UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

“Diseño de la Arquitectura de Software para el Sistema de Captura y Catalogación de Medias”.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

AUTOR:

Mailin Muñoz Peña

TUTOR:

Ing.Yoandri Quintana Rondón

Cuidad de La Habana, 29 de junio, 2011

Año 53 de la Revolución

DEDICATORIA

Dedico esta tesis a mis padres, especialmente a mi mamá por apoyarme y confiar plenamente en mí, todo lo que soy y seré es gracias a ella.

También quiero dedicárselo a mis hermanos, principalmente a Carlitos, espero poder verlo graduarse algún día como yo hoy.

A mis abuelos por hacer lo posible y lo imposible para ayudarme en estos 5 años.

A mi tía Merle por ser una guía en mis estudios y en mi vida.

A toda mi familia por estar siempre al tanto de mí y por quererme tanto.

AGRADECIMIENTOS

A mis padres por ser mi ejemplo y guía en todo momento, por todo el apoyo que me han dado toda la vida, por tanto amor, sacrificio y dedicación.

A mis tíos, primos y hermanos, por tener seguridad de que lograría este éxito, en especial a mi tía Merle, sé que estará muy orgullosa de mí y porque todo su sacrificio no ha sido en vano.

A mis abuelos, especialmente Merle y Leonel que de una forma u otra me han ayudado a llegar a donde estoy y confiaron siempre en mí.

A Camilo por ser la persona con la que he compartido todos estos años en los momentos malos y buenos.

A las niñas del apartamento por ser como una familia de amigos, y a todos los nuevos y viejos amigos UCI.

A mi tutor por todo el apoyo y dedicación a este trabajo, ha sido un honor trabajar con él.

OPINIONES Y AVALES

Jornada Científica Estudiantil

9na jornada científica estudiantil

UCI

Se le otorga el presente

Reconocimiento

A: Diseño de la arquitectura de software para el Sistema de Captura y Catalogación de media.

Por haber obtenido la categoría de

Destacado

En la 9na edición de la Jornada Científica Estudiantil

Alexis Zamue Hernández
Presidente Feu-6

Luis Manuel Vidal Piña
Presidente Consejo Científico
Fac-6

Yanet Villamueva Armenteros
Decano Facultad-6

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los _29_ días del mes de junio del año _2011__.

"[Insertar nombre(s) de autor(es)]" "[Insertar nombre(s) de tutor(es)]"

_____ _____

Mailin Muñoz Peña Yoandri Quintana Rondón

DATOS DE CONTACTO

Síntesis del Tutor

Nombre y Apellidos: Yoandri Quintana Rondón

Especialidad: Ingeniero en Ciencias Informáticas

Años de Experiencia: 2

Teléfono Particular: 53542029

Dirección electrónica para correspondencia: ygrondon@uci.cu

RESUMEN

En el mundo actual las Tecnologías de la Información y las Comunicaciones (TIC) cada día tienen mayor importancia. Uno de los medios más utilizados para difundir los avances de las TIC es la televisión. Este importante avance tecnológico además de mantener informados, permite modificar la forma de conocer y comprender la sociedad. Son muchas las empresas que se dedican a realizar aplicaciones relacionadas con la televisión, pero debido a la gran competencia que existe, las empresas y organizaciones demandan una buena realización de sistemas de software. Para lograr alcanzar los mejores resultados es muy importante el diseño de la arquitectura, pues con esto se contribuye a una evolución eficiente en el desarrollo del software.

No todos los sistemas logran el éxito y una de las causas es el mal diseño de la arquitectura. Muchas veces no se cuenta con la arquitectura correcta en el sistema, lo que lleva el fracaso del mismo. Este trabajo tiene como objetivo lograr una arquitectura robusta y confiable para el desarrollo del proyecto Sistema de Captura y Catalogación de Medias basándose en las principales funcionalidades que brindan las televisoras. Se analizan los errores cometidos en la primera arquitectura, aunque las prestaciones a brindar el proyecto hoy son diferentes al anterior. Se pretende que la arquitectura resultante sea factible para aplicarla en cualquier sistema de similar propósito.

PALABRAS CLAVES

Arquitectura, Catalogación, Streaming

ABSTRACT

In today's world the Information Technology and Communications (ICT) each day they are more important. One of the most used medium to communicate the progress of ICT is television. This technological advance also keeps people informed, to modify the form of knowledge and understanding of society. There are many companies that are devoted to television-related applications, but due to the strong competition that exists, companies and organizations require sound implementation of software systems. Have better results is very important to design architecture, because with this development contributes to efficient software development.

Not all systems achieve success and one of the causes is the poor design of the architecture. Many times you do not have the right architecture in the system, leading to project failure. This work aims to achieve a robust and reliable architecture for project development System for Capture and Cataloguing Medias based on the main features offered by television. We analyze the errors in the first architecture; although the project provides us benefits are different than before. It is intended that the resulting architecture is feasible to implement in any system of similar purpose.

Keywords:

Architecture, Catalogation, Streaming.

TABLAS Y FIGURAS

ÍNDICE

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 las Comunicaciones (TIC) ha contribuido a que los sistemas de almacenamiento sean cada vez más grandes y eficientes. Uno de los entornos en que se aprecia la evolución de estos sistemas es en la televisión (TV).

La Televisión nace a partir de la conjunción de una serie de fenómenos e investigaciones simultáneas pero desarrolladas aisladamente. El original descubrimiento de la "fototelegrafía" a mediados del siglo XIX (La palabra TELEVISIÓN no sería usada sino hasta 1900), debe sus avances y desarrollo a varios investigadores que experimentaron con la transmisión de imágenes vía ondas electromagnéticas. Las primeras transmisiones experimentales fueron realizadas en Estados Unidos de Norteamérica (U.S.A.). Fue en Julio de 1928 cuando desde la estación experimental W3XK de Washington, JENKINS comenzó a transmitir imágenes exploradas principalmente de películas con cierta regularidad y con una definición de 48 líneas.

La televisión ha alcanzado una gran expansión en todo el mundo. Según un análisis realizado en el año 2009 existen más de 300 canales de TV y una audiencia, según el número de aparatos por hogares (más de 60 millones), de más de doscientos millones de personas. Existen varias clasificaciones según se han ido desarrollando las tecnologías: la televisión terrestre, por cable, vía satélite, IP y 3D, donde la forma de difusión puede ser analógica o digital. Además existen otros tipos dentro de ellos se destaca la televisión informativa, que es aquella dedicada a la transmisión de informaciones de forma constante en diferentes formatos. Para un mejor entendimiento es importante saber que una señal de TV está compuesta por señales de audio y video, dichas señales se envían de forma conjunta y son moduladas de diferentes maneras.

El video y el audio tienen gran poder para transmitir información a otras personas, ya sea educativa, comercial o simplemente para el entretenimiento. Hace 20 años era muy difícil logar difundir información en estos formatos o de lograrse se hacía con muy mala calidad. Esto estaba dado por el poco ancho de banda con que contaban las redes en sus inicios y el gran tamaño que tienen los archivos de audio y video. Actualmente la sociedad utiliza la tecnología como un instrumento indispensable en la vida diaria.

Introducción

El uso de la multimedia en la informática es un fenómeno reciente, cuya importancia radica en su similitud con la forma en la que los humanos perciben la información del medio ambiente que les rodea, lo cual es el resultado de la combinación de diferentes medios en función de la comprensión y el entendimiento, siendo su fin el de dotar de mayor interactividad al ordenador y en consecuencia proporcionar una mayor usabilidad del software. En Cuba, esta se ha extendido hacia todos los centros educacionales del país, constituyendo un elemento imprescindible para la enseñanza como medio de aprendizaje.

La Universidad de las Ciencias Informáticas (UCI) es el centro principal del desarrollo de la informática en Cuba. Presenta una distribución por facultades, y por cada facultad centros de desarrollo. En el departamento de Señales Digitales perteneciente al centro de Geoinformática y Señales Digitales (GEySED) se realizan sistemas relacionados con las medias (audio, video, imágenes). Para que estos sistemas funcionen perfectamente debe haberse realizado un diseño eficiente del mismo, en el cual un factor determinante es la arquitectura de software.

Las grandes compañías productoras de software se apoyan en la Arquitectura de Software (AS), pues un buen desarrollo de ésta, posibilita en gran parte la evolución eficiente de sus sistemas. Según análisis realizados muchos sistemas no alcanzan sus metas porque no existe una definición clara de la AS, es por eso que se considera a la AS como una etapa fundamental para lograr obtener los resultados esperados en un software. Este tema muestra un gran impacto en el mundo de las televisoras, pues son varias las empresas que presentan información desorganizada, pérdidas de recursos, esfuerzos duplicados y gastos innecesarios. En Cuba existen problemas similares, pero la UCI como su centro más importante en el desarrollo de software, dirige sus fuerzas en la informatización de la sociedad, así como a dar solución a estos fallos.

La UCI es un eslabón significativo en la economía de Cuba, son varios los sistemas realizados con resultados relevantes a nivel mundial. El proyecto Captura y Catalogación de Medias del departamento de Señales Digitales en el año 2009 se encontraba desarrollando un sistema que permitiera capturar y catalogar materiales audiovisuales. El mismo debía brindar las siguientes funcionalidades: captura de señales de video según planificaciones definidas, fácil cambio de la señal que se está capturando en cada momento, almacenamiento de los ficheros de video capturados en un archivo, grabación de señales de audio o radio FM, visualización de los materiales audiovisuales almacenados, catalogación de los

materiales audiovisuales almacenados y trascripción de ficheros de audio. Este no obtuvo el resultado esperado puesto que no se logró integrar. Algunas de sus deficiencias fueron, el diseño de una base de datos estática que no posibilitaba añadir nuevas funcionalidades fuera de este marco. Las herramientas y tecnologías que se utilizaron eran libres y multiplataforma, entre ellas se encontraba el lenguaje de programación Java con el entorno de desarrollo Netbeans para el desarrollo de algunos de sus módulos. Se detectó que la utilización del framework Swing que trae Netbeans para el desarrollo de interfaces con algunos de los componentes de AWT es incompatible. Al utilizar los demás componentes del Swing y el Canvas de AWT la aplicación solo se mostraba por segundos y luego se cerraba. Se usaba Canvas porque el framework Swing no tenía un componente que permitiera pintar para ejecutar la reproducción de un material. La arquitectura que se utilizaba era en capas y orientada a objeto, sin embargo en esta no se tenía previsto la comunicación del sistema completo.

Hoy, el proyecto Sistema de Captura y Catalogación de Medias realiza un estudio sobre los diferentes flujos o procesos existentes en televisoras existentes en el mundo, las cuales son una de las principales fuentes de información. Estas televisoras, así como el ICRT (Instituto Cubano de Radio y Televisión) guardan sus archivos en varios formatos (15 mm, 30 mm, 1 pulgada, 2 pulgadas, U-Matic, Betamax, Betacan, M II, VHS y DVC_Pro). Estas entidades realizan préstamos de los materiales almacenados, de los cuales muchos regresan en mal estado o no regresan, perdiendo la información y los dispositivos de almacenamiento que tienen un alto costo en el mercado.

La mayoría de las grandes entidades que trabajan con medias se apoyan en un sistema para digitalizar la información almacenada, lo que posibilita ganar en organización y en búsquedas de los contenidos. Además los préstamos pueden ser controlados de forma eficiente y disminuir las pérdidas de los dispositivos de almacenamiento, porque los materiales quedarían almacenados en el servidor de medias. Es aquí donde la AS juega un papel importante, ya que en ésta se aportan todos los conocimientos, creatividad y experiencia para lograr satisfacer al cliente, buscándose el resultado esperado del sistema con la calidad requerida.

A partir del estudio realizado sobre las televisoras y empresas de este tipo, y basándose en las deficiencias encontradas en el proyecto SCCM realizado en el año 2009 se identificó el siguiente

problema a resolver: ¿Cómo contribuir a la gestión de Audio, Video y Streaming en el Sistema de Captura y Catalogación de Medias?

La investigación tiene como **objeto de estudio**: La arquitectura de software usada en el proceso de desarrollo de software, donde el **campo de acción** es: La arquitectura de software para el Sistema de Captura y Catalogación de Medias.

El **objetivo general** de este trabajo es diseñar una arquitectura de software para el Sistema de Captura y Catalogación de Medias enfocado a las entidades de televisión, que permita que el sistema sea modificable, interoperable, funcional y disponible, apoyándose en el uso de herramientas libres.

Como **idea a defender** se tiene que el diseño de una arquitectura de software robusta y confiable garantizará el desarrollo del Sistema de Captura y Catalogación de Medias.

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

1. Caracterizar diferentes sistemas que permitan la gestión de audio, video y streaming.
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. Caracterizar las herramientas y tecnologías a utilizar para el desarrollo.
4. Proponer la arquitectura para el Sistema de Captura y Catalogación de Medias.
5. Diseñar la estrategia de prueba para la arquitectura de software seleccionada.
6. Comparar al menos con 2 casos donde se hayan utilizado propuestas arquitectónicas en sistemas similares para la gestión de audio, video y Streaming.

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

Métodos Teóricos

- “Histórico – Lógico”: Para determinar las tendencias actuales, en este caso se empleó para realizar el estudio de la trayectoria real de determinados elementos que servirán de guía para la construcción de una buena arquitectura de software.

- “Analítico - Sintético”: Para conocer, reflexionar y aumentar los conocimientos acerca de la línea de investigación a partir de la consulta de la literatura científica correspondiente, y luego, haciendo uso de la síntesis confeccionar un cuerpo coherente en el cual se resumen los resultados obtenidos del análisis.

- “Modelación”: en el desarrollo de la investigación es necesaria la modelación de la arquitectura que se propone, es por eso que se utiliza este método para crear modelos con vistas que favorezcan la comprensión de la misma.

El documento está constituido por cuatro capítulos.

Capítulo 1.Estado del Arte y Fundamentación Teórica

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

Capítulo 2. Analizar las herramientas y tecnologías a utilizar para el desarrollo.

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

Capítulo 3. Propuesta de la Arquitectura del Sistema

En esta sección se reflejan la propuesta de la Arquitectura para el proyecto Sistema de Captura y Catalogación de Medias (SCCM) basada en el estudio realizado en los capítulos anteriores.

Capítulo 4.Evaluación de la Arquitectura

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

CAPÍTULO 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se aborda sobre las definiciones de Arquitectura de Software, estilos y patrones arquitectónicos. Para una mejor visión se definen las principales características que debe tener el sistema que se desea desarrollar. Una vez analizado los fundamentos de la arquitectura y las características del sistema, se realiza un estudio de los sistemas de digitalización y gestión de medias en el mundo. Después de esté análisis se determinan las principales ventajas y desventajas de estos sistemas que se tienen presente para el diseño de la arquitectura.

1.2 Arquitectura de Software

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.” **(Hilliard, 2000)**

“La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones”. **(Bass, 2003)**

La AS es clave en el desarrollo de los sistemas informáticos, constituye un basamento y guía para alcanzar el éxito. Es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación, aporta una visión abstracta de alto nivel para un mejor entendimiento de un sistema y poder definir los módulos principales así como también las responsabilidades que tendrán los mismos, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

A partir de estos conceptos se puede definir que la arquitectura de software aporta elementos que ayudan a una mejor toma de decisiones, establece la línea base para que los analistas, diseñadores, programadores y demás miembros trabajen en conjunto permitiendo cubrir todas las necesidades.

1.3 Estilos y Patrones Arquitectónicos

Los estilos y los patrones establecen un vocabulario común, y brindan soporte a los ingenieros para conseguir una solución que haya sido aplicada con éxito anteriormente, ante ciertas situaciones de diseño. Su aplicación en el diseño de la arquitectura del sistema es determinante para la satisfacción de los requerimientos de calidad.

1.3.1 Estilos Arquitectónicos

"Se define estilo arquitectónico como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción". **(Buchmann, 1996)**

Shaw y Garlan definen estilo arquitectónico como "una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores, y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes".

Los estilos son útiles para sintetizar estructuras de soluciones, también definen los patrones posibles de las aplicaciones y permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

Ejemplos de Estilos Arquitectónicos

- ✓ ***Estilos de Flujo de Datos***

  Tubería y filtros.

- ✓ ***Estilos Centrados en Datos***

  Arquitecturas de Pizarra o Repositorio.

- ✓ ***Estilos de Código Móvil***

  Arquitectura de Máquinas Virtuales.

- ✓ ***Estilos Peer-to-Peer***

  Arquitecturas Basadas en Eventos.

  Arquitecturas Orientadas a Servicios (SOA).

  Arquitecturas Basadas en Recursos.

- ✓ ***Estilos de Llamada y Retorno***

  Model-View-Controller (MVC).

  Arquitecturas en Capas.

  Arquitecturas Orientadas a Objetos.

  Arquitecturas Basadas en Componentes.

- ✓ ***Estilos Derivados***

  C2

  GenVoca

  Rest

- ✓ ***Estilos Heterogéneos***

  Sistema de Control de Procesos

  Arquitecturas Basadas en Atributos

1.3.2 Patrones

Christopher Alexander plantea que "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma".

"El concepto de patrón más ampliamente utilizado en el desarrollo del software es el patrón de diseño, que se utiliza particularmente en el paradigma orientado a objetos. En este contexto, un patrón de diseño es una descripción de las clases y de los objetos que trabajan conjuntamente para resolver un problema concreto". **(E. Gamma, 1998)**

Se considera que un patrón es un par problema – solución, resultado de la experiencia en el diseño de arquitecturas de sistemas y propone los patrones arquitectónicos como descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como éstos colaboran entre sí.

Un patrón arquitectónico se define por:

(1) un modelo de sistema que captura intuitivamente la forma en que están integrados los elementos.

(2) componentes.

(3) conectores que establecen las reglas de la interacción entre los componentes.

(4) una estructura de control que gobierna la ejecución.

Es muy importante saber la diferencia entre un estilo y un patrón. Los estilos se aplican en un nivel muy alto de abstracción, en el cual no interesa saber cuál es la estructura de los elementos que se están utilizando, en el caso de los patrones, se tiene en cuenta el significado de los elementos. A grandes rasgos se puede decir que los estilos son más genéricos que los patrones, por lo que pueden abarcar varios patrones, mientras los patrones van enfocados a una solución más específica.

Patrones Arquitectónicos más utilizados

- Arquitectura en capas

"Este patrón soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. El estilo admite muy naturalmente optimizaciones y refinamientos. Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas"**. (Kiccillof, 2004)**

Esta arquitectura sería de gran ventaja para el Sistema de Captura y Catalogación de Medias porque el desarrollo se puede llevar a cabo en varios niveles, donde es fácil separar los datos de la lógica del negocio. Si ocurre algún cambio, sólo se tendrían que realizar las correcciones necesarias en el nivel requerido sin tener que revisar código de otros niveles. Este patrón está compuesto principalmente por tres capas:

Capa de presentación: esta capa es la que ve el usuario, es conocida como interfaz gráfica, se comunica únicamente con la capa de negocio. Le presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso.

Capa de Negocio: aquí es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Es aquí donde se establecen las reglas a cumplirse.

Capa de Acceso a Datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

- Modelo vista controlador

"Se centra en la separación entre los datos o modelo, y la vista, mientras que el controlador es el encargado de relacionar a estos dos. Su principal característica es aislar la vista del modelo". **(Merlino, 2005)**

Modelo: Esta es la representación específica de la información con la cual el sistema opera, debe ser independiente. Las funciones y estructuras del modelo no deben ver a ninguna clase de los otros grupos.

Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Responde a eventos, usualmente acciones del usuario que puede invocar cambios en el modelo y probablemente en la vista. Actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página

Este es otro patrón que se debe tener en cuenta para la realización de este sistema, pues se puede reutilizar componentes fácilmente y se mantiene organizado y bien estructurado el proyecto. Además permite las siguientes ventajas:

- ✓ Generar componentes de las interfaces.
- ✓ Diferentes vistas simultaneas del mismo modelo.
- ✓ Aplicar fácilmente cambio de las interfaces.
- ✓ Permitir la sustitución de las interfaces de usuario.
- ✓ Crea independencia del funcionamiento
- ✓ Facilita el mantenimiento en caso de errores
- ✓ Permite el escalonamiento de la aplicación en caso de ser requerido.
    - ➢ Arquitectura Orientada a Servicios (SOA)

Es una arquitectura para conectar sistemas entre sí, permitiendo aplicar lógicas de control, negocio y procesos. Sirve para definir cómo se deben unir los distintos sistemas para conseguir que todo sea más eficiente, escalable, operable, mantenible, flexible y además se puedan realizar operaciones con los datos. Esta arquitectura aporta diferentes ventajas:

- ✓ Permite sustituir componentes individuales sin que eso afecte a otros componentes.
- ✓ Todos los sistemas se conectan al bus de la misma forma, con lo que se gana en homogeneidad.
- ✓ Facilidad en la operación y mantenimiento.
- ✓ Arquitectura sencilla, robusta y escalable.

SOA no es factible aplicarla en todas las empresas. Actualmente las empresas que la utilizan son porque disponen de una gran cantidad de información, que está distribuida entre muchos sistemas. Sus mayores ventajas surgen cuando existen varios sistemas aislados con funcionalidades independientes y estos se unen para obtener una información más compacta y provechosa. Se dice que para pequeñas empresas es

difícil que pueda aportar los mismos beneficios que para grandes organizaciones y es por eso que no se considera ideal para este sistema.

- Arquitectura multicapa orientada a objetos

”Una arquitectura multicapas que se adecue a los sistemas de información orientados a objetos, incluye la división de las responsabilidades que se encuentran en la arquitectura clásica de tres capas. Las responsabilidades se asignan a los objetos de software“. **(Larman, 1999)**

Los componentes de este estilo son los objetos, o más bien instancias de los tipos de dato abstractos. En la caracterización clásica de David Garlan y Mary Shaw, los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos. Los componentes del estilo se basan en principios Orientados a Objetos: encapsulamiento, herencia y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación.

1.4 Características del sistema

Para un mejor entendimiento del sistema se describen algunas de sus características. El mismo debe ser un software fiable y rentable, debe permitir:

- ✓ Carga inicial de la información audiovisual digitalizada al servidor central del Centro de Datos.
- ✓ Catalogación, según los datos necesarios, de los materiales digitalizados.
- ✓ Recuperación y visualización de los archivos existentes en el centro de datos a través de los flujos streaming y la descarga de los mismos según los permisos correspondientes.
- ✓ Gestión de los datos de usuarios y roles del sistema.
- ✓ Emisión de reportes de actividad de los usuarios en el sistema.

Para cumplir con todas estas características es necesario que:

- ✓ Las herramientas seleccionadas sean óptimas para cualquier plataforma permitiéndole al sistema que en futuras versiones sea multiplataforma.
- ✓ El personal de desarrollo esté bien dotado sobre las tecnologías a emplear para el desarrollo del sistema.
- ✓ El sistema gestor de base datos sea seguro.
- ✓ El tiempo de respuesta del sistema para cada una de sus funciones debe ser rápido.

1.5 Análisis de otras soluciones existentes

En el mundo actual existen otros sistemas de digitalización y gestión de medias con el objetivo de acercar la gestión de procesos de negocios a los activos multimedia, siendo de utilidad para cualquier organización que necesite archivar, buscar o usar grabaciones de video o audio. Estos productos se combinan para crear soluciones adaptadas a las necesidades precisas para una aplicación específica o un usuario concreto. Algunos de estos sistemas son:

1.5.1 Tarsys:

”Este software tiene Interfaz de cliente personalizable, cuenta con un modelo de datos adaptable a las necesidades del cliente. Ha sido implementado de acuerdo a estándares de sistemas abiertos: Extensible Markup Language (XML), Simple Object Access Protocol (SOAP). Realiza búsquedas avanzadas a través de un motor de búsqueda OracleTM. Genera informes a partir de metadatos, así como storyboard de manera automática, lo cual es muy útil para una fácil catalogación y posicionamiento en el código de tiempo. Utiliza patrones de catalogación basados en el estándar XML“. **(Tedial, 2009)**

Tarsys utiliza SOA en las aplicaciones facilitando su integración, así como el estándar CMIS (Content Management Interoperability Services) que facilita a las aplicaciones la lectura o la publicación de contenidos como parte del proceso de negocio, sin necesidad de conectores específicos a cada repositorio, además incorpora una implementación Java del estándar CMIS.

1.5.2 Tdficus:

"Optimiza los recursos y sistematiza todos los procesos de producción. Automatiza los movimientos de media entre subsistemas en la instalación. Asegura la disponibilidad de la media en cualquier puesto de trabajo en el formato apropiado. Minimiza la carga en la red mediante el uso de copia proxy, donde las imágenes de alta resolución no son obligatorias. Genera automáticamente órdenes de trabajo asociadas a cada paso del flujo de producción. Asigna automáticamente órdenes de trabajo a los recursos disponibles. Permite el seguimiento de las tareas realizadas por los operadores. Integra información sobre la llegada de nuevo material desde el sistema de tráfico. Comprueba la localización de la media en dispositivos externos como video servidores. Cataloga y almacena la media en sistemas de archivo. Garantiza la realización de los procesos productivos de la compañía conforme a sus normas de calidad". **(Tedial, 2009)**

1.5.3 Videoma:

"Permite la grabación, monitorización y visualización de múltiples canales. Se pueden realizar búsquedas por keyword (palabra clave), importación y exportación de información por XML. Permite la conexión con otras aplicaciones de data mining a través del Módulo Api de Videoma basado en Web Service. Realiza la monitorización de emisiones de definición estándar (SD) y alta definición (HD), catalogación personalizada por medio de campos de metadata. Videoma Broadcast Monitor incorpora la grabación y monitorización multicanal de la señal de Televisión Digital Terrestres (TDT), satelital, multicast y frecuencias analógicas. Además de ello realiza el streaming, catalogación, búsqueda, visionado y almacenamiento del contenido en tiempo real. Este sistema permite la inserción de contenido digital automatizado y sin necesidad de supervisión mediante directorios de escucha o watchfolders, además de la conversión automática de formatos en alta calidad a calidad de visionado web. Realiza la documentación del material mediante plantillas personalizables por el usuario, de las que derivan los campos de documentación. Los parámetros de búsqueda son configurables por el usuario. Permite guardar y crear clips o vídeos en disco, así como la fragmentación del vídeo en segmentos". **(Pozuelo, 2006)**

1.5.4 Codistream:

"Es un software de codificación y gestión de video para su distribución en streaming a través de Internet con calidad de alta definición donde es posible conseguir la máxima calidad de imagen a menores tasas

de bits. La versión H264 podrá codificar cualquier archivo de video y publicarlo en un sitio web de manera totalmente autónoma. La Interfaz de usuario es de última generación, compatible con todos los navegadores. Posee un medidor de ancho de banda incorporado además de dos modos de visualización de videos: videos on demand y directos. Se puede crear una miniatura del video a codificar en el punto que se desee. Se pueden editar los datos del video como título, descripción o link relacionado. Su principal desventaja es que solo es compatible con los Sistemas Operativos: Windows XP y Windows Vista". **(Multistream S.L, 2005)**

El API o interfaz de comunicación de Codistream permite la comunicación a través del protocolo SOAP 1.1, SOAP 1.2 y a través de conexiones HTTP GET y HTTP POST. El estándar de comunicación SOAP posibilita la comunicación de procesos a través del intercambio de datos XML. Las soluciones desarrolladas con este protocolo permiten una arquitectura en capas que provee de robustez, usabilidad e interoperabilidad a los mensajes.

Los sistemas analizados tienen como desventajas que son propietarias y por su gran utilidad resultan tener un alto valor en el mercado. Además no son eficientes para cualquier sistema operativo. Por otra parte utilizan potentes gestores de base datos en su mayoría propietarios, aspecto que permite identificar que se necesita un gestor que sea robusto y a su vez libre. También proporcionan la máxima calidad de transmisión de gráficos y video, al mismo tiempo reduce el uso del ancho de banda de red. Se realizan varios tipos de búsquedas, básica, y avanzada. Para el sistema a realizar sería una buena opción varios tipos de búsquedas pues permitirá mejores resultados. Estos sistemas están formados por decenas de servidores repartidos en diferentes puntos de acceso a la red con la capacidad de agregar servicio de cache/streaming para los usuarios finales. Tarsys utiliza SOA por ser una empresa grande, facilitando la integración de las aplicaciones, los demás no describen nada relacionado al respecto, al menos en bibliografía seria. Esta característica no se tiene en cuenta porque SOA presenta sus mayores ventajas para grandes compañías, no así para pequeñas empresas que lo que puede tributar es pérdida. Estos sistemas emplean balanceos de carga para que sus respuestas sean más rápidas, esta es otra característica a tener en cuenta para el software que se pretende realizar.

1.6 Conclusiones del capítulo

A partir de las características que el sistema debe brindar y el estudio realizado sobre los estilos y patrones arquitectónicos más comunes, se propone utilizar MVC, arquitectura en capas y arquitectura orientada a objetos. Después de analizar los sistemas de similar propósito en el mundo, se concluye que todos basan su desarrollo sobre herramientas propietarias y tratan de garantizar un buen rendimiento en sus aplicaciones porque el trabajo con materiales audiovisuales suele ser tedioso. No se propone arquitectura SOA porque el proyecto Sistema de Captura y Catalogación de Medias necesita de tiempos de respuestas lo más rápido posible y sin embargo la velocidad de intercambio de información entre sistemas que utilizan SOA es más lenta que una conexión directa.

CAPÍTULO 2: Herramientas y Tecnologías a utilizar para el desarrollo.

2.1 Introducción

En este capítulo se realiza un estudio sobre las diferentes herramientas que se van a utilizar para el desarrollo del sistema, como son la metodología de desarrollo, los lenguajes de programación, el gestor de bases de datos. También se explican las tecnologías más importantes a tener en cuenta para un buen diseño de la aplicación.

2.2 Metodología de Desarrollo de Software

Se entiende por metodología de desarrollo de software una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software con la finalidad de garantizar la eficacia y la eficiencia en el proceso de generación de software. Los riesgos a afrontar y los controles a establecer varían en función de las diferentes etapas del ciclo de vida de desarrollo que defina cada metodología en particular. A continuación se expondrán dos metodologías de desarrollo: Rational Unified Process y Extreme Programming.

2.2.1 Rational Unified Process (RUP)

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Esta metodología divide en 4 fases el desarrollo, estas fases son: Inicio, Elaboración, Construcción y Transición, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

Principales características

- ✓ Tiene una forma disciplinada de asignar tareas y responsabilidades, decide quién hace qué, cuándo y cómo.
- ✓ Guiado por los Casos de Uso.
- ✓ Desarrollo iterativo e incremental.
- ✓ Administración de requisitos.
- ✓ Uso de arquitectura basada en componentes.
- ✓ Modelado visual del software.

- ✓ Centrado en la Arquitectura.
- ✓ Verificación de la calidad del software.

No existen dos proyectos de desarrollo de software que sean iguales. Cada uno tiene prioridades, requerimientos, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar un software de calidad superior a tiempo.

Esta metodología tiene varios beneficios:

- ✓ ”Son las mejores prácticas de desarrollo adoptadas en cientos proyectos mundialmente y enseñadas como parte del currículo en cientos de universidades, la metodología RUP se convirtió rápidamente en el estándar de facto para el proceso de desarrollo en la industria de software.
- ✓ Diferente que otras metodologías comerciales, la plataforma RUP hace que el proceso sea práctico con bases de conocimiento y guías para ayudar en el despegue de la planificación del proyecto, integrar rápidamente a los miembros del equipo y poner en acción el proceso personalizado.
- ✓ Solo la plataforma RUP proporciona un framework de proceso configurable que permite seleccionar e implantar los componentes específicos de proceso necesarios para proporcionar un proceso consistente y customizado para cada equipo y proyecto“. **(Grupo Soluciones GSInnova, 2007)**

2.2.2 Extreme Programming (XP)

”Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Las iteraciones son relativamente cortas ya que se piensa que entre más rápido se le entreguen desarrollos al cliente, más retroalimentación se va a obtener y esto va a representar una mejor calidad del producto a largo plazo.

Proyecto XP

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

- ✓ El cliente define el valor de negocio a implementar.
- ✓ El programador estima el esfuerzo necesario para su implementación.
- ✓ El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
- ✓ El programador construye ese valor de negocio.
- ✓ Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Gran parte del éxito del proyecto XP se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada". **(Villegas, 2009)**

Después de haber realizado un análisis la metodología seleccionada es RUP, ya que permite seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. Cada miembro del equipo tiene acceso instantáneo a la base de conocimiento y guía de procesos del RUP desde su escritorio, lo que proporciona una amplia comunicación entre los miembros del equipo de desarrollo. La base de conocimiento unifica aún más al equipo identificando y asignando responsabilidades, artefactos y tareas de forma que cada miembro del equipo comprenda su contribución al proyecto. Unificando al equipo, se simplifica la comunicación, asegurando la asignación de recursos en forma eficiente, la entrega de los artefactos correctos, y el cumplimiento de los tiempos límite. Por la gran documentación que genera, permite que si algún miembro del equipo se retira, el proyecto no sufra grandes consecuencias porque todo queda documentado.

2.3 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (OO). Un artefacto es una información que es utilizada o producida mediante

un proceso de desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

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

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

”UML es un lenguaje gráfico de modelamiento que usa conceptos de orientación por objetos. Este lenguaje tiene una sintaxis y una semántica bien definida, sirviendo además para todas las etapas de desarrollo. En UML se utilizan para el modelamiento de un sistema, diferentes elementos y relaciones“. **(Javier, 2011)**

Estos elementos se agrupan en diagramas preestablecidos que corresponden a diferentes proyecciones del sistema. La gran ventaja de UML es que se ha venido adoptando en diferentes medios empresariales y académicos como el lenguaje “estándar” para el análisis y diseño de los sistemas de software.

El modelado no es más que la construcción de un modelo a partir de una especificación**.** De esta manera es posible enseñarle al cliente una posible aproximación de lo que será el producto final. Entre las características fundamentales por la que se escoge es porque toma un perfil orientado a objetos en el modelado de aplicaciones. Las extensiones de UML para el modelado de negocio aportan elementos muy importantes ya que proporcionan otras vistas de la arquitectura de negocio que son más difíciles de observar.

2.4 Herramientas CASE (Computer-Aided Software Engineering)

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software (Investigación Preliminar, Análisis, Diseño, Implementación e Instalación). CASE es también definido como el Conjunto de métodos, utilidades y técnicas que facilitan el

mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

2.4.1Visual Paradigm

Esta herramienta está desarrollada por Visual Paradigm Internacional, una de las principales compañías de herramientas CASE. Su mayor éxito consiste en la capacidad de ejecutarse sobre diferentes sistemas operativos lo que le confiere la característica de ser multiplataforma. Es una herramienta profesional para UML que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Entre sus características se destacan que es robusto y portable. Genera código y realiza ingeniería inversa para diez lenguajes de programación. Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue. Además exporta e importa los diagramas con estándar XML y como imágenes.

2.4.2 Rational Rose

"Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software". **(Jelsoft Enterprises Ltd, 2011)**

Rose no se integra con varios Entornos de Desarrollo Integrado(IDE) como Visual Paradigm, solo lo hace con Borland JBuilder de la versión 7.0 en lo adelante y Microsoft Visual Studio de la versión 2003 en adelante. Además no es multiplataforma y es aconsejable utilizar Windows 2000, Windows NT y Windows XP.

Para el desarrollo del sistema se seleccionó la herramienta Visual Paradigm ya que es muy fácil de usar y muestra un ambiente gráfico agradable para el usuario. Es la herramienta por excelencia para ser utilizada en un ambiente de software libre, permite el modelado de base de datos, el modelado de requerimientos, el modelado del proceso de negocio, la interoperabilidad, la generación de documentación y la generación de código base para diferentes lenguajes de programación como Java, C# y PHP. Igualmente permite la integración con varios IDE.

2.5 Lenguajes de Programación

Un lenguaje de programación es un lenguaje artificial usado para controlar el comportamiento de una máquina, especialmente una computadora. Se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas por máquinas. Existen muchos lenguajes de programación, para desarrollar aplicaciones tanto de escritorio como web, sin embargo para este trabajo de diploma se tomaron en consideración algunos de ellos: C++, PHP.

2.5.1 C Sharp (C#)

"C# nace de Microsoft con la idea de crear un lenguaje mejorado en todos los aspectos. Concebido como lenguaje nativo de su famosa plataforma .Net para aplicaciones web y de escritorio, se ha dicho que C# conjunta principalmente tanto aspectos de C++ como de Java y Visual Basic, pero de una forma más versátil y mejorada agregándole cada vez más elementos que faciliten su uso.

El ambiente de trabajo es muy cómodo ya que tiene un ambiente amigable y clásico de las aplicaciones de Windows. En cuanto a la forma de programar, es fácil de usar para quien está familiarizado con C++, ya que su estructuración básica es muy similar, sin embargo C# ahorra muchos pasos "tediosos" de otros lenguajes como la creación de funciones complejas desde cero y declaración de variables globales". **(Martínez, 2008)**

Este lenguaje de programación no se tiene en cuenta para la realización de la aplicación debido a que sus mejores prestaciones van atada al framework Visual Studio .NET, lo que implica tener algunos requerimientos mínimos del sistema para poder trabajar adecuadamente tales como contar con el Sistema Operativo Windows, tener alrededor de 4 gigas de espacio libre para la instalación.

2.5.2 Java

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

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el Garbage Collector (reciclador de memoria dinámica). El reciclador se encarga de liberar memoria y como es un thread (hilo) de baja

prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria“. **(Daum, 2005)**

2.5.3C++

Aunque en un principio C++ se plantea como una mejora de C, en la actualidad es un lenguaje independiente, versátil, potente y general. Mantiene las ventajas de C en cuanto a riquezas de operadores y expresiones, flexibilidad, concisión y eficiencia. Además ha eliminado algunas de las dificultades y limitaciones del C original. Este es a la vez un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

Principales características.

- ✓ “Programación orientada a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además permite la reutilización del código de una manera más lógica y productiva.
- ✓ Portabilidad: Un código escrito en C++ puede ser compilado en diferentes ordenadores y sistemas operativos sin hacer apenas cambios.
- ✓ Brevedad: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguaje es preferible el uso de caracteres especiales que las "palabras clave".
- ✓ Programación modular: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Esta característica permite unir código en C++ con código producido en otros lenguajes.
- ✓ Velocidad: El código resultante de una compilación en C++ es muy eficiente, por su capacidad de actuar como lenguaje de alto y bajo nivel”. **(Allison, 2000)**

C++ está considerado como un lenguaje potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos traen, obliga a hacerlo casi todo manualmente al igual que C, lo que "dificulta" mucho su aprendizaje.

Dada la eficiencia de este leguaje se decidió que debía ser usado para el desarrollo de algoritmos complejos, por ejemplo el trabajo que se debe realizar con las imágenes, la extracción de los fotogramas,

la codificación de los archivos multimedia, donde todos estos procedimientos serán desarrollados en un entorno de escritorio.

2.5.4 ASP.NET

"ASP.NET se ha mejorado para proporcionar compatibilidad instantánea para las situaciones más comunes de las aplicaciones Web. Con este lenguaje se puede poner en marcha los sitios y las páginas Web más fácilmente y con menos código que antes. Al mismo tiempo, puede agregar características personalizadas a ASP.NET para que cumplan sus propios requisitos ". **(Microsoft, 2011)**

Principales características:

- ✓ "ASP es totalmente gratuito para Microsoft Windows NT o Windows 95/98.
- ✓ El código ASP se puede mezclar con el código HTML en la misma página (no es necesario compilarlo por separado).
- ✓ Cómo el código ASP se ejecuta en el servidor, y produce como salida código HTML puro, su resultado es entendible por todos los navegadores existentes.
- ✓ ASP permite usar componentes escritos en otros lenguajes (C++, Visual Basic, Delphi), que se pueden llamar desde los guiones ASP.
- ✓ Sin modificar la instalación, los guiones ASP se pueden programar en JScript o VBScript (este último es el más usado porque más programadores lo dominan), pero también existen otros lenguajes, como Perlscript y Rexx, que se pueden emplear para programar ASP". **(Mora, 2009)**

Para utilizar ASP en Linux es necesaria la plataforma Mono, la cual no es más que una implementación libre de las herramientas y librerías .NET creadas por Window. Los mejores resultados de este lenguaje de programación están desarrollados sobre .NET y específicamente con el IDE Visual Studio.

2.5.5PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica.

Principales características

- ✓ Es un lenguaje multiplataforma.
- ✓ Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- ✓ El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- ✓ Permite aplicar técnicas de programación orientada a objetos.

”PHP posee soporte para la gran mayoría de las plataformas existentes en la actualidad. Es eficiente para el desarrollo de aplicaciones Web y logra integrarse con un gran número de servidores, incluso propietarios. Su sintaxis clara y bien definida es lo que permite que sea sencillo de aprender y utilizar“. **(Mehdi Achour, 2001)**

Se utiliza PHP para las aplicaciones web dado que es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Es una herramienta que permite que sea sencillo de aprender y utilizar gracias a su sintaxis clara y bien definida. Presenta una sencilla integración con múltiples bases de datos, esencial para una página web verdaderamente dinámica. Entre sus principales ventajas está el alto rendimiento que presenta, además de interfaces para una gran cantidad de sistemas de base de datos, diferentes bibliotecas incorporadas para muchas tareas web habituales, bajo coste facilidad de aprendizaje y uso, portabilidad y acceso a código abierto. Actualmente se usa por su versatilidad de uso con diferentes sistemas operativos. Produce sensación al usuario de mayor rapidez y mayor usabilidad ya que es poco pesado.

2.5.6 HTML

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

”Es un lenguaje sencillo, permite definir documentos de hipertexto a base de ciertas etiquetas que marcan partes del documento dándoles una estructura o jerarquía. Presenta el texto de una manera estructurada y agradable, con enlaces (hyperlinks) que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia (gráficos, sonido, video). El lugar donde se encuentra esta información puede ser el mismo documento o cualquier otro lugar de Internet“. **(virtualnauta.com, 2007)**

En un documento HTML se pueden ver tres partes bien diferenciadas:

- ✓ Una cabecera del tipo de documento, en ella se especifica el tipo de documento HTML.
- ✓ Una cabecera de documento, donde se especifica información del documento.
- ✓ El cuerpo del documento, donde se coloca toda la información que contiene la página Web.

2.5.7 JavaScript

Es un lenguaje de tipo script compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de internet, es un lenguaje de programación interpretado, como ventaja clave de este lenguaje se puede mencionar que es interpretado por todos los navegadores modernos, debido a que este lenguaje está provisto de una implementación del (*Document Object Model* o Modelo de Objetos del Documento) DOM estándar diseñado por W3C que incorporan Konqueror, Mozilla Firefox desde su primera versión, Internet Explorer desde su versión 6.0, Netscape Navigator, Opera desde la versión 7 entre otros. El navegador tiene la responsabilidad de interpretar las sentencias. Se considera un lenguaje orientado a objetos y a eventos del usuario. Con su uso se pueden desarrollar scripts que den respuesta a dichos eventos. Al ejecutarse en el lado cliente, evita la sobrecarga del servidor por peticiones que pueden tener respuestas locales.

El Sistema Captura y Catalogación de Medias será beneficiado por la característica mencionada anteriormente, siendo de vital importancia para su correcto funcionamiento. No se debe olvidar la vinculación existente entre el lenguaje JavaScript y el conjunto de tecnologías AJAX por lo que prescindir de su utilización en la realización de este trabajo no es una opción.

2.5.8 AJAX

Basado en los estándares abiertos Ajax está formado por las tecnologías JavaScript, HTML, XML, CSS, y XML HTTP Request Object, siendo este último el único que “no es” estandar pero es soportado por los navegadores más utilizados de internet como son los basados en mozilla, internet explorer, safari y opera.

“Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio“. **(Pérez, 2008)**

AJAX funciona en cualquier navegador, es compatible con cualquier del tipo de tecnología de servidor que se utilice. Es compatible con diversos lenguajes de programación web como PHP, ASP.NET.

2.6 Framework

Un framework ofrece una guía arquitectónica partiendo el diseño en clases abstractas y definiendo sus responsabilidades y sus colaboraciones. Un desarrollador personaliza el framework para una aplicación particular mediante herencia y composición de instancias de las clases del framework.

En el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

2.6.1 Symfony

’’Symfony es un framework PHP que facilita el desarrollo de las aplicaciones web. Se encarga de todos los aspectos comunes y aburridos de las aplicaciones web, permitiendo que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto.

Symfony es además el framework más documentado del mundo, ya que cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales.

Principales Características

- ✓ Fácil de instalar y configurar en sistemas Windows, Mac y Linux
- ✓ Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server)
- ✓ Compatible solamente con PHP 5 desde hace años, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP.
- ✓ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional
- ✓ Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización
- ✓ Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins
- ✓ Publicado bajo licencia MIT de software libre y apoyado por una empresa comprometida con su desarrollo.

Symfony es infinitamente escalable, además sigue una política de tipo LTS (*long term support*), por la que las versiones estables se mantienen durante 3 años sin cambios pero con una continua corrección de errores**". (Eguiluz, 2011)**

2.6.2 Dojo

"La fundación Dojo es una asociación sin ánimo de lucro fundada para ayudar a proyectos de código abierto. Su principal objetivo es la ayuda para las compañías en su adopción y animar el uso de los proyectos con los que colabora. Los complementos de Dojo son componentes pre empaquetados de código JavaScript, HTML y CSS que pueden ser usados para enriquecer aplicaciones web.

Dojo provee de una capa de abstracción (dojo.io.bind) para varios navegadores web con la que se pueden usar otros transportes (como IFrames ocultos) y diferentes formatos de datos. De esta forma se pueden obtener los campos que se van a enviar como parámetros del formulario de una manera sencilla". **(Peña, 2008)**

También es posible inicializar paquetes adicionales dentro o al mismo nivel que el paquete dojo, permitiendo extensiones o bibliotecas de terceros. Los paquetes de Dojo pueden contener múltiples archivos. Cualquier paquete o archivo puede depender de otro. En este caso, cuando el paquete es cargado, cualquier dependencia será también cargada. En el proyecto Sistema de Captura y Catalogación de Medias se utilizaran los frameworks Symfony para las aplicaciones Web y Dojo para la interfaz de usuario.

2.6.3 QT Framework

"Qt es un framework para el desarrollo de aplicaciones multiplataforma creado por la compañía Trolltech y que actualmente es propiedad de Nokia, la función más conocida de Qt es la de la creación de interfaces de usuario, sin embargo no se limita a esto, ya que también provee varias clases para facilitar ciertas tareas de programación como el manejo de sockets, soporte para programación multihilo, comunicación con bases de datos, manejo de cadenas de caracteres, entre otras.

Qt utiliza C++ de manera nativa, pero ofrece soporte para otros lenguajes como Python mediante PyQt, Java mediante QtJambi, o C# mediante Qyoto. Qt es un framework muy poderoso, comparable con Swing de Java o .NET de Microsoft, además ofrece una suite de aplicaciones para facilitar y agilizar las tareas de desarrollo.

Principales Características

- ✓ Compatibilidad multiplataforma con un sólo código fuente.
- ✓ Rendimiento en C++.
- ✓ Disponibilidad del código fuente.
- ✓ Excelente documentación.
- ✓ Fácilmente internacionalizable.
- ✓ Arquitectura lista para plugins.

Qt también provee poderosas herramientas de desarrollo, entre ellas destaca un completo entorno de desarrollo, llamado Qt Creator, que incluye un editor de texto con autocompletado, diseñador de interfaces gráficas, gestión de proyectos, sistema de depuración, integración con sistemas de control de versiones y muchas más características**". (López, 2011)**

En un principio, Qt sólo ofrecía bibliotecas de código para la creación de interfaces gráficas de usuario. Ahora existen bibliotecas para muchas cosas más, como: Bases de datos, XML, multimedia, comunicación en red, OpenGL. Este framework se utiliza para la parte de aplicaciones de escritorio.

A continuación se hace referencia a algunos elementos de esta biblioteca QT:

- ✓ **Klear**: Visualizador y grabador de TV digital.
- ✓ **JuK:** Reproductor de audio.
- ✓ **KAudioCreator**: Extractor digital de discos compactos.
- ✓ **KMPlayer**: Frontispicio para GStreamer, MPlayer o Xine.
- ✓ **Konverter**: Conversor de video, GUI para MEncoder.
- ✓ **KRadio**: Sintonizador de radio.
- ✓ **KsCD**: Reproductor de discos compactos.

2.7 Entornos de desarrollo integrados (IDE)

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

2.7.1 NetBeans

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

El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Les sirve a los programadores para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Además es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

”La versión de NetBeans IDE 6.9 introduce JavaFX Composer, una herramienta de diseño para la creación de aplicaciones gráficas JavaFX, parecido al constructor de aplicaciones gráficas Swing para aplicaciones Java SE. Otras notoriedades incluyen la interoperabilidad OSGi para aplicaciones de plataforma NetBeans, y la compatibilidad para desarrollar paquetes OSGi con Maven; compatibilidad para el SDK de JavaFX 1.3.1, Framework Zend PHP, y RoR (Ruby on Rails) 3.0; así como mejoras en el editor Java, Depurador Java y seguimiento de incidencias“. **(Oracle Corporation, 2011)**

Este IDE va a ser usado en el proyecto Sistema de captura y Catalogación de Medias en la integración con el lenguaje de programación php y los frameworks Symfony y Dojo.

2.7.2 Qt Creator

”QT Creator es un entorno de desarrollo multiplataforma muy completo. Está totalmente integrado con el Qt Designer para ayudarle a diseñar formas de la interfaz de usuario como lo haría con la versión independiente. La integración de Qt Designer incluye también la gestión y finalización del proyecto de código. Qt Creator abastece no solo a los desarrolladores que están acostumbrados a utilizar el ratón, sino también a los desarrolladores que se sienten más cómodos con el teclado. Una amplia gama de métodos abreviados de teclado y navegación están disponibles para ayudar a acelerar el proceso de desarrollo de su aplicación”. **(Wallace, 2010)**

Qt Creator utiliza la biblioteca de software Qt para crear interfaces gráficas de usuario. Aplicaciones como: Google Earth, Skype, Adobe Photoshop Album y VirtualBox entre muchas otras hacen uso de ésta.

Como características tiene el reconocimiento de métodos, la facilidad de creación de formularios y amplia documentación On-line, así como un sin fin de opciones que facilitan el desarrollo de cualquier aplicación. Por estas funcionalidades se va a usar QT Creator para el desarrollo del sistema en las aplicaciones de escritorio.

2.8 Librería ICE

”Para la comunicación de las aplicaciones se emplea Internet Communication Engine (ICE) que es un middleware orientado a objetos. Proporciona herramientas, APIs, y soporte de bibliotecas para construir

aplicaciones cliente-servidor orientadas a objetos. Una aplicación ICE se puede usar en entornos heterogéneos: los clientes y los servidores pueden escribirse en diferentes lenguajes de programación, pueden ejecutarse en distintos sistemas operativos y en distintas arquitecturas, y pueden comunicarse empleando diferentes tecnologías de red. Además, el código fuente de estas aplicaciones puede portarse de manera independiente al entorno de desarrollo". **(Fernandez, 2006)**

Este middleware es usado para comunicar las aplicaciones desarrolladas en C++, y estas a su vez con las que están desarrolladas en web.

2.9 Sistema Gestor de Base de Datos (SGBD)

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. Presenta una interfaz mediante la cual el usuario puede comunicarse con el sistema físico y realizar operaciones como almacenar o recuperar datos de ella. Las principales funciones que debe cumplir un SGBD son la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias.

2.9.1 Oracle

"Oracle es un sistema de gestión de base de datos relacional (*Relational Data Base Management System*), fabricado por Oracle Corporation. Se considera uno de los sistemas de bases de datos más completos. Es un sistema gestor de base de datos robusto, tiene muchas características que garantizan la seguridad e integridad de los datos. Las transacciones se ejecuten de forma correcta, sin causar inconsistencias. Ayuda a administrar y almacenar grandes volúmenes de datos. Presenta estabilidad, escalabilidad además de ser multiplataforma". **(Guiarte Multimedia S.L. , 2002)**

Aunque su dominio en el mercado de servidores empresariales ha sido casi total, recientemente sufrió la competencia de gestores de bases de datos comerciales y de la oferta de otros con licencia Software Libre como PostgreSQL.

2.9.2 PostgreSQL

"PostgreSQL es un sistema de gestión de bases de datos objeto-relacional y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Sus características técnicas lo hace una de las bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración *e* implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema". **(Martinez, 2011)**

Ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones. Además aportan potencia y flexibilidad adicional como son las restricciones (constraints), disparadores (triggers), reglas (rules) e integridad transaccional.

2.9.3 SQLite

"SQLite es un sistema de gestión de bases de datos relacional para bases de datos locales. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), es guardado como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción". **(Harreman, 2007)**

Este gestor de base de datos se utiliza solo en el núcleo de la aplicación para garantizar que la comunicación sea más rápida ya que este se encarga de la mayoría de las peticiones.

2.9.4 Pgpool-II

Pgpool-II es un middleware que funciona entre servidores PostgreSQL y un cliente de base de datos PostgreSQL. Ofrece varias funcionalidades que permiten mejorar el rendimiento y la seguridad en cualquier sistema. Con esta herramienta se puede configurar para crear un failover clúster donde la idea es que si uno de los nodos cae, otro empieza a dar servicio (proceso conocido como failover) sin necesidad de intervención humana. Entre sus principales características se encuentran:

- ✓ “Limita el excedente de conexiones. PostgreSQL soporta un cierto número de conexiones concurrentes y rechaza las que superen dicha cifra. Aumentar el límite máximo de conexiones incrementa el consumo de recursos y afecta al rendimiento del sistema. Pgpool-II tiene también un límite máximo de conexiones, pero las conexiones extras se mantienen en una cola en lugar de devolver un error inmediatamente.

- ✓ Pool de conexiones. Pgpool-II mantiene abiertas las conexiones a los servidores PostgreSQL y las reutiliza siempre que se solicita una nueva conexión con las mismas propiedades (nombre de usuario, base de datos y versión del protocolo). Ello reduce la sobrecarga en las conexiones y mejora la productividad global del sistema.

- ✓ Replicación. Pgpool-II puede gestionar múltiples servidores PostgreSQL. El uso de la función de replicación permite crear una copia en dos o más discos físicos, de modo que el servicio puede continuar sin parar los servidores en caso de fallo en algún disco.

- ✓ Balanceo de carga. Si se replica una base de datos, la ejecución de una consulta SELECT en cualquiera de los servidores devolverá el mismo resultado. Pgpool-II se aprovecha de la característica de replicación para reducir la carga en cada uno de los servidores PostgreSQL distribuyendo las consultas SELECT entre los múltiples servidores, mejorando así la productividad global del sistema. En el mejor caso, el rendimiento mejora proporcionalmente al número de servidores PostgreSQL. El balanceo de carga funciona mejor en la situación en la que hay muchos usuarios ejecutando muchas consultas al mismo tiempo.

- ✓ Paralelización de consultas. Al usar la función de paralelización de consultas, los datos pueden dividirse entre varios servidores, de modo que la consulta puede ejecutarse en todos los servidores de manera concurrente para reducir el tiempo total de ejecución. La paralelización de consultas es una solución adecuada para búsquedas de datos a gran escala”. **(Jaume Sabater, 2008)**

Para el SCCM se usarán preferiblemente la replicación, el balanceo de carga y la paralelización de consultas. Por otra parte la configuración del Pgpool-II suele ser un poco tediosa por lo que se usará PgpoolAdmin que no es más que una página web que permite realizar la configuración del middleware de una forma más sencilla.

2.10 Software para el Control de Versiones

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

2.10.1 Subversion

”Es un software de sistema de control de versiones de código abierto y gratuito. Soporta el manejo de ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto que recuerda todos los cambios hechos a sus ficheros y directorios, permitiendo recuperar versiones antiguas de datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Existen varias interfaces de subversion, ya sea programas individuales como interfaces que lo integran en entornos de desarrollo”. **(Pilato, 2004)**.

El control de versiones es usado con el IDE Netbeans y con Qt Creator para tener un repositorio de código, el cual le permite a los desarrolladores una mejor comunicación e integración.

2.11 Conclusiones del Capítulo

En este capítulo después de un estudio realizado quedaron definidas las principales herramientas tanto de modelado como de desarrollo haciendo énfasis en sus ventajas y desventajas para alcanzar una arquitectura, que garantice luego un software con la calidad, el tiempo y el costo requerido. Para garantizar un mejorar rendimiento, se definió C++ en el desarrollo de algoritmos complejos y pgpool para el balanceo de carga y consultas paralelas, lo que permite acelerar los tiempos de respuesta. Teniendo en cuenta que el sistema puede tener tanto aplicaciones web como de escritorio se seleccionaron las herramientas que fueran libres, fáciles de emplear y compatibles en varias plataformas para que en futuras versiones pueda ser multiplataforma. Además se garantiza la comunicación con el middleware ICE entre las aplicaciones de C++ y PHP.

CAPÍTULO 3: Propuesta de la Arquitectura del Sistema.

3.1 Introducción

En este capítulo se desarrolla la propuesta arquitectónica para dar solución al problema dado a conocer durante el diseño teórico de la investigación. Para alcanzar dicho objetivo se toma como punto de partida las descripciones de herramientas y tecnologías seleccionadas y descritas en el epígrafe anterior de este documento. Se obtiene de esta manera una mejor visión del sistema, ya que se especifican aspectos arquitectónicamente significativos, dentro de ellos se encuentran los requisitos no funcionales, los cuales son fundamental para el desarrollo de la arquitectura y otros artefactos de la metodología de desarrollo seleccionada RUP, como son las 4+1 vistas de la arquitectura.

3.2 Herramientas de desarrollo

A continuación se mencionan las herramientas tanto de modelación como de desarrollo que se definieron en el proyecto SCCM.

Herramientas de Modelado

Lenguaje de modelado: UML

Herramienta CASE: Visual Paradigm Enterprise Edition

Herramientas de Desarrollo

Lenguajes de programación: PHP, C++, Java Script, Ajax y HTML.

Entornos de desarrollo: Netbeans y Qt Creator.

Framework: Dojo, Symfony y Qt.

Gestor de Bases de Datos: PostgreSQL y SQlite.

Tecnología para replicación y balanceo de carga: Pgpool-ll.

Middleware para la comunicación: ICE.

Control de versiones: Subversion

3.3 Estructura del equipo de desarrollo

Se realizó una estructura de tipo formal para organizar el equipo de desarrollo, y de esta manera plasmar los diferentes roles asignados entre el equipo, la distribución de sus puestos de trabajo, la interacción entre ellos, así como la distribución de tareas y responsabilidades.

3.3.1 Distribución del equipo de trabajo

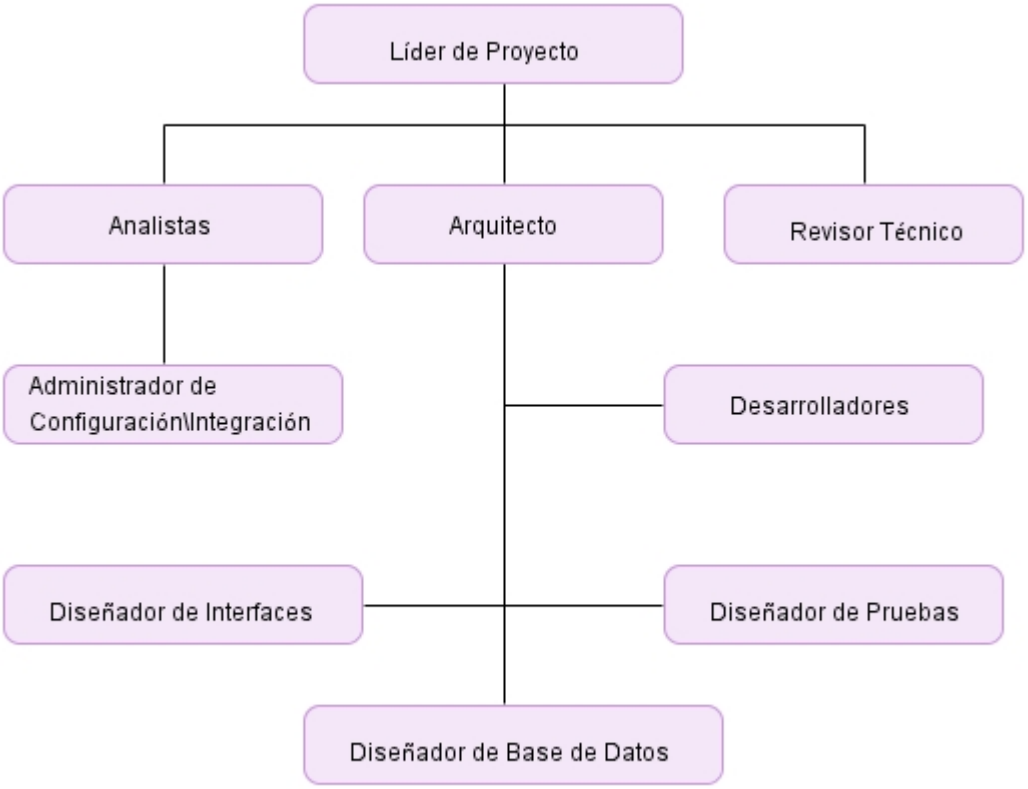

Figura 1 Distribución del Equipo de Trabajo.

3.3.2 Configuración de los Puestos de Trabajo por Roles

Líder del Proyecto

1. PC, con mouse y teclado.
2. Instalación de Visual Paradigm.
3. Instalación del paquete Office.
4. Instalación del pgadmin

Analista

1. PC, con mouse y teclado.
2. Instalación de Visual Paradigm.
3. Instalación del paquete Office.

Arquitecto de software

1. PC, con mouse y teclado

2. Instalación del Visual Paradigm.
3. Instalación del paquete Office.
4. Instalación del pgadmin

Diseñador de interfaces

1. PC, con mouse y teclado.
2. Instalación del Visual Paradigm.
3. Instalación del paquete Office.

Implementador

1. PC, con mouse y teclado
2. Instalación del IDE QT Creator o Netbeans con sus framework.
3. Instalación del paquete Microsoft Office.
4. Instalación del pgadmin.

Revisor Técnico

1. PC, con mouse y teclado.
2. Instalación de Visual Paradigm.
3. Instalación del paquete Office.

Diseñador de Base de Datos

1. PC, con mouse y teclado.
2. Instalación del Visual Paradigm.
3. Instalación del paquete Office.
4. Instalación del PostgreSQL.
5. Instalación del pgadmin

Administrador de Configuración/Integración

1. PC, con mouse y teclado.
2. Instalación del Visual Paradigm.
3. Instalación del paquete Office.
4. Instalación del Subversion.

Diseñador de Pruebas

1. PC, con mouse y teclado.
2. Instalación del Visual Paradigm.
3. Instalación del paquete Office.

3.4 Organigrama de la arquitectura

A partir de los elementos y necesidades mencionadas se propone la Arquitectura en Capas, específicamente Arquitectura en 3 Capas para las aplicaciones que se realizan en entornos de escritorio. Se emplea este estilo debido a que es un modelo que 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.

La Arquitectura en Capas define cómo organizar el modelo de diseño en capas que pueden estar físicamente distribuidas, por lo que los componentes de una capa solo pueden hacer referencias a componentes de capas inmediatas inferiores.

Capa de presentación: Es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. En ella se muestran las interfaces con las que interactúan los usuarios, las cuales son realizadas con la utilización de la librería QT, esta librería tiene como principales clases para el diseño de la interfaz a QMainWindow, QWidget y sus componentes visuales.

Capa de negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. Las principales clases y paquetes que se usan son: QProcess, QThread, QTNetwork y Phonon.

Capa de Acceso a Datos: Es donde residen los datos y es la encargada de acceder a los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio donde el gestor es PostgreSQL y las principales clases que se utilizan son QsqlDatabase, QSqlQuery y SQlite

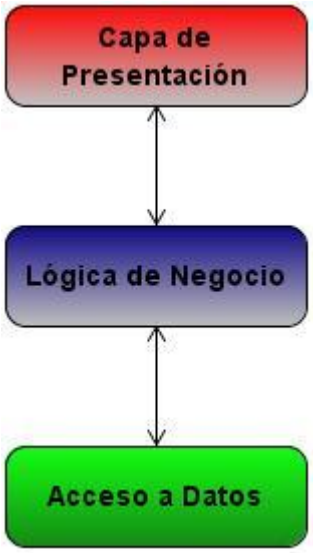

Figura 2 Arquitectura en 3 Capas

A continuación se detalla otro patrón el cual resulta de gran importancia para el desarrollo del proyecto, Modelo Vista Controlador (MVC). **"**El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador**" (Zaninotto, 2007)**.

Modelo: Contiene la lógica de negocio y es el encargado del acceso a datos. Se incluyen las clases de acceso a datos de PostgreSQL y clases con funcionalidades de acceso a datos característicos y relevantes del framework Symfony.

Vista: Define la interfaz de usuario, muestra al usuario la información que le necesita. Establece el control de los roles de usuario para la aceptación de peticiones. Captura los datos para la conformación de solicitudes al servidor.

Controlador: Se encarga de procesar las peticiones del usuario, de decidir cuál es la acción que se ejecutará a continuación y de realizar los cambios en la vista y en el modelo. El paquete contiene las clases base que ofrece el framework Symfony para validadores de formularios, helpers de objetos y formularios, plantillas, componentes de seguridad. Además se incluye la clase Controladora y otras clases de apoyo al proceso de control.

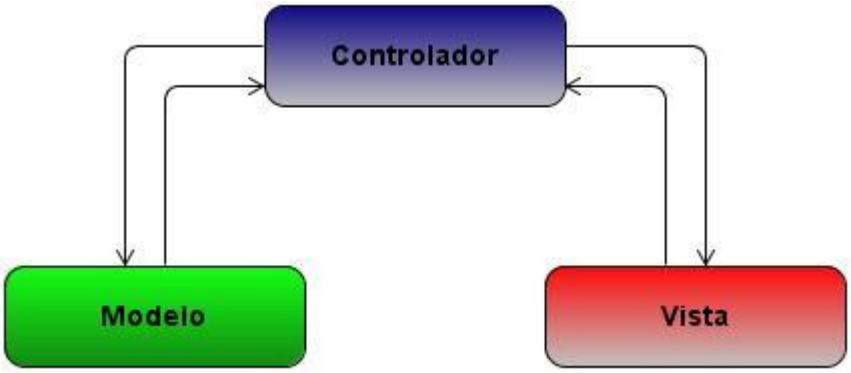

Figura 3 Modelo Vista Controlador

3.5 Módulos del Sistema

- ✓ Módulo Administración
- ✓ Módulo Catalogación
- ✓ Módulo Recuperación y Préstamo
- ✓ Módulo de Control de Ingesta
- ✓ Módulo de Procesamiento

3.5.1 Principales responsabilidad de cada módulo:

Módulo Administración

Este módulo permite insertar, modificar o eliminar usuarios, registrándose los cambios en la base de datos. Esta acción solo puede ser realizada por el administrador y es necesario que exista algún usuario al que se le desea cambiar su estado en el sistema. Permite buscar los usuarios presentes en el servidor LDAP con el objetivo de insertarlos al sistema posteriormente. Además el usuario puede establecer la configuración del Servidor Streaming activo en el sistema. El sistema permite al usuario emitir reportes sobre los datos almacenados, dentro de estos reportes se encuentran: reportes sobre datos de materiales almacenados, reporte de solicitudes y reporte de materiales insertados por editor.

Módulo Catalogación

El sistema permite realizar tres tipos de búsqueda: por clasificación, avanzada y básica, una vez obtenidos los resultados el usuario puede reproducir los videos para verificarlos. Durante el proceso de catalogación es posible realizar la misma acción sobre los subclips que se creen. La principal funcionalidad de este módulo es que permite al usuario insertar los datos descriptivos asociados a los materiales o subclips almacenados donde se muestra un listado de los materiales que no han sido catalogados.

Módulo Recuperación y Préstamo

El sistema permite realizar tres tipos de búsqueda: por clasificación, avanzada y básica. Permite reproducir los videos obtenidos a partir de las búsquedas realizadas. Además permite insertar, modificar, eliminar y atender solicitud. Previamente debe haberse insertado alguna solicitud para darle respuesta. Cuando un usuario con privilegios decide descargar los materiales que han sido solicitados se verifica que la solicitud se haya aprobado, se selecciona la solicitud y se inserta la ruta para la descarga. El sistema gestiona la transferencia de un fichero hacia un destino especificado, donde debe existir un directorio destino y un material en un directorio origen.

Módulo de Control de Ingesta

El sistema permite al digitalizador o supervisor del área de digitalización agregar un proyecto digitalizado al sistema o permite insertar, modificar o eliminar las tareas de edición del sistema. Permite al editor o supervisor del área de digitalización realizar y consultar las tareas de edición que le han sido asignadas, agregar un material editado al sistema, así como gestionar la transferencia de un fichero hacia un destino especificado.

***Módulo de* Procesamiento**

El sistema permite la ejecución de acciones de reencolado, cancelación y eliminación de los procesos de media que se ejecutan en las estaciones servidoras. Además el usuario puede realizar el proceso de codificación a baja calidad o codificar a un formato especificado. Se puede realizar la extracción del resumen visual de video, obteniéndose el mismo, esta funcionalidad permite analizar un material con el fin de que cumpla con el perfil establecido para los materiales originales, implícitamente se le extrae la duración de material. Permite notificar a la base de datos (BD) de un evento representativo en los flujos de procesos de media.

3.6 Descripción de la Arquitectura

Para una mejor comprensión de la arquitectura del sistema se utilizan varias vistas arquitectónicas. Las mismas están definidas por la metodología RUP. Serán modeladas utilizando Visual Paradigm Enterprise Edition.

- ✓ Vista de Casos de Uso
- ✓ Vista Lógica
- ✓ Vista de Implementación
- ✓ Vista de Despliegue
- ✓ Vista de Procesos

3.6.1 Vista de casos de Uso

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

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

El sistema cuenta con 32 casos de uso, de los cuales 27 son arquitectónicamente significativos, los mismos están representados en 5 módulos: Administración, Catalogación, Recuperación y Préstamo, Control de Ingesta y Procesamiento. Existen CUS que aparecen involucrados en varios módulos como son: Reproducir Material y Buscar Material, los cuales serán realizados en forma de componentes para que sean utilizados en los módulos que haga falta.

A continuación se presentan los diagramas que representan los Casos de Uso arquitectónicamente significativos para cada módulo:

- ✓ **Módulo de Administración**

Autenticar Usuario
Gestionar Usuario
Buscar Usuario LDAP
Configurar Acceso LDAP
Configurar Servidor Streaming

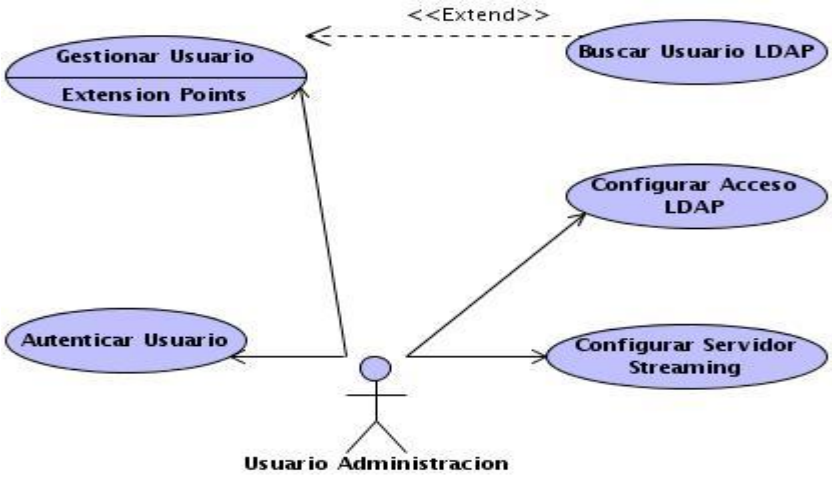

Figura 4 Diagrama de Caso de Uso del Módulo Administración.

Estos casos de uso constituyen la base para mantener la administración del sistema, haciendo énfasis en la seguridad del mismo.

- ✓ **Módulo de Recuperación y Préstamo**

Autenticar Usuario	Buscar Material	Reproducir Material
Descargar Material	Transferir Material	Insertar Solicitud
Eliminar Solicitud	Atender Solicitud	Modificar Solicitud

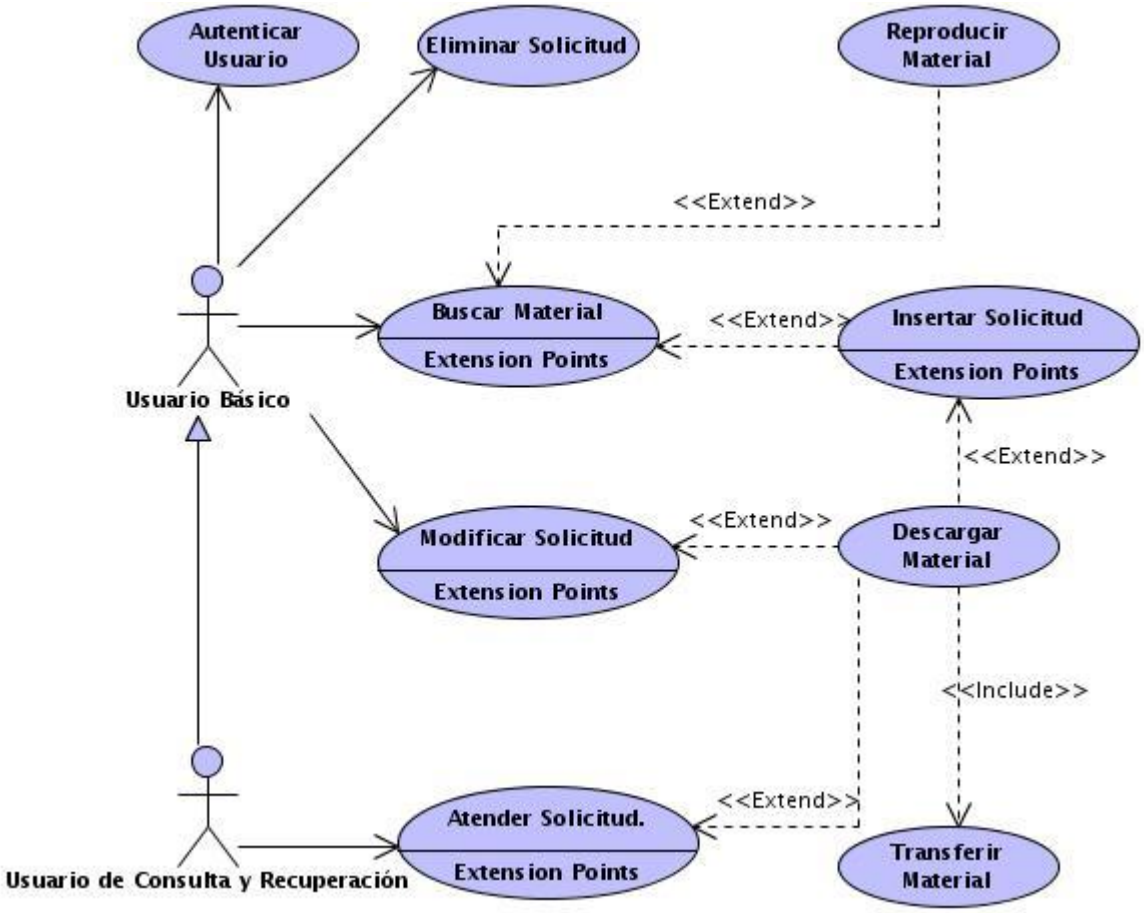

Figura 5 Diagrama de Caso de Uso del Módulo Recuperación y Préstamo.

Estos Casos de Uso son la base para la realizar las búsquedas y recuperación de los materiales.

- ✓ **Módulo de Control de Ingesta**

Autenticar Usuario	Adicionar Proyecto
Consultar Tarea	Transferir Material
Realizar Tarea	Adicionar Material Definitivo
Gestionar tarea	

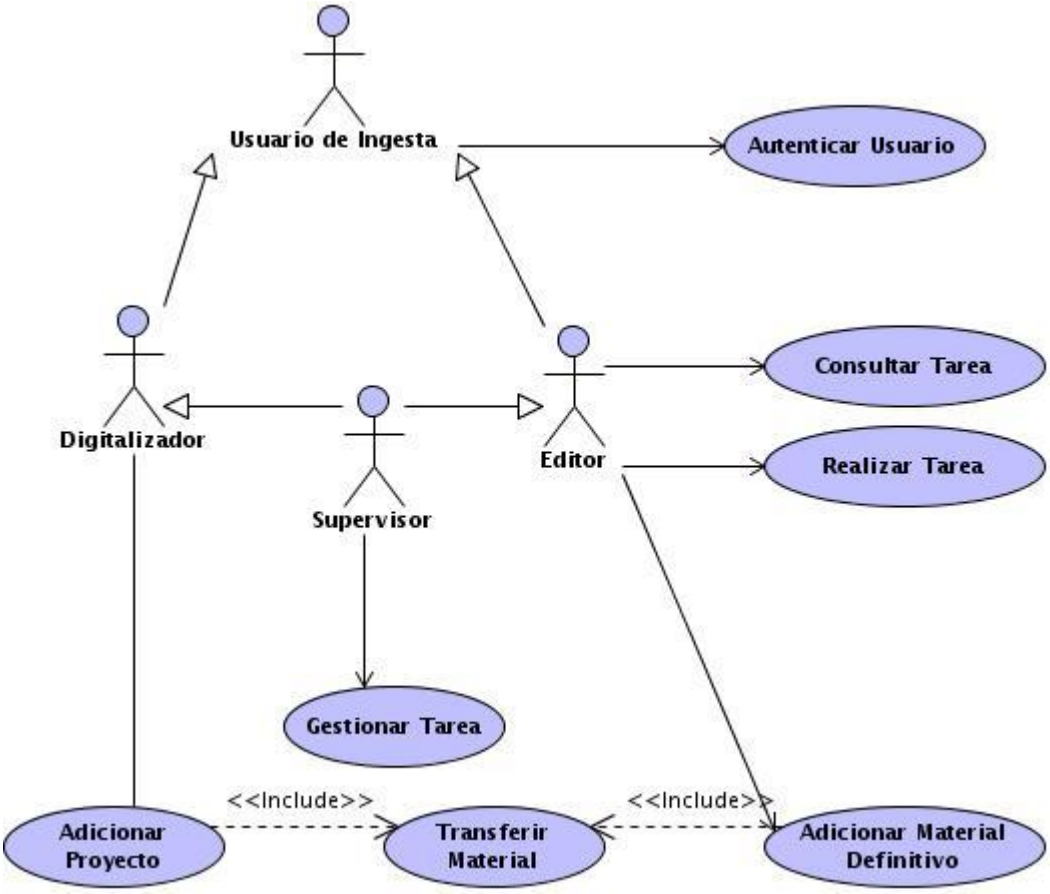

Figura 6 Diagrama de Caso de Uso del Módulo Control de Ingesta.

Los Casos de Uso del módulo Ingesta están muy relacionados con su nombre, pues se encargan de hacer llegar los materiales hacia el directorio destino luego de una breve descripción.

- ✓ **Módulo de Catalogación**

Autenticar Usuario	Catalogar Material
Buscar Material a Catalogar	Reproducir Material
Buscar Material	Administrar Subclip

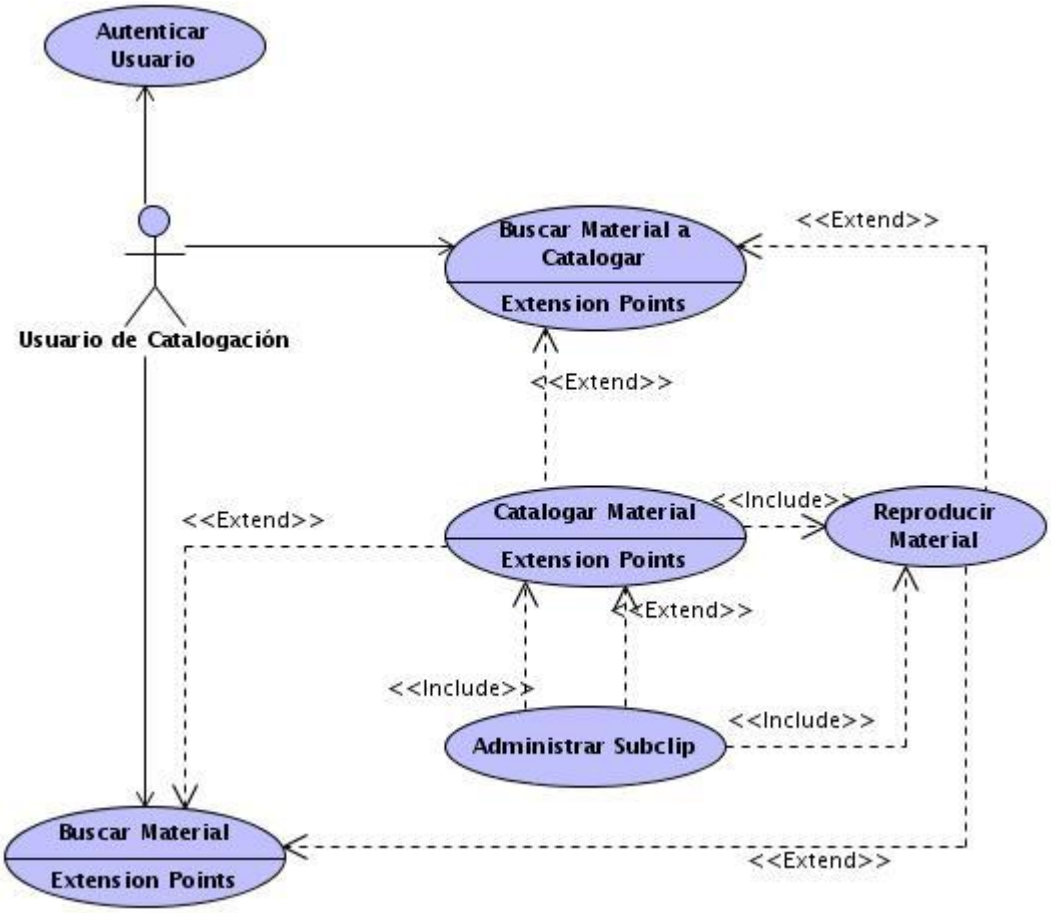

Figura 7 Diagrama de Caso de Uso del Módulo Catalogación.

Estos Casos de Uso son fundamentales para realizar la búsqueda y la descripción de los materiales almacenados.

- ✓ **Módulo de Procesamiento**

Autenticar Usuario	Analizar Video
Monitorear Procesos	Realizar Notificación
Gestionar Petición	Elaborar Resumen Visual de Video
Codificar Video	

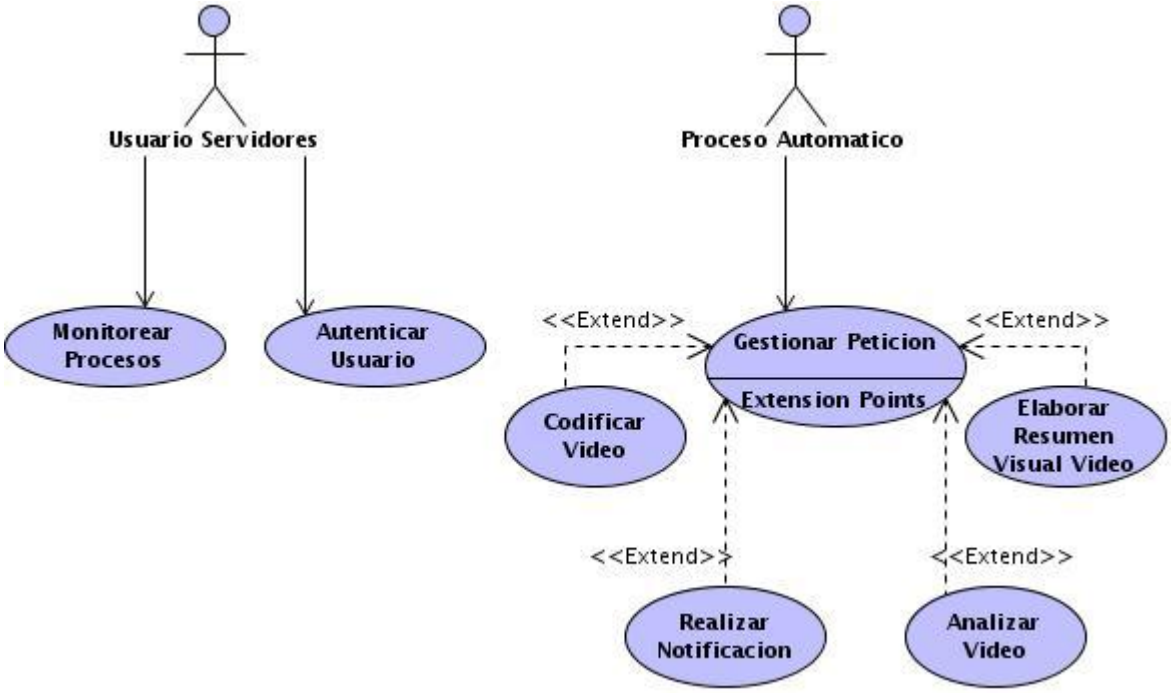

Figura 8 Diagrama de Caso de Uso del Módulo de Procesamiento.

Estos Casos de Uso son los encargados de dar respuesta a las peticiones realizadas por el usuario, brindan la posibilidad de monitorizar los procesos y de codificar un material en baja calidad o con un formato específico.

3.6.2 Vista Lógica

En la descripción de la arquitectura, la vista lógica describe las clases más importantes que formarán parte del ciclo de desarrollo. Se describen los paquetes más abstractos del sistema y las relaciones que entre ellos existen ya sea de dependencia o de uso. Las siguientes imágenes muestran la distribución de los paquetes más significativos dentro de cada una de las capas definidas para la aplicación y la dependencia entre ellos.

A continuación se detallan los paquetes más significativos de la vista lógica web:
Controlador: Todas las peticiones web son manejadas por un solo controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado. Entre las tareas comunes que realiza se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares.

Modelo: Se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que

dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Se pueden desarrollar aplicaciones que trabajen contra cualquier motor de bases de datos, sin cambiar la sintaxis ni las funciones a utilizar.

Vista: Se indica la forma de organizar los componentes dentro de un contenedor, determinando el tamaño y la posición de los mismos, se utilizan para la separación del código. Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. Normalmente sólo cambia el interior de la página. Por este motivo, la vista se separa en un layout y en una plantilla.

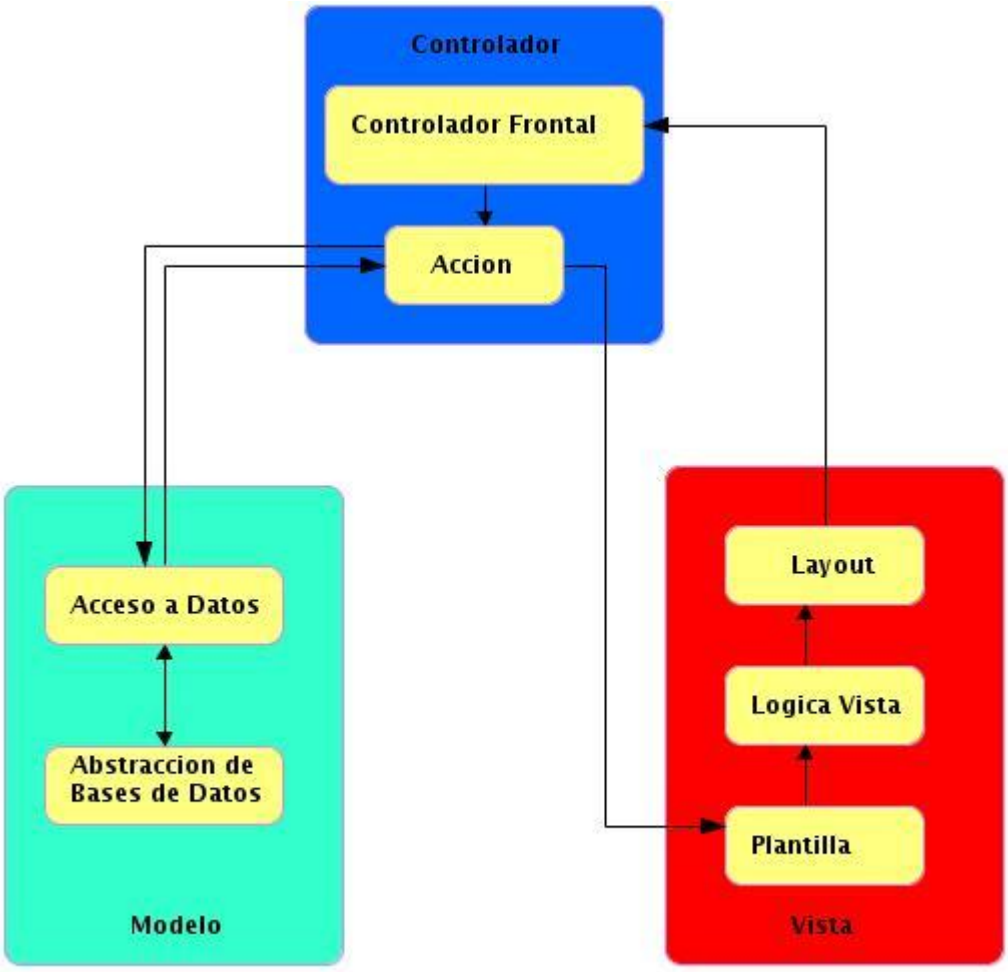

Figura 9 Diagrama de la Vista Lógica Web.

A continuación se describen los paquetes más significativos de la Vista Lógica Desktop:

Presentación: En este paquete están las clases y componentes de la librería QT. Está formado por componentes de interfaz, los componentes de esta capa administran la interactuación con el usuario, muestran los datos al mismo, obtienen sus datos personales e interpretan los eventos generados para actuar en los datos del negocio, cambiar el estado de la interfaz o facilitar la tarea al cliente.

Negocio: Contiene los principales paquetes del framework QT y las clases controladoras que encapsulan la lógica del negocio. Basado en estas clases es que se realiza el núcleo, la catalogación desktop y la plataforma de codificación. Dichas clases están enfocadas para el trabajo con medias, hilos, procesos y redes.

Acceso a Datos: Clases que controlan la lógica transaccional y la interacción con la base de datos. En este caso se usa SqLite para las bases de datos locales y PostgreSQL para las aplicaciones cliente-servidor. Las clases fundamentas a usar son SqlDatabase y SqlQuery.

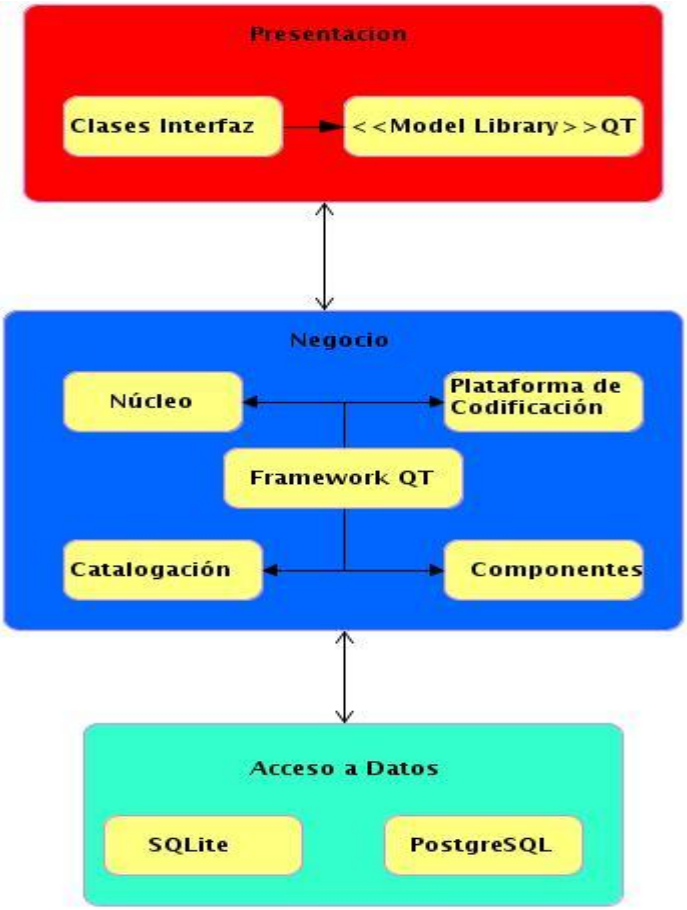

Figura 10 Diagrama de la Vista Lógica Desktop

3.6.3 Vista de despliegue

La vista de despliegue muestra la distribución física del sistema y cómo estarán distribuidos los componentes de la aplicación en ellos, describe el mapeo entre el software y el hardware reflejando su aspecto distribuido.

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

En la siguiente figura se muestra el diagrama de despliegue que se propone para el desarrollo del SCCM.

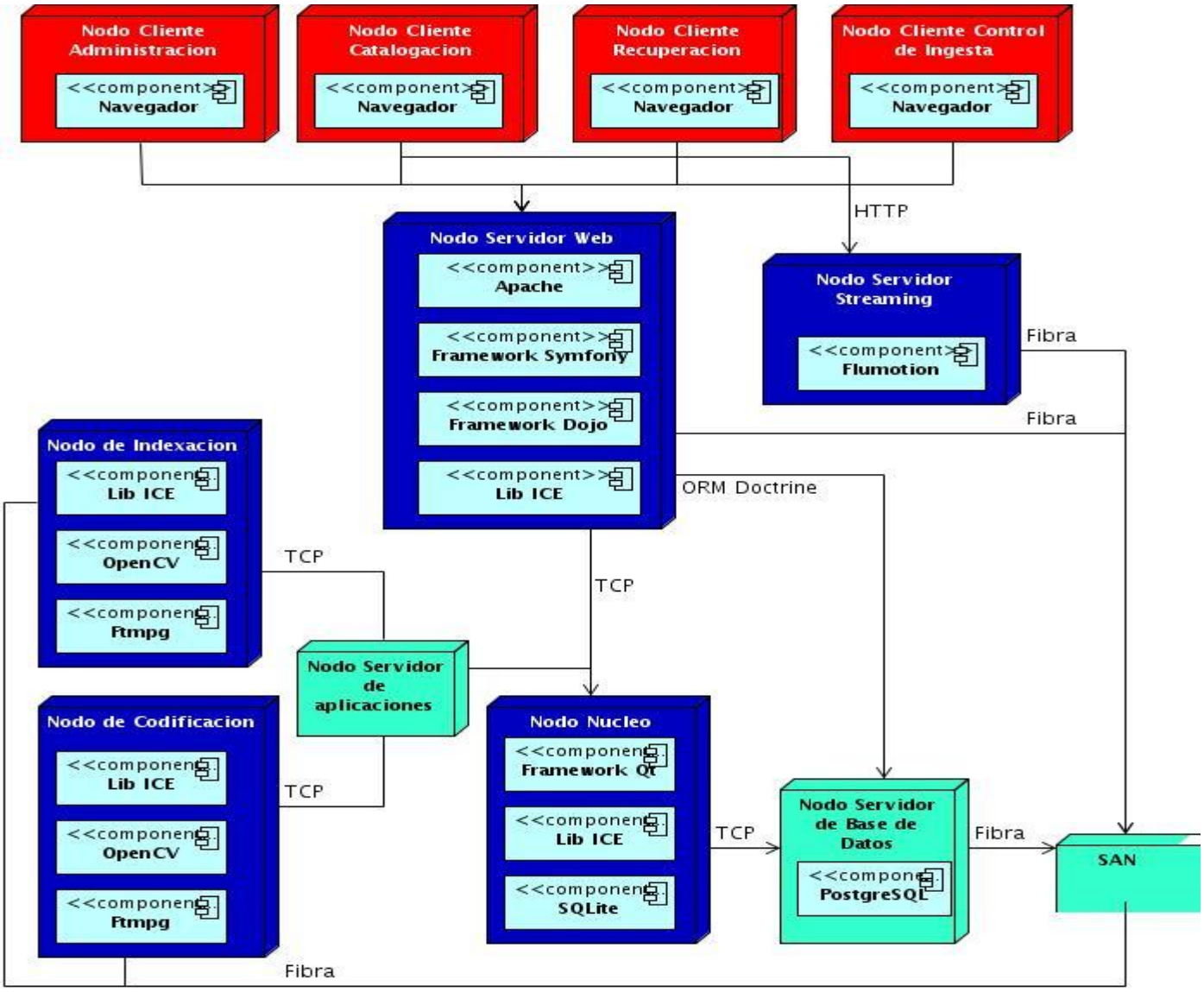

Figura 11 Diagrama de la Vista de Despliegue

A continuación se describirán los nodos físicos representados en el diagrama de despliegue:

Nodo Cliente Administración: permite lograr la administración del sistema y la organización de los materiales de interés.

Nodo Cliente Catalogación: permite realizar la búsqueda, la catalogación y la descripción de los materiales almacenados.

Nodo Cliente Recuperación: es la base para realizar las búsquedas y la recuperación de los materiales.

Nodo Cliente Control de Ingesta: se encargan de hacer llegar los materiales hacia el directorio destino luego de una breve descripción.

Nodo Servidor Web: se realiza la programación para las aplicaciones web del lado del servidor.

Nodo Servidor Streaming: Flumotion, para la transferencia de datos, la cual permite procesar de forma inmediata y continua un conjunto de información.

Nodo de Indexación: tiene como propósito mantener de forma ordenada la información, esto con la finalidad de obtener resultados de forma más rápida y relevante al momento de realizar una búsqueda.

Nodo de Codificación: específicamente diseñado para la compresión y descompresión de señales de audio y video.

Nodo Servidor de Aplicaciones: son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Nodo Núcleo: Es el principal responsable de facilitar la gestión de recursos, a través de servicios de llamada al sistema.

Nodo servidor de base de datos: permite crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

SAN: Su función es la de conectar de manera rápida, segura y fiable los distintos elementos que la conforman.

Descripción de los protocolos

Fibra: Representa el conector fibra óptica.

TCP: Transmission Control Protocol, en español Protocolo de Control de Transmisión.

HTTP: *Hypertext Transfer Protocol* , en español Protocolo de Transferencia de Hipertexto.

3.6.4 Vista de implementación

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

A continuación se detallan los paquetes principales de la vista de implementación web:

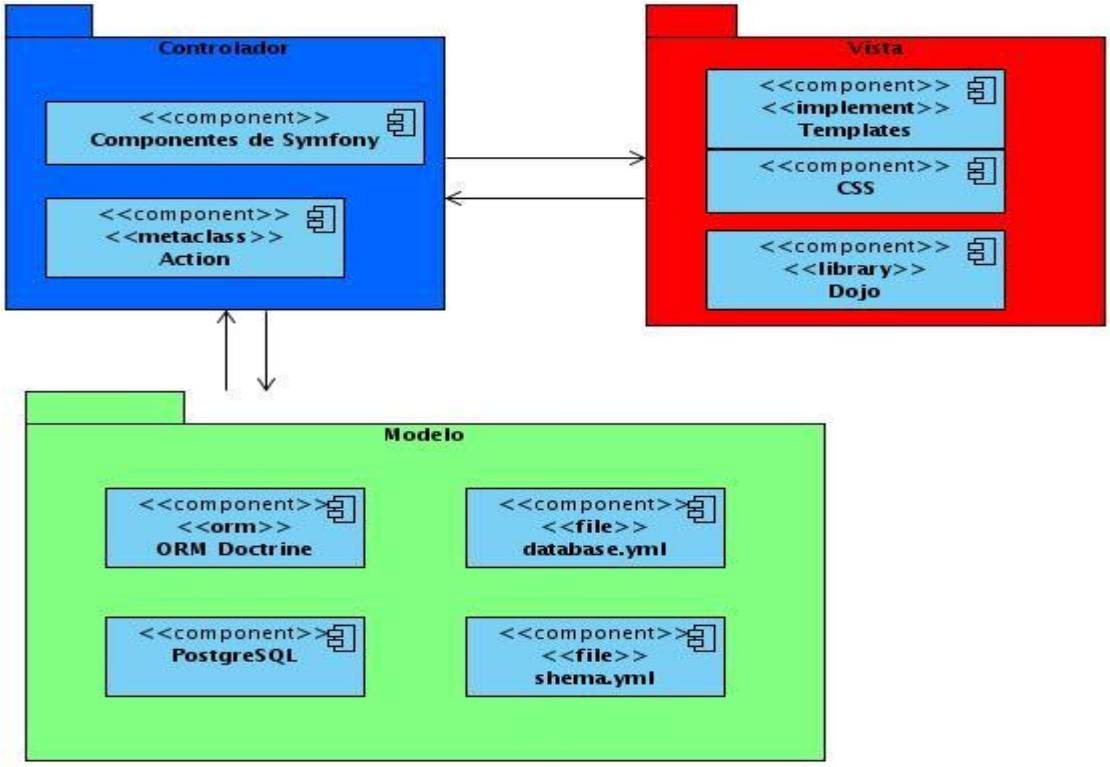

Figura 12 Diagrama de la Vista de Implementación Web

Controlador

- ✓ Componentes de Symfony: Representan todas las clases del framework que serán utilizadas durante el funcionamiento del sistema. Dígase validadores de formularios, helpers de objetos y formularios, plantillas, componentes de seguridad.
- ✓ Action: Se incluyen el código específico del controlador de cada página.

Vista

- ✓ Dojo: Contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web.
- ✓ CSS: El paquete contiene todos los estilos que se aplican en las interfaces del sistema.
- ✓ Templates: Plantillas para la realización de las páginas web.

Modelo

- ✓ ORM Doctrine: Para realizar un mapeo o conversión entre la información de la base de datos y los objetos PHP.
- ✓ PostgreSQL: Gestor de base de datos.
- ✓ Shema.yml: Es un archivo que describe todas las tablas y columnas de la base de datos.
- ✓ Database.yml: Contiene la definición de los accesos a bases de datos y las opciones de conexión de cada acceso.

A continuación se detallan los principales paquetes y clases de la vista de implementación desktop:

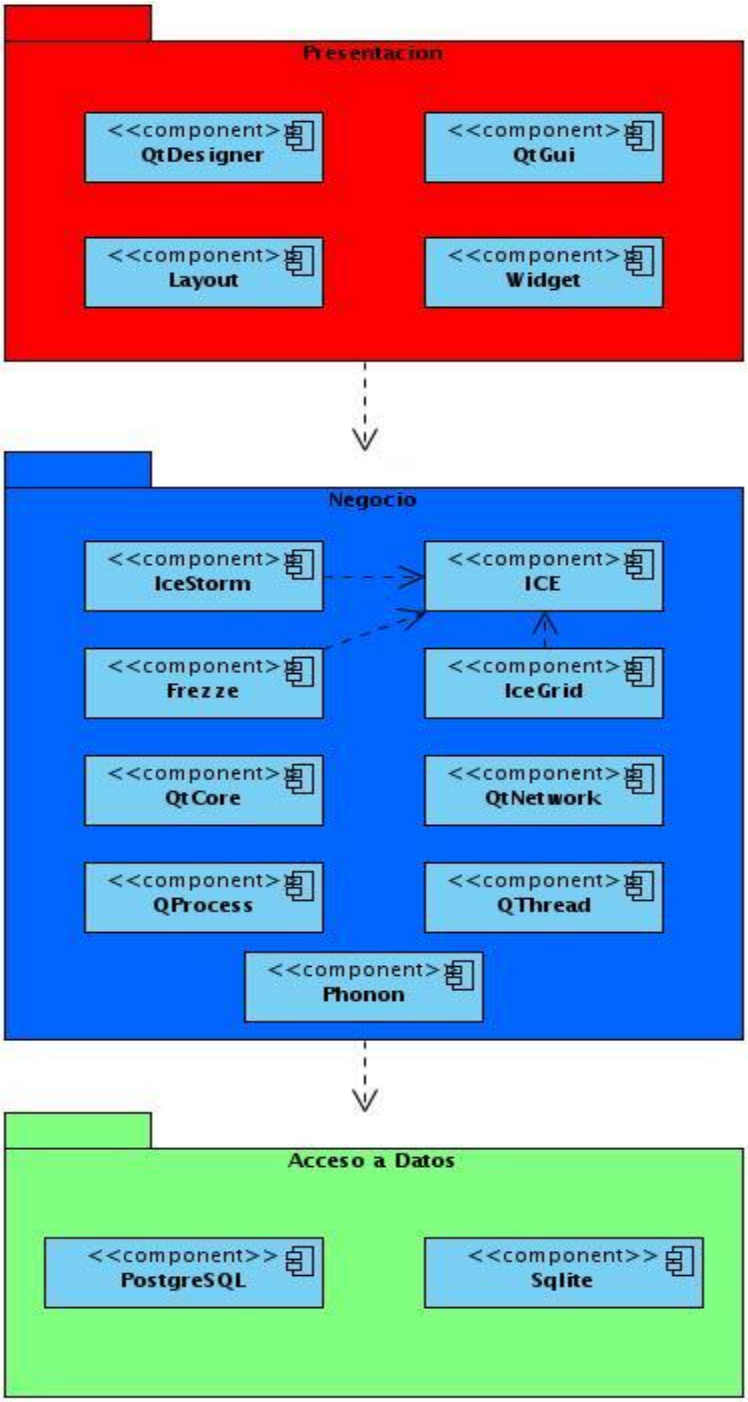

Figura 13 Diagrama de la Vista de Implementación Desktop

Presentación

- ✓ QtGui: Interfaces gráficas de usuario, esta técnica utiliza gráficos para proporcionar una interfaz fácil de usar en aplicaciones.
- ✓ QTDesigner: Este paquete se utiliza para diseñar las interfaces de usuario.

- ✓ Layout: Forma en que se establecen las conexiones.
- ✓ Widget: Proporciona fácil acceso a funciones frecuentemente usadas y provee de información visual

Negocio

- ✓ ICE (Internet Communication Engine): es un middleware orientado a objetos, proporciona herramientas y soporte de bibliotecas para el desarrollo de aplicaciones cliente-servidor.
- ✓ IceStorm: Es un servicio de publicación-subscripción.
- ✓ Frezze: Este paquete es para la persistencia de datos.
- ✓ IceGrid: Es la implementación de un servicio de localización.
- ✓ QtCore: Este paquete se utiliza pare el trabajo con medias.
- ✓ QtNetwork: Este paquete se utiliza para el trabajo con redes, ya sea HTTP, TCP/IP.
- ✓ QtProcess: Este paquete se utiliza para el trabajo con los procesos.
- ✓ QThread: Este paquete se utiliza para el trabajo con hilos.

Acceso a Datos

- ✓ PostgreSQL: Gestor de base de datos.
- ✓ Sqlite: Gestor de base de datos locales

3.6.5 Vista de Procesos

La vista de procesos suministra una base para el entendimiento de la organización de los procesos del sistema, ilustrados en el mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos o hilos concurrentes. En el proyecto existen algunos procesos que son de gran importancia que a continuación se detallan.

Ver anexo 1

Ver anexo 2

3.7 *Requisitos No Funcionales*

“Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los mismos están vinculados con los requisitos funcionales, es decir, una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse. Pueden ser clasificados más a fondo, en requisitos de funcionamiento, requisitos de capacidad de mantenimiento, requisitos de seguridad, requisitos de confiabilidad, o uno de muchos otros tipos de requisitos del software”. **(Sommerville, 2005)**

Requisitos de Seguridad

- ✓ La asignación de usuarios y sus opciones sobre el sistema se garantizan desde el módulo de Administración.
- ✓ Debe quedar constancia de quién, desde donde, y cuando se realizó una operación determinada en el sistema.
- ✓ La base de datos deberá estar disponible las 24 horas, pudiendo realizar operaciones sobre la misma en cualquier momento que se necesite. En caso de una falla en el servidor (interrupción del servicio eléctrico, desconexión momentánea o caída de la red) se deberá contar con un respaldo que le permita seguir con sus actividades.
- ✓ La información debe estar disponible en todo momento, donde los mecanismos de seguridad no deben impedir que los usuarios autorizados accedan a la misma.

Restricciones de acuerdo a la estrategia de diseño

- ✓ El diseño de las aplicaciones se hará utilizando la Programación Orientada a Objetos (POO). Encapsulación de la lógica por clases.
- ✓ Diseño e implementación de una arquitectura flexible, que permita la fácil integración o desintegración de componentes.
- ✓ El lenguaje de programación que se debe utilizar es PHP, HTML, Ajax y JavaScript para las aplicaciones web y C++ para las aplicaciones de escritorio.

Portabilidad, Escalabilidad y Usabilidad

- ✓ El sistema deberá hacer un uso racional de los recursos de hardware de la máquina, sobre todo en las PC clientes.
- ✓ Se desarrollará cada pieza del sistema en forma de componentes (elementos) con el fin de ser utilizados en futuras versiones del sistema.
- ✓ El sistema deberá contar con un documento de ayuda en formato digital, además de ser capaz de mostrar la ayuda en pantalla brindando indicaciones según el usuario lo necesite. De ser posible contará con la ayuda en línea.

Soporte

- ✓ El sistema permitirá modificar o añadirle módulos cuando sea necesario, asegurando su extensibilidad y mejores prestaciones.
- ✓ Las opciones de instalación y mantenimiento del sistema deben ser fáciles, además debe ser adaptable a cualquier plataforma.
- ✓ Los instaladores deben incluir todas las librerías y plugins que necesite el sistema para funcionar.

Usabilidad

- ✓ La aplicación debe ser concebida para ser utilizada por personas que tengan conocimientos básicos de informática y en el trabajo de gestión y catalogación de audiovisuales.
- ✓ El sistema deberá contar con combinaciones rápidas de teclas que faciliten el acceso a funcionalidades muy utilizadas del software para el uso de usuarios avanzados.
- ✓ El servidor streaming debe mantener buenas prestaciones (demanda de archivos).

Apariencia o Interfaz Externa

- ✓ La interfaz gráfica de la aplicación debe concebirse con un ambiente sencillo y de navegación fácil e intuitiva para el usuario.
- ✓ Las interfaces de la aplicación tendrán los componentes visuales necesarios para las operaciones correspondientes, evitando la sobrecarga de imágenes.
- ✓ Es conveniente no mostrarle al usuario muchos elementos funcionales al mismo tiempo con el objetivo de que este no pierda el hilo del proceso que está realizando y le sea más lógico el uso de cada una de las funcionalidades.

Rendimiento

- ✓ El sistema debe ser eficiente a las peticiones realizadas en cada momento, el flujo de trabajo que sigue la aplicación no permite el fallo de ninguna de las partes pues este influye de manera drástica sobre el próximo paso.
- ✓ El tiempo de respuesta promedio de las transacciones u operaciones realizadas en el sistema no debe exceder un tiempo máximo de 5 segundos.
- ✓ La BD deberá resistir una alta concurrencia, así como un gran número de operaciones simultáneas sobre la misma.

Requisitos de Software

- ✓ Se debe utilizar como servidor web el Apache 2.2.
- ✓ Para la instalación del NetBeans es necesario haber instalado la Máquina Virtual de Java o JVM.
- ✓ Se debe utilizar como sistema operativo Linux: OpenSuse, en su versión 11.3 para los servidores y para los clientes cualquier sistema operativo.
- ✓ Se debe utilizar como servidor streaming el Flumotion.
- ✓ Se debe utilizar como gestor de base de datos: PostgreSQL 9.0 o superior.
- ✓ Sistema operativo con un navegador web compatible con HTML 5.0 y CSS.

Requisitos de Hardware

Servidor de Bases de Datos

- ✓ Memoria RAM: 16 GB.
- ✓ Capacidad en disco duro: 80 GB.
- ✓ Procesador: 2 QuadCore.

Servidor de Aplicaciones

- ✓ Memoria RAM: 8 GB.
- ✓ Microprocesador: 1 QuadCore.
- ✓ Tarjeta de Red: Ethernet a 100 Mbps.

PC Cliente

- ✓ Procesador: Core2Duo.
- ✓ Memoria RAM 2 GB.
- ✓ Capacidad del disco duro: 160 GB.
- ✓ Punto de Red: 100 MB.

Servidor Streaming

- ✓ Debe contar con un microprocesador 2 QuadCore a 3 GHz de velocidad de procesamiento.
- ✓ Deberá poseer 16 Gb de memoria tipo RAM.

El hardware propuesto anteriormente es el ideal para este tipo sistemas, el mismo fue analizado y recomendado por especialistas en el tema de la dirección de la IP. Aún así las pruebas a realizarse son con un hardware mínimo: microprocesador dual core con una memoria RAM de 1 GB.

3.8 Conclusiones

En este capítulo, después de un estudio, de un total de 32 CUS se identificaron 27 arquitectónicamente significativos. Para garantizar la organización del proyecto se estableció la estructura del equipo de trabajo, donde quedaron identificados los roles más importantes, aspecto relevante en el desarrollo del software. Por su importancia se detallaron cada una de las vistas arquitectónicas, quedando como guía para los miembros del proyecto. También se establecieron los requisitos no funcionales con el propósito de establecer las propiedades o cualidades que el producto debe tener.

CAPÍTULO 4: Evaluación de la Arquitectura.

4.1 Introducción

La arquitectura es un artefacto definitivo para la calidad del software que se desarrolla. Durante la evaluación de la misma se puede conocer los diferentes riesgos asociados con el desarrollo del sistema, y como la arquitectura es un producto temprano son muchos los errores que se pueden corregir durante el flujo de requerimiento o diseño, siendo estos menos costosos que si tuvieran que aclararse en implementación o prueba. En este capítulo se analizan los resultados obtenidos a través de una estrategia de prueba y se valoran dos sistemas con arquitecturas similares a la del proyecto que se desarrolla para comparar los resultados obtenidos.

4.2 Análisis de Arquitectura de Software

Si las decisiones arquitectónicas determinan los atributos de calidad del sistema, entonces es posible evaluar las decisiones arquitectónicas con respecto a su impacto sobre dichos atributos. Cuanto más temprano se encuentre un problema en un proyecto de software, es mejor. Realizar una evaluación de la arquitectura es la manera más económica de evitar desastres. La evaluación de una arquitectura no produce resultados cuantitativos, sino ayuda a encontrar debilidades.

“El tema de evaluar una arquitectura, estimar el comportamiento de los atributos ante cambios arquitectónicos, lograr hacer predicciones de diseño en etapas tempranas, siempre han sido actividades de vital interés para la industria, debido a que todas estas predicciones aseguran la calidad del software, la disminución de los tiempos de desarrollo, y por lo tanto el esfuerzo y los costos de cada producto“. **(Regalado, 2008)**

La AS de los sistemas de software a ser construidos, se convierte en un factor de importancia para lograr que éste tenga un alto nivel de calidad. El poseer una buena Arquitectura de Software es de suma importancia, ya que ésta es el corazón de todo sistema informático y determina cuáles serán los niveles de calidad asociados al sistema.

“No sirve de nada un sistema que no cumple con los atributos de calidad que se especificaron en los requisitos no funcionales de los clientes. Por lo que diseñar una correcta arquitectura va a determinar el éxito o fracaso de un sistema de software, en la medida que esta cumpla o no con sus objetivos. Debido a esto para reducir tales riesgos, y como buena práctica de ingeniería, es recomendable realizar evaluaciones a la arquitectura“. **(López, 2008)**

4.2.1 Técnicas

"Las técnicas utilizadas para la evaluación de los atributos de calidad requieren grandes esfuerzos por parte de los ingenieros de software para crear especificaciones y predicciones. Estas técnicas requieren información del sistema a desarrollar que no está disponible durante el diseño arquitectónico, sino al principio del diseño detallado del sistema". **(Bosch, 2000)**

Entre las técnicas de evaluación de arquitecturas de software se encuentran: la evaluación basada en escenarios, evaluación basada en simulación, evaluación basada en modelos matemáticos y evaluación basada en experiencia.

4.2.2Métodos

Estos métodos son empleados para un alto nivel de análisis de toda arquitectura. Realizan solamente el análisis cualitativo de los atributos de calidad. Indican lo que se debe mejorar en la arquitectura para la satisfacción de los atributos de calidad esperados. Son métodos bastante generales, por lo que es grande el esfuerzo que se realiza para su aplicación en una arquitectura en particular. Para sus evaluaciones, se basan en la técnica de escenarios. De manera independiente, no logran abarcar una evaluación integral de los atributos de calidad. Algunos de estos métodos son:

"ATAM (Architecture Trade-off Analysis Method): está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM (Software Architecture Analysis Method).

SAAM: Este método evalúa con más profundidad, en relación con otros métodos cuestiones referentes a la arquitectura, como son los atributos de calidad.

ADR (Active Design Review): ADR es utilizado para la evaluación de diseños detallados de unidades del software como los componentes o módulos. Las preguntas giran en torno a la calidad y completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto.

ALMA (Architecture Level Modifiability Analysis): Este método es el resultado de los trabajos de investigación de Brengtsson y Lassing El atributo de calidad que analiza este método es la facilidad de modificación. Esto se refiere a la capacidad de un sistema para ser ajustado debido a cambios en los requerimientos, o en el entorno, así como la adición de nuevas funcionalidades". **(López, 2008)**

4.3 Estrategia de Prueba de la AS

A la hora de calificar la calidad de una aplicación, uno de los factores más importantes es la usabilidad. Es el atributo más visible ya que determina el grado de satisfacción del usuario respecto a la aplicación web; de ello depende que sea utilizada o no. La ISO 9126 [basada en el modelo de Mc Call] plantea un modelo normalizado que permite evaluar y comparar productos sobre la misma base. Aquí la calidad queda definida a un alto nivel de abstracción por seis características:

- ✓ "**Funcionalidad:** Las funciones satisfacen necesidades declaradas o implícitas.
- ✓ **Fiabilidad:** Capacidad de un sistema para mantener su nivel de rendimiento.
- ✓ **Usabilidad:** Esfuerzo necesario para el uso y la valoración individual de tal uso, por parte de un conjunto de usuarios.
- ✓ **Portabilidad:** Es la capacidad de un sistema para ser transferido de un entorno a otro.
- ✓ **Mantenibilidad:** Es el esfuerzo necesario para realizar modificaciones específicas.
- ✓ **Eficiencia:** Es la relación entre el nivel de prestaciones de un sistema y el volumen de recursos utilizados en condiciones declaradas". **(González, 2011)**

Para evaluar la calidad de la arquitectura se emplean los mismos atributos seleccionados y se establece una escala de valores de 1 a 10, la cual expresa de forma cuantitativa la forma en que se cumple cada uno de los atributos y se brinda una valoración final calculando el promedio de valores de cada uno de los parámetros de calidad.

[1-5] mal cumplimiento

[6-7] cumplimiento medio

[8-10] buen cumplimiento

Funcionalidad

La propuesta de la arquitectura cumple con este atributo ya que utiliza RUP como metodología de desarrollo permitiéndole al equipo de trabajo seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. Además se describen todos los diagramas necesarios para el buen funcionamiento del sistema. Se utiliza una herramienta Pgpool-II que permite la replicación de la base de datos para no perder la información de la BD y poder seguir funcionando.

Se considera que tiene un valor de 10 dentro de la escala.

Fiabilidad

El sistema debe responder a las peticiones realizadas en cada momento, el flujo de trabajo que sigue la aplicación no permite el fallo de ninguna de las partes pues este influye de manera drástica sobre el próximo paso. Sin embargo si ocurre una falla en la conexión con el servidor streaming el sistema se verá afectado grandemente.

Se considera que tiene un valor de 7 dentro de la escala.

Usabilidad

La propuesta de la arquitectura cumple con este atributo porque la aplicación debe ser concebida para ser utilizada por personas que tengan conocimientos básicos de informática y en el trabajo de gestión y catalogación de audiovisuales, las interfaces con las que interactúa el usuario son sencillas. Además el sistema deberá estar bajo las normas establecidas por la norma ISO 9241-11, la cual define los parámetros internacionales de usabilidad de cualquier software

Se considera que tiene un valor de 9 dentro de la escala.

Portabilidad

La propuesta de arquitectura cumple con este atributo porque el sistema está analizado para desarrollarse con herramientas libres, además de la característica de ser multiplataforma y seguir una estructura basada en componentes.

Se le considera un valor de 8 dentro de la escala.

Eficiencia

La propuesta de arquitectura cumple con este atributo porque la BD se diseña teniendo en cuenta que debe resistir una alta concurrencia, así como un gran número de operaciones simultáneas sobre la misma. El diseño e implementación de una arquitectura flexible permite la fácil integración o desintegración de componentes.

Se le considera un valor de 8 dentro de la escala.

Mantenibilidad

La propuesta de arquitectura cumple con este atributo porque el sistema permitirá modificar o añadirle módulos cuando sea necesario, asegurando su extensibilidad y mejores prestaciones. Si ocurre algún cambio, sólo se tendrían que realizar las correcciones necesarias en el nivel requerido. Además la base de datos se crea dinámica con el objetivo de que se pueda seguir aumentando la cantidad de funcionalidades.

Se le considera un valor de 8 dentro de la escala.

4.3.1 Análisis de los Resultados

Después del análisis del valor que alcanza cada uno de los atributos dentro de la escala propuesta se obtuvo el promedio de la calidad de la arquitectura dando como resultado 8.33 por lo que se considera que cumple con los parámetros de calidad analizados. Se debe mejorar el atributo fiabilidad en el cual se obtuvo el menor valor dentro de la escala definida, ya que un aspecto fundamental es la conexión del servidor streaming.

4.4 Valoración de Propuestas Arquitectónicas Similares

Esta valoración consiste en analizar propuestas arquitectónicas en sistemas de similares propósitos. Se seleccionaron dos aplicaciones, PTARTV (Plataforma de Transmisión Abierta para Radio y Televisión) perteneciente al departamento de Señales Digitales del Centro de Desarrollo GEySED y Videoma que se encarga del desarrollo de aplicaciones de gestión multimedia, la cual es un potente software a nivel internacional.

4.4.1 PTARTV

Esta aplicación perteneciente a la Universidad de las Ciencias Informáticas permite la transmisión manual de ficheros de video, la codificación del fichero y su transferencia al lugar de transmisión, además permite la gestión de medias, aspecto similar al proyecto SCCM.

Patrones Arquitectónicos utilizados

Estas dos arquitecturas se basan en el modelo cliente-servidor, además tienen en común que utilizan el patrón en capas para la parte de escritorio para la administración de la transmisión y la transmisión en PTARTV y para el gestor de proceso y la plataforma de codificación en SCCM, lo que facilita que en caso de realizar alguna modificación, sólo se tuvieran que realizar las correcciones necesarias en el nivel requerido sin tener que revisar código de otros niveles. También se emplea Modelo Vista Controlador para las aplicaciones Web, donde el acceso a la aplicación es de manera similar, con permisos definidos para cada usuario. Además es común en ambas aplicaciones el uso de la Arquitectura Multicapas Orientada a Objetos y de varios patrones de diseño como son: Experto, Creador, Bajo Acoplamiento, Alta Cohesión y Fachada.

Comparación entre las herramientas y las tecnologías.

PTARTV cuenta con el lenguaje de programación Java, esto se puede dificultar a la hora de desplegar la aplicación fuera del país, porque es necesaria la compra de la licencia de la máquina virtual de Java, en cambio en SCCM se utiliza C++ que es libre. En ambos proyectos se utiliza PHP junto con el framework Symfony para las aplicaciones web, de ambas existe una amplia documentación, se pueden utilizar en cualquier sistema operativo, lo que influye para que sea un software multiplataforma, aunque para la

creación de las interfaces web PTARTV utiliza ExtJs. Como entornos de desarrollo PTARTV escogió Netbeans para las aplicaciones de escritorio y Eclipse para las aplicaciones web. La utilización de PostgreSQL en ambas aplicaciones como sistema gestor de base de datos es factible ya que es muy rápido, tiene buenas utilidades de administración, es confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños, aunque en SCCM la base datos es dinámica, lo que permite la escalabilidad del sistema y se utiliza una importante herramienta: pgpool-II, lo cual permite el balanceo de cargas, las consultas paralelas y la replicación de la base datos que posibilita mejorar el rendimiento del sistema.

4.4.2 Videoma

"Es un sistema de análisis y seguimiento de TV que realiza la grabación, catalogación, clipping y búsqueda automática de contenidos de TV. Una solución especialmente diseñada para empresas, Gobierno, Administración Pública, agencias de noticias, medios de comunicación y para todas aquellas entidades que en su rutina de trabajo diario, consumen e interactúan con información procedente de Televisión". **(Garcia, 2002)**

Funcionalidades similares entre Videoma y SCCM

- ✓ Planificación y monitorización de la grabación.
- ✓ Documentación del material mediante plantillas personalizables por el usuario, de la que derivan los campos de documentación y metadatos.
- ✓ Selección de un set de palabras claves en el sistema (seguridad, terrorismo, manifestación).
- ✓ Extracción de clips analizados (fragmentos del vídeo grabado).
- ✓ Búsqueda del material grabado mediante nombre de la cadena, fecha y hora.
- ✓ Búsqueda por keyword (palabras clave).
- ✓ Guardar el clip para una edición y un posterior envío a los usuarios que lo seleccionen.

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

a la arquitectura, buscan mantener la seguridad, garantizar el rendimiento y que sea escalable. Videoma usa SOA y SCCM utiliza MVC y Capas, donde cada uno de ellos tienes sus ventajas en dependencia de las condiciones y donde se use.

4.5 Conclusiones

En este capítulo se realiza la evaluación de la arquitectura del Sistema de Captura y Catalogación de Medias, en el que se presentó una estrategia de prueba, basada en la norma ISO-9126, donde los resultados obtenidos fueron satisfactorios. Se realizó una valoración de la arquitectura con dos sistemas similares, PTARTV por la parte nacional y Videoma como internacional, donde la SCCM cumple con características más fuertes que PTRTV y similares a las de Videoma. Además como la AS tiene relación directa con los requisitos no funcionales, se introduce ICE para comunicar las aplicaciones de manera muy rápida, Sqlite para ganar en tiempo de respuesta y Pgpool-II para mejorar la seguridad y el rendimiento, en estos términos la propuesta para SCCM es más completa que las estudiadas.

CONCLUSIONES GENERALES

Una vez culminado el desarrollo de la investigación es posible arribar a las siguientes conclusiones:

Con la selección de los patrones arquitectónicos, Capas para las aplicaciones de escritorio, MVC para las web y Arquitectura Orientada a Objeto para ambas, se mejora la estructura y disminuye la complejidad en el sistema.

Para evitar conflictos, ganar en tiempo de respuesta y establecer la comunicación entre las aplicaciones de diferentes lenguajes, en este sistema C++ y PHP se identifica el middleware ICE.

Con la utilización de la herramienta Pgpool-II se mejora el rendimiento y la seguridad del Sistema de Captura y Catalogación de Medias debido a sus funcionalidades de replicación, balanceo de carga y paralelización de consultas.

A partir de la realización de la evaluación de la arquitectura del Sistema de Captura y Catalogación de Medias, a través de la estrategia de prueba basada en la norma ISO-9126, enfocada en un grupo de atributos de calidad se obtuvieron resultados satisfactorios.

RECOMENDACIONES

Al concluir el desarrollo de este trabajo se recomienda:

Se recomienda utilizar esta arquitectura en otras aplicaciones de similar propósito dentro y fuera del departamento de Señales Digitales.

Se recomienda el refinamiento constante de la arquitectura pero enmarcado en las diez vistas propuestas por el expediente de arquitectura a nivel UCI.

GLOSARIO DE TÉRMINOS

API: Application Programming Interface. Es una propiedad mediante la cual los programas pueden solicitar peticiones para ser atendidos o utilizar un servicio del sistema operativo y de otros programas.

Catalogación: Es la forma de introducir datos a las medias, registrarlas y clasificarlas.

CSS: Hojas de estilo en cascada, ofrece la posibilidad de separar la estructura de un documento estructurado escrito en HTML o XML de su presentación.

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

Codificación: Cualquier representación de un conjunto de datos por medio de otro. En este caso serían los archivos de multimedias analógicas convertidos digitalmente.

Failover: Procedimiento contra fallos.

Framework: Marco de trabajo, solución reutilizable y extensible.

Hardware: parte física de un computador y más ampliamente de cualquier dispositivo electrónico.

IEEE: Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

Interfaz: Zona de contacto o conexión entre dos componentes de "hardware"; entre dos aplicaciones, o entre un usuario y una aplicación. Apariencia externa de una aplicación informática.

ORM: Mapeo Relacional de Objetos, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

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

***Sistema*:** Conjunto de elementos que guardan entre sí algún tipo de relación. En las áreas de informática, los sistemas son todos aquellos programas que se elaboran para satisfacer las posibles necesidades de información automatizada de cada área en particular.

Storyboard: Un conjunto de ilustraciones mostradas en secuencia, con el objetivo de servir de guía para entender una historia.

Streaming: Es una técnica de transferencia de datos, la cual permite procesar de forma inmediata y continua un conjunto de información. Esta tecnología posibilita incrementar en forma importante las posibilidades de Internet de visualizar grandes cantidades de información multimedia donde se puede visualizar videos sin tener que descargarlos.

Watchfolders: Es una aplicación para Windows que permite agregar tantas carpetas como se quieran, entre otras funcionalidades.

BIBLIOGRAFÍA

Allison, Chuck. 2000. ***Fundamentos para Java y C++. s.l. : MindView, Inc.*** **2000.**
Bass, Len, Clement, Paul y Kazman, Rick. 2003. ***Software Architecture in Practice . s.l. : Addison-Wesley Professional; 2 edition .*** **2003.**
Bosch, Jan. 2000. ***Design and Use of Software Architectures .*** **s.l. : Addison-Wesley Professional, 2000. 0201674947.**
Buchmann, Frank. 1996. ***Pattern Oriented Software Architecture 1 edition.*** **1996.**
Daum, Berthold. 2005. ***Eclipse 3 Para Desarrolladores Java. s.l.*** **2005. 8441518815.**
E. Gamma, R Helm, R Johnson, J Glissades. 1998. ***Design Patterns. Elements of Reusable Object-Oriented Software. .*** **1998.**
Eguiluz, Javier. 2011. **symfony.es. [En línea] 2011. [Citado el: 05 de 02 de 2011.] http://www.symfony.es/que-es-symfony/.**
Fernandez, David Vallejo. 2006. ***Libreria ICE.*** **2006.**
Garcia, Antonio Albacete. 2002. ***Videoma ISID.*** **Madrid : s.n., 2002.**
González, Carlos D. 2011. **USABILIDADWEB.COM.AR. [En línea] abril de 2011. [Citado el: 05 de abril de 2011.] http://www.usabilidadweb.com.ar/metodos_eval_calidad_web.php.**
Grupo Soluciones GSInnova. 2007. **GSInnova. [En línea] 2007. [Citado el: 10 de 01 de 2011.] http://www.rational.com.ar/herramientas/rup.html.**
Guiarte Multimedia S.L. . 2002. ***desarrolloweb.com.*** **[En línea] 2002. http://www.desarrolloweb.com/articulos/840.php..**
Harreman, David. 2007. **SQLite Latino America . [En línea] 9 de noviembre de 2007. [Citado el: 11 de 01 de 2011.] http://sqlite-latino.blogspot.com/2007/11/sqlite-de-cdigo-legible.html .**
Hilliard, Rich Recommended Practice for Architectural Description of Software- Intensive Systems. 2000. **[En línea] 2000. http://www.enterprisearchitecture.info/Images/Documents/IEEE%201471-2000.pdf].**
Jaume Sabater. 2008. **Replicación y alta disponibilidad de PostgreSQL con pgpool-II. [En línea] 1 de noviembre de 2008. [Citado el: 5 de febrero de 2011.] http://linuxsilo.net/articles/postgresql-pgpool.html.**
Javier, Enrique y. 2011. **BuenasTareas.com. [En línea] 2011. [Citado el: 10 de 02 de 2011.] http://www.buenastareas.com/ensayos/Lenguaje-Unificado-De-Modelado/2144752.html.**
Jelsoft Enterprises Ltd. 2011. **ProDescargas. [En línea] 2011. [Citado el: 05 de 02 de 2011.] http://foro.prodescargas.com/showthread.php?t=3631.**
Kiccillof, Nicolás. 2004. ***Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.*** **2004.**
Larman, Craig. 1999. ***UML y Patrones. Introducción al análisis y diseño orientado a objetos.*** **1999.**
Lopez, Alfredo Castañeda y Miguel. 2011. **Zona QT. [En línea] 2011. [Citado el: 15 de 01 de 2011.] http://www.zonaqt.com/tutoriales/tutorial-b%C3%A1sico-de-qt-4.**
Lopez, Carlos. 2008. **GestioPolis.com. [En línea] 2008. [Citado el: 1 de abril de 2011.] http://www.gestiopolis.com/administracion-estrategia/procedimiento-para-la-evolucion-de-las-arquitecturas-de-software.htm.**

Martínez, Claudia. 2008. **Comunidad de Programadores. [En línea] 14 de 07 de 2008. [Citado el: 10 de 06 de 2011.] http://www.lawebdelprogramador.com/foros/C_sharp/284565-Ventajas_de_C_.html.**
Martinez, Rafael. 2011. **PostgreSQL-es. [En línea] 2011. [Citado el: 17 de 02 de 2011.] http://www.postgresql-es.org/sobre_postgresql.**
Mehdi Achour, Friedhelm Betz, Antony Dovgal. 2001. **PHP. [En línea] 2001. [Citado el: 20 de 01 de 2011.] http://php.net/manual/es/index.php.**
Merlino, H. 2005. **[En línea] 2005. [Citado el: 28 de 11 de 2010.] http://laboratorios.fi.uba.ar/lsi/rtis-8-2-30-35.pdf.**
Microsoft. 2011. **Lo nuevo de asp.net. [En línea] 2011. [Citado el: 15 de 03 de 2011.] http://msdn.microsoft.com/es-es/library/fa1h9d0d%28v=vs.80%29.aspx.**
Mora, Sergio Luján. 2009. **Grupo de Investigación en Procesamiento del Lenguaje Natural y Sistemas de Información. [En línea] 2009. [Citado el: 02 de 05 de 2011.] http://gplsi.dlsi.ua.es/~slujan/asp/Introduccion_a_ASP.htm.**
Multistream S.L. 2005. **Codistream. *Codistream.* [En línea] 2005. [Citado el: 5 de 12 de 2010.] http://www.multistream.tv/productos-codistream-fullhd-streaming-esp.aspx.**
Oracle Corporation. 2011. **NetBeans. [En línea] 2011. [Citado el: 15 de 01 de 2011.] http://netbeans.org/community/releases/69/index_es.html.**
Peña, Lilian Teresa Castro Mecías y Rolando Bermúdez. 2008. ***Desarrollo de una libreria de Componentes JavaScript basado en Dojo Toolkit.* 2008.**
Pérez, Javier Eguíluz. 2008. **Introducciona a AJAX. [En línea] 2008. [Citado el: 06 de 03 de 2011.] http://www.librosweb.es/ajax/.**
Pilato, Ben Collins-Sussman Brian W. Fitzpatrick y C. Michael. 2004. ***Control de versiones con Subversion.* 2004.**
Pozuelo, Aurelio Hernández y Amado José. 2006. ***Soluciones Videoma.* [En línea] 2006. [Citado el: 28 de 11 de 2010.]**
Regalado, Yamila Vigil. 2008. ***Metodología de evaluación de arquitecturas de software. [Tesis de Maestria].* La Habana : s.n., 2008.**
Sommerville, Iam. 2005. ***Ingenieria del Software 6ta Edición.* Espanna, Madrid : s.n., 2005.**
Tedial. 2009. **Tedial. [En línea] Tedial, 2009. [Citado el: 10 de 12 de 2010.] http://www.tedial.com/index.php?option=com_content&view=article&id=102&Itemid=66.**
—. 2009. **Tedial. *Tedial.* [En línea] Tedial, 2009. [Citado el: 10 de 12 de 2010.] http://www.tedial.com/index.php?Itemid=65.**
Villegas, Adrian Anaya. 2009. **monografías.com. [En línea] 2009. [Citado el: 10 de 01 de 2011.] http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema.shtml#program.**
virtualnauta.com. 2007. **Virtualnauta. [En línea] 2007. [Citado el: 15 de enero de 2011.] http://www.virtualnauta.com/es/html/home.php.**
Wallace, Chris. 2010. **Tux Maya Blogs . [En línea] 2010. [Citado el: 05 de 02 de 2011.] http://tuxmaya.wordpress.com/2010/01/23/programacion-en-c-y-qt-qt-creator/.**

Zaninotto, Fabien Potencier y François. 2007. **Symfony 1.0, la guía definitiva. *librosweb.es.* [En línea] 2007. [Citado el: 10 de 04 de 2011.] http://www.librosweb.es/symfony/capitulo2/el_patron_mvc.html. 978-1590597866.**

ANEXOS

Diagrama del Proceso Recuperar Material.

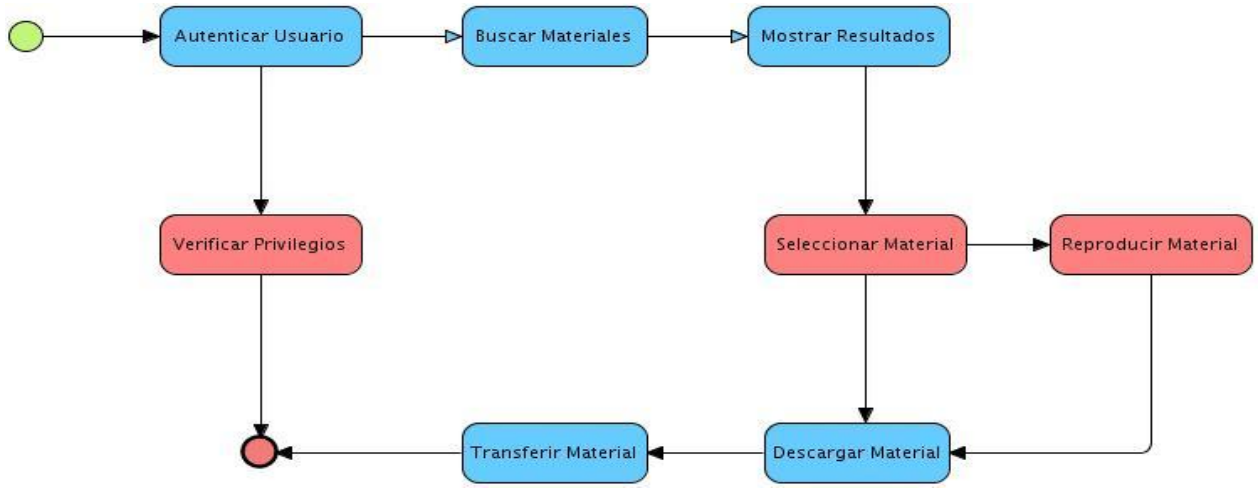

Diagrama del Proceso Catalogar Media.