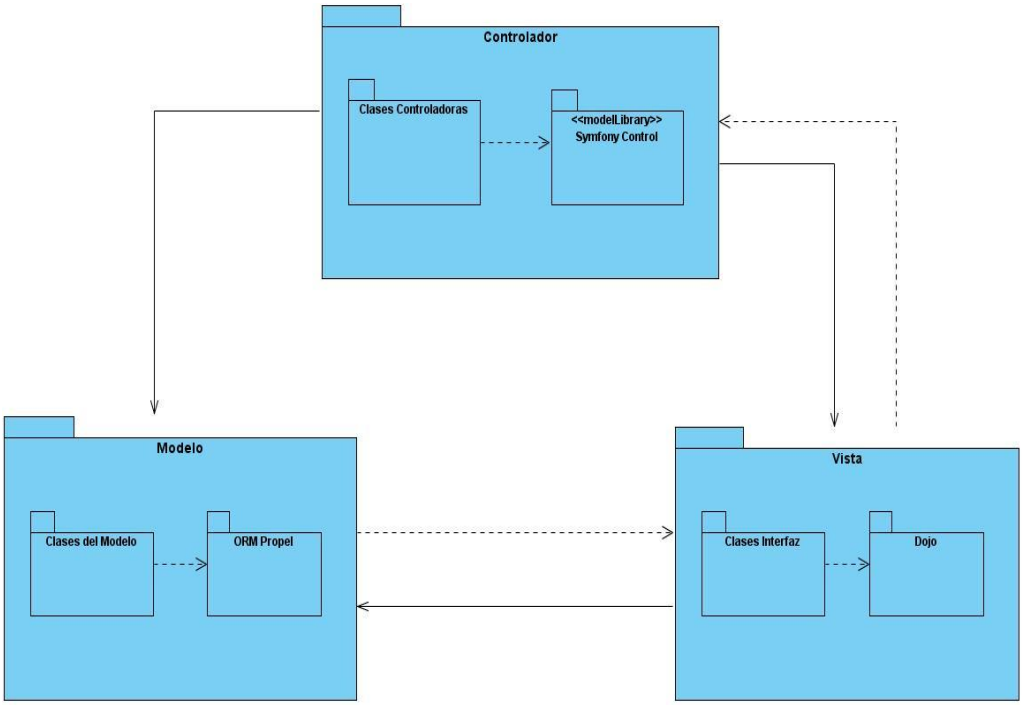

Figura 13: Vista de implementación de los subsistemas web

Cada uno de dichos paquetes encapsulan uno o más componentes que se interrelacionan entre ellos para darle solución a la aplicación.

A continuación se muestra la vista de implementación para los subsistemas de escritorio pertenecientes a la plataforma.

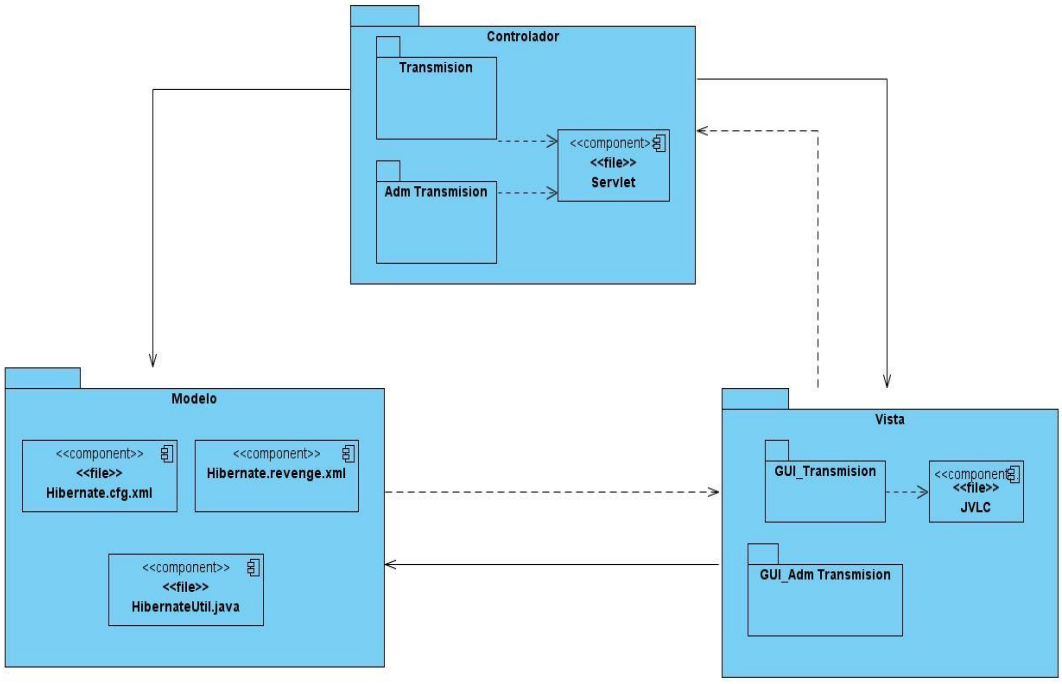

Figura 14: Vista de implementación de los subsistemas de escritorio

3.4.5 Vista de Despliegue

La vista de despliegue propone la distribución física del sistema y la distribución de los componentes de la aplicación en ellos. Existe una traza directa del modelo de implementación, puesto que cada componente físico debe estar almacenado en un nodo, esto incluye también la asignación de tareas provenientes de la vista de procesos en los nodos.

3.4.5.1 Diagrama de Despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema. Es un conjunto de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos y procesos.

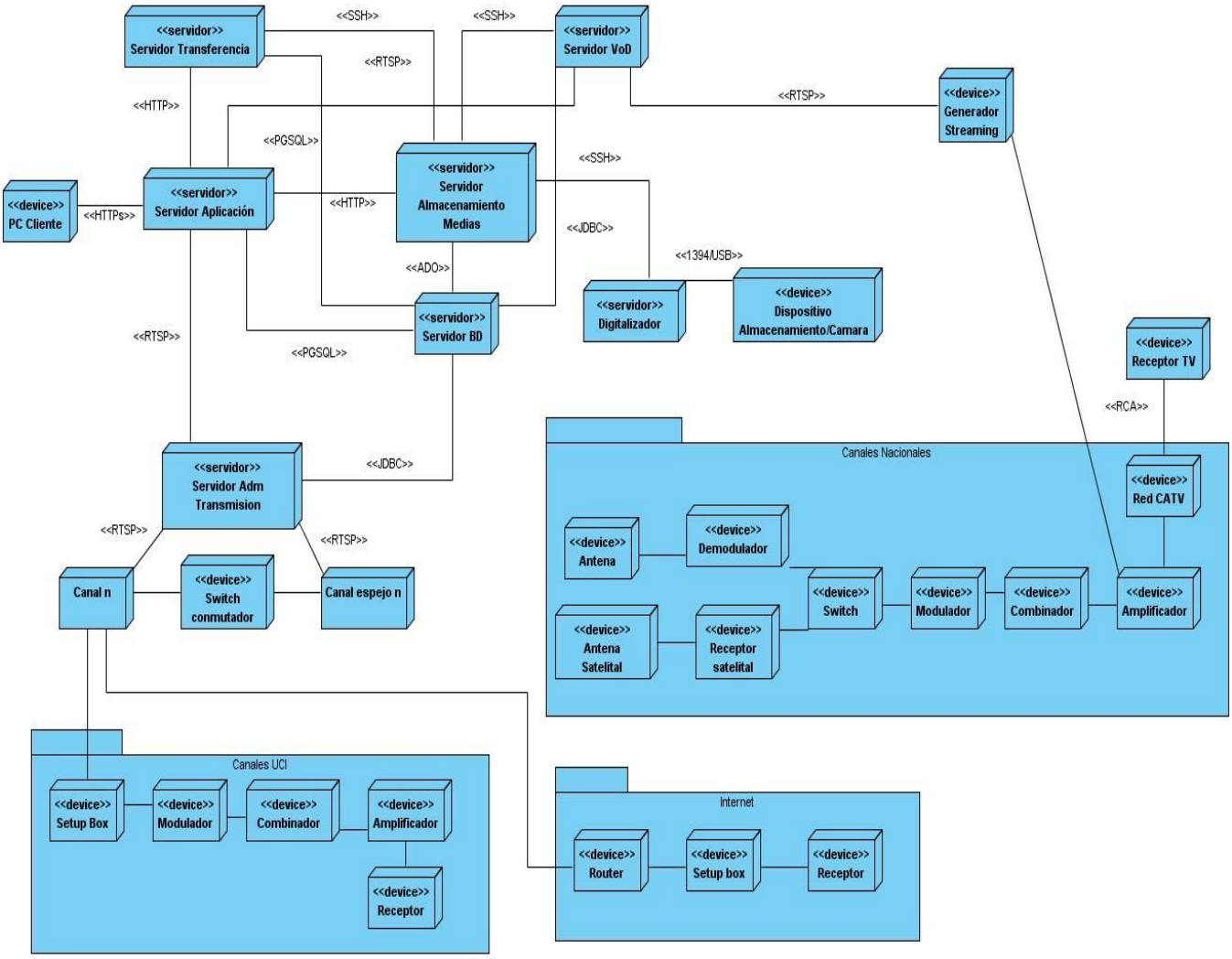

Figura 15: Diagrama de despliegue

3.5 Requerimientos no funcionales del sistema

3.5.1 Requerimientos de seguridad

- La seguridad se trata desde las primeras fases de desarrollo del sistema.
- La asignación de usuarios y sus opciones sobre el sistema se garantizan desde el subsistema de Seguridad.
- Los mecanismos de seguridad no deben impedir que los usuarios autorizados accedan a la información, la misma debe estar disponible en todo momento.

- Debe quedar constancia de quién, desde donde, y cuando se realizó una operación determinada en el sistema.
- La base de datos deberá estar disponible las 24 horas, pudiendo realizar operaciones sobre la misma en cualquier momento que se necesite. En caso de una falla en el servidor (interrupción del servicio eléctrico, desconexión momentánea o caída de la red) se deberá contar con un respaldo para cada módulo que le permita seguir con sus actividades.

3.5.2 Restricciones de acuerdo a la estrategia de diseño

- El diseño de las aplicaciones se hará utilizando la Programación Orientada a Objetos (POO). Encapsulación de la lógica por clases.
- Diseño e implementación de una arquitectura flexible, que permita la fácil integración o desintegración de componentes.
- La arquitectura debe soportar migrar la interfaz de usuario de forma rápida.
- El patrón arquitectónico que se debe emplear en el desarrollo es el modelo-vista-controlador y el patrón orientado a objetos.
- El lenguaje de programación que se debe utilizar es PHP para la aplicación web y Java para el subsistema que será una aplicación de escritorio.
- Los protocolos para la comunicación que se deben usar son HTTP entre el cliente y el servidor web y TCP/IP entre el cliente y el servidor streaming.

3.5.3 Portabilidad, escalabilidad, reusabilidad

- El sistema deberá hacer un uso racional de los recursos de hardware de la máquina, sobre todo en las PC clientes.
- Se desarrollará cada pieza del sistema en forma de componentes (elementos) con el fin de ser utilizados en futuras versiones del sistema.
- La documentación de la arquitectura deberá ser reutilizable para poder documentarla como una familia de productos.

3.5.4 Redes

- La red existente en las instalaciones debe de soportar la transacción de paquetes de información de al menos unas 10 máquinas a la vez.

- Para hacer más fiable la aplicación debe de estar protegida contra fallos de corriente y de conectividad, para lo que se deberá parametrizar los tiempos para realizar copias de seguridad. Implementar las transacciones de paquetes en la red con el protocolo TCP/ IP que permite la recuperación de los datos.

3.5.5 Soporte

- El sistema permitirá la modificación o agregarle módulos cuando sea necesario, asegurando su extensibilidad y lograr mejores prestaciones.
- El soporte y/o mantenimiento del sitio no debe detener el servicio.
- La capa del modelo, debe soportar una migración del SGBD en proyecciones futuras.

3.5.6 Usabilidad

- Disponibilidad de visualizar los archivos multimedia en un reproductor.
- Mostar la información de forma lógica y correctamente estructurada.
- El servidor streaming debe mantener buenas prestaciones (demanda de archivos).

3.5.7 Apariencia o Interfaz Externa

- Interfaz amigable, interactiva, intuitiva y de fácil comprensión para el usuario.

3.5.8 Rendimiento

- El sistema debe responder en un tiempo relativamente rápido a las peticiones del usuario (menos de 5 segundos).

3.5.9 Requerimientos de Software

- Se debe utilizar como servidor web el Apache.
- Se debe utilizar como servidor streaming el Darwin Streaming Server.
- Se debe utilizar para captura y transmisión en vivo el Video Lan.
- Se debe utilizar como sistema operativo en los servidores el Ubuntu.
- Para la reproducción se los archivos multimedia se debe utilizar el reproductor VLC, como reproductor externo y como *plugin* embebido en el navegador, por ser multiplataforma y soportar tanto el protocolo de transmisión como el formato de fichero empleado.

3.5.10 Requerimientos de Hardware

Servidor BD

- Deberá tener 2 GB de memoria tipo RAM como mínimo.
- Procesador Pentium IV 3.0 GHz.
- Un Disco duro (HDD) 160 GB.
- Tarjeta de red 100mbps.

Servidor streaming

- Debe contar con un microprocesador Core 2 duo a 1 GHz de velocidad de procesamiento.
- Deberá poseer 1 GB de memoria tipo RAM como mínimo.
- Sistema operativo Ubuntu Linux.
- Un HDD 40 GB.
- Tarjeta de red 1GB Ethernet.

Servidor Aplicación

- Deberá tener 2 Gb de memoria tipo RAM como mínimo.
- Procesador Pentium IV 3.0 GHz.
- Un HDD 160 GB.
- Tarjeta de red 100mbps.

Servidor de medias

- Deberán tener 1 GB de memoria tipo RAM como mínimo.
- Procesador Pentium IV 3.0 GHz.
- Un HDD 1 TB.
- Tarjeta de red 1GB Ethernet.

Servidor de transferencia

- Deberá tener 2 GB de memoria tipo RAM como mínimo.
- Procesador Core 2 Duo / 2.33 GHz.
- Un HDD 160 GB.
- Tarjeta de red 100mbps.

Servidor VoD

- Deberá tener 4 Gb de memoria tipo RAM como mínimo.
- Procesador Core 2 Duo 1.6 GHz.
- Un HDD 160 GB.
- Tarjeta de red 1GB Ethernet.

Pc Cliente

- Debe contar con un procesador de 1 GHz de velocidad.
- Memoria RAM mínima 256 MB.
- Tarjeta de red 100 Mbps.

3.6 Conclusiones

Con el desarrollo de este capítulo se generaron las vistas de la arquitectura que son de mucha importancia para la evolución del sistema. Además se definieron los requerimientos no funcionales con el propósito de que la plataforma funcione de manera eficiente. Se obtuvo una explicación más detallada del sistema que posibilita una guía para los miembros del equipo. También se especificó las tecnologías a utilizar en cada una de las capas del patrón MVC.

Capítulo 4
Evaluación de la Arquitectura del Sistema

4.1 Introducción

Realizar la evaluación de la arquitectura de software de un sistema es una tarea fundamental para conocer el impacto que tiene sobre los atributos de calidad definidos. Durante esta evaluación se pueden conocer los diferentes riesgos asociados con el desarrollo del sistema y como la arquitectura es un producto anticipado son muchos los errores que se pueden corregir durante el flujo de requerimiento o diseño, siendo estos menos costosos que si tuvieran que aclararse en implementación o prueba. En este capítulo se plantean dos estrategias de prueba a emplear para la evaluación de la arquitectura y se valoran dos sistemas con arquitecturas similares a la del proyecto que se desarrolla para comparar los resultados obtenidos.

4.2 Propósito de evaluar una arquitectura de software

El principal objetivo de evaluar una arquitectura de software es conocer si puede permitir los requerimientos especificados, atributos de calidad y restricciones para asegurar que el sistema a ser construido cumple con las necesidades de los stakeholders.

El propósito de realizar evaluaciones a la arquitectura, es analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad. Cabe señalar que los requerimientos no funcionales también son llamados atributos de calidad. **(25)**

Un atributo de calidad es una característica de calidad que afecta a un elemento. Donde el término característica se refiere a aspectos no funcionales y el término elemento a componente.

4.3 Evaluando la arquitectura de software

Para realizar la evaluación de la arquitectura de software, un aspecto fundamental a tomar en consideración es el momento que se realiza. Existen dos variantes útiles según Kazman **(26)** para realizar la misma, la evaluación temprana y la evaluación tardía.

- **La evaluación temprana:** Para realizar este tipo de evaluación no es necesario que la arquitectura se encuentre completamente realizada. Permite efectuar decisiones sobre la arquitectura en cualquier nivel, puesto que se pueden atribuir cambios arquitectónicos producto de una evaluación en función de los atributos de calidad esperados.
- **La evaluación tardía:** Para realizar este tipo de evaluación es necesario que la arquitectura del sistema se encuentre establecida y su implementación esté concluida, es decir, en el momento que el sistema ya esté terminado. Se considera que la evaluación en este punto es muy importante y útil, puesto que puede observarse el cumplimiento de los atributos de calidad asociados al sistema y su comportamiento general.

4.4 Resultados que se obtienen al evaluar una arquitectura de software

La evaluación de la arquitectura de software no define si la arquitectura es buena o mala, simplemente expresa donde se encuentran los riesgos y fortalezas de la arquitectura de software. Esta evaluación produce un informe, el cual puede ser diferente en dependencia del método utilizado. Este informe produce respuesta a dos tipos de preguntas fundamentalmente:

- ¿Es esta arquitectura adecuada para el sistema para el cual fue diseñada?
- ¿Cuál de las arquitecturas propuestas es la más adecuada para el sistema?

Entonces una arquitectura es adecuada cuando cumple dos criterios:

- El sistema resultante cumple con los atributos de calidad definidos.
- El sistema puede ser construido con los recursos disponibles para el mismo.

Lo fundamental en la evaluación es conocer como los atributos de calidad son afectados por las decisiones del diseño arquitectónico.

4.5 Atributos por los cuales puede ser evaluada la arquitectura de software

Los atributos de calidad son requerimientos del sistema que hacen referencia a características que éste debe satisfacer. Se pueden clasificar en:

- **Observables vía ejecución o externos:** aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. (3) (Tabla #6)
- **No observables vía ejecución o internos:** aquellos atributos que se establecen durante el desarrollo del sistema. (3) (Tabla #7)

Algunos de estos atributos son:

Atributo de Calidad	Descripción
Disponibilidad	Es la medida de disponibilidad del sistema para el uso. (27)
Confidencialidad	Es la ausencia de acceso no autorizado a la información. (27)
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido. (26)
Desempeño	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (28)
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo. (27)
Seguridad externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información. (27)
Seguridad interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos. (26)

Tabla 3: Descripción de atributos de calidad observables vía ejecución

Atributo de Calidad	Descripción
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema. (29)
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. (3)
Integridad	Es la ausencia de alteraciones inapropiadas de la información. (27)
Interoperabilidad	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de *integrabilidad.* (3)
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema. (29)
Mantenibilidad	Es la capacidad de someter a un sistema a reparaciones y evolución (27). Capacidad de modificar el sistema de manera rápida y a bajo costo. (29)
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos. (26)
Reusabilidad	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones. (3)
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (9)
Capacidad de Prueba	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba. (3)

Tabla 4: Descripción de atributos de calidad no observables vía ejecución

4.6 Principales métodos de prueba de arquitectura de software

Existen numerosos métodos para realizar pruebas a la arquitectura de software, cada uno con sus características determinadas. Es necesario tener en cuenta las particularidades fundamentales del software para escoger el método ideal, teniendo en cuenta las fortalezas y debilidades del mismo. Entre los principales métodos se encuentran: Software Architecture Analysis Method (SAAM), Active

Design Review (ADR), Architecture Tradeoff Analysis Method (ATAM) y Active Review Intermediate Designs (ARID).

4.6.1 ATAM

Es un método de identificación de riesgos, un medio de detectar áreas de riesgo potencial dentro de la arquitectura de un sistema intensivo de software complejo. Dejar ver la forma en que una arquitectura satisface ciertos atributos de calidad, provee una visión de la forma que interactúan estos atributos con otros. ATAM se inspira en las áreas de estilos arquitectónicos, análisis de atributos de calidad y el método de evaluación SAAM.

No presenta ninguna restricción con respecto a la característica de calidad a evaluar. Es relativamente barato y rápido (porque se trata de evaluar el diseño de la arquitectura de los artefactos). El análisis de ATAM produce resultados en consonancia con el nivel de detalle de la especificación de la arquitectura. Además, no es necesario producir análisis detallados de cualquier atributo de calidad del sistema (como el tiempo medio de retardo o el tiempo de fallo) para tener éxito. (26)
ATAM consta de nueve pasos, divididos en cuatro fases (Presentación, Investigación y Análisis, Pruebas y Reporte).

4.6.2 ARID

Es un híbrido entre los métodos ADR y ATAM, constituye un procedimiento conveniente para la evaluación de diseños parciales en las etapas tempranas del desarrollo usando técnicas de evaluación basada en escenarios.

Utiliza para la evaluación del diseño unidades del software como los componentes o módulos. Las preguntas giran en torno a la calidad, completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto. Define los roles de arquitecto, equipo de verificación y stakeholders y comprende nueve pasos agrupados en dos fases (Actividades Previas y Evaluación).

El método ARID evalúa el grado en que los atributos de calidad satisfacen en cada uno de los escenarios definidos. Como resultado de la aplicación de dicho procedimiento se obtiene un diseño de alta fidelidad acompañado de una alta familiarización con el diseño de los stakeholders. **(25)**

Este método comprende nueve pasos agrupados en dos fases:

Fases	Pasos
Actividades previas	Identificación de los encargados de la revisión
	Preparar el informe de diseño
	Preparar los escenarios base
	Preparar los materiales
Evaluación	Presentación del ARID
	Presentación del diseño
	Lluvia de ideas y establecimiento de prioridad de escenarios
	Aplicación de los escenarios
	Conclusiones

Tabla 5: Fases de ARID

4.7 Propuesta de estrategia de evaluación de la arquitectura

Para la evaluación de la arquitectura de software de la PTARTV se proponen dos estrategias de evaluación. Se realizará una primera evaluación en etapas tempranas del desarrollo de sistema con el propósito de medir los principales atributos de calidad propuestos y tomar medidas de acuerdo a los resultados arrojados por la misma. La segunda evaluación se efectuará cuando la arquitectura del sistema se encuentre establecida y su implementación esté concluida lo cual podrá arrojar resultados beneficiosos para el conocimiento del grado de cumplimiento de los atributos de calidad propuestos cuando el sistema esté en funcionamiento.

4.7.1 Estrategia de evaluación temprana (ARID)

Se empleará el método de evaluación ARID que posibilita la evaluación en etapas tempranas del desarrollo de software evaluando el grado en que los atributos de calidad satisfacen en cada uno de los escenarios definidos. Se analizarán los atributos de calidad de modificabilidad, interoperabilidad, seguridad, funcionalidad y disponibilidad. Se debe seleccionar al menos tres miembros de equipo de

desarrollo (preferiblemente los de mayor conocimientos sobre las funcionalidades de la aplicación) para conformar el equipo de evaluación.

Se plantea para la selección de los escenarios, que los miembros del equipo de desarrollo designados para el proceso de evaluación establezcan un escenario por cada caso de uso del sistema. En estos escenarios se realizarán las pruebas en correspondencia con los atributos de calidad definidos. A continuación se definirán puntos claves a tener en cuenta durante la realización de la estrategia de evaluación según los atributos de calidad:

Evaluación del atributo modificabilidad:

La modificabilidad es la habilidad de realizar cambios futuros al sistema, se propone entonces valorar las consecuencias y el costo de los siguientes factores:

- Cambiar la interfaz de la aplicación.
- Cambiar el SGBD.
- Cambiar el servidor streaming.

Analizar los requerimientos que puedan cambiar para futuras versiones de la plataforma y valorar la factibilidad de la arquitectura para adaptarse a los mismos.

Evaluación del atributo interoperabilidad

La interoperabilidad es la medida de la habilidad de que un grupo de partes del sistema trabajen conjuntamente con otro sistema, se propone entonces:

El equipo de evaluación designado escogerá las funcionalidades que puedan ser de interés para otras aplicaciones. Valorará el costo de realizar las funcionalidades necesarias para posibilitar la interrelación con otras aplicaciones (preferiblemente usando servicios web).

Se deben tener en cuenta las siguientes funcionalidades:

- Subsistema de transferencia.
- Subsistema web.

El equipo de desarrollo deberá incluir otras funcionalidades.

Evaluación del atributo seguridad

La seguridad es la medida de ausencia de errores que generan pérdidas de información y la habilidad que posee un sistema para resistir a intentos de usos no autorizados y negación del servicio, mientras se sirve a usuarios legítimos, se propone:

- El equipo de evaluación seleccionará diferentes escenarios donde el acceso a los recursos esté limitado a un rol de usuario, al menos dos escenarios por cada rol del sistema. Se probará en cada escenario definido el acceso de usuario con diferente roles.
- Realizar intentos de acceso a la interfaz web de administración del Darwin Streaming Server y el Video Lan por usuarios con privilegios y usuarios sin privilegios.
- Durante la publicación de contenidos, introducir cadenas que contengan códigos que puedan dañar el funcionamiento de la aplicación.
- Durante la transmisión de ficheros, introducir cadenas que contengan códigos que puedan dañar el funcionamiento de la aplicación.
- Introducir cadenas que contengan códigos que puedan dañar el funcionamiento de la aplicación en el formulario de registro.

Evaluación del atributo funcionalidad

La funcionalidad es la habilidad del sistema para realizar el trabajo para el cual fue concebido, se propone:

El equipo de evaluación medirá el grado de cumplimiento de cada uno de los requerimientos funcionales definidos en el sistema.

Evaluación del atributo disponibilidad

Teniendo en cuenta que la disponibilidad es la medida de disponibilidad del sistema para el uso, se propone:

Valorar las consecuencias sobre los servicios cuando se efectúan las acciones siguientes:

- Realizar modificaciones en el SGBD sin afectar la conexión con el resto de la aplicación.
- Realizar modificaciones en la configuración del servidor de Streaming sin afectar la conexión con el resto de la aplicación.
- Realizar modificaciones en el servidor de almacenamiento de medias sin afectar la conexión con el resto de la aplicación.
- Realizar modificaciones en el servidor web sin afectar la conexión con el resto de la aplicación.

- Detener los servicios del SGBD.
- Detener los servicios del servidor de streaming.
- Detener los servicios del servidor de almacenamiento de medias.
- Detener los servicios del servidor web.
- Solicitar el mismo recurso por múltiples usuarios.
- Solicitar recursos independientes por múltiples usuarios.

4.7.2 Estrategia de evaluación tardía (ATAM)

Esta estrategia se realizará una vez que el sistema este implementado completamente y esté puesto en funcionamiento. Se analizarán los atributos de calidad definidos en la evaluación temprana modificabilidad, interoperabilidad, seguridad, funcionalidad y disponibilidad conjuntamente con los atributos de mantenibilidad, reusabilidad y flexibilidad. Se empleara el método de evaluación ATAM que revela la forma en que la arquitectura satisface los atributos de calidad definidos y provee una visión de la forma que estos atributos interactúan entre sí.

ATAM define tres tipos de escenarios:

- Escenarios de casos de uso: describen las interacciones de los usuarios con el sistema.
- Escenarios de crecimiento: representa la anticipación de los cambios.
- Escenarios exploratorios: tienen como objetivo fundamental exponer al sistema a condiciones extremas, llevar el diseño a casos extremos de condiciones.

Este método consta de nueve pasos, divididos en cuatro fases.

Presentación	
Presentación del ATAM	El líder de evaluación describe el método al resto del equipo y se establece el alcance.
Presentación de las metas de negocio	Se realiza la descripción de las metas del negocio que motivan el esfuerzo y aclara que se persiguen objetivos de tipo arquitectónico.
Presentación de la Arquitectura	El arquitecto describe la arquitectura, enfocándose en cómo esta cumple con los objetivos del negocio.

Investigación y análisis	
Identificación de los enfoques arquitectónicos.	Se detectan sin entrar en detalles.
Generación del Utility Tree.	Se priorizan los atributos de calidad que engloban las funcionalidades del sistema, especificados en forma de escenarios. Se tienen en cuenta los estímulos y respuestas, así como se establece las prioridades entre ellos.
Análisis de los enfoques arquitectónicos.	Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos identificados. Se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.
Pruebas	
Lluvia de ideas y establecimiento de prioridad de escenarios.	Con la colaboración de todos los involucrados, se complementa el conjunto de escenarios.
Análisis de los enfoques arquitectónicos.	Este paso repite las actividades del análisis de los enfoques arquitectónicos, haciendo uso de los resultados de la lluvia del paso anterior. Los escenarios son considerados como caso de prueba para confirmar el análisis realizado hasta el momento.
Reporte	
Presentación de los resultados.	Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

Tabla 6: Fases de ATAM

Similar a la estrategia anterior los resultados obtenidos de la evaluación se expondrán ante todos los involucrados en la realización de la PTARTV. Si se detectan dificultades se procederá a valorar la factibilidad de realizar cambios en la arquitectura u otras variantes que contribuyan con la mitigación de estos problemas. De ser necesario se realizarán los cambios y al concluir el mismo se proponen realizar nuevamente la evaluación de la arquitectura empleando el método ATAM.

4.8 Valoración de propuestas arquitectónicas similares

Esta valoración consiste en analizar dos propuestas arquitectónicas en sistemas de equivalentes propósitos, a partir de consultar la bibliografía especializada en este tema estos sistemas son Estructure y Videoma.

4.8.1 Estructure

Es una plataforma audiovisual (Anexo 7) que aporta soluciones tecnológicas avanzadas para el empleo de contenidos audiovisuales. Sobre esta se construyen y configuran sistemas informáticos de gestión y utilización de contenidos multimedia, integrando todas las funcionalidades y tareas necesarias de forma flexible y modular, permitiendo una correcta y ágil organización, explotación y flujo del trabajo.

En Estructure todos los procesos están integrados y se interrelacionan entre sí aprovechando al máximo el trabajo en colectivo. También facilita las tareas mediante una fácil y agradable interfaz de trabajo que provee altos rendimientos de productividad.

Los sistemas se basan en tecnología abierta, lo que permite reducir drásticamente la inversión y utilizar tecnología estándar y probada, e incorporar nuevas tecnologías, facilitando la integración de dispositivos y funcionalidades. Se puede adaptar a las necesidades reales de cada empresa desde un único puesto de trabajo a un gran número de puestos. (30)

A continuación se exponen algunas características pertenecientes a esta plataforma:

- Soporte de múltiples formatos de entrada y salida: H264, MXF, DV, DVCPRO, MPEG2, MJPEG1, MPEG-4, WMW, FLV, 3G, 3G2.
- Captura simultánea en alta y baja calidad.
- Audio Digital Estéreo 48 KHz y analógico.
- Componentes analógicos, Y/C (S-Video) y compuesto.
- Soporte IEEE 1394.

- Arquitectura escalable para múltiples canales en captura y en salida.
- Acceso simultáneo a toda la información del sistema: desde múltiples estaciones de trabajo.
- Configuraciones hasta 150 puestos asegurados.
- Configuración de permisos por módulos y niveles de acceso; usuarios y grupos de usuarios.
- Sincronización automática de todo el sistema de trabajo.
- Guardado automático del trabajo en módulos críticos.
- Detección automática de cambios de plano en las estaciones de captura.
- Bases de datos seguras SQL Server, MySQL u Oracle.
- Programación automática de salvas.
- Plataformas estándares y líderes: Windows 2000, XP y 2003.
- Administración de usuarios, grupos y funciones de cada módulo.
- Componentes hardware compatibles certificados.
- Estándares de tecnología abierta.
- Conexión con robóticas para almacenamiento.
- Potentes motores de indexación.

Mediante estas características se pueden realizar algunas evaluaciones entre la arquitectura de Estructure y la que se propone para la PTARTV.

Estructure presenta una arquitectura ya probada y empleada en varios sistemas relacionados con el tema de las medias que utilizan herramientas propietarias, es una arquitectura en capas lo que facilita que en caso de realizar alguna modificación, sólo se tuvieran que realizar las correcciones necesarias en el nivel requerido sin tener que revisar código de otros niveles, mientras que la arquitectura propuesta para la PTARTV es el MVC.

El acceso a la aplicación Estructure se realiza de forma similar al sistema que se propone en este trabajo o sea con permisos definidos para cada usuario. Además Estructure utiliza programación automática de salvas mientras que en este sistema se propone Bacula para salvas automáticas.
Estructure emplea bases de datos seguras como SQL Server, MySQL y Oracle, el sistema propuesto emplea PostgreSQL que en análisis realizados se ha demostrado que cuanto mayor es el volumen de información más robusto resulta.

Otra aspecto es que Estructure es factible para Windows (2000, 2003, XP) no así para el sistema propuesto, es decir es multiplataforma.

Estructure presenta un buen funcionamiento pero además es muy escalable y esto provoca en varias ocasiones que se descuide la seguridad, y es debido a que en sistema donde ocurren muchos cambios la seguridad es difícil de mantener, no ocurre así en la PTARTV que luego de la funcionalidad lo más importante es la seguridad.

4.8.2 Videoma

Es un producto (Anexo 8) de la empresa española líder en el desarrollo de soluciones software de gestión multimedia ISID que comenzó su proyecto en 1996.

A continuación se exponen algunas características pertenecientes a esta plataforma:

- Arquitectura modular para la captura de vídeo analógico o digital, en tiempo real o diferido.
- Permite el análisis, codificación, clasificación y publicación de un archivo digital.
- Cataloga fácilmente el material audiovisual y lo vincula con otros activos digitales como imágenes, documentos, de forma que su recuperación y localización permite la reutilización de dichos materiales.
- Vincula las versiones digitales de estos contenidos de forma que se puedan obtener, reproducir y reutilizar directamente.
- Su arquitectura Internet/Intranet permite una integración sencilla en redes corporativas que soporten TCP/IP.
- Realiza la captura de la información, y la segmentación automática con inserción de metadata asociado al material a digitalizar, facilitando la necesaria documentación de grandes archivos de información.
- Funcionamiento estable y continuo 24x7.
- Flexible e integrable con otros sistemas utilizados en el entorno de trabajo. (31)

Mediante estas características se pueden realizar algunas evaluaciones entre la arquitectura de Videoma y la que se propone para la PTARTV.

La arquitectura del sistema varía en función de las fuentes de entrada que se tengan, de la cantidad de almacenamiento que se quiera mantener en línea, de los formatos a los que se hayan digitalizado los contenidos y de la cantidad de información que se vayan a mover a través de la red local.

Videoma utiliza aplicaciones estándar para la programación de sus módulos, PHP y C++. Como base de datos soporta Oracle 9i y MySQL y como Servidor Web IIS y Apache. El resto de necesidades para

poner en marcha el sistema pueden ser aquellas disponibles en el mercado, como diversos sistemas de almacenamiento, diversos servidores de datos, diversos codificadores de vídeo, etc.

Videoma es una aplicación web desarrollada con los lenguajes PHP y C++ mientras PTARTV está dividida en una aplicación de escritorio con Java y una aplicación web con symfony. Videoma utiliza como gestores de base de datos Oracle o MySQL, no así el sistema propuesto que utiliza PostgreSQL, ambos sistemas realizan replica en cada una de las PCs para trabajar con o sin conexión a la BD. En cuanto a la administración, los sistemas emplean la misma forma de acceso a la aplicación es decir mediante usuario y contraseña y con sus respectivos permisos para interactuar con el sistema. Ambos sistemas presentan sistema de salva automático de los datos. Los dos sistemas consideran la seguridad como un aspecto importante pero en el caso de Videoma se considera muy escalable y este aspecto posibilita que el sistema sea poco seguro por la cantidad de modificaciones. En la selección de herramientas existen algunas diferencias, casi todas las empleadas por Videoma son para dar soporte en Windows no así en el sistema PTARTV que esta selección se realizó para dar soporte en cualquier plataforma.

4.9 Conclusiones

En el presente capítulo se han analizado los elementos fundamentales relacionados con la evaluación de la Arquitectura de Software.

Se elaboró una propuesta de evaluación de la arquitectura en etapas tempranas del desarrollo usando el método ARID, definiendo los atributos de calidad a medir para mitigar los errores que puedan aparecer, con el objetivo de reducir el impacto de estos en el futuro. De igual forma se define una estrategia de prueba para la etapa tardía empleando el método ATAM, para medir los atributos de calidad definidos y comprobar el grado de cumplimiento de los mismos en el producto final.

Se realizó una valoración con dos sistemas de similar propósito donde los resultados obtenidos en el sistema propuesto son admisibles. Entre los sistemas escogidos no hubo ninguno nacional pues en Cuba no existen aplicaciones que resulten similares a PTARTV, por lo que se decidió hacer la comparación con dos que tuviesen gran prestigio internacional. A partir de esta comparación se demostró que todos están basados en patrones arquitectónicos ya probados, que Estructure y Videoma por ser tan escalables descuidan la seguridad mientras PTARTV la fortalece.

Conclusiones Generales

En el presente trabajo se analizaron los procesos asociados a la Plataforma de Transmisión Abierta para Radio y Televisión. Se analizaron las funcionalidades de plataformas que brindan servicios de audio y video a través de la web, entre ellas yahoo.com, es.youtube.com, TV.com de carácter internacional e Inter-nos en Cuba, de los cuales se pudo apreciar un conjunto de funcionalidades claves para un producto de este tipo.

Se realizó un estudio de los principales patrones arquitectónicos y se seleccionó para la solución el patrón Modelo Vista Controlador, definiéndose sus principales componentes, configuraciones y restricciones.

La selección de las herramientas de modelado, las herramientas de desarrollo y los requisitos no funcionales se realizó a partir de las características o funcionalidades que debía brindar la PTARTV.

A través de la descripción de la arquitectura mediante las vistas definidas por RUP se garantizó que existiera una visión de todos los modelos del sistema y de los elementos arquitectónicamente significativos durante el proceso de desarrollo.

Con el desarrollo del trabajo se generaron los artefactos necesarios que permiten la implementación del sistema a desarrollar.

Recomendaciones

Se recomienda para ampliar la presente investigación el refinamiento constante de la arquitectura propuesta durante el ciclo de desarrollo. Además se recomienda el uso de Bacula solo para los ficheros de programación y realizar la evaluación de la arquitectura mediante las estrategias de evaluación definidas.

Trabajos citados

1. **Corredoira y Alfonso, Loreto.** http://www.universia.es. [En línea] [Citado el: 21 de 11 de 2009.]

2. **Sarduy Domínguez, Urra González.** bvs. *Sistemas de gestión de contenidos: En busca de una plataforma ideal.* [En línea] [Citado el: 15 de 10 de 2009.] http://bvs.sld.cu/revistas/aci/vol14_4_06/aci11406.htm.

3. **Bass, Len, Clements, Paul and Kazman, Rick.** *Software Architecture in Practice.* Addison Wesley : Second Edition, Abril 11,2003. 0-321-15495-9.

4. **Hilliard, Rich.** IEEE. *Recommended Practice for Architectural Description of Software-Intensive Systems.* [En línea] http://www.standards.ieee.org/reading/ieee/std_public/.../se/1471-2000_desc.html.

5. **Frank Buschmann, Regine Meunier, Hans Rohnert , Peter Sommerlad ,Michael Stal.** *Pattern-Oriented Software Architecture.* England : Wiley; 1 edition (August 8, 1996), 1996. 0 417 95869 7.

6. **II, Ingeniería de Software.** *Arquitectura y Patrones de diseño.* Habana, Cuba : Semana:3, Conferencia #2, Curso 2009-2010. eva.uci.cu.

7. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999. 84-205-3438-2.

8. **Corporation, Microsoft.** *La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real.* s.l. : Microsoft Corporation, Diciembre 2006.

9. **Pressman, Roger Putnam y Myers.** *Five core metrics.* s.l. : Dorset House, 2003.

10. **S, Rosenblum Nenad Medvidovic y David.** *Domains of Concern in Software Architectures and Architecture Description Languages.* California : s.n., 1997.

11. UDLAP Universidad de las Americas Puebla. [En línea] [Citado el: 18 de enero de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo_5.html#.

12. **Garrett, Jesse James.** Adobe user group Cordoba AUG CBA. *Ajax: Un Nuevo acercamiento a las Aplicaciones Web.* [En línea] 18 de febrero de 2005. [Citado el: 18 de enero de 2010.] http://mmugcba.com.ar/foro/viewtopic.php?id=36.

13. **Austerberry, David.** *The Technology of Video and Audio Streaming Second Edition.* s.l. : Focal Press, 2005. 0-240-80580-1.

14. El lenguaje HTML. [En línea] [Citado el: 15 de enero de 2010.] http://www.xpps.net/contenido_xppsnet/areatec/HMTL.pdf.

15. **Kopylenko, Johnson/ Hoeller/ Arendsen/ Rsiberg/.** *Professional Java Development With The Spring Framework.* Texas : John Wiley & Sons, 2004. 764574833.

16. **Fabien Potencier, François Zaninotto.** *Symfony, la guia definitiva.* 2008.

17. DotConsulting. *Symfony: framework por excelencia para PHP.* [En línea] [Citado el: 17 de enero de 2010.] http://www.dotconsulting.com.ar/blog/?p=6.

18. **Gavin, Christian Bauer y King.** *Hibernate in Action.* United States of America : Manning Publications Co, 2005. 1932394-15-X.

19. **Group, PostgreSQL Global Development.** PostgreSQL. [En línea] 1996. [Citado el: 3 de febrero de 2010.] http://www.postgresql.org/.

20. **Ballester, Eva Gómez.** Universidad de Alicante. *Bases de Datos 1.* [En línea] [Citado el: 17 de enero de 2010.] http://www.alu.ua.es/j/jmr36/Conectate/Base%20Datos/Apuntes2006.pdf.

21. **PostgreSQL, Equipo de desarrollo de.** TLDP. *Manual de Usuario de PostgreSQL.* [En línea] [Citado el: 17 de enero de 2010.] http://es.tldp.org/Postgresql-es/web/navegable/ user/user.html.

22. masadelante.com. *¿Qué es un servidor web?* [En línea] [Citado el: 31 de enero de 2010.] http://www.masadelante.com/faq-sistema-operativo.htm.

23. Softonic. [En línea] [Citado el: 3 de febrero de 2010.] http://rapidsvn.softonic.com/.

24. **abierta, Bacula. La solución de red de origen de copia de seguridad.** bacula.org. [En línea] [Citado el: 11 de febrero de 2010.] http://www.bacula.org/.

25. **Gómez, Omar Salvador.** *Evaluando Arquitecturas de Software. Parte 1. Panorama General.* México : Brainworx S.A, 2007. 1870-0888.

26. **Olson, Rich Kazman y David.** *From requirements negotiation to software architectural decisions.International Workshop from Software Requirements to Architectures.* Toronto, Canada : s.n., 2001.

27. **M, Barbacci.** *Quality Attributes.* Pittsburgh : Software Engineering Institute, Carnegie Mellon University, 1995.

28. IEEE. *IEEE-Std-610.12-1990.* [En línea] 1990. http://www.standards.ieee.org.

29. **J, Bosch.** *Building Application Frameworks, chapter Object-oriented frameworks - Problems & Experiences.*

30. **Systems, Estructure Media.** Estructure. [En línea] [Citado el: 8 de abril de 2010.] http://www.estructuretv.com/.

31. ISID. [En línea] [Citado el: 5 de abril de 2010.] http://www.isid.es/.