Universidad de las Ciencias Informáticas
Facultad 9

Desarrollo de la arquitectura de la Plataforma de Televisión Informativa PRIMICIA

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:

Douglas Páez Castillo

Tutor:

Ing. Ruber Hernández García

Ciudad de La Habana, mayo de 2009
"Año 50 de la Revolución"

DEDICATORIA

A mis padres por ser mis guías en todo momento.

Mami te agradezco por todo lo que has hecho en la vida por mí, solo darme la posibilidad de venir al mundo fue lo más hermoso que cualquier ser humano puede pedir. Muchas gracias, siempre estarás conmigo. Por la educación que siempre me has enseñado, y estar en los momentos cuando más te he necesitado.

A mi padre por engendrarme todo el respeto, el amor y el cariño que un hijo pueda desear. Este es un sueño hecho realidad para él, a ti te lo dedico. Gracias por tu respeto y guía en las decisiones que he tomado.

A mis abuelos por ser el gran ejemplo en mi vida, en especial a mi abuelo Tata por apoyarme y ayudarme en todo momento, por su ejemplo de sacrificio y sabiduría.

En especial también a mis amistades de la UCI, por acompañarme estos 5 maravillosos años, a mis grandes amigos Julio y Daniel por su confianza y apoyo incondicional en todo momento, a todos los que me apoyaron de una manera u otra… muchas gracias!!!

A todos muchas gracias

Douglas

AGRADECIMIENTOS

Primeramente a la Revolución cubana y a nuestro Comandante en Jefe Fidel Castro Ruz, por su brillante idea de crear esta Universidad.

A mis padres y familiares que tanto esfuerzo y empeño han puesto por ver este sueño hecho realidad.

A mis compañeros y amigos por estar juntos estos maravillosos 5 años…

A los que se van, a los que se quedan y a los que no llegaron, suerte en la vida…

A mi tutor, y todos los que de una forma u otra hicieron posible la realización de este trabajo.

RESUMEN

El presente trabajo de diploma está enmarcado en el producto Plataforma de Televisión Informativa, PRIMICIA, de la Universidad de las Ciencias Informáticas, en la Facultad 9. La solución propuesta es diseñar una Arquitectura de Software que de cumplimiento a las funcionalidades del producto de sistema informativo.

Este documento contiene un estudio de los fundamentales elementos que constituyen la Arquitectura de Software, partiendo de sus principales conceptos, tendencias de los patrones y estilos arquitectónicos, además de las herramientas de modelado y desarrollo, así como los frameworks de desarrollo para la Web, Entorno de Desarrollo (IDE), lenguajes de programación y Sistema Gestor de Base de Datos.

El producto PRIMICIA está compuesto por dos subsistemas, con distintas responsabilidades cada uno. A través del Documento de Descripción de la Arquitectura se describe la solución arquitectónica del sistema, incluyendo además otros artefactos como son las vistas arquitectónicas definidas por la metodología RUP.

Palabras claves: *arquitectura de software, PRIMICIA, televisión.*

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Los seres humanos tienen la necesidad de comunicarse e intercambiar información lo que ha proporcionado un auge en el desarrollo de los medios de comunicación dado el avance tecnológico de la época.

El avance continuo de las Tecnologías de la Información y la Comunicación (TIC) ha contribuido a que los sistemas de almacenamiento se desarrollen y pasen a ser servidores informáticos. La existencia de los servidores de video posibilita la automatización de las emisiones y de los programas informativos mediante la realización de listas de emisión.

La televisión informativa es aquella dedicada a la transmisión de informaciones de forma constante en diferentes formatos. En la Universidad de las Ciencias Informáticas (UCI) se ha logrado desarrollar varias versiones de sistemas de este tipo que han dado como resultados canales informativos.

La Plataforma de Televisión Informativa (PRIMICIA) constituye una solución integral capaz de proveer un canal de televisión informativa, con posibilidades de usar en distintos entornos tales como hoteles, sedes ministeriales, terminales, escuelas o cualquier otro tipo de institución que necesite mantener informado de manera rápida y constante a un grupo de personas. PRIMICIA permite la transmisión automática de informaciones en formato texto, imagen, audio y video, de acuerdo a las publicaciones realizadas a través de la administración del sistema.

Los primeros sistemas informativos desarrollados en la UCI fueron creados utilizando herramientas privativas, lo cual va en contraste a las condiciones del bloqueo económico implantado a Cuba.

Muchos han sido los sistemas informativos desarrollados internacionalmente, los cuales implementan variadas alternativas de arquitectura para el desarrollo de este tipo de software. Por lo que se propondrá una arquitectura que de solución a la demanda del producto, como por ejemplo el hardware óptimo y la cantidad de servidores necesarios para el correcto funcionamiento, teniendo en cuenta que debe ser adaptable a cualquier local, área, región o situación física donde se desee instalar la plataforma.

En el documento se hace énfasis en la arquitectura ideal para el desarrollo del producto PRIMICIA pero es bueno destacar que en conjunto con este producto debe interactuar un sistema operativo libre que se adapte a las funcionalidades del producto.

De acuerdo a lo planteado anteriormente y a la importancia que tiene para el desarrollo de software la definición de una correcta arquitectura, la principal tarea de este trabajo estaría encaminada a resolver el siguiente **problema**:

No se encuentra definida una arquitectura robusta y confiable, con el uso de herramientas libres, que permita el desarrollo de la Plataforma de Televisión Informativa PRIMICIA.

De esta manera se propone como **objetivo general** definir una arquitectura basada en herramientas libres que permita el desarrollo de PRIMICIA de manera que el sistema cumpla con los requisitos establecidos.

Para cumplir satisfactoriamente el objetivo general del trabajo se trazan los siguientes **objetivos específicos**:

- Determinar la arquitectura de software idónea para que el producto PRIMICIA como sistema informativo.
- Valorar todas las posibles herramientas libres con las cuales se pueda desarrollar el producto PRIMICIA.
- Establecer la línea base de la arquitectura.
- Desarrollar las actividades del rol de arquitecto durante el ciclo de vida del producto.

El **objeto de estudio** del trabajo se enmarca en las arquitecturas de software usadas para el desarrollo de sistemas informativos para televisión. El **campo de acción** se centra en el desarrollo de la arquitectura de software de la Plataforma de Televisión Informativa PRIMICIA.

A partir de lo planteado la **idea a defender** es que si se define una línea base de arquitectura coherente con las funcionalidades del sistema se puede lograr una mayor confiabilidad en la calidad del producto final y se garantiza la total facilidad de soporte del sistema.

Al concluir el trabajo de diploma se espera obtener como **posibles resultados** prácticos la línea base de la arquitectura de PRIMICIA, la definición de las herramientas a utilizar para el desarrollo del sistema y

configuración de los puestos de trabajo y los artefactos desarrollados por el rol Arquitecto durante las fases de desarrollo.

Para llevar a cabo la investigación se plantean las siguientes **tareas**:

1. Evaluar alternativas de arquitecturas para la implementación de sistemas informativos para televisión, teniendo en cuenta los presentes en el área nacional e internacional.
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. Definir de las herramientas a utilizar para el desarrollo del sistema y configuración de los puestos de trabajo.
4. Elaborar la propuesta de línea arquitectónica a utilizar en el desarrollo del sistema.
5. Modelar los artefactos correspondientes al rol Arquitecto en el desarrollo de PRIMICIA.
6. Comparar la arquitectura propuesta con dos casos donde sea utilizada en otros proyectos de desarrollo de software.

En la etapa investigativa del presente trabajo se utilizaron varios métodos de investigación, tanto teóricos como empíricos. Fue necesario evaluar todas las posibles arquitecturas que existen en el mundo sobre sistemas informativos y estudiar el estado del arte de los patrones y estilos arquitectónicos para encontrar el más idóneo para la Plataforma de Televisión Informativa (***histórico - lógico***).Durante esta etapa resultó imprescindible resumir los documentos referenciados con el objetivo de plasmar la información encontrada en las diferentes bibliografías de forma más analizada, madura y elaborada para lograr un mejor entendimiento del contenido (***analítico - sintético***).

Se emplearon además otros métodos como la ***observación*** y la ***entrevista***. El primero se utilizó en el estudio de algunos sistemas informativos en funcionamiento con el fin de analizar la interacción de la arquitectura. Fueron entrevistadas profesionales capacitados en temas de arquitectura de software para obtener información de algunos contenidos específicos.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se aborda el análisis del estado del arte de la Arquitectura de Software, partiendo de un estudio de los principales conceptos de la arquitectura. También se hace un análisis sobre los principales sistemas informativos similares al producto PRIMICIA.

1.2 La Arquitectura de Software

Existen muchas definiciones de Arquitectura del Software como:

…"Arquitectura de software es el estudio de la estructura a gran escala y el rendimiento de los sistemas de software. Aspectos importantes de la arquitectura de un sistema incluye la división de funciones entre los módulos del sistema, los medios de comunicación entre módulos, y la representación de la información compartida." (1)

"La arquitectura de software 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". (2)

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." (3)

Ninguna de ellas haya sido totalmente aceptada por los estudiosos del tema. En un sentido amplio se podría estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales
- Definir las responsabilidades que tendrá cada uno de estos módulos
- Definir la interacción que existirá entre dichos módulos:
- Control y flujo de datos
- Secuenciación de la información
- Protocolos de interacción y comunicación
- Ubicación en el hardware

La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

La Arquitectura del Software se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiarnos en el desarrollo del software dentro de un sistema informático.

Como ejemplos de Arquitecturas se puede citar las monolíticas (grupos funcionales del software que están altamente acoplados entre sí), cliente-servidor (se reparte la carga de cómputo en dos partes independientes), y la arquitectura de tres niveles (la carga se divide entre tres partes: presentación, cálculo y almacenamiento).

1.3 Objeto de Estudio

Se enmarca en las arquitecturas de software usadas para el desarrollo de sistemas informativos para televisión. En el documento se hace referencia a los principales sistemas informativos que existen en el mundo, y que de ellos se vea información sobre su arquitectura, para así poder hacer un análisis profundo y seleccionar aquellas ventajas que pudieran beneficiar a la Plataforma de Televisión Informativa.

1.3.1 Descripción General

El sistema informativo PRIMICIA se debe desarrollar completamente con herramientas de software libre. El sistema se encuentra estructurado en dos subsistemas que se relacionan entre sí y actúan como un todo para brindar un resultado final a través de una red de televisión. El Subsistema de Administración es el responsable de que a través de él se realice la administración del canal y toda la gestión de las noticias y recursos multimedia. El Subsistema de Transmisión es el encargado de visualizar las noticias y materiales publicados.

La Plataforma de Televisión Informativa provee un canal de televisión el cual es soportado y transmitido utilizando computadoras. La solución ha sido pensada para que sea adaptable a distintos entornos.

1.3.2 Descripción actual del dominio del problema

Actualmente el sistema PRIMICIA abarca el contexto o dominio referente a la gestión noticias y recursos multimedia, constituyendo una solución integral para la transmisión de un canal informativo. Actualmente existen varios sistemas informativos (alternativas que se abordaran más adelante) que se enmarcan en darle solución de llevar información en tiempo real a un lugar determinado, ya sea por ser un local cerrado donde se encuentre un cúmulo de personas, o porque la prensa escrita no satisface las necesidades de los clientes.

1.4 Situación Problemática

La necesidad de los seres humanos de comunicarse e intercambiar información de forma continua ha llevado a desarrollar medios de comunicación como la radio y la televisión según el avance de la tecnología de la época.

Hay situaciones especiales en que es necesario llevar a una gran masa de personas un conjunto de información en diferentes formatos: texto, imagen, audio o video. Este es el caso de un entorno cerrado donde no es posible emplear los medios tradicionales de comunicación, la situación se enmarca en sedes ministeriales o empresariales, universidades, hoteles, aeropuertos, entre otros.

El avance continuo de las Tecnologías de la Información y la Comunicación (TIC) ha contribuido al desarrollo de sistemas de producción basado en el tratamiento informático de la señal de televisión. Los sistemas de almacenamiento, como los magnetoscopios, pasaron a ser sustituidos por servidores informáticos de video y los archivos pasaron a guardar sus informaciones en discos duros y cintas de datos. La existencia de los servidores de video posibilita la automatización de las emisiones y de los programas informativos mediante la realización de listas de emisión.

Durante cinco años en la Universidad de las Ciencias Informáticas un grupo de personas se ha dedicado al desarrollo de sistemas informáticos para dar solución a la transmisión de canales de televisión utilizando computadoras, entre las soluciones desarrolladas se pueden mencionar sistemas para la transmisión automática de televisión tradicional y sistemas automatizados para la transmisión de televisión informativa.

La televisión informativa es aquella dedicada a la transmisión de informaciones de forma constante en diferentes formatos. En la UCI se ha logrado desarrollar varias versiones de sistemas de este tipo que han dado como resultados canales informativos como canal de la universidad Señal 3 y el canal de teletexto de la Plataforma Satelital Cubana Señal ACN, el cual está dirigido a los cooperantes cubanos en el extranjero y a los habitantes de las zonas de silencio de la geografía cubana como soporte alternativo de la prensa plana nacional. El desarrollo de las anteriores soluciones de software ha posibilitado la conceptualización de un producto informático para darle solución a la automatización de la transmisión de televisión informativa, de esta manera se ha decidido desarrollar la Plataforma de Televisión Informativa, PRIMICIA.

PRIMICIA se ha conceptualizado como una solución integral capaz de proveer un canal de televisión informativa que integra informaciones en formato texto, imagen, audio y video, posibilitando la transmisión automática de las informaciones de acuerdo a las publicaciones realizadas a través de la administración del sistema.

Los primeros sistemas informativos desarrollados en la UCI fueron creados utilizando herramientas privativas, lo cual va en contraste a las condiciones del bloqueo económico implantado a Cuba, las posibles adquisiciones de las licencias necesarias para la utilización legal de estas tecnologías, así como las directivas de software libre establecidas en la universidad y el país para el desarrollo de software. De

esta manera se hace necesario que el desarrollo del producto sea en software libre, para que brinde la oportunidad de comercializarlo sin interferencias de patentes, licencias u otros términos legales que impida su exportación.

Muchos han sido los sistemas informativos desarrollados internacionalmente, los cuales implementan variadas alternativas de arquitectura para el desarrollo de este tipo de software. Por lo que existe la necesidad primeramente de una arquitectura que de solución a la demanda del producto, como por ejemplo el hardware óptimo y la cantidad de servidores necesarios para el correcto funcionamiento, teniendo en cuenta que debe ser adaptable a cualquier local, área, región o situación física donde se desee instalar la plataforma. Estos entornos pueden tener diferentes características en cuanto a la distribución de las tecnologías que soportan las aplicaciones, la más significativa está dada por la separación en distancia de los locales de gestión y transmisión del canal. De igual manera se plantea la necesidad de crear una arquitectura distribuida que defina componentes y subsistemas de software que interactúen entre sí y permitan la escalabilidad, soporte y reutilización de la plataforma.

Una solución integral como se define la Plataforma de Televisión Informativa incluye componentes tanto de software como hardware que deben estar alineados para lograr un rendimiento óptimo del sistema. Teniendo en cuenta esto se puede afirmar que es necesario obtener un sistema operativo libre que se adapte a las funcionalidades del producto. La intención de este sistema operativo es que funcione integralmente con el sistema teniendo solo las funcionalidades necesarias y optimizadas para lograr un mejor rendimiento.

1.5 Alternativas de arquitecturas para la implementación de sistemas de informativos para televisión.

Muchas han sido las alternativas de arquitectura para los sistemas informativos, las cuales dan solución según la problemática planteada en cada lugar de trabajo. Existen alternativas de este tipo de software, tanto nacional como internacional. A continuación se presenta algunas de las más relevantes que pueden contribuir con elementos al desarrollo de la Plataforma de Televisión Informativa. Dentro de todas las aplicaciones existentes solo se analizan Canal Informativo Señal ACN, VSN Matic, Estructure y Escripnauta, debido a que no todas dan características técnicas sino rasgos comerciales.

1.5.1 Alternativas Internacionales

1.5.1.1 VSN Matic

Otra alternativa a sistemas informativos es VSN que es una plataforma de software de automatización y servidor de vídeo de bajo coste para TV. Incluye módulos de noticias, ingesta, edición en red, archivo, gráficos, publicación en Web, continuidad, grabación legal y SMS TV.

VSN desarrolla video servidores económicos y software de automatización broadcast[1] para sistemas abiertos y escalables. Ofrece soluciones para producción de noticias junto con una revolucionaria solución de intercambio de contenidos y contribución sobre IP. Hoy día más de 600 canales de TV en 50 países confían en VSN. (4)

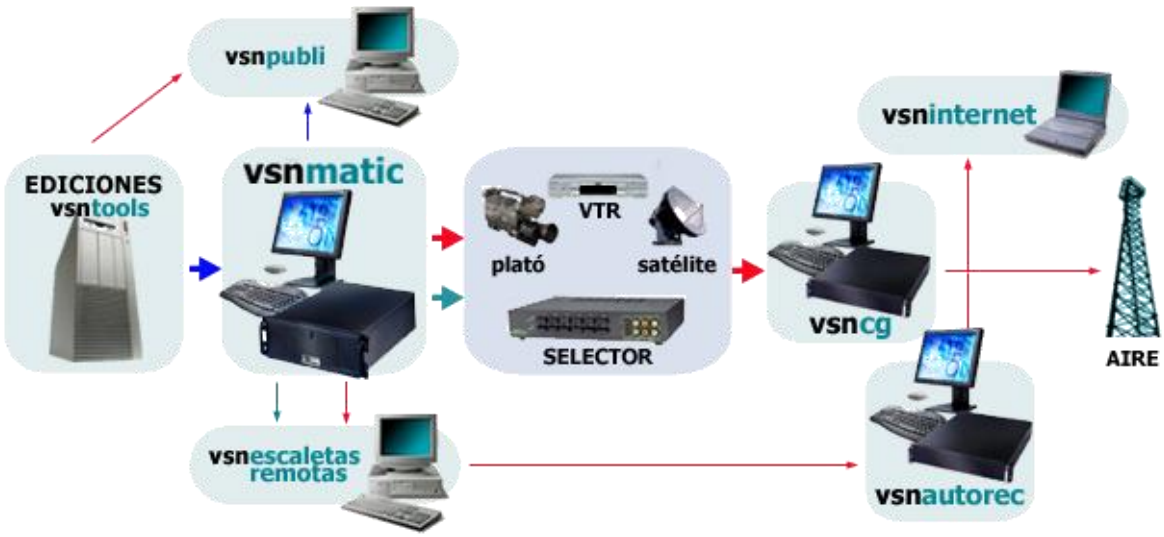

Figura 1 Modelo de despliegue de VSN.

Es una solución compuesta de varios módulos, diseñados para seguir el flujo lógico de trabajo de una emisora de TV. Se puede integrar con otros módulos VSN para añadir prestaciones inteligentes de una manera modular y escalable.

[1] Paquete de datos enviado a todos los nodos de una red.

Todos los procesos automatizados están comunicados por red y cada departamento el responsable de gestionar las tareas correspondientes.

Estas características se podrían adaptar para el producto PRIMICIA para un mejor desempeño en red. Además de la importancia de ser un sistema modular, donde se divida el sistema por módulos responsables de realizar un conjunto de actividades o funcionalidades según se haya definido, permitiendo una mayor flexibilidad ante los cambios, mantenimiento y una mayor adaptación, según donde se instale, esto conlleva a que la vida del software o del sistema sea más perdurable.

Como se puede apreciar esta es una aplicación de escritorio, esto tiene como ventaja que se puede trabajar directamente con el hardware y trabajar con videos y otros recursos que requieran de un mayor rendimiento de la computadora. El trabajo con videos e imágenes sugiere la ventaja de la reutilización de algoritmos especializados disponibles.

Figura 2 Aplicación escritorio VSN Matic.

Por todas las ventajas antes expuestas, se puede decir que sería de gran utilidad en el futuro desarrollar la Plataforma de Televisión Informativa como una aplicación de escritorio, como VSN Matic. Esto enriquecería al sistema informativo por ser modular y poder agregarles módulos que cumplan con diferentes funcionalidades según desee el cliente o usuario, permitiendo que sea una aplicación más abierta, o sea, lograr que sea adaptable a cualquier institución donde se vaya instalar.

1.5.1.2 Estructure

Estructure es básicamente una avanzada plataforma sobre la que se construyen y configuran sistemas informáticos de gestión y explotación de contenidos audiovisuales, integrando todas las funcionalidades y

tareas necesarias de forma flexible y modular, permitiendo una correcta y ágil organización, explotación y flujo del trabajo. (5)

Mediante una potente estructura de trabajo en grupo, permite traspasar flujos de datos e información de manera ordenada y simultánea a todos los usuarios del sistema, permitiéndoles compartir los archivos, su documentación y trabajar un proyecto o contenido en común.

Figura 3 Esquema básico de funcionamiento de Estructure.

Con Estructure todos los procesos productivos están integrados y se conectan entre sí aprovechando al máximo el trabajo en grupo: desde las áreas técnicas mediante la digitalización y control de emisión, hasta el visionado, redacción, edición, documentación, Internet o control del archivo, ofreciendo un nivel superior de funcionalidad.

Facilita las tareas mediante un extremadamente sencillo y agradable interfaz de trabajo. Todas las herramientas están disponibles en una única pantalla por área funcional, manteniendo una estructura lógica de trabajo mediante módulos productivos relacionados y jerarquizados con un doble nivel de organización: por contenidos y por proyectos a modo de carpetas contenedores.

Los sistemas se basan en tecnología abierta, 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, configurando grupos de usuarios y redes, dimensionando el sistema de forma modular y escalable en base al enfoque productivo de cada empresa usuaria.

Esta aplicación tiene grandes ventajas para el producto PRIMICIA pues es sencillo e intuitivo, tiene una agradable interfaz, todas las herramientas están visibles en cada módulo productivo con ausencia de menús y relacionados entre sí. Además es de fácil aprendizaje con comandos intuitivos.

También se puede decir que tiene gran ventaja al ser modular y escalable ya que tiene gran flexibilidad de crecimiento del sistema mediante fácil incorporación gradual de módulos y usuarios. Tienen gestión de grupos de trabajos y de usuarios, una estructura abierta a incorporación de nuevas tecnologías y permite construir el tamaño de red que se necesite.

Otra característica válida es que es rápido y potente debido a que todos los procesos se realizan en tiempo real. Tiene gran optimización en las redes con altas tasas de transferencias, posee acceso simultáneo de todos los usuarios al sistema.

La única desventaja que tiene es que es una herramienta privativa e interactúa nada más con este tipo de herramientas como son los sistemas operativos de Microsoft.

1.5.1.3 Escripnauta

El objetivo de la herramienta es que combine imágenes, texto y audio para crear presentaciones multimedia, usando un navegador web. De esta forma se ofrece a los usuarios un entorno de creación común y universal. Esto evita las desigualdades o ventajas derivadas de que algunos centros cuenten con más recursos de software que otros.

Además el sistema permite modificar cualquier elemento de las animaciones, en cualquier momento, en cualquier lugar.

Otro de los objetivos es que los usuarios trabajen en grupos y desde cualquier lugar con conexión a Internet. De esta manera soporta el trabajo colaborativo, es decir, que varias personas puedan trabajar

simultáneamente sin que se produzcan inconsistencias en la información. La aplicación es multiplataforma y compatible con los estándares, lo que permite que funcione en la mayoría de los navegadores (con un mayor rendimiento en Mozilla, Internet Explorer y Safari).

El sistema utiliza la gestión de concursos, permitiendo usuarios con varios niveles de privilegios (administradores, responsables de autores) y centros.

1.5.2 Alternativas Nacionales

1.5.2.1 Sistema Automatizado de Teletexto, Señal ACN

En el ámbito nacional, se encuentra como sistema más relevante Señal ACN, desarrollado en la Universidad de Ciencias Informáticas. Señal ACN es un sistema automatizado de teletexto el cual transmite su señal a través de la Plataforma de Televisión Satelital Cubana, su objetivo es informar a los colaboradores cubanos en el exterior y a los habitantes de las zonas de silencio. Este software posee las características necesarias para PRIMICIA, puesto que le da la posibilidad al usuario de observar a través de una interfaz amigable, en el televisor, las diferentes noticias gestionadas por el equipo de realización del canal.

El sistema está compuesto por dos subsistemas:

- Administración: en el cual el equipo de realización, realiza la gestión de las informaciones y recursos del canal.
- Transmisión: el cual se encarga de la visualización de las noticias y demás recursos audiovisuales de forma automática.

Los lenguajes utilizados para el desarrollo de Señal ACN fueron HTML, Java Script, ASP, Lingo y C Sharp. Los entornos de desarrollo empleados fueron: Macromedia Dreamweaver y Macromedia Director, herramientas patentadas, siendo necesario licencias; además existen otros obstáculos que según la situación que tiene el país sería engorroso y no beneficioso desarrollar el producto PRIMICIA con herramientas privativas.

En el siguiente modelo de componentes (Figura 1) se puede apreciar la ventaja en cuanto a base de datos que presenta el sistema, debido a la realización de réplicas entre las bases de datos que lo soportan. Esto ocurre debido a que los servidores están distantes, unidos por un enlace lento e inestable, por lo que si se interrumpe el enlace de comunicación un servidor no puede acceder a la base de datos y esto traería como resultado la falla total del sistema. Esto hace necesario la existencia de un sistema de réplicas de una base de datos a otras continuamente, para que la misma información este en ambos servidores y así evitar la falla del canal.

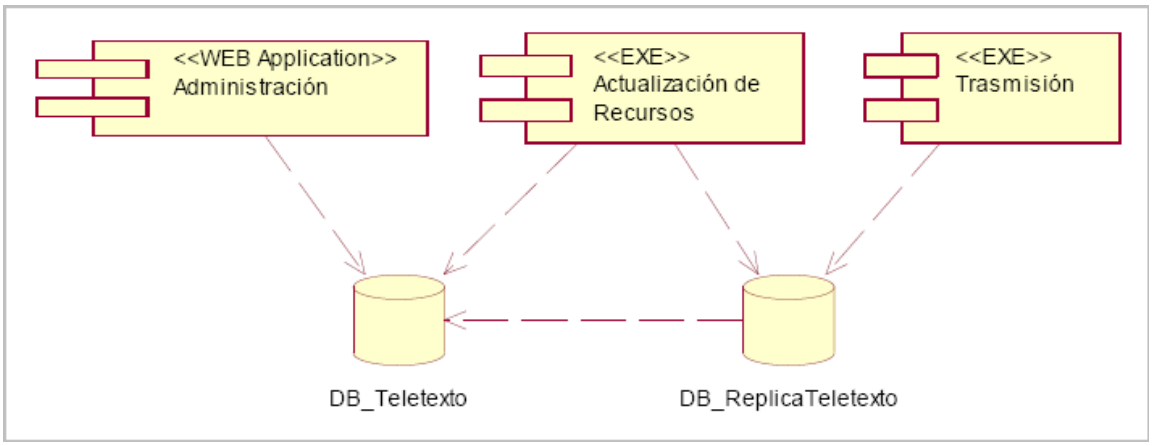

Figura 4 Modelo de componentes de Señal ACN.

El siguiente diagrama ilustra las dependencias entre los principales recursos de hardware del sistema. Existe un recurso para la administración web con el objetivo de obtener una mayor abstracción del sistema.

Señal ACN por su parte tiene una arquitectura física adaptable a cualquier situación, puesto que el servidor de administración está separado físicamente del servidor de transmisión a una distancia considerable, teniendo que buscar así una manera óptima de comunicación entre ellos para el envío de los datos a mostrar por el canal de teletexto. Esto sería muy factible para PRIMICIA con vistas a lograr la generalidad de la instalación del producto.

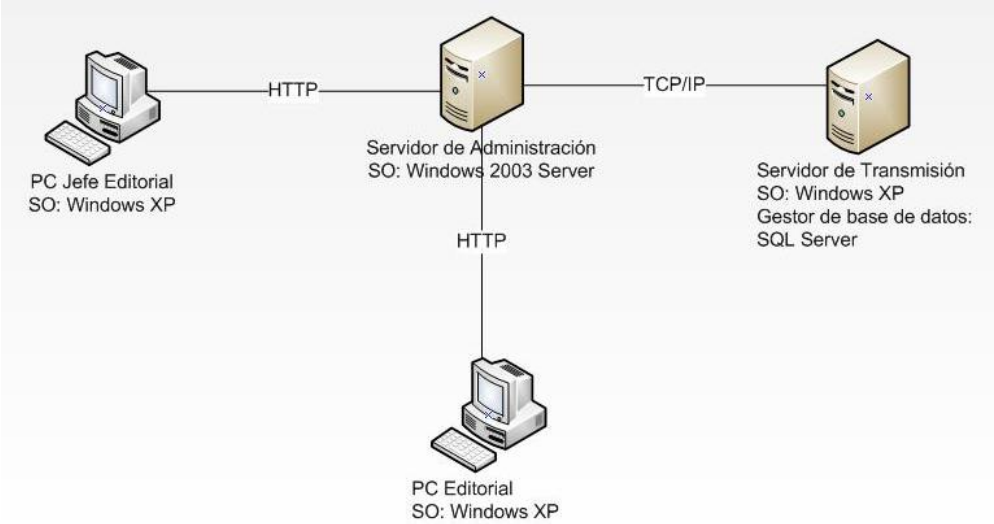

Figura 5 Modelo de despliegue de Señal ACN.

1.6 Conclusiones

En este capítulo se ha abordado sobre las diferentes arquitecturas de software que existen en el ámbito nacional e internacional para sistemas informativos. Pero solo se han analizado profundamente aquellas que realmente puedan contribuir a la conceptualización del producto PRIMICIA, para poder definir una arquitectura que de solución a la creación de este tipo de software. También se abordaron los diferentes conceptos en los cuales se apoya la arquitectura, donde se llega a la conclusión de que son muchas las tecnologías que hoy nacen y están implementando sistemas informativos, algunas aplicando ya lo que es televisión digital y sus conceptos, otras a través de Internet hacen llegar la información a los diferentes usuarios. Además se puede concluir que hay una gran tendencia según los sistemas analizados a las arquitecturas cliente – servidor centralizadas, utilizando recursos como la web para poner en funcionamiento el sistema informativo.

CAPITULO 2

TENDENCIAS Y TECNOLOGÍAS ACTUALES

2.1 Introducción

En el presente capítulo se presentan las principales tendencias de herramientas para el desarrollo de aplicaciones web con el fin de analizar cuáles son las más indicadas para el desarrollo de PRIMICIA. Además se tiene en cuenta y se describen los principales estilos y patrones arquitectónicos, con el objetivo de reutilizar una solución que permita optimizar y ganar en tiempo en el desarrollo del producto.

2.2 Estilos y Patrones Arquitectónicos

El tópico más urgente y exitoso en arquitectura de software en los últimos cuatro o cinco años es, sin duda, el de los patrones tanto en lo que concierne a los patrones de diseño como a los de arquitectura. Inmediatamente después, en una relación a veces de complementariedad, otras de oposición, se encuentra la sistematización de los llamados estilos arquitectónicos.

Robert T. Monroe plantea que un estilo arquitectónico provee una colección de elementos edificadores del diseño en bloque, reglas y restricciones para componer los bloques constructivos, y las herramientas para analizar y manipular los diseños creados en el estilo. Los estilos generalmente proveen guía y análisis para crear una clase amplia de arquitecturas en un dominio específico donde los patrones se enfocan en solucionar los problemas más pequeños, más específicos dentro de un estilo dado. (6)

Los estilos arquitectónicos son una generalización y abstracción de los patrones de diseño. Caracteriza una familia de sistemas que están relacionados por compartir propiedades estructurales y funcionales. También puede definirse como la descripción de los tipos componente y de los patrones de interacción entre ellos. A continuación se expondrán algunos ejemplos de estilos arquitectónicos y sus principales características, luego se describirán los patrones considerados a utilizar en el desarrollo del producto PRIMICIA.

Estilos de flujo de datos

Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos. Ejemplares de la misma serían las arquitecturas de tubería-filtros y las de proceso secuencial en lote.

Tubería y filtros

Este estilo está compuesto por componentes que son los filtros, cada filtro tiene un conjunto de entradas y uno de salidas. Cada filtro lee flujos de datos en sus entradas y produce flujos de datos en sus salidas. (7)

Los conectores que son los tubos transmiten la salida de un filtro como entrada de otro y no se efectúa ningún otro procesamiento visible.

Algunas invariantes sobre los filtros es que deben ser independientes, en particular no deben compartir el estado con otros, no conocen la identidad de sus continuadores o predecesores y la corrección de la salida del sistema no debe depender en el orden en que actúan.

A continuación se puede apreciar, en la figura 6, un ejemplo del estilo Tuberías y Filtro para un compilador.

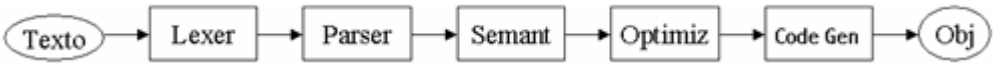

Figura 6 Compilador en tubería-filtro del estilo.

Estilos centrados en datos

Esta familia de estilos enfatiza la integrabilidad de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento. Sub-estilos característicos de la familia serían los repositorios, las bases de datos, las arquitecturas basadas en hipertextos y las arquitecturas de pizarra.

Arquitecturas de pizarra o repositorio

En esta arquitectura hay dos componentes principales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él. En base a esta distinción se han definidos dos subcategorías principales del estilo:

1. Si los tipos de transacciones en el flujo de entrada definen los procesos a ejecutar, el repositorio puede ser una base de datos tradicional (implícitamente no cliente-servidor).
2. Si el estado actual de la estructura de datos dispara los procesos a ejecutar, el repositorio es lo que se llama una pizarra pura o un tablero de control.

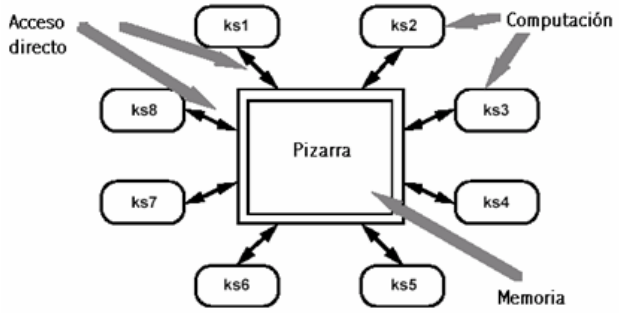

Figura 7 Ilustra la arquitectura Pizarra del estilo arquitectónico centrado en datos.

Estilos de llamada y retorno

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. Miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas.

Arquitecturas Orientadas a Objetos

Nombres alternativos para este estilo han sido Arquitecturas Basadas en Objetos, Abstracción de Datos y Organización Orientada a Objetos. Los componentes de este estilo son los objetos, o más bien instancias de los tipos de dato abstractos.

2.2.1 Modelo-Vista-Controlador (MVC)

Varios problemas pueden surgir cuando las aplicaciones contienen una mezcla de código de acceso a datos, lógica de negocio, y la presentación. Estas aplicaciones son difíciles de mantener, debido a las interdependencias entre todos los componentes. El alto acoplamiento de las clases hace difícil o imposible volver a utilizar el código, porque dependen de muchas otras clases. Adición de nuevos datos de puntos de vista a menudo requiere cortar y pegar código de lógica de negocio, que requiere de mantenimiento en varios lugares. Código de acceso de datos adolece del mismo problema, al ser cortada y pegada de los métodos de la lógica de negocio.

El modelo-vista-controlador como patrón de diseño resuelve estos problemas mediante la disociación de acceso a datos, lógica de negocio, y la presentación de los datos y la interacción del usuario. (8)

El patrón se divide en tres capas:

- Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista: Maneja la visualización de la información.
- Controlador: Controla el flujo entre la vista y el modelo (los datos).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

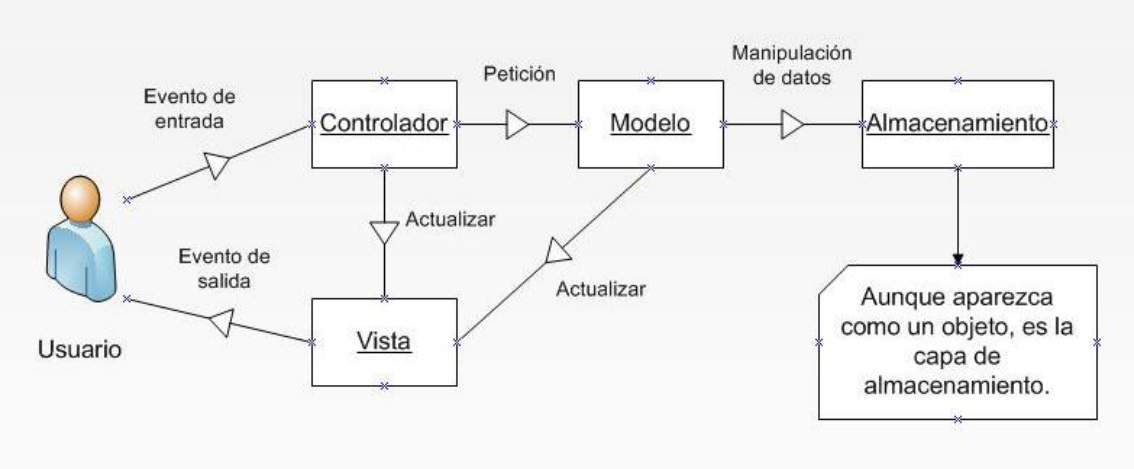

Figura 8 Estructura del patrón MVC.

Este patrón es el más indicado para el desarrollo del producto PRIMICIA por los beneficios que aporta. Entre las ventajas del estilo Modelo-Vista-Controlador para el desarrollo de la plataforma están las siguientes:

- Aislamiento entre las diferentes capas. Por ejemplo, si la vista es una aplicación Web basada en JavaScript (JS) y se quiere cambiar el modelo, para que acceda a otra base de datos, la vista no será afectada por el cambio.

- También permite utilizar los mismos objetos del modelo para diferentes vistas. Por ejemplo se puede hacer que la aplicación tenga dos tipos de presentación: una en HTML para visualizarla en un navegador y otra en XML para exportarla. El controlador podrá decidir qué vista presentar. Esta ventaja se ve muy útil en la parte del módulo de administración.

- Soporte de múltiples vistas: dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes.

- Adaptación al cambio: los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de

representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

2.2.2 Arquitectura en Capas

La idea fundamental es organizar los operadores y le información en capas. La capa inferior tiene operadores que actúan directamente sobre el estado de solución del problema. Las capas superiores tienen *meta*-operadores que actúan sobre los operadores de la capa inmediata inferior. El control se implementa capa por capa, siendo la responsabilidad de una capa controlar la ejecución de los operadores (o *meta*-operadores) de la capa inmediata inferior. La comunicación entre capas se realiza por medio de mensajes. (9)

Este patrón define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores.

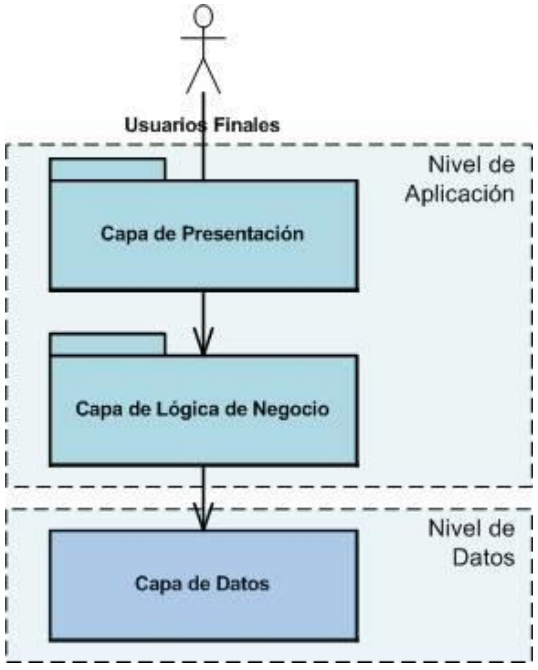

Figura 9 Diagrama de arquitectura en capas.

Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, ayuda a identificar qué puede reutilizarse, y proporciona una estructura que ayuda a tomar decisiones sobre qué partes comprar y qué partes construir.

2.3 Patrones

El término patrón se utiliza inicialmente en el campo de la arquitectura, por Christopher Alexander, a finales de los 70. Este conocimiento es trasportado al ámbito del desarrollo de software orientado por objetos y se aplica al diseño.

Un patrón es una unidad de información nombrada, instructiva e intuitiva que captura la esencia de una familia exitosa de soluciones probadas a un problema recurrente dentro de un cierto contexto. (6)

El objetivo de los patrones es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones.

Pueden referirse a distintos niveles de abstracción, desde un proceso de desarrollo hasta la utilización eficiente de un lenguaje de programación.

Un buen patrón debería:

- Solucionar un problema
- Ser un concepto probado
- La solución no es obvia
- Describe participantes y relaciones entre ellos
- Tiene un componente humano alto: estética y utilidad

2.4 Frameworks

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Se puede encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier ámbito. En general, el término framework, es una estructura software compuesta de componentes personalizables e intercambiables

para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se puede añadirle las últimas piezas para construir una aplicación concreta. (10)

Un framework para el desarrollo de la Plataforma de Televisión Informativa proporciona ventajas como: permitir el desarrollo rápido de la aplicación web, pues los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel. Además se puede reutilizar a gran en escala los componentes de software, debido a que los frameworks son los paradigmas de la reutilización. También se puede hacer uso de componentes que siguen una política de diseño uniforme. Puesto que un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

Las desventajas que puede tener la utilización de un framework para PRIMICIA es la dependencia del código fuente de la aplicación con respecto al mismo, debido a que si se desea cambiarlo, el código debe reescribirse. Además hay una demanda de grandes cantidades de recursos computacionales debido a que la característica de reutilización de los frameworks tiende a generalizar la funcionalidad de los componentes.

2.4.1 Frameworks para el desarrollo de aplicaciones web

Existen varios frameworks que proporcionan muchas de las funcionalidades requeridas para el desarrollo de la aplicación, en este caso particular sólo se analizarán cuatro de ellos, debido a que están aprobados por la Universidad de las Ciencias Informáticas, estos son: CakePHP, Symfony, PHPPrado y Kumbia.

Symfony diseñado con el objetivo de optimizar la creación de las aplicaciones web, con el uso de sus características. Posee una librería de clases que permiten reducir el tiempo de desarrollo. Symfony está desarrollado en PHP5, se puede utilizar en plataformas *nix (Unix, Linux) y Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón MVC, soporta AJAX, plantillas y un gran número de bases de datos.

PHPPrado está basado en componentes y eventos con el objetivo de acelerar el desarrollo de aplicaciones web usando PHP 5. El concepto del desarrollo de aplicaciones en Prado es diferente, se

utilizan componentes, eventos y propiedades en vez de procedimientos, URL y parámetros. Este Framework combina especificaciones en un archivo XML, plantillas HTML y una clase PHP. Prado, cuenta con soporte para AJAX, validación, autentificación, plantillas, múltiples bases de datos.

CakePHP es un Framework similar a CodeIgniter de desarrollo rápido. Es una estructura de librerías y clases para programar aplicaciones web. Su base es el Framework de Ruby on Rails. Brinda la posibilidad de interactuar con las base de datos, usando ActiveRecord. Incorpora el patrón MVC, compatible con PHP4 y PHP5, Soporta AJAX, incluye caching (caché) y validación.

Kumbia es un framework para aplicaciones web libre escrito en PHP5. Basado en las prácticas de desarrollo web para software comercial y educativo. Kumbiaphp fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer. KumbiaPHP Framework intenta proporcionar facilidades para construir aplicaciones robustas para entornos comerciales. Esto significa que el framework es muy flexible y configurable.

Se han descrito los principales frameworks utilizados por la universidad que pueden ser utilizados para el desarrollo de aplicaciones web. Evidenciando las ventajas que tiene estos frameworks para el desarrollo rápido y eficiente de productos informáticos.

2.5 Metodologías de Desarrollo

2.5.1 Proceso Unificado de Desarrollo (RUP[2])

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)

Para la realización de un software RUP es una metodología importante debido a las características que presenta: centrada en la arquitectura, guiada por casos de usos e iterativo incremental. No constituye una herramienta para construir software, es una metodología que brinda al equipo de desarrollo una idea de cómo realizar el sistema.

[2] Por sus siglas en inglés: *Rational Unified Process.*.

Presenta cuatro fases por la cual transita el software, las cuales son:

- Inicio o Conceptualización.
- Elaboración.
- Construcción.
- Transición.

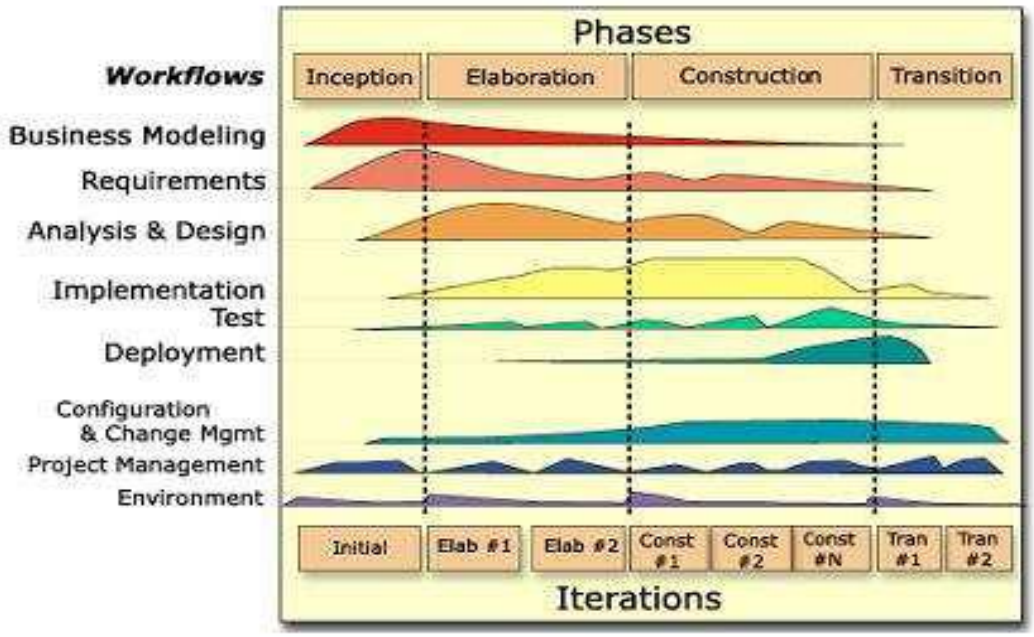

Figura 10 Ilustra las fases de RUP.

2.5.2 XP

XP es una metodología de desarrollo ágil. Como toda metodología ágil se caracteriza por ser iterativa. A continuación se verán sus cinco principios básicos.

- Realimentación rápida: El tiempo entre una acción y la retroalimentación de la misma es crítico.
- Asumir la Simplicidad: La solución simple es la mejor solución. Hacer una buena tarea: resolver hoy la tarea de hoy y confiar a la habilidad del desarrollador para añadir complejidad en el futuro.

- Cambio Incremental: Grandes cambios de una vez no funcionan. Los problemas se resuelven con una serie de pequeños cambios que hacen diferencia.
- Adherirse (Abrazar) al Cambio: La mejor estrategia es la que preserve la mayor parte de las opciones mientras resuelve el problema de mayor presión.
- Trabajo de Alta Calidad: Debe realizarse con la mayor calidad y rehacerlo, cuando exista algún motivo para ello. La calidad es una variable dependiente.

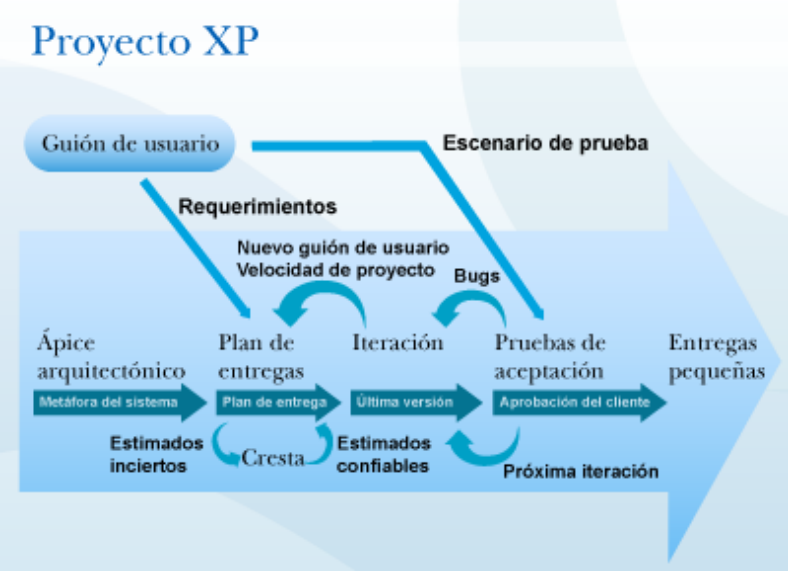

Figura 11 Ilustra un proyecto con metodología XP.

Esta metodología está diseñada para grupos pequeños donde hay que interactuar con el cliente en el proceso de desarrollo del software, esta característica impide el uso de XP en el desarrollo de PRIMICIA debido a que los clientes pudieran estar en el extranjero y no poder tener una interacción directa en el desarrollo del software.

2.6 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas. Representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas grandes y complejos. Se convirtió en estándar del Object Management

Group (OMG) en 1997, después de tres años de trabajo, como consecuencia de la llamada Guerra de las Metodologías. (11)

La principal ventaja de UML es ser un lenguaje de propósito general, aunque esto en ocasiones se puede convertir en una desventaja, porque no se pueden representar cabalmente las situaciones o características propias de dominios específicos. 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.

2.7 Herramientas CASE[3]

Una herramienta CASE es una tecnología para automatizar el desarrollo y mantenimiento del software, combinando herramientas de software y metodologías. Estas herramientas deben constituir un conjunto integrado que automatice todas las partes del ciclo de vida de desarrollo de un software y por tanto ahorren trabajo.

2.7.1 Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Su nombre completo es Visual Paradigm, la última versión estable es la 3.0 y fue creado en el 2006 por la compañía *Visual Paradigm International Limited*, el tipo de licencia que usa es: Floating License. UML Permite representar los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona tutoriales, demostraciones interactivas y proyectos UML.

Las características más importantes que tiene esta herramienta es que posee soporte para UML versión 2.1, tiene una mejora en la calidad de la documentación de base de datos con sofisticadas ERD[4] y Objeto-Relacional, tiene un intercambio de diagramas UML y modelos con otras herramientas. También tiene soporte de desarrollo de software, para un ciclo de vida completo.

[3] Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador)
[4] Entity Relationship Diagram (Modelo entidad - Relacional)

2.7.2 Rational Rose Enterprise

Rational Rose Enterprise es una buena elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java™/J2EE™, Visual C++® y Visual Basic®. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado para el equipo, facilita la creación de software de calidad rápidamente.

Entre sus principales características están las integraciones con los IDE como Borland JBuilder y Microsoft Visual Studio (versiones 2003 en adelante). Además presenta características adicionales como modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos, capacidad de crear definiciones de tipo de documento XML para el uso en la aplicación, integración con otras herramientas de desarrollo de Rational, y provee visualización, modelado y las herramientas para desarrollar aplicaciones Web.

Rational Rose en resumen facilita la representación y modelación del software, a partir de los artefactos propuestos por la metodología.

2.8 Sistemas Gestores de Bases de Datos

Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. (12)

Ayuda a realizar las siguientes acciones:

- Definición de los datos.
- Mantenimiento de la integridad de los datos dentro de la base de datos.
- Control de la seguridad y privacidad de los datos.
- Manipulación de los datos.

2.8.1 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Usar MySQL en el desarrollo de PRIMICIA tiene sus ventajas, pues es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla. Además una de las ventajas más importantes para el desarrollo de la aplicación es que permite conexiones entre diferentes máquinas con distintos sistemas operativos.

2.8.2 PostgreSQL

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

Tiene como ventajas para la Plataforma de Televisión Informativa que tiene soporte multiusuario, transacciones, optimización de consultas. Además presenta PL/pgSQL: Lenguaje de procedimientos almacenados, PostgreSQL es una base de datos diseñada para ser de tipo empresarial. También existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin , pgAccess).

Este gestor de base de datos brinda grandes potencialidades para el desarrollo de software, pues además de ser libre presenta una comunidad grande en la universidad y en el país que aboga por la realización de software que usen este gestor por las ventajas antes expuestas.

2.9 Lenguajes de Programación

La programación web permite la creación de sitios dinámicos en Internet. Esto se consigue generando los contenidos del sitio a través de una base de datos mediante lenguajes de script como pueden ser PHP, ASP o ASP.NET. En esta parte del documento solo se abordarán aquellos lenguajes utilizados en el desarrollo de aplicaciones web.

2.9.1 PHP

Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal.

PHP, es un lenguaje de programación interpretado, diseñado desde sus inicios para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (del lado del servidor de secuencias de comandos) pero en la actualidad puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre. Existen varias versiones de PHP, desde el antiguo PHP4 hasta ahora el muy utilizado PHP5, que tiene como ventajas para PRIMICIA que corre en casi cualquier plataforma utilizando el mismo código fuente y puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix y PostgreSQL.

2.9.2 JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Una de las ventajas de JavaScript es que todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web.

2.9.3 HTML

Para publicar información de distribución mundial, se necesita un lenguaje entendido universalmente, una especie de publicación de la lengua materna a todos los equipos que potencialmente pueden entender. El lenguaje utilizado por la Wold Wide Web es el HTML (Lenguaje de marcado de hipertexto).

HTML proporciona los medios para los autores:

- Publicar online con títulos, textos, tablas, listas, fotos, entre otros.
- Recuperar la información en línea a través de enlaces de hipertexto.
- Diseño de formularios para realizar transacciones con servicios remotos, para su uso en la búsqueda de información, hacer reservas, pedir productos, etc.
- Incluir hojas de cálculo, videoclips, clips de sonido y otras aplicaciones directamente en sus documentos. (13)

Este lenguaje tiene como ventaja para PRIMICIA que da la posibilidad de publicitar los productos o servicios las 24 horas del día, con fotografías ó videos, además facilita el contacto directo y asesoramiento técnico en línea, con el proveedor del equipo o servicio.

2.9.4 AJAX

Ajax es una nueva tecnología que es nombrada por primera vez por Jesse James Garret, es el acrónimo para Asynchronous JavaScript + XML, que en realidad no es una tecnología sino la combinación de muchas tecnologías como son: HTML, XHTML y CSS para la presentación de la información, el DOM (Document Object Model) y JavaScript que permiten el intercambio directo con la información y XMLHttpRequest que es el protocolo sobre el que está apoyado, y muy importante, es verdaderamente el corazón de Ajax. Utiliza JavaScript para manejar el objeto XMLHttpRequest, el HTML que distribuye en la ventana del navegador los elementos de la aplicación y la información recibida por el servidor, CSS que define el aspecto de cada elemento y dato de la aplicación y XML que no es más que el formato de los datos transmitidos del servidor al cliente (navegador) y que posteriormente serán mostrados.

Figura 12 El modelo tradicional para las aplicaciones Web comparado con el modelo de AJAX.

2.10 Ambiente Integrado de Desarrollo (IDE)

Un Entorno de Desarrollo Integrado o, en inglés, *Integrated Development Environment* (*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.

Un IDE 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 GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

2.10.1 Eclipse

Eclipse es una comunidad de fuente abierta, cuyos proyectos se centran en la construcción de una plataforma de desarrollo abierta formada por marcos extensibles y herramientas para crear, desplegar y gestionar software en todo el ciclo de vida de un proyecto. La Fundación Eclipse es una organización sin

fines de lucro, la corporación que alberga los proyectos de Eclipse ayuda a cultivar tanto una comunidad de fuente abierta como un ecosistema de productos y servicios. (14)

El proyecto eclipse fue creado originalmente por IBM en noviembre de 2001 y apoyado por un consorcio de proveedores de software. La Fundación Eclipse se crea en enero de 2004 como un independiente sin fines de lucro que actúe como administrador de la comunidad Eclipse. Hoy en día, la comunidad se compone de individuos y organizaciones de una sección transversal de la industria de software.

El entorno de trabajo presenta un ambiente de desarrollo integrado en perspectivas personalizables. Las perspectivas son combinaciones formadas por vistas y editores que muestran los diversos aspectos de los recursos del proyecto y están organizados por el rol o la tarea del desarrollador. Se proporcionan más de una perspectiva en un momento dado, además de pasar de una a otra mientras se está trabajando.

Eclipse incluye un *plugin* o complemento llamado PDT, el mismo presenta las siguientes características:

- Tipo jerárquico que navega orientado a objetos de PHP de código más rápido y más fácilmente.
- Tipo y método de navegación que permite una fácil búsqueda de código PHP basado en el tipo de información.
- Un nuevo motor de indexación y almacenamiento en caché, basado en el lenguaje dinámico de herramientas de Eclipse, que mejora significativamente el rendimiento global de las operaciones comunes PDT.
- Una nueva marca de sucesos con indicador que hace más fácil para los desarrolladores para ver cuando un elemento se hace referencia.

2.10.2 Zend Studio

Zend Studio se ha diseñado para una amplia gama de programadores y existen dos ediciones: Standard y Professional. Zend Studio, concebido con el fin de crear aplicaciones altamente fiables, proporciona una facilidad de uso inigualable, escalabilidad, fiabilidad, y la extensión que los programadores profesionales y de empresas requieren para desarrollar, distribuir, depurar y administrar aplicaciones PHP críticas de negocios.

Zend Studio proporciona la visualización, edición y la capacidad de ejecución para bases de datos populares SQL incluyendo MySQL, Oracle, IBM DB2 y Cloudscape, Microsoft SQL Server, SQLite y PostgreSQL.

2.11 Conclusiones

En este capítulo se abordó sobre las principales tecnologías que podrían servir o ser utilizadas para el desarrollo de la Plataforma de Televisión Informativa. Se ha hecho un análisis y caracterización de cada herramienta, metodología y tecnología que se pueda utilizar. Como se pudo ver en este capítulo en cuanto a las herramientas CASE de modelado se puede agregar que de las variantes expuestas con anterioridad resulta más práctico utilizar Visual Paradigm, puesto que tiene generación de código para lenguajes libres además de propietarios. También se puede concluir que de los gestores de bases de datos analizados uno que sería beneficioso utilizar PostgreSQL puesto que tiene licencia libre.

Además se puede argumentar que las metodologías expuestas no todas son beneficiosas para el desarrollo del producto PRIMICIA puesto que XP presenta una desventaja fundamental y es que necesita que el cliente este todo el tiempo junto con el equipo de desarrollo. Se concluye que existen muchos estilos arquitectónicos con ventajas para la Plataforma de Televisión Informativa, puesto que resuelven parte del problema de crear una arquitectura robusta y confiable basada en herramientas libres.

CAPITULO 3

SOLUCIÓN PROPUESTA

3.1 Introducción

A continuación se muestra la solución propuesta para el desarrollo de la Plataforma de Televisión Informativa. Se explica la línea base de la arquitectura para un mejor entendimiento del sistema, las herramientas y metodologías seleccionadas para realizar la aplicación y los subsistemas por los que está formado PRIMICIA.

3.2 Línea Base de la Arquitectura

En este tema se tiene como propósito lograr un mejor entendimiento del sistema proponiendo la arquitectura a tomar para el desarrollo del producto PRIMICIA. La definición de la arquitectura base es un paso esencial en el desarrollo de una línea de productos. Sin embargo, no existe ninguna forma estándar de hacer ésta definición.

3.3 Arquitectura seleccionada

La arquitectura a utilizar es: (cliente – servidor) por las ventajas que le facilita a PRIMICIA, por ejemplo: permite en el despliegue el uso de ordenadores especializados (servidores de base de datos, servidores de ficheros, estaciones de trabajo , etc.), contribuyendo a que el sistema sea multiplataforma, pues en cada servidor especializado puede encontrarse ejecutando alguna aplicación en una plataformas especializada, además permite centralizar el control de sistemas que estaban descentralizados, como por ejemplo la gestión de los ordenadores personales que antes estuvieran aislados.

3.3.1 Patrón arquitectónico seleccionado

Para la realización del producto se escogió el patrón MVC por las principales ventajas que brinda para la Plataforma de Televisión Informativa. Por ejemplo da la posibilidad de múltiples vistas ya que la vista no depende del modelo.

Tienen como única desventaja que de cambiar constantemente la parte del modelo las vistas caen en un diluvio de refrescamiento y la única manera de evitarlo es programando la aplicación para que esto no pase.

Otro criterio para seleccionar este patrón arquitectónico es debido a la selección del framework Symfony para desarrollar a PRIMICIA, pues el núcleo del framework restringe el uso del patrón MVC.

3.4 Framework seleccionado

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 de 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.

es el framework seleccionado, pues es diseñado para optimizar, gracias a sus características el desarrollo de las aplicaciones web. Es un marco de trabajo que restringe a usar el patrón arquitectónico MVC; proporciona varias herramientas y clases como acceso a datos, *helper*[5] para enriquecer la interfaz visual, además de una jerarquía de carpetas diseñadas para diseñar módulos, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Además es Symfony es escalable si se disponen de los recursos necesarios, ha sido probado con éxito durante varios años en aplicaciones muy diferentes, sigue una política de tipo LTS (*long term support*). Las versiones estables se mantienen durante 3 años sin cambios pero con una continua corrección de los errores conocidos, utiliza una licencia, con la que puedes hacer aplicaciones web comerciales, gratuitas y/o de software libre, desde su primera versión Symfony ha sido creado para PHP 5, se trata de un

[5] Conjunto de clases de apoyo a la interfaz visual.

framework PHP muy bien documentado, su código fuente incluye más de 8.000 pruebas unitarias y funcionales y se pueden crear aplicaciones en varios idiomas.

3.5 Subsistemas del sistema

En todas las aplicaciones el primer paso para diseñar un sistema consiste en dividir el sistema en un pequeño número de componentes. Cada uno de los componentes principales de un sistema se llama subsistema. Cada subsistema abarca aspectos del sistema que comparten alguna propiedad común.

Un subsistema no es ni una función ni un objeto, sino un paquete de clases, asociaciones, operaciones, sucesos y restricciones interrelacionados, y que tienen una interfaz razonablemente bien definida y pequeña con los demás subsistemas. Normalmente, un subsistema se identifica por los servicios que proporciona. Por estas razones es que se divide el sistema informativo en subsistemas.

El sistema informativo PRIMICIA cuenta con dos subsistemas, el Subsistema de administración es el encargado de insertar toda la información que será visualizada en el Subsistema de Transmisión, además gestiona toda la información presente en el sistema. También controla los roles de usuarios (autenticación según el nivel de privilegios), también se redactan las noticias y toda configuración de la interfaz que se trasmite en el canal. Y el Subsistema de Transmisión es el encargado de visualizar toda la información gestionada en el Subsistema de Administración en formato texto, imagen, video y sonido.

3.6 Restricciones de acuerdo a la estrategia de diseño

Las políticas que se tiene en cuenta a la hora de diseñar el sistema dependen de las herramientas de desarrollo.

El diseño de la aplicación, se hará aplicando el paradigma de la programación orientada a objetos. Se utilizarán las tecnologías que brinda el framework definido (Symfony), como núcleo se empleará el patrón arquitectónico MVC, además del conjunto de librerías y clases que faciliten el desarrollo del sistema.

El subsistema de administración será capaz de generar e imprimir reportes, los cuales se podrán exportar a formato PDF con la utilización de librerías (TCPDF) de PHP. En el subsistema de transmisión se hará uso del VLC (VideoLAN Client) para la transmisión vía streaming[6] de los videos a visualizar.

El sistema contará con dos servidores, uno para la gestión de recursos, datos y administración del canal; y el segundo encargado de la transmisión del canal, los mismos estarán ubicados en el nodo central o estar separados físicamente, por lo que si se encuentran a una corta distancia el uno del otro así podrán estar conectados punto a punto a una velocidad de 1 Gbps, pero si no estarían conectados por modem, router inalámbrico u otro dispositivo.

3.7 Lenguaje de Modelado UML seleccionado

El Lenguaje Modelado Unificado (UML) seleccionado es la versión 2.1, una norma ampliamente usada en la industria del software para el modelamiento de software. Ayuda a los practicantes a visualizar, comunica, y lleva a cabo sus planes. UML ha evolucionado durante los años y está ahora en su versión 2.x. La versión de UML para la herramienta de modelado Visual Paradigm se ha sincronizado con el nuevo desarrollo de UML 2.x para proporcionar un ambiente modelado visual que satisface la tecnología del software de hoy y necesidades de comunicación. (4)

3.8 Herramienta CASE seleccionada

Visual Paradigm es la herramienta CASE seleccionada, 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.

[6] Tecnología utilizada para permitir la visualización y la audición de un archivo mientras se está descargando.

Otro aspecto al tener en cuenta para la selección de la herramienta Visual Paradigm, es que posee una característica fundamental: soporte multiplataforma y está legalmente aprobada por la Dirección Técnica, su licencia es pagada por la Universidad.

3.9 Lenguaje de programación seleccionado

El lenguaje de programación seleccionado para el desarrollo de la aplicación es PHP5, soportado por el marco de trabajo o framework, Symfony. Es un lenguaje libre y es unos de los lenguajes de creación de páginas Web dinámicas más utilizados del mercado. La quinta versión cuenta con innumerables mejoras que consolidan su éxito. Se ofrece la posibilidad de hacer programas orientados a objetos, lectura de archivos XML de forma sencilla, utilización de la base de datos, como PostgreSQL, implementación de servicios Web, el envío de correo electrónico o la utilización del protocolo FTP, todo un conjunto de herramientas que hacen rica la implementación del sistema PRIMICIA.

3.10 Sistema Gestor de Base de Datos seleccionado

Como Sistema Gestor de Bases de Datos a utilizar para el desarrollo de la base de datos del sistema se seleccionó: PostgreSQL, por todas las características que brinda, ya antes mencionadas, siendo una de las fundamentales: es código abierto, además de permitir soporte en el lenguaje de programación seleccionado (PHP) y compatibilidad con el gestor de base de datos PostgreSQL.

Además se pueden encontrar algunas características de PostgreSQL que aún no están disponibles o estables en MySQL como son:

- Triggers (5.1 rudimentarios)
- Vistas (5.0)
- Secuencias
- Herencia
- Cursores (5.0)
- Stored Procedures (5.0 lenguaje único)

3.11 Entorno Integrado de Desarrollo seleccionado

Se seleccionó el IDE Eclipse que es una herramienta muy utilizada por la comunidad de desarrolladores web. Eclipse es una gran estructura formada por un núcleo y muchos *plugins* o complementos que van conformando la funcionalidad final. La forma en que los *plugins* interactúan es mediante interfaces o puntos de extensión; así, las nuevas aportaciones se integran sin dificultad ni conflictos.

Para la utilización de este IDE se le incluye el *plugin* PDT (*Developer PHP Tools*) versión 2.0, para permitir ejecutar el programa o el sistema (en la implementación) sobre la plataforma (Linux, Windows, etc.) utilizada. Entre las características que ofrece el *plugin* PDT, se encuentra que es fácil de usar e intuitivo, integrable con *Web Tools* de Eclipse, lo que permite a los desarrolladores del PRIMICIA disminuir la curva de aprendizaje en cuanto a esta herramienta.

3.12 Conclusiones

Luego de una búsqueda y análisis de información de las tecnologías a utilizar para el producto PRIMICIA, en este capítulo se han definido las herramientas que se ajustan para la solución del problema permitiendo al equipo de desarrollo tener las tecnologías adecuadas para iniciar otra fase de desarrollo del proyecto. Se puede concluir que las herramientas y lenguajes seleccionados por las ventajas que tienen ofrecerán una mayor rapidez en el desarrollo del producto. Se definió la línea base de la arquitectura que tiene la importancia de definir ya las funcionalidades que va a tener el sistema y por el cual se va a regir para el desarrollo del proyecto. A partir de la línea base de la arquitectura ya se puede pasar a la siguiente fase de RUP, la fase de construcción, para entrar a la implementación del sistema informativo.

CAPÍTULO 4

DESCRIPCIÓN DE LA ARQUITECTURA

4.1 Introducción

En este capítulo se aborda como tema fundamental algunos artefactos del rol arquitecto. Se lleva a detalle la arquitectura de software visualizando el Modelo de Análisis, de Diseño, de Implementación y de Despliegue, detallando así aun más el sistema y dándole una aproximación al código. Dejando de esta manera plasmado en esta parte del documento los artefactos que describirán aun más el producto PRIMICIA

4.2 Vista de Casos de Uso

La vista de casos de uso brinda información de escenarios y casos de usos de interés para cada iteración del ciclo de desarrollo. Se describen en esta los casos de uso de interés arquitectónico que encapsulan la funcionalidad del sistema. Los casos de uso arquitectónicamente significativos, que son aquellos que sirven para validar la arquitectura propuesta y describen las funcionalidades imprescindibles para el sistema.

El sistema consta con Casos de Uso de los cuales 9 son arquitectónicamente significativos los cuales se representan por módulo en la siguiente tabla.

Módulo	Cantidad de Casos de Uso
Seguridad	2
Redacción	1

Editorial	3
Gestión de Archivos	1
Reportes	0
Gestión de Señal	1
Transmisión de Señal	1

Tabla 1 Cantidad de Casos de Uso del sistema por módulos.

A continuación se describen brevemente los casos de uso arquitectónicamente significativos del sistema:

- **Autenticar Usuario:** El caso de uso inicia cuando una persona desea acceder al sistema, dando su usuario y contraseña, el caso de uso verifica la existencia de ese usuario, de no ser correcto, envía un mensaje informando que el usuario no es válido, en caso contrario lo lleva a la página de inicio según los privilegios de dicho usuario finalizando así el caso de uso.
- **Gestionar usuario:** El caso de uso se inicia cuando el actor necesita realizar operaciones con los usuarios del sistema: adicionar uno nuevo, modificar datos de uno existente, o eliminar alguno; quedando actualizados los usuarios del sistema finalizando el caso de uso.
- **Redactar noticia:** El caso de uso inicia cuando el actor accede a la función Redactar noticia, el sistema le mostrará un formulario para que entre el título, seleccione la sección y el fondo musical de la noticia y después entrará el cuerpo de la noticia dividido en pantallas que pueden ser de varios tipos (texto, imagen, texto-imagen, video). Concluyendo guarda la noticia finalizando así el caso de uso.
- **Publicar Noticia:** El caso de uso inicia cuando el actor decide publicar las noticias que han sido redactadas, escoge la sección y selecciona la noticia que desea publicar, revisa la noticia pudiendo efectuar algunas modificaciones como: eliminar pantallas, modificar los textos si el tipo de pantalla es de texto o texto imagen, cambiar recurso multimedia si la pantalla es de texto-imagen, imagen o video y cambiar el fondo musical, luego le asigna el período de tiempo que se estará mostrando en el canal, terminando de esta forma el caso de uso.

- **Gestionar sección:** El caso de uso inicia cuando el actor accede a la función de Gestionar secciones, donde establecerá el orden de las secciones, horario en que será mostrada y habilitarlas o deshabilitarlas. Al finalizar el caso de uso estará definido como será la visualización de las secciones.
- **Gestionar Infocintas:** El caso de uso comienza cuando el actor decide agregar nuevas infocintas, modificarlas, eliminarlas o establecer el orden en que se mostraran en sus secciones correspondientes. El caso de uso termina cuando el actor ha realizado alguna de las actividades mencionadas anteriormente.
- **Administrar archivo de recursos multimedia:** El caso de uso comienza cuando el actor accede a la función de Administrar archivo de recursos multimedia. Aquí se manejan todos los recursos multimedia del sistema, permitiendo, adicionar, eliminar y modificar la información. La música estará organizada por país, género y título y los videos e imágenes por sección temática, palabras claves, fecha y nombre. Al terminar de realizar las operaciones que se deseen en el archivo finaliza el caso de uso.
- **Gestionar señal del canal**: El caso de uso inicia cuando el actor accede a la funcionalidad de gestionar la señal del canal, dándole la posibilidad de cambiar la señal que se está transmitiendo en el momento o programar cambios de señal que se realizaran en un momento determinado.
- **Transmitir canal:** El caso de uso se inicializa cuando el actor Reloj detecta que existe alguna señal activada para transmitir. Se muestra inicialmente el video de presentación del canal. Si la señal activa es la de las televisoras nacionales o la de otro canal se muestra la presentación del enlace y seguidamente la señal. En el caso de que sea la propia del canal, se muestra la cartelera generada para el ciclo noticioso correspondiente. La cartelera mostrará la sección temática a la que pertenece la noticia y su título, mostrándose según el orden en que fueron publicadas. Al terminar de mostrar la cartelera se mostrarán las noticias según el orden que se estableció. Al inicio de cargar cada pantalla que conforman la noticia se verificará si existe algún cambio de señal en ese instante o en el intervalo que duraría mostrar la pantalla. Las noticias estarán acompañadas de un fondo musical que se reproduce durante su visualización, excepto en las pantallas de tipo video. Se mostrarán además las infocintas publicadas que estén relacionadas a la sección temática a la que pertenece la noticia, estas infocintas no se mostrarán en las pantallas de tipo imagen o video.

4.3 Vista de procesos

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 Base de Datos se maneja a través de los propios servidores.

El sistema no cuenta con tareas que requieran ejecutarse periódicamente sin la intervención del cliente, por esta razón no se requiere una Vista de Procesos.

4.4 Modelo de Análisis

Este modelo es usado para representar la estructura global del sistema, describe la realización de casos de uso, sirve como una abstracción del Modelo de Diseño y se centra en los requerimientos no funcionales.

Este modelo de análisis no es un diagrama final que describe todos los posibles conceptos y sus relaciones, es un primer intento por definir los conceptos claves que describen el sistema.

Para representar los diagramas del Modelo de Análisis se emplean diferentes diagramas de UML tales como:

- Diagramas de Clase.
- Diagramas de Secuencia.

En el siguiente subepígrafe se exponen los diagramas de secuencia de los casos de uso críticos.

4.4.1 Diagramas de Secuencias de los casos de usos críticos del sistema:

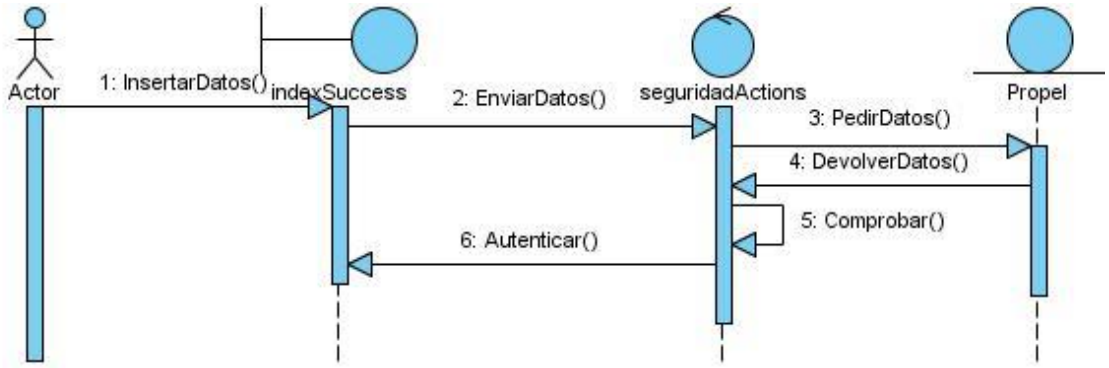

Figura 13 Diagrama de Secuencia del CUS (Autenticar Usuario).

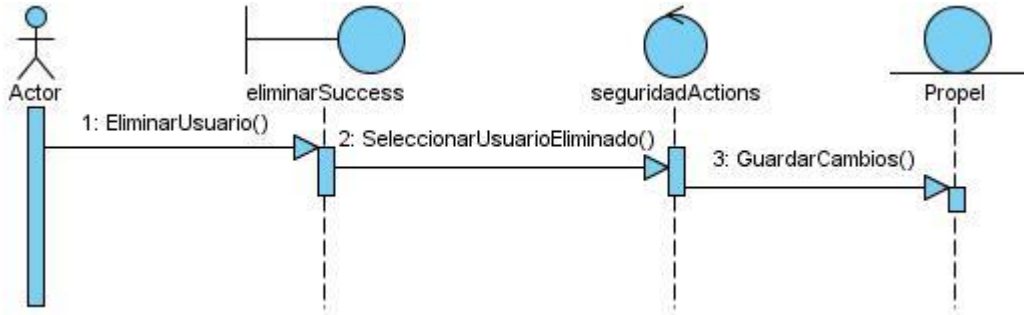

Figura 14 Diagrama de secuencia del CUS (Gestionar Usuario): Evento eliminar usuario.

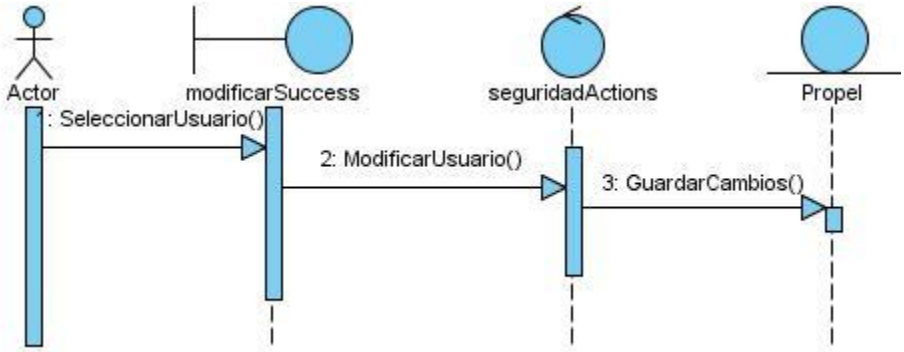

Figura 15 Diagrama de secuencia del CUS (Gestionar Usuario): Evento modificar usuario.

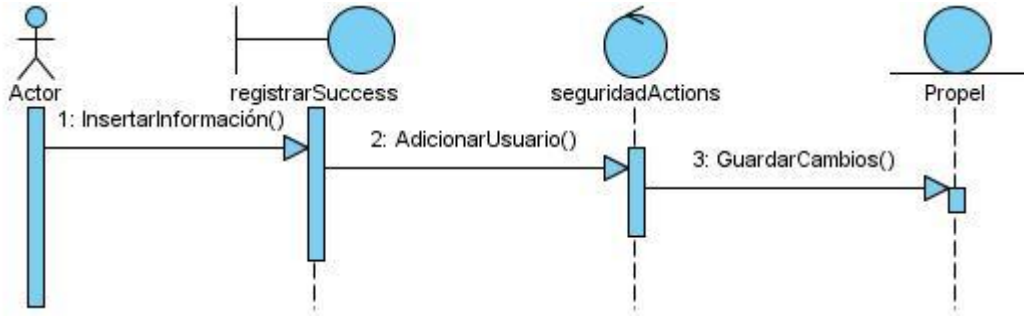

Figura 16 Diagrama de secuencia del Caso de Uso (Gestionar Usuario): Evento registrar usuario.

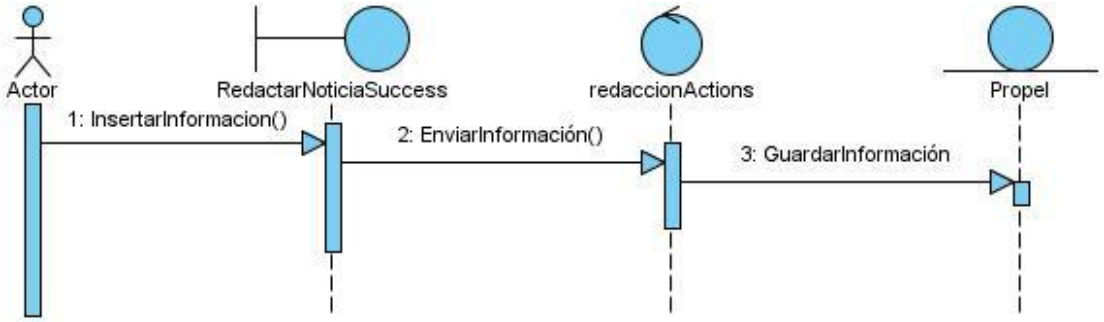

Figura 17 Diagrama de secuencia del CUS (Redactar Noticia).

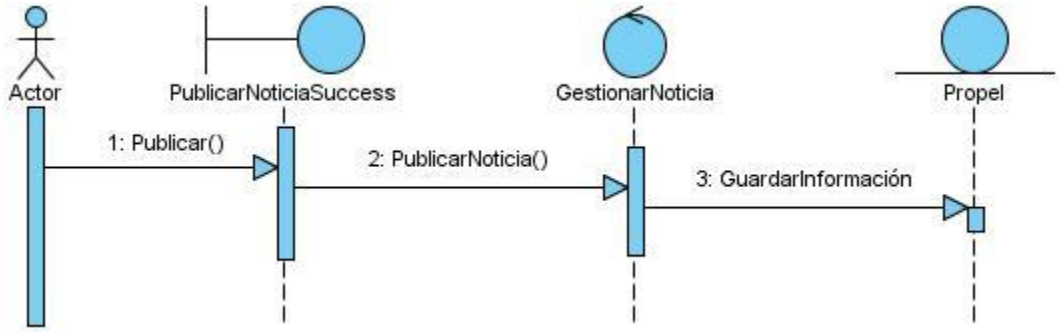

Figura 18 Diagrama de secuencia del CUS (Publicar Noticia).

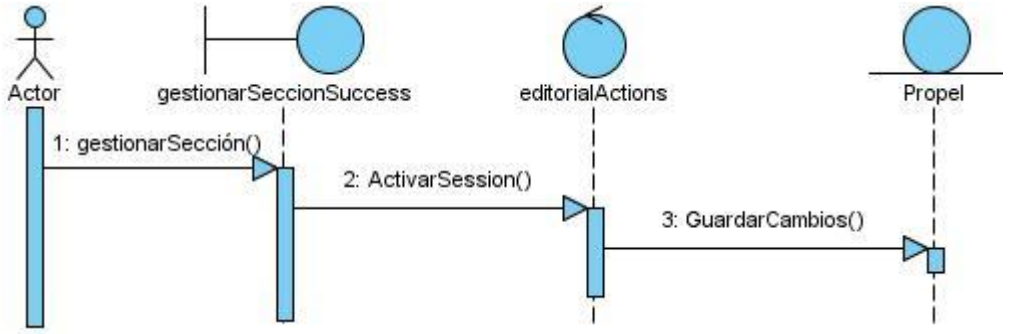

Figura 19 Diagrama de secuencia del CUS (Gestionar Sección).

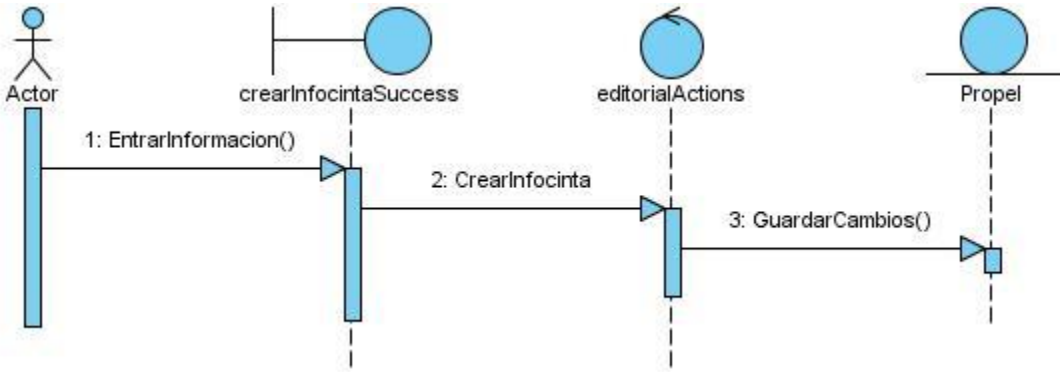

Figura 20 Diagrama de secuencia del CUS (Gestionar Infocintas) Evento: Crear Infocinta.

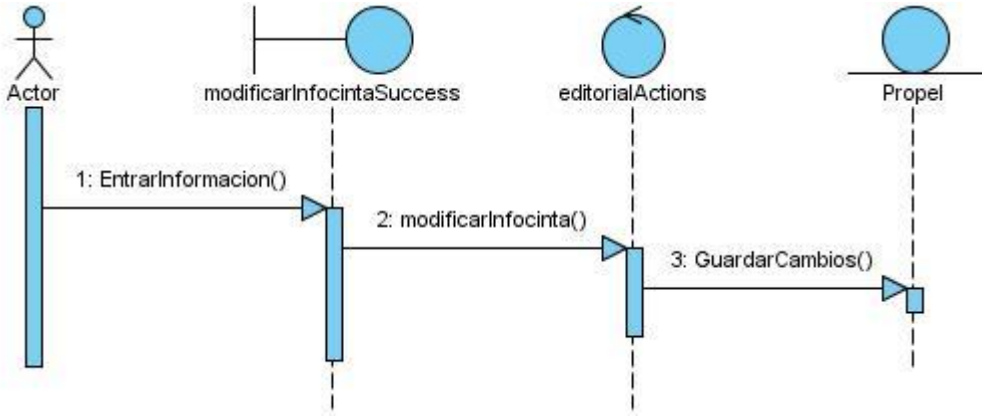

Figura 21 Diagrama de secuencia del CUS (Gestionar Infocintas) Evento: Modificar Infocinta.

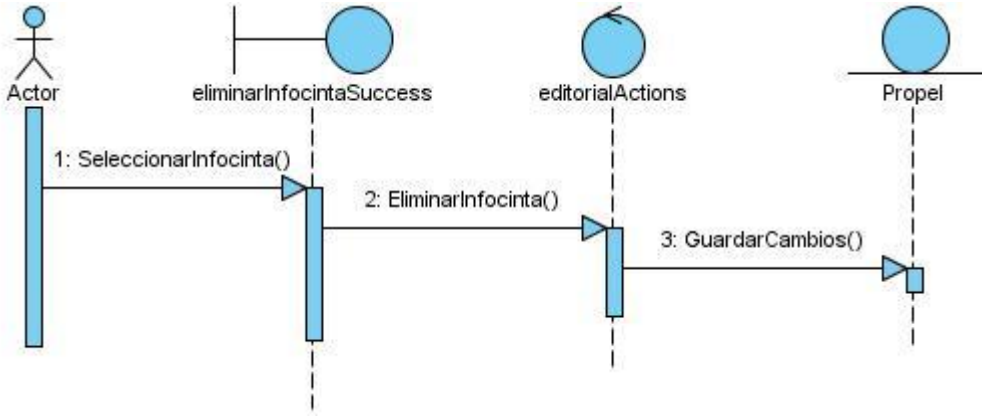

Figura 22 Diagrama de secuencia del CUS (Gestionar Infocintas) Evento: Eliminar Infocinta.

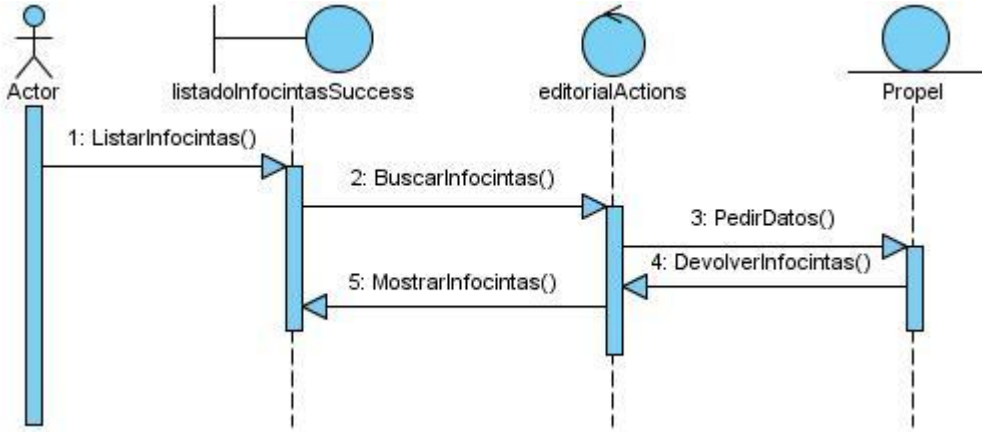

Figura 23 Diagrama de secuencia del CUS (Gestionar Infocintas) Evento: Listar Infocintas.

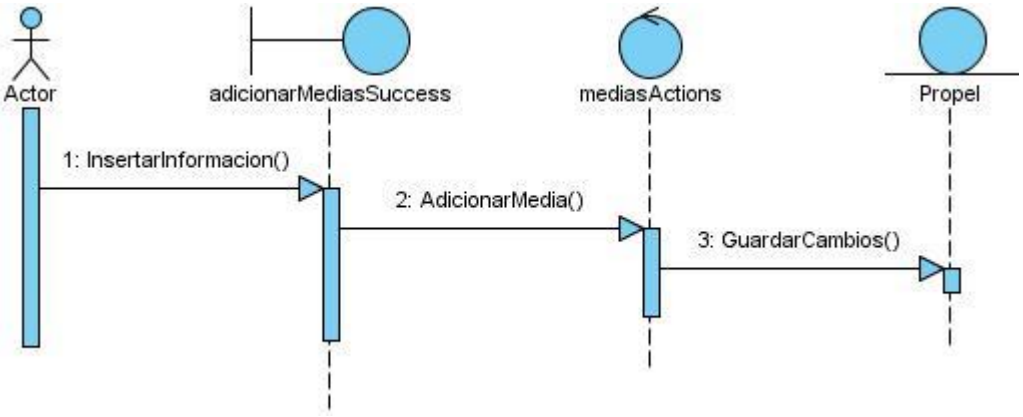

Figura 24 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Adicionar Media.

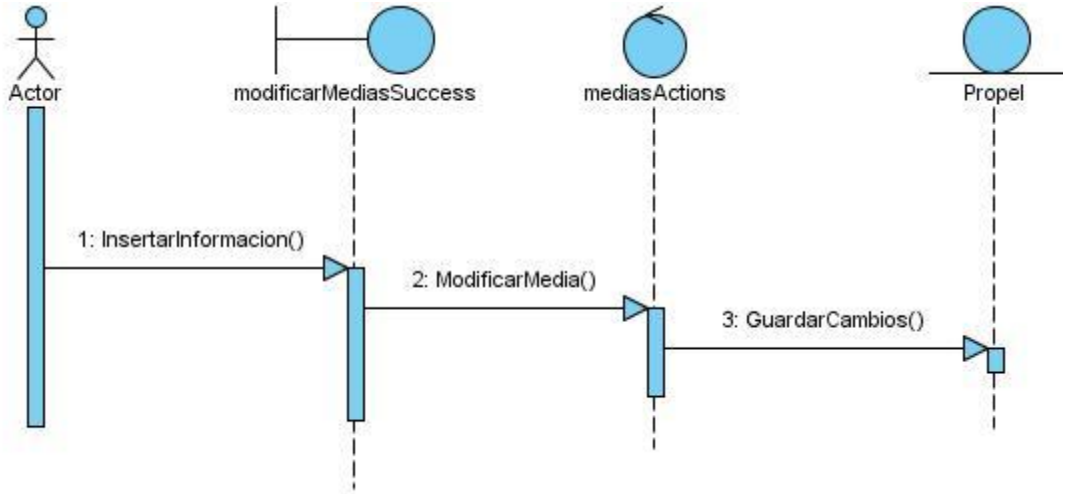

Figura 25 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Modificar Media.

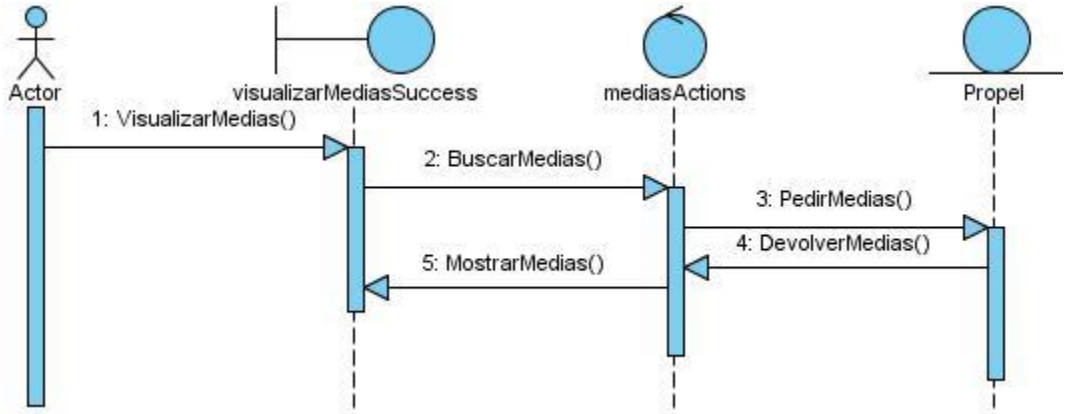

Figura 26 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Visualizar Media.

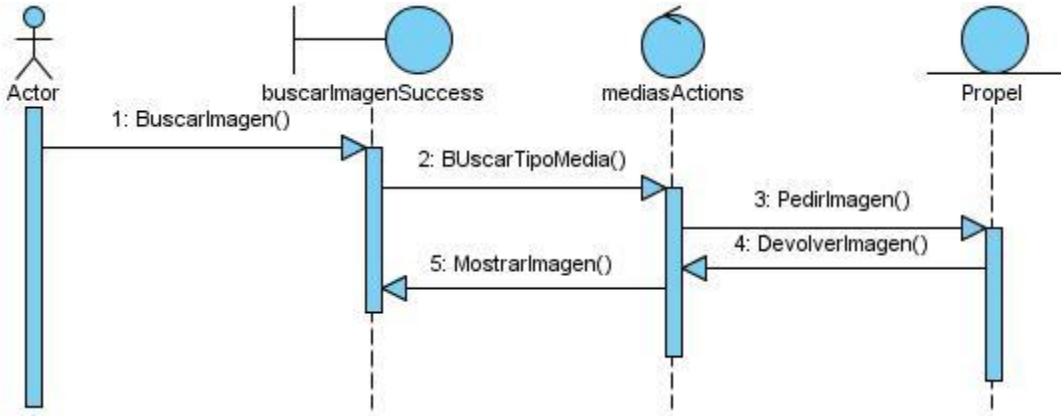

Figura 27 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Buscar Imagen.

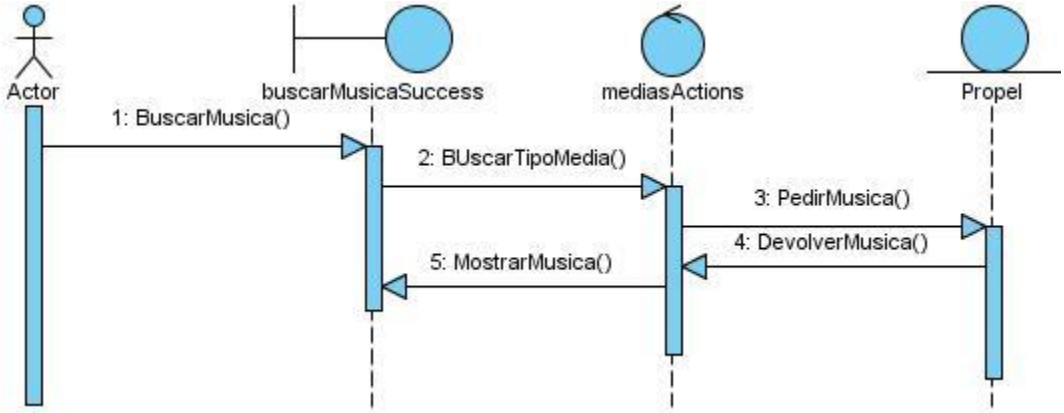

Figura 28 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Buscar Música.

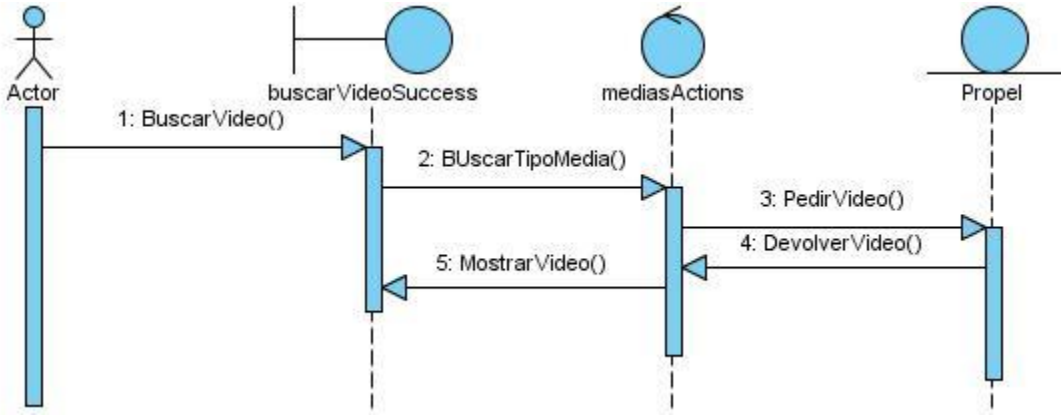

Figura 29 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Buscar Video.

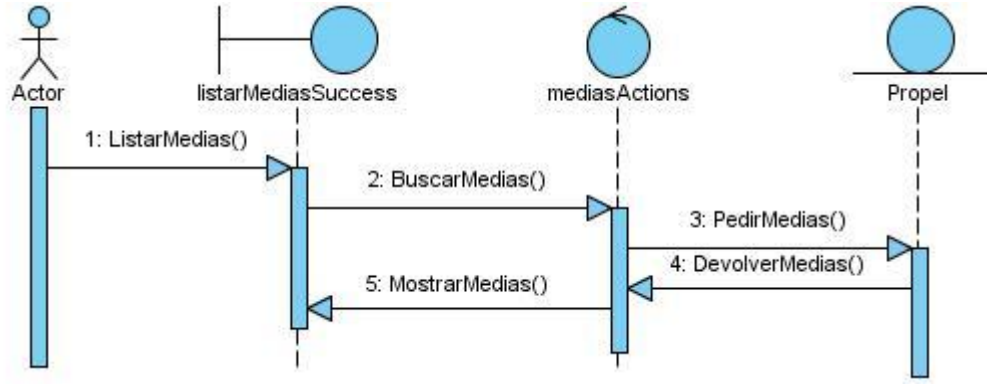

Figura 30 Diagrama de secuencia del CUS (Administrar archivo de recursos multimedia) Evento: Listar Medias.

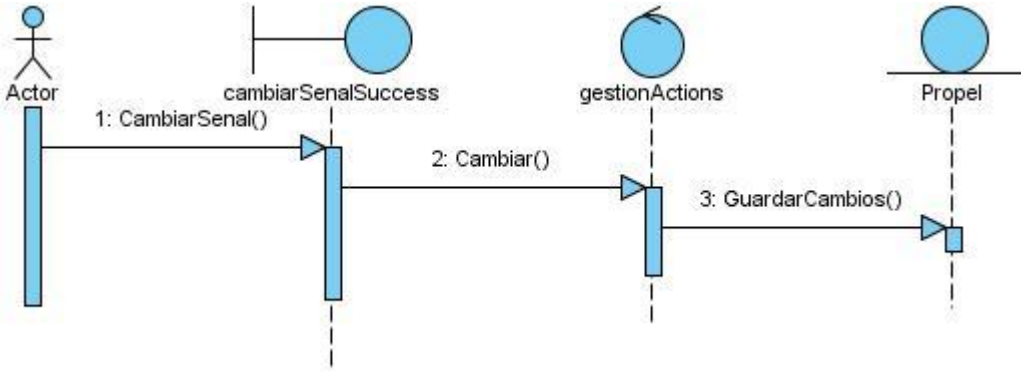

Figura 31 Diagrama de secuencia del CUS (Gestionar señal del canal) Evento: Cambiar Señal.

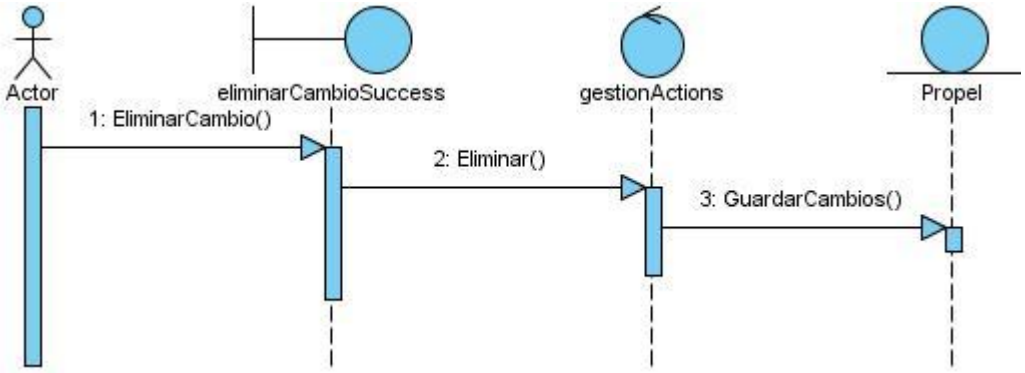

Figura 32 Diagrama de secuencia del CUS (Gestionar señal del canal) Evento: Eliminar Cambio.

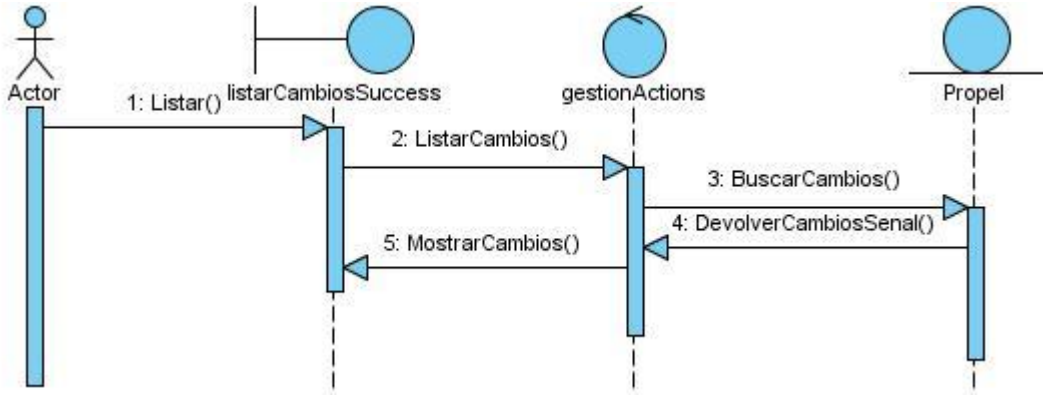

Figura 33 Diagrama de secuencia del CUS (Gestionar señal del canal) Evento: Listar Cambios de Señal.

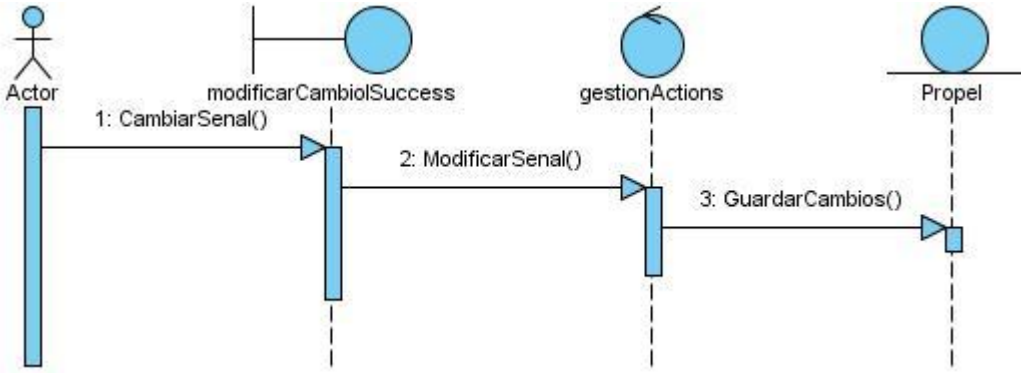

Figura 34 Diagrama de secuencia del CUS (Gestionar señal del canal) Evento: Modificar Cambios de Señal.

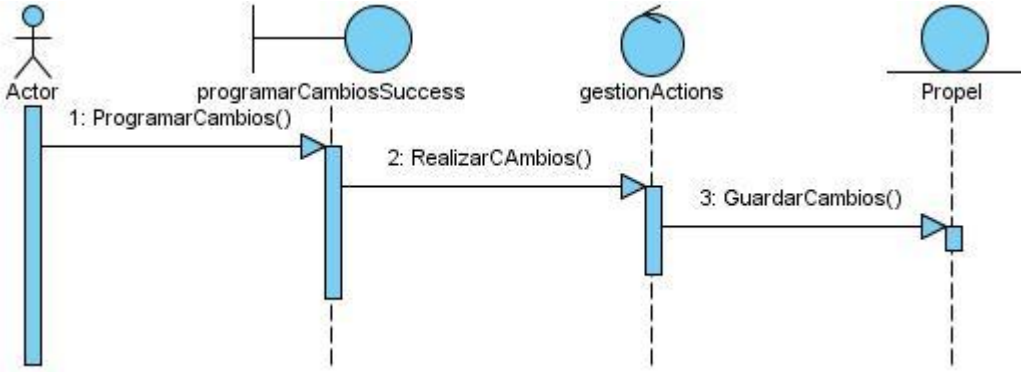

Figura 35 Diagrama de secuencia del CUS (Gestionar señal del canal) Evento: Programar Cambios de Señal.

4.5 Modelo de Diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución.

A continuación se presentan los diagramas de clase del diseño de los casos de usos: autenticar usuario, gestionar usuario, redactar noticia, publicar noticia, gestionar sección, gestionar infocintas, administrar archivo de recursos multimedia y gestionar señal.

Todos los diagramas de clases del diseño tienen un paquete nombrado “Componentes de Symfony”, aquí se encuentran los principales componentes que usa el framework Symfony como:

- sfRequestHandler
- sfRequest
- sfRouting
- sfLogger
- sfI18N
- sfUser
- sfTemplate
- sfResponse

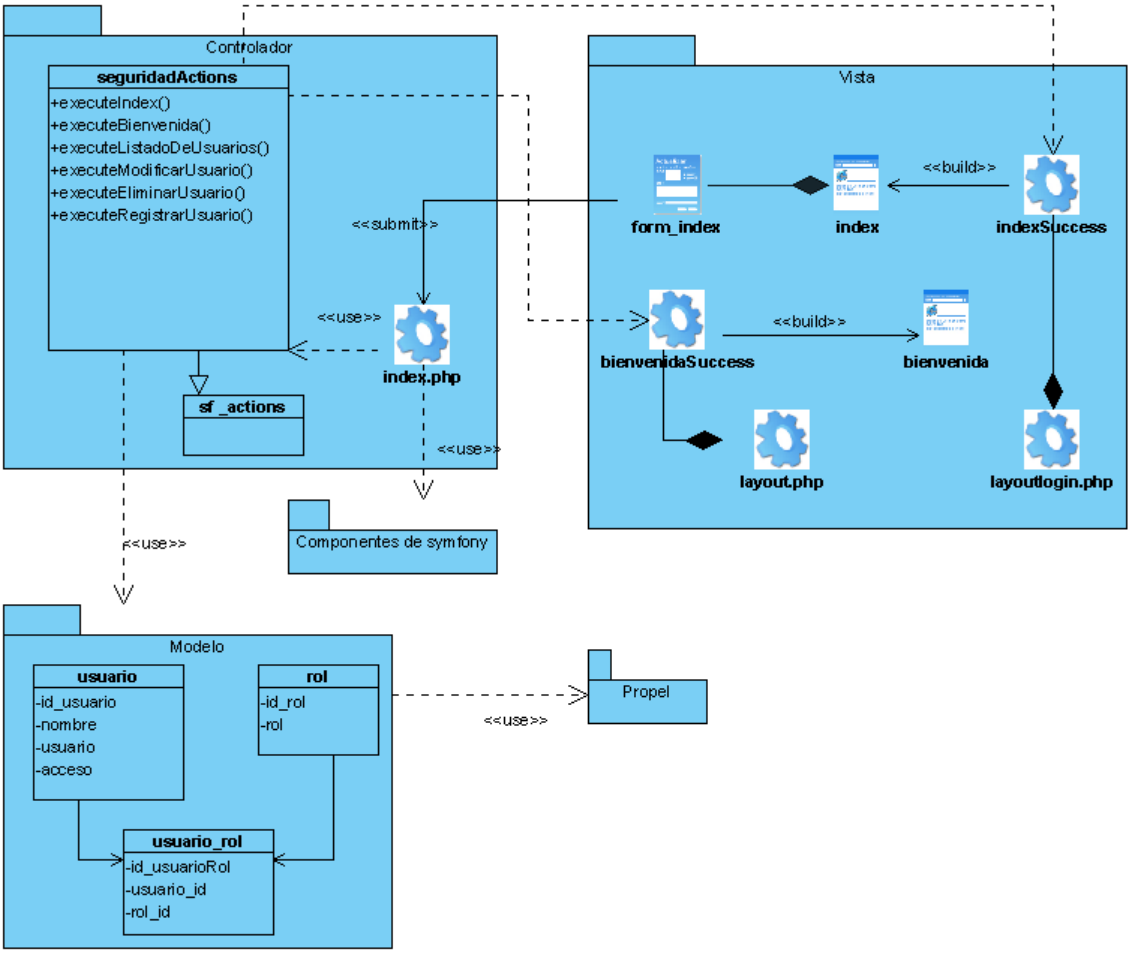

Figura 36 Diagrama de clases: CUS Autenticar Usuario.

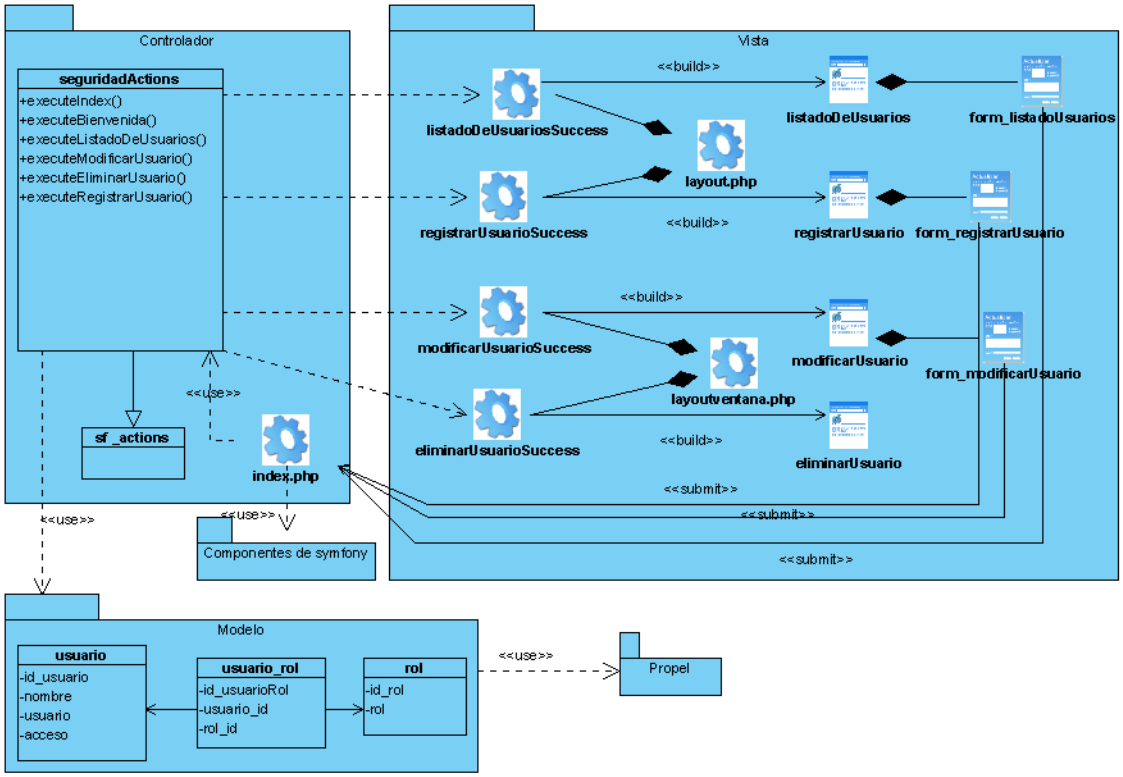

Figura 37 Diagrama de clases CUS Gestionar Usuario.

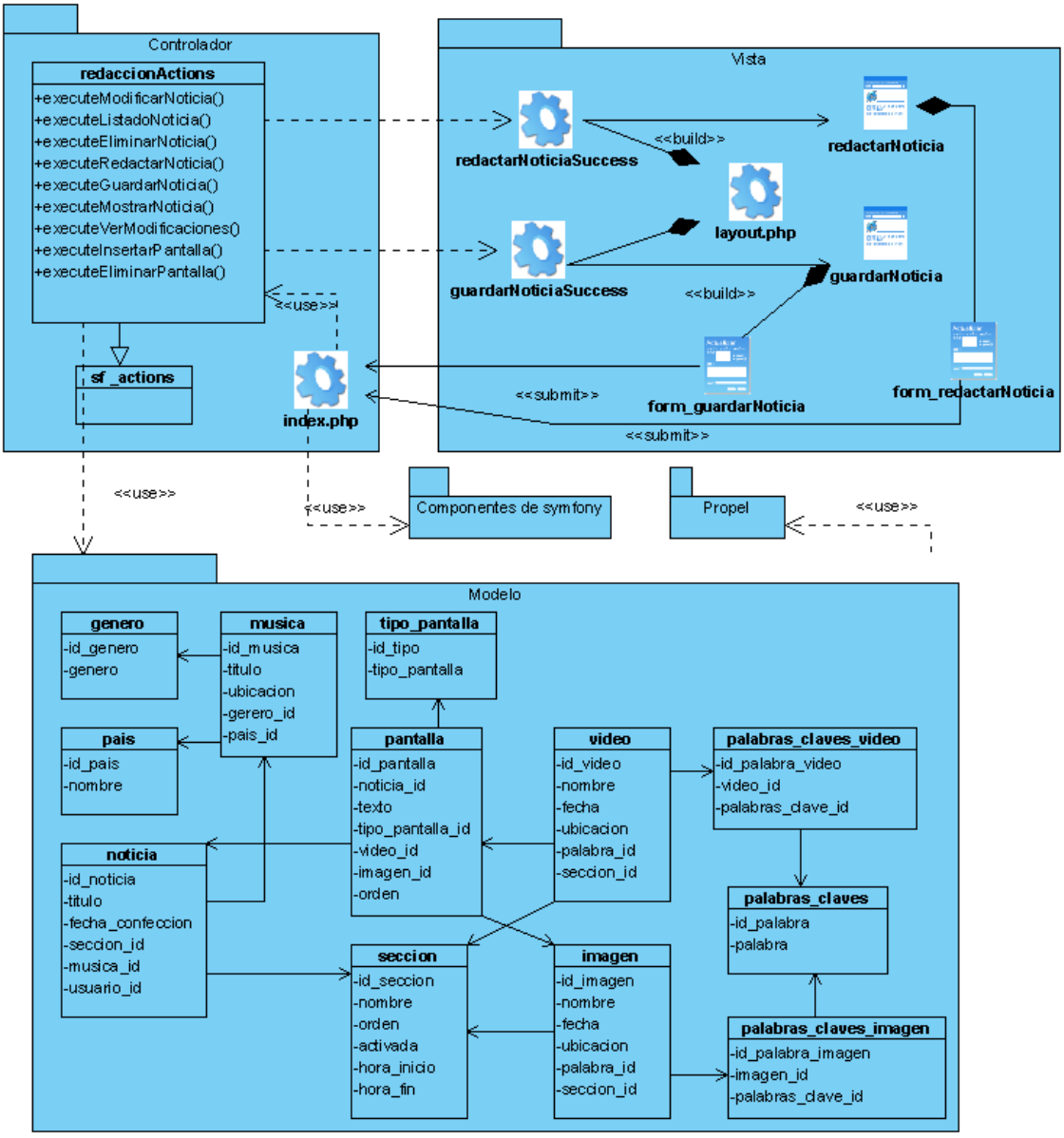

Figura 38 Diagrama de clases: CUS Redactar Noticia.

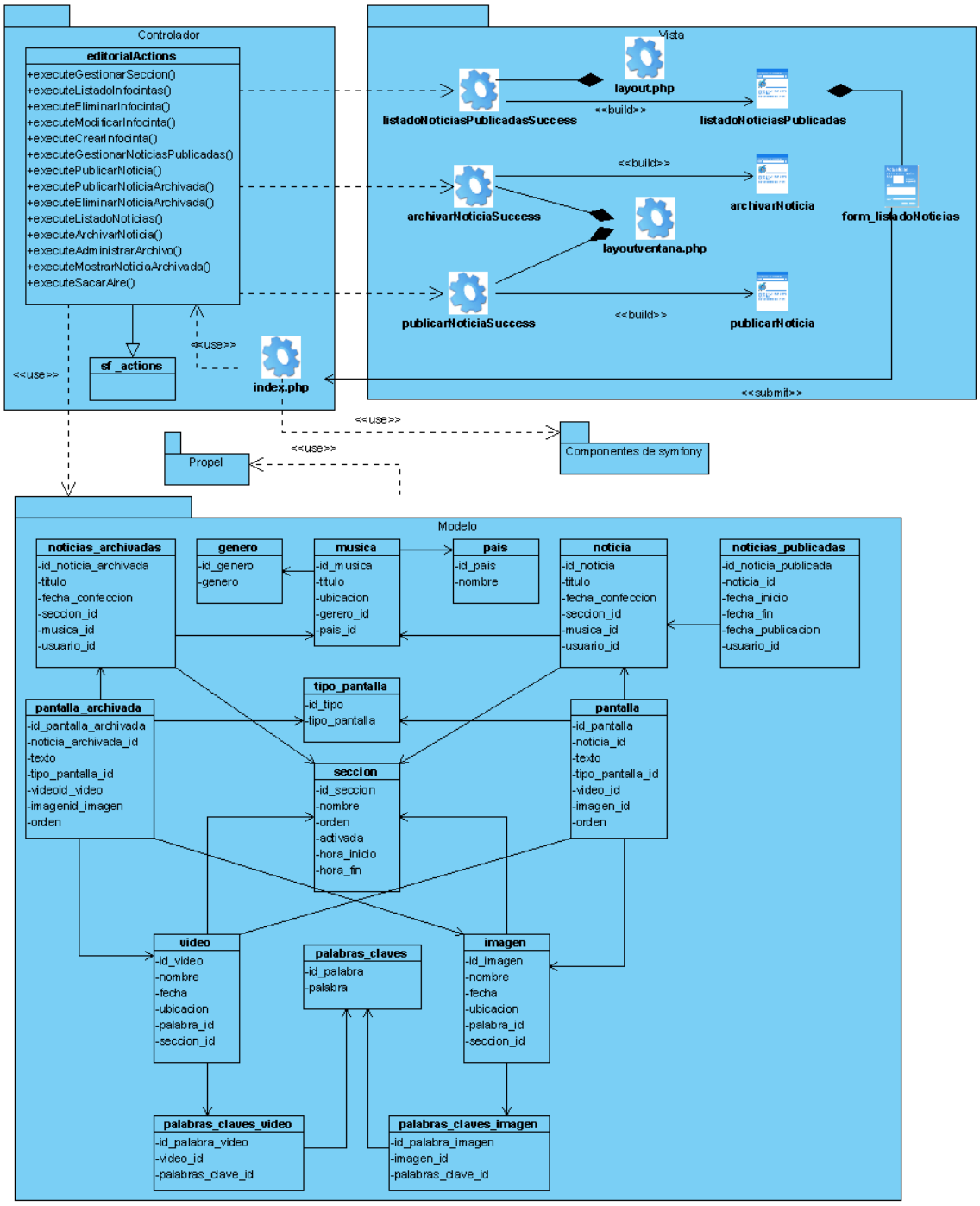

Figura 39 Diagrama de clases: CUS Publicar Noticia.

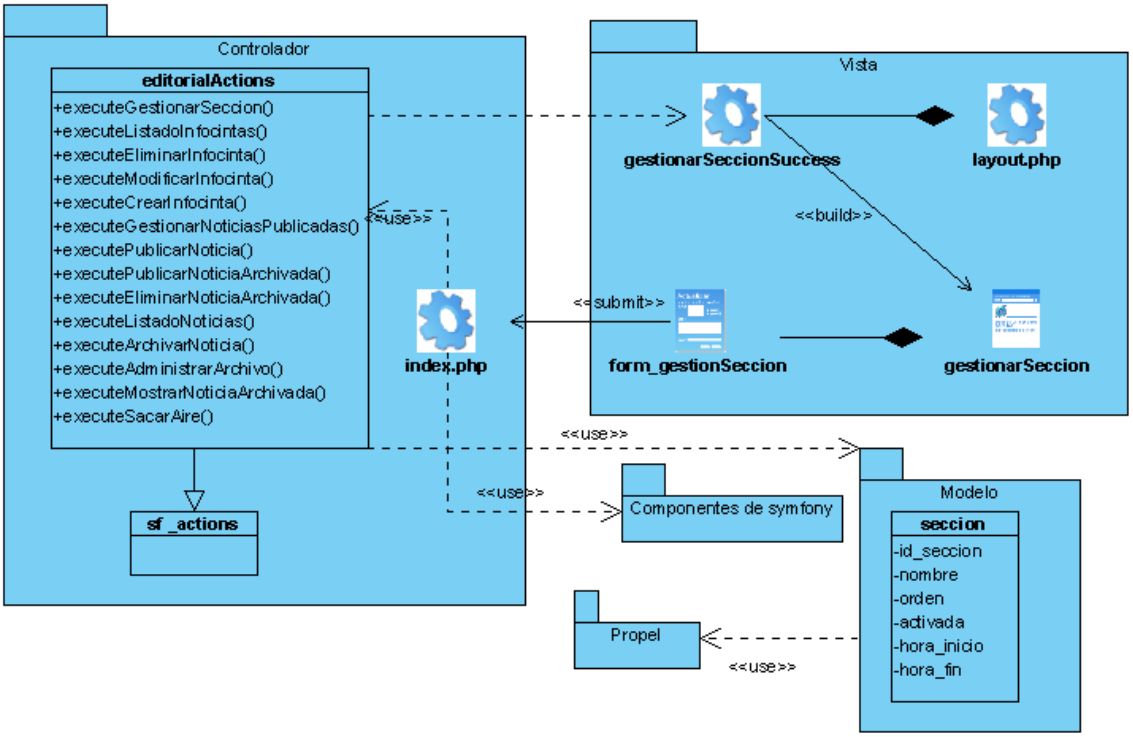

Figura 40 Diagrama de clases: CUS Gestionar Sección.

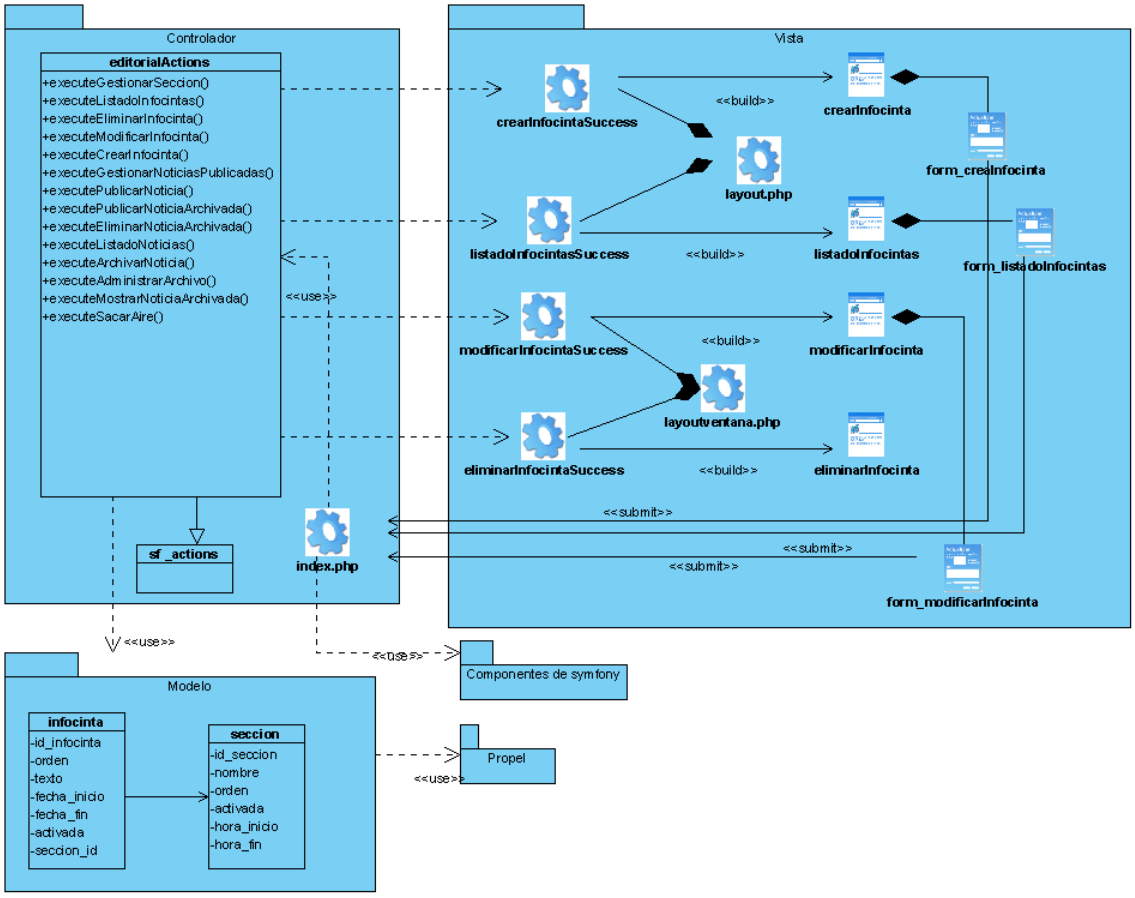

Figura 41 Diagrama de clases: CUS Gestionar Infocintas.

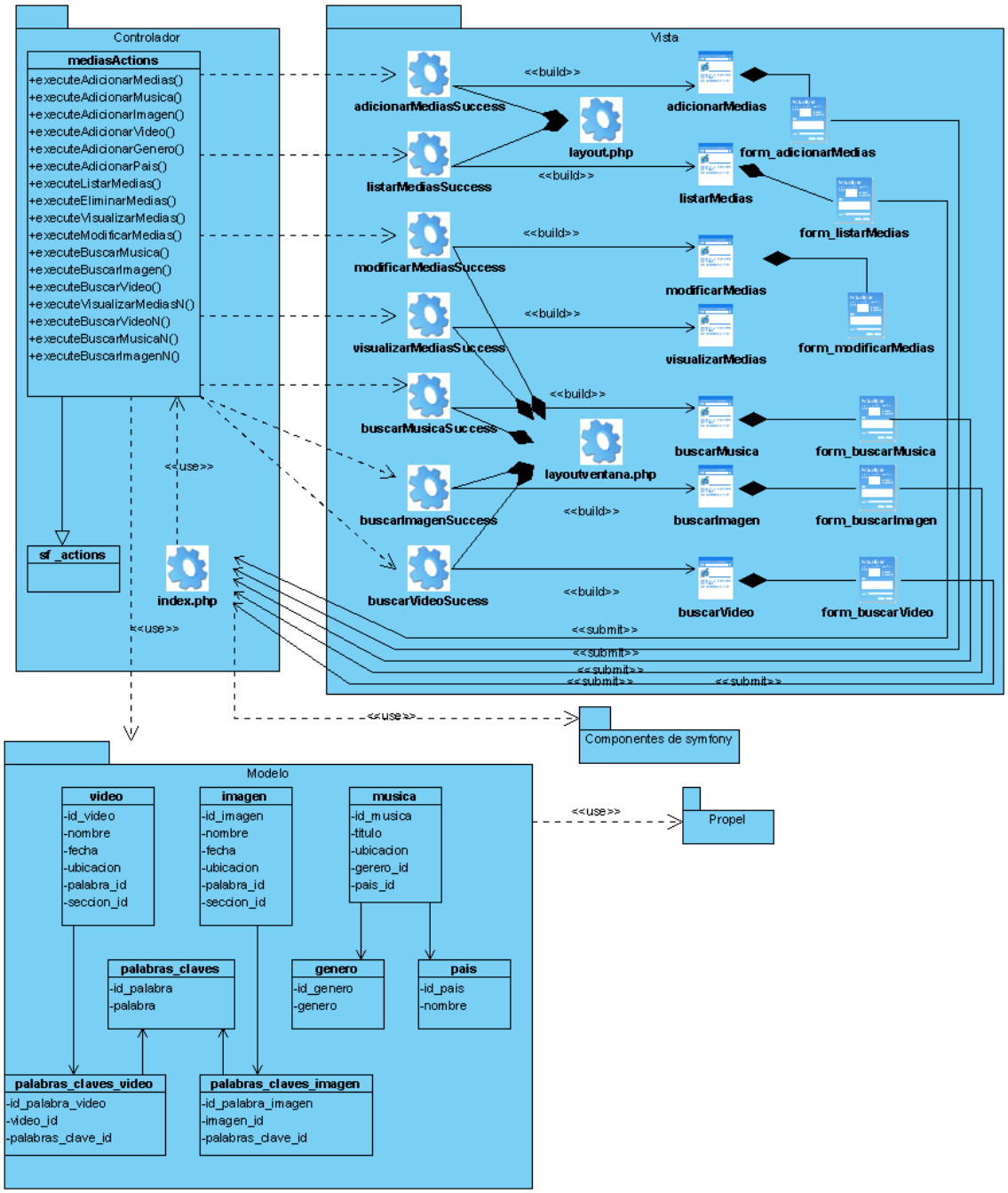

Figura 42 Diagrama de clases: CUS Administrar archivo de recursos multimedia.

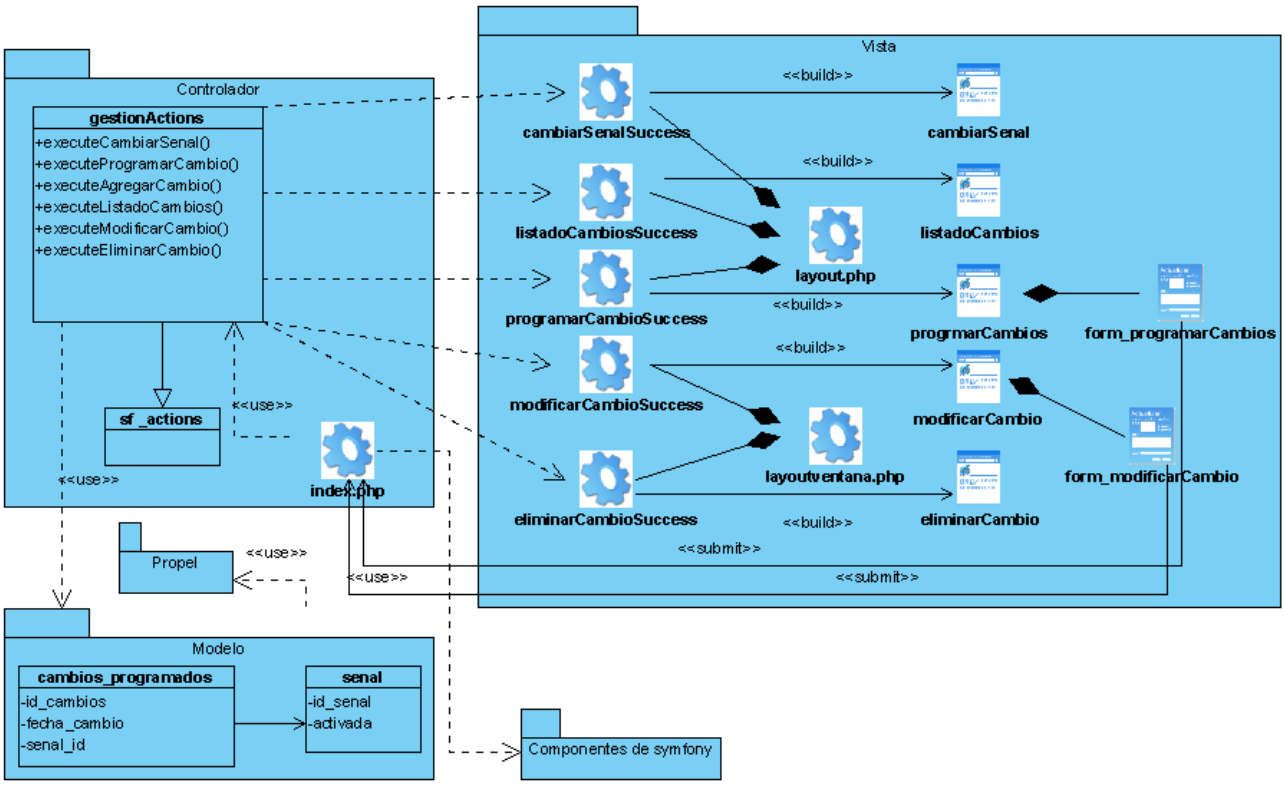

Figura 43 Diagrama de clases: CUS Gestionar señal del canal.

4.6 Vista Lógica

La vista lógica describe las clases más importantes que formarán parte del ciclo de desarrollo. Se describen los paquetes del sistema de una forma abstracta y las relaciones que existen entre ellos.

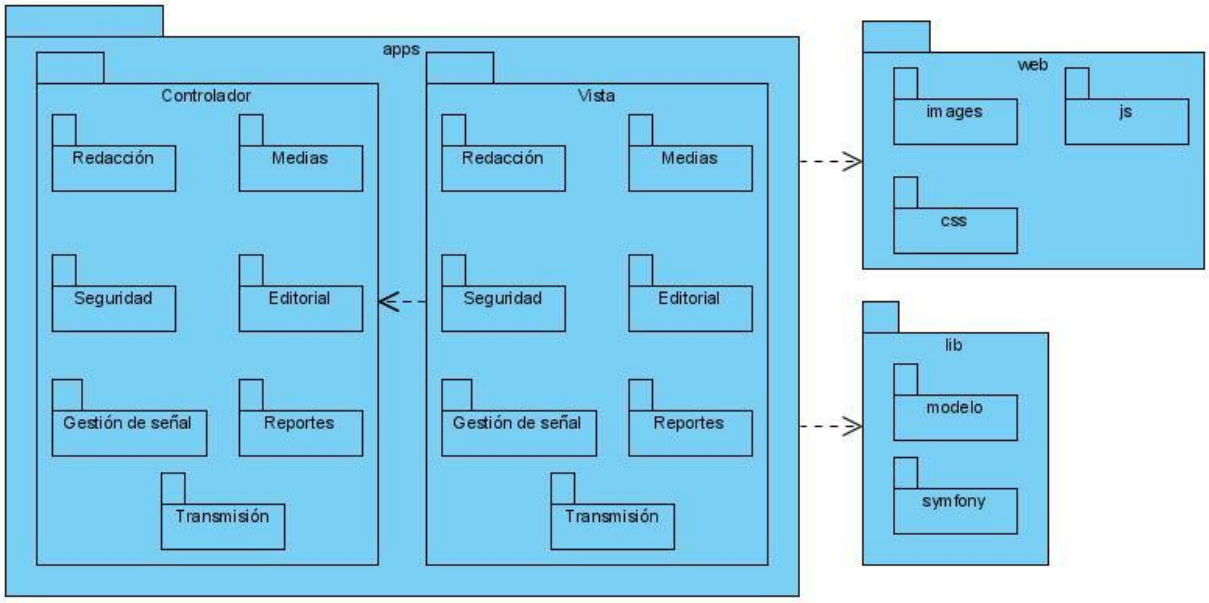

Figura 44 Diagrama de Paquetes.

Paquete	**Descripción**
apps	Contiene un directorio por cada aplicación del proyecto
Vista	Contiene las clases Interfaz del sistema, separadas por paquetes.
Controlador	Contiene las clases Control del sistema, separadas por paquetes.

web	La raíz del servidor web. Los únicos archivos accesibles desde Internet son los que se encuentran en este directorio
images	Contiene todas las imágenes que se utilizan en las interfaces del sistema.
css	Contiene los estilos que se aplican en las interfaces del sistema.
js	Contiene las clases desarrolladas en Java Script que se utilizan en las interfaces del sistema.
lib	Almacena las clases y librerías externas. Se suele guardar todo el código común a todas las aplicaciones del proyecto.
modelo	Guarda el modelo de objetos del proyecto.
symfony	Contiene los principales archivos del framework. Ejemplo clases y helper.

Tabla 2 Descripción de los paquetes del sistema.

A continuación se da una pequeña explicación de cada paquete del sistema:

- **Redacción**: Permite la redacción de las noticias y la gestión de las que no estén publicadas.
- **Medias**: Permite la administración del archivo de recursos multimedia, en este paquete se suben al sistema las medias que van a ser utilizadas en la confección de las noticias, se modifican y eliminan de acuerdo a los intereses del Ministerio.
- **Seguridad**: Controla el acceso de los usuarios al sistema y permite su gestión.
- **Editorial**: Permite la gestión de las secciones temáticas, las infocintas promocionales y las noticias que ya han sido publicadas, además permite que se administre el archivo de noticias del canal.

- **Gestión de Señal del Canal**: Permite toda la gestión de la señal del canal.
- **Reportes**: Da la posibilidad de generar reportes que ayuden a controlar la actividad de los redactores.
- **Transmisión**: Da la posibilidad de visualizar toda la información gestionada en los demás módulos.

Cada usuario que interactúe con el sistema debe tener al menos un rol asignado, estos roles determinarán el grado de acceso que tendrán dentro del sistema.

4.7 Vista de Implementación

La vista de implementación proporciona una descripción de las principales capas y subsistemas de componentes de la aplicación.

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se puede encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

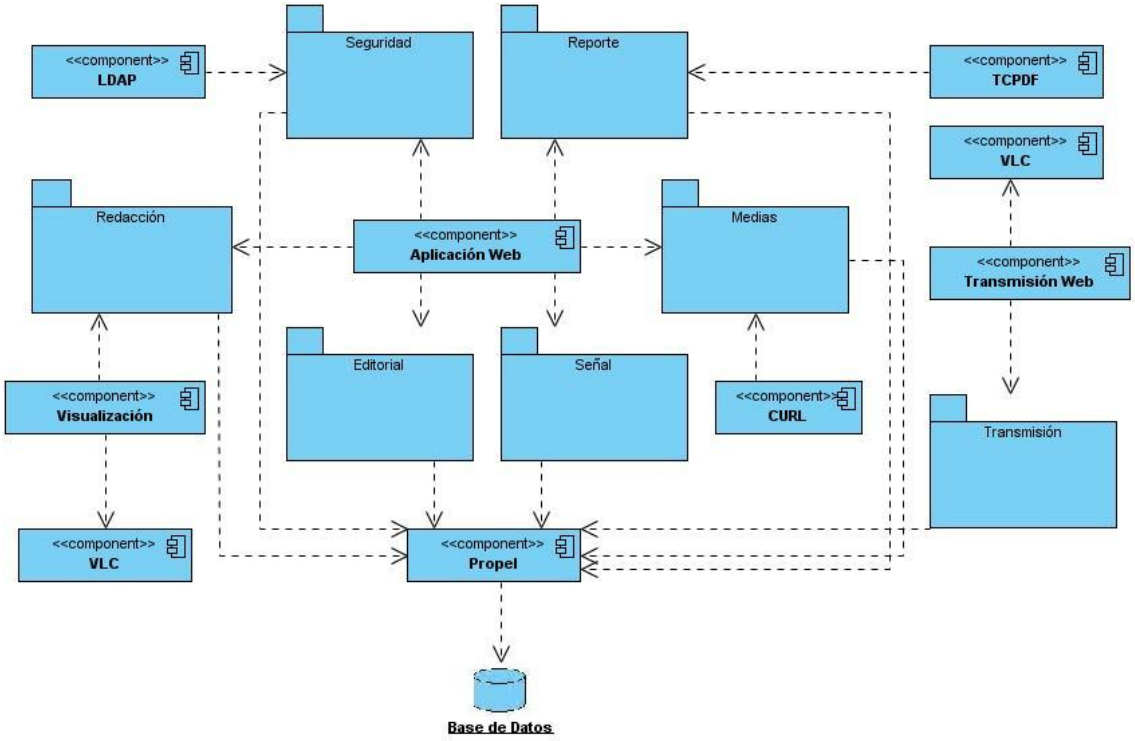

Figura 45 Diagrama de componente.

Cada uno de los paquetes representados en el diagrama anterior encapsulan varios componentes que se interrelacionan para concebir el correcto funcionamiento del sistema, estableciendo además las dependencias representadas.

4.8 Diagrama de despliegue

El diagrama de despliegue brinda información sobre la distribución física del sistema y cada uno de sus componentes, además de dar solución a parte de los requisitos no funcionales (hardware y software).

La arquitectura física del sistema puede ser acoplada en dependencia del tipo de red de televisión, seguidamente se muestran diagramas de instalación del sistema para una red satelital y otra por cable:

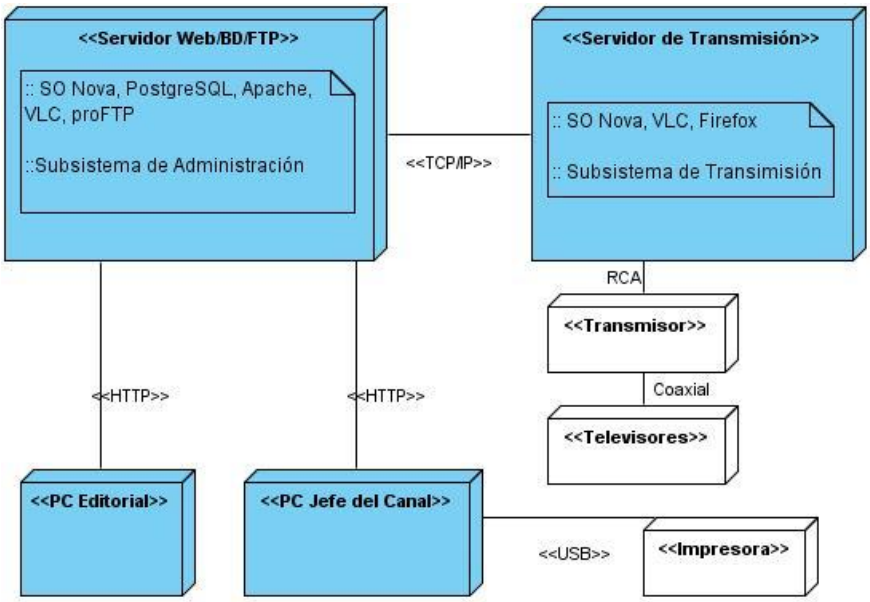

Figura 46 Diagrama de despliegue por cable.

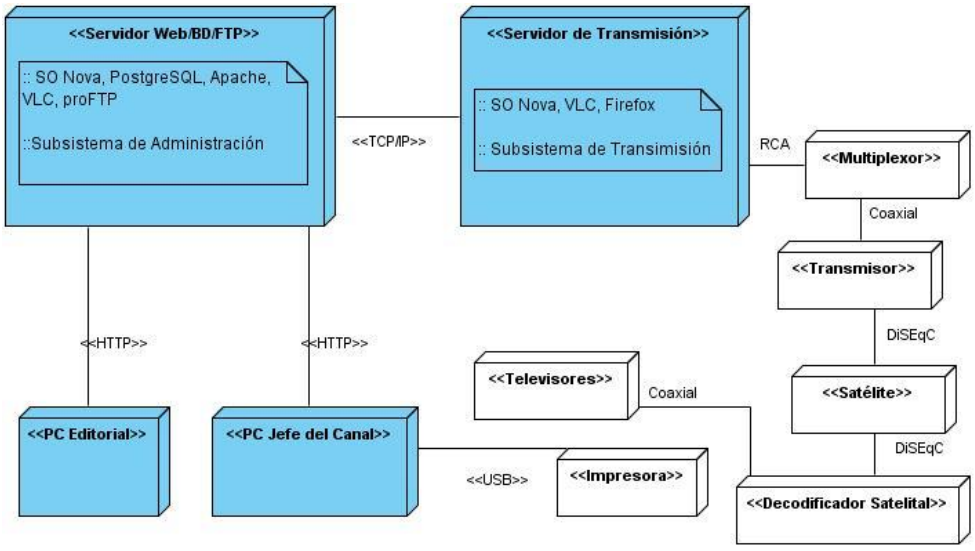

Figura 47 Diagrama de despliegue por satélite.

Como se puede apreciar el protocolo de comunicaciones o impulsos es sencillamente la forma en que un equipo receptor digital de satélite se "comunica" con accesorios. En este caso el protocolo que se usa

para la comunicación entre el transmisor, el satélite y los televisores es DiSEqC (Digital Satellite Equipment Control) quiere decir Control de equipo digital de satélite.

La Plataforma de Televisión Informativa provee un canal de televisión el cual es soportado y transmitido utilizando computadoras. La solución ha sido pensada para que sea adaptable a distintos entornos, los que pueden ser más sofisticados o en algún caso estar restringidos por un bajo presupuesto.

En caso que la entidad cliente no cuente con un gran presupuesto es posible utilizar un solo servidor que cumpla con ambas funciones. Otra solución posible sería disminuir las características tecnológicas hasta los requerimientos mínimos del equipamiento sin incidir negativamente en el rendimiento del sistema.

A continuación se muestran los requerimientos mínimos (filas azules claras) y óptimos (filas rojas claras) recomendados que debe cumplir el equipamiento tecnológico de la plataforma:

Variante de un (1) servidor:

Servidor	Procesador	Memoria RAM	Disco Duro	Tarjeta de Red	Tarjeta de Video
Administración	Pentium IV 2.8 GHz	1 Gb	120 Gb	Ethernet 10/100 Mbs	Capturadora Hauppauge WinTV PVR 350
-					
Transmisión	Dual-Core Xeon 1.60 GHz	4 Gb	500 Gb	Ethernet 10/100 Mbs	Exportadora ATI Radeon X300

Tabla 3 Ilustra los requerimientos mínimos para un servidor.

Variante de dos (2) servidores:

Servidor	Procesador	Memoria RAM	Disco Duro	Tarjeta de Red	Tarjeta de Video
Administración	Pentium IV 2.8 GHz	1 Gb	120 Gb	2 puertos Ethernet 10/100 Mbs	Capturadora Hauppauge WinTV PVR 350
	Dual-Core Xeon 1.60 GHz	4 Gb	500 Gb	2 puertos Ethernet Gigabit	
Transmisión	Pentium IV 2.8 GHz	512 Mb	40 Gb	Ethernet 10/100 Mbs	Exportadora ATI Radeon X300
	Dual-Core Xeon 1.60 GHz	1 Gb	80 Gb	Ethernet Gigabit	

Tabla 4 Ilustra los requerimientos mínimos para dos servidores

4.9 Vista de Datos

El sistema consta con tablas que son utilizadas por los subsistemas y módulos para permitir la persistencia de datos, para manejar estas tablas se utiliza como Gestor de Base de Datos el PostgreSQL, el mismo está bajo la licencia libre Berkeley Software Distribution (BSD) y presenta como principales características las siguientes:

- Máximo de base de datos: ILIMITADO
- Máximo de tamaño de tabla: 32TB
- Máximo de tamaño de registro: 1.6TB
- Máximo de tamaño de campo: 1GB
- Máximo de registros por Tabla: ILIMITADO
- Máximo de campos por tabla: 250 a 1600 (depende de los tipos usados)
- Máximo de índices por tabla: ILIMITADO

Con la utilización del framework de PHP Symfony se puede usar una de sus ventajas, la misma consiste en la abstracción de la base de datos.

Una capa de abstracción de bases de datos, permitirá cambiar el motor de la base de datos, de forma fácil y sencilla, de forma que la aplicación continúe funcionando de forma normal y sin efectuar ningún cambio en ella. Esto significa, que también se podrá desarrollar la aplicación para que trabaje contra cualquier motor de bases de datos, sin cambiar nuestra sintaxis ni las funciones a utilizar.

4.10 Conclusiones

En este capítulo se ha abordado la descripción de la arquitectura haciendo apoyo en artefactos de la arquitectura como son el modelo de análisis y modelo de diseño. Esto lleva consigo un acercamiento más al código y a un mejor entendimiento de cómo se va a desarrollar la Plataforma de Televisión Informativa. Se llega a la conclusión de que la arquitectura presentada reúne los parámetros y características necesarias para adaptarse de manera eficiente a sistemas de teletexto. Se concluye además que los artefactos, las 4 + 1 vistas de RUP arquitectónicas, presentados tienen gran importancia debido a que permite la documentación y esclarecimiento para los demás miembros del equipo de desarrollo.

CAPITULO 5

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

5.1 Introducción

En este capítulo se realizará una comparación de la arquitectura propuesta con dos arquitecturas de sistemas informativos que ya estén en funcionamiento. El análisis se centrará en las arquitecturas de los sistemas *Escripnauta* y *Estructure por las numerosas semejanzas que tienen con PRIMICIA*. Viendo así las ventajas y desventajas que pueda tener una de otra permitirá validar la solución propuesta.

5.2 Sistema Escripnauta

El Escripnauta es una aplicación desarrollada totalmente con Software Libre, en el sistema operativo Debian, orientada a la creación de redacciones multimedia en línea. El objetivo fundamental es la realización de una herramienta que permita la creación remota y cooperativa de presentaciones multimedia, integrando texto, imágenes y sonidos mediante el uso de sólo un navegador web estándar y una conexión a Internet. El sistema ha sido íntegramente desarrollado con el software disponible en la distribución Debian, principalmente Apache, PHP, MySQL, Ming [7]y utilidades de ImageMagick[8].

El sistema se basa en una arquitectura cliente-servidor centralizada. La interfaz de usuario reside completamente en el navegador web. En el servidor residen todos los demás módulos: base de datos y middleware de acceso a datos, lógica de edición, gestor de contenidos (galería) y el generador de las presentaciones.

El hecho de ser una aplicación cuya lógica está totalmente centralizada confiere al sistema una gran flexibilidad, tanto de desarrollo como de acceso y utilización. De esta forma se da la oportunidad de

[7] Ming es una librería para generar archivos de Macromedia Flash (swf).
[8] ImageMagick es una suite de software para crear, editar y componer imágenes de mapa de bits.

desarrollar completamente el sistema sobre Linux al ser independiente de las instalaciones y recursos de los usuarios.

El sistema funciona sobre un servidor Apache, con base de datos MySQL, PHP como lenguaje de programación y las librerías Ming para la generación dinámica de las presentaciones a partir de la información almacenada en la base de datos. En los clientes se utiliza principalmente DHTML para la edición de las presentaciones y SWF para la salida multimedia.

5.2.1 Conclusiones de la arquitectura de Escripnauta

Luego de haber analizado la arquitectura del sistema Escripnauta, en resumen se detectaron los siguientes parámetros:

- **Patrón Arquitectónico**: Modelo Vista Controlador (MVC).
- **Arquitectura de Aplicación**: Cliente – Servidor (aplicación Web).
- **Escenario de despliegue**: Un servidor donde se ejecuta la aplicación.
- **Lenguajes de Programación**: PHP, DHTML.
- **Gestor de Bases de Datos**: MySQL.
- **Servidor Web**: Apache.
- **Componentes**: librerías Ming.
- **Sistema Operativo**: Debian (Plataforma *nix).
- **Conexión de Red:** Ethernet Gigabit.

Según los parámetros que constituyen los elementos principales del sistema Escripnauta, se evidencia un 62.5 % de semejanza, mientras en un 37.5% los parámetros no convergen con PRIMICIA. Aquellos parámetros donde no existen coincidencias son:

Parámetros	Escripnauta	PRIMICIA
Gestor de Base de Dato	MySQL	PostgreSQL
Componentes	Ming	Symfony
Sistema Operativo	Debian	Nova

Tabla 5 Ilustra los parámetros donde no existen coincidencia.

No se tiene en cuenta la plataforma de Sistema Operativo, pues ambas pertenecen a la familia Unix. La diferencia es que ambos sistemas usan diferentes distribuciones de Linux. Esto tiene gran ventaja pues ambas arquitecturas abogan por software libre, que es más económico, tiene libertad de uso y distribución y una independencia tecnológica.

5.3 Sistema Estructure

La plataforma audiovisual Estructure tiene gran nivel en la gestión y producción de contenidos audiovisuales, donde los procesos suceden totalmente integrados, organizados y controlados. Controla la cadena productiva en la realización de noticias, gracias a las ventajas que aporta la informática.

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, configurando grupos de usuarios y redes, dimensionando el sistema de forma modular y escalable en base al enfoque productivo de cada empresa usuaria.

Los sistemas Estructure se componen de muy diversas tecnologías tanto de software como de hardware, superpuestas en capas de funcionamiento e interrelacionadas y comunicadas entre sí. Además tienen las siguientes características:

- Soporte de múltiples formatos de entrada y salida: H264, MXF, DV, DVCPRO, MPEG2, MJPEG1, I-Frame, MPEG-4, WMW, FLV, 3G, 3G2, entre otros.

- Captura simultánea en alta y baja calidad.
- Audio Digital Estéreo 48 KHz y analógico.
- Componentes analógicos, S-Video y compuesto.
- Soporte IEEE 1394.
- Uso de redes Ethernet Gigabit.

Entre otras muchas funcionalidades que no se plasman en este documento debido a que no son de gran interés para la comparación de este producto con la Plataforma de Televisión Informativa.

5.3.1 Conclusiones de la arquitectura de Estructure

Luego de haber analizado la Arquitectura del sistema Estructure, en resumen se detectaron los siguientes parámetros:

- **Patrón Arquitectónico**: En capas.
- **Arquitectura de Aplicación**: Cliente – Servidor (aplicación de Escritorio).
- **Escenario de despliegue**: Cuatros Servidores (captura, almacenamiento, administración y emisión). .
- **Gestor de Base de Dato**: SQL Server
- **Conexión de Red:** Ethernet Gigabit.

Según los parámetros que constituyen los elementos principales del sistema Estructure, se evidencia un 60 % de semejanza, mientras en un 40% los parámetros no convergen con PRIMICIA. Aquellos parámetros donde no existen coincidencias son:

Parámetros	**Estructure**	**PRIMICIA**
Patrón Arquitectónico	En capas	MVC
Gestor de Base de Dato	SQL Server	PostgresSQL

Tabla 6 Ilustra los parámetros donde no existen coincidencia.

5.4 Conclusiones

En este capítulo se abordó la descripción de dos arquitecturas internacionales de sistema informativo, analizando sus características, ventajas y similitudes con respecto a PRIMICIA. De esta se concluye que ambas arquitecturas presentan un punto arquitectónicamente aproximado, lo cual contribuye a la validación de la arquitectura del producto PRIMICIA. Además se evidencia que la arquitectura propuesta tiene puntos de acercamiento en cuanto a las arquitecturas presentadas internacionalmente para sistemas informativos. Esto contribuye a demostrar que los parámetros de la arquitectura para la Plataforma de Televisión Informativa están aceptados en todo el mundo en cuanto a sistemas de teletexto.

CONCLUSIONES GENERALES

La arquitectura desarrollada en esta investigación constituye el núcleo principal del sistema informativo PRIMICIA, software que va a dar la posibilidad de mantener informados a un grupo de personas según el lugar donde se despliegue como producto final.

En el presente trabajo se hizo un estudio de los principales aspectos de la descripción arquitectónica, seleccionando el estilo arquitectónico adecuado para la solución que forma parte del núcleo de la arquitectura, así como sus componentes, conexiones, configuraciones y restricciones. Como parte de la documentación se generaron las 4 +1 vistas arquitectónicas de RUP, tales como: *Vista de Proceso*, *Vista lógica*, *Vista de Implementación, Vista de Despliegue* y *Vista de Casos de Uso,* de contribuyendo a la conceptualización de la arquitectura del producto PRIMICIA.

Además se obtuvo como resultado la definición de las herramientas a utilizar para el desarrollo del sistema y configuración de los puestos de trabajo, logrando así consolidar los lenguajes de programación a utilizar. Y se definió la línea base de la arquitectura que tiene la importancia de definir ya las funcionalidades que va a tener el sistema y por el cual se va a regir para el desarrollo del proyecto. A partir de la línea base de la arquitectura ya se puede pasar a la siguiente fase de RUP, la fase de construcción, para entrar a la implementación del sistema informativo.

Finalmente se puede afirmar que se cumplieron los objetivos trazados, logrando definir la arquitectura de software del producto PRIMICIA en concordancia con los requisitos establecidos. La arquitectura fue desarrollada bajo la premisa de lograr una mayor confiabilidad en la calidad del producto final.

RECOMENDACIONES

Al término del presente trabajo de diploma el autor recomienda:

- El refinamiento constante de la arquitectura propuesta durante el ciclo de desarrollo y en vistas de versiones superiores del producto.
- Realizar un estudio de la Arquitectura SOA para la incorporación de un componente de Servicios Web que tendrá la responsabilidad de brindar y/o consumir información de sistemas externos.
- Validar la arquitectura propuesta mediante algún método evaluación de arquitecturas, con el objetivo de identificar las principales debilidades de la misma, tanto en la descripción arquitectónica como en el propio diseño arquitectónico.

REFERENCIAS BIBLIOGRÁFICAS

1. *Departamento de Ciencias de la Computación y Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pensilvania. Sistemas de gran escala requieren de más alto nivel abstracciones.* EE.UU : ACM New York, 1989. 14. 0163-5948.

2. **Paul Clements, len Bass,Rick Kazman.** *Arquitectura de Software en la práctica (2 ª edición), Addison-Wesley Profesional.* 2003. 0321154959.

3. **Gómez, Omar Salvador.** Software Guru. [En línea] http://www.sg.com.mx/content/view/792].

4. Catálogo de productos VSN Matic. *Catálogo de productos VSN Matic.* [En línea] http://www.provideo.es/productos/vsn_matic.html..

5. Estructure Media Systems. *Estructure Media Systems.* [En línea] 2008. http://www.estructuretv.com/.

6. Introducción a Patrones. *Introducción a Patrones.* [En línea] http://agamenon.uniandes.edu.co/~pfiguero/soo/Magister_Patrones/intropatrones.html.

7. Unidad I. *Arquitectura de software Unidad I.* [En línea] http://www.fceia.unr.edu.ar/ingsoft/unidad11-4.pdf.

8. **Durán, Julio Alberto Leyva.** *Desarrollo de la Arquitectura del Sistema Automatizado de Control de Gestión de Indicadores de Refinación, SACGIR.* Habana : s.n., 2008.

9. Arquitectura de Capas. *Arquitectura de Capas.* [En línea] http://ccc.inaoep.mx/~emorales/Cursos/RdeC/node112.html.

10. **Javier J. Gutiérrez.** ¿Qué es un framework web? *¿Qué es un framework web?* [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

11. **Lovelle, Sonia Pérez.** *Perfil para representar una arquitectura de componentes en UML.* CUJAE, CUBA : s.n.

12. Desarrollo Web. *Desarrollo Web.* [En línea] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

13. **W3C.** HTML 4.0 Specification. *HTML 4.0 Specification.* [En línea] 24 de 4 de 1998. http://ftp.tdcnorge.no/pub/networking/www/doc/html40.pdf.

14. Eclipse. *Eclipse.* [En línea] 2009. http://www.eclipse.org/org/.

15. El Proceso Unificado de Desarrollo. *El Proceso Unificado de Desarrollo.* [En línea] http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

BIBLIOGRAFÍA CONSULTADA

1. **Stopford, Andrew.** *Proyectos profesionales con PHP .* s.l. : Anaya Multimedia - Anaya Interactiva , 2002. 84-415-1418-6.

2. PHP 5. s.l. : Anaya Multimedia- Anaya Interactiva , 2004. 84-415-1770-3.

3. **Corp, Microsoft.** *Enterprise Solution Patterns.* s.l. : MIcrosoft Press, 2003.

4. **Flower, Martin.** *Enterprise Application Architecture Patterns .* s.l. : Addison- Wesley , 2003.

5. **Cueva Lovelle, Juan Manuel.** Tecnología de Objetos: Patrones de Diseño. 2004.
6. **Shalloway, Alan.** Design Patterns Explained: A New perspective on Object Oriented Desing. s.l. : Pearson Education , 2001.
7. **Bosch, J.** Design Patterns as Language Constructs: Journal of Object Oriented Programming(JOOP). 1998.
8. **Hernandez Hernandez, Ma. E, Alvarez Carrión, G y MUñoz Arteaga, J.** Patrones de Interacción para el Diseño de Interfaces Web Usables. [En línea] 9 de Abril de 2003. http://ccc.inaoep.mx/~grodrig/Descargas/com10017.pdf .
9. **Pressman, Roger.** Putnam y Myers: Five core metrics. Dorset House. 2003.
10. **PostgreSQL Global Development Group .** PostgreSQL. [Online] tinysofa, 1996. http://www.postgresql.org/.

11. *An Introduction to Software Architecture.* **David Garlan, Mary Shaw.** New Jersey : V.Ambriola and G.Tortora, 1994, Vol. 1. 15213-3890.

12. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : Prentice Hall. 84-205-3438-2.

13. **Microsystems, Sun.** MySQL. [Online] Sun Microsystems, Inc. [Cited: enero 30, 2009.] http://www.mysql.com/.

14. **David Gallardo, Ed Burnette , Robert McGovern.** *Eclipse in Action: A Guide for Web Developers.* s.l. : Manning Publications, 2003.

15. **Zarzuela, Jorge Ferrer.** *Metodologías Ágiles.* [Documento publicado en http://libresoft.dat.escet.urjc.es/html/downloads/ferrer-20030312.pdf] Madrid : s.n.

GLOSARIO DE TÉRMINOS

Apache: Servidor de páginas Web de código abierto para diferentes plataformas (UNIX, Windows, etc.).

APIs (Application Programming Interface): Interfaz de Programación de Aplicaciones.

Asíncrona: Sistema que no responde a un reloj constante.

Base de datos: Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar.

CGI (Common Gateway Interface) : Interfaz de entrada común (en español) es una importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web.

Clase(s): Abstracciones de objetos. Una clase es una definición de un objeto. Un objeto es una instancia de una clase.

Cliente/Servidor: Método de distribución de información o de archivos en el cual la agrupación central, servidor, almacena los archivos y los hace disponibles para solicitudes de aplicaciones cliente.

Consultas: Una consulta SQL es tipo de consulta a una base de datos empleando lenguaje SQL.

Frameworks: Plataforma, entorno, marco de trabajo.

Herramientas CASE (Computer Aided Software Engineering): Se incluyen una serie de herramientas, lenguajes y técnicas de programación que permiten la generación de aplicaciones de manera semiautomática.

Hipertexto (Hypertext): Es un sistema para escribir y mostrar texto que enlaza a información adicional sobre ese texto.

HTML: Lenguaje de marcado de hipertexto, es el lenguaje autoritario para crear documentos en la World Wide Web. Define la estructura de un documento web usando etiquetas y atributos.

Plug-ins: Es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

RUP: Metodología de desarrollo de software basada en UML. Organiza el desarrollo de software en 4 fases.

Triggers (Trigger o disparador): Se defina así a una subrutina que es ejecutada de manera automática cuando se produce algún tipo de transacción (inserción, borrado o actualización) en la tabla de una base de datos.